USB82642/2642 SDIO Over USB User's Guide

Using USB Mass Storage Class Bulk-Only Transport & SCSI Pass Through

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

ISBN: 9781632762481

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
═ ISO/TS 16949 ═

Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

Table of Contents

Preface

NOTICE TO CUSTOMERS

All documentation becomes dated, and this manual is no exception. Microchip tools and documentation are constantly evolving to meet customer needs, so some actual dialogs and/or tool descriptions may differ from those in this document. Please refer to our web site (www.microchip.com) to obtain the latest documentation available.

Documents are identified with a "DS" number. This number is located on the bottom of each page, in front of the page number. The numbering convention for the DS number is "DSXXXXXA", where "XXXXX" is the document number and "A" is the revision level of the document.

For the most up-to-date information on development tools, see the MPLAB® IDE online help. Select the Help menu, and then Topics to open a list of available online help files.

INTRODUCTION

This chapter contains general information that will include:

- Document Layout
- Conventions Used in this Guide
- The Microchip Web Site
- Development Systems Customer Change Notification Service
- Customer Support
- Document Revision History

DOCUMENT LAYOUT

This manual proposes a standard mechanism by which a remote SD Host Controller may be managed via USB Mass Storage Class (MSC) Bulk-Only Transport and SCSI Pass Through. The layout is as follows:

- Summary - Identifies fundamental standards.
- Protocol Overview - Gives overview of protocol.
- SD Over USB Commands - Determines protocol version supported.
- SCSI Sense Codes - Identifies SCSI Sense Code list.
- Sample Code - Provides references for sample code.
- Blank CDB Template - Provides diagram of blank template.
- Proprietary References - Provides reference diagrams.
- USB82464/2642 API Reference Guide for USB SDIO - Provides reference guide.

CONVENTIONS USED IN THIS GUIDE

This manual uses the following documentation conventions:

DOCUMENTATION CONVENTIONS

Description	Represents	Examples
Arial font:		
Italic characters	Referenced books	*MPLAB® IDE User's Guide*
	Emphasized text	...is the *only* compiler...
Initial caps	A window	the Output window
	A dialog	the Settings dialog
	A menu selection	select Enable Programmer
Quotes	A field name in a window or dialog	"Save project before build"
Underlined, italic text with right angle bracket	A menu path	*File>Save*
Bold characters	A dialog button	Click **OK**
	A tab	Click the **Power** tab
N'Rnnnn	A number in verilog format, where N is the total number of digits, R is the radix and n is a digit.	4'b0010, 2'hF1
Text in angle brackets < >	A key on the keyboard	Press <Enter>, <F1>
Courier New font:		
Plain Courier New	Sample source code	`#define START`
	Filenames	`autoexec.bat`
	File paths	`c:\mcc18\h`
	Keywords	`_asm, _endasm, static`
	Command-line options	`-Opa+, -Opa-`
	Bit values	`0, 1`
	Constants	`0xFF, 'A'`
Italic Courier New	A variable argument	*`file`*`.o`, where *`file`* can be any valid filename
Square brackets [ ]	Optional arguments	`mcc18 [options]` *`file`* `[options]`
Curly brackets and pipe character: { | }	Choice of mutually exclusive arguments; an OR selection	`errorlevel {0|1}`
Ellipses...	Replaces repeated text	`var_name [, var_name...]`
	Represents code supplied by user	`void main (void) { ... }`

THE MICROCHIP WEB SITE

Microchip provides online support via our web site at www.microchip.com. This web site is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the web site contains the following information:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQs), technical support requests, online discussion groups, Microchip consultant program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

DEVELOPMENT SYSTEMS CUSTOMER CHANGE NOTIFICATION SERVICE

Microchip's customer notification service helps keep customers current on Microchip products. Subscribers will receive e-mail notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, access the Microchip web site at www.microchip.com, click on Customer Change Notification and follow the registration instructions.

The Development Systems product group categories are:

- **Compilers** – The latest information on Microchip C compilers, assemblers, linkers and other language tools. These include all MPLAB C compilers; all MPLAB assemblers (including MPASM assembler); all MPLAB linkers (including MPLINK object linker); and all MPLAB librarians (including MPLIB object librarian).
- **Emulators** – The latest information on Microchip in-circuit emulators.This includes the MPLAB REAL ICE and MPLAB ICE 2000 in-circuit emulators.
- **In-Circuit Debuggers** – The latest information on the Microchip in-circuit debuggers. This includes MPLAB ICD 3 in-circuit debuggers and PICkit 3 debug express.
- **MPLAB IDE** – The latest information on Microchip MPLAB IDE, the Windows Integrated Development Environment for development systems tools. This list is focused on the MPLAB IDE, MPLAB IDE Project Manager, MPLAB Editor and MPLAB SIM simulator, as well as general editing and debugging features.
- **Programmers** – The latest information on Microchip programmers. These include production programmers such as MPLAB REAL ICE in-circuit emulator, MPLAB ICD 3 in-circuit debugger and MPLAB PM3 device programmers. Also included are nonproduction development programmers such as PICSTART Plus and PIC-kit 2 and 3.

CUSTOMER SUPPORT

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support

Customers should contact their distributor, representative or field application engineer (FAE) for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in the back of this document.

Technical support is available through the web site at:
http://www.microchip.com/support

DOCUMENT REVISION HISTORY

Revision	Section/Figure/Entry	Correction
DS 50002277A (07-18-14)	Document Release	

Chapter 1. Summary

1.1 ABOUT THIS DOCUMENT

This document explains a standard mechanism by which a remote SD Host Controller may be managed via USB Mass Storage Class (MSC) Bulk-Only Transport and SCSI Pass Through.

This document assumes the reader is familiar with the concepts outlined in the documents listed in **Section 1.1.1 "Fundamental Standards"**.

1.1.1 Fundamental Standards

It is the intent of this document to propose standardizable mechanisms compliant with the applicable portions of the standards listed in this section. Where conflicts occur or information is incomplete, the listed standards have precedence.

- USB Mass Storage Class Specification Overview Rev. 1.2
- USB Mass Storage Class Bulk-Only Transport Rev. 1.0
- SCSI Architecture Model - 2 (SAM-2)
- SCSI Primary Commands - 2 (SPC-2)
- SD Specifications Part 1 Physical Layer Specification Version 2.00
- SD Specifications Part E1 SDIO Specification Version 2.00
- SD Specifications Part A2 SD Host Controller Standard Specification Version 2.00

Chapter 2. Protocol Overview

2.1 USB ENUMERATION

A compliant device will enumerate at least one Interface Descriptor as USB BaseClass 0x08 (Mass Storage Device Class), SubClass 0x06 (SCSI transparent command set), Protocol 0x50 (Bulk-Only Transport). See *USB Mass Storage Class Specification Overview Rev 1.2* Tables 2.1 and 3.1.

2.2 USB BULK-ONLY TRANSPORT COMMAND / DATA / STATUS PROTOCOL

The proposed protocol will be encapsulated by Command Block Wrappers (CBWs) as described in *USB Mass Storage Class Bulk-Only Transport Rev 1.0* Section 5. Specifically, the field *CBWCB* depicted in Table 5.1 of the aforementioned document shall contain the SCSI Command Descriptor Blocks (CDBs) described in this document.

Data will be returned according to the specific definition of each CDB.

Execution status of each CDB will be returned in a Command Status Wrapper (CSW) as defined in *USB Mass Storage Class Bulk-Only Transport Rev 1.0* Section 5.2.

If a command returns Command Failed (0x01) in the USB CSW *bCSWStatus* field, additional information is available by issuing a SCSI REQUEST SENSE CDB, see *SCSI Primary Commands - 2 (SPC-2)* Section 7.20.

2.3 COMMAND DESCRIPTOR BLOCK (CDB) NOTES

All CDB and Data fields are either big endian or SD bus ordered. A value's most significant byte should appear in the lowest numbered byte offset of the CDB. Or in the case of string IDs or multi-byte SD registers, the first byte onto or off of the SD bus should appear in the lowest address.

All CDBs defined in this document contain a *SCSI Operation Code*, a *SCSI Service Action*, and a *SCSI Control field*.

The *Operation Code* for all CDBs in this specification is 0x5D.

The *Service Action* is dependant on the function being performed. *Service Action* codes 0x0000 through 0x7FFF are reserved for this specification. *Service Action* codes 0x8000 through 0xFFFF are reserved for vendor specific operations. The most significant byte of *Service Action* codes greater than 0x7FFF is defined as the *Vendor ID*.

The *Control* field is unused and should be specified as 0x00.

In the *Control* field, if the NACA bit is set to one, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

In the *Control* field, if the LINK bit is set to one, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

2.4 DATA TRANSFER LENGTH REQUIREMENTS

All SD commands that utilized the transfer of data on the SD data lines are required to specify the data transfer lengths on a 4 byte alignment. The minimum length for a data transfer is 4 bytes and all other lengths must be aligned on 4 byte boundaries. The SD_DATA_IO (section 3.4), SD COMMAND WITH DATA (section 3.5), and SD COMMAND (section 3.3) must conform this requirement.

Chapter 3. SD Over USB Commands

3.1 SD_INQUIRY

Determines the *Protocol Version* supported, if any.

3.1.1 SD_INQUIRY CDB

Bit / Byte	7	6	5	4	3	2	1	0
0	SCSI Operation Code (0x5D)							
1	SCSI Service Action (0x0000)							
2								
3	Reserved (0)							
4								
5								
6	Allocation Length (>=0x07)							
7	Signature (0x5D5C)							
8								
9	SCSI Control (0)							
	Vendor Specific (0)		Reserved (0)			NACA (0)	Obsolete (0)	LINK (0)

(x) = Only supported value

3.1.2 SD_INQUIRY Data

Bit / Byte	7	6	5	4	3	2	1	0
0	Protocol Version (0x01)							
1	Signature							
2								
3	SD Host Controller Version Major							
4	Reserved (0)	SD Host Controller Version Minor						
5	Protocol Class				Slot			
	SC	AIC	DTOC	HCRC				
6	Vendor ID (0x80 – 0xFF)							
n	Vendor Specific Data							

(x) = Only supported value

Signature is echoed back from the CDB. *SD Host Controller Version* (Major, Minor) is the version of the *SD Specifications Part A2 SD Host Controller Standard Specification* supported. *Slot* is assigned by the Host Controller and is used in conjunction with the *Slot Interrupt Status* Host Controller register.

Protocol Class maps to specific combinations of the following bits: *HCRC* is set to indicate support of *Host Controller Register Commands*. *DTOC* bit is set to indicate support of *Data Transfer Optimization Commands*. *AIC* bit is set to indicate support of *Auto Initialization Commands*. *SC* bit is set to indicate support of *Status Commands*. Bit combinations not listed in the following table are not allowed:

Protocol Class Name	SC	AIC	DTOC	HCRC	Performance	Host Driver Simplicity	Host Driver Reusability	Flexibility
'0'	0	0	0	1	-	-	+	+
'1'	0	0	1	1	+	-	0	+
'2'	0	1	1	1	+	0	0	+
'3'	1	1	1	0	+	+	-	-

Performance relates to the amount of overhead injected into SD Data transfers for a particular *Protocol Class*. *Host Driver Simplicity* relates to the simplicity of the SD Host Controller layers of the host driver. *Host Driver Reusability* relates to the direct reusability of a host driver which conforms to *SD Specifications Part A2 SD Host Controller Standard Specification*. *Flexibility* relates to the characteristic of a particular *Protocol Class* to be able to modify its host driver to support new card types and future SD standards.

Vendor Specific Data will be returned up to a total of *Allocation Length* bytes.

3.2 SD_HC_REGISTER_IO

This *Host Controller Register Command* will read / write SD Host Controller Standard Register data at *Register Offset* according to the number of bytes set in *Allocation Length*. For register information see *SD Specifications Part A2 SD Host Controller Standard Specification Version 2.00* Table 2-1: SD Host Controller Register Map. This command is optional if Auto Initialization (SD_CARD_INITIALIZE) is supported, required if not.

3.2.1 SD_HC_REGISTER_IO CDB

Bit / Byte	7	6	5	4	3	2	1	0
0	SCSI Operation Code (0x5D)							
1	SCSI Service Action (0x0002 or 0x0003)							
2								
3	Reserved (0)							
4								
5								
6	Allocation Length (0x01, 0x02, 0x04 or 0x08)							
7	Reserved (0)							
8	Register Offset							
9	SCSI Control (0)							
	Vendor Specific (0)		Reserved (0)			NACA (0)	Obsolete (0)	LINK (0)

(x) = Only supported value

Service Action 0x0002 indicates data will be read, *Service Action* 0x0003 indicates a data write. To allow for simplified emulation of this register set, several rules must be followed. Operations must always occur on register aligned boundaries. Operations may not span multiple registers (except in the case of adjacent registers with sequentially numbered names i.e. *Response0* – *Response7*, adjacent registers with names suffixed by (High) and (Low) i.e. – *Argument 1 (Low)* and *Argument 1 (High)*, and the register block including offsets 0x00 – 0x0F). *Capabilities* Register (0x40) locations 19, 20 and 22 will always read 0 (SDMA, ADMA1 and ADMA2 are not supported).

3.2.2 SD_HC_REGISTER_IO Data

SD Host Controller Standard Register data will be read / written at *Register Offset* according to the number of bytes set in *Allocation Length*.

3.3 SD_COMMAND

This *Data Transfer Optimization Command* is an optimized method to generate an SD Command and return the SD Response.

3.3.1 SD_COMMAND CDB

Bit / Byte	7	6	5	4	3	2	1	0
0	SCSI Operation Code (0x5D)							
1	SCSI Service Action (0x0004)							
2								
3	Block Size [0x04 – 0x05]							
4								
5	Block Count [0x06 – 0x07]							
6								
7	SD Part A2 v2.00 Argument 0 & 1 / SD Part A2 v3.00 Argument 1 [0x08 – 0x0B]							
8								
9								
10								
11	Transfer Mode [0x0C – 0x0D]							
12								
13	Command [0x0E – 0x0F]							
14								
15	SCSI Control (0)							
	Vendor Specific (0)		Reserved (0)			NACA (0)	Obsolete (0)	LINK (0)

(x) = Only supported value

[x] = SD Host Standard Register offset locations *SD Specifications Part A2 SD Host Controller Standard Specification Version 2.00* Table 2-1: SD Host Controller Register Map

This command will write the supplied registers to the SD Host Controller Register set and execute the state machine depicted in Figures 3-10, 3-11 and steps 1 – 8 of Figure 3-12 of *SD Specifications Part A2 SD Host Controller Standard Specification Version 2.00*. Finally, *Response0* – *Response7* will be returned. *SDIO Interrupt* indicates that an SDIO interrupt is pending.

Note: The number of bytes defined for the data transfer in the combination of block size and block count must be specified on a four byte alignment.

3.3.2 SD_COMMAND Data

Bit Byte	7	6	5	4	3	2	1	0
0	Response0 [0x10 – 0x11]							
1								
2	Response1 [0x12 – 0x13]							
3								
4	Response2 [0x14 – 0x15]							
5								
6	Response3 [0x16 – 0x17]							
7								
8	Response4 [0x18 – 0x19]							
9								
10	Response5 [0x1A – 0x1B]							
11								
12	Response6 [0x1C – 0x1D]							
13								
14	Response7 [0x1E – 0x1F]							
15								
16	Reserved(0)							SDIO Interrupt

[x] = SD Host Standard Register offset locations SD Specifications Part A2 SD Host Controller Standard Specification Version 2.00 Table 2-1: SD Host Controller Register Map

3.4 SD_DATA_IO

This *Data Transfer Optimization Command* is an optimized method to read or write SD Data.

3.4.1 SD_DATA_IO CDB

Bit / Byte	7	6	5	4	3	2	1	0
0	SCSI Operation Code (0x5D)							
1	SCSI Service Action (0x0006, 0x0007)							
2								
3	Allocation Length							
4								
5								
6								
7	Reserved (0)							
8								
9	SCSI Control (0)							
	Vendor Specific (0)		Reserved (0)			NACA (0)	Obsolete (0)	LINK (0)

(x) = Only supported value

Service Action 0x0006 indicates data will be read, *Service Action* 0x0007 indicates a data write. This command will initiate the execution of the state machine depicted in Figure 3-12: Transaction Control with Data Transfer Using DAT Line Sequence (Not using DMA) of *SD Specifications Part A2 SD Host Controller Standard Specification Version 2.00* beginning at step 9. Data associated with this command corresponds to data written / read in steps 12 and 16 respectively. Steps 18 through 20 are executed if *Block Count Enable* is set in *Transfer Mode* and when *Block Count* transitions to 0. This command may be called multiple times to send / receive all of the data in an SD Transfer, though *Allocation Length* must be a multiple of *Block Size* until the last call to this command (final block). The aggregate value of *Allocation Length* in all reads or writes for a particular SD_COMMAND sequence should not exceed the initial values of *Block Size* * *Block Count*.

3.4.2 SD_DATA_IO Data

Data will be accepted / returned according to the number of bytes set in *Allocation Length*. Streaming data may be read / written by issuing this command multiple times after setting up an SD Data read / write operation with SD_COMMAND or by manually setting up an SD Data operation using multiple SD_HC_REGISTER_IO operations.

Note: Allocation length must be specified on a four byte alignment.

3.5 SD_COMMAND_WITH_DATA

This Data Transfer Optimization Command is an optimized method to generate an SD Command that has associated data, and simultaneously transfer that data in one transaction.

3.5.1 SD_COMMAND_WITH_DATA CDB

Bit / Byte	7	6	5	4	3	2	1	0
0	SCSI Operation Code (0x5D)							
1	SCSI Service Action (0x0005)							
2								
3	Block Size [0x04 – 0x05]							
4								
5	Block Count [0x06 – 0x07]							
6								
7	SD Part A2 v2.00 Argument 0 & 1 / SD Part A2 v3.00 Argument 1 [0x08 – 0x0B]							
8								
9								
10								
11	Transfer Mode [0x0C – 0x0D]							
12								
13	Command [0x0E – 0x0F]							
14								
15	SCSI Control (0)							
	Vendor Specific (0)		Reserved (0)			NACA (0)	Obsolete (0)	LINK (0)

(x) = Only supported value

[x] = SD Host Standard Register offset locations *SD Specifications Part A2 SD Host Controller Standard Specification Version 2.00* Table 2-1: SD Host Controller Register Map

This command will write the supplied registers to the SD Host Controller Register set and execute the state machine depicted in Figures 3-10, 3-11 and steps 1 – 8 of Figure 3-12 of *SD Specifications Part A2 SD Host Controller Standard Specification Version 2.00*. Finally, *Response0 – Response7* will be returned.

3.5.2 SD_COMMAND_WITH_DATA Data

Data will be accepted / returned according to the number of bytes in *Block Size* and *Block Count* and according to the settings in *Transfer Mode*.

> **Note:** The number of bytes defined for the data transfer in the combination of block size and block count must be specified on a four byte alignment.

3.6 SD_CARD_INITIALIZE

This Auto Initialization Command initializes an attached SD card.

3.6.1 SD_CARD_INITIALIZE CDB

Bit / Byte	7	6	5	4	3	2	1	0
0	SCSI Operation Code (0x5D)							
1	SCSI Service Action (0x0008)							
2								
3	Reserved (0)							
4								
5								
6	Allocation Length (0x1C)							
7	Reserved (0)							
8	Init Type (0 = Full Initialization)							
	Status Only	Reserved (0)	SDIO Interrupt	Bus Width	Bus Clock	Basic Init	Power On	Power Off
9	SCSI Control (0)							
	Vendor Specific (0)		Reserved (0)			NACA (0)	Obsolete (0)	LINK (0)

(x) = Only supported value

This command executes the card Auto Initialization sequence as depicted in Figure 3-9: Card Initialization and Identification of SD Specifications Part A2 SD Host Controller Standard Specification. The version of the Host Controller Standard Specification followed should be the same as indicated in SD_INQUIRY. Additionally, SET_DSR (CMD4) and any other physical layer initialization should be executed as necessary. The card should be left in Stand-by State (stby) as depicted in Figure 4-3 of SD Specifications Part 1 Physical Layer Specification Version 2.00. Init Type allows the selective execution of portions of the initialization sequence. An Init Type value of 0 causes a complete initialization sequence to be performed. If Power Off and Power On are both specified (or a complete initialization is performed), a minimum of 1.25 seconds of delay will elapse between the two operations to allow the card power rail to reach 0V. Bus Width will cause the card and controller to be configured for the maximum bus width commonly supported. Bus Clock will cause the card and controller to be configured for the highest bus clock speed commonly supported. SDIO Interrupt will enable the card controller's SDIO interrupt functionality.

3.6.2 SD_CARD_INITIALIZE Data

Bit / Byte	7	6	5	4	3	2	1	0
0	Card Present	Card Error	Card Locked	SDIO	SD Memory Function Type			
1	Reserved (0)		Bus Clock Speed (MHz)					
2	Card Power	SDIO Functions			Reserved (0)	Bus Width		
3	RCA							
4								
5	CID							
…								
20								
21	Memory OCR							
22								
23								
24								
25	SDIO OCR							
26								
27								

(x) = Only supported value

If a card is present, *Card Present* will be set. If a card cannot be transitioned into Standby state, *Card Error* should be set. If the card is detected as write protected during initialization, *Card Locked* will be set.

SD Memory Function Type is defined as follows:

SD Memory Function Type	Description
0	None
1	SDSC v1.01 or v1.10
2	SDSC v2.00 or v3.00
3	SDHC or SDXC

CID and *Memory OCR* are valid only when *SD Memory Function Type* is not equal to *None*, otherwise they read 0. *SDIO Functions* and *SDIO OCR* are valid only when the SDIO bit is set.

3.7 SD_STATUS

This Status Command returns SD status information.

3.7.1 SD_STATUS CDB

Bit / Byte	7	6	5	4	3	2	1	0
0	SCSI Operation Code (0x5D)							
1	SCSI Service Action (0x000A)							
2								
3	Reserved (0)							
4								
5								
6	Allocation Length (0x02)							
7	Reserved (0)							
8	Reserved (0)							Read Int Pending
9	SCSI Control (0)							
	Vendor Specific (0)		Reserved (0)			NACA (0)	Obsolete (0)	LINK (0)

(x) = Only supported value

3.7.2 SD_STATUS Data

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved (0)						SDIO Interrupt	Card Present
1	SDIO Int Pending							

(x) = Only supported value

If a card is present, *Card Present* will be set. *SDIO Interrupt* will be set if an SDIO Interrupt has been detected. *SDIO Int Pending* contains the contents of the SDIO register of the same name if *Read Int Pending* was set in the CDB.

Chapter 4. SCSI Sense Codes

See *SCSI Primary Commands - 2 (SPC-2)* Section 7.20.

Sense Key (SK)
Additional Sense Code (ASC)
Additional Sense Code Qualifier (ASCQ)

SK/ASC/ASCQ
00h/00h/00h

TODO: Define error codes for specific CDBs.

SENSE_NO_MEDIA
02h/3ah/00h

SENSE_READ_ERROR
03h/11h/00h

SENSE_WRITE_ERROR
03h/03h/00h

SENSE_NOT_READY
02h/04h/ffh

SENSE_ILLEGAL_REQUEST
05h/26h/00h

SENSE_MEDIA_CHANGE
06h/28h/00h

SENSE_FMT_FAILED
03h/31h/01h

Chapter 5. Sample Code

5.1 REFERENCES

DeviceIoControl Function

http://msdn.microsoft.com/en-us/library/aa363216(VS.85).aspx

SCSI Pass Through Interface

http://msdn.microsoft.com/en-us/library/dd163410.aspx

IOCTL_SCSI_PASS_THROUGH

http://msdn.microsoft.com/en-us/library/ms803657.aspx

http://msdn.microsoft.com/en-us/library/ms810309.aspx

IOCTL_SCSI_PASS_THROUGH_DIRECT

http://msdn.microsoft.com/en-us/library/ms803668.aspx

http://msdn.microsoft.com/en-us/library/ms810301.aspx

Chapter 6. Blank CDB Template

Bit Byte	7	6	5	4	3	2	1	0
0	SCSI Operation Code (0x5D)							
1	SCSI Service Action (0x00XX)							
2								
3								
4								
5								
6								
7								
8								
9								
10								
11								
12								
13								
14								
15	SCSI Control (0)							
	Vendor Specific (0)		Reserved (0)			NACA (0)	Obsolete (0)	LINK (0)

(x) = Only supported value

Chapter 7. Proprietary References

7.1 USB MASS STORAGE CLASS BULK-ONLY TRANSPORT REV 1.0

FIGURE 7-1: COMMAND BLOCK WRAPPER

Byte \ bit	7	6	5	4	3	2	1	0
0-3	*dCBWSignature*							
4-7	*dCBWTag*							
8-11 (08h-0Bh)	*dCBWDataTransferLength*							
12 (0Ch)	*bmCBWFlags*							
13 (0Dh)	Reserved (0)				*bCBWLUN*			
14 (0Eh)	Reserved (0)				*bCBWCBLength*			
15-30 (0Fh-1Eh)	*CBWCB*							

FIGURE 7-2: COMMAND STATUS WRAPPER

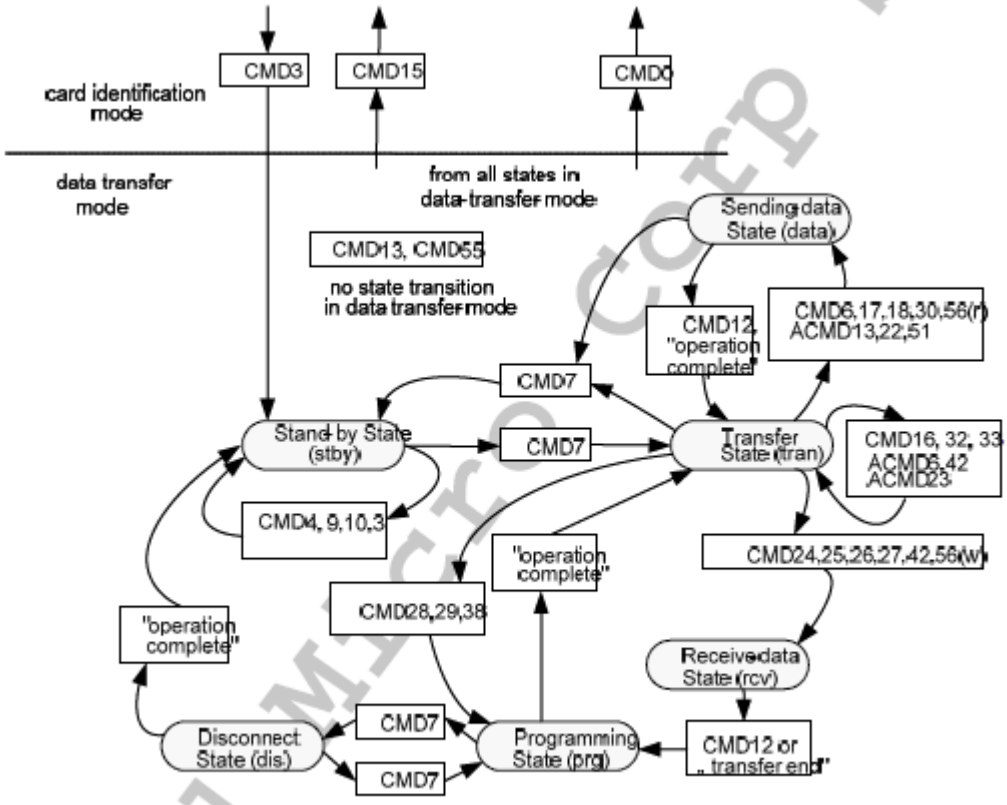
Byte \ bit	7	6	5	4	3	2	1	0
0-3	*dCBWSignature*							
4-7	*dCBWTag*							
8-11 (08h-0Bh)	*dCBWDataTransferLength*							
12 (0Ch)	*bmCBWFlags*							
13 (0Dh)	Reserved (0)				*bCBWLUN*			
14 (0Eh)	Reserved (0)				*bCBWCBLength*			
15-30 (0Fh-1Eh)	*CBWCB*							

FIGURE 7-3: COMMAND BLOCK STATUS VALUES

Value	Description
00h	Command Passed ("good status")
01h	Command Failed
02h	Phase Error
03h and 04h	Reserved (Obsolete)
05h to FFh	Reserved

7.2 SD SPECIFICATIONS PART 1 PHYSICAL LAYER SPECIFICATION VERSION 2.0

FIGURE 7-1: SD MEMORY CARD STATE DIAGRAM (DATA TRANSFER MODE)



7.3 SD SPECIFICATIONS PART A2 SD HOST CONTROLLER STANDARD SPECIFICATION VERSION 2.0

FIGURE 7-4: SD HOST CONTROLLER REGISTER MAP

Offset	15-08 bit	07-00 bit	Offset	15-08 bit	07-00 bit
002h	SDMA System Address (High)		000h	SDMA System Address (Low)	
006h	Block Count		004h	Block Size	
00Ah	Argument1		008h	Argument0	
00Eh	Command		00Ch	Transfer Mode	
012h	Response1		010h	Response0	
016h	Response3		014h	Response2	
01Ah	Response5		018h	Response4	
01Eh	Response7		01Ch	Response6	
022h	Buffer Data Port1		020h	Buffer Data Port0	
026h	Present State		024h	Present State	
02Ah	Wakeup Control	Block Gap Control	028h	Power Control	Host Control
02Eh	Software Reset	Timeout Control	02Ch	Clock Control	
032h	Error Interrupt Status		030h	Normal Interrupt Status	
036h	Error Interrupt Status Enable		034h	Normal Interrupt Status Enable	
03Ah	Error Interrupt Signal Enable		038h	Normal Interrupt Signal Enable	
03Eh	---		03Ch	Auto CMD12 Error Status	
042h	Capabilities		040h	Capabilities	
046h	Capabilities (Reserved)		044h	Capabilities (Reserved)	
04Ah	Maximum Current Capabilities		048h	Maximum Current Capabilities	
04Eh	Maximum Current Capabilities (Reserved)		04Ch	Maximum Current Capabilities (Reserved)	
052h	Force Event for Error Interrupt Status		050h	Force Event for Auto CMD12 Error Status	
	---		054h	---	ADMA Error Status
05Ah	ADMA System Address [31:16]		058h	ADMA System Address [15:00]	
05Eh	ADMA System Address [63:48]		05Ch	ADMA System Address [47:32]	
---	---		---	---	
0F2h	---		0F0h	---	
---	---		---	---	
0FEh	Host Controller Version		0FCh	Slot Interrupt Status	

FIGURE 7-5: SD COMMAND ISSUE SEQUENCE

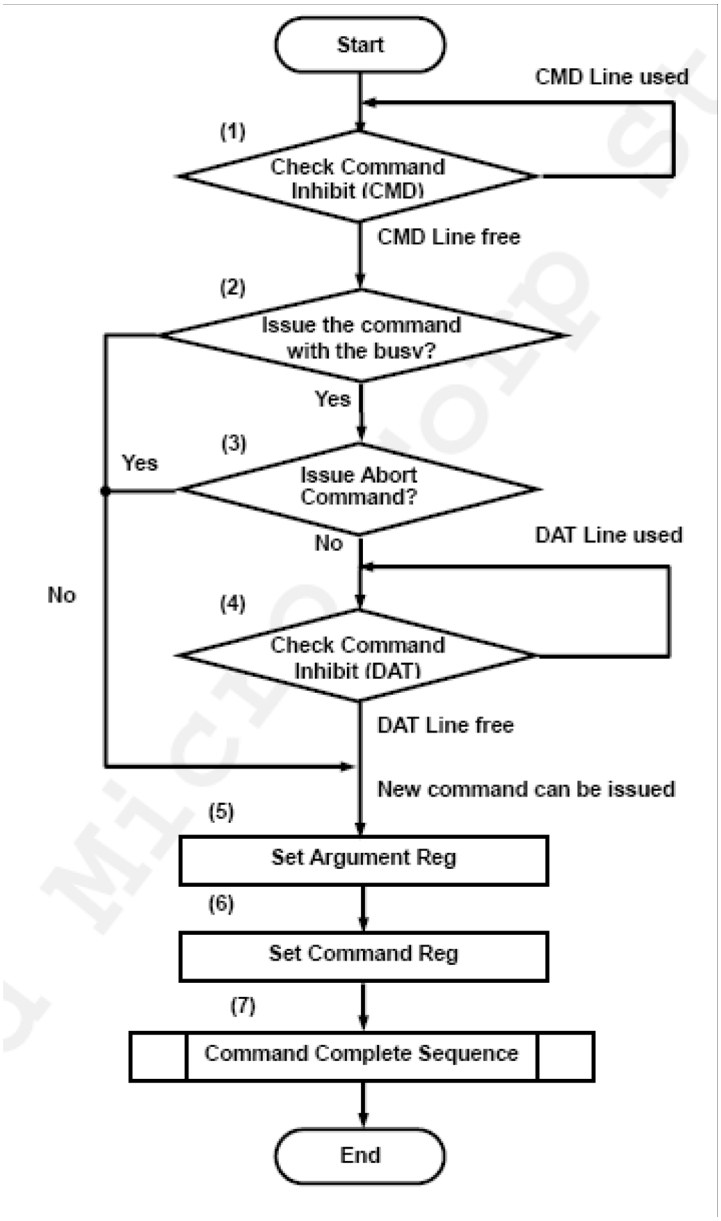


FIGURE 7-6: COMMAND COMPLETE SEQUENCE

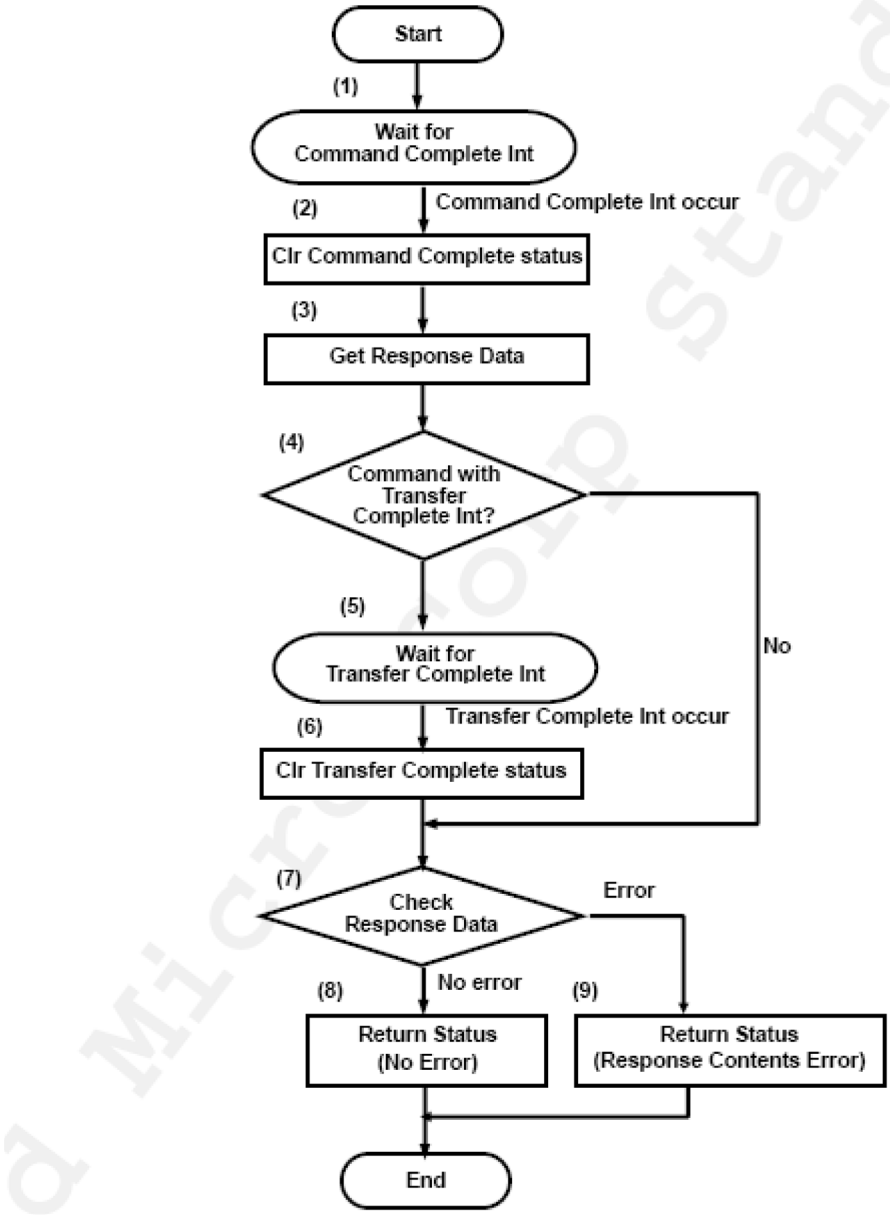
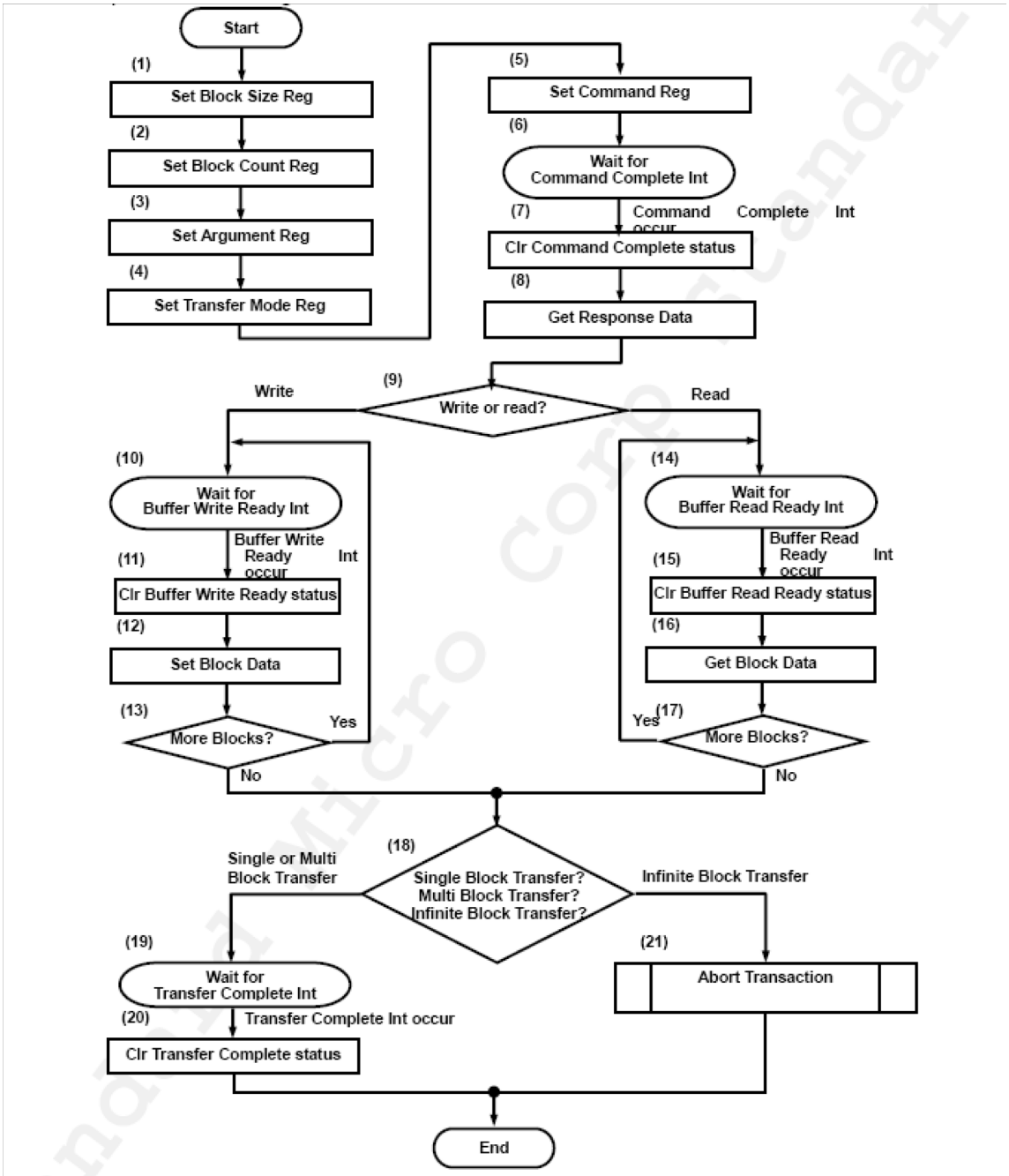


FIGURE 7-7: TRANSACTION CONTROL WITH DATA TRANSFER USING DAT LINE SEQUENCE (NOT USING DMA)



Chapter 8. USB82464/2642 API Reference Guide for USB SDIO

8.1 API'S SUPPORTING SDIO (USB_DEV_SDIO.DLL)

8.1.1 MchpUsbUSB2642_Open

Prototype:

MCHP_USB_ERROR MchpUsbUSB2642_Open(int VID, (*int PID,HANDLE *hFile*)

Description:

Provide Handle for USB mass storage class devices.

Parameters:

	Parameter	Description
	int VID	Device Vendor ID (0x0424)
	int PID	Product ID(0x4041)
	*HANDLE *hFile*	

Returns:

MCHP_USB_ERROR.

Sample:

```
UINT VID;
    UINT PID;
    UINT DID = 0;

    wprintf(L"Enter VID of the device:");
    wscanf (L"%x",&VID);
    wprintf(L"Enter PID of the device:");
    wscanf (L"%x",&PID);
    wprintf(L"Enter DeviceID if multiple devices are present(0,1,2...):");
    wscanf (L"%d",&DID);
    bRet =USB2642_Open_Demo(VID,PID,DID, &hFile );
    if(hFile == INVALID_HANDLE_VALUE)
    {
        return MCHP_Error_Invalid_Device_Handle;
    }
```

8.1.2 MchpUsbSDCardInitialize

Prototype:

MCHP_USB_ERROR MchpUsbSDCardInitialize(HANDLE hFile, UINT8 nInitType, SD_CARD_INITIALIZE_DATA &scidInitializationData, UINT8 initDelay)

Devices Supported:

USB82642 family

Description:

Issue SD Card Initialize to query current card setting such as RCA, OCR, memory type etc.Init Type allows the selective execution of portions of the initialization sequence. An Init Typevalue of 0 causes a complete initialization sequence to be performed.

This function access the value of the card interface registers: OCR, CID,RCA.

The values are present in the SD_CARD_INITIALIZE_DATA structure:

```
UCHAR RCA[2];
       UCHAR CID[16];
       UCHAR MemoryOCR[4];
```

Parameters:

	Parameter	Description
	Handle	A valid handle of the device
	InitType	Allows the selective execution of portions of the initialization
	scidInitializationData	Refer section 2
	initDelay	Delay for initialization

Returns:

MCHP_USB_ERROR

Sample:

```
SD_CARD_INITIALIZE_DATA scidInitializationData;
UINT8 initDelay2 = 20;

bRet = USB2642_SDCardInitialize_Demo(hFile, SD_OVER_USB_INIT_STATUS_ONLY,
scidInitializationData,initDelay2);
```

8.1.3 MchpUsbSdUsbIssueCmd

Prototype:

MCHP_USB_ERROR MchpUsbSdUsbIssueCmd*(HANDLE hFile, const SDHC_REG* reg, SD_COMMAND_TRANSFER_TYPE transfer, WORD service_action, BYTE buffer[], DWORD buffer_length)*

Devices Supported:

USB82642 family

Description:

This Data Transfer Optimization Command is an optimized method to generate an SD Command and return the SD Response.This command will write the supplied registers to the SD Host Controller Register set. Response0- Response7 will be returned.

Parameters:

	Parameter	Description
	Handle	A valid handle of the device
	Reg	SDHC register
	Transfer	SD command transfer type
	Service action	SCSI pass through service action
	Buffer	array of data
	Buffer_length	size of the data buffer

Returns:

MCHP_USB_ERROR

8.1.4 MchpUsbSDInquiry

Prototype:

MCHP_USB_ERROR MchpUsbSDInquiry*(HANDLE hFile, SD_INQUIRY_DATA &souidData)*

Description:

Determines the Protocol Version supported, if any Signature is echoed back. SD Host Controller Version (Major, Minor) is the version of the SD Specifications Protocol Class.

Parameters:

	Parameter	Description
	Handle	A valid handle of the device
	souidData	structure containing the inquiry data values. Refer section 2

Returns:

MCHP_USB_ERROR.

Sample:

```
    SD_INQUIRY_DATA souidData;
    MCHP_USB_ERROR bRet;
bRet = USB2642_SDInquiry_Demo(hFile, souidData);
wprintf(L" Executing SDInquiry\n");
if(bRet != MCHP_Error_Success)
{
    CloseHandle(hFile);
    return -1;
}
```

8.1.5 MchpUsbSendSDUsbCommnd

Prototype:

MCHP_USB_ERROR MchpUsbSendSDUsbCommnd

HANDLE hFile,

BYTE command,

DWORD argument,

SD_RESPONSE_TYPE response_type,

BYTE response[],

SD_COMMAND_TRANSFER_TYPE transfer,

DWORD block_size,

DWORD block_num,

BYTE block_data[])

Description:

This Data Transfer Command is a method to generate an SD Command and return the SD Response.

Parameters:

	Parameter	Description
	Handle	A valid handle of the device
	Command	The CMD to the card
	Argument	The arguments of the command
	Response type	type of the response(R0-R7).refer sec 2
	Response	buffer containing the response data
	Transfer	Read or write transfer. Refer sec 2
	Block size	size or length of the data to be read or written
	Blocknum	number of blocks to be read
	*Block*_data	array contain in the data to be written or read data

Returns:

MCHP_USB_ERROR.

Sample:

```
//Issue SD over USB CMD8 to retrieve Extended CSD registers
   bRet = USB2642_SendSDUsbCommnd_Demo(hFile, K_SD_CMD_8, 0, Response1, response,
SD_TRANSFER_READ, 512, 1, ext_csd);
```

8.1.6 MchpUsbSDHostControllerRegisterIO

Prototype:

MCHP_USB_ERROR MchpUsbSDHostControllerRegisterIO*(HANDLE hFile, BYTE nRegisterOffset, void *pRegisterData, BYTE nRegisterLength, BOOL bWrite)*

Description:

This Host Controller Register Command will read / write SD Host Controller Standard

Register data at Register Offset according to the number of bytes set in Allocation-Length.

SD Host Controller Standard Register data will be read / written at Register Offset according to the number of bytes set in Allocation Length. Service Action0x0002 indicates data will be read, Service Action0x0003 indicates a data write. To allow for simplified emulation of this register set, several rules must be followed. Operations must always occur on register aligned boundaries.

Parameters:

	Parameter	Description
	hFile	A valid handle to the device
	nRegisterOffset	the host controller register offset
	pRegisterData	data of the register
	nRegisterLength	length of the register
	bWrite	set to 1 if writing to register

Returns:

MCHP_USB_ERROR.

8.1.7 MchpUsbSDGPIO1_Get

Prototype:

MCHP_USB_ERROR MchpUsbSD_GPIO1_Get(HANDLE hFile, BOOL *pinState)

Description:

Get the GPIO1Status

Parameters:

	Parameter	Description
	hFile	A valid handle of the device
	Bool pinState	

Returns:

MCHP_USB_ERROR.

8.1.8 MchpUsbSD_GPIO1_Set

Prototype:

MCHP_USB_ERROR MchpUsbSD_GPIO1_Set (HANDLE hFile, BOOL pinState)

Description:

Set the GPIO1

Parameters:

	Parameter	Description
	hFile	A valid handle of the device
	Bool pinState	

Returns:

MCHP_USB_ERROR.

8.1.9 MchpUsbSdGetCSD

Prototype:

MCHP_USB_ERROR MchpUsbSdGetCSD (HANDLE hFile,BYTE *value)

Description:

Sends the Approproiate CMD as per eMMC spec to retrieve the CSD[16] register

Parameters:

	Parameter	Description
	hFile	A valid handle of the device
	value	Pointer to response data

8.1.10 MchpUsbSdGetCID

Prototype:

MCHP_USB_ERROR MchpUsbSdGetCSD (HANDLE hFile,BYTE *value)

Description:

Sends a vendor specific CDB to retrieve the CSD[16] register

Parameters:

	Parameter	Description
	hFile	A valid handle of the device
	value	Pointer to response data

8.1.11 MchpUsbSdGetSCR

Prototype:

MCHP_USB_ERROR MchpUsbSdGetSCR (HANDLE hFile,BYTE *value)

Description:

Sends a vendor specific CDB to retrieve the SCR[8] register

Parameters:

	Parameter	Description
	hFile	A valid handle of the device
	value	Pointer to response data

8.2 STRUCTURE DEFINITIONS

8.2.1 scidInitializationData

```
typedef struct _SD_CARD_INITIALIZE_DATA {
    UCHAR SDMemoryFunctionType:4;
    UCHAR SDIO:1;
    UCHAR CardLocked:1;
    UCHAR CardError:1;
    UCHAR CardPresent:1;
  UCHAR BusClockSpeed:6;
  UCHAR Reserved3:2;
  UCHAR BusWidth:3;
  UCHAR Reserved2:1;
  UCHAR SDIOFunctions:3;
  UCHAR CardPower:1;
  UCHAR RCA[2];
    UCHAR CID[16];
    UCHAR MemoryOCR[4];
    UCHAR SDIOOCR[3];
} SD_CARD_INITIALIZE_DATA;
```

8.2.2 souidData

```
typedef struct _SD_INQUIRY_DATA {
    UCHAR ProtocolVersion;
    UCHARSignature[2];
    UCHARSDHostControllerVersionMajor;
    UCHARSDHostControllerVersionMinor:7;
  UCHARReserved:1;
    UCHARSlot:4;
    UCHARProtocolClass:4;
    UCHAR VendorID;
    UCHAR VendorData[1];
} SD_INQUIRY_DATA;
```

8.2.3 response_type

```
typedef enum
{
    NoResponse,
    Response1,
    Response1b,
    Response2,
    Response3,
    Response6,
    Response7
} SD_RESPONSE_TYPE;
```

8.2.4 transfer

```
typedef enum
{
   SD_TRANSFER_READ,
   SD_TRANSFER_WRITE,
   SD_TRANSFER_COMMAND
} SD_COMMAND_TRANSFER_TYPE;
```

8.2.5 Error Types

```
typedef enum tagMCHP_USB_ERROR
{
   MCHP_Error_OnExit= 0x0000,/*!< Operation Success */
   MCHP_Error_Success= 0x0001,
   MCHP_Error_Device_Not_Found= 0x0002,/*!< The specific device was not found */
   MCHP_Error_Invalid_Argument= 0x0003,/*!< Argument passed to the API is invalid */
   MCHP_Error_Invalid_Device_Handle= 0x0004,/*!< Device handle passed to the API is
not valid */
   MCHP_Error_WinUSBAPI_Fail= 0x0005,/*!< API of the winusb library failed */
   MCHP_Error_ApiNotSupported= 0x0006,/*!< This particular API is not supported for
this hub family */

   MCHP_SDUsb_LargeBlockNum= 0x1000,/*!< Send SDUSBcommand failed due to large block
num */
   MCHP_SDUsb_LargeBlockSize= 0x1001,/*!< Send SDUSBcommand failed due to large block
size */
   MCHP_SDUsb_InvalidResponse= 0x1002,/*!< Send SDUSBcommand fails with Invalid
Response Type */
   MCHP_SDUsb_InquiryError= 0x1003,

   MCHP_Error_Undefined= 0xFFFF/*!< Unknown error occurred */
} MCHP_USB_ERROR;
```

8.3 SUPPORTED COMMAND TYPES

Here is a list of the command types supported in FW as defined in the code:

```
/*## attribute K_SD_CMD_0 */
#define K_SD_CMD_0 (0x00)

/*## attribute K_SD_CMD_1 */
#define K_SD_CMD_1 (1)

/*## attribute K_SD_CMD_12 */
#define K_SD_CMD_12 (12)

/*## attribute K_SD_CMD_13 */
#define K_SD_CMD_13 (13)

/*## attribute K_SD_CMD_14 */
#define K_SD_CMD_14 (14)

/*## attribute K_SD_CMD_16 */
#define K_SD_CMD_16 (16)

/*## attribute K_SD_CMD_17 */
/* READ_SINGLE_BLOCK */
#define K_SD_CMD_17 (17)

/*## attribute K_SD_CMD_24 */
/* WRITE_BLOCK */
#define K_SD_CMD_24 (24)

/*## attribute K_SD_CMD_19 */
#define K_SD_CMD_19 (19)

/*## attribute K_SD_CMD_2 */
#define K_SD_CMD_2 (2)

/*## attribute K_SD_CMD_3 */
#define K_SD_CMD_3 (3)

/*## attribute K_SD_CMD_55 */
#define K_SD_CMD_55 (55)

/*## attribute K_SD_CMD_6 */
#define K_SD_CMD_6 (0x06)

/*## attribute K_SD_CMD_7 */
#define K_SD_CMD_7 (7)

/*## attribute K_SD_CMD_8 */
#define K_SD_CMD_8 (8)

/**
 * SEND_CSD
 */
/*## attribute K_SD_CMD_9 */
#define K_SD_CMD_9 (9)

/*## attribute K_SD_CMD_15 */
#define K_SD_CMD_15 (15)
```

```
/**
 *   SDIO: IO_RW_DIRECT
 */
/*## attribute K_SD_CMD_52 */
#define K_SD_CMD_52 52

/**
 *   SDIO: IO_RW_EXTEND Optional
 */
/*## attribute K_SD_CMD_53 */
#define K_SD_CMD_53 (53)
```

8.4 CARD REGISTER ACCESS

8.4.1 EXT CSD Register Access

- Access through the SD over USB command SD COMMAND WITH DATA to issue command and read the data. Alternatively the host can issue an SD COMMAND with DATA IO command to read the data.
- Accessible after card has been initialized by the firmware. Card will be in transfer state.
- Read Access through CMD8
- Write Access through CMD6

8.4.2 CSD Register

- Can access through GET SD CSD vendor commands provided in SMSC Vendor SCSI Command for SD and MMC document.
- Read accessible after card has been initialized by the firmware and is cached to read access.

8.4.3 OCR Register Access

- Access through the SD over USB command CARD INITIALIZE with status only.
- Accessible after card has been initialized by the firmware.
- Only available version 2.09 of the firmware. Will currently only be available if executing firmware from external SPI Flash until tape out of new release is completed.

8.4.4 RCA Register Access

- Access through the SD over USB command CARD INITIALIZE with status only.
- Accessible after card has been initialized by the firmware.
- Only available version 2.09 of the firmware. Will currently only be available if executing firmware from external SPI Flash until tape out of new release is completed.
- Write access is currently not provided since RCA is setup early in the initialization process and can only be written when card is in CARD IDENTIFICATION state.

8.4.5 CID Register Access

- Cached by the firmware. Can access through GET SD CID vendor command provided in SMSC Vendor SCSI Command for SD and MMC document.
- Accessible after card has been initialized by the firmware.

8.4.6 DSA Register Access

- DSA Register - This register is currently not accessible.

8.5 SAMPLE CODE

```
SD_CARD_INITIALIZE_DATA scidInitializationData;

int _tmain(int argc, _TCHAR* argv[])
{
    wprintf(L"                      Microchip Technology, Inc.\n");
    wprintf(L"            Sample Application using usb_dev_sdio dll\n");
    wprintf(L"        Copyright (c) 2013 Microchip . All rights reserved\n\n");

    HMODULE LoadLib= LoadLibrary (_T("usb_dev_sdio.dll"));

     if (LoadLib != NULL)
     {
        printf("Loadlib library Loaded\n");
    }


    USB2642_Open_Demo       = (pfnUSB2642_Open)GetProcAddress( LoadLib
,"MchpUsbUSB2642_Open");
    USB2642_SDCardInitialize_Demo= (pfnSDCardInitialize)GetProcAddress( LoadLib
,"MchpUsbSDCardInitialize" );
    USB2642_SCSIPassThroughDirect_Demo         = (pfnSCSIPassThroughDirect
)GetProcAddress( LoadLib ,"MchpUsbSCSIPassThroughDirect" );
    USB2642_SdUsbIssueCmd_Demo              = (pfnSdUsbIssueCmd )GetProcAddress(
LoadLib ,"MchpUsbSdUsbIssueCmd" );
    USB2642_SendSDUsbCommnd_Demo                = (pfnSendSDUsbCommnd )GetProcAddress(
LoadLib ,"MchpUsbSendSDUsbCommnd" );
    USB2642_SDInquiry_Demo                     = (pfnSDInquiry )GetProcAddress( LoadLib
,"MchpUsbSDInquiry" );
    USB2642_SDStatus_Demo                       = (pfnSDStatus)GetProcAddress( LoadLib
,"MchpUsbSDStatus" );
    USB2642_SDHostControllerRegisterIO_Demo     =
(pfnSDHostControllerRegisterIO)GetProcAddress( LoadLib
,"MchpUsbSDHostControllerRegisterIO" );
    USB2642_SDHostDataIo_Demo                = (pfnSDHostDataIo )GetProcAddress(
LoadLib ,"MchpUsbSDHostDataIo" );
    libSDHostCmd               = (pfnSDHostCmd  )GetProcAddress( LoadLib
,"MchpUsbSDHostCmd" );

    USB2642_SDGetCID_Demo= (pfnSDGetCID  )GetProcAddress( LoadLib ,"MchpUsbSDGetCID" );
    USB2642_SDGetCSD_Demo= (pfnSDGetCSD  )GetProcAddress( LoadLib ,"MchpUsbSDGet_CSD"
);

    USB2642_SDGetSCR_Demo= (pfnSDGetSCR  )GetProcAddress( LoadLib ,"MchpUsbSDGetSCR" );
    /* ------------------ GPIO ------------------------*/

    USB2642_SD_GPIO1_Get_Demo= (pfnSD_GPIO1_Get)GetProcAddress( LoadLib,
"MchpUsbSD_GPIO1_Get");
    USB2642_SD_GPIO1_Set_Demo= (pfnSD_GPIO1_Set)GetProcAddress( LoadLib,
"MchpUsbSD_GPIO1_Set");
    USB2642_SD_GPIO1_Init_Demo= (pfnSD_GPIO1_Init)GetProcAddress( LoadLib,
"MchpUsbSD_GPIO1_Init");

    /* ------------------ I2C -------------------------*/

    USB2642_I2COpen_Demo= (pfnUSB2642_I2COpen)GetProcAddress( LoadLib
,"MchpusbI2Copen");
    USB2642_I2CSetClk_Demo= (pfnUSB2642_I2CSetClk)GetProcAddress( LoadLib
,"MchpUsbI2CSetClk");
```

```
   USB2642_I2CWrite_Demo= (pfnUSB2642_I2CWrite)GetProcAddress( LoadLib
,"MchpUsbI2CWrite");
   USB2642_I2CRead_Demo= (pfnUSB2642_I2CRead)GetProcAddress( LoadLib
,"MchpUsbI2CRead");

   MCHP_USB_ERROR error;
   HANDLE hFile;
   HANDLE hFile1;
   BYTE response[MAX_RESPONSE_BYTE] = {0};
   CDB cdbCDB;

   SCSI_PASS_THROUGH_STATUS sptsStatus;
   MCHP_USB_ERROR bRet;
   SD_INQUIRY_DATA souidData;
   BYTE initDelay = 0x00;
   UINT32 block_count = 1;
   UINT8 initDelay2 = 20;
   UINT VID;
   UINT PID;
   UINT DID = 0;

   unsigned char WrData;
   unsigned char SlavAddr;
   unsigned char RegAddr;
   char overflow[5];
   unsigned char data[4];


   wprintf(L"Enter VID of the device:");
   //wscanf (L"%x",&VID);
   wprintf(L"Enter PID of the device:");
//  wscanf (L"%x",&PID);
   wprintf(L"Enter DeviceID if multiple devices are present(0,1,2...):");
   //wscanf (L"%d",&DID);
   VID = 0x0424;
   PID = 0x4041;
   DID = 0;
   bRet =USB2642_Open_Demo(VID,PID,DID, &hFile );

   if(hFile == INVALID_HANDLE_VALUE)
   {
      return MCHP_Error_Invalid_Device_Handle;
   }

   int i = 5;

   USB2642_SD_GPIO1_Init_Demo(hFile, NULL);
   do
   {
      if( FALSE == USB2642_SD_GPIO1_Set_Demo(hFile, TRUE))
      {
         printf("Failed to set GPIO1 high\n");
         break;
      }
      Sleep(1000);
      if(FALSE == USB2642_SD_GPIO1_Set_Demo(hFile, FALSE))
      {
         printf("Failed to set GPIO1 low\n");
         break;
      }
```

```
       printf("count is %x\n", i);
       Sleep(1000);
    }while (i--);

    if(i)
    {
       printf("Failed GPIO test loop\n");

    }

//  Standard SCSI Inquiry
    ZeroMemory(&cdbCDB, sizeof(CDB));
    cdbCDB.CDB6INQUIRY3.OperationCode = SCSIOP_INQUIRY;
    cdbCDB.CDB6INQUIRY3.AllocationLength = INQUIRYDATABUFFERSIZE;
    INQUIRYDATA iInquiryData;
    ZeroMemory(&iInquiryData, sizeof(INQUIRYDATA));

    bRet = USB2642_SCSIPassThroughDirect_Demo(hFile, &cdbCDB,
sizeof(CDB::_CDB6INQUIRY3), sptsStatus, &iInquiryData, INQUIRYDATABUFFERSIZE, FALSE);
    wprintf(L" Executing SCSiInquiryData\n");
    if(bRet != MCHP_Error_Success)
    {
       CloseHandle(hFile);
       return -1;
    }

    //SD Inquiry
    bRet = USB2642_SDInquiry_Demo(hFile, souidData);
    wprintf(L" Executing SDInquiry\n");
    if(bRet != MCHP_Error_Success)
    {
       CloseHandle(hFile);
       return -1;
    }

    if((souidData.ProtocolVersion == 1) &&
       (souidData.SDHostControllerVersionMajor == 0x02) &&
(souidData.SDHostControllerVersionMinor == 0x00))
    {

       switch(souidData.ProtocolClass)
       {

          case SD_OVER_USB_PROTOCOL_CLASS_CLASS3:

             //Issue SD Card Initialize to query current card setting such as RCA,
OCR,CID memory type etc..
             bRet = USB2642_SDCardInitialize_Demo(hFile, SD_OVER_USB_INIT_STATUS_ONLY,
scidInitializationData,initDelay2);

             //Read the 16byte CID register0
             BYTE value[16];
             memset(value, 0xff, 16);
             bRet = USB2642_SDGetCID_Demo(hFile,value);

             //Read the 8 byte SCR register
             memset(value, 0xff, 8);
             bRet = USB2642_SDGetSCR_Demo(hFile,value);

             //Access the CSD 16 byte register
```

```
            memset(value, 0xff, 16);
            bRet = USB2642_SDGetCSD_Demo(hFile,value);

            wprintf(L" Executing SDCardInitialize\n");
            printf("SD Card Initialize\n");

            printf("  SD memory function type %u\n",
scidInitializationData.SDMemoryFunctionType);

            printf("  SDIO %u, Card Locked %u/Error %u/Present %u/Power %u\n",
         scidInitializationData.SDIO, scidInitializationData.CardLocked,
         scidInitializationData.CardError,
         scidInitializationData.CardPresent,
         scidInitializationData.CardPower);

         printf("  Bus Clock Speed 0x%x\n", scidInitializationData.BusClockSpeed);

         printf("  Bus Width %u\n", scidInitializationData.BusWidth);

         printf("  SDIO Functions %u\n", scidInitializationData.SDIOFunctions);

         printf("  RCA %02x %02x\n",
            scidInitializationData.RCA[0], scidInitializationData.RCA[1]);
         printf("  CID\n    ");

                  int i;

                  for (i=0; i<16; i++)
                     printf(" %02x", scidInitializationData.CID[i]);
                  printf("\n");

                  printf("  Memory OCR\n    ");

                  for (i=0; i<4; i++)
                     printf(" %02x", scidInitializationData.MemoryOCR[i]);
                     printf("\n");

                  printf("  SDIO OCR\n    ");

                  for (i=0; i<3; i++)
                     printf(" %02x", scidInitializationData.SDIOOCR[i]);


            if(bRet != TRUE)
            {
               CloseHandle(hFile);
               return MCHP_Error_OnExit;
            }

            if((scidInitializationData.CardPower == TRUE) &&
(scidInitializationData.CardPresent == TRUE) && (scidInitializationData.CardError ==
FALSE))
            {
               if(((scidInitializationData.SDMemoryFunctionType >
SD_MEMORY_FUNCTION_TYPE_NONE) && (scidInitializationData.SDMemoryFunctionType <=
SD_MEMORY_FUNCTION_TYPE_SDHC_SDXC)) || (scidInitializationData.SDIO == TRUE))
               {
                  //Memory and/or SDIO functions present on card
                  //Card in Stand-by state
```

```
                    //Example_EnhancedModeConfig(hFile,*((WORD
*)&scidInitializationData.RCA[0]));

                    MchpUsbGetExtCSD(hFile,*((WORD *)&scidInitializationData.RCA[0]));
                }
            }
             else
            {
                    //Unusable/Unknown card
                    CloseHandle(hFile);
                    return -1;
            }
            break;


      }
   }
   else
   {
      //Incorrect versions for this driver
      CloseHandle(hFile);
      return MCHP_Error_OnExit;
   }

   MchpUsbGetExtCSD(hFile,*((WORD *)&scidInitializationData.RCA[0]));


   CloseHandle(hFile);
   wprintf(L" Press any key to continue... ");
   getchar();

   return MCHP_Error_Success;
}
```

Worldwide Sales and Service

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
http://www.microchip.com/support
Web Address:
www.microchip.com

Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

Austin, TX
Tel: 512-257-3370

Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Cleveland
Independence, OH
Tel: 216-447-0464
Fax: 216-447-0643

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Novi, MI
Tel: 248-848-4000

Houston, TX
Tel: 281-894-5983

Indianapolis
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608

New York, NY
Tel: 631-435-6000

San Jose, CA
Tel: 408-735-9110

Canada - Toronto
Tel: 905-673-0699
Fax: 905-673-6509

ASIA/PACIFIC

Asia Pacific Office
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon
Hong Kong
Tel: 852-2943-5100
Fax: 852-2401-3431

Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

China - Beijing
Tel: 86-10-8569-7000
Fax: 86-10-8528-2104

China - Chengdu
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

China - Chongqing
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500

China - Hangzhou
Tel: 86-571-8792-8115
Fax: 86-571-8792-8116

China - Hong Kong SAR
Tel: 852-2943-5100
Fax: 852-2401-3431

China - Nanjing
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

China - Qingdao
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

China - Shanghai
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

China - Shenyang
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

China - Shenzhen
Tel: 86-755-8864-2200
Fax: 86-755-8203-1760

China - Wuhan
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

China - Xian
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

China - Xiamen
Tel: 86-592-2388138
Fax: 86-592-2388130

China - Zhuhai
Tel: 86-756-3210040
Fax: 86-756-3210049

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123

India - New Delhi
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

India - Pune
Tel: 91-20-3019-1500

Japan - Osaka
Tel: 81-6-6152-7160
Fax: 81-6-6152-9310

Japan - Tokyo
Tel: 81-3-6880- 3770
Fax: 81-3-6880-3771

Korea - Daegu
Tel: 82-53-744-4301
Fax: 82-53-744-4302

Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 or
82-2-558-5934

Malaysia - Kuala Lumpur
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

Malaysia - Penang
Tel: 60-4-227-8870
Fax: 60-4-227-4068

Philippines - Manila
Tel: 63-2-634-9065
Fax: 63-2-634-9069

Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

Taiwan - Hsin Chu
Tel: 886-3-5778-366
Fax: 886-3-5770-955

Taiwan - Kaohsiung
Tel: 886-7-213-7830

Taiwan - Taipei
Tel: 886-2-2508-8600
Fax: 886-2-2508-0102

Thailand - Bangkok
Tel: 66-2-694-1351
Fax: 66-2-694-1350

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany - Dusseldorf
Tel: 49-2129-3766400

Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Germany - Pforzheim
Tel: 49-7231-424750

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Italy - Venice
Tel: 39-049-7625286

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

Poland - Warsaw
Tel: 48-22-3325737

Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

Sweden - Stockholm
Tel: 46-8-5090-4654

UK - Wokingham
Tel: 44-118-921-5800
Fax: 44-118-921-5820

03/25/14