

User Manual

DA16200 SDIO Host Interface

UM-WI-053

Abstract

This user manual describes the communication method between DA16200 SDIO Slave and Host Processor

Contents

Abstract	1
Contents	2
Figures	2
Tables	2
1 Terms and Definitions	3
2 References	3
3 Introduction	4
3.1 PIN MUX Configuration	4
4 SDIO Protocol	5
4.1 Message Format	5
4.1.1 Address (included in the header)	5
4.1.2 Length (included in the header)	5
4.1.3 Interrupt	5
4.2 Write Sequence	6
4.3 Read Sequence and Structure	8
5 AT Command - Sequences and Structures	9
6 Definition & Structures for Implementation	11
Revision History	12

Figures

Figure 1 : Basic Format	5
Figure 2 : Write Sequence	6
Figure 3: Structure	6
Figure 4: Read Sequence	8
Figure 5: Structure	8
Figure 6: AT Command Sequence	9
Figure 7: Structure	9

Tables

Table 1: Pin MUX Configuration of SDIO	4
Table 2: Address List	5
Table 3 : Definition	11
Table 4: Response Structure	11
Table 5: Request Structure	11

DA16200 SDIO Host Interface

1 Terms and Definitions

SDIO Secure Digital Input Output

2 References

- [1] DA16200, Datasheet, Dialog Semiconductor
- [2] DA16200, EVK User Manual, User Manual, Dialog Semiconductor
- [3] DA16200, SDK Programmer User Manual, User Manual, Dialog Semiconductor

DA16200 SDIO Host Interface

3 Introduction

This application note describes how an external processor system, called "External Host" hereafter, communicates with a DA16200 over SDIO physical interface protocol. This document also includes the "AT Command Protocol" for use with the External Host.

3.1 PIN MUX Configuration

SDIO slave is assigned to GPIOA[9:4] in DA16200. In the case of interrupt, the D1 port of SDIO can be used, but in some cases, GPIO can also be used.

However, there may be a pin mux initialization code in Dialog's SDK that may look as follows:

- `_da16x_io_pinmux(PIN_CMUX, CMUX_GPIO);` // For GPIOA 4,5
- `_da16x_io_pinmux(PIN_DMUX, DMUX_GPIO);` // For GPIOA 6,7
- `_da16x_io_pinmux(PIN_EMUX, EMUX_GPIO);` // For GPIOA 8,9

This means GPIOA[9:4] is to be used as GPIOs, not SDIO slave.

Therefore, it needs to change to the following code for SDIO slave at GPIOA[9:4]:

- `_da16x_io_pinmux(PIN_CMUX, CMUX_SDs);` // For GPIOA 4,5
- `_da16x_io_pinmux(PIN_DMUX, DMUX_SDs);` // For GPIOA 6,7
- `_da16x_io_pinmux(PIN_EMUX, EMUX_SDs);` // For GPIOA 8,9

If SDIO D1 Port is not used as Interrupt and GPIO is used as Interrupt, the following PAD Mux Setting is additionally required.

- `_da16x_io_pinmux(PIN_FMUX, FMUX_GPIO);` // For GPIOA 10,11

Table 1: Pin MUX Configuration of SDIO

GPIO	Signal Name
GPIOA[4]	CMD
GPIOA[5]	CLK
GPIOA[6]	D3
GPIOA[7]	D2
GPIOA[8]	D1
GPIOA[9]	D0

DA16200 SDIO Host Interface

4 SDIO Protocol

4.1 Message Format

The format of the messages sent/received to/from the external processor is the DA16200 protocol format over SDIO physical interface. The format of the message in DA16200 and the parameters included are outlined in [Figure 1](#). For more details on SDIO Header configuration, refer to SDIO Specification. And, when using SDIO Protocol of DA16200, data should be aligned in units of 4 Bytes length.



Figure 1 : Basic Format

4.1.1 Address (included in the header)

The address list used by External Host is outlined in [Table 2](#).

Table 2: Address List

Address Type	Address
General Command (Write Request)	0x50080254
AT Command	Received from slave in initial stage.
Response Command	0x50080258
Buffer Address	Received from slave in response message

4.1.2 Length (included in the header)

Payload Length to follow.

4.1.3 Interrupt

According to SDIO Specification, Slave can cause Interrupt to Host by using D1 port. However, if the host cannot receive the interrupt using the D1 port, it must use the GPIO to generate the interrupt. The interrupt line used in the sequence diagram of this document means that the D1 port or GPIO is used.

DA16200 SDIO Host Interface

4.2 Write Sequence

Host to Slave write operations are performed in a three SDIO transaction. The following sequence

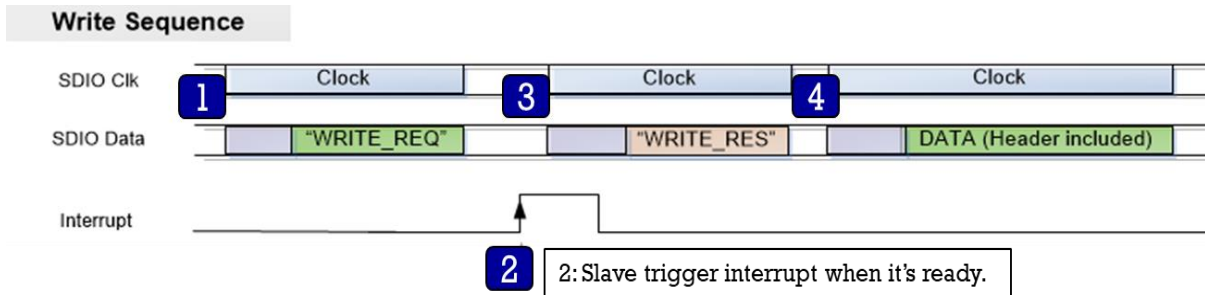


Figure 2 : Write Sequence

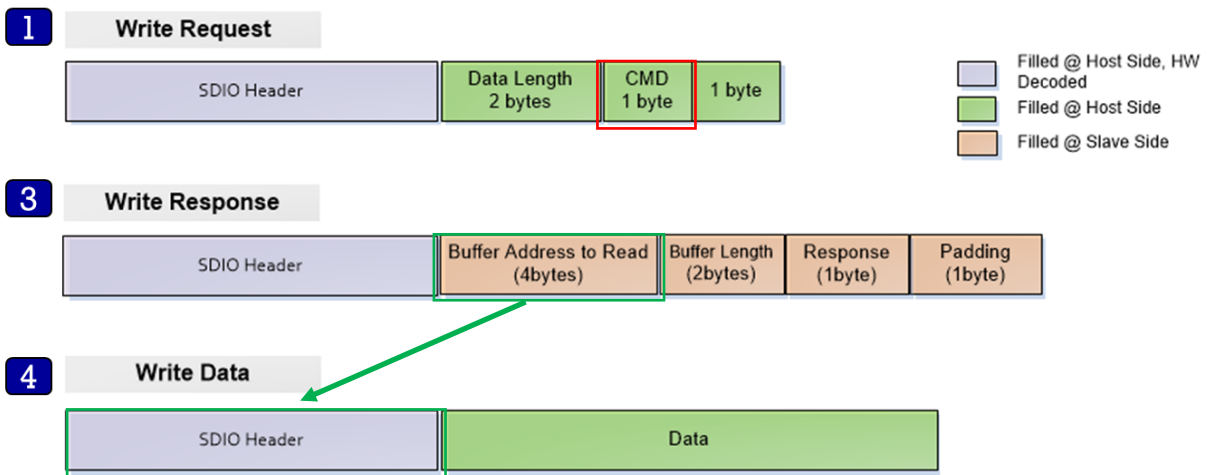


Figure 3: Structure

1. The Host sends a "WRITE_REQ" command (0x80, red rectangle in Figure 3) to the "General Command" address (0x50080254).
2. The Host should wait for Interrupt from slave.
3. The Host reads the Write Response message by "Response Command" address (0x50080258) and parse it using "struct _st_host_response" (see Table 4) .
4. The Host sends data to address (BUFF_ADDR) which is received from the Slave in the Write Response message.(Green rectangle in Figure 3).

An interval of several hundred microseconds is required between the “3” and “4” stages. If the interval between the two stages is too short, there is a possibility that two Interrupt Events are recognized as one. The interval differs depending on the type of application or CPU load. Roughly, when the CPU clock is 120 MHz, an interval of around 300 µs is required.

Example

When the host wants to write 8 bytes data (0x88776655,0x44332211) to DA16200:

1. Host sends:
(SDIO Write-0x50080254, 4Bytes) - (0x08-0x00-0x80-0x00)
2. Host waits for interrupt triggering from DA16200.

3. Host sends:

(SDIO Read-0x50080258, 8Bytes) then read response from DA16200.

Assume the buffer address from Slave is 0x12345678 for easy description.

Then the read data should be 0x78-0x56-0x34-0x12-0x08-0x00-0x81-0x00.

4. Host sends

(SDIO Write-0x12345678, 8Bytes)-(0x55-0x66-0x77-0x88-0x11-0x22-0x33-0x44)

DA16200 SDIO Host Interface

4.3 Read Sequence and Structure

Figure 4 shows a Slave device transmitting data to the Host when payload is available. This sequence is performed in a two SDIO transaction.

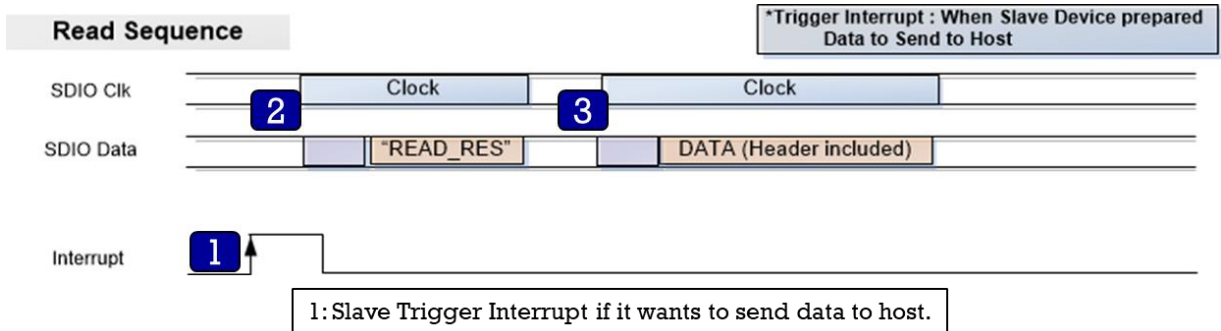


Figure 4: Read Sequence

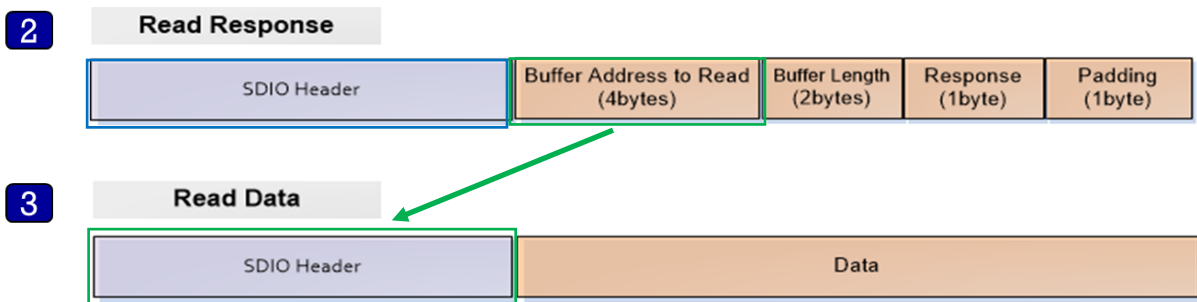


Figure 5: Structure

1. The Slave will trigger interrupt to inform the Host when data is available.
2. The Host reads the response message from "Response Command" address (0x50080258, blue rectangle in Figure 5), and parse it using "struct _st_host_response". (see Table 4).
3. The Host reads data from address (BUFF_ADDR) which is received from Slave in the response message. (Green rectangle in Figure 5)

An interval of several hundred microseconds is required between the "2" and "3" stages. If the interval between the two stages is too short, there is a possibility that two Interrupt Events are recognized as one. The interval differs depending on the type of application or CPU load. Roughly, when the CPU clock is 120 MHz, an interval of around 300 μs is required.

Example

1. When the host received interrupt from DA16200,
2. Host sends:
(SDIO Read-0x50080258, 8Bytes) then read response from DA16200.

Assume the buffer address from Slave is 0x12345678 for easy description and the data length to be sent from DA16200 is 8 bytes.

The read data should be 0x78-0x56-0x34-0x12-0x08-0x00-0x83-0x00.

3. Host sends:
(SDIO Read-0x12345678, 8Bytes) then read data from DA16200.

5 AT Command - Sequences and Structures

AT commands are instructions used to control a modem. AT is the abbreviation of ATtention. Every command line starts with "AT" or "at". ... Note that the starting "AT" is the prefix that informs the modem about the start of a command line. It is not part of the AT command name.

In order to use AT Command, the address of AT Command Buffer must be read at the initial stage. Therefore, read the value of address 0x50080264 after SDIO is initialized, and write the command to that address when sending AT command afterwards.

Figure 6 illustrates how to use the AT Command through SDIO in "DA16200". This is because AT Command uses a predetermined address and the maximum size of data is defined.

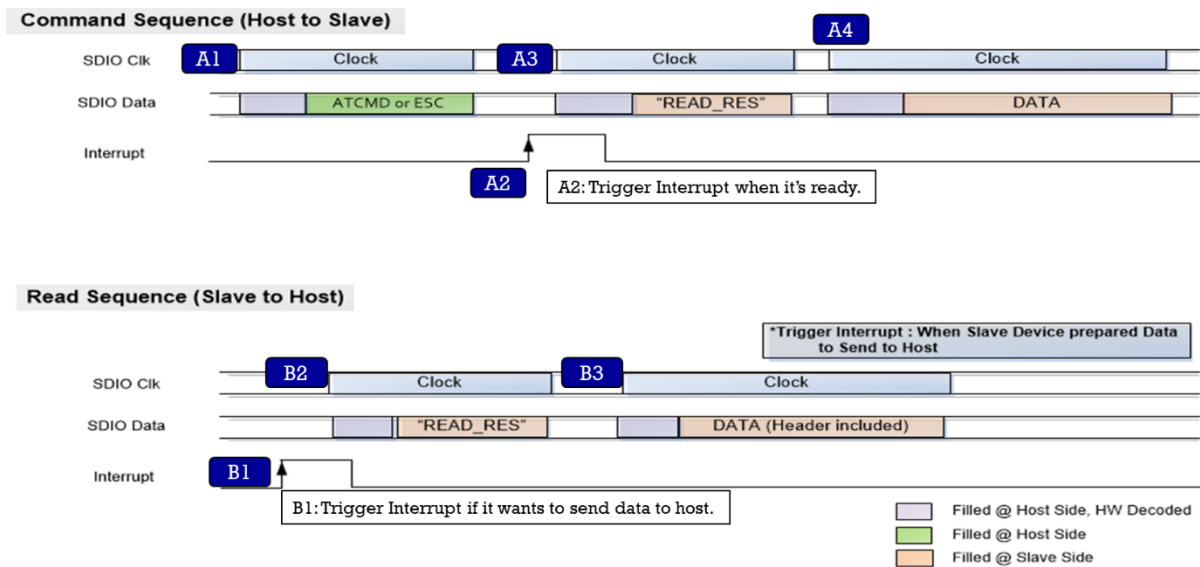


Figure 6: AT Command Sequence

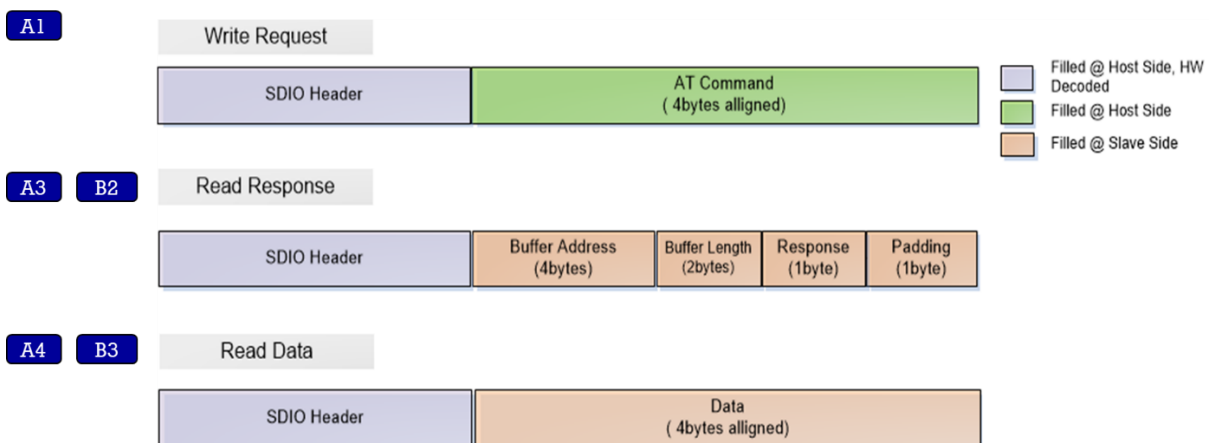


Figure 7: Structure

DA16200 SDIO Host Interface

Description of [Figure 6](#) and [Figure 7](#).

A1: The Host sends a "AT" or "ESC" command to "AT Command" address.

A2: The Host waits for interrupt trigger.

A3: The Host reads the response message from address and parses it using "struct _st_host_response"

A4: The Host reads "OK", "Error" or data from address (BUF_ADDR), depending on the type of command. The result for "esc" command is sent to the host in the "response" field of "struct _st_host_response". The "response" field is a 1 byte decimal value. A value of 0x20 is a result of "OK". All other values are an "ERROR".

B1: The Slave will toggle high the interrupt line to inform Host when data is available

B2: The Host reads the response message from "Response Command" address, and parses it using "struct _st_host_response".

B3: The Host reads data from address (BUF_ADDR) parsed from the response message.

An interval of several hundred microseconds is required between the "A3" and "A4" stages, "B2" and "B3" stages. If the interval between the two stages is too short, there is a possibility that two Interrupt Events are recognized as one. The interval differs depending on the type of application or CPU load. Roughly, when the CPU clock is 120 MHz, an interval of around 300 μ s is required.

6 Definition & Structures for Implementation

Table 3 : Definition

#define	HOST_MEM_WRITE_REQ	(0x80)	
#define	HOST_MEM_WRITE_RES	(0x81)	
#define	HOST_MEM_READ_REQ	(0x82)	
#define	HOST_MEM_READ_RES	(0x83)	
#define	FC9K_GEN_CMD_ADDR	(0x50080254)	// Address to Write Command
#define	FC9K_RESP_ADDR	(0x50080258)	// Address to Read Response

Table 4: Response Structure

```
typedef struct _st_host_response
{
    u32 buf_address;
    u16 host_length;
    u8  resp;
    u8  dummy;
} st_host_response;
```

Table 5: Request Structure

```
typedef struct _st_host_request
{
    u16 host_write_length;
    u8  host_cmd;
    u8  dummy;
} st_host_request;
```

Revision History

Revision	Date	Description
1.3	22-Jun-2022	Response for "esc" command.
1.2	23-May-2022	Update logo, disclaimer, copyright.
1.1	25-Nov-2021	Delay between two SDIO access
1.0	01-Sep-2021	First Release

DA16200 SDIO Host Interface**Status Definitions**

Status	Definition
DRAFT	The content of this document is under review and subject to formal approval, which may result in modifications or additions.
APPROVED or unmarked	The content of this document has been approved for publication.

RoHS Compliance

Dialog Semiconductor's suppliers certify that its products are in compliance with the requirements of Directive 2011/65/EU of the European Parliament on the restriction of the use of certain hazardous substances in electrical and electronic equipment. RoHS certificates from our suppliers are available on request.

DA16200 SDIO Host Interface

Important Notice and Disclaimer

RENESAS ELECTRONICS CORPORATION AND ITS SUBSIDIARIES ("RENESAS") PROVIDES TECHNICAL SPECIFICATIONS AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES "AS IS" AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for developers skilled in the art designing with Renesas products. You are solely responsible for (1) selecting the appropriate products for your application, (2) designing, validating, and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. Renesas grants you permission to use these resources only for development of an application that uses Renesas products. Other reproduction or use of these resources is strictly prohibited. No license is granted to any other Renesas intellectual property or to any third party intellectual property. Renesas disclaims responsibility for, and you will fully indemnify Renesas and its representatives against, any claims, damages, costs, losses, or liabilities arising out of your use of these resources. Renesas' products are provided only subject to Renesas' Terms and Conditions of Sale or other applicable terms agreed to in writing. No use of any Renesas resources expands or otherwise alters any applicable warranties or warranty disclaimers for these products.

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu

Koto-ku, Tokyo 135-0061, Japan

www.renesas.com

Contact Information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:

<https://www.renesas.com/contact/>

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.