

User Manual

DA16200 SPI Host Interface

UM-WI-020

Abstract

This user manual describes the communication method between DA16200 SPI 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 SPI Protocol	5
4.1 Message Format	5
4.1.1 Address.....	5
4.1.2 CMD.....	5
4.1.3 Length	5
4.2 Write Sequence.....	6
4.3 Read Sequence and Structure.....	7
5 AT Command – Sequences and Structures	9
6 Header Format	10
7 Definition and Structures for Implementation	11
Revision History	12

Figures

Figure 1: Basic Format	5
Figure 2: Write Sequence.....	6
Figure 3: Structure for Write Operation	6
Figure 4: Read Sequence.....	7
Figure 5: Structure for Read Operation	7
Figure 6: AT Command Sequence.....	9
Figure 7: Structure of AT CMD	9
Figure 8: SPI Signals for Write Request.....	10
Figure 9: SPI Signals for Read Response.....	10

Tables

Table 1: Pin MUX Configuration of SPI	4
Table 2: Address List.....	5
Table 3: CMD Format (Command).....	5
Table 4: Definition.....	11
Table 5: Response Structure.....	11
Table 6: Request Structure.....	11

DA16200 SPI Host Interface

1 Terms and Definitions

SPI Serial Peripheral Interface

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 SPI Host Interface

3 Introduction

This section introduces the subject or problem described in this document. This application note describes how an external processor system, called “External Host” hereafter, communicates with a DA16200 over SPI physical interface protocol. This document also includes the AT Command Protocol to be used with the External Host.

3.1 PIN MUX Configuration

SPI slave is assigned to GPIOA[3:0] or GPIOA[9:6] in DA16200. Specifically, GPIOA[9:6] is assigned as SPI slave by HW default, and is recommended to be used as SPI slave operation.

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

- `_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[9:6] is to be used as GPIOs, not SPI slave.

Therefore, the code should be changed to the following code for SPI slave at GPIOA[9:6]:

- `_da16x_io_pinmux(PIN_DMUX, DMUX_SPIs); // For GPIOA 6,7`
- `_da16x_io_pinmux(PIN_EMUX, EMUX_SPIs); // For GPIOA 8,9`

Table 1: Pin MUX Configuration of SPI

GPIO	Signal Name
GPIOA[6]	CS
GPIOA[7]	CLK
GPIOA[8]	MISO
GPIOA[9]	MOSI

DA16200 SPI Host Interface

4 SPI Protocol

4.1 Message Format

The format of the messages sent/received to/from the external processor is the DA16200 protocol format over SPI physical interface. The format of the message in DA16200 and the parameters included are outlined in [Figure 1](#).



Figure 1: Basic Format

4.1.1 Address

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	0x50080260
Response Command	0x50080258
Buffer Address	Received from slave in response message

4.1.2 CMD

The format of CMD field is outlined in [Table 3](#).

Table 3: CMD Format (Command)

Bit Field	Abr.	Description
7	Auto_Inc	1: Internal Address auto-increment, 0: Address Fixed (Not used)
6	Read/Write	1: Read, 0: Write
5:2		Not Used
1:0	CHIP_ID[1:0]	00: CHIP #0 (Default)

4.1.3 Length

Payload Length of the Data field.

4.2 Write Sequence

Host to Slave write operations are performed in three SPI transactions as shown in Figure 2.

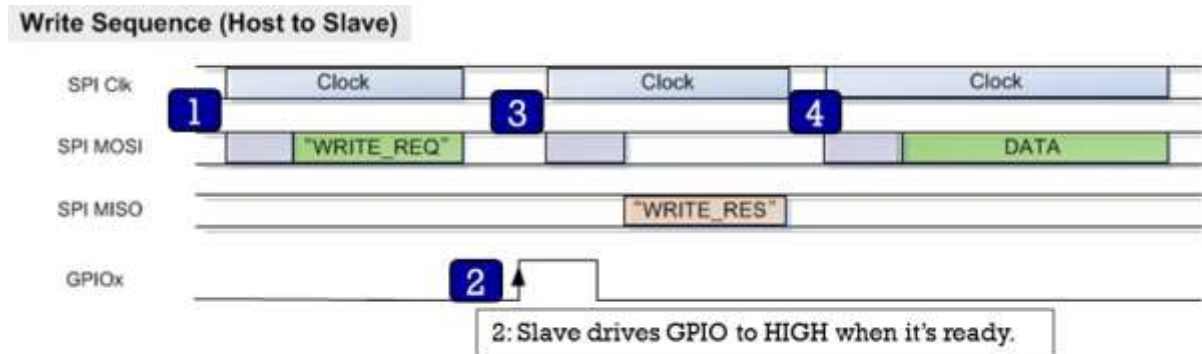


Figure 2: Write Sequence

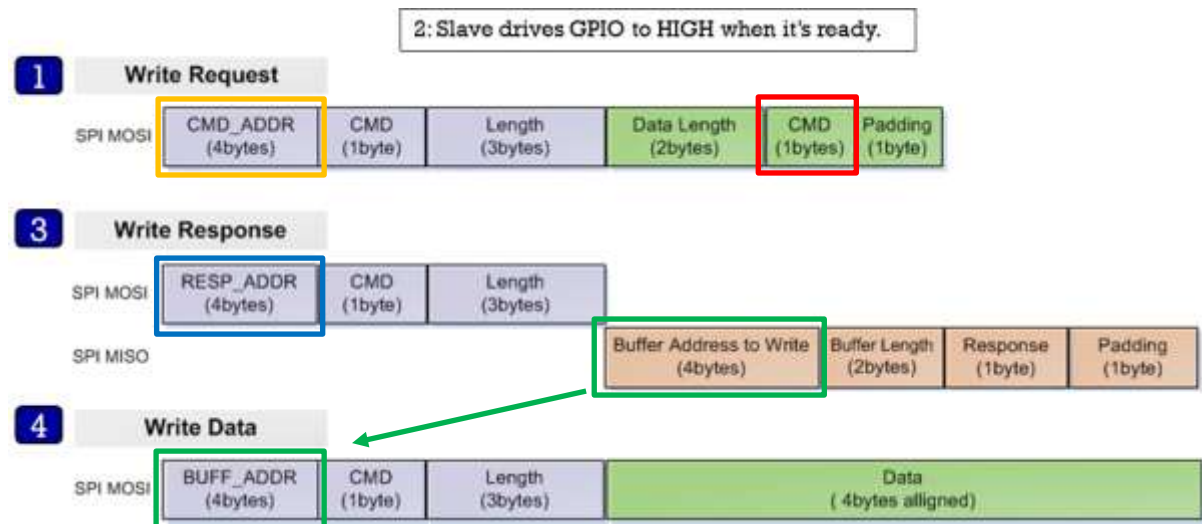


Figure 3: Structure for Write Operation

1. The Host sends a WRITE_REQ command (0x80, red rectangle in Figure 3) to the General Command address (0x50080254).(yellow rectangle in Figure 3).
2. The Host should wait for GPIO High from slave.
3. The Host reads the Write Response message by Response Command address (0x50080258, blue rectangle in Figure 3) 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.

DA16200 SPI Host Interface

Example

When the host wants to write eight bytes data (0x8877665544332211) to DA16200:

1. Host sends: (0x50-0x08-0x02-0x54)-(0x80)-(0x00-0x00-0x04)-(0x08-0x00-0x80-0x00)

NOTE
Payload data is LSB first. See Figure 8 .

2. Host waits until GPIO is high from DA16200.
3. Host sends (0x50-0x08-0x02-0x58)-(0xC0)-(0x00-0x00-0x08), then read response from DA16200.
Let's 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 (0x12-0x34-0x56-0x78)-(0x80)-(0x00-0x00-0x08)-(0x11-0x22-0x33-0x44-0x55-0x66-0x77-0x88)

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* SPI transaction.

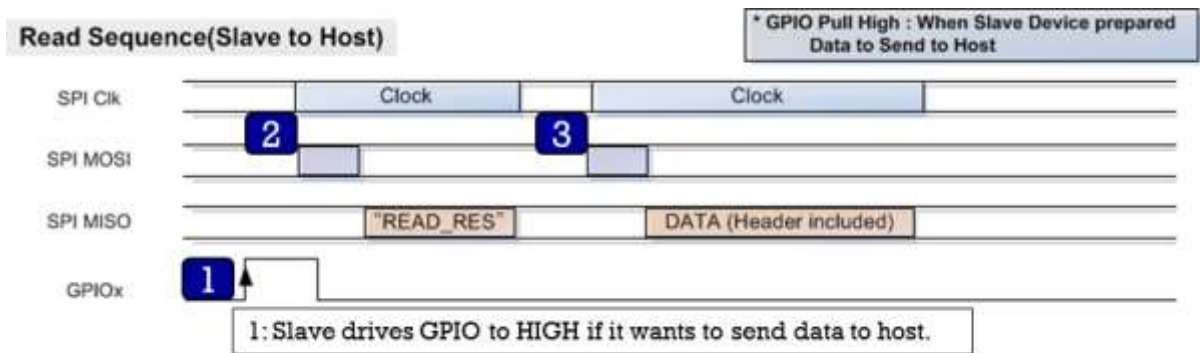


Figure 4: Read Sequence

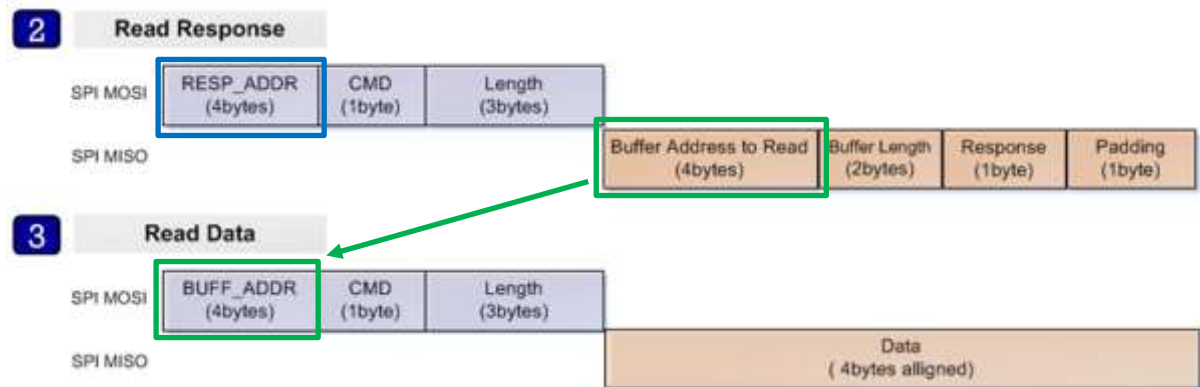


Figure 5: Structure for Read Operation

1. The Slave toggles *high* the interrupt line 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](#))

DA16200 SPI Host Interface

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 μ s is required.

Example

1. When the host becomes *high* on GPIO from DA16200, the host sends:
(0x50-0x08-0x02-0x58)-(0xC0)-(0x00-0x00-0x08), then read response from DA16200.
Let's assume the buffer address from Slave is 0x12345678 for easy description and the data length to be sent from DA16200 is eight bytes.
2. The read data should be 0x78-0x56-0x34-0x12-0x08-0x00-0x83-0x00.
3. Host sends: (0x12-0x34-0x56-0x78)-(0xC0)-(0x00-0x00-0x08), then read data from DA16200.
Note that read data is LSB first. (see [Figure 9](#))

DA16200 SPI Host Interface

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.

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

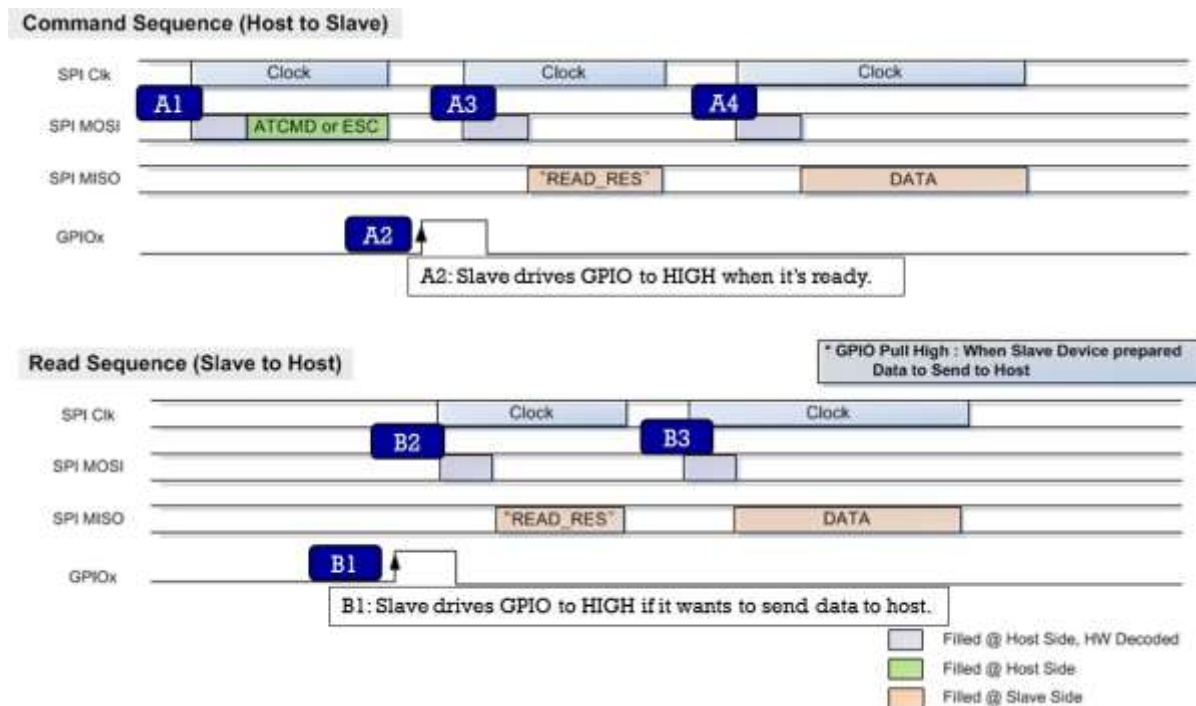


Figure 6: AT Command Sequence

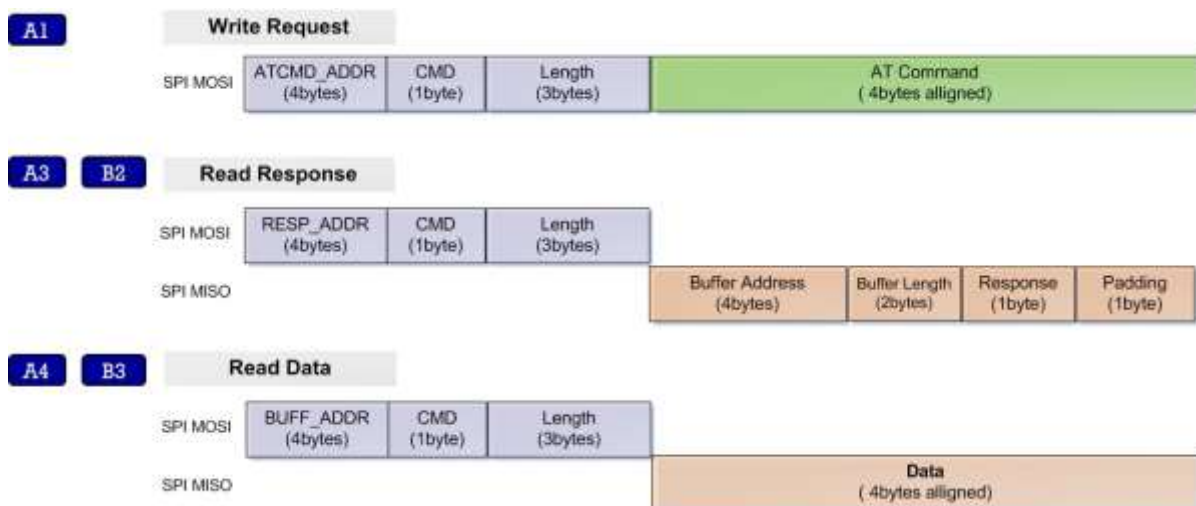


Figure 7: Structure of AT CMD

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

A2: The Host waits for GPIO interrupt to go high.

DA16200 SPI Host Interface

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.

B1: The Slave toggles 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 μs is required.

6 Header Format

Write Request (Host to Slave)

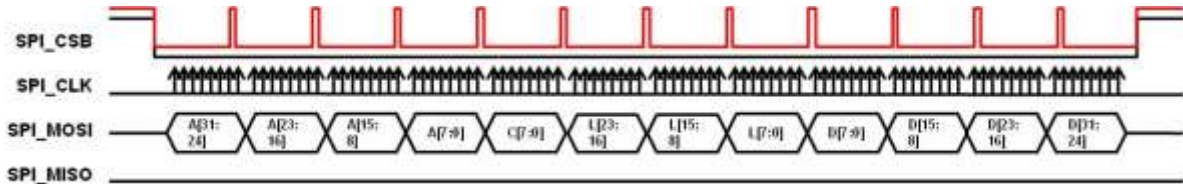
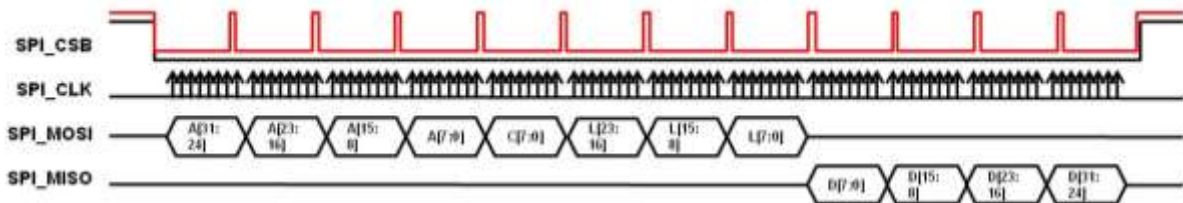


Figure 8: SPI Signals for Write Request

Read Response (Slave to Host)



CMD (Bit field)	Abc.	Description	
7	Auto_Inc	1 = internal Address auto-increment	0 = address fixed (not used)
6	Read/Write	1 = Read	0 = Write
5:2		Not used	
1:0	CHIP_ID[1:0]	00 = Chip #0 (default)	

Figure 9: SPI Signals for Read Response

DA16200 SPI Host Interface

7 Definition and Structures for Implementation

Table 4: 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
#define	FC9K_ATCMD_ADDR	(0x50080260)	// Address to Send AT Command

Table 5: Response Structure

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

Table 6: Request Structure

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

DA16200 SPI Host Interface**Revision History**

Revision	Date	Description
1.2	23-Nov-2021	Delay between two SDIO access
1.1	23-Aug-2021	Changed API from _fc9k_io_pinmux() to _da16x_io_pinmux().
1.0	7-Apr-2020	First Release.

DA16200 SPI 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.

Disclaimer

Unless otherwise agreed in writing, the Dialog Semiconductor products (and any associated software) referred to in this document are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of a Dialog Semiconductor product (or associated software) can reasonably be expected to result in personal injury, death or severe property or environmental damage. Dialog Semiconductor and its suppliers accept no liability for inclusion and/or use of Dialog Semiconductor products (and any associated software) in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Information in this document is believed to be accurate and reliable. However, Dialog Semiconductor does not give any representations or warranties, express or implied, as to the accuracy or completeness of such information. Dialog Semiconductor furthermore takes no responsibility whatsoever for the content in this document if provided by any information source outside of Dialog Semiconductor.

Dialog Semiconductor reserves the right to change without notice the information published in this document, including, without limitation, the specification and the design of the related semiconductor products, software and applications. Notwithstanding the foregoing, for any automotive grade version of the device, Dialog Semiconductor reserves the right to change the information published in this document, including, without limitation, the specification and the design of the related semiconductor products, software and applications, in accordance with its standard automotive change notification process.

Applications, software, and semiconductor products described in this document are for illustrative purposes only. Dialog Semiconductor makes no representation or warranty that such applications, software and semiconductor products will be suitable for the specified use without further testing or modification. Unless otherwise agreed in writing, such testing or modification is the sole responsibility of the customer and Dialog Semiconductor excludes all liability in this respect.

Nothing in this document may be construed as a license for customer to use the Dialog Semiconductor products, software and applications referred to in this document. Such license must be separately sought by customer with Dialog Semiconductor.

All use of Dialog Semiconductor products, software and applications referred to in this document is subject to Dialog Semiconductor's [Standard Terms and Conditions of Sale](#), available on the company website (www.dialog-semiconductor.com) unless otherwise stated.

Dialog, Dialog Semiconductor and the Dialog logo are trademarks of Dialog Semiconductor Plc or its subsidiaries. All other product or service names and marks are the property of their respective owners.

© 2021 Dialog Semiconductor. All rights reserved.

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.

Contact Dialog Semiconductor

General Enquiry:

[Enquiry Form](#)

Local Offices:

<https://www.dialog-semiconductor.com/contact/sales-offices>