User Manual

DA16200 ThreadX Doorbell Application Guide

UM-WI-013

Abstract

DA16200 is a highly integrated ultra-low power Wi-Fi system on a chip (SoC) and allows users to develop the Wi-Fi solution on a single chip. The user implements their application with the DA16200 SDK and the compiling environment is configured by means of IAR Embedded Workbench IDE of IAR Systems.

This document is an SDK Guide document intended for developers that want to program the DA16200 chipset. And describes the SDK API and peripheral device drivers and interfaces.

Contents

Abstract **1**

Contents **2**

Figures **2**

Tables **2**

1 Terms and Definitions **3**

2 References **3**

3 Introduction **4**

3.1 Development Environment 4

3.2 Wi-Fi Setup 4

4 Implementation **6**

4.1 SDK Workspace 7

4.2 Hardware Configuration 7

4.3 RTSP (Real Time Streaming Protocol) 9

4.4 Interface Flow with ISP 10

Revision History **11**

Figures

Figure 1: Doorbell Development Environment 4
Figure 2: Wi-Fi Mode Setup 4
Figure 3: Connection to an AP 5
Figure 4: Doorbell System Procedure 6
Figure 5: IAR Workbench Workspace 7
Figure 6: Interface Flow with ISP 10

Tables

Table 1: Hardware Configuration for Doorbell Application 7
Table 2: SPI Slave Interface Configuration 9
Table 3: RTSP Request Commands 9
Table 4: Interface Command for Request and Response 10

1 Terms and Definitions

AP	Access Point
DPM	Dynamic Power Management
ISP	In-System Programming
RTSP	Real Time Streaming Protocol
SDK	Software Development Kit
SDIO	Secure Digital Input/Output
SPI	Serial Peripheral Interface
UDP	User Datagram Protocol

2 References

[1] UM-WI-002, DA16200 DA16600 ThreadX SDK Programmer Guide, User Manual, Dialog Semiconductor

[2] UM-WI-005, DA16200 DA16600 DPM Manager, User Manual, Dialog Semiconductor

[3] UM-WI-023, DA16200 Evaluation Kit, User Manual, Dialog Semiconductor

3 Introduction

This document serves as a programming guide for a user doorbell application with the use of the DA16200 Doorbell Application SDK. This SDK is designed to operate with an ISP for video/audio capturing. See the “DA16200 SDK Programmer Guide” [1] on how to use the SDK and see the “DA16200 DPM Manager Guide” [2] on how to use the DPM function.

3.1 Development Environment

The doorbell application operates in an environment as shown in Figure 1.

Figure 1: Doorbell Development Environment

- Message protocol: RTSP(by UDP)
- DA16200: Video data Tx and audio data Tx/Rx
- Mobile phone: User application

The DA16200 doorbell application implements the following actions:

- Connects to the ISP through SPI/SDIO and operates the ISP
- Connects to a mobile phone application with RTSP
- Receives video/audio data from the ISP and sends the data to the mobile phone
- Receives audio data from the mobile phone and sends that data to the ISP

The mobile phone application operates as follows:

- Connects to the DA16200 with RTSP
- Receives video/audio data
- Sends audio data

3.2 Wi-Fi Setup

The DA16200 Doorbell SDK does not implement AP provisioning, which means that the device has to be connected to an AP via a console command.

1. Use the `setup` command to set the station (STA) interface. See Figure 2.
   See section “Wi-Fi Mode Setup” in “DA16200 EVK User Manual” [3] about STA interface setup.

```
[/DA16200] #
[/DA16200] #
[/DA16200] # setup

Stop all services for the setting.
 Are you sure ? [Yes/No] : y
```

Figure 2: Wi-Fi Mode Setup

2. After reboot, confirm the connection to an AP. See Figure 3.

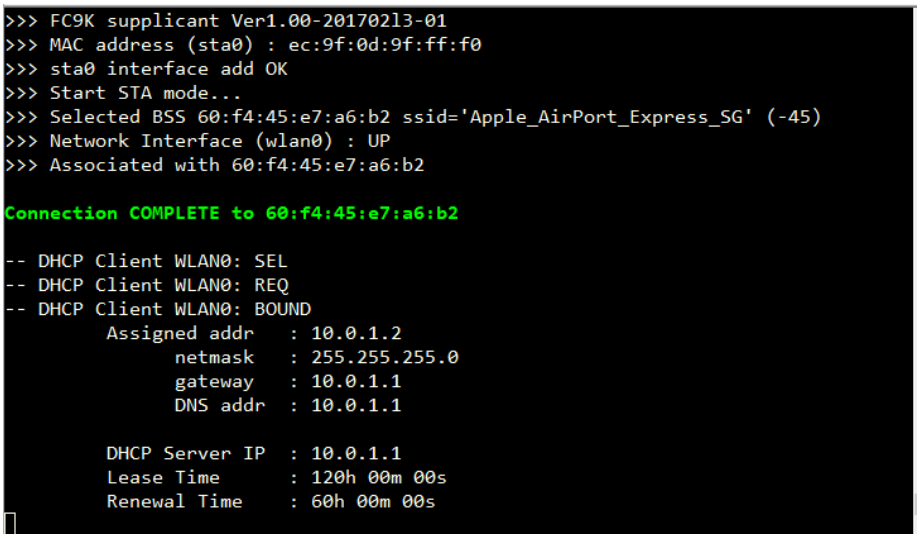

Figure 3: Connection to an AP

4 Implementation

Data transmission between DA16200 and a mobile phone application occurs through DA16200 controls and communicates with the ISP through a selected interface such as SPI or SDIO. See Figure 4.

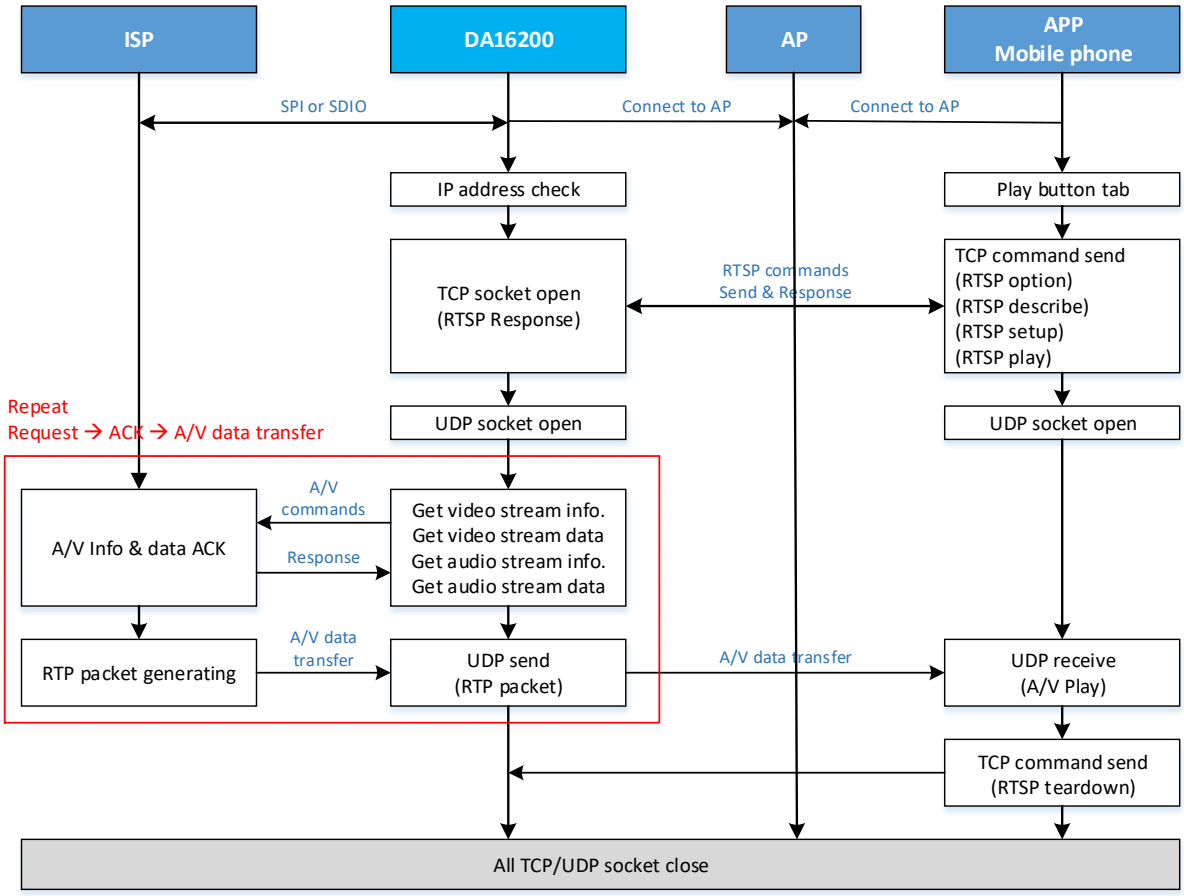

Figure 4: Doorbell System Procedure

4.1 SDK Workspace

After an application is written, right-click on the project name in the IAR Embedded Workbench workspace and then run **Make** or **Rebuild All**. When you compile for the first time, it is always good to run option **Clean** first. See Figure 5.

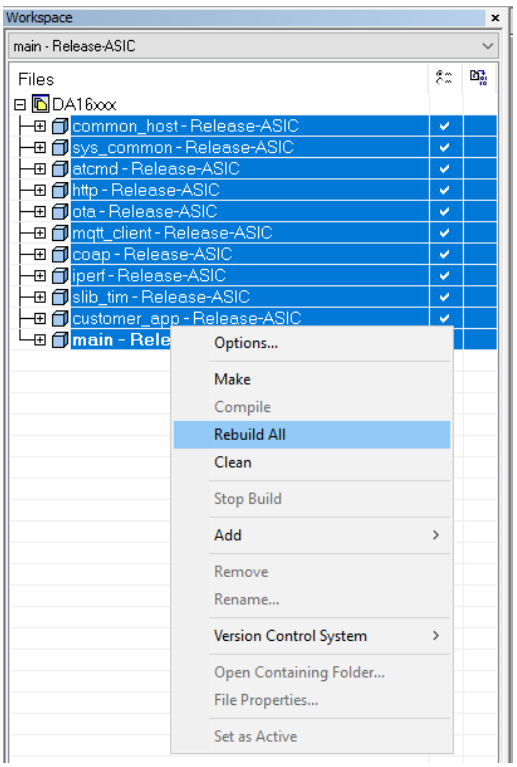

Figure 5: IAR Workbench Workspace

4.2 Hardware Configuration

The hardware pin configuration for a doorbell is implemented in the function shown in Table 1. CMUX and DMUX are for SPI, and EMUX is for the ISP interrupt (GPIO_6) to communicate.

Table 1: Hardware Configuration for Doorbell Application

```
[\core\main.c]

int config_pin_mux(void)
{
    /* DA16200 default pin-configuration */

#if defined ( __FOR_GENERIC_SDK__ )

  #if defined ( __DOORBELL_REF__ )
    /* DA16200 default pin-configuration for Doorbell */
    #if defined ( __TEST_DA16200_EVB__ )
        _da16x_io_pinmux(PIN_AMUX, AMUX_GPIO);
        _da16x_io_pinmux(PIN_BMUX, BMUX_GPIO);
        _da16x_io_pinmux(PIN_CMUX, CMUX_UART1d);
        _da16x_io_pinmux(PIN_DMUX, DMUX_GPIO);      // For GPIO 6,7
        _da16x_io_pinmux(PIN_EMUX, EMUX_GPIO);
        _da16x_io_pinmux(PIN_FMUX, FMUX_GPIO);
```



```
        _da16x_io_pinmux(PIN_HMUX, HMUX_JTAG);

    #elif defined ( __DOORBELL_IF_SDIO__ )

        _da16x_io_pinmux(PIN_CMUX, CMUX_SDs);
        _da16x_io_pinmux(PIN_DMUX, DMUX_SDs);       // For GPIO 6,7
        _da16x_io_pinmux(PIN_EMUX, EMUX_SDs);
        _da16x_io_pinmux(PIN_UMUX, UMUX_GPIO);

    #elif defined ( __DOORBELL_IF_SPI__ )

        _da16x_io_pinmux(PIN_DMUX, DMUX_SPIs);      // For GPIO 6,7
        _da16x_io_pinmux(PIN_EMUX, EMUX_SPIs);
        _da16x_io_pinmux(PIN_FMUX, UMUX_GPIO);

    #endif

    ...
    return TX_TURE;
}
```

- The SPI slave interface configuration is implemented in the function shown in Table 2.

Table 2: SPI Slave Interface Configuration

```
[\apps\da16200\doorbell_ref\src\doorbell_host_if_spi_user.c]

static void host_spi_slave_init(void)
{
      char     *nvram_string = NULL;

      nvram_string = read_nvram_string(DOORBELL_SPI_HIGHSPEED_MODE);

      if (!strcmp(nvram_string, "HALF")) {
        PRINTF(" ### SPI SPEED MODE (%s) \n", nvram_string);
        doorbellParams.spimode = SPI_HALFMODE;
        doorbellParams.delay = 30;
      } else {
        PRINTF(" ### SPI SPEED MODE (%s) \n", nvram_string);
        doorbellParams.spimode = SPI_NORMALMODE;
        doorbellParams.delay = 50;
      }

      if (doorbellParams.spimode == SPI_HALFMODE)
        FC9K_SLAVECTRL->SPI_Slave_CR = 0xF90B;
      else
        FC9K_SLAVECTRL->SPI_Slave_CR = 0xFD0B;

      FC9K_SLAVECTRL->Base_Addr = 0x00005000; // base 0x5000xxxx
      FC9K_DIOCFG->EXT_INTB_CTRL =  0xa;

}
```

- The SDIO slave interface configuration is processed automatically and does not need to be implemented.

4.3 RTSP (Real Time Streaming Protocol)

When the mobile application requests doorbell service initiation, the DA16200 and mobile application exchange the RTSP messages in Table 3 to configure the data transmission environment (Request – Response).

Table 3: RTSP Request Commands

Request	Operation
RTSP OPTIONS	Send supported RTSP options
RTSP DESCRIBE	Send video/audio data format (H264, PCM, Sample rate and so on)
RTSP SETUP	Send port number and session ID for video/audio data transmission
RTSP PLAY	Request for starting data streaming
RTSP TEARDOWN	Request for stopping data streaming

4.4 Interface Flow with ISP

Figure 6 shows the interface flow between ISP and DA16200 for video streaming. An audio streaming request and response has the same sequence as video streaming. See also Table 4.

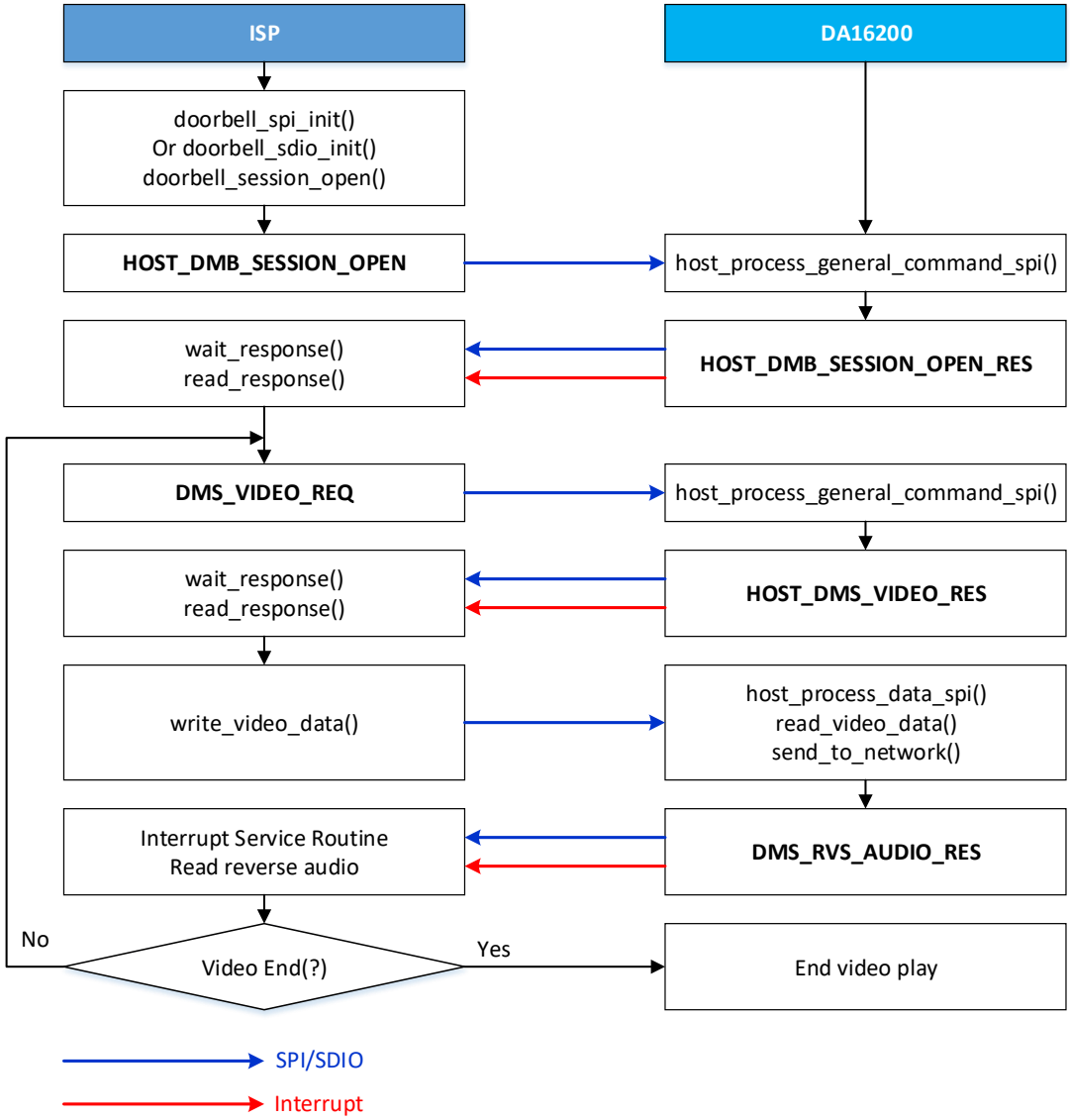

Figure 6: Interface Flow with ISP

Table 4: Interface Command for Request and Response

Command Request (ISP)	DA16200 Response
HOST_DMS_SESSION_OPEN	HOST_DMS_SESSION_OPEN_RES
DMS_VIDEO_REQ	HOST_DMS_VIDEO_RES
DMS_AUDIO_REQ	HOST_DMS_AUDIO_RES
	DMS_RVS_AUDIO_RES

Revision History

Revision	Date	Description
1.5	28-Mar-2022	Update logo, disclaimer, copyright.
1.4	29-Nov-2021	Title was changed
1.3	18-Aug-2021	Applied changes to SDK folder hierarchy
1.2	26-Nov-2019	Finalized for publication
1.1	21-Nov-2019	Editorial review
1.0	02-Aug-2019	Preliminary DRAFT Release

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.

Important Notice and Disclaimer

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/