User Manual

DA16200 DA16600 FreeRTOS SDK Programmer Guide

UM-WI-046

Abstract

The DA16200 (DA16600) is a highly integrated ultra-low power Wi-Fi system on chip (SoC) that allows users to develop the Wi-Fi solution on a single chip. This document is an SDK guide document intended for developers who want to program using the DA16200 (DA16600) chipset and describes the SDK API and peripheral device drivers and interfaces.

Contents

Figures

Tables

1 References

[1] DA16200, Datasheet, Dialog Semiconductor

[2] DA16200 FreeRTOS, EVK User Manual, User Manual, Dialog Semiconductor

[3] DA16200 FreeRTOS, Example Application Manual, User Manual, Dialog Semiconductor

[4] DA16200 DA16600 FreeRTOS, Getting Started Guide, User Manual, Dialog Semiconductor

[5] DA16200, Provisioning the Mobile App, Dialog Semiconductor

2 Introduction

The DA16200 (DA16600) is a highly integrated ultra-low power Wi-Fi system on chip (SoC) that allows users to develop the Wi-Fi solution on a single chip. The user implements the application with the DA16200 (DA16600) SDK, the compile environment is GNU Eclipse IDE system.

2.1 Overview

The DA16200 (DA16600) FreeRTOS SDK has seven folders:

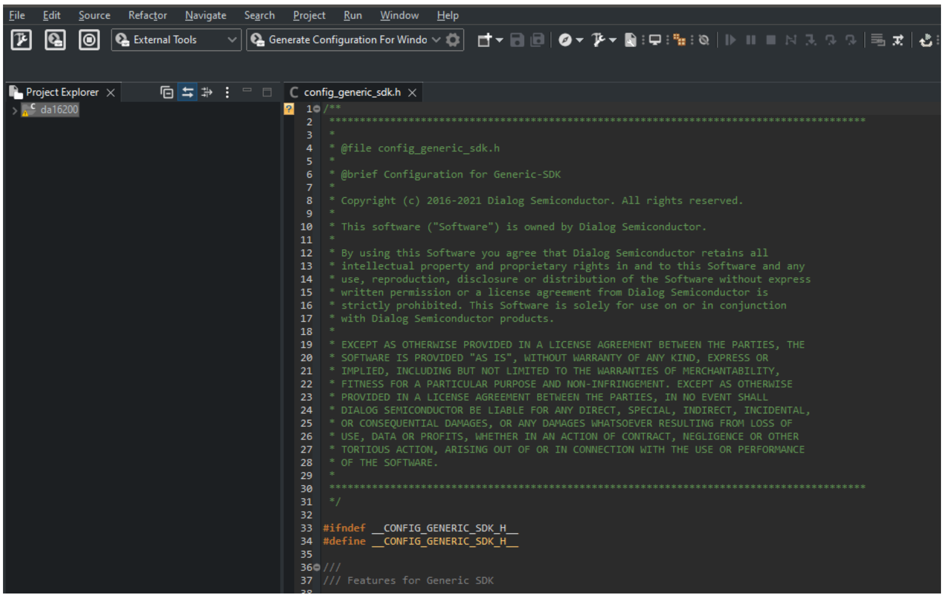
- **apps** : Project files, source codes for feature configurations
  - **apps/common/examples**: to demonstrate common use cases of what the DA16200 (DA16600) SDK provides
  - **apps/da16200/get_started/img**: to which the images built / pre-compiled are copied
- **core** : source codes
- **docs** : user documents (user guides, programmer guides, etc.)
- **library** : to which the pre-compiled lib files (.a) are saved
- **tools** : build tools/ scripts, temporary build artifacts, or environment files
- **utility** : utility for sample, eclipse and j-link
- **version** : version files to include when Image created

The DA16200 (DA16600) SDK may be provided with different features per customer or per certain applications and Customer/Developer can change the features easily in SDK.

All generic features are defined in *~/FreeRTOS_SDK/apps/da16200/get_started/include/user_main/config_generic_sdk.h* (the file name may follow its reference type) where users can enable/disable some features. And detailed features are defined in *~/FreeRTOS_SDK/apps/da16200/get_started/include/user_main/sys_common_features.h*

NOTE
Not all features can be freely enabled/disabled. This depends on the pre-compiled libraries included in the SDK package. Ask Dialog Semiconductor for more details.

The typical Eclipse project for the DA16200 (DA16600) SDK is shown in Figure 1. There is the possibility to add new user application files to the existing projects or create your own project.



```
/**
 *****************************************************************************
 *
 * @file config_generic_sdk.h
 *
 * @brief Configuration for Generic-SDK
 *
 * Copyright (c) 2016-2021 Dialog Semiconductor. All rights reserved.
 *
 * This software ("Software") is owned by Dialog Semiconductor.
 *
 * By using this Software you agree that Dialog Semiconductor retains all
 * intellectual property and proprietary rights in and to this Software and any
 * use, reproduction, disclosure or distribution of the Software without express
 * written permission or a license agreement from Dialog Semiconductor is
 * strictly prohibited. This Software is solely for use on or in conjunction
 * with Dialog Semiconductor products.
 *
 * EXCEPT AS OTHERWISE PROVIDED IN A LICENSE AGREEMENT BETWEEN THE PARTIES, THE
 * SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT. EXCEPT AS OTHERWISE
 * PROVIDED IN A LICENSE AGREEMENT BETWEEN THE PARTIES, IN NO EVENT SHALL
 * DIALOG SEMICONDUCTOR BE LIABLE FOR ANY DIRECT, SPECIAL, INDIRECT, INCIDENTAL,
 * OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF
 * USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER
 * TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE
 * OF THE SOFTWARE.
 *
 *****************************************************************************
 */

#ifndef __CONFIG_GENERIC_SDK_H__
#define __CONFIG_GENERIC_SDK_H__

///
/// Features for Generic SDK
```

Figure 1: Eclipse Project Configuration

2.2 Development Environment

The DA16200 (DA16600) FreeRTOS SDK needs Eclipse IDE environment. See Ref. [4] for Eclipse installation and Getting Started Guide.

2.3 Startup Main()

After system reboot, the system library invokes function `main()`. The following steps are run:

- Initialize HW resources (PIN_MUX, RTC, Console …)
- Start function system_start() to run the DA16200 (DA16600) as Wi-Fi IoT device

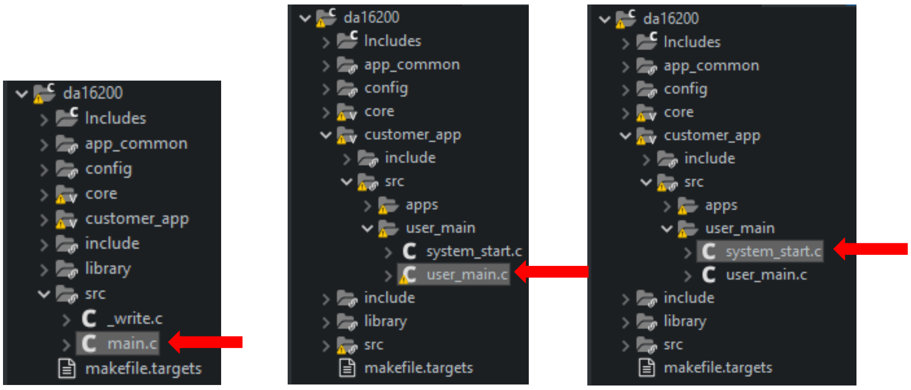


Figure 2: Startup Files on DA16200 (DA16600) Project

```
[ ~/FreeRTOS_SDK/core/main/src/main.c ]
 int main(char init_state)
 {
      ...
      xTaskCreate(system_launcher,
                           "system_launcher",
                           256*3,                      // for SecureBoot
                           (void *)NULL,
                           (tskIDLE_PRIORITY+1),
                           NULL);
      ...
      vTaskStartScheduler();
 }

 void system_launcher( void *pvParameters)
 {
      ...
      // Initialize and run system application
      // and run user application if needed.
      start_da16x();
      ...
 }

 static void start_da16x(void)
 {
      ...
      /* Configure Pin-Mux of DA16200 */
      config_pin_mux();

      /* Initialize WLAN interface */
      wlaninit();

      /* Start DA16200 IoT system layer */
      user_main(ramlib_ptim_init_status);      // USER main
 }
```

The following system initialization is done before applications start:

- Configure H/W and S/W features
- Configure system resources for system clock and TX power
- Initialize Wi-Fi function in `wlaninit()`

```
int user_main(char init_state)
{
    ...
    /* Entry point for customer main */
    if (init_state == pdTRUE) {
        system_start();
    } else {
        Printf("\nFailed to initialize the RamLib or pTIM !!!\n");
    }

    return status;
}
```

After the basic HW resources are initialized, function *system_start()* is called to run system/user applications. The following happens:

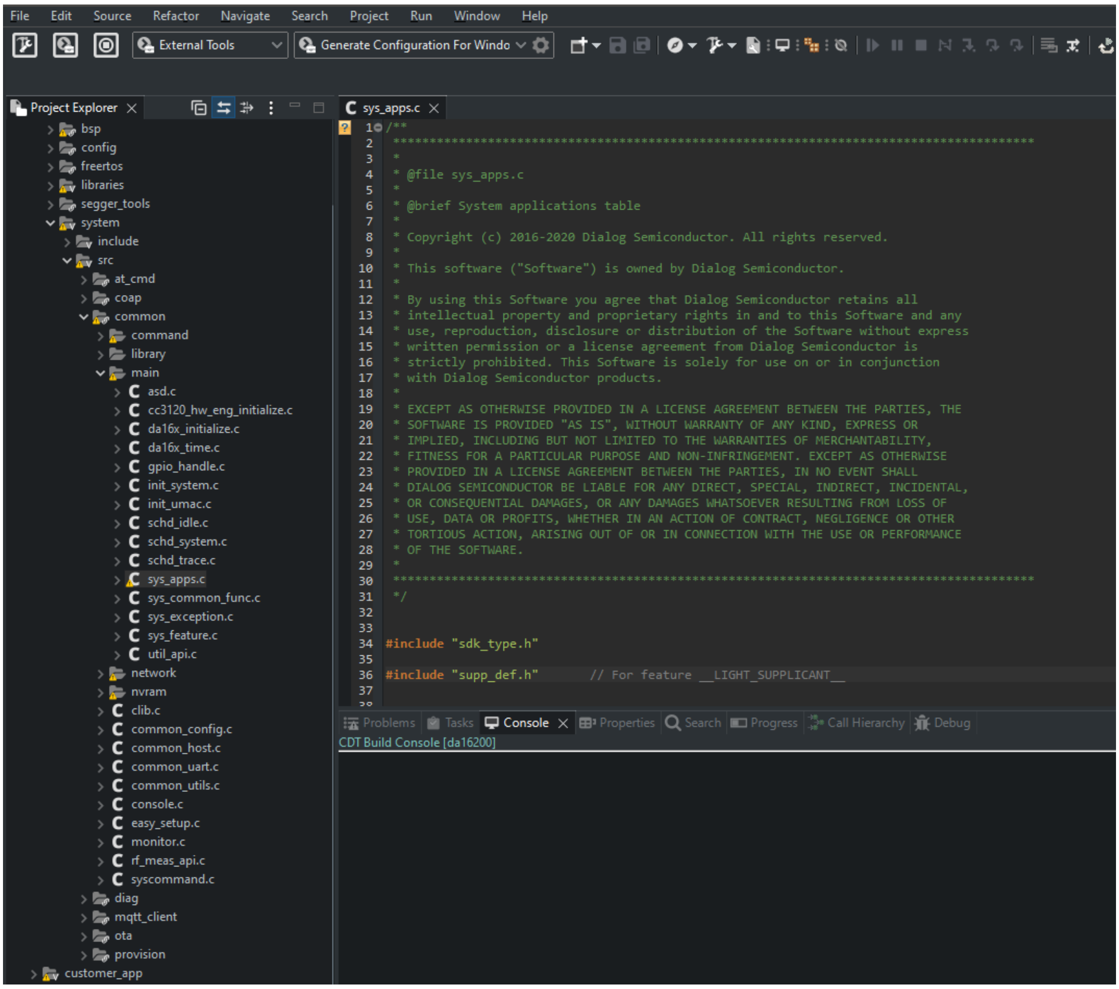
- Start of system-provided applications in start_sys_apps()
- Start of user applications in start_user_apps()

```
[~/FreeRTOS_SDK/apps/da16200/get_started/src/user_main/system_start.c ]
int system_start(void)
{
/* Config HW wakeup resource */
      config_user_wu_hw_resource();

      /* Set configuration for H/W button */
      config_gpio_button();

      /* Set paramters for system running */
      set_sys_config();

      /* Initialize WLAN interface */
      wlaninit();

... ...

      /* Start system applications for DA16XXX */
      start_sys_apps();

      /*
       * Entry point of user's applications
       *     : defined in user_apps_table.c
       */
      /* Start system applications for DA16XXX */
      start_user_apps();
}
```

NOTE
The features supported in the SDK are defined in file *config_xxxx_sdk.h* (namely *config_generic_sdk.h*) and all features of *config_xxx_sdk.h* can be enabled/disabled freely. If the user wants to change more detail features to handle delicate operations, some features in file *sys_common_feature.h* can be changed, but that requires the support from a support engineer of Dialog Semiconductor.

2.4 Startup System Applications

After running the main function, the DA16200 (DA16600) SDK runs some system provided applications and user-written applications. Each system application is started by the customer’s define features.



```
/**
 ****************************************************************************************
 *
 * @file sys_apps.c
 *
 * @brief System applications table
 *
 * Copyright (c) 2016-2020 Dialog Semiconductor. All rights reserved.
 *
 * This software ("Software") is owned by Dialog Semiconductor.
 *
 * By using this Software you agree that Dialog Semiconductor retains all
 * intellectual property and proprietary rights in and to this Software and any
 * use, reproduction, disclosure or distribution of the Software without express
 * written permission or a license agreement from Dialog Semiconductor is
 * strictly prohibited. This Software is solely for use on or in conjunction
 * with Dialog Semiconductor products.
 *
 * EXCEPT AS OTHERWISE PROVIDED IN A LICENSE AGREEMENT BETWEEN THE PARTIES, THE
 * SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT. EXCEPT AS OTHERWISE
 * PROVIDED IN A LICENSE AGREEMENT BETWEEN THE PARTIES, IN NO EVENT SHALL
 * DIALOG SEMICONDUCTOR BE LIABLE FOR ANY DIRECT, SPECIAL, INDIRECT, INCIDENTAL,
 * OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF
 * USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER
 * TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE
 * OF THE SOFTWARE.
 *
 ****************************************************************************************
 */


#include "sdk_type.h"

#include "supp_def.h"       // For feature __LIGHT_SUPPLICANT__

```

Figure 3: Applications on Eclipse Project

```
[ ~/FreeRTOS_SDK/core/system/src/common/main/sys_apps.c ]
void start_sys_apps(void)
{
... ...

    /* Start user application functions */
    run_sys_apps();
}
```

The system applications run in two parts:

- Applications that should be executed regardless of network settings
- Application that should be executed after the network setting is completed

```
static void run_sys_apps(void)
{
    ... ...

    /* Create network independent apps */
    create_sys_apps(sysmode, FALSE);

/* Create user's network independent apps */    create_user_apps(sysmode, FALSE);

    ...

      /* wait for network initialization */
      while (1) {
            if (check_net_init(iface) == pdPASS) {
                   i = 0;
                   break;
            }

            i++;

            vTaskDelay(1);
      }
...

      /* Check IP address resolution status */
      while (check_net_ip_status(iface)) {
            vTaskDelay(1);
      }

    /* Create network apps */
    create_sys_apps(sysmode, TRUE);
}
```

All system applications are provided in the sys_apps_table[] as shown in the example code below:

```
[ ~/FreeRTOS_SDK/core/system/src/common/main/sys_apps.c ]
static const app_task_info_t  sys_apps_table[] =
{
/* name, entry_func, stack_size, priority, timeslice, net_chk_flag, dpm_flag,
port_no, run_sys_mode */

    /****** For function features ***********************************/

… …

#if defined ( __SUPPORT_MQTT__ )
      { APP_MQTT_SUB,                   mqtt_auto_start,
      320,    (tskIDLE_PRIORITY + 7), TRUE,      TRUE,  UNDEF_PORT,
      RUN_STA_MODE  },
#endif      // __SUPPORT_MQTT__

… …

    /******* End of List *********************************************/
    { NULL, NULL,    0, 0, FALSE, FALSE, UNDEF_PORT, 0    }

};
```

NOTE

The user does not need to modify the system application tables provided in the DA16200 (DA16600) SDK.

If the user does want to modify the system application table, then that is possible but only with the support of a Dialog Semiconductor Engineer.

2.5 Startup User Applications

After running the main function, the DA16200 (DA16600) SDK can run user-written applications.

The user applications also run in two parts:

- Applications that should be executed regardless of network settings

```
[ ~/FreeRTOS_SDK/core/system/src/common/main/sys_apps.c]
static void run_sys_apps(void)
{
    ... ...

    /* Start user's network independent applications */
    create_user_apps(sysmode, FALSE);
    ... ...

```

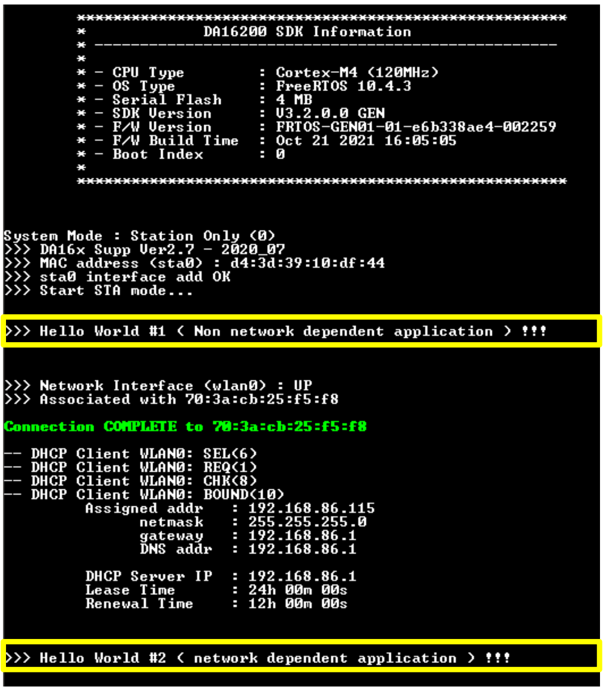
- Applications that should be executed after the network settings are completed

```
[ ~/FreeRTOS_SDK/core/system/src/common/main/sys_apps.c ]
void start_user_apps(void)
{
    int sysmode;
    … …

    /* Run user's network dependent apps */
    create_user_apps(sysmode, TRUE);
}
```

All user applications can be written in the user_apps_table[ ] as shown in the example code below. The DA16200 (DA16600) SDK provides a “hello_world” application. (enable __SUPPORT_HELLO_WORLD__ in *~/FreeRTOS_SDK/apps/da16200/get_started/include/user_main/config_generic_sdk.h* to use it).

```
[ ~/FreeRTOS_SDK/apps/da16200/get_started/src/apps/user_apps.c ]
const app_task_info_t user_apps_table[] = {
/* name, func, stack_size, pri, net_flag, dpm_flag, port_no, sys_mode */

#if defined (__SUPPORT_HELLO_WORLD__)
{ HELLO_WORLD_1,  customer_hello_world_1,          64,    (tskIDLE_PRIORITY +
2),   FALSE, FALSE,  UNDEF_PORT, RUN_ALL_MODE    },
  { HELLO_WORLD_2,  customer_hello_world_2,          64,    (tskIDLE_PRIORITY +
2),   TRUE,  FALSE,  UNDEF_PORT, RUN_ALL_MODE    },
#endif  // __SUPPORT_HELLO_WORLD__

    { NULL, NULL,   0, 0, FALSE, FALSE, UNDEF_PORT, 0   }

};
```

- `HELLO_WORLD_1` Not network-dependent, this application starts after system start
- `HELLO_WORLD_2` Network-dependent, this application starts after the Wi-Fi interface is up and running

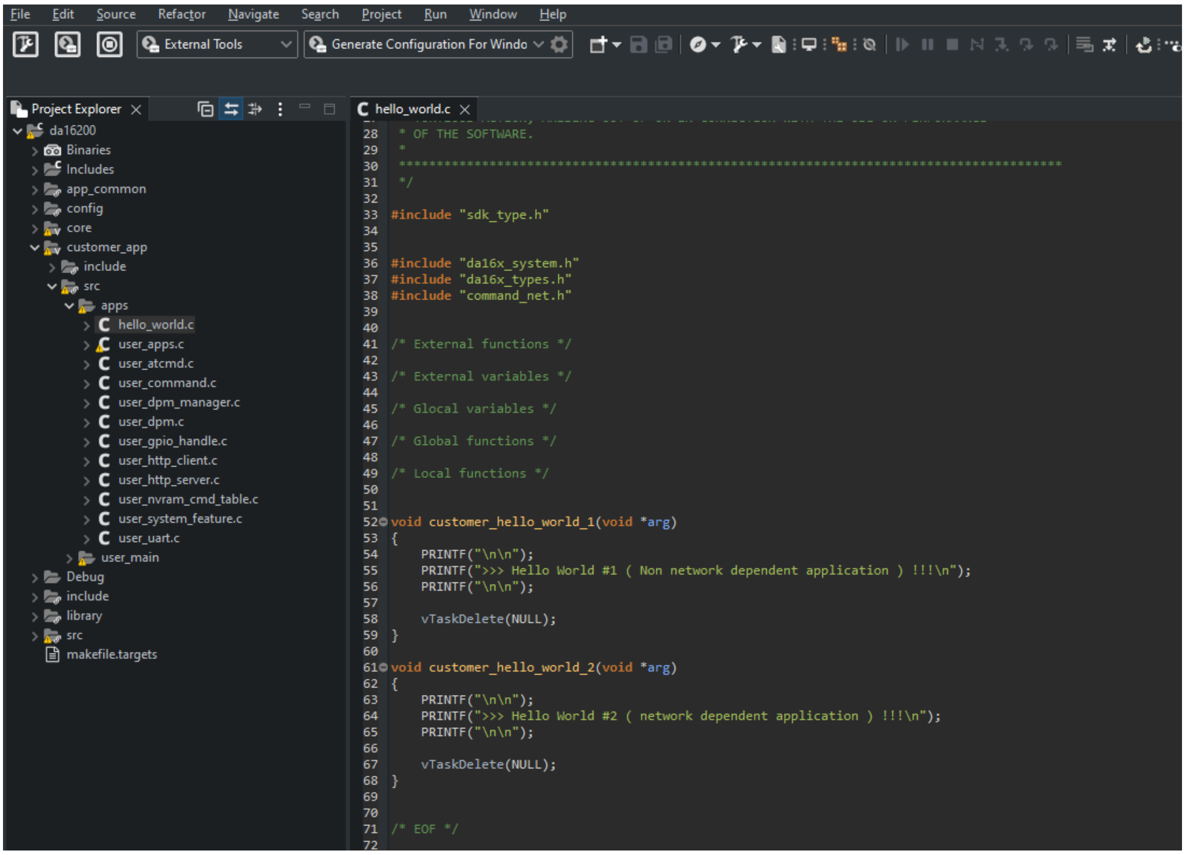


```
        ************************************************************
        *                  DA16200 SDK Information
        * ----------------------------------------------------------
        *
        * - CPU Type          : Cortex-M4 (120MHz)
        * - OS Type           : FreeRTOS 10.4.3
        * - Serial Flash      : 4 MB
        * - SDK Version       : V3.2.0.0 GEN
        * - F/W Version       : FRTOS-GEN01-01-e6b338ae4-002259
        * - F/W Build Time    : Oct 21 2021 16:05:05
        * - Boot Index        : 0
        *
        ************************************************************


System Mode : Station Only (0)
>>> DA16x Supp Ver2.7 - 2020_07
>>> MAC address (sta0) : d4:3d:39:10:df:44
>>> sta0 interface add OK
>>> Start STA mode...

>>> Hello World #1 ( Non network dependent application ) !!!


>>> Network Interface (wlan0) : UP
>>> Associated with 70:3a:cb:25:f5:f8

Connection COMPLETE to 70:3a:cb:25:f5:f8

-- DHCP Client WLAN0: SEL(6)
-- DHCP Client WLAN0: REQ(1)
-- DHCP Client WLAN0: CHK(8)
-- DHCP Client WLAN0: BOUND(10)
         Assigned addr   : 192.168.86.115
               netmask   : 255.255.255.0
               gateway   : 192.168.86.1
               DNS addr  : 192.168.86.1

         DHCP Server IP  : 192.168.86.1
         Lease Time      : 24h 00m 00s
         Renewal Time    : 12h 00m 00s


>>> Hello World #2 ( network dependent application ) !!!
```

Figure 4: Results of Running the ‘Hello World’ Applications

2.6 Write User Application

The user can add new application code in the folder *~/FreeRTOS_SDK/apps/da16200/get_started/src/apps* and can add a newly written application file in the project such as *hello_world.c*. See Figure 5.



```
 * OF THE SOFTWARE.
 *
 ******************************************************************************************
 */

#include "sdk_type.h"


#include "da16x_system.h"
#include "da16x_types.h"
#include "command_net.h"


/* External functions */

/* External variables */

/* Glocal variables */

/* Global functions */

/* Local functions */


void customer_hello_world_1(void *arg)
{
    PRINTF("\n\n");
    PRINTF(">>> Hello World #1 ( Non network dependent application ) !!!\n");
    PRINTF("\n\n");

    vTaskDelete(NULL);
}

void customer_hello_world_2(void *arg)
{
    PRINTF("\n\n");
    PRINTF(">>> Hello World #2 ( network dependent application ) !!!\n");
    PRINTF("\n\n");

    vTaskDelete(NULL);
}


/* EOF */
```

Figure 5: Customer Project in Eclipse IDE

The DA16200 (DA16600) SDK provides an interface to add a user application. The interface is designed to create a user thread. For this purpose, define your application with this interface and then a user thread is automatically created and run when the DA16200 (DA16600) starts.

The structure of the application thread information is as shown in the example code below:

```
[ ~/FreeRTOS_SDK/apps/da16200/get_started/include/apps/application.h ]
typedef struct  _app_task_info {
      /// Thread Name
      char   *name;

      /// Funtion Entry_point
      VOID   (*entry_func)(ULONG);

      /// Thread Stack Size
      USHORT stksize;

      /// Thread Priority
      USHORT priority;
```

```
        /// Flag to check network initializing
        UCHAR  net_chk_flag;

        /// Usage flag for DPM running
        UCHAR  dpm_flag;

        /// Port number for network communitation
        USHORT port_no;

        /// Running mode of DA16xxx
        int    run_sys_mode;
  } app_thread_info_t;
```

- `name` Unique thread name
- `entry_func` Thread entry point
- `stksize` Stack size of thread
- `priority` Thread running priority
- `net_chk_flag` [DA16200 (DA16600) feature] Indicate if the software must wait until the network interface is up and running before the user thread runs. If set to 1, the user thread waits until the network interface is up and running. You must set the value to 1 if your program is a network application
- `dpm_flag` [DA16200 (DA16600) feature] *[To Be Used Later When DPM feature is enabled – TBU_DPM]* Indicate if the user thread uses the DPM function
- `port_no` [DA16200 (DA16600) feature] *[TBU_DPM]* Data transfer port number for DPM mode. When a user thread has UDP/TCP operation with a specific port number, this port number should be registered to distinguish the data in DPM mode. This port number should be unique in the user thread table
- `run_sys_mode` [DA16200 (DA16600) feature] Runs a Wi-Fi mode (STA / Soft-AP). The application runs only the specified Wi-Fi mode

To add user application code in the DA16200 (DA16600) SDK:

1. Write new user code files and put the files in the customer folder. For example, *hello_world.c*.

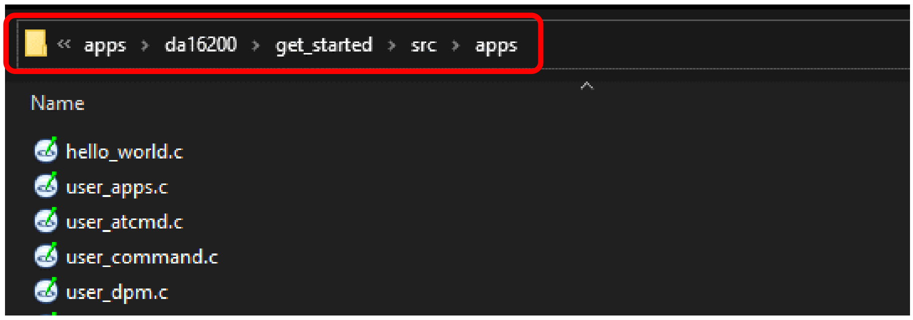


Figure 6: Location of User Codes

2. Add the written user code files to the Eclipse project. See Figure 7.

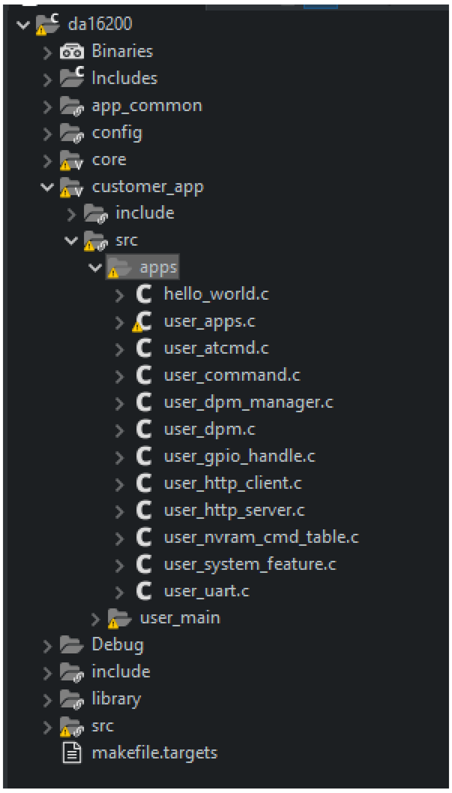


Figure 7: Add User Files to the Eclipse Project

2.7 SDK Compilation

After an application is written, right-click on the project **DA16200 (DA16600)**, and then click **Build Project**. If you compile for the first time, then the advice is to run command `Clean` first. See Figure 8.

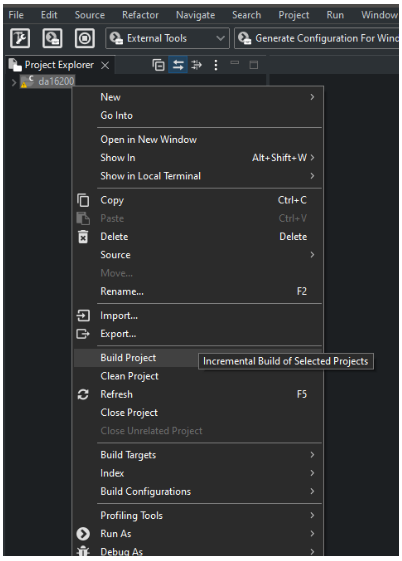
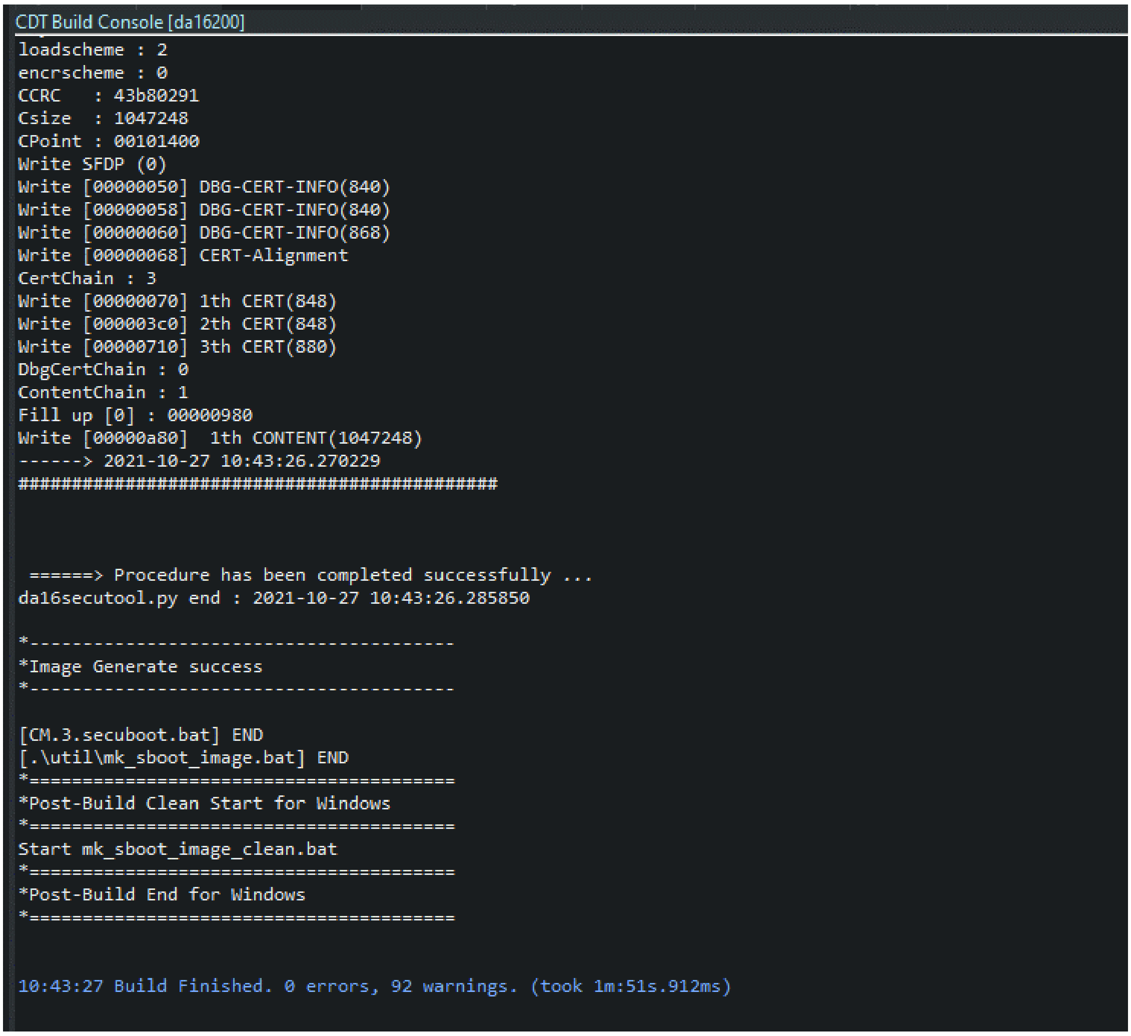


Figure 8: Compile SDK on Eclipse IDE



CDT Build Console [da16200]

```
loadscheme : 2
encrscheme : 0
CCRC   : 43b80291
Csize  : 1047248
CPoint : 00101400
Write SFDP (0)
Write [00000050] DBG-CERT-INFO(840)
Write [00000058] DBG-CERT-INFO(840)
Write [00000060] DBG-CERT-INFO(868)
Write [00000068] CERT-Alignment
CertChain : 3
Write [00000070] 1th CERT(848)
Write [000003c0] 2th CERT(848)
Write [00000710] 3th CERT(880)
DbgCertChain : 0
ContentChain : 1
Fill up [0] : 00000980
Write [00000a80]  1th CONTENT(1047248)
------> 2021-10-27 10:43:26.270229
#############################################



 ======> Procedure has been completed successfully ...
da16secutool.py end : 2021-10-27 10:43:26.285850

*----------------------------------------
*Image Generate success
*----------------------------------------

[CM.3.secuboot.bat] END
[.\util\mk_sboot_image.bat] END
*========================================
*Post-Build Clean Start for Windows
*========================================
Start mk_sboot_image_clean.bat
*========================================
*Post-Build End for Windows
*========================================


10:43:27 Build Finished. 0 errors, 92 warnings. (took 1m:51s.912ms)
```

Figure 9: Build Success on Eclipse IDE

If the build is successful, then there are two binary images created in folder *~/FreeRTOS_SDK/apps/da16200/get_started/img*. The names of the image files are:

- RTOS : DA16200_FRTOS-GEN01-01-XXXXXXXXX-000000.img
- 2nd Bootloader : DA16200_FBOOT-GEN01-01-XXXXXXXXX-000000_W25Q32JW.img

  (In case of Winbond W25Q32JW SFLASH)

For more information about the firmware download, see Ref. [2].

2.8 Make fcCSP Low-Power RTOS Image

The DA16200 (DA16600) SDK provides a QFN-type RTOS SFLASH image file. After a compilation with the DA16200 (DA16600) SDK, the QFN-type RTOS image with filename **DA16200_FRTOS-GEN01-01-XXXXX-000000.img** is created in folder **~/SDK/apps/da16200/get_started/img/**.

To create a RTOS image for the fcCSP Low-Power chipset with the DA16200 (DA16600) SDK, change the files mentioned below, and then do the SDK Compilation instructions given in Section 2.7.

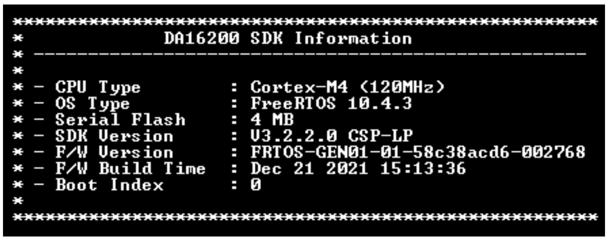
- binary file : ~/SDK/library/liblmac.a.fcCSP_LP

> ~/SDK/library/liblmac.a

- Compile feature : ~/SDK/apps/da16200/get_started/include/user_main/sys_common_features.h

#undef __FOR_FCCSP_SDK__ > #define __FOR_FCCSP_SDK__

After the compilation, load the RTOS image into the SFLASH and boot the system. To distinguish it from the QFN type, it shows SDK Version information as "V3.2.X.0 CSP LP" when booting. See Figure 10.



```
*******************************************************
*             DA16200 SDK Information
* ---------------------------------------------------
*
* - CPU Type          : Cortex-M4 (120MHz)
* - OS Type           : FreeRTOS 10.4.3
* - Serial Flash      : 4 MB
* - SDK Version       : V3.2.2.0 CSP-LP
* - F/W Version       : FRTOS-GEN01-01-58c38acd6-002768
* - F/W Build Time    : Dec 21 2021 15:13:36
* - Boot Index        : 0
*
*******************************************************
```

Figure 10: Boot Logo with fcCSP-LP RTOS Image

3 Memory Map

3.1 Memory Types

The DA16200 (DA16600) supports Mask ROM, Retention memory, SRAM, OTP, and Serial Flash memory. Mask ROM boots the system and starts the Main image. Retention memory is a special memory to preserve the contents when in power save mode. The DA16200 (DA16600) SoC contains 512 kB SRAM. OTP is used to store some permanent information and its size is 8 kB. A separate document is provided to use the OTP memory.

3.2 SRAM Memory Map

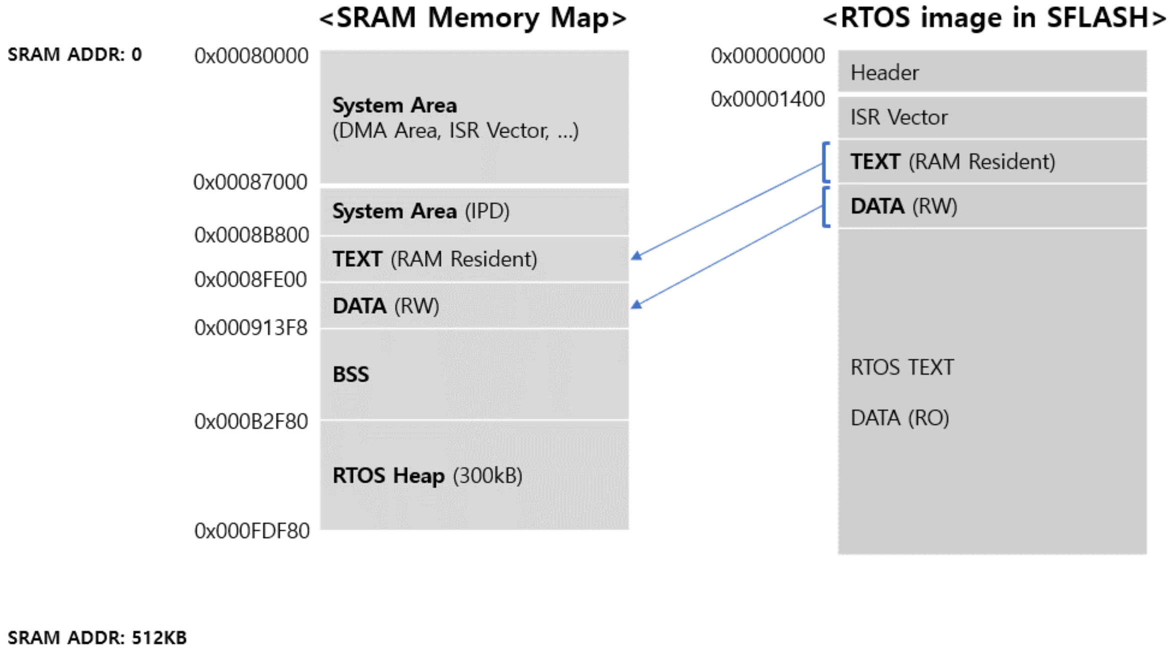
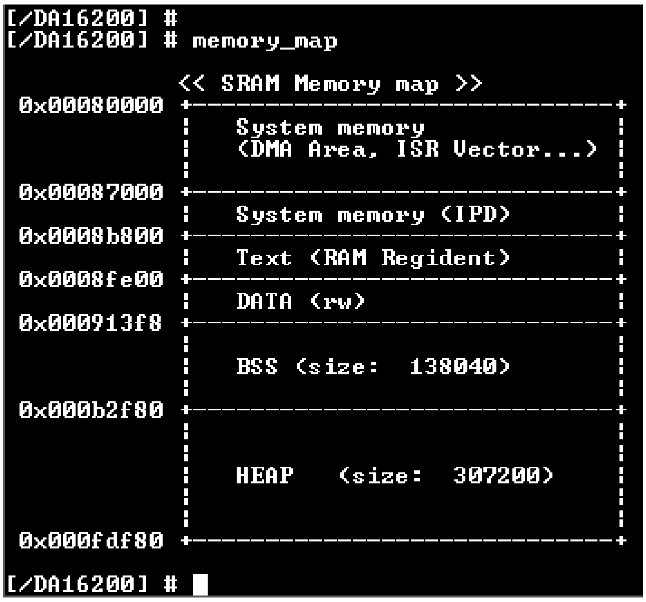


Figure 11: SRAM Memory Map

Figure 11 shows the SRAM memory map of the DA16200 (DA16600) FreeRTOS SDK. Depending on the decrease or increase of DATA(RW) or BSS area, HEAP size may change. By default, SDK provides about 300 kB of HEAP for user applications.

The DA16200 (DA16600) supports XIP hence TEXT is directly run in cache area (Serial Flash), but some functions are copied and run in SRAM. See TEXT (RAM Resident).

To get the current memory map info, type in the command in Figure 12.



```
[/DA16200] #
[/DA16200] # memory_map

            << SRAM Memory map >>
 0x00080000 +------------------------------+
            |    System memory             |
            |    (DMA Area, ISR Vector...) |
            |                              |
 0x00087000 +------------------------------+
            |    System memory (IPD)       |
 0x0008b800 +------------------------------+
            |    Text (RAM Regident)       |
 0x0008fe00 +------------------------------+
            |    DATA (rw)                 |
 0x000913f8 +------------------------------+
            |                              |
            |    BSS (size:  138040)       |
            |                              |
 0x000b2f80 +------------------------------+
            |                              |
            |                              |
            |    HEAP   (size:  307200)    |
            |                              |
            |                              |
 0x000fdf80 +------------------------------+

[/DA16200] #
```

Figure 12: memory_map Command

The end address shown as 0x000F_DF80 can change depending on the variable size of DATA/BSS but less than 0x0010_0000 (512 kB).

3.3 Serial Flash Memory Map

The DA16200 (DA16600) FreeRTOS SDK supports 4 MB SFLASH only.

Table 1: 4 MB SFLASH Map

Address	Name		Size (byte)
0x0000_0000	Bootloader		139,264
0x0002_2000	Boot Index		4,096
0x0002_3000	RTOS #0		1,835,008
0x001E_2000	RTOS #1		1,835,008
0x003A_2000	Debug / RMA Certificate		4,096
0x003A_3000	TLS Certificate #0 (MQTT)	CA	16,384
0x003A_4000		Cert	
0x003A_5000		Private key	
0x003A_6000		DH	
0x003A_7000	TLS Certificate #1 (HTTPS / OTA)	CA	16,384
0x003A_8000		Cert	
0x003A_9000		Private key	
0x003A_A000		DH	
0x003A_B000	NVRAM #0		4,096
0x003A_C000	NVRAM #1		4,096
0x003A_D000	User Area		335,872

4 Peripheral Driver

NOTE
This document may be further updated with more detailed descriptions later when the DA16200 (DA16600) SLR SoC is available.

4.1 SPI Slave

4.1.1 Introduction

The SPI slave interface enables support to control the DA16200 (DA16600) from an external host. The range of the SPI clock speed is the same as that of the internal bus clock speed. The SPI slave supports both burst mode and non-burst mode. In the burst mode, SPI_CSB remains active from the start to the end of communication. In the non-burst mode, SPI_CLK remains active at every 8-bit.

The communication protocols of the SPI slave interface use either 4-byte or 8-byte control signals. Between the two available communication protocols, the CPU chooses one before initiating the control.

Table 2: SPI Interface API Elements

Pin Name	Pin Number		I/O	Function Name
	QFN	fcCSP		
GPIOA2	37	B2	I	SPI_CSB
GPIOA6	32	E3	I	
F_CSN	18	J5	I	
GPIOA3	36	D4	I	SPI_CLK
GPIOA7	31	E1	I	
F_CLK	19	K4	I	
GPIOA1	38	C3	I	SPI_MOSI
GPIOA9	29	H2	I	
GPIOA11	27	G1	I	
F_IO0	14	K8	I	
GPIOA0	39	A3	O	SPI_MISO
GPIOA8	30	G3	O	
GPIOA10	28	F2	O	
F_IO1	15	L7	O	

4.1.2 Application Programming Interface

Table 3: SPI Slave Interface API Elements

void host_spi_slave_init(void)
Change Slave I/F to SPI protocol. Enable clock to SPI slave device and GPIO Interrupt Set
void host_i2c_slave_init(void)
Change Slave I/F to I2C protocol. Enable clock to I2C slave device and GPIO Interrupt Set

4.1.3 Sample Code

See Ref. [3].

4.2 SDIO Master

4.2.1 SDIO Introduction

Secure Digital Input Output (SDIO) is a full/high speed card suitable for memory card and I/O card applications with low power consumption. The full/high speed card supports SPI, 1-bit SD and 4-bit SD transfer modes at the full clock range of 0~50 MHz. To be compatible with the serviceable SDIO clock, the internal BUS clock should be set to a minimum of 50 MHz. The CIS and CSA area are inside the internal memory and the SDIO registers (CCCR and FBR) are programmed by the SD host.

For more details, see Ref. [1].

4.2.2 Application Programming Interface

Table 4: SDIO Interface API Elements

HANDLE EMMC_CREATE(void);		
Parameter	void	Void
Return		If succeeded return handle for such device, if failed return NULL
Function create handle. If memory allocation failed, return NULL		
int EMMC_INIT(HANDLE handler)		
Parameter	handler	Device handle
Return		If succeeded return ERR_NONE, if failed return ERR_MMC_INIT
Initialize the SD/eMMC or SDIO card If the function returns ERR_NONE, the card information is saved in the handle		
int EMMC_CLOSE(HANDLE handler)		
Parameter	handler	Device handle
Return		If succeeded return ERR_NONE
int SDIO_ENABLE_FUNC(HANDLE handler, UINT32 func_num)		
Parameter	handler	Device handle
	func_num	Function number to enable
Return		If succeeded return ERR_NONE

HANDLE EMMC_CREATE(void);		
int SDIO_DISABLE_FUNC(HANDLE handler, UINT32 func_num)		
Parameter	handler	Device handle
	func_num	Function number to disable
Return		If succeeded return ERR_NONE
int SDIO_SET_BLOCK_SIZE(HANDLE handler, UINT32 func_num, UINT32 blk_size)		
Parameter	handler	Device handle
	func_num	Function number
	blk_size	Block size
Return		If succeeded return ERR_NONE

int SDIO_READ_BYTE(HANDLE handler, UINT32 func_num, UINT32 addr, UINT8 *data)		
Parameter	handler	Device handle
	func_num	Function number
	addr	Address in the function
	data	Data pointer
Return		If succeeded return ERR_NONE. And byte data is stored in `data`
int SDIO_WRITE_BYTE(HANDLE handler, UINT32 func_num, UINT32 addr, UINT8 *data)		
Parameter	handler	Device handle
	func_num	Function number
	addr	Address in the function
	data	Data pointer
Return		If succeeded return ERR_NONE
int SDIO_READ_BURST(HANDLE handler, UINT32 func_num, UINT32 addr, UINT32 incr_addr, UINT8 *data, UINT32 count, UINT32 blksz)		
Parameter	handler	Device handle
	func_num	Function number
	addr	Function address
	Incr_addr	Increase address option (1: address increase, 0: address fix)
	data	Data pointer
	count	Count of blocks
	blksz	Block size
Return		If succeeded return ERR_NONE. If failed, Error Code return, see also EMMC.h
int SDIO_WRITE_BURST(HANDLE handler, UINT32 func_num, UINT32 addr, UINT32 incr_addr, UINT8 *data, UINT32 count, UINT32 blksz)		
Parameter	handler	Device handle

int SDIO_READ_BYTE(HANDLE handler, UINT32 func_num, UINT32 addr, UINT8 *data)		
	func_num	Function number
	addr	Function address
	Incr_addr	Increase address option (1: address increase, 0: address fix)
	data	Data pointer
	count	Count of blocks
	blksz	Block size
Return		If succeeded return ERR_NONE

4.2.3 Example Code

See Ref. [3].

4.3 SDIO Slave

4.3.1 Introduction

The GPIO4 and GPIO5 pins are set to SDIO CMD and CLK by default. If SDIO initialization is done and SDIO communication is enabled, then the SDIO data pin setting is done automatically. In other words, when the SDIO communication is detected, the pin used as the SDIO data among the GPIO pins is automatically activated in the SDIO use mode. However, the auto setting function is not supported for the F_xx pin used as the flash function.

Table 5: SDIO Slave Pin Configuration

Pin Name	Pin Number		I/O	Function Name
	QFN	fcCSP		
GPIOA4	34	F4	I/O	SDIO_CMD
F_CSN	18	J5	I/O	
GPIOA5	33	D2	I	SDIO_CLK
F_CLK	19	K4	I	
GPIOA9	29	H2	I/O	SDIO_D0
F_IO0	14	K8	I/O	
GPIOA8	30	G3	I/O	SDIO_D1
F_IO1	15	L7	I/O	
GPIOA7	31	E1	I/O	SDIO_D2
F_IO2	16	J7	I/O	
GPIOA6	32	E3	I/O	SDIO_D3
F_IO3	17	K6	I/O	

For more details, see Ref. [1].

4.3.2 Application Programmer Interface

Table 6: SDIO Interface API Elements

UINT32 SDIO_SLAVE_INIT(void)		
Parameter	Void	Void
Return		return 0
Description		SDIO Slave Initialization
void SDIO_SLAVE_CALLBACK_REGISTER(void (* p_rx_callback_func)(UINT32 status))		
Parameter	p_rx_callback_func	The callback function to use the offload protocol
Return		void
Description		SDIO Slave callback registration
void SDIO_SLAVE_CALLBACK_DEREGISTER(void)		
Parameter	void	void
Return		void
Description		SDIO Slave callback de-registration
void SDIO_SLAVE_DEINIT (void)		
Parameter	void	void
Return		void
Description		SDIO Slave de-initialization

4.3.3 Sample Code

See Ref. [3].

4.4 I2C

4.4.1 I2C Master

The DA16200 (DA16600) includes an I2C master module. There are two supportable clock speeds for I2C in the DA16200 (DA16600); standard is 100 kbps and fast mode is 400 kbps.

Table 7 shows the pin definition of the I2C master interface in GPIO Pin Configuration.

Table 7: I2C Master Pin Configuration

Pin Name	Pin Number		I/O	Function Name
	QFN	fcCSP		
GPIOA1	38	C3	O	I2C_CLK
GPIOA5	33	D2	O	
GPIOA9	29	H2	O	
GPIOA0	39	A3	I/O	I2C_SDA
GPIOA4	34	F4	I/O	
GPIOA8	32	G3	I/O	

For more details, see Ref. [1].

4.4.2 I2C Slave

The I2C slave interface gives support to control the DA16200 (DA16600) from an external host.

The pin mux condition is defined in Table 8. The I2C slave interface also supports the standard (100 kbps) or fast (400 kbps) transmission speeds.

Table 8: I2C Slave Pin Configuration

Pin Name	Pin Number		I/O	Function Name
	QFN	fcCSP		
GPIOA1	38	C3	I	I2C_CLK
GPIOA3	36	D4	I	
GPIOA5	33	D2	I	
GPIOA7	31	E1	I	
GPIOA0	39	A3	I/O	I2C_SDA
GPIOA2	37	B2	I/O	
GPIOA4	34	F4	I/O	
GPIOA6	32	E3	I/O	

For more details, see Ref. [1].

4.4.3 Application Programming Interface

Table 9: I2C Interface API Elements

HANDLE DRV_I2C_CREATE(UINT32 dev_id)		
Parameter	dev_id	Device ID number to create a handle
Return		If succeeded return handle for the device, if failed return NULL
Description		Create a handle with parameter "dev_id" designated
Int DRV_I2C_INIT(HANDLE handler)		
Parameter	handler	Device handle to initialize
Return		If succeeded return TRUE, if failed return FALSE
Description		
int DRV_I2C_IOCTL(HANDLE handler, UINT32 cmd, VOID *data)		
Parameter	handler	Device handle to control
	cmd	See <sys_i2c.h> in our SDK
	*data	Data pointer when there is any. If not, NULL
Return		If succeeded return TRUE, if failed return FALSE
Description		

int DRV_I2C_IOCTL(HANDLE handler, UINT32 cmd, VOID *data)		
I2C_GET_CONFIG	Get "i2c_cr0" Register Value. See Register Map	Read
I2C_GET_STATUS	Get "i2c_sr" Register Value. See Register Map	Read
I2C_SET_DMA_WR	I2C Write via uDMA Tx Enable / Disable	[TRUE / FALSE]
I2C_SET_DMA_RD	I2C READ via uDMA Rx Enable / Disable	[TRUE / FALSE]
I2C_GET_DMA_WR	Get uDMA Tx Enabled	[0x2 / FALSE]
I2C_GET_DMA_RD	Get uDMA Rx Enabled	[TRUE / FALSE]
I2C_SET_RESET	Set I2C Device Reset / set	[TRUE / FALSE]
I2C_SET_CHIPADDR	Set I2C Slave Device Address (8 bits)	Write
I2C_GET_CHIPADDR	Get I2C Slave Device Address (8 bits)	Read
I2C_SET_CLOCK	Set I2C Clock [KHz] (Max = 1200)	Write
int DRV_I2C_WRITE_DMA(HANDLE handler, VOID *p_data, UINT32 p_dlen, UINT32 dummy)		
Parameter	handler	Device handle to write with DMA
	*p_data	Buffer pointer to write
	p_dlen	Length to write
	dummy	Reserved (set to ‘0’)
Return		If succeeded return TRUE, if failed return FALSE
Description		I2C write function through DMA
int DRV_I2C_WRITE(HANDLE handler, VOID *p_data, UINT32 p_dlen, UINT32 stopen, UINT32 dummy)		
Parameter	handler	Device handle to write
	*p_data	Buffer pointer to write
	p_dlen	Length to read
	stopen	Flag stop bit enable
	dummy	Reserved (set to ‘0’)
Return		If succeeded return TRUE, if failed return FALSE
Description		I2C write function
int DRV_I2C_READ(HANDLE handler, VOID *p_data, UINT32 p_dlen, UINT32 addr_len,UINT32 dummy)		
Parameter	handler	Device handle to read
	*p_data	Buffer pointer to read
	p_dlen	Length to read
	addr_len	Length of register address inside of slave device. if 0, Read only operation
	dummy	Reserved (set to ‘0’)
Return		If succeeded return TRUE, if failed return FALSE
Description		I2C read function
Int DRV_I2C_CLOSE(HANDLE handler);		
Parameter	handler	Device handle to close
Return		If succeeded return TRUE, if failed return FALSE

int DRV_I2C_IOCTL(HANDLE handler, UINT32 cmd, VOID *data)		
Description		I2C driver close
void DRV_I2C_REGISTER_INTERRUPT (HANDLE handler);		
Parameter	handler	Device handle to register Interrupt Handler
Return		NULL
Description		I2C Interrupt Registration

4.4.4 Sample Code

See Ref. [3].

4.5 SD/eMMC

4.5.1 Introduction

The SD/eMMC host IP has a function for the DA16200 (DA16600) to access SD or eMMC cards. The maximum data rate is less than 100 Mbps. So, this SD/eMMC host IP only supports a 4-bit data bus and the maximum clock speed is 50 MHz. The maximum data rate is 25 MB/s (200 Mbps) under 4-bit data bus and 50 MHz clock speed. The SD/eMMC pin mux condition is defined in Table 10.

Table 10: SD/eMMC Master Pin Configuration

Pin Name	Pin Number		I/O	Function Name
	QFN	fcCSP		
GPIOA4	34	F4	I/O	SD/eMMC_CMD
GPIOA5	33	D2	O	SD/eMMC_CLK
GPIOA9	29	H2	I/O	SD/eMMC_D0
GPIOA8	30	G3	I/O	SD/eMMC_D1
GPIOA7	31	E1	I/O	SD/eMMC_D2
GPIOA6	32	E3	I/O	SD/eMMC_D3
GPIOA10	28	F2	I	SD/eMMC_WRP
GPIOA1	38	C3	I	

For more details, see Ref. [1].

4.5.2 Application Programming Interface

Table 11: SD/eMMC Interface API Elements

HANDLE EMMC_CREATE(void)		
Parameter	Void	Void
Return		If succeeded return handle for such device, if failed return NULL
Description		Function create handle. If memory allocation fails, return NULL
int EMMC_INIT(HANDLE handler)		
Parameter	handler	Device handle

HANDLE EMMC_CREATE(void)		
Return		If succeeded return ERR_NONE, if failed return ERR_MMC_INIT
Description		Initialize the SD/eMMC or SDIO card. If the function returns ERR_NONE, the card information is stored in the handle
int EMMC_READ(HANDLE handler, UINT32 dev_addr, VOID *p_data, UINT32 block_count)		
Parameter	handler	Device handle
	dev_addr	Address
	p_data	Data pointer
	block_count	Block counter for read
Return		If succeeded return ERR_NONE
Description		EMMC read command
int EMMC_WRITE(HANDLE handler, UINT32 dev_addr, VOID *p_data, UINT32 block_count)		
Parameter	handler	Device handle
	dev_addr	Address
	p_data	Data pointer
	block_count	Block counter for write
Return		If succeeded return ERR_NONE
Description		EMMC write command
void EMMC_SEND_CMD(HANDLE handler, UINT32 cmd, UINT32 cmd_arg)		
Parameter	handler	Device handle
	cmd	SDIO command without response. Defined in <SDIO.h>
	cmd_arg	SDIO command argument
Return		If succeeded return TRUE, if failed return FALSE
Description		
void EMMC_SEND_CMD_RES(HANDLE handler, UINT32 cmd, UINT32 cmd_arg, UINT32 *rsp)		
Parameter	handler	Device handle
	cmd	SDIO command with response
	cmd_arg	SDIO command argument
	rsp	Response pointer
Return		Void
Description		After this function call, the response is stored in `rsp`
int EMMC_IOCTL(HANDLE handler, UINT32 cmd, VOID *data)		
Parameter	handler	Device handle
	cmd	The command that is defined in EMMC.h
	data	Data pointer
Return		If succeeded return ERR_NONE
Description		EMMC IOCTL command

HANDLE EMMC_CREATE(void)		
int EMMC_CLOSE(HANDLE handler)		
Parameter	handler	Device handle
Return		If succeeded return ERR_NONE
Description		EMMC driver close command

4.5.3 Sample Code

See Ref. [3].

4.6 PWM

4.6.1 Introduction

Pulse-Width Modulation (PWM) is a modulation technique used to encode a message into a pulse signal. The blocks are designed to adjust the output pulse duration by means of the CPU bus clock (HCLK).

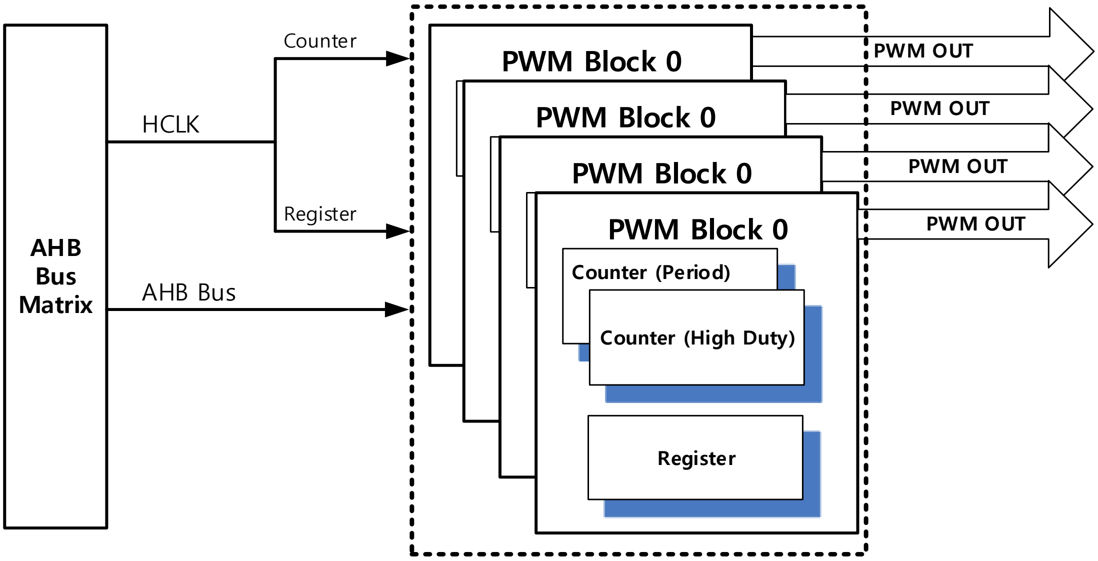


Figure 13: PWM Block Diagram

Table 12: PWM Pin Configuration

Pin Name	Pin Number	I/O	Pin Selection	Function Name
GPIOx		O	Reg. GPIO_SEL.xMUXx	PWM[3:0] output

For more details, see Ref. [1].

4.6.2 Application Programming Interface

Table 13: PWM Interface API Elements

HANDLE DRV_PWM_CREATE(UINT32 dev_id)		
Parameter	dev_id	Device number to create handle

HANDLE DRV_PWM_CREATE(UINT32 dev_id)		
Return		If succeeded return handle for such device, if failed return NULL
Description		Function create handle with parameter "dev_id" designated
int DRV_PWM_INITf(HANDLE handler)		
Parameter	handler	Device handle to initialize
Return		If succeeded return TRUE, if failed return FALSE
Description		Change GPIO multiplex to PWM mode
int DRV_PWM_START(HANDLE handler, UINT32 period_us, UINT32 hduty_percent, UINT32 dummy)		
Parameter	handler	Device handle to enable pwm device output
	Period_us	1 cycle period in microsecond
	Hduty_percent	Output high time in percentage while every 1 cycle
	dummy	TBD
Return		If succeeded return TRUE, if failed return FALSE

Description		Enable PWM block in the DA16200 (DA16600) with specified parameters
		`period = (((period_us * 10) * (clock / 1000000))/10)-1;`
		`// minimum system clock 1mhz`
		`hduty = (((period + 1) * hduty_percent) / 100)-1;`

int DRV_PWM_STOP(HANDLE handler, UINT32 dummy)		
Parameter	handler	Device handle to stop pwm out
	cmd	See <pwm.h> in our SDK
Return		If succeeded return TRUE, if failed return FALSE
Description		Disable PWM block in the DA16200 (DA16600)
int DRV_PWM_CLOSE(HANDLE handler)		
Parameter	handler	Device handle to close and de-initialize device
Return		If succeeded return TRUE, if failed return FALSE
Description		Destroy handle

4.6.3 Sample Code

See Ref. [3].

4.7 ADC

4.7.1 Introduction

The DA16200 (DA16600) has Analog-to-Digital Converters (ADC): a four-channel single-end ADC of 12-bit resolution. Analog input is measured by means of 4 pins from GPIO0 to GPIO3, and the pin selection is changed through the register setting. See Figure 14 and Table 14.

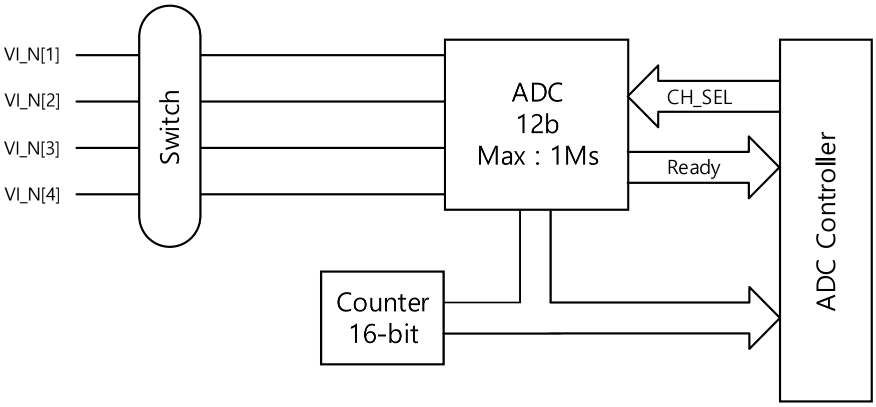


Figure 14: ADC Control Block Diagram

Table 14: AUX ADC Pin Configuration

Pin Name	Pin Number		I/O	Function Name
	QFN	fcCSP		
GPIOA3	36	D4	A	Analog signal
GPIOA2	37	B2	A	Analog signal
GPIOA1	38	C3	A	Analog signal
GPIOA0	39	A3	A	Analog signal

For more details, see Ref. [1].

4.7.2 Application Programming Interface

Table 15: ADC Interface API Elements

HANDLE DRV_ADC_CREATE(UINT32 dev_id)		
Parameter	dev_id	Device number to create a handle
Return		If succeeded return handle for such device, if failed return NULL
Description		Function create handle with parameter `dev_id` designated
int DRV_ADC_INIT(HANDLE handler, unsigned int use_timestamp)		
Parameter	handler	Device handle to initialize
Return		If succeeded return TRUE, if failed return FALSE
Description		ADC Initialization command
Int DRV_ADC_IOCTL(HANDLE handler, UINT32 cmd, VOID *data)		
Parameter	handler	N/A
	cmd	N/A
	data	N/A
Return		N/A
Description		ADC IOCTL command

HANDLE DRV_ADC_CREATE(UINT32 dev_id)		
int DRV_ADC_START(HANDLE handler, UINT32 divider12, UINT32 dummy)		
Parameter	handler	Device handle to start
	divider12	Fs = sys_clk / 15 / (div12 +1)
Return		If succeeded return TRUE, if failed return FALSE
Description		ADC start command
int DRV_ADC_STOP(HANDLE handler, UINT32 dummy)		
Parameter	handler	Device handle to stop
Return		If succeeded return TRUE, if failed return FALSE
Description		ADC stop command
Int DRV_ADC_CLOSE(HANDLE handler)		
Parameter	handler	Device handle to close
Return		If succeeded return TRUE, if failed return FALSE
Description		ADC driver close
int DRV_ADC_READ(HANDLE handler, UINT32 channel, UINT32 *data, UINT32 dummy)		
Parameter	handler	Device handle to read
	channel	Channel number to read instant ADC value
	*data	Buffer to read
Return		If succeeded return TRUE, if failed return FALSE
Description		ADC read command
int DRV_ADC_READ_DMA(HANDLE handler, UINT32 channel, UINT16 *p_data, UINT32 p_dlen, UINT32 dummy)		
Parameter	handler	Device handle to read with specified length
	channel	Channel number to read
	*p_data	Buffer block to read
	p_dlen	Number of samples to read with DMA, not buffer length
Return		If succeeded return TRUE, if failed return FALSE
Description		ADC read command through DMA
int DRV_ADC_ENABLE_CHANNEL(HANDLE handler, UINT32 channel, unsigned int sel_adc, UINT32 dummy)		
Parameter	handler	Device handle
	channel	Channel number to set ADC devices
	sel_adc	12: SMI 12B ADC, 0: disable
Return		If succeeded return TRUE, if failed return FALSE
Description		ADC channel enable command

HANDLE DRV_ADC_CREATE(UINT32 dev_id)		
int DRV_ADC_SET_INTERRUPT(HANDLE handler, UINT32 channel, UINT32 enable, UINT32 type, UINT32 dummy)		
Parameter	handler	Device handle
	channel	Channel number to set interrupt
	enable	1: enable interrupt, 0: disable interrupt
	type	ADC_INTERRUPT_FIFO_HALF (0) ADC_INTERRUPT_FIFO_FULL (1) ADC_INTERRUPT_THD_OVER (2) ADC_INTERRUPT_THD_UNDER (3) ADC_INTERRUPT_THD_DIFF (4) ADC_INTERRUPT_ALL (0xf)
Return		If succeeded return TRUE, if failed return FALSE
Description		ADC interrupt set command
int DRV_ADC_SET_THD_VALUE(HANDLE handler, UINT32 type, UINT32 enable, UINT32 thd, UINT32 dummy);		
Parameter	handler	Device handle
	type	ADC_THRESHOLD_TYPE_12B_OVER (0) ADC_THRESHOLD_TYPE_12B_UNDER (2) ADC_THRESHOLD_TYPE_12B_DIFF (4)
	thd	Interrupt threshold. 0 ~ 65535 range. Upper 12 bits of 16-bit data are valid values.
Return		If succeeded return TRUE, if failed return FALSE
Description		ADC interrupt threshold set command

int DRV_ADC_WAIT_INTERRUPT(HANDLE handler, UNSIGNED *mask_evt);		
Parameter	handler	Device handle

int DRV_ADC_WAIT_INTERRUPT(HANDLE handler, UNSIGNED *mask_evt);		
	*mask_evt	Mask for waiting interrupt bit[19] : Interrupt status for Threshold Difference of CHANNEL 3 bit[18] : Interrupt status for Threshold Difference of CHANNEL 2 bit[17] : Interrupt status for Threshold Difference of CHANNEL 1 bit[16] : Interrupt status for Threshold Difference of CHANNEL 0 bit[15] : Interrupt status for Threshold Under level of CHANNEL 3 bit[14] : Interrupt status for Threshold Under level of CHANNEL 2 bit[13] : Interrupt status for Threshold Under level of CHANNEL 1 bit[12] : Interrupt status for Threshold Under level of CHANNEL 0 bit[11] : Interrupt status for Threshold Over level of CHANNEL 3 bit[10] : Interrupt status for Threshold Over level of CHANNEL 2 bit[9] : Interrupt status for Threshold Over level of CHANNEL 1 bit[8] : Interrupt status for Threshold Over level of CHANNEL 0 bit[7] : Interrupt status for full level of CHANNEL 3 bit[6] : Interrupt status for full level of CHANNEL 2 bit[5] : Interrupt status for full level of CHANNEL 1 bit[4] : Interrupt status for full level of CHANNEL 0 bit[3] : Interrupt status for half level of CHANNEL 3 bit[2] : Interrupt status for half level of CHANNEL 2 bit[1] : Interrupt status for half level of CHANNEL 1 bit[0] : Interrupt status for half level of CHANNEL 0
Return		If receive masked interrupt return
Description		ADC interrupt wait command

4.7.3 Interrupt Description

ADC_INTERRUPT_FIFO_HALF: the interrupt that occurs when the FIFO Level is 4 or higher.

ADC_INTERRUPT_FIFO_FULL: the interrupt that occurs when FIFO Level is 8.

ADC_INTERRUPT_THD_OVER: this interrupt is issued when the current input value to the ADC device is greater than the value set in the "ADC_THRESHOLD_TYPE_12B_OVER" type.

ADC_INTERRUPT_THD_UNDER: this interrupt is issued when the current input value to the ADC device is smaller than the value set in the " ADC_THRESHOLD_TYPE_12B_UNDER " type.

ADC_INTERRUPT_THD_DIFF: this interrupt occurs when the difference between the current input value to the ADC device and the previously input value is greater than the value set in "ADC_INTERRUPT_THD_DIFF" type.

4.7.4 Sample Code

See Ref. [3].

4.8 GPIO

4.8.1 Introduction

All digital pads can be used as GPIO. Each GPIO port is mixed with a multi-functional interface. The GPIO features for this device are:

- Input or output lines in a programmable direction
- Word and half word read/write access
- Address-masked byte writes to facilitate quick bit set and clear operations
- Address-based byte reads to facilitate quick bit test operations
- Make a GPIO pin to an interrupt pin possible to be the output signal of PWM [3:0], external Interrupt, SPI_CSB [3:1], RF_SW [1:0] and UART_TXDOE [1:0] on any GPIO pin

It provides special functions for GPIO pin use. PWM [3:0], external interrupt, SPI_CSB [3:1], RF_SW [1:0] and UART_TXDOE [1:0] signals can be output if any of the unused pins among the GPIO pins are selected. It is possible to select the function to be output from the GPIO register setting and select the remaining GPIO pin and not output the specific function to any desired GPIO pin.

Table 16: GPIO Pin Configuration

Pin Name	Pin Number	I/O	Pin Selection	Function Name
GPIOA0	39	I/O	Reg. GPIO_SEL.AMUX9	GPIOA[0]
GPIOA1	38	I/O	Reg. GPIO_SEL.AMUX9	GPIOA[1]
GPIOA2	37	I/O	Reg. GPIO_SEL.BMUX9	GPIOA[2]
GPIOA3	36	I/O	Reg. GPIO_SEL.BMUX9	GPIOA[3]
GPIOA4	34	I/O	Reg. GPIO_SEL.CMUX9	GPIOA[4]
GPIOA5	33	I/O	Reg. GPIO_SEL.CMUX9	GPIOA[5]
GPIOA6	32	I/O	Reg. GPIO_SEL.DMUX9	GPIOA[6]
GPIOA7	31	I/O	Reg. GPIO_SEL.DMUX9	GPIOA[7]
GPIOA8	30	I/O	Reg. GPIO_SEL.EMUX9	GPIOA[8]
GPIOA9	29	I/O	Reg. GPIO_SEL.EMUX9	GPIOA[9]
GPIOA10	28	I/O	Reg. GPIO_SEL.FMUX7	GPIOA[10]
GPIOA11	27	I/O	Reg. GPIO_SEL.FMUX7	GPIOA[11]
GPIOC6	10	I/O	Reg. GPIO_SEL.UMUX2	GPIOC[6]
GPIOC7	9	I/O	Reg. GPIO_SEL.UMUX2	GPIOC[7]
GPIOC8	8	I/O	Reg. GPIO_SEL.UMUX2	GPIOC[8]

If you want to keep GPIO PIN state high or low in sleep state, you need to use one of the following API functions:

- "GPIO_RETAIN_HIGH"
- "GPIO_RETAIN_LOW"

Note that only for GPIOA[11:4], GPIOC[8:6] is possible to set GPIO retention high or low.

On how to use this API, see Ref. [3].

When using GPIO and GPIO Retention API, the status of GPIO PIN is shown in Table 17.

Table 17: The Status of GPIO PIN

	PIN info	Before sleep (RTOS booting)	Sleep period	Sleep period (with SAVE_PULLUP_PINS_INFO)	After sleep(wakeup)
GPIO input configured	GPIOA[3:0]	high-z	high-z	high-z	high-z
	GPIOA[11:8], GPIOC[8:6]	high-z	low(PD)	high-z	high-z
GPIO output high configured	GPIOA[3:0]	high	high-z	high-z	high-z
	GPIOA[11:8], GPIOC[8:6]	high	low(PD)	high-z	high-z
GPIO output low configured	GPIOA[3:0]	low	high-z	high-z	high-z
	GPIOA[11:8], GPIOC[8:6]		low(PD)	high-z	
GPIO retention high configured	GPIOA[11:8], GPIOC[8:6]	high	high	high	high
GPIO retention low configured	GPIOA[11:8], GPIOC[8:6]	low	low	low	low

If you want to keep GPIO PIN in high-z state in sleep period, you should use the API described in the next Section 4.8.2:

- "SAVE_PULLUP_PINS_INFO"

This function should be used when an external pull-up register is connected to a GPIO PIN. If this function is not used, leakage current may occur.

4.8.2 Application Programming Interface

Table 18: GPIO Interface API Elements

HANDLE GPIO_CREATE(UINT32 dev_type)		
Parameter	dev_type	Device index
Return		If succeeded, return handle for the device. If failed return NULL
Description		The DA16200 (DA16600) can set GPIO_UNIT_A and GPIO_UNIT_C
int GPIO_INIT (HANDLE handler)		
Parameter	handler	Device handle
Return		If succeeded return ERR_NONE
Description		Configure the GPIO setting
int GPIO_IOCTL(HANDLE handler, UINT32 cmd, VOID *data)		
Parameter	handler	Device handle
	cmd	Commands are defined <gpio.h> in our SDK
	data	Data pointer
Return		If succeeded return ERR_NONE

HANDLE GPIO_CREATE(UINT32 dev_type)		
Description		The necessary configuration of GPIO can be set with this function. Commands are as below: ● GPIO_GET_DEVREG = 1, ● GPIO_SET_OUTPUT, // set gpio as an output ● GPIO_SET_INPUT, // set gpio as an input ● GPIO_GET_DIRECTION, // get gpio direction ● GPIO_SET_INTR_MODE, // set gpio interrupt mode [edge/level] ● GPIO_GET_INTR_MODE, // get gpio interrupt mode ● GPIO_SET_INTR_ENABLE, // enable gpio interrupt ● GPIO_SET_INTR_DISABLE, // disable gpio interrupt ● GPIO_GET_INTR_ENABLE, // get gpio interrupt enable status ● GPIO_GET_INTR_STATUS, // get gpio interrupt pending status ● GPIO_SET_INTR_CLEAR, // clear gpio interrupt status ● GPIO_SET_CALLACK, // set a callback function for gpio interrupt
int GPIO_READ (HANDLE handler, UINT32 addr, UINT16 *pdata, UINT32 dlen)		
Parameter	handler	Device handle
	addr	gpio index
	p_data	Data buffer pointer
	p_dlen	Data buffer length
Return		If succeeded return ERR_NONE
Description		GPIO value contained in p_data

int GPIO_WRITE (HANDLE handler, UINT32 addr, VOID *p_data, UINT32 p_dlen)		
Parameter	handler	Device handle
	addr	gpio index
	p_data	Data buffer pointer
	p_dlen	Data buffer length
Return		If succeeded return ERR_NONE
Description		GPIO value contained in `p_data`
int GPIO_CLOSE(HANDLE handler)		
Parameter	handler	Device handle
Return		If succeeded return ERR_NONE
Description		GPIO close command
INT32 GPIO_GET_ALT_FUNC (HANDLE handler, GPIO_ALT_FUNC_TYPE altFuncType, UINT32 * regVal)		
Parameter	handler	Device handle
	altFuncType	GPIO alternate function type

int GPIO_WRITE (HANDLE handler, UINT32 addr, VOID *p_data, UINT32 p_dlen)		
	regVal	GPIO alternate function setting value
Return		If succeeded return 0
Description		Gets GPIO alternate function setting value
INT32 GPIO_SET_ALT_FUNC(HANDLE handler, GPIO_ALT_FUNC_TYPE altFuncType, GPIO_ALT_GPIO_NUM_TYPE gpioType)		
Parameter	handler	Device handle
	altFuncType	GPIO alternate function type
	gpioType	GPIO number
Return		If succeeded return 0
Description		Sets GPIO alternate function
INT32 _GPIO_RETAIN_HIGH(UINT32 gpio_port, UINT32 gpio_num)		
Parameter	gpio_port	GPIO port number
	gpio_num	GPIO pin number
Return		TRUE if successfully configured, else FALSE.
Description		Note that only for GPIOA[11:4], GPIOC[8:6] is possible to set GPIO retention high. And this API function should not be called from the “config_pin_mux” function
INT32 _GPIO_RETAIN_LOW(UINT32 gpio_port, UINT32 gpio_num)		
Parameter	gpio_port	GPIO port number
	gpio_num	GPIO pin number
Return		TRUE if successfully configured, else FALSE.
Description		Note that only for GPIOA[11:4], GPIOC[8:6] is possible to set GPIO retention high. And this API function should not be called from the “config_pin_mux” function
void SAVE_PULLUP_PINS_INFO(UINT32 port_num, UINT32 pinnum)		
Parameter	port_num	GPIO port number
	pinnum	GPIO pin number
Description		It keeps GPIO PIN in high-z state in sleep period This function should be used when an external pull-up register is connected to a GPIO PIN. If this function is not used, leakage current may occur.

4.8.3 Sample Code

See Ref. [3].

4.9 UART

4.9.1 Introduction

The DA16200 (DA16600) has two UARTs (Universal Asynchronous Receiver-Transmitter), which have the following features:

- Programmable use of UART
- Compliance to the AMBA AHB bus specification for easy integration into SoC implementation
- Supports both byte and word access for reduction of bus burden
- Supports both RS-232 and RS-485
- Separate 32x8 bit transmit and 32x12 bit receive FIFO memory buffers to reduce CPU interrupts
- Programmable FIFO disabling for 1-byte depth
- Programmable baud rate generator
- Standard asynchronous communication bits (start, stop and parity). These are added before transmission and removed upon reception
- Independent masking of transmit FIFO, receive FIFO, receive timeout
- Support for Direct Memory Access (DMA)
- False start bit detection
- Programmable flow control
- Fully programmable serial interface characteristics:
  - Data can be 5, 6, 7 or 8 bits
  - Even, odd, stick or no-parity bit generation and detection
  - 1 or 2 stop bit generation
  - Baud rate generation

Table 19: UART Pin Configuration

Pin Name	Pin Number		I/O	Function Name
	QFN	fcCSP		
UART0_RXD	12	M10	I	UART0_RXD
UART0_TXD	11	L9	O	UART0_TXD
GPIOA7	31	E1	I	UART1_RXD
GPIOA5	33	D2	I	
GPIOA3	36	D4	I	
GPIOA1	38	C3	I	
GPIOA6	32	E3	O	UART1_TXD
GPIOA4	34	F4	O	
GPIOA2	37	B2	O	
GPIOA0	39	A3	O	
GPIOA5	33	D2	I	UART1_CTS
GPIOA4	34	F4	O	UART1_RTS
GPIOA11	27	G1	I	UART2_RXD

Pin Name	Pin Number		I/O	Function Name
	QFN	fcCSP		
GPIOC7	9	K12	I	
F_IO2	16	J7	I	
GPIOA10	28	F2	O	UART2_TXD
GPIOC6	10	L11	O	
F_IO3	17	K6	O	

4.9.2 Application Programming Interface

Table 20: UART Interface API Elements

HANDLE UART_CREATE(UART_UNIT_IDX dev_idx)		
Parameter	dev_idx	Device index
Return		If succeeded return handle for such device, if failed return NULL

Description		Function to create a handle with parameter `dev_idx` designated
		The DA16200 (DA16600) has two UART ports
		`typedef enum __uart_unit__ {`
		`   UART_UNIT_0 = 0,`
		`   UART_UNIT_1,`
		`   UART_UNIT_MAX`
		`} UART_UNIT_IDX;`
		Normally, UART0 is used for debug console, and UART1 is used for data transfer

int UART_INIT (HANDLE handler)		
Parameter	handler	Device handle
Return		If succeeded return ERR_NONE
Description		The UART configuration should be set before this function is called After this function is called, UART operation starts
int UART_CHANGE_BAUERATE (HANDLE handler, UINT32 baudrate)		
Parameter	handler	Device handle
	baudrate	Baud rate to set
Return		If succeeded return ERR_NONE
Description		This function changes the baud rate of UART during UART operation

int UART_IOCTL(HANDLE handler, UINT32 cmd, VOID *data)		
Parameter	handler	Device handle
	cmd	Commands are defined in <UART.h> in the DA16200 (DA16600) SDK
	data	Data pointer
Return		If succeeded return ERR_NONE

int UART_IOCTL(HANDLE handler, UINT32 cmd, VOID *data)	
Description	The user can set the configuration of UART with this function Configurations of UART should be called before the UART_INIT() function. Commands are as below: ● UART_GET_DEVREG = 1, // get device physical address ● UART_SET_CLOCK, // set base clock ● UART_SET_BAUDRATE, // set baud rate ● UART_GET_BAUDRATE, // get baud rate ● UART_SET_LINECTRL, // set line control ● UART_GET_LINECTRL, // get line control ● UART_SET_CONTROL, // set UART control ● UART_GET_CONTROL, // get UART control ● UART_SET_QUESIZE, // set queue size ● UART_SET_INT, // set interrupt configuration ● UART_GET_INT, // get interrupt configuration ● UART_SET_FIFO_INT_LEVEL, // set FIFO level ● UART_GET_FIFO_INT_LEVEL, // get FIFO level ● UART_SET_USE_DMA, // set DMA use ● UART_GET_USE_DMA, // get DMA use ● UART_CHECK_RXEMPTY, // check RX FIFO empty ● UART_CHECK_RXFULL, // check RF FIFO full ● UART_CHECK_TXEMPTY, // check TX FIFO empty ● UART_CHECK_TXFULL, // check TX FIFO full ● UART_CHECK_BUSY, // check UART busy ● UART_SET_RX_SUSPEND, // set the RX function to suspend ● UART_CLEAR_ERR_INT_CNT, // clear error interrupt counter ● UART_GET_ERR_INT_CNT, // get error interrupt counter ● UART_SET_ERR_INT_CALLBACK, // set error interrupt callback function ● UART_CLEAR_FRAME_INT_CNT, //clear frame error interrupt counter ● UART_GET_FRAME_INT_CNT, // get frame error interrupt counter ● UART_SET_FRAME_INT_CALLBACK, // set frame error interrupt callback ● UART_CLEAR_PARITY_INT_CNT, // clear parity error interrupt counter ● UART_GET_PARITY_INT_CNT, // get frame error interrupt counter ● UART_SET_PARITY_INT_CALLBACK, // set frame error interrupt callback ● UART_CLEAR_BREAK_INT_CNT, // clear break error interrupt counter ● UART_GET_BREAK_INT_CNT, // get break error interrupt counter ● UART_SET_BREAK_INT_CALLBACK, // set break error interrupt callback ● UART_CLEAR_OVERRUN_INT_CNT, // clear overrun error interrupt counter ● UART_GET_OVERRUN_INT_CNT, // get overrun error interrupt counter ● UART_SET_OVERRUN_INT_CALLBACK, // set overrun interrupt callback The user can find more information in 'uart.h' file

int UART_READ (HANDLE handler, VOID *p_data, UINT32 p_dlen)		
Parameter	handler	Device handle
	p_data	Data pointer

int UART_READ (HANDLE handler, VOID *p_data, UINT32 p_dlen)		
	p_dlen	Length to read
Return		If succeeded return ERR_NONE
Description		User can use the UART_SET_RX_SUSPEND ioctl command to set the UART READ operation to suspend or not
int UART_WRITE (HANDLE handler, VOID *p_data, UINT32 p_dlen)		
Parameter	handler	Device handle
	p_data	Data pointer
	p_dlen	Length to write
Return		If succeeded return ERR_NONE
Description		UART write command
int UART_DMA_READ (HANDLE handler, VOID *p_data, UINT32 p_dlen)		
Parameter	handler	Device handle
	p_data	Data pointer
	p_dlen	Length to read
Return		If succeeded return ERR_NONE
Description		The operation of this function is the same with UART_READ, except DMA is used
int UART_DMA_WRITE (HANDLE handler, VOID *p_data, UINT32 p_dlen)		
Parameter	handler	Device handle
	p_data	Data pointer
	p_dlen	Length to write
Return		If succeeded return ERR_NONE
Description		The operation of this function is same with UART_WRITE, except DMA is used
int UART_FLUSH(HANDLE handler)		
Parameter	handler	Device handle
Return		If succeeded return ERR_NONE
Description		Flush the FIFO buffer of UART
int UART_CLOSE(HANDLE handler)		
Parameter	handler	Device handle
Return		If succeeded return ERR_NONE
Description		UART driver close command

4.9.3 Sample Code

See Ref. [3].

4.10 SPI Master

4.10.1 Introduction

The SPI master communicates in full duplex mode that uses a master-slave architecture with a single master. The master device originates the frame to be read or written. Multiple slave-devices are supported with the selection of individual chip select (CS) lines.

Table 21 shows the pin definition of the SPI master interface. To use as an SPI master, the CSB signal can be used with any of the GPIO pins. CSB [3:2] can be selected from the GPIO special function. This is done through register settings in the GPIO.

Table 21: SPI Master Pin Configuration

Pin Name	Pin Number		I/O	Function Name
	QFN	fcCSP		
GPIOx			O	E_SPI_CSB[3:1]
GPIOA6	32	E3	O	E_SPI_CSB[0]
GPIOA7	31	E1	O	E_SPI_CLK
GPIOA8	30	G3	I/O	E_SPI_MOSI or E_SPI_D[0]
GPIOA9	29	H2	I/O	E_SPI_MISO or E_SPI_D[1]
GPIOA10	28	F2	I/O	E_SPI_D[2]
GPIOA11	27	G1	I/O	E_SPI_D[3]

4.10.2 Application Programming Interface

Table 22: SPI Interface API Elements

HANDLE SPI_CREATE(UINT32 dev_id)		
Parameter	dev_id	Instance Number of SPI (UINT32))
Return		Handler of SPI Driver (HANDLE)
Description		Returns the SPI Handler that is defined in file "spi.h" ● create the GPIO handler for chip selection
int SPI_INIT (HANDLE handler)		
Parameter	Handler	SPI Driver (HANDLE)
Return		TRUE / FALSE (int)
Description		Initializes the SPI Handler to set up GPIO and activate the ISR ● create the MUTEX for support to control multi-slaves

int SPI_IOCTL(HANDLE handler, UINT32 cmd, VOID *data)		
Parameter	Handler	SPI Driver (HANDLE)
	Cmd	IOCTL command
	data	IOCTL parameters
Return		TRUE / FALSE (int)

int SPI_IOCTL(HANDLE handler, UINT32 cmd, VOID *data)		
Description		SPI_SET_SPEED ● set the target SPI clock SPI_GET_SPEED ● get the current value of SPI clock SPI_SET_FORMAT ● set the SPI interface mode ○ SPI_TYPE_MOTOROLA_O0H0 ○ SPI_TYPE_MOTOROLA_O1H1 SPI_SET_DMAMODE ● set the DMA transfer mode to the DMA mode SPI_GET_MAX_LENGTH ● the maximum burst size SPI_SET_MAX_LENGTH ● set the maximum burst size (up to 63 kB) SPI_SET_CALLACK ● set the user defined callbacks. ○ SPI_INTIDX_RORINT: the receive overrun interrupt ○ SPI_INTIDX_RTMINT: the receive timeout interrupt ○ SPI_INTIDX_RXINT: when there are four or more data in the RX FIFO ○ SPI_INTIDX_TXINT: when there are four or less data in the TX FIFO SPI_SET_CONCAT ● set the SPI burst mode to the concatenation mode SPI_SET_BUSCONTROL ● set the SPU bus access mode SPI_GET_BUSCONTROL ● get the current value of SPI bus access mode SPI_GET_DELAYSEL ● get the parameters of current delay model SPI_SET_LOCK ● lock/unlock the mutex of SPI driver
int SPI_READ(HANDLE handler, void *pdata, UINT32 dlen)		
Parameter	Handler	SPI Driver (HANDLE)
Return		zero - false, non-zero - data length (int)
Description		SPI read operation ● pdata: RX data buffer ● dlen: byte length

int SPI_WRITE(HANDLE handler, void *pdata, UINT32 dlen)		
Parameter	Handler	SPI Driver (HANDLE)
Return		zero - false, non-zero - data length (int)
Description		SPI write operation ● pdata: TX data buffer ● dlen: byte length
int SPI_WRITE_READ(HANDLE handler, void *snddata, UINT32 sndlen, void *rcvdata, UINT32 rcvlen)		
Parameter	Handler	SPI Driver (HANDLE)
Return		zero - false, non-zero - data length (int)
Description		SPI write and read operation (write before read) This function will run in concatenation mode internally ● snddata: TX data buffer ● sndlen: byte length ● rcvdata: TX data buffer ● rcvlen: byte length
Int SPI_TRANSMIT(HANDLE handler, VOID *snddata, UINT32 sndlen, VOID *rcvdata, UINT32 rcvlen)		
Parameter	Handler	SPI Driver (HANDLE)
Return		zero - false, non-zero - data length (int)
Description		Basic operation running once in SPI burst mode (send before receive) This function does not support to change a bus mode automatically ● snddata: TX data buffer ● sndlen: byte length ● rcvdata: TX data buffer ● rcvlen: byte length
Int SPI_CLOSE(HANDLE handler)		
Parameter	Handler	SPI Driver (HANDLE)
Return		TRUE / FALSE (int)
Description		Release the SPI handler

4.10.3 Sample Code

See Ref. [3].

4.11 OTP

4.11.1 Introduction

The DA16200 (DA16600) includes a one-time electrically field programmable non-volatile CMOS memory.

This memory is to protect and manage major information essential for mass production and management of products, such as booting information, MAC address, serial number, and others.

The OTP is also used for storing secret information which is used for the advanced security functions like secure booting, secure debugging, and secure asset storage. But this secret information cannot be accessed in a normal way of CPU read or write access so that it is protected from the external access.

Table 23: OTP Map

Offset	Field	Size (Bytes)
0x000	Dialog Reserved	1024
0x100	MAC Address #0 Low	4
0x101	MAC Address #0 High	4
0x102	MAC Address #1 Low	4
0x103	MAC Address #1 High	4
0x104	MAC Address #2 Low	4
0x105	MAC Address #2 High	4
0x106	MAC Address #3 Low	4
0x107	MAC Address #3 High	4
0x10A	XTAL Offset #0	4
0x10B	XTAL Offset #1	4
0x10C to 0x1FE	User Area	972

4.11.2 Application Programming Interface

Table 24: OTP API Elements

void otp_mem_create(void)		
Parameter	void	void
Return		void

void otp_mem_create(void)	

Description	Initialize OTP HW
	Before calling this function, it needs otp_clock_enable
	{
	DA16200_SYSCLOCK ->PLL_CLK_EN_4_PHY = 1;
	DA16200_SYSCLOCK ->CLK_EN_PHYBUS = 1;
	extern void DA16X_SecureBoot_OTPLock(unsigned int mode);
	DA16X_SecureBoot_OTPLock(1); // unlock
	#define CLK_GATING_OTP 0x50006048
	MEM_BYTE_WRITE(CLK_GATING_OTP, 0x00);
	otp_mem_create();
	}

void otp_mem_close(void)		
Parameter	void	void
Return		void
Description		Close the OTP HW
int otp_mem_read(UINT32 offset, UINT32 *data)		
Parameter	offset	OTP memory offset (0x00 ~ 0x1FE)
	data	[out] data pointer of buffer
Return		OTP_OK if successes
Description		Each offset store 32-bit data Offset 0x00 to 0x2c used for secure purpose. So, it may not accessible See Table 23
int otp_mem_write (UINT32 offset, UINT32 data)		
Parameter	offset	OTP memory offset (0x00 ~ 0x1FE)
	data	Data to write
Return		OTP_OK if successes
Description		Offset 0x00 to 0x2c used for secure purpose. Do not write any data within. See Table 23
int otp_mem_lock_read (UINT32 offset, UINT32 *data)		
Parameter	offset	Lock status offset. Always (0xFFF)
	data	Data pointer of lock status
Return		OTP_OK if successes

void otp_mem_close(void)		
Description		The OTP memory can be locked. Each lock bit can lock range ~ 0x40 For example: lock status value 0x00000002, it means that offset 0x40~0x7F OTP memory locked. lock bit 0 lock offset 0 ~ 0x3F lock bit 1 lock offset 0x40 ~ 0x7F lock bit 2 lock offset 0x80 ~ 0xBF lock bit 3 lock offset 0xC0 ~ 0xFF lock bit 4 lock offset 0x100 ~ 0x13F lock bit 5 lock offset 0x140 ~ 0x17F lock bit 6 lock offset 0x180 ~ 0x1BF lock bit 7 lock offset 0x1C0 ~ 0x1FE
int otp_mem_lock_write (UINT32 offset, UINT32 *data)		
Parameter	offset	Lock status offset. Always (0xFFF)
	data	Lock status value.
Return		OTP_OK if successes
Description		Refer otp_mem_lock_read()

5 NVRAM

The DA16200 (DA16600) has an NVRAM area on the flash memory to store system data and user data. NVRAM has various system configuration parameters to control the Wi-Fi function.

5.1 Application Programming Interface

There are NVRAM items of datatype integer and string. You need to use the following functions according to the item datatype.

Table 25: NVRAM API Elements

int write_nvram_int(const char *name, int val)		
Parameter	name	NVRAM item name to write
	value	Integer value to write
Return		If succeeded return 0, if failed return an error code
Description		Write a specific NVRAM item with an integer value
int write_nvram_string(const char *name, const char *val)		
Parameter	name	NVRAM item name to write
	value	Pointer to the string buffer to write
Return		If succeeded return 0, if failed return an error code
Description		Write a specific NVRAM item with a string value
int read_nvram_int(const char *name, int *_val)		
Parameter	name	NVRAM item name to read
	value	Pointer to the integer value to read the value
Return		If succeeded return 0, if failed return an error code
Description		Read an integer value of a specific NVRAM item
char *read_nvram_string(const char *name)		
Parameter	name	NVRAM item name to get
	value	Pointer to the string buffer to read the value
Return		If succeeded return 0, if failed return an error code
Description		Read an integer value of a specific NVRAM item

6 HW Accelerators

6.1 Set SRAM to Zero

6.1.1 Application Programming Interface

Table 26: HW Acc API Elements

void da16x_memset32(UINT32 *data, UINT32 seed, UINT32 length)		
Parameter	data	Buffer pointer to set
	seed	value to fill
	length	length
Return		None
Description		Fill up memory with a certain value via HW acceleration

6.1.2 Sample Code

```
#include <hal.h>

/* fill up a 1024 bytes buffer memory with 0 */
UINT32 buffer[1024];
da16x_memset32(buffer, 0, 1024);
```

6.2 CRC Calculation

6.2.1 Application Programming Interface

Table 27: CRC API Elements

UINT32 da16x_hwcrc32(UINT32 dwidth, UINT8 *data, UINT32 length, UINT32 seed)		
Parameter	dwidth	Data width to calculate CRC
	Data	Data pointer
	length	Length
	seed	CRC32 seed value (default value is 0xFFFFFFFF)
Return		Calculated CRC32 value
Description		Calculate CRC via HW accelerator

6.2.2 Sample Code

```
#include <hal.h>

/* calculate a CRC value of data buffer */
UINT8 data[64], i;
For (i=0; I < 64; i++)
           data[i] = I;
UINT32 crc_value = da16x_hwcrc32(sizeof(UINT32), (void *)data, sizeof(data), (~0));
```

6.3 Pseudo Random Number Generator (PRNG)

6.3.1 Application Programming Interface

Table 27: PRNG API Elements

UINT32 da16x_random(void)		
Parameter		void
Return		32 bits random value
Description		Generates 32 bits random value via HW accelerator

6.3.2 Sample Code

```
#include <hal.h>

UINT32 random = da16x_random();
```

6.4 Memory Copy Using DMA

6.4.1 Application Programming Interface

Table 28: HW DMA Elements

int memcpy_dma (void *dest, void *src, unsigned int len, unsigned int wait_time)		
Parameter	dest	A pointer to where you want the function to copy the data (4 B Aligned)
	src	A pointer to the buffer that you want to copy data from (4 B Aligned)
	len	The number of bytes to copy
	wait_time	0: After starting DMA operation, return from function N: Wait until memory copy is finished. If DMA operation time is greater than N milliseconds, the function returns after N milliseconds. N must have a value of at least 10 ms
Return		Always '0'
Description		Copy bytes from one buffer to another, using DMA

6.4.2 Sample Code

```
#include <sys_dma.h>

char dest[100], src[100]

memcpy_dma(dest, src, 100, 0);
```

7 Wi-Fi Interface Configuration

The DA16200 (DA16600) SDK defines various parameters for Wi-Fi interface configuration and they are saved as profiles in the NVRAM. After system reset, the DA16200 (DA16600) reads an existing profile and sets the Wi-Fi interface based on that profile.

7.1 Application Programming Interface

The DA16200 (DA16600) SDK provides several functions with the following features to get or set system profiles:

- Four simple functions to get or set each parameter
- Error code to verify the result

Each parameter is related to an NVRAM item so there are integer datatype parameters and string datatype parameters. You need to use these functions according to parameter type.

Table 26: Wi-Fi Configuration API

int da16x_set_config_int(int name, int value)		
Parameter	name	Parameter index to set
	value	Integer value to set
Return		If succeeded return 0 (CC_SUCCESS), if failed return an error code
Description		Set a specific parameter with an integer value For example: `ret = da16x_set_config_int (Da16x_CONF_INT_CHANNEL, 11)` ● Set the operating channel of the AP interface to 11
int da16x_set_config_str (int name, char *value)		
Parameter	name	Parameter index to set
	value	Pointer to the string value to set
Return		If succeeded return 0 (CC_SUCCESS), if failed return an error code
Description		Set a specific parameter with a string value For example: `ret = da16x_set_config_int(Da16x_CONF_STR_IP_0, “10.0.0.1”)` ● Set the IP address of the STA interface to 10.0.0.1
int da16x_get_config_int (int name, int *value)		
Parameter	name	Parameter index to get
	value	Pointer to the integer variable to get the parameter value
Return		If succeeded return 0 (CC_SUCCESS), if failed return an error code
Description		Get an integer value of a specific parameter For example: `ret = da16x_get_config_int(Da16x_CONF_INT_CHANNEL, &channel)` ● Get the operating channel of the AP interface
int da16x_get_config_str (int name, char *value)		
Parameter	name	Parameter index to get
	value	Pointer to the string buffer to get the parameter value
Return		If succeeded return 0 (CC_SUCCESS), if failed return an error code
Description		Get a string value of a specific parameter

int da16x_set_config_int(int name, int value)		
		For example: `ret = da16x_get_config_str(Da16x_CONF_STR_IP_0, ip_addr)` ● Get the IP address of the STA interface
int da16x_set_nvcache_str(int name, char *value)		
Parameter	name	Parameter name to set
	value	Points to the value (str) to set
Return		If succeeded, return 0 (CC_SUCCESS), if failed return an error code
Description		Set name/value pair to NVRAM cache area (not in sflash). To make permanent, invoke `da16x_nvcache2flash()` For example: `ret = da16x_set_nvcache_str(Da16x_CONF_STR_IP_0, ip_addr)` ● Set IP address of the STA interface
int da16x_set_nvcache_int(int name, int value)		
Parameter	name	Parameter name to set
	value	Points to the value (int) to set
Return		If succeeded, return 0 (CC_SUCCESS), if failed return an error code
Description		Set name/value pair to NVRAM cache area (not in sflash). To make permanent, invoke da16x_nvcache2flash () For example: `ret = da16x_set_nvcache_int(Da16x_CONF_INT_CHANNEL, 11)` ● Set the operating channel of the AP interface to 11
void da16x_nvcache2flash(void)		
Parameter	void	void
Return		void
Description		Commit parameters (set by da16x_set_nvcache_int/str) in NVRAM cache to flash

7.1.1 Integer Type Parameters

Table 27: NVRAM Integer Type

Name	Description
DA16X_CONF_INT_MODE	Wi-Fi operation mode ● 0: STA ● 1: Soft-AP
DA16X_CONF_INT_AUTH_MODE_0	Wi-Fi authentication mode for STA interface ● CC_VAL_AUTH_OPEN ● CC_VAL_AUTH_WEP ● CC_VAL_AUTH_WPA ● CC_VAL_AUTH_WPA2 ● CC_VAL_AUTH_WPA_AUTO (WPA & WPA2)

Name	Description
DA16X_CONF_INT_AUTH_MODE_1	Wi-Fi authentication mode for Soft-AP interface ● CC_VAL_AUTH_OPEN ● CC_VAL_AUTH_WPA ● CC_VAL_AUTH_WPA2 ● CC_VAL_AUTH_WPA_AUTO (WPA & WPA2) (WEP is unsupported on the DA16200 (DA16600) AP mode)
DA16X_CONF_INT_WEP_KEY_INDEX	Wi-Fi WEP key index number (0~3)
DA16X_CONF_INT_ENCRYPTION_0	Wi-Fi data encryption mode for STA interface ● CC_VAL_ENC_TKIP ● CC_VAL_ENC_CCMP ● CC_VAL_ENC_AUTO (TKIP & CCMP)
DA16X_CONF_INT_ENCRYPTION_1	Wi-Fi data encryption mode for Soft-AP interface ● CC_VAL_ENC_TKIP ● CC_VAL_ENC_CCMP ● CC_VAL_ENC_AUTO (TKIP & CCMP)
DA16X_CONF_INT_WIFI_MODE_0	Wi-Fi mode based on IEEE 802.11 standard for STA interface ● CC_VAL_WFMODE_BGN ● CC_VAL_WFMODE_GN ● CC_VAL_WFMODE_BG ● CC_VAL_WFMODE_N ● CC_VAL_WFMODE_G ● CC_VAL_WFMODE_B
DA16X_CONF_INT_WIFI_MODE_1	Wi-Fi mode based on IEEE 802.11 standard for Soft-AP interface ● CC_VAL_WFMODE_BGN ● CC_VAL_WFMODE_GN ● CC_VAL_WFMODE_BG ● CC_VAL_WFMODE_N ● CC_VAL_WFMODE_G ● CC_VAL_WFMODE_B
DA16X_CONF_INT_CHANNEL	Soft-AP operation channel setting by channel number ● 1~11: for US ● 0: Auto
DA16X_CONF_INT_FREQUENCY	Soft-AP operation channel setting by frequency value (MHz)
DA16X_CONF_INT_ROAM	Operating roaming function for STA interface ● 0: Stop ● 1: Run
DA16X_CONF_INT_ROAM_THRESHOLD	Roaming threshold for STA interface (-95 ~ 0 dBm)
DA16X_CONF_INT_BEACON_INTERVAL	IEEE 802.11 beacon interval (msec.)
DA16X_CONF_INT_INACTIVITY	Inactive STA disconnecting time (sec.)
DA16X_CONF_INT_RTS_THRESHOLD	IEEE 802.11 RTS threshold (byte)

Name	Description
DA16X_CONF_INT_WMM	WMM On/Off setting ● 0: Off ● 1: On
DA16X_CONF_INT_WMM_PS	WMM-PS On/Off setting ● 0: Off ● 1: On
DA16X_CONF_INT_DHCP_CLIENT	DHCP client On/Off for STA interface ● 0: Off ● 1: On
DA16X_CONF_INT_DHCP_SERVER	DHCP server On/Off for Soft-AP interface ● 0: Off ● 1: On
DA16X_CONF_INT_DHCP_LEASE_TIME	DHCP server lease time (sec.)

7.1.2 String Type Parameters

Table 28: NVRAM String Type

Name	Description
DA16X_CONF_STR_SSID_0	AP SSID to connect (~ 32 letters)
DA16X_CONF_STR_SSID_1	Soft-AP SSID to operate (~ 32 letters)
DA16X_CONF_STR_WEP_KEY0 DA16X_CONF_STR_WEP_KEY1 DA16X_CONF_STR_WEP_KEY2 DA16X_CONF_STR_WEP_KEY3	WEP keys of the AP to connect (5 or 13 letters with ASCII / 10 or 26 letters with hexadecimal)
DA16X_CONF_STR_PSK_0	PSK of the AP to connect (~ 63 letters)
DA16X_CONF_STR_PSK_1	Soft-AP PSK to operate (~ 63 letters)
DA16X_CONF_STR_COUNTRY	Country code (2 or 3 letters, for example KR, US, JP, CH, etc.) defined by ISO 3166-1 alpha-2 standard
DA16X_CONF_STR_DEVICE_NAME	DA16200 (DA16600) device name (for WPS or Wi-Fi Direct)
DA16X_CONF_STR_IP_0	STA interface IP address
DA16X_CONF_STR_NETMASK_0	STA interface netmask
DA16X_CONF_STR_GATEWAY_0	STA interface gateway address
DA16X_CONF_STR_IP_1	Soft-AP interface IP address
DA16X_CONF_STR_NETMASK_1	Soft-AP interface netmask
DA16X_CONF_STR_GATEWAY_1	Soft-AP interface gateway address
DA16X_CONF_STR_DNS_0	STA interface DNS address
DA16X_CONF_STR_DHCP_START_IP DA16X_CONF_STR_DHCP_END_IP	DHCP server IP range assigned
DA16X_CONF_STR_DHCP_DNS	DHCP server DNS IP address assigned

7.1.3 Sample Code

If you need to set many name/value NVRAM parameters at the same time, then use `da16x_set_nvcache_int/str()` and `da16x_nvcache2flash()`. Use of `da16x_set_config_str/int()` is good for setting one or two values, but if there is a need to set many NVRAM parameters (that is Soft-AP / STA setup), then always use cache function `da16x_set_nvcache_int/str` followed by `da16x_nvcache2flash()`, which will give much better performance to your application.

The following example explains how to set STA mode.

Table 29: NVRAM Sample Code on STA Mode

```
/* Wi-Fi Configuration */

clear_tmp_nvram_env(); // Clear Cache


// start setting name/value NVRAM parameters to NVRAM Cache (no delay)

da16x_set_nvcache_int(DA16X_CONF_INT_MODE, 0);
da16x_set_nvcache_str(DA16X_CONF_STR_SSID_0, ssid);
da16x_set_nvcache_int(DA16X_CONF_INT_AUTH_MODE_0, auth_type);
if (auth_type == CC_VAL_AUTH_WEP) {
        da16x_set_nvcache_str(DA16X_CONF_STR_WEP_KEY0, wep_key[0]);
        da16x_set_nvcache_str(DA16X_CONF_STR_WEP_KEY1, wep_key[1]);
        da16x_set_nvcache_str(DA16X_CONF_STR_WEP_KEY2, wep_key[2]);
        da16x_set_nvcache_str(DA16X_CONF_STR_WEP_KEY3, wep_key[3]);
        da16x_set_nvcache_str(DA16X_CONF_INT_WEP_KEY_INDEX, wep_key_index);
} else if (auth_type > CC_VAL_AUTH_WEP) {
    da16x_set_nvcache_str(DA16X_CONF_STR_PSK_0, psk);
    da16x_set_nvcache_int(DA16X_CONF_INT_ENCRYPTION_0, encryption);
}
da16x_set_nvcache_int(DA16X_CONF_INT_WIFI_MODE_0, wifi_mode);

/* IP & DHCP Client Setting */
da16x_set_nvcache_int(DA16X_CONF_INT_DHCP_CLIENT, dhcp_client);
if (!dhcp_client) {
    da16x_set_nvcache_str(DA16X_CONF_STR_IP_0, ip);
    da16x_set_nvcache_str(DA16X_CONF_STR_NETMASK_0, subnet);
    da16x_set_nvcache_str(DA16X_CONF_STR_GATEWAY_0, gateway);
    da16x_set_nvcache_str(DA16X_CONF_STR_DNS_0, dns);
}

da16x_nvcache2flash(); // commit name/value params in Cache to flash memory


reboot_func(SYS_REBOOT);
```

The following example explains how to set Soft-AP mode.

Table 30: NVRAM Sample Code on Soft-AP Mode

```
/* SoftAP Configuration */

clear_tmp_nvram_env(); // Clear Cache

...
// start setting name/value NVRAM parameters to NVRAM Cache (no delay)
da16x_set_nvcache_int(DA16X_CONF_INT_MODE, 1);
```

```
da16x_set_nvcache_str(DA16X_CONF_STR_SSID_1, ssid);
da16x_set_nvcache_int(DA16X_CONF_INT_AUTH_MODE_1, auth_type);
if (auth_type > CC_VAL_AUTH_WEP) {
       da16x_set_nvcache_str (DA16X_CONF_STR_PSK_1, psk);
       da16x_set_nvcache_int(DA16X_CONF_INT_ENCRYPTION_1, encryption);
}
da16x_set_nvcache_int(DA16X_CONF_INT_CHANNEL, channel);
da16x_set_nvcache_int(DA16X_CONF_STR_COUNTRY, country_code);
da16x_set_nvcache_int(DA16X_CONF_INT_WIFI_MODE_1, wifi_mode);
da16x_set_nvcache_int(DA16X_CONF_INT_WMM, wmm);
da16x_set_nvcache_int(DA16X_CONF_INT_WMM_PS, wmm_ps);

/* IP Setting */
da16x_set_nvcache_str(DA16X_CONF_STR_IP_1, ip);
da16x_set_nvcache_str(DA16X_CONF_STR_NETMASK_1, subnet);
da16x_set_nvcache_str(DA16X_CONF_STR_GATEWAY_1, gateway);

/* DHCP Server Setting */
if (dhcp_server) {
       da16x_set_nvcache_str(DA16X_CONF_STR_DHCP_START_IP, start_ip);
       da16x_set_nvcache_str(DA16X_CONF_STR_DHCP_END_IP, end_ip);
      da16x_set_nvcache_str(DA16X_CONF_STR_DHCP_DNS, dhcp_dns);
       da16x_set_nvcache_str(DA16X_CONF_INT_DHCP_LEASE_TIME, dhcp_lease_time);
}
da16x_set_nvcache_int(DA16X_CONF_INT_DHCP_SERVER, dhcp_server);


da16x_nvcache2flash(); // commit name/value params in Cache to flash memory


reboot_func(SYS_REBOOT);
```

7.2 Soft-AP Configuration by Factory Reset

Many IoT devices start as a Soft-AP device to operate AP provisioning. The DA16200 (DA16600) has a Factory Reset function to change to Soft-AP mode after the Factory Reset button is clicked. This button is described in the section **Board Description** in Ref. [2] and is connected to GPIO 7 on the DA16200 (DA16600) EVB.

You can configure the Soft-AP interface with your own values. The DA16200 (DA16600) SDK provides a simple way to do this. This section describes how to configure the default values in the DA16200 (DA16600) SDK.

7.2.1 Configuration Data Structure Integer Type Parameters

The DA16200 (DA16600) SDK has the structure shown in Table 31 to configure Soft-AP interface.

Table 31: Soft-AP Interface Code

```
[ ~/FreeRTOS_SDK/core/system/include/common/da16x_network_common.h ]


/* For Customer's Soft-AP configuration */
#define       MAX_SSID_LEN        32
#define       MAX_PASSKEY_LEN     64
#define       MAX_IP_ADDR_LEN     16
```

```
#define        AP_OPEN_MODE        0
#define        AP_SECURITY_MODE    1

#define        IPADDR_DEFAULT      0
#define        IPADDR_CUSTOMER     1

#define        DHCPD_DEFAULT       0
#define        DHCPD_CUSTOMER      1

typedef struct _softap_config {
        int     customer_cfg_flag;  // MODE_ENABLE, MODE_DISABLE


       char     ssid_name[MAX_SSID_LEN+1];
       char     psk[MAX_PASSKEY_LEN+1];
        char    auth_type;          // AP_OPEN_MODE, AP_SECURITY_MODE
       char     country_code[4];


        int     customer_ip_address; // IPADDR_DEFAULT, IPADDR_CUSTOMER
       char     ip_addr[MAX_IP_ADDR_LEN];
       char     subnet_mask[MAX_IP_ADDR_LEN];
       char     default_gw[MAX_IP_ADDR_LEN];
       char     dns_ip_addr[MAX_IP_ADDR_LEN];


        int     customer_dhcpd_flag; // DHCPD_DEFAULT, DHCPD_CUSTOMER
       //int    dhcpd_ip_cnt;
       int      dhcpd_lease_time;
       char     dhcpd_start_ip[MAX_IP_ADDR_LEN];
       char     dhcpd_end_ip[MAX_IP_ADDR_LEN];
       char     dhcpd_dns_ip_addr[MAX_IP_ADDR_LEN];
} softap_config_t;
```

- int customer_cfg_flag: Flag for user configuration
  - MODE_DISABLE (0)   : Do not use user configuration
  - MODE_ENABLE (1)   : Use user configuration
- char ssid_name[MAX_SSID_LEN+1]: SSID of Soft-AP. Max length is 32 bytes
- char psk[MAX_PASSKEY_LEN]: Pairwise key. Max length is 64 bytes
- char auth_type: Authentication type
  - OPEN_MODE (0)
  - AP_SECURITY_MODE (1)
- char country_code [4]: Country code
  See the section on **Country Code** in Ref. [3] or Appendix B.1
- int customer_ip_address: IP address type
  - IPADDR_DEFAULT (0)   : IP class is 10.0.0.1
  - IPADDR_CUSTOMER (1)   : User defined IP address
    The following parameters should be defined:
    char   ip_addr[MAX_IP_ADDR_LEN]

char	subnet_mask[MAX_IP_ADDR_LEN]
char	default_gw[MAX_IP_ADDR_LEN]
char	dns_ip_addr[MAX_IP_ADDR_LEN]

- int customer_dhcpd_flag: DHCP server IP address range
  - DHCPD_DEFAULT (0) : 10.0.0.2 ~ 10.0.0.11 (10 clients)
  - DHCPD_CUSTOMER (1) : User defined range

    Need to define the following parameters:

int	dhcpd_lease_time
char	dhcpd_start_ip[MAX_IP_ADDR_LEN]
char	dhcpd_end_ip[MAX_IP_ADDR_LEN]
char	dhcpd_dns_ip_addr[MAX_IP_ADDR_LEN]

7.2.2 How to Configure the Soft-AP Interface

The DA16200 (DA16600) SDK has the function shown in Table 32 to configure the Soft-AP interface. You can write your own values. This function is invoked when a factory reset is done.

Table 32: Soft-AP Configuration Code

```
[ ~/FreeRTOS_SDK/customer/user_main/src/system_start.c ]

void set_customer_softap_config(void)
{
#ifdef __SUPPORT_FACTORY_RST_APMODE__
      /* Set to user costomer's configuration */
      ap_config_param->customer_cfg_flag = MODE_DISABLE;
        // MODE_ENABLE, MODE_DISABLE
      /*
       * Wi-Fi configuration
       */
        /* SSID prefix */
      sprintf(ap_config_param->ssid_name, "%s", "DA16200");

      /* Default open mode: AP_OPEN_MODE, AP_SECURITY_MODE */
      ap_config_param->auth_type = AP_OPEN_MODE;
      if (ap_config_param->auth_type == AP_SECURITY_MODE);
       sprintf(ap_config_param->psk, "%s", "12345678");

      /* Country Code: Default country US */
      sprintf(ap_config_param->country_code, "%s", DFLT_AP_COUNTRY_CODE);

      /*
       * Network IP address configuration
       */
      ap_config_param->customer_ip_address = IPADDR_DEFAULT;
      if (ap_config_param->customer_ip_address == IPADDR_CUSTOMER) {
       sprintf(ap_config_param->ip_addr, "%s", "192.168.1.1");
       sprintf(ap_config_param->subnet_mask,    "%s", "255.255.255.0");
       sprintf(ap_config_param->default_gw,     "%s", "192.168.1.1");
       sprintf(ap_config_param->dns_ip_addr,    "%s", "8.8.8.8");
      }
      /*
       * DHCP Server configuration
       */
      ap_config_param->customer_dhcpd_flag = DHCPD_DEFAULT;
      if (ap_config_param->customer_dhcpd_flag == DHCPD_CUSTOMER) {
       ap_config_param->dhcpd_lease_time = 3600;

       sprintf(ap_config_param->dhcpd_start_ip, "%s", "192.168.1.101");
       sprintf(ap_config_param->dhcpd_end_ip, "%s", "192.168.1.108");
       sprintf(ap_config_param->dhcpd_dns_ip_addr, "%s", "8.8.8.8");
      }
#endif /* __SUPPORT_FACTORY_RST_APMODE__ */
}
```

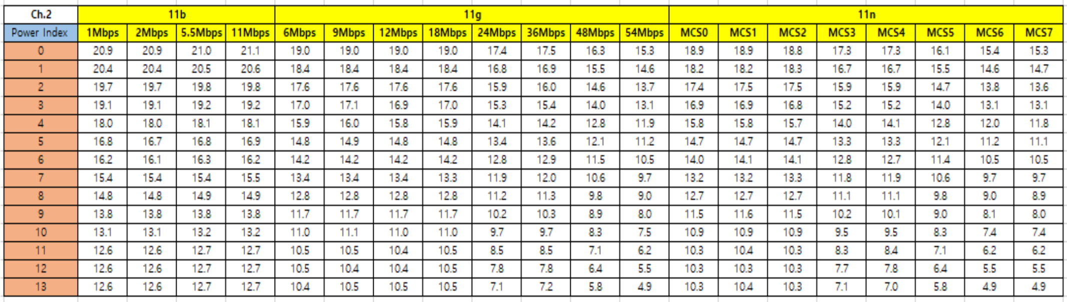
7.3 Soft-AP Provisioning Protocol

The DA16200 (DA16600) supports the Soft-AP mode for a Wi-Fi interface setup. The provisioning thread automatically runs when the DA16200 (DA16600) starts in Soft-AP mode.

See DA16200 (DA16600) Provisioning the Mobile App in Ref. [5].

8 TX Power Table Edit

The DA16200 (DA16600) SDK allows users to tune and edit Tx Power (per channel) for FCC or country-dependent product customization/optimization.



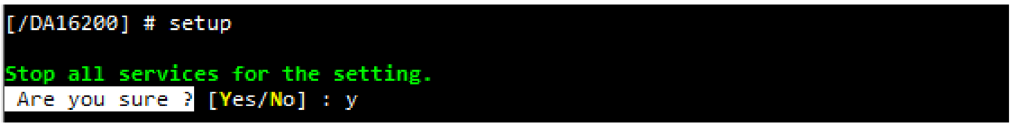
Ch.2	11b				11g								11n							
Power Index	1Mbps	2Mbps	5.5Mbps	11Mbps	6Mbps	9Mbps	12Mbps	18Mbps	24Mbps	36Mbps	48Mbps	54Mbps	MCS0	MCS1	MCS2	MCS3	MCS4	MCS5	MCS6	MCS7
0	20.9	20.9	21.0	21.1	19.0	19.0	19.0	19.0	17.4	17.5	16.3	15.3	18.9	18.9	18.8	17.3	17.3	16.1	15.4	15.3
1	20.4	20.4	20.5	20.6	18.4	18.4	18.4	18.4	16.8	16.9	15.5	14.6	18.2	18.2	18.3	16.7	16.7	15.5	14.6	14.7
2	19.7	19.7	19.8	19.8	17.6	17.6	17.6	17.6	15.9	16.0	14.6	13.7	17.4	17.5	17.5	15.9	15.9	14.7	13.8	13.6
3	19.1	19.1	19.2	19.2	17.0	17.1	16.9	17.0	15.3	15.4	14.0	13.1	16.9	16.9	16.8	15.2	15.2	14.0	13.1	13.1
4	18.0	18.0	18.1	18.1	15.9	16.0	15.8	15.9	14.1	14.2	12.8	11.9	15.8	15.8	15.7	14.0	14.1	12.8	12.0	11.8
5	16.8	16.7	16.8	16.9	14.8	14.9	14.8	14.8	13.4	13.6	12.1	11.2	14.7	14.7	14.7	13.3	13.3	12.1	11.2	11.1
6	16.2	16.1	16.3	16.2	14.2	14.2	14.2	14.2	12.8	12.9	11.5	10.5	14.0	14.1	14.1	12.8	12.7	11.4	10.5	10.5
7	15.4	15.4	15.4	15.5	13.4	13.4	13.4	13.3	11.9	12.0	10.6	9.7	13.2	13.2	13.3	11.8	11.9	10.6	9.7	9.7
8	14.8	14.8	14.9	14.9	12.8	12.8	12.8	12.8	11.2	11.3	9.8	9.0	12.7	12.7	12.7	11.1	11.1	9.8	9.0	8.9
9	13.8	13.8	13.8	13.8	11.7	11.7	11.7	11.7	10.2	10.3	8.9	8.0	11.5	11.6	11.5	10.2	10.1	9.0	8.1	8.0
10	13.1	13.1	13.2	13.2	11.0	11.1	11.0	11.0	9.7	9.7	8.3	7.5	10.9	10.9	10.9	9.5	9.5	8.3	7.4	7.4
11	12.6	12.6	12.7	12.7	10.5	10.5	10.4	10.5	8.5	8.5	7.1	6.2	10.3	10.4	10.3	8.3	8.4	7.1	6.2	6.2
12	12.6	12.6	12.7	12.7	10.5	10.4	10.4	10.5	7.8	7.8	6.4	5.5	10.3	10.3	10.3	7.7	7.8	6.4	5.5	5.5
13	12.6	12.6	12.7	12.7	10.4	10.5	10.5	10.5	7.1	7.2	5.8	4.9	10.3	10.4	10.3	7.1	7.0	5.8	4.9	4.9

Figure 15: TX Power Table

8.1 Tune TX Power

Before setting TX power to your Main image, you may need to tune and test TX power. The following procedure shows how to change the TX power index for each channel with console commands, TX power indices, and corresponding power values.

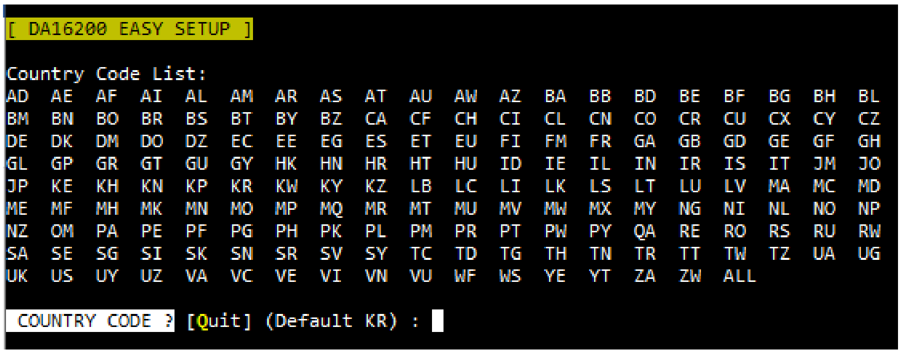
1. Run command `setup` to configure the station interface. See Figure 16.



```
[/DA16200] # setup

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

Figure 16: Tune TX Power: Setup

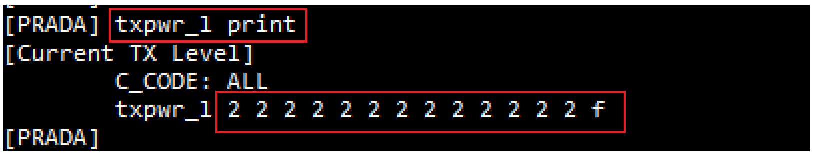
2. At prompt `COUNTRY CODE? [Quit] (Default KR)`**:** enter **ALL** as the country code for Tx Power tuning purpose. See Figure 17.



```
[ DA16200 EASY SETUP ]

Country Code List:
AD   AE   AF   AI   AL   AM   AR   AS   AT   AU   AW   AZ   BA   BB   BD   BE   BF   BG   BH   BL
BM   BN   BO   BR   BS   BT   BY   BZ   CA   CF   CH   CI   CL   CN   CO   CR   CU   CX   CY   CZ
DE   DK   DM   DO   DZ   EC   EE   EG   ES   ET   EU   FI   FM   FR   GA   GB   GD   GE   GF   GH
GL   GP   GR   GT   GU   GY   HK   HN   HR   HT   HU   ID   IE   IL   IN   IR   IS   IT   JM   JO
JP   KE   KH   KN   KP   KR   KW   KY   KZ   LB   LC   LI   LK   LS   LT   LU   LV   MA   MC   MD
ME   MF   MH   MK   MN   MO   MP   MQ   MR   MT   MU   MV   MW   MX   MY   NG   NI   NL   NO   NP
NZ   OM   PA   PE   PF   PG   PH   PK   PL   PM   PR   PT   PW   PY   QA   RE   RO   RS   RU   RW
SA   SE   SG   SI   SK   SN   SR   SV   SY   TC   TD   TG   TH   TN   TR   TT   TW   TZ   UA   UG
UK   US   UY   UZ   VA   VC   VE   VI   VN   VU   WF   WS   YE   YT   ZA   ZW   ALL

 COUNTRY CODE ? [Quit] (Default KR) : 
```

Figure 17: Tune TX Power: Choose Country Code

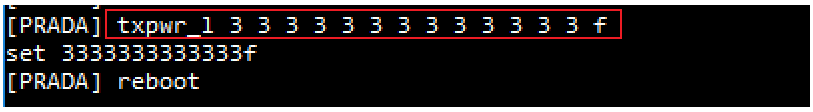
3. Reboot and connect to an AP.
4. Examine the current TX power indices. See Figure 18.



```
[PRADA] txpwr_l print
[Current TX Level]
        C_CODE: ALL
        txpwr_l 2 2 2 2 2 2 2 2 2 2 2 2 2 2 f
[PRADA]
```

Figure 18: Tune TX Power: Check TX Power Indices

5. Change the power indices as you want and reboot.



```
[PRADA] txpwr_l 3 3 3 3 3 3 3 3 3 3 3 3 3 3 f
set 33333333333333f
[PRADA] reboot
```

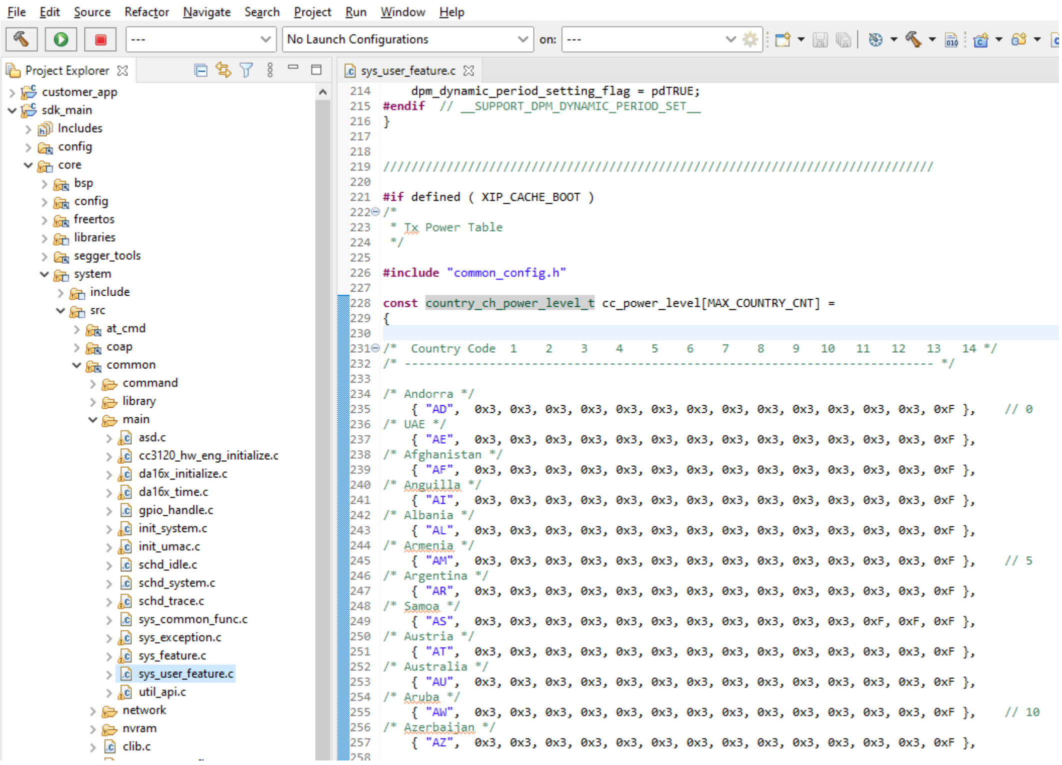
Figure 19: Tune TX Power: Modify TX Power Indices

6. Measure the Tx power value for each channel with WLAN Test equipment, such as MT8860C (Network Mode), and check the Tx power values.
7. Repeat each step until the Tx power values that you want are obtained.

8.1 Apply Tuned TX Power to Main Image

The following procedure describes how to set the tuned TX power indices to your Main image.

1. In the DA16200 (DA16600) SDK, open *~/FreeRTOS_SDK/core/system/src/common/main/sys_user_feature.c*.



```
    dpm_dynamic_period_setting_flag = pdTRUE;
#endif  // __SUPPORT_DPM_DYNAMIC_PERIOD_SET__
}


////////////////////////////////////////////////////////////////////////////////

#if defined ( XIP_CACHE_BOOT )
/*
 * Tx Power Table
 */

#include "common_config.h"

const country_ch_power_level_t cc_power_level[MAX_COUNTRY_CNT] =
{

/*  Country Code  1    2    3    4    5    6    7    8    9    10   11   12   13   14 */
/* ------------------------------------------------------------------------------ */

/* Andorra */
    { "AD",  0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0xF },    // 0
/* UAE */
    { "AE",  0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0xF },
/* Afghanistan */
    { "AF",  0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0xF },
/* Anguilla */
    { "AI",  0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0xF },
/* Albania */
    { "AL",  0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0xF },
/* Armenia */
    { "AM",  0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0xF },    // 5
/* Argentina */
    { "AR",  0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0xF },
/* Samoa */
    { "AS",  0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0xF, 0xF, 0xF },
/* Austria */
    { "AT",  0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0xF },
/* Australia */
    { "AU",  0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0xF },
/* Aruba */
    { "AW",  0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0xF },    // 10
/* Azerbaijan */
    { "AZ",  0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0xF },
```

Figure 20: TX Power Table Source Code

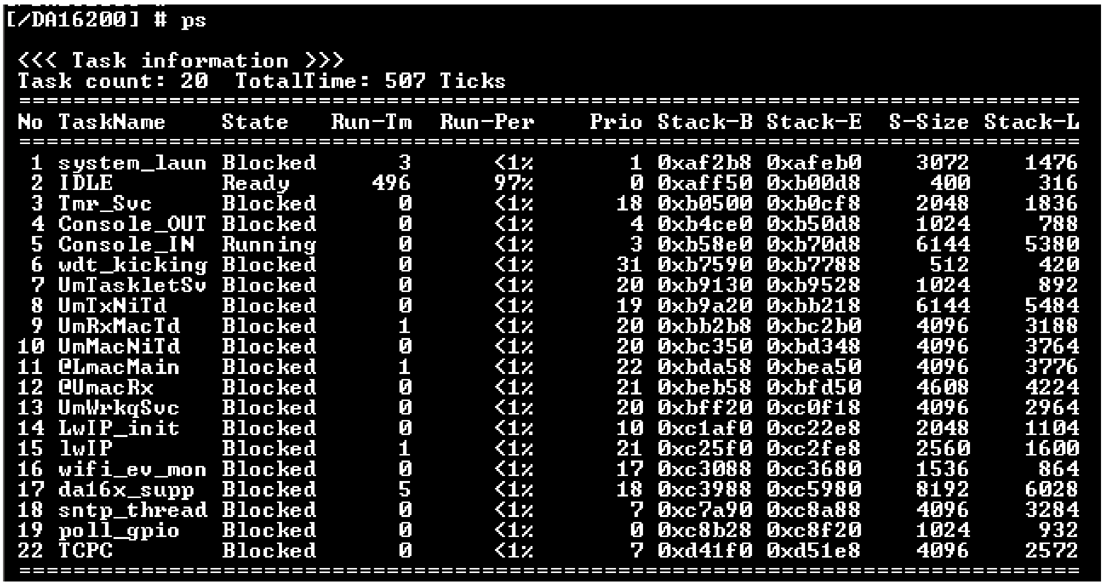
2. The array `cc_power_level` contains the default values customized for FCC. Edit the power values for a specific country, or whatever countries you like with tuned values.
3. Re-build the SDK.
4. When the rebuilt software is started and the country is selected, the corresponding Tx power value that is set for the channel will take effect.

9 Tips

9.1 Find/Optimize Stack Size for Your Application

The stack size for an application may vary per application. The DA16200 (DA16600) has a tool (a console command) called ps that shows the list of threads and the status of each application stack.

Figure 21 is a snapshot of command ps when tcp_client_sample.c is run.



```
[/DA16200] # ps

 <<< Task information >>>
 Task count: 20  TotalTime: 507 Ticks
 ==============================================================================
 No TaskName    State   Run-Tm  Run-Per    Prio Stack-B Stack-E  S-Size Stack-L
 ==============================================================================
  1 system_laun Blocked      3      <1%       1 0xaf2b8 0xafeb0    3072    1476
  2 IDLE        Ready      496      97%       0 0xaff50 0xb00d8     400     316
  3 Tmr_Svc     Blocked      0      <1%      18 0xb0500 0xb0cf8    2048    1836
  4 Console_OUT Blocked      0      <1%       4 0xb4ce0 0xb50d8    1024     788
  5 Console_IN  Running      0      <1%       3 0xb58e0 0xb70d8    6144    5380
  6 wdt_kicking Blocked      0      <1%      31 0xb7590 0xb7788     512     420
  7 UmTaskletSv Blocked      0      <1%      20 0xb9130 0xb9528    1024     892
  8 UmTxNiTd    Blocked      0      <1%      19 0xb9a20 0xbb218    6144    5484
  9 UmRxMacTd   Blocked      1      <1%      20 0xbb2b8 0xbc2b0    4096    3188
 10 UmMacNiTd   Blocked      0      <1%      20 0xbc350 0xbd348    4096    3764
 11 @LmacMain   Blocked      1      <1%      22 0xbda58 0xbea50    4096    3776
 12 @UmacRx     Blocked      0      <1%      21 0xbeb58 0xbfd50    4608    4224
 13 UmWrkqSvc   Blocked      0      <1%      20 0xbff20 0xc0f18    4096    2964
 14 LwIP_init   Blocked      0      <1%      10 0xc1af0 0xc22e8    2048    1104
 15 lwIP        Blocked      1      <1%      21 0xc25f0 0xc2fe8    2560    1600
 16 wifi_ev_mon Blocked      0      <1%      17 0xc3088 0xc3680    1536     864
 17 da16x_supp  Blocked      5      <1%      18 0xc3988 0xc5980    8192    6028
 18 sntp_thread Blocked      0      <1%       7 0xc7a90 0xc8a88    4096    3284
 19 poll_gpio   Blocked      0      <1%       0 0xc8b28 0xc8f20    1024     932
 22 TCPC        Blocked      0      <1%       7 0xd41f0 0xd51e8    4096    2572
 ==============================================================================
```

Figure 21: Check Stack Size

TCPC is the name of the tread for this sample application, and the stack size is 1020 (which is defined in sample_apps.c).

Table 33: TCP Client Sample Code

```
…
#if defined (__TCP_CLIENT_SAMPLE__)
     { SAMPLE_TCP_CLI, tcp_client_sample, 1024, (tskIDLE_PRIORITY + 7), TRUE,
FALSE, TCP_CLI_TEST_PORT, RUN_ALL_MODE },
#endif // (__TCP_CLIENT_SAMPLE__)
…
```

Command ps shows the following information:

- Stack-B/E: the stack address
- S-Size: the stack size allocated
- Stack-L: peak usage size of the stack

To find and optimize the stack size for this application, for example if this application has four use cases, do the following:

1. First, over-allocate stack memory as a precaution, like 2K, "just to be safe".
2. Run each use case and examine the peak stack usage with command ps.

3. Allocate optimal memory based on peak usage info, to find and optimize stack size
   (If you do not know all the possible use case scenarios, then give the stack size enough room just to be safe.).

Appendix A Open-Source License

Mosquitto 1.4.14 License

Eclipse Distribution License 1.0

Copyright (c) 2007, Eclipse Foundation, Inc. and its licensors.

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of the Eclipse Foundation, Inc. nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Appendix B Country Code and TX Power

This section lists the country codes that the DA16200 (DA16600) supports and the supported channels of 2.4 GHz bandwidth in the STA and the Soft-AP mode.

B.1 Country Code and Channels

Table 34: Country Code

Country Code	Country	STA Channels	Soft-AP Channels
"AD"	Andorra	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“AE”	United Arab Emirates	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“AF”	Afghanistan	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“AI”	Anguilla	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“AL”	Albania	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“AM”	Netherlands Antilles	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“AR”	Argentina	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“AS”	American Samoa	1,2,3,4,5,6,7,8,9,10,11	1,2,3,4,5,6,7,8,9,10,11
“AT”	Austria	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“AU”	Australia	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“AW”	Aruba	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“AZ”	Azerbaijan	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“BA”	Bosnia and Herzegovina	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“BB”	Barbados	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“BD”	Bangladesh	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“BE”	Belgium	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“BF”	Burkina Faso	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“BG”	Bulgaria	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“BH”	Bahrain	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“BL”	Saint-Barthelemy	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“BM”	Bermuda	1,2,3,4,5,6,7,8,9,10,11	1,2,3,4,5,6,7,8,9,10,11
“BN”	Brunei Darussalam	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“BO”	Bolivia	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“BR”	Brazil	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“BS”	Bahamas	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“BT”	Bhutan	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“BY”	Belarus	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“BZ”	Belize	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“CA”	Canada	1,2,3,4,5,6,7,8,9,10,11	1,2,3,4,5,6,7,8,9,10,11

Country Code	Country	STA Channels	Soft-AP Channels
“CF”	Central African Republic	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“CH”	Switzerland	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“CI”	Ivory Coast	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“CL”	Chile	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“CN”	China	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“CO”	Colombia	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“CR”	Costa Rica	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“CU”	Cuba	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“CX”	Christmas Island	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
"CY”	Cyprus	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“CZ”	Czech Republic	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“DE”	Germany	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“DK”	Denmark	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“DM”	Dominica	1,2,3,4,5,6,7,8,9,10,11	1,2,3,4,5,6,7,8,9,10,11
“DO”	Dominican Republic	1,2,3,4,5,6,7,8,9,10,11	1,2,3,4,5,6,7,8,9,10,11
“DZ”	Algeria	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“EC”	Ecuador	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“EE”	Estonia	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“EG”	Egypt	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“ES”	Spain	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“ET”	Ethiopia	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“EU”	Europe	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“FI”	Finland	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“FM”	Micronesia, Federated States of	1,2,3,4,5,6,7,8,9,10,11	1,2,3,4,5,6,7,8,9,10,11
“FR”	France	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“GA”	Gabon	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“GB”	United Kingdom	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“GD”	Grenada	1,2,3,4,5,6,7,8,9,10,11	1,2,3,4,5,6,7,8,9,10,11
“GE”	Georgia	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“GF”	French Guiana	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“GH”	Ghana	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“GL”	Greenland	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“GP”	Guadeloupe	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“GR”	Greece	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“GT”	Guatemala	1,2,3,4,5,6,7,8,9,10,11	1,2,3,4,5,6,7,8,9,10,11

Country Code	Country	STA Channels	Soft-AP Channels
“GU”	Guam	1,2,3,4,5,6,7,8,9,10,11	1,2,3,4,5,6,7,8,9,10,11
“GY”	Guyana	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“HK”	Hong Kong	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“HN”	Honduras	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“HT”	Haiti	1,2,3,4,5,6,7,8,9,10,11	1,2,3,4,5,6,7,8,9,10,11
“HU”	Hungary	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“ID”	Indonesia	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“IE”	Ireland	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“IL”	Israel	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“IN”	India	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“IR”	Iran, Islamic Republic of	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“IS”	Iceland	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“IT”	Italy	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“JM”	Jamaica	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“JO”	Jordan	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
"JP”	Japan	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“KE”	Kenya	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“KH”	Cambodia	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“KN”	Saint Kitts and Nevis	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“KP”	North Korea	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“KR”	South Korea	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“KW”	Kuwait	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“KY”	Cayman Islands	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“KZ”	Kazakhstan	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“LB”	Lebanon	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“LC”	Saint Lucia	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“LI”	Liechtenstein	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“LK”	Sri Lanka	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“LS”	Sesotho	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“LT”	Lithuania	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“LU”	Luxembourg	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“LV”	Latvia	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“MA”	Morocco	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“MC”	Monaco	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“MD”	Moldova	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13

Country Code	Country	STA Channels	Soft-AP Channels
“ME”	Montenegro	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“MF”	Saint-Martin (French part)	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“MH”	Marshall Islands	1,2,3,4,5,6,7,8,9,10,11	1,2,3,4,5,6,7,8,9,10,11
“MK”	Macedonia, Republic of	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“MN”	Mongolia	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“MO”	Macao	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“MP”	Northern Mariana Islands	1,2,3,4,5,6,7,8,9,10,11	1,2,3,4,5,6,7,8,9,10,11
“MQ”	Martinique	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“MR”	Mauritania	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“MT”	Malta	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“MU”	Mauritius	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“MV”	Maldives	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“MW”	Malawi	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“MX”	Mexico	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“MY”	Malaysia	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“NG”	Nigeria	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“NI”	Nicaragua	1,2,3,4,5,6,7,8,9,10,11	1,2,3,4,5,6,7,8,9,10,11
“NL”	Netherlands	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“NO”	Norway	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“NP”	Nepal	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“NZ”	New Zealand	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
"OM”	Oman	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“PA”	Panama	1,2,3,4,5,6,7,8,9,10,11	1,2,3,4,5,6,7,8,9,10,11
“PE”	Peru	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“PF”	French Polynesia	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“PG”	Papua New Guinea	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“PH”	Philippines	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“PK”	Pakistan	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“PL”	Poland	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“PM”	Saint Pierre and Miquelon	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“PR”	Puerto Rico	1,2,3,4,5,6,7,8,9,10,11	1,2,3,4,5,6,7,8,9,10,11
“PT”	Portugal	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“PW”	Palau	1,2,3,4,5,6,7,8,9,10,11	1,2,3,4,5,6,7,8,9,10,11
“PY”	Paraguay	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13

Country Code	Country	STA Channels	Soft-AP Channels
“QA”	Qatar	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“RE”	Reunion	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“RO”	Romania	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“RS”	Serbia	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“RU”	Russian Federation	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“RW”	Rwanda	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“SA”	Saudi Arabia	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“SE”	Sweden	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“SG”	Singapore	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“SI”	Slovenia	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“SK”	Slovak Republic	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“SN”	Senegal	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“SR”	Suriname	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“SV”	El Salvador	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“SY”	Syrian Arab Republic	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“TC”	Turks and Caicos Islands	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“TD”	Chad	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“TG”	Togo	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“TH”	Thailand	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“TN”	Tunisia	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“TR”	Turkey	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“TT”	Trinidad and Tobago	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“TW”	Taiwan	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“TZ”	Tanzania	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“UA”	Ukraine	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
"UG”	Uganda	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“US”	United States of America	1,2,3,4,5,6,7,8,9,10,11	1,2,3,4,5,6,7,8,9,10,11
“UY”	Uruguay	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“UZ”	Uzbekistan	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“VC”	Saint Vincent and Grenadines	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“VE”	Venezuela	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“VI”	Virgin Islands	1,2,3,4,5,6,7,8,9,10,11	1,2,3,4,5,6,7,8,9,10,11
“VN”	Vietnam	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13

Country Code	Country	STA Channels	Soft-AP Channels
“VU”	Vanuatu	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“WF”	Walls and Futuna Islands	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“WS”	Samoa	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“YE”	Yemen	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“YT”	Mayotte	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“ZA”	South Africa	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“ZW”	Zimbabwe	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“00”	Worldwide	1,2,3,4,5,6,7,8,9,10,11,12,13	1,2,3,4,5,6,7,8,9,10,11,12,13
“XX”		1,2,3,4,5,6,7,8,9,10,11	1,2,3,4,5,6,7,8,9,10,11

B.2 Programming

The power level setting for “ALL mode” is 0x0, and the setting of specific countries mode is 0x3. The power level is only the default value, so it is required to set according to the customer's specifications. Countries such as CA, CN, JP, KR, US are required to be specified in the manufacturing process by the customer.

In the DA16200 (DA16600) SDK, user can change the supporting “country code list” for their product. See Table 35.

- *FreeRTOS_SDK/core/system/src/common/main/sys_user_feature.c*

Table 35: Programming Example for Country Code

```
Const country_ch_powr_level_t cc_power_level[MAX_COUNTRY_CNT] =
{
/*  Country Code  1   2   3   4   5   6   7   8   9   10   11   12   13   14 */
/* --------------------------------------------------------------------------- */

/* Andorra: ETSI 1 ~ 13 */
  { "AD",  0x2, 0x2, 0x2, 0x2, 0x2, 0x2, 0x2, 0x2, 0x2, 0x2, 0x2, 0x2, 0x2, 0xF },
/* UAE: FCC  1 ~ 13 */
  { "AE",  0x2, 0x2, 0x2, 0x2, 0x2, 0x2, 0x2, 0x2, 0x2, 0x2, 0x2, 0x2, 0x2, 0xF },
/* Afghanistan: ETSI 1 ~ 13 */
  { "AF",  0x2, 0x2, 0x2, 0x2, 0x2, 0x2, 0x2, 0x2, 0x2, 0x2, 0x2, 0x2, 0x2, 0xF },
```

Appendix C How to Use J-Link Debugger

See Ref. [4].

Revision History

Revision	Date	Description
1.4	22-Dec-2021	Added description about fcCSP Low Power RTOS Image.
1.3	26-Nov-2021	Title was changed.
1.2	09-Nov-2021	TW Editorial.
1.1	25-Oct-2021	Added description about OTP.
1.0	13-Apr-2021	First 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.

DA16200 DA16600 FreeRTOS SDK Programmer Guide

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