

# User Manual

## SDK Programmer Guide

### UM-WI-002

#### Abstract

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

---

## SDK Programmer Guide

### Contents

<b>Abstract .....</b>	<b>1</b>
<b>Contents .....</b>	<b>2</b>
<b>Figures.....</b>	<b>5</b>
<b>Tables .....</b>	<b>6</b>
<b>1 References .....</b>	<b>7</b>
<b>2 Introduction.....</b>	<b>8</b>
2.1 Overview .....	8
Development Environment .....	8
2.2 Startup Main() .....	9
2.3 Startup System Applications .....	11
2.4 Startup User Applications.....	13
2.5 Write User Application.....	15
2.6 SDK Compilation .....	18
2.7 Make 4 MB SFLASH Images .....	19
2.8 Make fcCSP Low-Power SLIB Image .....	19
<b>3 Memory Map.....</b>	<b>20</b>
3.1 System Memory Map .....	20
3.2 Memory Types .....	20
3.3 Serial Flash Memory Map .....	21
<b>4 Peripheral Driver .....</b>	<b>23</b>
4.1 SPI Slave .....	23
4.1.1 Introduction .....	23
4.1.2 Application Programming Interface .....	24
4.1.3 Sample Code .....	24
4.2 SDIO Master .....	24
4.2.1 SDIO Introduction .....	24
4.2.2 Application Programming Interface .....	24
4.2.3 Sample Code .....	26
4.3 SDIO Slave .....	26
4.3.1 Introduction .....	26
4.3.2 Application Programmer Interface .....	26
4.3.3 Sample Code .....	27
4.4 I2C.....	27
4.4.1 I2C Master .....	27
4.4.2 I2C Slave .....	27
4.4.3 Application Programming Interface .....	29
4.4.4 Sample Code .....	30
4.5 SD/eMMC.....	30
4.5.1 Introduction .....	30
4.5.2 Application Programming Interface .....	31
4.5.3 Sample Code .....	32
4.6 PWM.....	33
4.6.1 Introduction .....	33

## SDK Programmer Guide

4.6.2	Application Programming Interface .....	33
4.6.3	Sample Code .....	34
4.7	ADC.....	35
4.7.1	Introduction .....	35
4.7.2	Application Programming Interface .....	36
4.7.3	Interrupt Description .....	39
4.7.4	Sample Code .....	40
4.8	GPIO .....	41
4.8.1	Introduction .....	41
4.8.2	Application Programming Interface .....	43
4.8.3	Sample Code .....	45
4.9	UART.....	45
4.9.1	Introduction .....	45
4.9.2	Application Programming Interface .....	46
4.9.3	Sample Code .....	48
4.10	SPI Master .....	49
4.10.1	Introduction .....	49
4.10.2	Application Programming Interface .....	49
4.10.3	Sample Code .....	51
4.11	Pulse Counter .....	51
4.11.1	Introduction .....	51
4.11.2	Application Programming Interface .....	52
<b>5</b>	<b>NVRAM .....</b>	<b>54</b>
5.1	Application Programming Interface .....	54
<b>6</b>	<b>HW Accelerators.....</b>	<b>55</b>
6.1	Set SRAM to Zero .....	55
6.1.1	Application Programming Interface .....	55
6.1.2	Sample Code .....	55
6.2	CRC Calculation.....	55
6.2.1	Application Programming Interface .....	55
6.2.2	Sample Code .....	55
6.3	Pseudo Random Number Generator (PRNG) .....	56
6.3.1	Application Programming Interface .....	56
6.3.2	Sample Code .....	56
6.4	Memory Copy Using DMA.....	56
6.4.1	Application Programming Interface .....	56
6.4.2	Sample Code .....	56
<b>7</b>	<b>Wi-Fi Interface Configuration .....</b>	<b>57</b>
7.1	Application Programming Interface .....	57
7.1.1	Integer Type Parameters .....	59
7.1.2	String Type Parameters.....	60
7.1.3	Sample Code .....	61
7.2	Soft-AP Configuration by Factory Reset.....	62
7.2.1	Configuration Data Structure Integer Type Parameters .....	63
7.2.2	How to Configure .....	64

## SDK Programmer Guide

7.3	Soft-AP Provisioning Protocol .....	65
7.3.1	Provisioning Specification .....	65
7.3.2	User Soft-AP Provisioning Application .....	67
7.3.2.1	Example User Soft-AP Provisioning Thread .....	67
7.3.2.2	Example Peer (Mobile) Application .....	68
<b>8</b>	<b>Tx Power Table Edit .....</b>	<b>72</b>
8.1	Tune Tx Power .....	72
8.2	Apply Tuned Tx Power to Main Image .....	73
<b>9</b>	<b>Tips .....</b>	<b>74</b>
9.1	Find/Optimize Stack Size for Your Application .....	74
9.2	Debug Stack Overflow .....	75
	<b>Appendix A Open-Source License .....</b>	<b>78</b>
	<b>Appendix B Country Code and Tx Power .....</b>	<b>79</b>
B.1	Country Code and Channels .....	79
B.2	Programming .....	84
	<b>Appendix C Doxygen Documents .....</b>	<b>85</b>
	<b>Appendix D How to use I-Jet debugger .....</b>	<b>88</b>
D.1	Notice to Use Debugger on IAR Workbench .....	88
D.2	I-Jet Debug Setting .....	88
D.3	J-Link Debug Setting .....	95
D.4	IAR build Setting .....	100
	<b>Revision History .....</b>	<b>101</b>

## SDK Programmer Guide

### Figures

Figure 1: IAR Embedded Workbench Project Configuration .....	8
Figure 2: Update IAR Embedded Workbench .....	9
Figure 3: Install IAR Embedded Workbench .....	9
Figure 4: Checking Version of IAR Embedded Workbench .....	9
Figure 5: Startup Files on IAR Project .....	10
Figure 6: Application on IAR Project .....	12
Figure 7: Results of Running the 'Hello World' Applications .....	15
Figure 8: Customer Project in IAR Workbench .....	15
Figure 9: Location of User codes .....	17
Figure 10: Add User Files to the IAR Project .....	17
Figure 11: Compile SDK on IAR Workbench .....	18
Figure 12: Build Success on IAR Embedded Workbench .....	18
Figure 13: Boot Logo with fcCSP-LP SLIB Image.....	19
Figure 14: System Memory Map .....	20
Figure 15: APB and System Peripherals Memory Map.....	21
Figure 16: PWM Block Diagram .....	33
Figure 17: ADC Control Block Diagram.....	35
Figure 18: TX Power Table.....	72
Figure 19: Tune Tx Power: Setup .....	72
Figure 20: Tune Tx Power: Choose Country Code .....	72
Figure 21: Tune Tx Power: Check Tx Power Indices .....	72
Figure 22: Tune Tx Power: Modify Tx Power Indices .....	73
Figure 23: TX power Table Source Code.....	73
Figure 24: Check Stack Size .....	74
Figure 25: IAR Debug Window for Stack Overflow .....	77
Figure 26: Doxygen Document of the DA16200 SDK .....	87
Figure 27: Connect I-Jet Debugger to the DA16200 EVB.....	88
Figure 28: Select Debugger Option.....	89
Figure 29: Debugger Setup Setting.....	90
Figure 30: Debugger Download Setting .....	91
Figure 31: I-Jet JTAG/SWD Setting .....	91
Figure 32: Trace Mode Setting .....	92
Figure 33: Rebuild SDK.....	92
Figure 34: Download and Debug.....	93
Figure 35: Pop-up Message .....	93
Figure 36: Download and Debug Windows .....	94
Figure 37: Break Point Window .....	94
Figure 38: Connect J-Link Debugger to the DA16200 EVB.....	95
Figure 39: Select Debugger Option.....	95
Figure 40: Debugger Setup Setting.....	96
Figure 41: Debugger Download Setting .....	97
Figure 42: J-Link/J-Trace JTAG/SWD Setting.....	97
Figure 43: Rebuild SDK.....	98
Figure 44: Download and Debug.....	98
Figure 45: Pop-Up Message.....	99
Figure 46: Download and Debug Windows .....	99
Figure 47: Break Point Window .....	100

## SDK Programmer Guide

### Tables

Table 1: 2 MB SFLASH Map .....	21
Table 2: 4 MB SFLASH Map .....	22
Table 3: SPI Interface API Elements .....	23
Table 4: SPI Slave Interface API Elements .....	24
Table 5: SDIO Interface API Elements .....	24
Table 6: SDIO Slave Pin Configuration .....	26
Table 7: SDIO Interface API Elements .....	26
Table 8: I2C Master Pin Configuration .....	27
Table 9: I2C Slave Pin Configuration .....	28
Table 10: I2C Interface API Elements .....	29
Table 11: SD/eMMC Master Pin Configuration .....	30
Table 12: SD/eMMC Interface API Elements .....	31
Table 13: PWM Pin Configuration .....	33
Table 14: PWM Interface API Elements .....	33
Table 15: AUX ADC Pin Configuration .....	35
Table 16: ADC Interface API Elements .....	36
Table 17: GPIO Pin Configuration .....	41
Table 18: The Status of GPIO PIN .....	42
Table 19: GPIO Interface API Elements .....	43
Table 20: UART Pin Configuration .....	45
Table 21: UART Interface API Elements .....	46
Table 22: SPI Master Pin Configuration .....	49
Table 23: SPI Interface API Elements .....	49
Table 24: NVRAM API Elements .....	54
Table 25: Wi-Fi Configuration API .....	57
Table 26: NVRAM Integer Type .....	59
Table 27: NVRAM String Type .....	60
Table 28: NVRAM Sample Code on STA mode .....	61
Table 29: NVRAM Sample Code on Soft-AP Mode .....	62
Table 30: Soft-AP Interface Code .....	63
Table 31: Soft-AP Configuration Code .....	64
Table 32: Provisioning Protocol Code .....	65
Table 33: Soft-AP Provisioning Thread Sample Code (TCP Sample) .....	67
Table 34: Soft-AP Provisioning Thread Sample Code for Peer Application (TCP Sample) .....	68
Table 35: TCP Client Sample Code .....	74
Table 36: Corrupted Stack Overflow .....	75
Table 37: Force Stack Overflow .....	76
Table 38: Stack Overflow Debug Code .....	76
Table 39: Country Code .....	79
Table 40: Programming Example for Country Code .....	84

---

**SDK Programmer Guide****1 References**

- [1] DA16200, Datasheet, Dialog Semiconductor
- [2] DA16200, EVK User Manual, User Manual, Dialog Semiconductor
- [3] DA16200, Example Application Manual, User Manual, Dialog Semiconductor

## SDK Programmer Guide

## 2 Introduction

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

### 2.1 Overview

The DA16200 SDK has eight folders:

- **build:** Build scripts, temporary build artifacts, or environment files
- **customer:** IAR project files and applications for customer.
- **doc:** user documents (user guides, programmer guides, etc.)
- **img:** to which the images built / pre-compiled are copied
- **lib:** to which the pre-compiled lib files (.a) are saved
- **sample:** to demonstrate common use cases of what the DA16200 SDK provides
- **src:** source codes
- **src\_tim:** source codes for TIM SDK
- **version:** version files to include when Image created

The DA16200 SDK may be provided with different features per customer or per certain applications so the source code configuration and the system libraries can differ according to the customer / application requirements. Dialog Semiconductor pre-compiles the system libraries with the relevant features enabled before the SDK is packaged. As a result, the customer can modify the pre-compiled libraries. Features are defined in `customer\main\inc\config_XXX_sdk.h` (the file name may follow its reference type) where users can enable / disable some features.

#### 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 IAR project for the DA16200 SDK is shown in [Figure 1](#). There is the possibility to add new user application files to the existing group `Customer_Apps` under the `customer_app` project. There is also the possibility to create your own group, to which you can add files in the “Customer” folder.

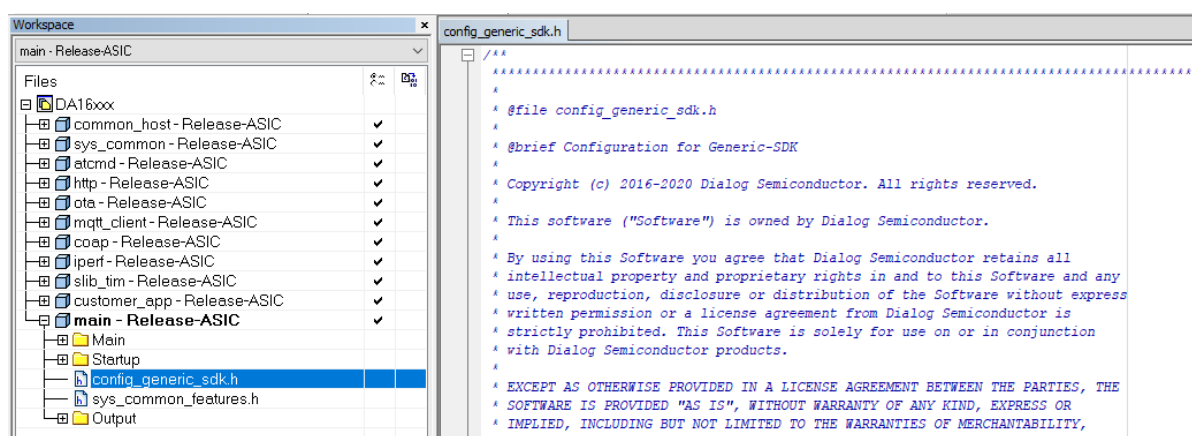


Figure 1: IAR Embedded Workbench Project Configuration

### Development Environment

The DA16200 SDK only supports the IAR Embedded Workbench to build a project. Users with an IAR license can download a specific version of IAR Embedded Workbench from the IAR website.



## SDK Programmer Guide

### NOTE

Due to compiler compatibility, the DA16200 SDK user should use the exact version of IAR Embedded Workbench, which is version **7.30.4**.

1. Open the URL <https://www.iar.com/support/customer-care/my-pages>.
2. After successful login, click on **Find updates**. See Figure 2.

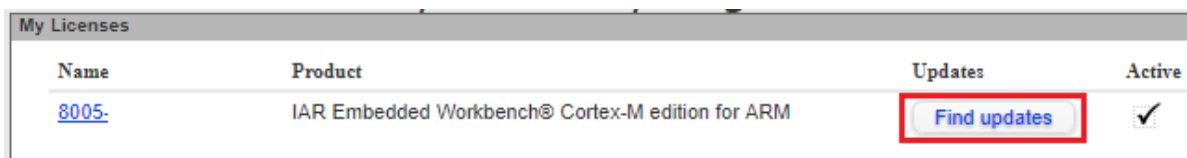


Figure 2: Update IAR Embedded Workbench

3. Click on **Other Versions** to download an old installer version. See Figure 3.

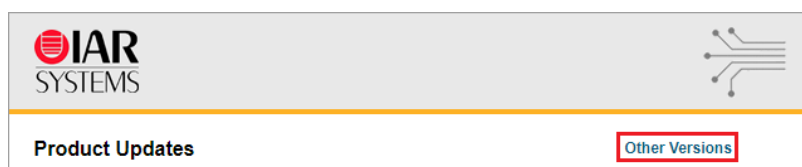


Figure 3: Install IAR Embedded Workbench

4. Click on **7.30** to go to the download page. See Figure 4.

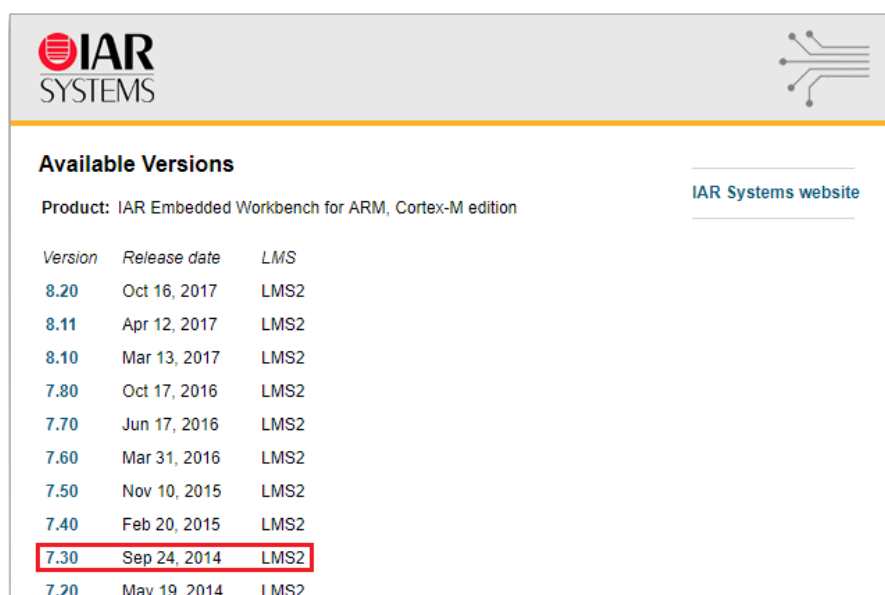


Figure 4: Checking Version of IAR Embedded Workbench

5. Choose version **7.30.4.XXX**.

## 2.2 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 as Wi-Fi IoT device

## SDK Programmer Guide

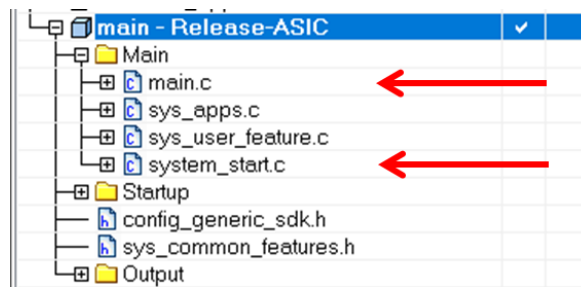


Figure 5: Startup Files on IAR Project

[ ~/src/main.c ]

```
int main(char init_state)
{
    int status;

    /* clear RETMEM GPIO_PULLUP_INFO */
    clear_retmem_gpio_pullup_info();

    /* Configure Pin-Mux of the DA16200 */
    config_pin_mux();

    /*
     * 1. Restore saved GPIO PINs
     * 2. RTC PAD connection
     */
    __GPIO_RETAIN_HIGH_RECOVERY();

    /* Logo Display */
    if (!get_boot_mode())
        version_display(0, NULL);

    /* Initialize the FC9K's Console */
    if (Create_ConsoleThread() == FALSE)
        return FALSE;

    /* Entry point for customer main */
    if (init_state == TRUE)
    {
        status = 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 the Wi-Fi operation. The following happens:

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

## SDK Programmer Guide

- Start of user applications in `start_user_apps()`

[ `~/customer/main/src/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 parameters for system running */
    set_sys_config();

    /* Initialize WLAN interface */
    wlaninit();

    /* Setup WPS button */
    if (check_wps_button(wps_btn, wps_led, wps_btn_chk_time) == TX_TRUE)
    {
        wps_setup(TX_NULL);
    }

    /* Start GPIO polling thread */
    start_gpio_thread();

    set_dpm_abnorm_user_wakeup_interval();

    /* Regist User DPM application before start system application */
    regist_user_apps_to_DPM_manager();

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

    /*
     * Entry point of user's applications
     *: defined in user_apps_table.c
     */
    start_user_apps();

    return TRUE;
}
```

### NOTE

The features supported in the SDK are defined in file `config_xxxx_sdk.h` (i.e. `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.

## 2.3 Startup System Applications

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

## SDK Programmer Guide

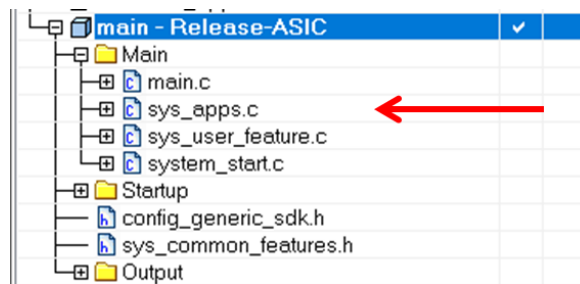


Figure 6: Application on IAR Project

```
[ ~/src/sys_apps/src/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)
{
    ... ..

    /* Start network independent applications */
    create_sys_apps(sysmode, FALSE);

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

    ... ..

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

        i++;
        tx_thread_sleep(1);
    }

    ... ..

    /* Check IP address status */
    while (check_net_ip_status(iface) != NX_SUCCESS)
    {
        tx_thread_sleep(1);
    }
}
```

## SDK Programmer Guide

```

    /* Start network dependent applications */
    create_sys_apps(sysmode, TRUE);
}

```

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

```

[ ~/src/sys_apps/src/sys_apps.c ]
static const app_thread_info_t sys_apps_table[] =
{
/* name, func, stack_size, pri, net_flag, dpm_flag, port_no, sys_mode */

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

    ... ..

    #if defined (__SUPPORT_MQTT__)
        { APP_MQTT_SUB, mqtt_auto_start, 1024, USER_PRI_APP(1),
          TRUE, TRUE, UNDEF_PORT, RUN_STA_MODE },
    #endif // __SUPPORT_MQTT__

    #if defined (__HTTP_SVR_AUTO_START__)
        { APP_HTTP_SVR, auto_run_http_svr, 1024, USER_PRI_APP(1),
          TRUE, FALSE, HTTP_SVR_PORT, RUN_AP_MODE },
    #endif // __HTTP_SVR_AUTO_START__

    #if defined (__HTTPS_SVR_AUTO_START__)
        { APP_HTTPS_SVR, auto_run_https_svr 2048, USER_PRI_APP(1),
          TRUE, FALSE, HTTP_SVR_PORT, RUN_AP_MODE },
    #endif // __HTTPS_SVR_AUTO_START__

    #if defined (__DPM_MDNS_AUTO_START__)
        { APP_MDNS, thd_mdns_service, 1024, USER_PRI_APP(1),
          TRUE, TRUE, MULTICAST_PORT, RUN_ALL_MODE },
    #endif // __DPM_MDNS_AUTO_START__

    ... ..

    /***** 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 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.

## 2.4 Startup User Applications

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

The user applications also run in two parts:

- Applications that should be executed regardless of network settings

## SDK Programmer Guide

```
[ ~/src/sys_apps/src/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

```
[ ~/src/sys_apps/src/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 SDK provides a “hello\_world” application by default.

```
[ ~/customer/apps/src/user_apps.c ]
const app_thread_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, 1024, USER_PRI_APP(0),
          FALSE, FALSE, UNDEF_PORT, RUN_ALL_MODE },
        { HELLO_WORLD_2, customer_hello_world_2, 1024, USER_PRI_APP(0), TRUE,
          FALSE, UNDEF_PORT, RUN_ALL_MODE },
    #endif // __SUPPORT_HELLO_WORLD__

    { NULL, NULL, 0, 0, FALSE, FALSE, UNDEF_PORT, 0 }
};
```

## SDK Programmer Guide

- 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

```
System Mode : Station Only (0)
>>> DA16XXX supplicant Ver1.00-20170213-01
>>> MAC address (sta0) : ec:9f:0d:9f:fd:36
>>> sta0 interface add OK
>>> Start STA mode...

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

>>> Selected BSS 70:5d:cc:53:29:b6 ssid='Bright_iptime' (-25)
>>> Network Interface (wlan0) : UP
>>> Associated with 70:5d:cc:53:29:b6
Connection COMPLETE to 70:5d:cc:53:29:b6
-- DHCP Client WLAN0: SEL
-- DHCP Client WLAN0: REQ
-- DHCP Client WLAN0: BOUND
    Assigned addr : 192.168.0.23
    netmask       : 255.255.255.0
    gateway       : 192.168.0.1
    DNS addr      : 210.220.163.82

    DHCP Server IP : 192.168.0.1
    Lease Time     : 02h 00m 00s
    Renewal Time   : 01h 00m 00s

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

Figure 7: Results of Running the 'Hello World' Applications

## 2.5 Write User Application

The DA16200 SDK provides an independent customer project named as **customer\_app**. See Figure 8. And in the SDK, the user can add new application code in the folder `~/customer/apps/src` and can add a newly written application file in the project **customer\_app** such as `user_app.c`.

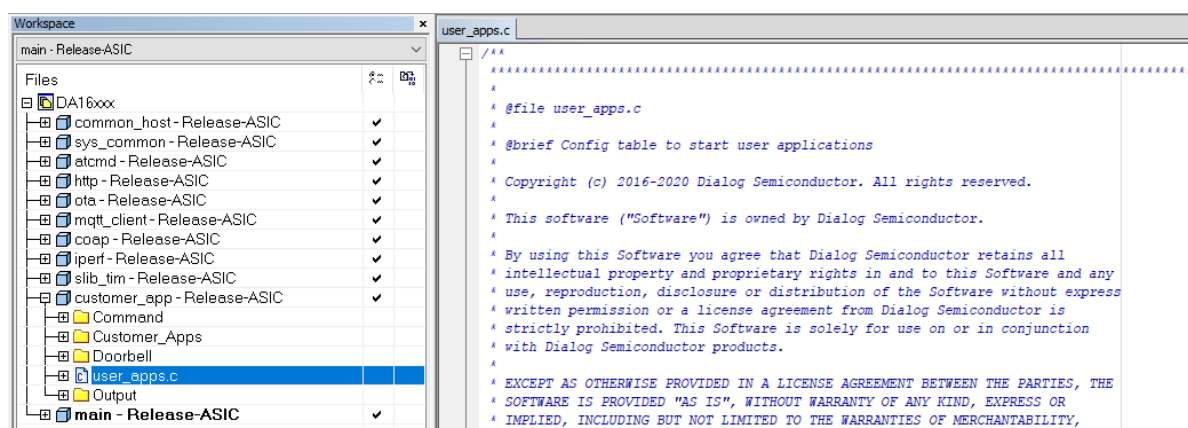


Figure 8: Customer Project in IAR Workbench

If the user needs to change to a new SDK, copy the contents of the `~/customer/apps/src` folder to a new SDK and `~/build/customer_apps.ewp` which is the build compile configuration file for the **customer\_app** project.

## SDK Programmer Guide

The DA16200 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 starts.

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

```
[ ~/src/sys_apps/inc/application.h ]
typedef struct _app_thread_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 the DA16200
    int run_sys_mode;
} app_thread_info_t;
```

- name [ThreadX feature] Unique thread name  
This name is also used to register to DPM sub-system
- entry\_function [ThreadX feature] Thread entry point
- stksize [ThreadX feature] Stack size of thread
- priority [ThreadX feature] Thread running priority
- net\_chk\_flag [DA16200 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 feature] Indicate if the user thread uses the DPM function
- port\_no [DA16200 feature] 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 feature] Runs a Wi-Fi mode (STA / Soft-AP). The application runs only the specified Wi-Fi mode



## SDK Programmer Guide

To add user application code in the DA16200 SDK:

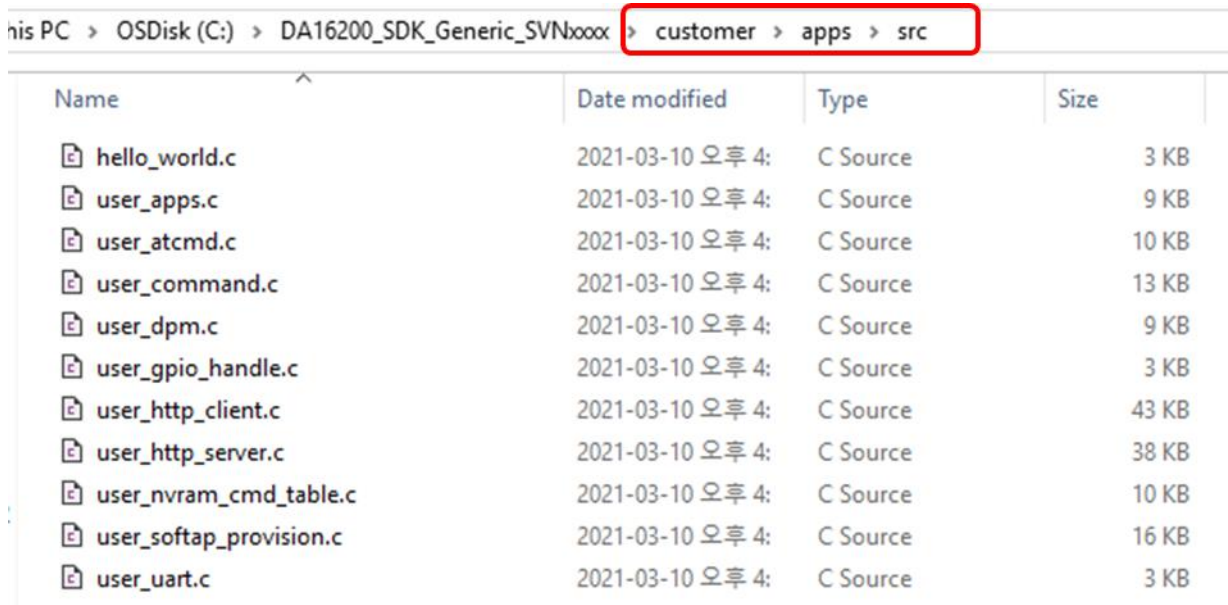


Figure 9: Location of User codes

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

1. Add the written user code files to the IAR project. See [Figure 10](#).

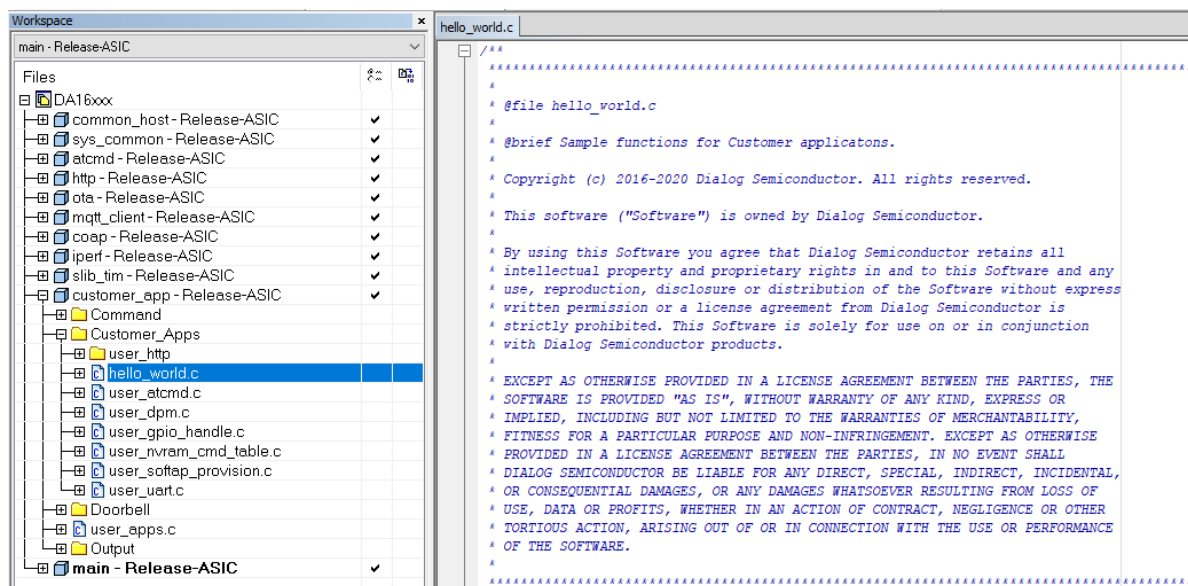
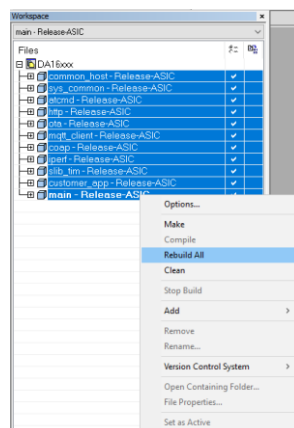


Figure 10: Add User Files to the IAR Project

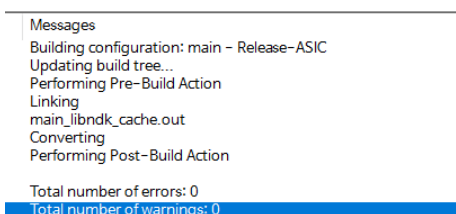
## SDK Programmer Guide

### 2.6 SDK Compilation

After an application is written, right-click on the project name in the IAR Embedded Workbench workspace and run command **Make** or **Rebuild All**. If you compile for the first time, then the advice is to run command **Clean** first. See [Figure 11](#).



**Figure 11: Compile SDK on IAR Workbench**



**Figure 12: Build Success on IAR Embedded Workbench**

If the build is successful, then there are three binary images created in folder “~/SDK/img”. The names of the image files are:

- RTOS : DA16200\_RTOS\_GEN01-01-XXXXX-000000.img
- System Library : DA16200\_SLIB\_GEN01-01-YYYYY-000000.img
- 2nd Bootloader : DA16200\_BOOT-GEN01-01-ZZZZZ-000000\_W25Q32JW.img

For more information about the firmware download, see document DA16200 EVK User Manual [\[2\]](#).

## SDK Programmer Guide

### 2.7 Make 4 MB SFLASH Images

The DA16200 SDK basically supports 2 MB SFLASH memory map. To create an image for a 4 MB SFLASH memory map using the DA16200 SDK, change some files for 4 MB memory map as follows, and then execute SDK Compilation from Section 2.6.

- 2<sup>nd</sup> Bootloader file : ~/SDK/build/SBOOT/image/DA16xxx\_ueboot.bin.4MB  
→ ~/SDK/build/SBOOT/image/DA16xxx\_ueboot.bin
- Config file : ~/SDK/build/SBOOT/cmconfig/fc9ktpmconfig.cfg.W25Q32JW(4MB)  
→ ~/SDK/build/SBOOT/cmconfig/fc9ktpmconfig.cfg
- Load script file : ~/SDK/build/ldscripts/DA16xxx\_rtos\_cache.icf.4MB  
→ ~/SDK/build/ldscripts/DA16xxx\_rtos\_cache.icf
- Macro file : ~/SDK/build/macro/da16200\_asic\_cache.mac.4MB  
→ ~/SDK/build/macro/da16200\_asic\_cache.mac
- Compile feature : ~/SDK/customer/main/inc/config\_generic\_sdk.h  
#undef \_\_FOR\_4MB\_SFLASH\_\_ → #define \_\_FOR\_4MB\_SFLASH\_\_

### 2.8 Make fcCSP Low-Power SLIB Image

The DA16200 SDK provides a QFN-type Ram Library SFLASH image file. After a compilation is made with the DA16200 SDK, the QFN-type SLIB image with filename **DA16200\_SLIB-GEN01-01-XXXXX-000000.img** is created in folder ~/SDK/img/.

To create a RAM Library image for the fcCSP Low-Power chipset with the DA16200 SDK, change the files mentioned below, and then do the SDK Compilation instructions given in Section 2.6.

- binary file : ~/SDK/build/SBOOT/image/DA16xxx\_slrib\_ramlib.bin.fcCSP\_LP  
→ ~/SDK/build/SBOOT/image/DA16xxx\_slrib\_ramlib.bin  
: ~/SDK/build/SBOOT/image/DA16xxx\_slrib\_ramlib.rtm.fcCSP\_LP  
→ ~/SDK/build/SBOOT/image/DA16xxx\_slrib\_ramlib.rtm

#### NOTE

To make fcCSP image in the DA16200 Generic SDK, Customer/Developer have to remove the RamLib binary files if exist in ~/SDK/build/asic/Release/Exe folder.

~/SDK/build/asic/Release/Exe/DA16xxx\_slrib\_ramlib.bin  
~/SDK/build/asic/Release/Exe/DA16xxx\_slrib\_ramlib.out

- Compile feature : ~/SDK/customer/main/inc/sys\_common\_features.h  
#undef \_\_FOR\_FCCSP\_SDK\_\_ → #define \_\_FOR\_FCCSP\_SDK\_\_

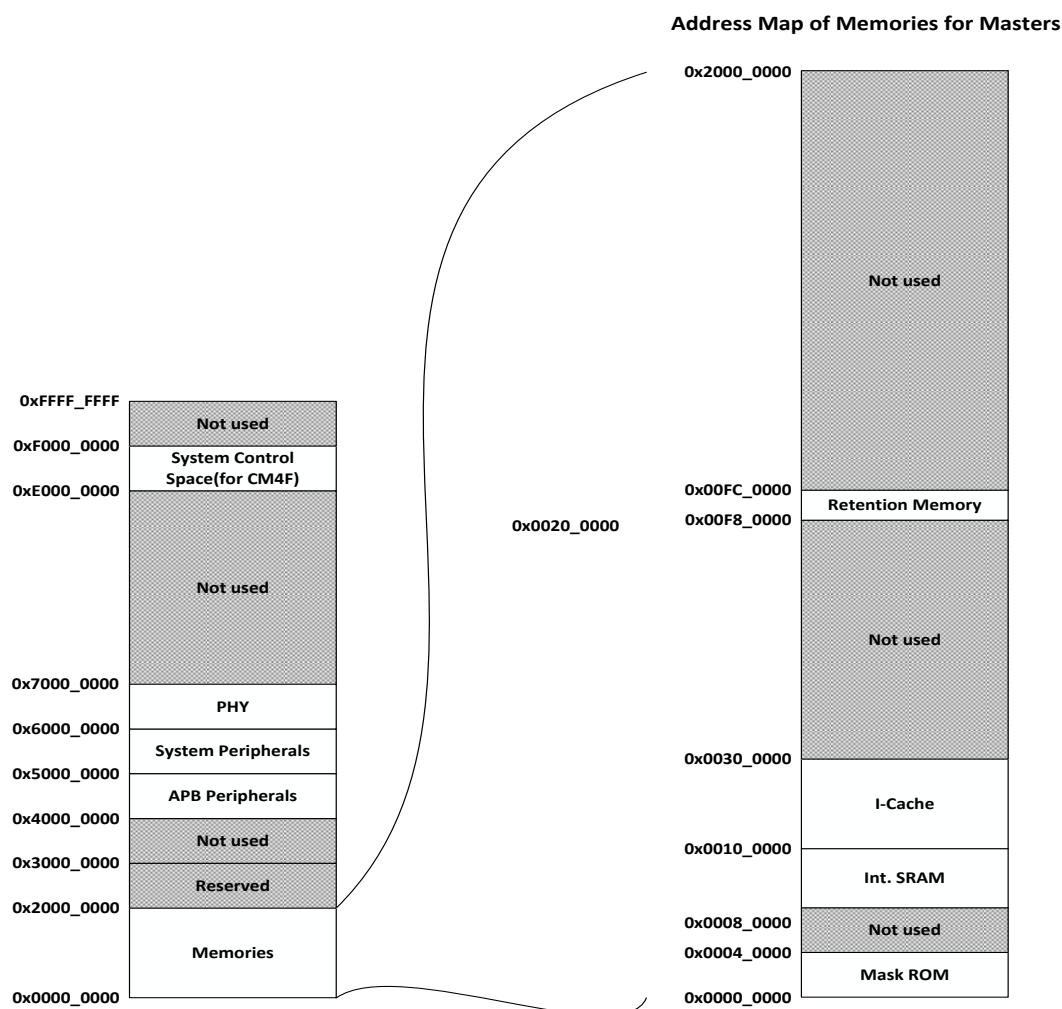
After the compilation is finished, load the SLIB image and RTOS image into the SFLASH and boot the system. To distinguish it from the QFN type, it shows SDK Version information as "V2.3.X.0 CSP LP" as below when booting. See Figure 13.

```
*****
*                               DA16200 SDK Information
*                               -----
*
* - CPU Type       : Cortex-M4 (120MHz)
* - OS Type        : ThreadX 5.7
* - Serial Flash   : 2 MB
* - SDK Version    : V2.3.3.0 CSP LP
* - F/W Version    : RTOS-GEN01-01-13314-000000
*                  : SLIB-GEN01-01-12283-000000
* - F/W Build Time : Dec 14 2020 15:42:29
* - Boot Index     : 0
*
*****
```

Figure 13: Boot Logo with fcCSP-LP SLIB Image

## 3 Memory Map

### 3.1 System Memory Map



**Figure 14: System Memory Map**

### 3.2 Memory Types

The DA16200 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 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.

PHY is a region for 802.11 MAC HW. The APB and System peripherals region is detailed in [Figure 15](#). In addition to these memory regions, there is an external Serial Flash memory region provided on the EVB. Since Serial Flash memory is connected to an internal SPI (Serial to Peripheral

## SDK Programmer Guide

Interface) Master Controller, which has a separate DMA controller, it is not shown in the memory map of Figure 15. See Section 3.3 for more information about the serial flash memory map.

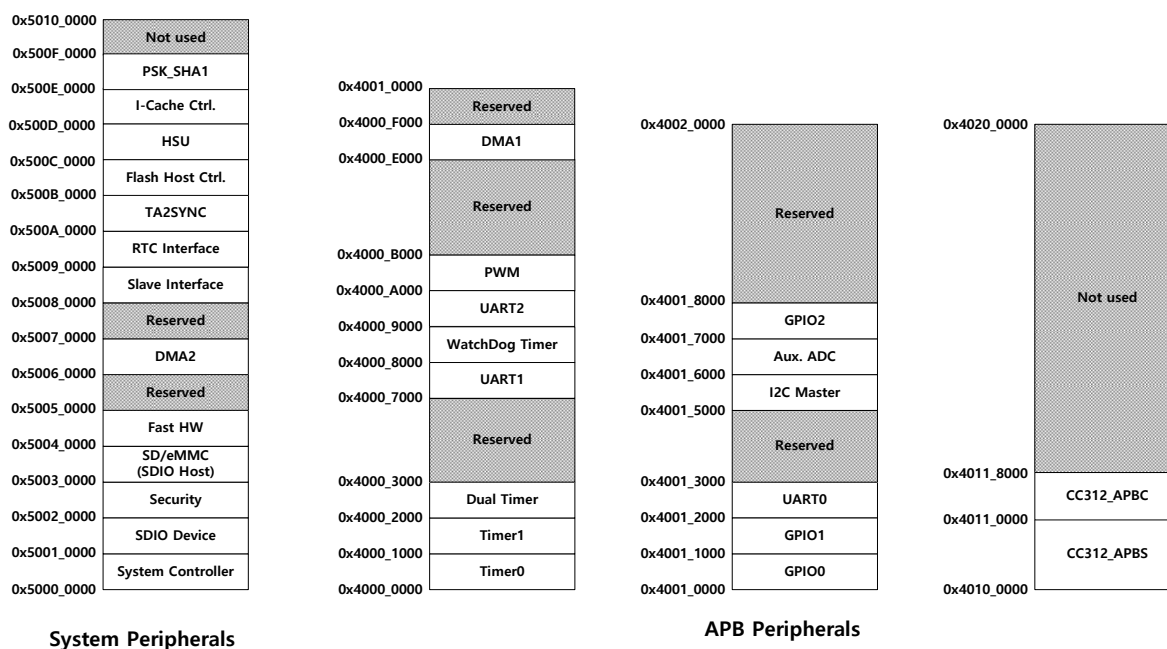


Figure 15: APB and System Peripherals Memory Map

### 3.3 Serial Flash Memory Map

The DA16200 supports two types of SFLASH sizes: 2 MB and 4 MB. The default size of the DA16200 SDK is 2 MB SFLASH.

Table 1: 2 MB SFLASH Map

DA16200 2 MB SFLASH Map - Standard		
0x0000_0000	2nd Bootloader	36 kB
0x0000_9000	Boot Index	4 kB
0x0000_A000	RTOS #0	924 kB
0x000F_1000	SLIB #0 (RamLib + TIM)	52 kB
0x000F_E000	RTOS #1	924 kB
0x001E_5000	SLIB #1 (RamLib + TIM)	52 kB
0x001F_2000	User Area	12 kB
0x001F_5000	Debug/RMA Certificate	4 kB
0x001F_6000	TLS Certificate key #0	16 kB
0x001F_A000	TLS Certificate key #1	16 kB
0x001F_E000	NVRAM#0	4 kB
0x001F_F000	NVRAM#1	4 kB

## SDK Programmer Guide

**Table 2: 4 MB SFLASH Map**

DA16200 4 MB SFLASH Map - Standard		
0x0000_0000	2nd Bootloader	36 kB
0x0000_9000	Boot Index	4 kB
0x0000_A000	RTOS #0	1536 kB
0x0018_A000	SLIB #0 (RamLib + TIM)	64 kB
0x0019_A000	User Area #0	364 kB
0x001F_5000	Debug/RMA Certificate	4 kB
0x001F_6000	TLS Certificate key #0	16 kB
0x001F_A000	TLS Certificate key #1	16 kB
0x001F_E000	NVRAM#0	4 kB
0x001F_F000	NVRAM#1	4 kB
0x0020_0000	RTOS #1	1536 kB
0x0038_0000	SLIB #1 (RamLib + TIM)	64 kB
0x0039_0000	User Area #1	448 kB

### NOTE

The size of the RTOS # 0 and # 1 images are the size of RTOS images size plus the interrupt vector table (5 kB size). So, the size of the actual RTOS image is the size excluding 5 kB from the total size.

## 4 Peripheral Driver

### NOTE

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

### 4.1 SPI Slave

#### 4.1.1 Introduction

The SPI slave interface gives support to control the DA16200 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 3: 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	

## SDK Programmer Guide

### 4.1.2 Application Programming Interface

**Table 4: SPI Slave Interface API Elements**

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

### 4.1.3 Sample Code

See the DA16200 Example Application Guide [\[3\]](#).

## 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 the DA16200 Datasheet [\[1\]](#).

### 4.2.2 Application Programming Interface

**Table 5: SDIO Interface API Elements**

<b>HANDLE EMMC_CREATE(void);</b>		
Parameter	void	Void
Return		If succeeded return handle for such device, if failed return NULL
Function create handle. If memory allocation failed, return NULL		
<b>int EMMC_INIT(HANDLE handler)</b>		
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		
<b>int EMMC_CLOSE(HANDLE handler)</b>		
Parameter	handler	Device handle
Return		If succeeded return ERR_NONE
<b>int SDIO_ENABLE_FUNC(HANDLE handler, UINT32 func_num)</b>		
Parameter	handler	Device handle
	func_num	Function number to enable
Return		If succeeded return ERR_NONE
<b>int SDIO_DISABLE_FUNC(HANDLE handler, UINT32 func_num)</b>		
Parameter	handler	Device handle



## SDK Programmer Guide

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

<b>int SDIO_READ_BYTE(HANDLE handler, UINT32 func_num, UINT32 addr, UINT8 *data)</b>		
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
<b>int SDIO_WRITE_BYTE(HANDLE handler, UINT32 func_num, UINT32 addr, UINT8 *data)</b>		
Parameter	handler	Device handle
	func_num	Function number
	addr	Address in the function
	data	Data pointer
Return		If succeeded return ERR_NONE
<b>int SDIO_READ_BURST(HANDLE handler, UINT32 func_num, UINT32 addr, UINT32 incr_addr, UINT8 *data, UINT32 count, UINT32 blksize)</b>		
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
	blksize	Block size
Return		If succeeded return ERR_NONE. If failed, Error Code return, see also EMMC.h
<b>int SDIO_WRITE_BURST(HANDLE handler, UINT32 func_num, UINT32 addr, UINT32 incr_addr, UINT8 *data, UINT32 count, UINT32 blksize)</b>		
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

## SDK Programmer Guide

int SDIO_READ_BYTE(HANDLE handler, UINT32 func_num, UINT32 addr, UINT8 *data)		
	count	Count of blocks
	blksz	Block size
Return		If succeeded return ERR_NONE

### 4.2.3 Sample Code

See the DA16200 Example Application Guide [\[3\]](#).

## 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 6: 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 the DA16200 Datasheet [\[1\]](#).

### 4.3.2 Application Programmer Interface

**Table 7: SDIO Interface API Elements**

UINT32 SDIO_SLAVE_INIT(void)		
Parameter	Void	Void
Return		return 0
Description		SDIO Slave Initialization

## SDK Programmer Guide

<b>UINT32 SDIO_SLAVE_INIT(void)</b>		
<b>void SDIO_SLAVE_CALLBACK_REGISTER(void (* p_rx_callback_func)(UINT32 status))</b>		
Parameter	p_rx_callback_func	The callback function to use the offload protocol
Return		void
Description		SDIO Slave callback registration
<b>void SDIO_SLAVE_CALLBACK_DEREGISTER(void)</b>		
Parameter	void	void
Return		void
Description		SDIO Slave callback de-registration
<b>void SDIO_SLAVE_DEINIT (void)</b>		
Parameter	void	void
Return		void
Description		SDIO Slave de-initialization

### 4.3.3 Sample Code

See the DA16200 Example Application Guide [3].

## 4.4 I2C

### 4.4.1 I2C Master

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

Table 8 shows the pin definition of the I2C master interface in GPIO Pin Configuration.

**Table 8: 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 the DA16200 Datasheet [1].

### 4.4.2 I2C Slave

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

The pin mux condition is defined in Table 9. The I2C slave interface also supports the standard (100 kbps) or fast (400 kbps) transmission speeds.

## SDK Programmer Guide

Table 9: 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 the DA16200 Datasheet [\[1\]](#).

## SDK Programmer Guide

## 4.4.3 Application Programming Interface

Table 10: 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

## SDK Programmer Guide

<b>int DRV_I2C_IOCTL(HANDLE handler, UINT32 cmd, VOID *data)</b>		
	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
<b>int DRV_I2C_READ(HANDLE handler, VOID *p_data, UINT32 p_dlen, UINT32 addr_len,UINT32 dummy)</b>		
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
<b>Int DRV_I2C_CLOSE(HANDLE handler);</b>		
Parameter	handler	Device handle to close
Return		If succeeded return TRUE, if failed return FALSE
Description		I2C driver close
<b>void DRV_I2C_REGISTER_INTERRUPT (HANDLE handler);</b>		
Parameter	handler	Device handle to register Interrupt Handler
Return		NULL
Description		I2C Interrupt Registration

### 4.4.4 Sample Code

See the DA16200 Example Application Manual [\[3\]](#).

## 4.5 SD/eMMC

### 4.5.1 Introduction

The SD/eMMC host IP has a function for the DA16200 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 11](#).

**Table 11: 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

## SDK Programmer Guide

Pin Name	Pin Number		I/O	Function Name
	QFN	fcCSP		
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 the DA16200 Datasheet [1].

### 4.5.2 Application Programming Interface

**Table 12: 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
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

## SDK Programmer Guide

<b>HANDLE EMMC_CREATE(void)</b>		
Return	If succeeded return TRUE, if failed return FALSE	
Description		
<b>void EMMC_SEND_CMD_RES(HANDLE handler, UINT32 cmd, UINT32 cmd_arg, UINT32 *rsp)</b>		
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 <code>rsp</code>	
<b>int EMMC_IOCTL(HANDLE handler, UINT32 cmd, VOID *data)</b>		
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	
<b>int EMMC_CLOSE(HANDLE handler)</b>		
Parameter	handler	Device handle
Return	If succeeded return ERR_NONE	
Description	EMMC driver close command	

### 4.5.3 Sample Code

See the DA16200 Example Application Manual [\[3\]](#).



## 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).

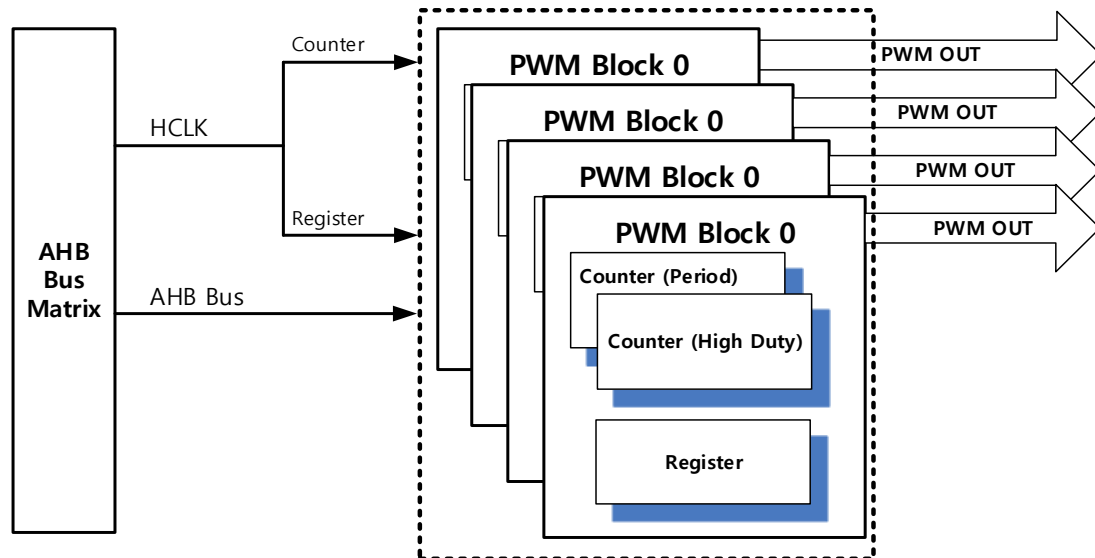


Figure 16: PWM Block Diagram

Table 13: 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 the DA16200 Datasheet [1].

### 4.6.2 Application Programming Interface

Table 14: PWM Interface API Elements

HANDLE DRV_PWM_CREATE(UINT32 dev_id)		
Parameter	dev_id	Device number to create handle
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 micro second
	Hduty_percent	Output high time in percentage while every 1 cycle

## SDK Programmer Guide

HANDLE DRV_PWM_CREATE(UINT32 dev_id)		
	dummy	TBD
Return		If succeeded return TRUE, if failed return FALSE
Description		Enable PWM block in the DA16200 with specified parameters $\text{period} = (((\text{period\_us} * 10) * (\text{clock} / 1000000)) / 10) - 1;$ // minimum system clock 1mhz $\text{hduty} = (((\text{period} + 1) * \text{hduty\_percent}) / 100) - 1;$
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
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 the DA16200 Example Application Manual [\[3\]](#).

## SDK Programmer Guide

### 4.7 ADC

#### 4.7.1 Introduction

The DA16200 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 17](#) and [Table 15](#).

The DA16200 has an external sensor wake-up function that uses the analog input signal through an Aux ADC. Even in sleep mode, the Aux ADC detects the change of an external analog signal, wakes up from sleep mode, and converts the DA16200 into a normal operation. This function can be used in up to four channels. Also, when multiple external sensors are used, analog signals are detected while the channels are automatically changed. For example, if all four channels are set as input sources, which have their threshold register respectively, the channels are measured sequentially from 0 to 3.

If one of the four values exceed the allowed range of values set by the threshold register, the DA16200 awakes from the sleep mode. The value setting of the input change can be either over threshold or under threshold.

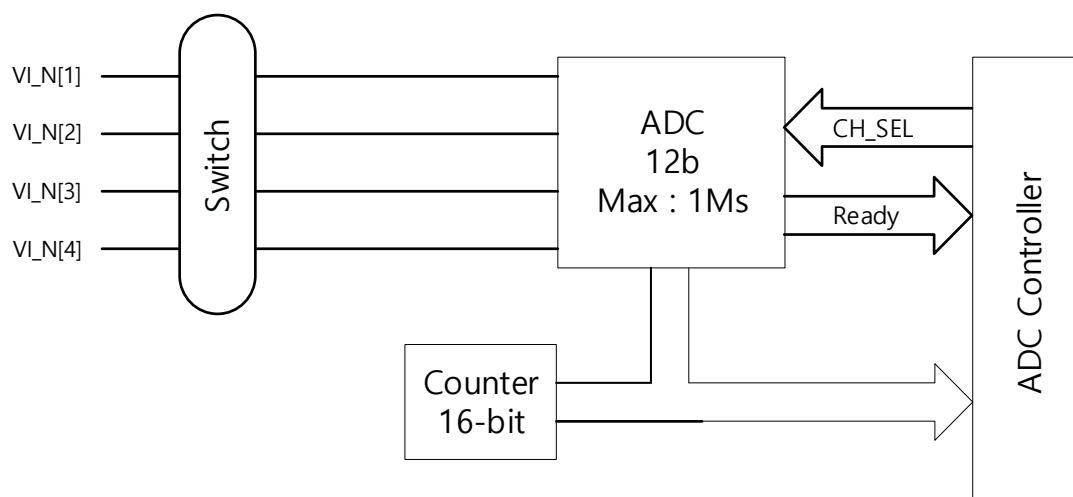


Figure 17: ADC Control Block Diagram

Table 15: 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 the DA16200 Datasheet [\[1\]](#).

## SDK Programmer Guide

### 4.7.2 Application Programming Interface

**Table 16: ADC Interface API Elements**

<b>HANDLE DRV_ADC_CREATE(UINT32 dev_id)</b>		
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
<b>int DRV_ADC_INIT(HANDLE handler, unsigned int use_timestamp)</b>		
Parameter	handler	Device handle to initialize
Return		If succeeded return TRUE, if failed return FALSE
Description		ADC Initialization command
<b>Int DRV_ADC_IOCTL(HANDLE handler, UINT32 cmd, VOID *data)</b>		
Parameter	handler	N/A
	cmd	N/A
	data	N/A
Return		N/A
Description		ADC IOCTL command
<b>int DRV_ADC_START(HANDLE handler, UINT32 divider12, UINT32 dummy)</b>		
Parameter	handler	Device handle to start
	divider12	$F_s = \text{sys\_clk} / 15 / (\text{div12} + 1)$
Return		If succeeded return TRUE, if failed return FALSE
Description		ADC start command
<b>int DRV_ADC_STOP(HANDLE handler, UINT32 dummy)</b>		
Parameter	handler	Device handle to stop
Return		If succeeded return TRUE, if failed return FALSE
Description		ADC stop command
<b>Int DRV_ADC_CLOSE(HANDLE handler)</b>		
Parameter	handler	Device handle to close
Return		If succeeded return TRUE, if failed return FALSE
Description		ADC driver close
<b>int DRV_ADC_READ(HANDLE handler, UINT32 channel, UINT32 *data, UINT32 dummy)</b>		
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

## SDK Programmer Guide

<b>HANDLE DRV_ADC_CREATE(UINT32 dev_id)</b>		
<b>int DRV_ADC_READ_DMA(HANDLE handler, UINT32 channel, UINT16 *p_data, UINT32 p_dlen, UINT32 dummy)</b>		
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
<b>int DRV_ADC_ENABLE_CHANNEL(HANDLE handler, UINT32 channel, unsigned int sel_adc, UINT32 dummy)</b>		
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
<b>int DRV_ADC_SET_INTERRUPT(HANDLE handler, UINT32 channel, UINT32 enable, UINT32 type, UINT32 dummy)</b>		
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
<b>int DRV_ADC_SET_THD_VALUE(HANDLE handler, UINT32 type, UINT32 enable, UINT32 thd, UINT32 dummy);</b>		
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

## SDK Programmer Guide

<b>int DRV_ADC_WAIT_INTERRUPT(HANDLE handler, UNSIGNED *mask_evt);</b>		
Parameter	handler	Device handle
	*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

<b>int DRV_ADC_SET_THRESHOLD(HANDLE handler, UNSIGNED channel, UNSIGNED threshold, UNSIGNED mode)</b>		
Parameter	handler	Device handle
	channel	Channel to set threshold
	threshold	Interrupt threshold. 0 ~ 4095 range
	mode	<ul style="list-style-type: none"> <li>ADC_RTC_THRESHOLD_TYPE_OVER : Wake up when adc value is bigger than threshold value</li> <li>ADC_RTC_THRESHOLD_TYPE_UNDER: Wake up when adc value is smaller than threshold value</li> </ul>
Return		If succeeded return TRUE, if failed return FALSE
Description		Set ADC interrupt threshold set command in sleep mode

<b>int DRV_ADC_SET_DELAY_AFTER_WKUP(HANDLE handler, UNSIGNED delay1, UNSIGNED delay2)</b>		
Parameter	handler	Device handle
	delay1	[0] fixed
	delay2	[0~59] Delay = delay2 x (8 / 32768Hz)(us)

## SDK Programmer Guide

<b>int DRV_ADC_SET_DELAY_AFTER_WKUP(HANDLE handler, UNSIGNED delay1, UNSIGNED delay2)</b>	
Return	If succeeded return TRUE, if failed return FALSE
Description	<p>After the ADC device starts to operate in sleep mode, it determines the time from "ext_sense_out" pad high to the actual ADC starts to operate.</p> <p>For example:            delay2 = 8            delay from "ext_sense_out" pad high to actual ADC operation is <math>8 \times 244.2 \mu\text{s}</math>            = 1.95 ms</p>

<b>int DRV_ADC_SET_RTC_ENABLE_CHANNEL(HANDLE handler, UNSIGNED ch, UNSIGNED enable)</b>		
Parameter	handler	Device handle
	ch	Channel to enable. [0~3]
	enable	1: enable, 0: disable
Return	If succeeded return TRUE, if failed return FALSE	
Description	<p>Enable wakeup channel in sleep mode.</p> <p>If all channels are disabled, adc in sleep mode is changed to disable state.</p>	

<b>int DRV_ADC_SET_SLEEP_MODE(HANDLE handler, UNSIGNED x12_clk, UNSIGNED reg_ax12b_timer, UNSIGNED adc_step)</b>		
Parameter	handler	Device handle
	x12_clk	0 : 7.81 ms 1 : 31.25 ms 2 : 62.5 ms 3 : 250 ms 4 : 1000 ms 5 : 4000 ms 6 : 16000 ms 7 : 64000 ms
	reg_ax12b_timer	0~15
	adc_step	[0~7] Number of samples to calculate the average value $2^{\text{adc\_step} + 2}$ For example: Adc_step = 3 Averaging 32 samples, every measuring period.
Return	If succeeded return TRUE, if failed return FALSE	
Description	<p>ADC measuring interval = <math>(x12\_clk \times reg\_ax12b\_timer + 1)</math></p> <p>For example: <math>x12clk = 2</math>, <math>ax12b\_timer = 7</math>            Interval in sleep mode = <math>62.5\text{ms} \times 8 = 500\text{ms}</math></p>	

## 4.7.3 Interrupt Description

**ADC\_INTERRUPT\_FIFO\_HALF:** the interrupt that occurs when the FIFO Level is 4 or higher.

---

## SDK Programmer Guide

**ADC\_INTERRUPT\_FIFO\_FULL:** the interrupt that occurs when FIFO Level is 8.

**ADC\_INTERRUPT\_THD\_OVER:** this interrupt is issued when the value currently input 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 value currently input 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 value currently input 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 the DA16200 Example Application Manual [\[3\]](#).



## SDK Programmer Guide

### 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 17: 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 to DA16200 Example Application Guide [\[3\]](#).

When using GPIO & GPIO Retention API, the status of GPIO PIN is shown in [Table 18](#).

## SDK Programmer Guide

**Table 18: 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 below:

- "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.

## SDK Programmer Guide

### 4.8.2 Application Programming Interface

**Table 19: GPIO Interface API Elements**

<b>HANDLE GPIO_CREATE(UINT32 dev_type)</b>		
Parameter	dev_type	Device index
Return		If succeeded, return handle for the device. If failed return NULL
Description		The DA16200 can set GPIO_UNIT_A and GPIO_UNIT_C
<b>int GPIO_INIT (HANDLE handler)</b>		
Parameter	handler	Device handle
Return		If succeeded return ERR_NONE
Description		Configure the GPIO setting
<b>int GPIO_IOCTL(HANDLE handler, UINT32 cmd, VOID *data)</b>		
Parameter	handler	Device handle
	cmd	Commands are defined <gpio.h> in our SDK
	data	Data pointer
Return		If succeeded return ERR_NONE
Description		<p>The necessary configuration of GPIO can be set with this function. Commands are as below:</p> <ul style="list-style-type: none"> <li>GPIO_GET_DEVREG = 1,</li> <li>GPIO_SET_OUTPUT, // set gpio as an output</li> <li>GPIO_SET_INPUT, // set gpio as an input</li> <li>GPIO_GET_DIRECTION, // get gpio direction</li> <li>GPIO_SET_INTR_MODE, // set gpio interrupt mode [edge/level]</li> <li>GPIO_GET_INTR_MODE, // get gpio interrupt mode</li> <li>GPIO_SET_INTR_ENABLE, // enable gpio interrupt</li> <li>GPIO_SET_INTR_DISABLE, // disable gpio interrupt</li> <li>GPIO_GET_INTR_ENABLE, // get gpio interrupt enable status</li> <li>GPIO_GET_INTR_STATUS, // get gpio interrupt pending status</li> <li>GPIO_SET_INTR_CLEAR, // clear gpio interrupt status</li> <li>GPIO_SET_CALLACK, // set a callback function for gpio interrupt</li> </ul>
<b>int GPIO_READ (HANDLE handler, UINT32 addr, UINT16 *pdata, UINT32 dlen)</b>		
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

## SDK Programmer Guide

<b>int GPIO_WRITE (HANDLE handler, UINT32 addr, VOID *p_data, UINT32 p_dlen)</b>		
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
<b>int GPIO_CLOSE(HANDLE handler)</b>		
Parameter	handler	Device handle
Return		If succeeded return ERR_NONE
Description		GPIO close command
<b>INT32 GPIO_GET_ALT_FUNC (HANDLE handler, GPIO_ALT_FUNC_TYPE altFuncType, UINT32 * regVal)</b>		
Parameter	handler	Device handle
	altFuncType	GPIO alternate function type
	regVal	GPIO alternate function setting value
Return		If succeeded return 0
Description		Gets GPIO alternate function setting value
<b>INT32 GPIO_SET_ALT_FUNC(HANDLE handler, GPIO_ALT_FUNC_TYPE altFuncType, GPIO_ALT_GPIO_NUM_TYPE gpioType)</b>		
Parameter	handler	Device handle
	altFuncType	GPIO alternate function type
	gpioType	GPIO number
Return		If succeeded return 0
Description		Sets GPIO alternate function
<b>INT32 _GPIO_RETAIN_HIGH(UINT32 gpio_port, UINT32 gpio_num)</b>		
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
<b>INT32 _GPIO_RETAIN_LOW(UINT32 gpio_port, UINT32 gpio_num)</b>		
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
<b>void SAVE_PULLUP_PINS_INFO(UINT32 port_num, UINT32 pinnum)</b>		
Parameter	port_num	GPIO port number

## SDK Programmer Guide

int GPIO_WRITE (HANDLE handler, UINT32 addr, VOID *p_data, UINT32 p_dlen)		
	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 the DA16200 Example Application Guide [3].

## 4.9 UART

### 4.9.1 Introduction

The DA16200 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 20: 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	

## SDK Programmer Guide

Pin Name	Pin Number		I/O	Function Name
	QFN	fcCSP		
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
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 21: 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 has two UART ports <pre>typedef enum __uart_unit__ {     UART_UNIT_0 = 0,     UART_UNIT_1,     UART_UNIT_MAX } UART_UNIT_IDX;</pre> 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_BAUDRATE (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

## SDK Programmer Guide

int UART_IOCTL(HANDLE handler, UINT32 cmd, VOID *data)		
	cmd	Commands are defined in <UART.h> in the DA16200 SDK
	data	Data pointer
Return	If succeeded return ERR_NONE	
Description	<p>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:</p> <ul style="list-style-type: none"> <li>• UART_GET_DEVREG = 1, // get device physical address</li> <li>• UART_SET_CLOCK, // set base clock</li> <li>• UART_SET_BAUDRATE, // set baud rate</li> <li>• UART_GET_BAUDRATE, // get baud rate</li> <li>• UART_SET_LINECTRL, // set line control</li> <li>• UART_GET_LINECTRL, // get line control</li> <li>• UART_SET_CONTROL, // set UART control</li> <li>• UART_GET_CONTROL, // get UART control</li> <li>• UART_SET_QUE_SIZE, // set queue size</li> <li>• UART_SET_INT, // set interrupt configuration</li> <li>• UART_GET_INT, // get interrupt configuration</li> <li>• UART_SET_FIFO_INT_LEVEL, // set fifo level</li> <li>• UART_GET_FIFO_INT_LEVEL, // get fifo level</li> <li>• UART_SET_USE_DMA, // set DMA use</li> <li>• UART_GET_USE_DMA, // get DMA use</li> <li>• UART_CHECK_RXEMPTY, // check RX fifo empty</li> <li>• UART_CHECK_RXFULL, // check RX fifo full</li> <li>• UART_CHECK_TXEMPTY, // check TX fifo empty</li> <li>• UART_CHECK_TXFULL, // check TX fifo full</li> <li>• UART_CHECK_BUSY, // check UART busy</li> <li>• UART_SET_RX_SUSPEND, // set the RX function to suspend</li> <li>• UART_CLEAR_ERR_INT_CNT, // clear error interrupt counter</li> <li>• UART_GET_ERR_INT_CNT, // gets error interrupt counter</li> <li>• UART_SET_ERR_INT_CALLBACK, // set error interrupt callback function</li> <li>• UART_CLEAR_FRAME_INT_CNT, //clear frame error interrupt counter</li> <li>• UART_GET_FRAME_INT_CNT, // get frame error interrupt counter.</li> <li>• UART_SET_FRAME_INT_CALLBACK, // set frame error interrupt callback</li> <li>• UART_CLEAR_PARITY_INT_CNT, // clear parity error interrupt counter</li> <li>• UART_GET_PARITY_INT_CNT, // get frame error interrupt counter</li> <li>• UART_SET_PARITY_INT_CALLBACK, // set frame error interrupt callback</li> <li>• UART_CLEAR_BREAK_INT_CNT, // clear break error interrupt counter</li> <li>• UART_GET_BREAK_INT_CNT, // get break error interrupt counter</li> <li>• UART_SET_BREAK_INT_CALLBACK, // set break error interrupt callback</li> <li>• UART_CLEAR_OVERRUN_INT_CNT, // clear overrun error interrupt counter</li> <li>• UART_GET_OVERRUN_INT_CNT, // get overrun error interrupt counter</li> <li>• UART_SET_OVERRUN_INT_CALLBACK, // set overrun interrupt callback</li> </ul> <p>The user can find more information in 'uart.h' file</p>	

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

## SDK Programmer Guide

int UART_READ (HANDLE handler, VOID *p_data, UINT32 p_dlen)		
	p_data	Data pointer
	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 the DA16200 Example Application Manual [3].



## SDK Programmer Guide

### 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 22 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 22: 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 23: 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" <ul style="list-style-type: none"><li>● create the GPIO handler for chip selection</li></ul>
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 <ul style="list-style-type: none"><li>● create the MUTEX for support to control multi-slaves</li></ul>

## SDK Programmer Guide

int SPI_IOCTL(HANDLE handler, UINT32 cmd, VOID *data)		
Parameter	Handler	SPI Driver (HANDLE)
	Cmd	IOCTL command
	data	IOCTL parameters
Return		TRUE / FALSE (int)
Description		<p>SPI_SET_SPEED</p> <ul style="list-style-type: none"> <li>set the target SPI clock</li> </ul> <p>SPI_GET_SPEED</p> <ul style="list-style-type: none"> <li>get the current value of SPI clock</li> </ul> <p>SPI_SET_FORMAT</p> <ul style="list-style-type: none"> <li>set the SPI interface mode <ul style="list-style-type: none"> <li>SPI_TYPE_MOTOROLA_O0H0</li> <li>SPI_TYPE_MOTOROLA_O1H1</li> </ul> </li> </ul> <p>SPI_SET_DMAMODE</p> <ul style="list-style-type: none"> <li>set the DMA transfer mode to the DMA mode</li> </ul> <p>SPI_GET_MAX_LENGTH</p> <ul style="list-style-type: none"> <li>the maximum burst size</li> </ul> <p>SPI_SET_MAX_LENGTH</p> <ul style="list-style-type: none"> <li>set the maximum burst size (up to 63 kB)</li> </ul> <p>SPI_SET_CALLACK</p> <ul style="list-style-type: none"> <li>set the user defined callbacks. <ul style="list-style-type: none"> <li>SPI_INTIDX_RORINT: the receive overrun interrupt</li> <li>SPI_INTIDX_RTMININT: the receive timeout interrupt</li> <li>SPI_INTIDX_RXINT: when there are four or more data in the RX FIFO</li> <li>SPI_INTIDX_TXINT: when there are four or less data in the TX FIFO</li> </ul> </li> </ul> <p>SPI_SET_CONCAT</p> <ul style="list-style-type: none"> <li>set the SPI burst mode to the concatenation mode</li> </ul> <p>SPI_SET_BUSCONTROL</p> <ul style="list-style-type: none"> <li>set the SPU bus access mode</li> </ul> <p>SPI_GET_BUSCONTROL</p> <ul style="list-style-type: none"> <li>get the current value of SPI bus access mode</li> </ul> <p>SPI_GET_DELAYSEL</p> <ul style="list-style-type: none"> <li>get the parameters of current delay model</li> </ul> <p>SPI_SET_LOCK</p> <ul style="list-style-type: none"> <li>lock/unlock the mutex of SPI driver</li> </ul>
int SPI_READ(HANDLE handler, void *pdata, UINT32 dlen)		
Parameter	Handler	SPI Driver (HANDLE)
Return		zero - false, non-zero - data length (int)
Description		<p>SPI read operation</p> <ul style="list-style-type: none"> <li>pdata: RX data buffer</li> <li>dlen: byte length</li> </ul>

## SDK Programmer Guide

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 <ul style="list-style-type: none"><li>● pdata: TX data buffer</li><li>● dlen: byte length</li></ul>	
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 <ul style="list-style-type: none"><li>● snddata: TX data buffer</li><li>● sndlen: byte length</li><li>● rcvdata: TX data buffer</li><li>● rcvlen: byte length</li></ul>	
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 <ul style="list-style-type: none"><li>● snddata: TX data buffer</li><li>● sndlen: byte length</li><li>● rcvdata: TX data buffer</li><li>● rcvlen: byte length</li></ul>	
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 the DA16200 Example Application Manual [\[3\]](#).

## 4.11 Pulse Counter

### 4.11.1 Introduction

The pulse counter is a device that counts external electrical pulses. It can be used in both normal operation mode and sleep mode. Its current consumption is very low when used in sleep mode.

The pulse counter has a function to wake up in Sleep Mode, when the number of pulses is the same as the set number.

## SDK Programmer Guide

### 4.11.2 Application Programming Interface

<b>HANDLE DRV_PULSE_COUNTER_CREATE( UINT32 dev_id )</b>	
dev_id	Instance Number of Pulse Counter
Return	If succeeded return handle for such device, if failed return NULL
Description	Function create handle with parameter "dev_id" designated

<b>int DRV_PULSE_COUNTER_INIT (HANDLE handler)</b>	
Handler	Device handle to initialize
Return	If succeeded return TRUE, if failed return FALSE
Description	Initialize Pulse Counter Driver

<b>unsigned int DRV_PULSE_COUNTER_READ (HANDLE handler)</b>	
Handler	Device handle
Return	Current Pulse Counter value.
Description	Read current Pulse Counter value.

<b>int DRV_PULSE_COUNTER_INTERRUPT_CLEAR (HANDLE handler)</b>	
Handler	Device handle
Return	If succeeded return TRUE, if failed return FALSE
Description	Clear Pulse Counter interrupt.

<b>int DRV_PULSE_COUNTER_INTERRUPT_ENABLE (HANDLE handler, unsigned int en, unsigned int thd)</b>	
Handler	Device handle to initialize
en	1:enable interrupt, 0:disable interrupt
thd	Interrupt threshold value to generate interrupt
Return	If succeeded return TRUE, if failed return FALSE
Description	Enable/Disable interrupt with threshold value

<b>int DRV_PULSE_COUNTER_ENABLE(HANDLE handler, unsigned int en)</b>	
Handler	Device handle
en	1:enable interrupt, 0:disable interrupt
Return	If succeeded return TRUE, if failed return FALSE
Description	Enable/Disable Pulse Counter Device

## SDK Programmer Guide

<b>int DRV_PULSE_COUNTER_RESET(HANDLE handler)</b>	
Handler	Device handle to
Return	If succeeded return TRUE, if failed return FALSE
Description	Reset Counter

<b>int DRV_PULSE_COUNTER_EDGE(HANDLE handler, unsigned int isfalling)</b>	
Handler	Device handle
isfalling	1 : falling edge, 0 : rising edge
Return	If succeeded return TRUE, if failed return FALSE
Description	Select capturing edge of Pulse Counter

<b>int DRV_PULSE_COUNTER_SET_GPIO(HANDLE handler, unsigned int num, unsigned int mode)</b>				
Handler	Device handle to			
num	Number of GPIO for counting pulse. Refer Description			
mode	0 or 1			
Return	If succeeded return TRUE, if failed return FALSE			
Description	num	Mode = 0	mode = 1	
	0	GPIOA4	GPIOC0	
	1	GPIOA5	GPIOC1	
	2	GPIOA6	GPIOC2	
	3	GPIOA7	GPIOC3	
	4	GPIOA8	GPIOC4	
	5	GPIOA9	GPIOC6	
	6	GPIOA10	GPIOC7	
	7	GPIOA11	GPIOC8	
	8	GPIOC5	GPIOA12	
	9	GPIOA13	GPIOA14	

<b>int DRV_PULSE_COUNTER_CALLBACK_REGISTER(HANDLE handler, UINT32 vector, UINT32 callback, UINT32 param)</b>	
Handler	Device handle to initialize
vector	N/A
callback	Callback function, called when interrupt generated.
param	Parameter, using when function called.
Return	If succeeded return TRUE, if failed return FALSE
Description	Initialize Pulse Counter Driver

## 5 NVRAM

The DA16200 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 24: NVRAM API Elements**

<b>int write_nvram_int(const char *name, int val)</b>		
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
<b>int write_nvram_string(const char *name, const char *val)</b>		
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
<b>int read_nvram_int(const char *name, int *_val)</b>		
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
<b>char *read_nvram_string(const char *name)</b>		
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

## SDK Programmer Guide

## 6 HW Accelerators

### 6.1 Set SRAM to Zero

#### 6.1.1 Application Programming Interface

void fc9k_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];
fc9k_memset32(buffer, 0, 1024);
```

### 6.2 CRC Calculation

#### 6.2.1 Application Programming Interface

UINT32 fc9k_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 = fc9k_hwcrc32(sizeof(UINT32), (void *)data, sizeof(data), (~0));
```

## SDK Programmer Guide

### 6.3 Pseudo Random Number Generator (PRNG)

#### 6.3.1 Application Programming Interface

UINT32 fc9k_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 = fc9k_random();
```

### 6.4 Memory Copy Using DMA

#### 6.4.1 Application Programming Interface

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 (4B Aligned)
	src	A pointer to the buffer that you want to copy data from (4B 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);
```



## SDK Programmer Guide

## 7 Wi-Fi Interface Configuration

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

### 7.1 Application Programming Interface

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

- Four simple functions to get or set each parameter
- Verify the result with the error code

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 25: Wi-Fi Configuration API**

<b>int da16x_set_config_int(int name, int value)</b>		
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: <code>ret = da16x_set_config_int (DA16X_CONF_INT_CHANNEL, 11)</code> <ul style="list-style-type: none"> <li>• Set the operating channel of the AP interface to 11</li> </ul>
<b>int da16x_set_config_str (int name, char *value)</b>		
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: <code>ret = da16x_set_config_str (DA16X_CONF_STR_IP_0, "10.0.0.1")</code> <ul style="list-style-type: none"> <li>• Set the IP address of the STA interface to 10.0.0.1</li> </ul>
<b>int da16x_get_config_int (int name, int *value)</b>		
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: <code>ret = da16x_get_config_int (FC9k_CONF_INT_CHANNEL, &amp;channel)</code> <ul style="list-style-type: none"> <li>• Get the operating channel of the AP interface</li> </ul>
<b>int da16x_get_config_str (int name, char *value)</b>		
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 For example: <code>ret = da16x_get_config_str (DA16X_CONF_STR_IP_0, ip_addr)</code> <ul style="list-style-type: none"> <li>• Get the IP address of the STA interface</li> </ul>

## SDK Programmer Guide

<b>int da16x_set_config_int(int name, int value)</b>		
<b>int da16x_set_nvcache_str(int name, char *value)</b>		
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 <code>da16x_nvcache2flash()</code> For example: <code>ret = da16x_set_nvcache_str(DA16X_CONF_STR_IP_0, ip_addr)</code> <ul style="list-style-type: none"> <li>Set IP address of the STA interface</li> </ul>
<b>int da16x_set_nvcache_int(int name, int value)</b>		
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 <code>da16x_nvcache2flash()</code> For example: <code>ret = da16x_set_nvcache_int(DA16X_CONF_INT_CHANNEL, 11)</code> <ul style="list-style-type: none"> <li>Set the operating channel of the AP interface to 11</li> </ul>
<b>void da16x_nvcache2flash(void)</b>		
Parameter	Void	void
Return		void
Description		Commit parameters (set by <code>da16x_set_nvcache_int/str</code> ) in NVRAM cache to flash

## SDK Programmer Guide

### 7.1.1 Integer Type Parameters

**Table 26: NVRAM Integer Type**

Name	Description
DA16X_CONF_INT_MODE	Wi-Fi operation mode <ul style="list-style-type: none"> <li>0: STA</li> <li>1: Soft-AP</li> </ul>
DA16X_CONF_INT_AUTH_MODE_0	Wi-Fi authentication mode for STA interface <ul style="list-style-type: none"> <li>CC_VAL_AUTH_OPEN</li> <li>CC_VAL_AUTH_WEP</li> <li>CC_VAL_AUTH_WPA</li> <li>CC_VAL_AUTH_WPA2</li> <li>CC_VAL_AUTH_WPA_AUTO (WPA &amp; WPA2)</li> </ul>
DA16X_CONF_INT_AUTH_MODE_1	Wi-Fi authentication mode for Soft-AP interface <ul style="list-style-type: none"> <li>CC_VAL_AUTH_OPEN</li> <li>CC_VAL_AUTH_WPA</li> <li>CC_VAL_AUTH_WPA2</li> <li>CC_VAL_AUTH_WPA_AUTO (WPA &amp; WPA2)</li> </ul> (WEP is unsupported on the DA16200 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 <ul style="list-style-type: none"> <li>CC_VAL_ENC_TKIP</li> <li>CC_VAL_ENC_CCMP</li> <li>CC_VAL_ENC_AUTO (TKIP &amp; CCMP)</li> </ul>
DA16X_CONF_INT_ENCRYPTION_1	Wi-Fi data encryption mode for Soft-AP interface <ul style="list-style-type: none"> <li>CC_VAL_ENC_TKIP</li> <li>CC_VAL_ENC_CCMP</li> <li>CC_VAL_ENC_AUTO (TKIP &amp; CCMP)</li> </ul>
DA16X_CONF_INT_WIFI_MODE_0	Wi-Fi mode based on IEEE 802.11 standard for STA interface <ul style="list-style-type: none"> <li>CC_VAL_WFMODE_BGN</li> <li>CC_VAL_WFMODE_GN</li> <li>CC_VAL_WFMODE_BG</li> <li>CC_VAL_WFMODE_N</li> <li>CC_VAL_WFMODE_G</li> <li>CC_VAL_WFMODE_B</li> </ul>
DA16X_CONF_INT_WIFI_MODE_1	Wi-Fi mode based on IEEE 802.11 standard for Soft-AP interface <ul style="list-style-type: none"> <li>CC_VAL_WFMODE_BGN</li> <li>CC_VAL_WFMODE_GN</li> <li>CC_VAL_WFMODE_BG</li> <li>CC_VAL_WFMODE_N</li> <li>CC_VAL_WFMODE_G</li> <li>CC_VAL_WFMODE_B</li> </ul>

## SDK Programmer Guide

Name	Description
DA16X_CONF_INT_CHANNEL	Soft-AP operation channel setting by channel number <ul style="list-style-type: none"> <li>1~11: for US</li> <li>0: Auto</li> </ul>
DA16X_CONF_INT_FREQUENCY	Soft-AP operation channel setting by frequency value (MHz)
DA16X_CONF_INT_ROAM	Operating roaming function for STA interface <ul style="list-style-type: none"> <li>0: Stop</li> <li>1: Run</li> </ul>
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)
DA16X_CONF_INT_WMM	WMM On/Off setting <ul style="list-style-type: none"> <li>0: Off</li> <li>1: On</li> </ul>
DA16X_CONF_INT_WMM_PS	WMM-PS On/Off setting <ul style="list-style-type: none"> <li>0: Off</li> <li>1: On</li> </ul>
DA16X_CONF_INT_DHCP_CLIENT	DHCP client On/Off for STA interface <ul style="list-style-type: none"> <li>0: Off</li> <li>1: On</li> </ul>
DA16X_CONF_INT_DHCP_SERVER	DHCP server On/Off for Soft-AP interface <ul style="list-style-type: none"> <li>0: Off</li> <li>1: On</li> </ul>
DA16X_CONF_INT_DHCP_LEASE_TIME	DHCP server lease time (sec.)

## 7.1.2 String Type Parameters

Table 27: 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 device name (for WPS or Wi-Fi Direct)

## SDK Programmer Guide

Name	Description
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 `dal6x_set_nvcache_int/str()` and `dal6x_nvcache2flash()`. Use of `dal6x_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 `dal6x_set_nvcache_int/str` followed by `dal6x_nvcache2flash()`, which will give much better performance to your application.

The following example explains how to set STA mode.

**Table 28: 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)
dal6x_set_nvcache_int(DA16X_CONF_INT_MODE, 0);
dal6x_set_nvcache_str(DA16X_CONF_STR_SSID_0, ssid);
dal6x_set_nvcache_int(DA16X_CONF_INT_AUTH_MODE_0, auth_type);
if (auth_type == CC_VAL_AUTH_WEP) {
    dal6x_set_nvcache_str(DA16X_CONF_STR_WEP_KEY0, wep_key[0]);
    dal6x_set_nvcache_str(DA16X_CONF_STR_WEP_KEY1, wep_key[1]);
    dal6x_set_nvcache_str(DA16X_CONF_STR_WEP_KEY2, wep_key[2]);
    dal6x_set_nvcache_str(DA16X_CONF_STR_WEP_KEY3, wep_key[3]);
    dal6x_set_nvcache_str(DA16X_CONF_INT_WEP_KEY_INDEX, wep_key_index);
} else if (auth_type > CC_VAL_AUTH_WEP) {
    dal6x_set_nvcache_str(DA16X_CONF_STR_PSK_0, psk);
    dal6x_set_nvcache_int(DA16X_CONF_INT_ENCRYPTION_0, encryption);
}
dal6x_set_nvcache_int(DA16X_CONF_INT_WIFI_MODE_0, wifi_mode);
dal6x_set_nvcache_int(DA16X_CONF_INT_DPM, dpm);

/* IP & DHCP Client Setting */
dal6x_set_nvcache_int(DA16X_CONF_INT_DHCP_CLIENT, dhcp_client);
if (!dhcp_client) {
    dal6x_set_nvcache_str(DA16X_CONF_STR_IP_0, ip);
    dal6x_set_nvcache_str(DA16X_CONF_STR_NETMASK_0, subnet);
    dal6x_set_nvcache_str(DA16X_CONF_STR_GATEWAY_0, gateway);
    dal6x_set_nvcache_str(DA16X_CONF_STR_DNS_0, dns);
}
dal6x_nvcache2flash(); // commit name/value params in Cache to flash memory

```

## SDK Programmer Guide

```
reboot_func(SYS_REBOOT, DISCONNECT_SEND);
```

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

**Table 29: 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)
dal16x_set_nvcache_int(DA16X_CONF_INT_MODE, 1);
dal16x_set_nvcache_str(DA16X_CONF_STR_SSID_1, ssid);
dal16x_set_nvcache_int(DA16X_CONF_INT_AUTH_MODE_1, auth_type);
if (auth_type > CC_VAL_AUTH_WEP) {
    dal16x_set_nvcache_str (DA16X_CONF_STR_PSK_1, psk);
    dal16x_set_nvcache_int(DA16X_CONF_INT_ENCRYPTION_1, encryption);
}
dal16x_set_nvcache_int(DA16X_CONF_INT_CHANNEL, channel);
dal16x_set_nvcache_int(DA16X_CONF_STR_COUNTRY, country_code);
dal16x_set_nvcache_int(DA16X_CONF_INT_WIFI_MODE_1, wifi_mode);
dal16x_set_nvcache_int(DA16X_CONF_INT_WMM, wmm);
dal16x_set_nvcache_int(DA16X_CONF_INT_WMM_PS, wmm_ps);

/* IP Setting */
dal16x_set_nvcache_str(DA16X_CONF_STR_IP_1, ip);
dal16x_set_nvcache_str(DA16X_CONF_STR_NETMASK_1, subnet);
dal16x_set_nvcache_str(DA16X_CONF_STR_GATEWAY_1, gateway);

/* DHCP Server Setting */
if (dhcp_server) {
    dal16x_set_nvcache_str(DA16X_CONF_STR_DHCP_START_IP, start_ip);
    dal16x_set_nvcache_str(DA16X_CONF_STR_DHCP_END_IP, end_ip);
    dal16x_set_nvcache_str(DA16X_CONF_STR_DHCP_DNS, dhcp_dns);
    dal16x_set_nvcache_str(DA16X_CONF_INT_DHCP_LEASE_TIME, dhcp_lease_time);
}
dal16x_set_nvcache_int(FC9K_CONF_INT_DHCP_SERVER, dhcp_server);

dal16x_nvcache2flash(); // commit name/value params in Cache to flash memory

reboot_func(SYS_REBOOT, DISCONNECT_SEND);
```

## 7.2 Soft-AP Configuration by Factory Reset

Many IoT devices start as a Soft-AP device to operate AP provisioning. The DA16200 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 the DA16200 EVK User Manual [2] and is connected to GPIO 7 on the DA16200 EVB.

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

## SDK Programmer Guide

### 7.2.1 Configuration Data Structure Integer Type Parameters

The DA16200 SDK has the structure shown in [Table 30](#) to configure Soft-AP interface.

**Table 30: Soft-AP Interface Code**

```
[\\src\\common\\inc\\dal6x_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 the DA16200 EVK User Manual [\[3\]](#) or Appendix [B.1](#)
- int customer\_ip\_address: IP address type
  - IPADDR\_DEFAULT (0): IP class is 10.0.0.1

## SDK Programmer Guide

- 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 DA16200 SDK has the function shown in [Table 31](#) to configure the Soft-AP interface. You can write your own values. This function is invoked when a factory reset is done.

**Table 31: Soft-AP Configuration Code**

```
[\\customer\\main\\src\\system_start.c]
void set_customer_softap_config(void){
#ifdef __SUPPORT_FACTORY_RST_APMODE__
    /* Set to user customer'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
     */
}
```



## SDK Programmer Guide

```

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

#### 7.3.1 Provisioning Specification

The system-provided provisioning reference thread is run as a TLS or TCP server with port number 9900. A user-defined provisioning thread can also be used instead of the stock thread. See Section 7.3.2.

To use a system-provided provisioning reference thread, User needs to write a peer (mobile) application that sends provisioning data based on the data structure `prov_config_t`.

**Table 32: Provisioning Protocol Code**

```

[\\customer\\apps\\inc\\user_provision.h]

#define MAX_SSID_LEN      128
#define MAX_PASSKEY_LEN  128
#define MAX_WEP_KEY_LEN  16

#define DEFAULT_AUTH_TYPE 4    // WPA-PSK AUTO Mode

/* Local provisioning structure */
typedef struct _prov_config {
    /* Auto reboot flag - 0: No reboot, 1: Auto reboot */
    int auto_restart_flag;

    char ssid[MAX_SSID_LEN + 1];
    char psk[MAX_PASSKEY_LEN + 1];

    /* 0: OPEN, 1:WEP, 2:WPA-PSK, 3:WPA2-PSK, 4:WPA-AUTO */
    int auth_type;

    /* For WEP-Key */
    int wep_key_index;
    char wep_key[4][MAX_WEP_KEY_LEN + 1];

    /*
    * 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
    */

```

## SDK Programmer Guide

```

* 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
*/
char country[4];

int ip_addr_mode;    // 0:DHCP Client, 1:STATIC

int sntp_flag;
char sntp_server[32];
int sntp_period;

int dpm_mode;        // 0:Disable, 1:Enable
int dpm_ka;          // DPM keep-alive periodic time
int dpm_user_wu;     // DPM user-wakeup time
int dpm_tim_wu;      // DPM TIM wakeup period time
} prov_config_t;

```

- **int auto\_restart\_flag:** Auto-reboot flag after provisioning is completed
  - 0: No reboot (default)
  - 1: Auto reboot
- **char ssid [MAX\_SSID\_LEN + 1]:** SSID of AP

### NOTE

The reason for 128 bytes in `user_softap_provision.c` instead of 32 bytes is (specified in 802.11 spec) for Unicode handling. Some commercial APs allow, for example, Unicode characters (i.e. Korean, Chinese, etc.) to be used as SSID. For an SSID in Unicode, UTF-8 encoding is used. As one Unicode character can occupy a maximum of 4 bytes in UTF-8 encoding, 128 bytes (32 Unicode characters \* 4) is allocated.

- **char psk [MAX\_PASSKEY\_LEN + 1]:** Pre-shared key of AP
- **int auth\_type:** Authentication type
  - 0: Open
  - 1: WEP
  - 2: WPA-PSK
  - 3: WPA2-PSK
  - 4: WPA-AUTO
- **int wep\_key\_index:** When WEP is used, WEP key index
- **char wep\_key [4] [MAX\_WEP\_KEY\_LEN + 1]:** When WEP is used, WEP key
- **char country [4]:** Country code  
See section on **Country Code** in the DA16200 EVK User Manual [\[2\]](#)
- **int ip\_addr\_mode:** IP address mode
  - 0: DHCP client (default)
  - 1: Static IP address
- **int sntp\_flag:** Flag to use SNTP server
  - 0: Disable
  - 1: Enable
- **char sntp\_server [32]:** SNT server IP address
- **int sntp\_period:** SNTP update period
- **int dpm\_mode:** Flag to use DPM mode
  - 0: Disable
  - 1: Enable
  - If DPM mode is enabled, fill in 0 (zero) for the items below for default DPM action

## SDK Programmer Guide

- int dpm\_ka: DPM keep-alive period
- int dpm\_user\_wu: DPM user wakeup period
- int dpm\_tim\_wu: DPM TIM wakeup period

### 7.3.2 User Soft-AP Provisioning Application

When the DA16200 boots in Soft-AP mode, a thread runs to handle “Provisioning” data by default. This thread is system-provided, a simple TCP-based kind of Connection Manager thread that is talking to a TLS or TCP peer to get and set AP profile info, server info, etc.

To test the functionality, create a peer TLS or TCP client application that communicates with default provisioning thread. For the Customer's product, user can change the provisioning data structure as their own provisioning structure with TLS/TCP protocol.

The SDK has a feature called “User Soft-AP Provisioning”, and based on the sample implementation, users can develop their provisioning functions.

To enable the user-defined provisioning function, enable `__SUPPORT_USER_PROVISION__` in file `config_xxx_sdk.h`.

In file `user_softap_provision.c`, you can register your own function with `regist_user_softap_prov_fn`.

See the DA16200 SoftAP User Provisioning Manual [3].

#### 7.3.2.1 Example User Soft-AP Provisioning Thread

**Table 33: Soft-AP Provisioning Thread Sample Code (TCP Sample)**

```
[\\customer\\apps\\inc\\user_provision.h]
/*
This sample source is showing TCP based sample implementation of SoftAP
provisioning, hence, you can customize on your need.
*/
// you are able to extend this structure as you like
typedef struct _prov_config {
    /* Auto reboot flag - 0: No reboot, 1: Auto reboot */
    int auto_restart_flag;

    char ssid[MAX_SSID_LEN + 1];
    char psk[MAX_PASSKEY_LEN + 1];
    ...
    int dpm_user_wu;        // DPM user-wakeup time
    int dpm_tim_wu;        // DPM TIM wakeup period time
} prov_config_t;

[\\customer\\apps\\src\\user_softap_provision.c]
// sample helper function for saving prov data to NVRAM.
static void user_save_prov_config(UCHAR *data) {
... // once data is received, those prov info are saved in NVRAM and make DUT reboot
}

// TCP based sample provisioning thread implementation (TCP server)
void user_softap_2_sta_prov(ULONG arg)
{
... // tcp server waiting for a client to get prov info
}

/*
```

## SDK Programmer Guide

This function is called at boot up when the system boots in soft-AP mode. A user-defined provisioning function is provided here.

For example, if you give NULL to this function pointer, Dialog Semiconductor's default implementation of provisioning thread (using the default prov\_config\_t) will run.

If you want to use your own implementation (you can customize "user\_softap\_2\_sta\_prov" provided in this source file as well).

```
*/
void regist_user_softap_prov_fn(void)
{
...
/* Regist customer provisioning function */
user_softap_prov_fn = user_softap_2_sta_prov;
...
}
```

### 7.3.2.2 Example Peer (Mobile) Application

The mobile application operates as a TCP client and connects to the gateway IP address with the port number of the TCP server. The mobile application should send TCP data with the data structure, prov\_config\_t as shown in [Table 34](#).

[Table 34](#) is an example of source code for a peer application that runs provisioning with the DA16200.

**Table 34: Soft-AP Provisioning Thread Sample Code for Peer Application (TCP Sample)**

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <time.h>
#include <unistd.h>

#define PROVISION_PORT_NO    9900
#define TX_BUF_SIZE          1024

#define MAX_SSID_LEN          128
#define MAX_PASSKEY_LEN       128
#define MAX_WEP_KEY_LEN       16

#define DEFAULT_AUTH_TYPE     4        // WPA-PSK Auto Mode

/* Local provisioning structure */
typedef struct _prov_config {
    /* 0: No reboot, 1: Auto Reboot */
    int    auto_restart_flag;

    char    ssid[MAX_SSID_LEN+1];
    char    psk[MAX_PASSKEY_LEN+1];

    /* 0: OPEN, 1:WEP, 2:WPA-PSK, 3:WPA2-PSK, 4:WPA-AUTO */
    int    auth_type;

    /* For WEP-Key */
    int    wep_key_index;
    char    wep_key[4][MAX_WEP_KEY_LEN+1];

    /*
```

## SDK Programmer Guide

```

* 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
*/
char    country[4];

int      ip_addr_mode;           // 0:DHCP Client, 1:STATIC

int      sntp_flag;
char     sntp_server[32];
int      sntp_period;

int      dpm_mode;
int      dpm_ka;
int      dpm_user_wu;
int      dpm_tim_wu;
} prov_config_t;

#define TEST_SSID            "N_A1004_WPAx-PSK"
#define TEST_PSK             "N12345678"
#define TEST_AUTH_TYPE      4           // WPA-PSK

#define TEST_COUNTRY         "KR"
#define TEST_IP_ADDR_TYPE   0           // DHCP Client

void make_prov_config_data(prov_config_t *config)
{
    memset(config, 0, sizeof(prov_config_t));

    /* Config Wi-Fi information: Simple */
    sprintf(config->ssid, "%s", TEST_SSID);
    config->auth_type = DEFAULT_AUTH_TYPE;
    if (config->auth_type > 0)
        sprintf(config->psk, "%s", TEST_PSK);

    /*
     * Additional information
     * - Default country code is "US"
     * - Default ip address mode is "DHCP Client"
     */
    sprintf(config->country, "%s", TEST_COUNTRY);
    config->ip_addr_mode = TEST_IP_ADDR_TYPE;           // DHCP Client

    config->auto_restart_flag = 1;
}

void print_config(prov_config_t *config)
{
    printf("\n--- Test configuration:\n");
    printf("\t- ssid\t\t: %s\n", config->ssid);
    printf("\t- psk\t\t: %s\n", config->psk);
    printf("\t- auth_type\t: %d (0:OPEN, 1:WPA-PSK, 2:WEP, 3:WPA2-PSK, 4:WPA-AUTO)\n",
config->auth_type);

```

## SDK Programmer Guide

```

    if (config->auth_type == 2) {
        printf("\t- wep_key_index\t: %d\n", config->wep_key_index);
        printf("\t\twep_key_0\t\t: %s\n", config->wep_key[0]);
        printf("\t\twep_key_1\t\t: %s\n", config->wep_key[1]);
        printf("\t\twep_key_2\t\t: %s\n", config->wep_key[2]);
        printf("\t\twep_key_3\t\t: %s\n", config->wep_key[3]);
    }

    printf("\n");
    printf("\t- Country Code\t: %s\n", config->country);

    printf("\t- SNTP mode\t: %d\n", config->sntp_flag);
    if (config->sntp_flag == 1) {
        printf("\t\tSNTP Server\t\t: %s\n", config->sntp_server);
        printf("\t\tSNTP period\t\t: %d\n", config->sntp_period);
        printf("\n");
    }

    printf("\t- DPM mode\t: %d\n", config->dpm_mode);
    if (config->dpm_mode == 1) {
        printf("\t\tDPM Keep-Alive Time\t: %d\n", config->dpm_ka);
        printf("\t\tDPM User Wakeup Time\t: %d\n", config->dpm_user_wu);
        printf("\t\tDPM TIM Wakeup Time\t: %d\n", config->dpm_tim_wu);
    }
}

int main(int argc, char *argv[])
{
    int    sock;
    struct sockaddr_in server;
    char    peer_ip[16], peer_port[8];
    char    *input_str, *semi_col;
    prov_config_t    *tx_msg;
    int    i = 0;
    int    length, result;
    int    port_no;

    if (argc < 2) {
        printf("Usage: tcpc_provision peer_ip:port\n");
        return 0;
    }
    memset(peer_ip, 0, 16);
    memset(peer_port, 0, 8);

    input_str = argv[1];
    length = strlen(input_str);
    while (length-- > 0 && input_str[i] != ':') {
        peer_ip[i] = input_str[i];
        i++;
    }

    /* Get port number */
    if (length > 0) {
        semi_col = strstr(argv[1], ":");
        strcpy(peer_port, semi_col+1);
        port_no = atoi(peer_port);
    } else {
        port_no = PROVISION_PORT_NO;
    }

    /* Create socket */
    sock = socket(AF_INET, SOCK_STREAM, 0);
    if (sock == -1) {
        printf("Could not create socket");
    }

```

## SDK Programmer Guide

```
        goto test_exit;
    }
    printf("Socket created ...");

    server.sin_addr.s_addr = inet_addr(peer_ip);
    server.sin_family = AF_INET;
    server.sin_port = htons(port_no);

    // Connect to remote server
    if (connect(sock, (struct sockaddr *)&server, sizeof(server)) < 0) {
        printf("connect failed. Error");

        goto test_exit;
    }
    printf("Connected ...\n");

    tx_msg = (prov_config_t *)malloc(sizeof(prov_config_t));

    make_prov_config_data(tx_msg);

    /* Send some data */
    if (result = send(sock, tx_msg, sizeof(prov_config_t), 0) < 0) {
        printf("!!! Send fail (result=0x%x)\n", result);
        goto test_exit;
    }

    print_config(tx_msg);

    /* For safe tcp transmission */
    sleep(1);

test_exit:
    close(sock);
    free(tx_msg);

    return 0;
}
```

## SDK Programmer Guide

## 8 Tx Power Table Edit

The DA16200 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 18: 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. Here is 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 19.

```
[/DA16200] # setup
Stop all services for the setting.
Are you sure ? [Yes/No] : y
```

Figure 19: 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 20.

```
[ 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 20: Tune Tx Power: Choose Country Code

3. Reboot and connect to an AP.
4. Examine the current TX power indices. See Figure 21.

```
[PRADA] txpwr_1 print
[Current TX Level]
C_CODE: ALL
txpwr_1 2 2 2 2 2 2 2 2 2 2 2 2 2 f
[PRADA]
```

Figure 21: Tune Tx Power: Check Tx Power Indices



## SDK Programmer Guide

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

```
[PRADA] txpwr_1 3 3 3 3 3 3 3 3 3 3 3 3 3 f
set 3333333333333f
[PRADA] reboot
```

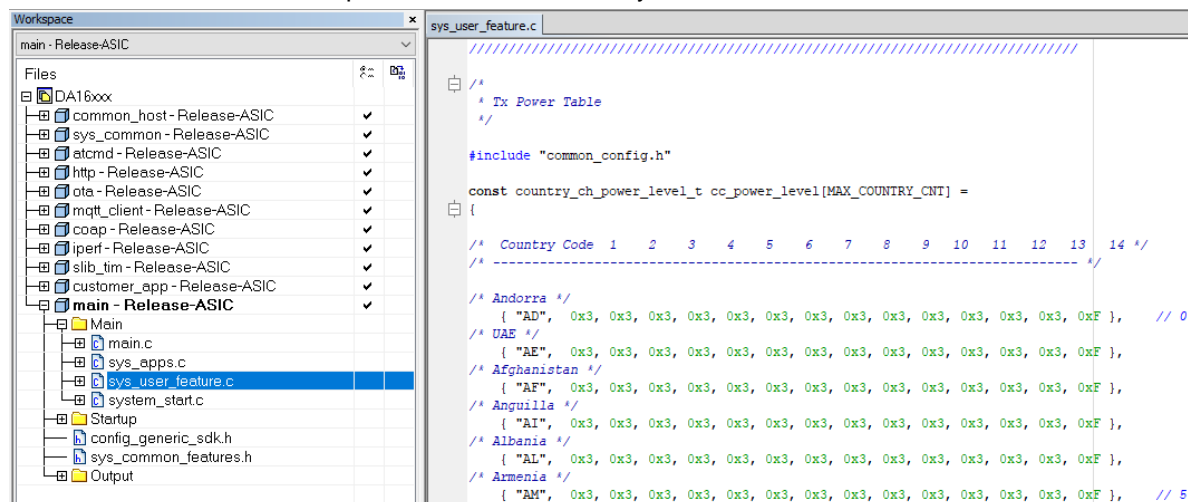
**Figure 22: Tune Tx Power: Modify Tx Power Indices**

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.

## 8.2 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 DA162000 SDK, open \src\common\main\sys\_user\_feature.c.



**Figure 23: 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.

## SDK Programmer Guide

### 9 Tips

#### 9.1 Find/Optimize Stack Size for Your Application

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

Figure 24 is a snapshot of command `ps` when `tcp_client_sample.c` is run.

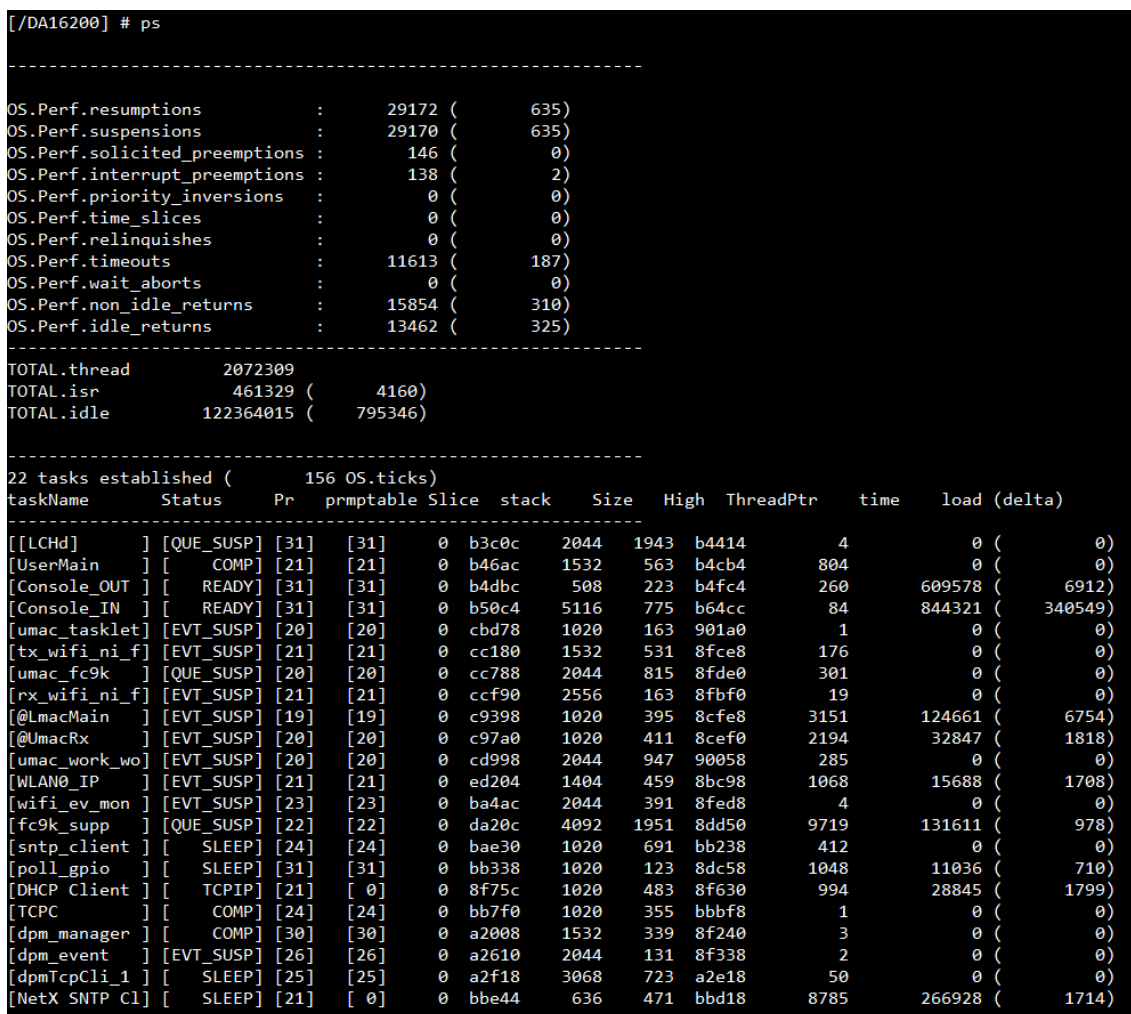


Figure 24: Check Stack Size

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

Table 35: TCP Client Sample Code

```

...
#ifdef __TCP_CLIENT_SAMPLE__
{ SAMPLE_TCP_CLI, tcp_client_sample, 1024, OAL_PRI_APP(0), TRUE, FALSE,
  TCP_CLI_TEST_PORT, RUN_ALL_MODE },
#endif /* __TCP_CLIENT_SAMPLE__ */
...

```

Command `ps` shows the following information:

## SDK Programmer Guide

- Stack: the stack address
- Size: the stack size allocated
- High: peak usage size of the stack
- ThreadPtr: the current stack pointer

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

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.)

## 9.2 Debug Stack Overflow

Often, the consequences of a stack overflow are manifested far removed from the cause of the overflow itself. As a result, to identify and solve the cause is much more difficult. When stack overflow happens, sometimes the system hangs or, sometimes, luckily, a crash dump is printed in the console window as shown in [Table 36](#) that gives some hint, and there is possibly a very good clue for the cause of stack overflow. But note that there is not always an Oops dump like in [Table 36](#), because that depends on which part of memory got corrupted by stack overflow.

**Table 36: Corrupted Stack Overflow**

```
[[OOPS Dump: c0f0]]
--RTC Time: 00000000.01d7db15
Registers
    R0:0000000d, R1:0023dd81, R2:00306811, R3:00306811
    R4:00305510, R5:00310594, R6:00306804, R7:00310947
    R8:00000000, R9:00000000, R10:00000000, R11:00000000
    R12:00000000, LR:002db52d, PC:00000000, PSR:20000000
    SP:00310468, eLR:fffffffd,
Stack
[0x00310468]: 9FCB4F45 0031095F 0031095F 0031064C 00310654 0022892B 00000000
00000000
[0x00310488]: 00000000 00000000 00000000 74736F48 7574203A 646E756D 61702E6F
6F646172
[0x003104a8]: 616C6178 632E7362 00006D6F 00000000 00000000 00000000 00000000
00000000
[0x003104c8]: 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000

ThreadX
Thread: @userapp1
run.cnt: 580
stack.ptr: 310830
stack.start: 3104f0
stack.end: 3108eb
stack.usage: bb
state: 0
dly_suspend: 0
suspending: 0

[0x00310830]: 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000
```

## SDK Programmer Guide

```
[0x00310850]: 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000
[0x00310870]: 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000
[0x00310890]: 00000000 000001D6 00000000 00310940 003104E0 00224D71 FFFFFFFF
0031091C

// in above example, you can suspect stack overflow at the thread "userapp1", because "stack.ptr" points to
almost the end of the stack.
```

If "stack overflow" is suspected in your application, there is a useful tip to utilize. Let us take an example of `tcp_client_sample.c` again. In the `ps` snapshot (Figure 24) you see the peak size is about 623 bytes. At that time, the `ps` command ran right after this app finished communication with a peer (DUT connected to a TCP server and received a simple data from the TCP server - using IONINJA). As a result, in this simple use case, the stack size for the application is the example should be at least 600 bytes.

Table 37 shows an example to force stack overflow. In `sample_apps.c`, the value is set to 500 bytes as an example to force stack overflow.

**Table 37: Force Stack Overflow**

```
#ifdef TCP_CLIENT_SAMPLE
{ SAMPLE_TCP_CLI, tcp_client_sample, 500, OAL_PRI_APP(0), TRUE, FALSE,
  TCP_CLI_TEST_PORT, RUN_ALL_MODE },
#endif /* TCP_CLIENT_SAMPLE */
```

Table 38 gives an example of stack overflow debug code to add in `tcp_client_sample.c`.

**Table 38: Stack Overflow Debug Code**

```
[\\sample\\Network\\TCP_Client\\src\\tcp_client_sample.c]
(stack overflow debug codes)

...
/* External functions */
extern int check_net_init(int iface);
extern long iptolong(char *ip);
extern VOID _nx_tcp_packet_send_ack(NX_TCP_SOCKET *, ULONG);
extern USHORT get_random_value_ushort(void);
extern void UART_lowlevel_Printf(const char *fmt,...);
...
void print_stack_checker(OAL_THREAD_TYPE *thread)
{
    if(thread!= NULL) {
// normal printf cannot be used here as it is under an exceptional condition, hence, you have to use
this function to print something. In below code, thread pointer is handed over
        UART_lowlevel_Printf("Stk.Over: %s, %x, %x\\n"
            , thread->tx_thread_name
            , thread->tx_thread_stack_highest_ptr
            , thread->tx_thread_stack_start);
    }
    while(1);
}

...
void tcp_client_sample(ULONG arg)
{
    char *server_ip = TX_NULL;
```

## SDK Programmer Guide

```
int      server_port = 0;

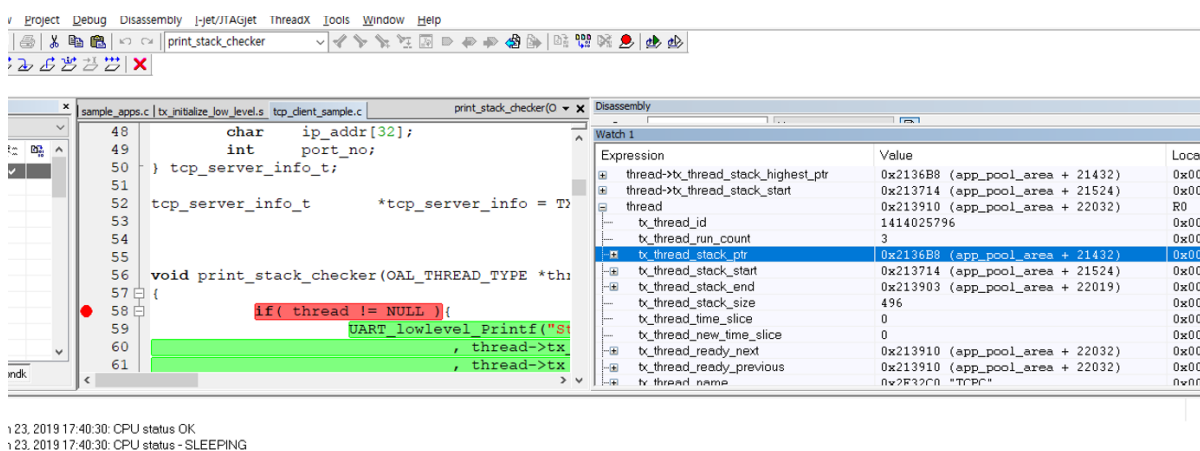
tx_thread_stack_error_notify(&print_stack_checker); //add at application init

tcp_server_info = (tcp_server_info_t *)malloc(sizeof(tcp_server_info_t));

server_ip = read_nvramp_string("TCP_SERVER_IP");
if (server_ip == TX_NULL)
```

In IAR (with Ijet – jtag debugger - connected), set a breakpoint at UART\_lowlevel\_printf(), and then run. See [Appendix D](#) 'How to I-Jet debugger'.

Figure 25 shows that the breakpoint is reached while running, which means a stack is overflown / corrupted. Examine the IAR debug window for more info.



**Figure 25: IAR Debug Window for Stack Overflow**

## 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.

---

## SDK Programmer Guide

## Appendix B Country Code and Tx Power

This section lists the country codes that the DA16200 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 39: 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
"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

## SDK Programmer Guide

"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
"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



## SDK Programmer Guide

"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
"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

## SDK Programmer Guide

"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
"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

## SDK Programmer Guide

"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
"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"	World Wide	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

## SDK Programmer Guide

### 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 SDK, user can change the supporting “country code list” for their product. See [Table 40](#).

- `~/src/common/main/sys_user_features.c`

**Table 40: 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 Doxygen Documents

The DA16200 SDK supports online documents that comply with the Doxygen document format.

To start a Doxygen document page, open index.html in your web browser.

<b>NOTE</b>
Compatible with all types of web browsers.

~/DA16200\_SDK/doc/html/index.html

## SDK Programmer Guide

# DA16200 SDK 1.0.0

Ultra-low power Wi-Fi SoC

Main Page	Related Pages	Modules	Data Structures ▾	Files ▾	Search
-----------	---------------	---------	-------------------	---------	--------

### DA16200 SDK Documentation

```
*****
* Copyright (C) 2016-2019, Dialog Semiconductor.
* This computer program includes Confidential, Proprietary Information
* of Dialog Semiconductor. All Rights Reserved.
*****
```

DA16200 is a highly integrated ultra-low power Wi-Fi system on a chip (SoC) and allows users to develop the Wi-Fi solution on a single chip. The user implements their application with the DA16200 SDK and the compile environment is the IAR Embedded Workbench IDE of IAR Systems. DA16200 supports various applications and utilities to users as below.

#### Applications for Users

- MQ Telemetry Transport (MQTT)
- Constrained Application Protocol (CoAP)
- AT-Commands
- Over The Air (OTA) Update
- Zero Configuration Networking (Zeroconf)
- HTTP Server/Client

#### Utilities

- Command Line Interface (CLI)
- Configuration with Non-Volatile Random-Access Memory (NVRAM)
- Dynamic Power Management (DPM)
- JavaScript Object Notation (JSON)

Generated by  1.8.16

# DA16200 SDK 1.0.0

Ultra-low power Wi-Fi SoC

Main Page	Related Pages	Modules	Data Structures ▾	Files ▾	Search
-----------	---------------	---------	-------------------	---------	--------

### DA16200 SDK Documentation

```
*****
* Copyright (C) 2016-2019, Dialog Semiconductor.
* This computer program includes Confidential, Proprietary Information
* of Dialog Semiconductor. All Rights Reserved.
*****
```

DA16200 is a highly integrated ultra-low power Wi-Fi system on a chip (SoC) and allows users to develop the Wi-Fi solution on a single chip. The user implements their application with the DA16200 SDK and the compile environment is the IAR Embedded Workbench IDE of IAR Systems. DA16200 supports various applications and utilities to users as below.

#### Applications for Users

- MQ Telemetry Transport (MQTT)
- Constrained Application Protocol (CoAP)
- AT-Commands
- Over The Air (OTA) Update
- Zero Configuration Networking (Zeroconf)
- HTTP Server/Client

#### Utilities

- Command Line Interface (CLI)
- Configuration with Non-Volatile Random-Access Memory (NVRAM)
- Dynamic Power Management (DPM)
- JavaScript Object Notation (JSON)

Generated by  1.8.16

## SDK Programmer Guide

### DA16200 SDK 1.0.0

Ultra-low power Wi-Fi SoC

Main Page	Related Pages	Modules	Data Structures ▾	Files ▾	Q Search
-----------	---------------	---------	-------------------	---------	----------

#### DA16200 SDK Documentation

```
*****
* Copyright (C) 2016-2019, Dialog Semiconductor.
* This computer program includes Confidential, Proprietary Information
* of Dialog Semiconductor. All Rights Reserved.
*****
```

DA16200 is a highly integrated ultra-low power Wi-Fi system on a chip (SoC) and allows users to develop the Wi-Fi solution on a single chip. The user implements their application with the DA16200 SDK and the compile environment is the IAR Embedded Workbench IDE of IAR Systems. DA16200 supports various applications and utilities to users as below.

#### Applications for Users

- MQ Telemetry Transport (MQTT)
- Constrained Application Protocol (CoAP)
- AT-Commands
- Over The Air (OTA) Update
- Zero Configuration Networking (Zeroconf)
- HTTP Server/Client

#### Utilities

- Command Line Interface (CLI)
- Configuration with Non-Volatile Random-Access Memory (NVRAM)
- Dynamic Power Management (DPM)
- JavaScript Object Notation (JSON)

Generated by  1.8.16

Figure 26: Doxygen Document of the DA16200 SDK

## Appendix D How to use I-Jet debugger

### D.1 Notice to Use Debugger on IAR Workbench

When the DA16200 boots directly from SFLASH memory, additional header information in the image is required. On the other hand, when the Customer/Debugger uses the I-JET or J-Link debugger on the IAR workbench environment, an image that is not included in the header information should be written to the SFLASH because of the IAR specification.

So, if the customer/debugger wants to boot from the SFLASH memory directly after using the I-Jet or J-Link debugger on IAR workbench environment, the customer/debugger has to download three images again, which are supported with the SDK.

#### • SFLASH 2 MB

[MROM] loady 0	// 2nd Bootloader
	// DA16200_BOOT-GEN01-01-XXXXX-000000_W25Q32JW.img
[MROM] loady a000	// RTOS Image
	// DA16200_RTOS-GEN01-01-YYYYY-000000.img
[MROM] loady f1000	// SLIB Image
	// DA16200_SLIB-GEN01-01-ZZZZZ-000000.img

#### • SFLASH 4 MB

[MROM] loady 0	// 2nd Bootloader
	// DA16200_BOOT-GEN01-01-XXXXX-000000_W25Q32JW.img
[MROM] loady a000	// RTOS Image
	// DA16200_RTOS-GEN01-01-YYYYY-000000.img
[MROM] loady 18a000	// SLIB Image
	// DA16200_SLIB-GEN01-01-ZZZZZ-000000.img

### D.2 I-Jet Debug Setting

The IAR I-Jet Debugger proceeds by downloading and debugging an image at a temporary SFlash address. So, after the normal image is generated, the SFlash downloader or command `loady` is used to load the image to the formal address.

1. Connect the I-Jet Debugger to the DA16200 EVB. See [Figure 27](#).



Figure 27: Connect I-Jet Debugger to the DA16200 EVB

2. Debugger set-up:



## SDK Programmer Guide

- In the IAR Embedded Workbench IDE window, right-click on **main – Release-ASIC** (1) and select **Set as Active** (2) and **Options** (3) See Figure 28.

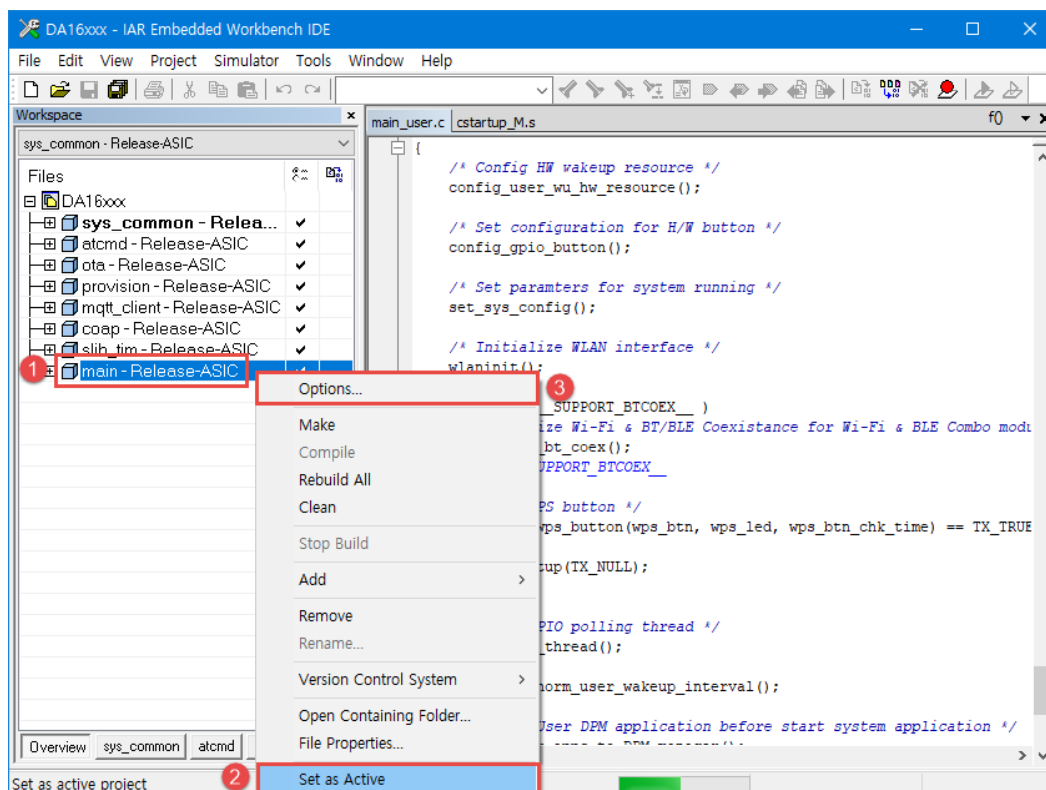
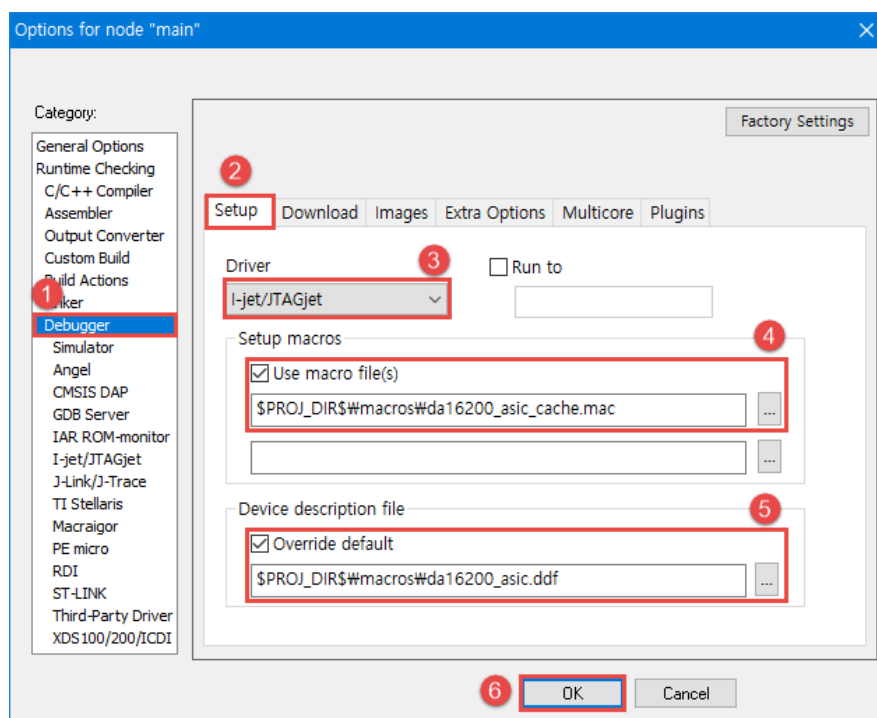


Figure 28: Select Debugger Option

- In the **Options for node "main"** window, in the **Category** list, select **Debugger** (1). See Figure 29.
- Select the **Setup** tab (2).
- In the **Driver** drop-down list, select **I-jet/JTAGjet** (3).
- In the **Setup macros** area (4), select the checkbox **Use macro file(s)** and set the filename to `$PROJ_DIR$\macros\da16200_asic_cache.mac`.

## SDK Programmer Guide

- f. In the **Device description file** area (5), select the checkbox **Override default** and set the filename to `$PROJ_DIR$\macros\da16200_asic.ddf`.

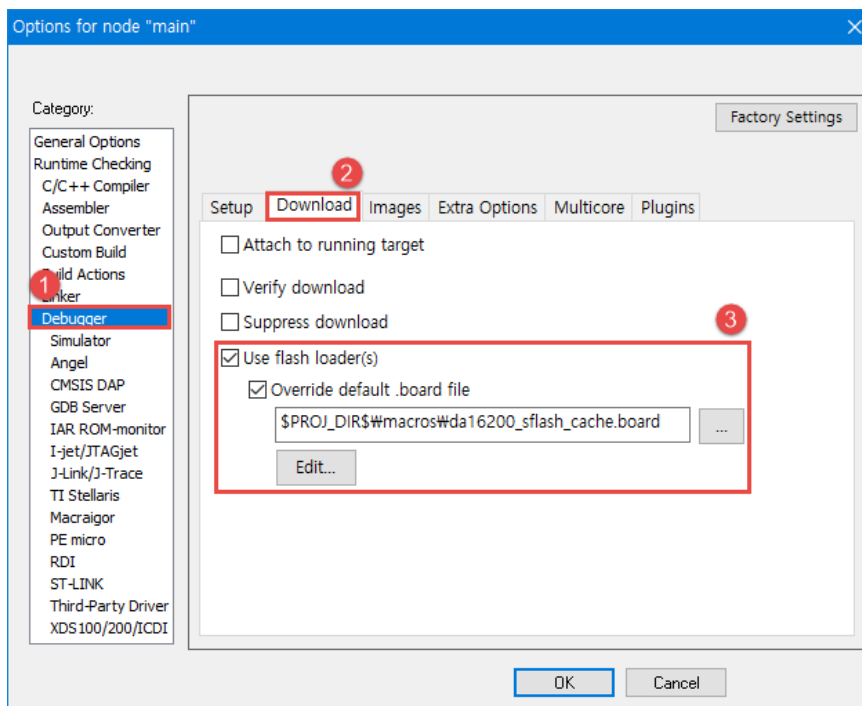


**Figure 29: Debugger Setup Setting**

- g. Select the **Download** tab (2). See [Figure 30](#).  
 h. Select the checkbox **Use flash loader(s)** (3).  
 i. Select the checkbox **Override default .board file** (3).

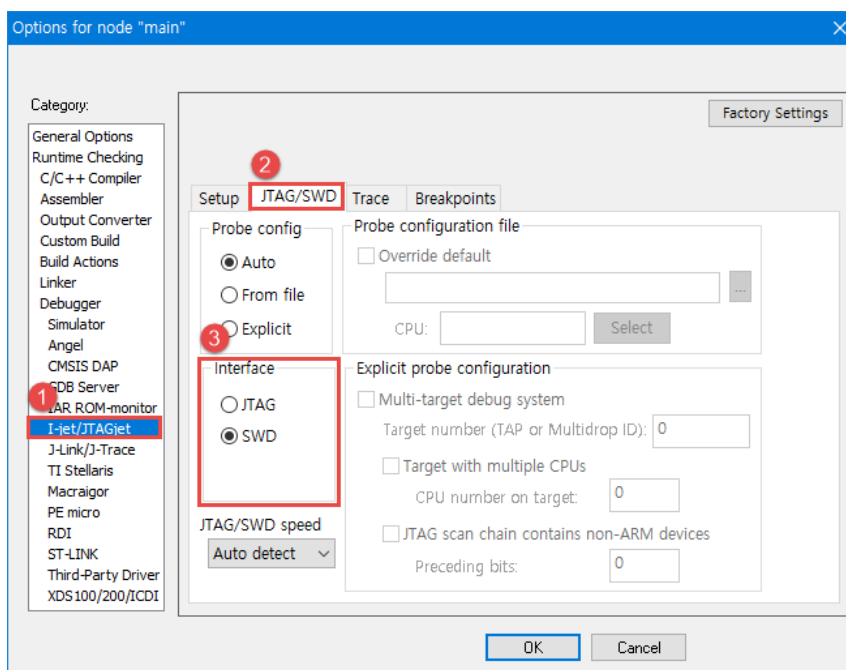
## SDK Programmer Guide

- j. Set the filename to `$PROJ_DIR$\macros\da16200_sflash_cache.board` (3).



**Figure 30: Debugger Download Setting**

- k. In the Category list, select **JTAG/SWD** (1). See Figure 31.

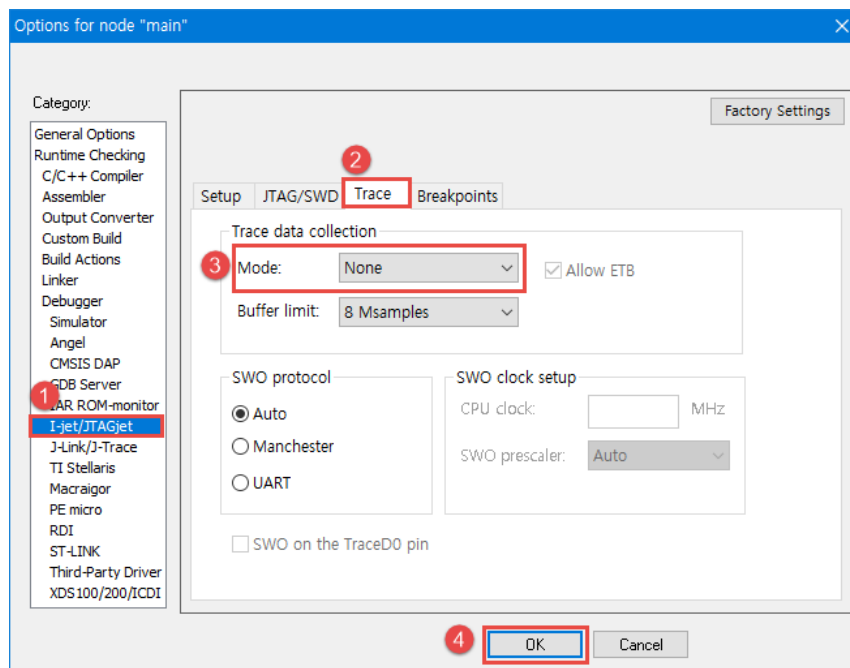


**Figure 31: I-Jet JTAG/SWD Setting**

- l. Select the **JTAG/SWD** tab (2).  
 m. In the **Interface** area, select the **SWD** radio button (3).  
 n. Select the **Trace** tab (2). See Figure 32.

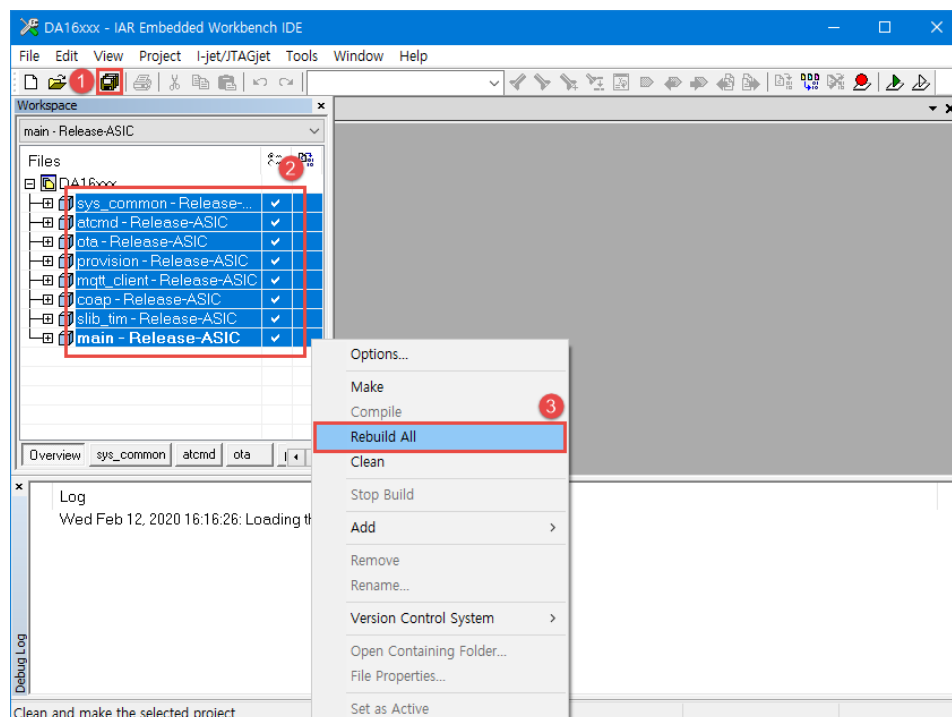
## SDK Programmer Guide

- o. Set the **Mode:** drop-down list to **None** (3).
- p. Click **OK** (4).



**Figure 32: Trace Mode Setting**

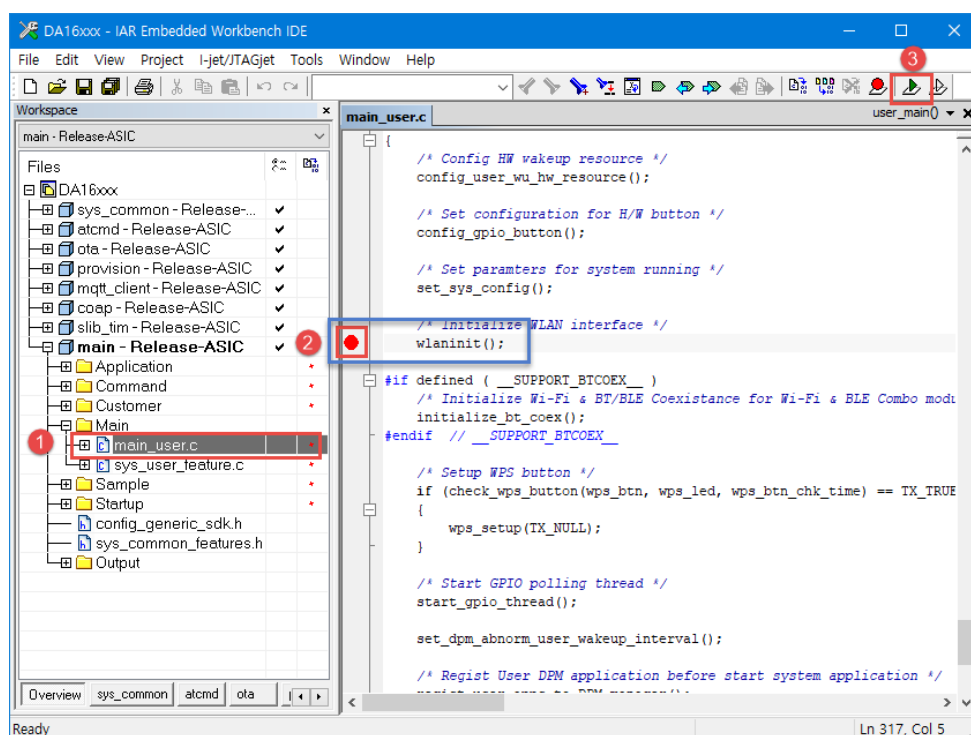
3. Rebuild the SDK. See [Figure 33](#).



**Figure 33: Rebuild SDK**

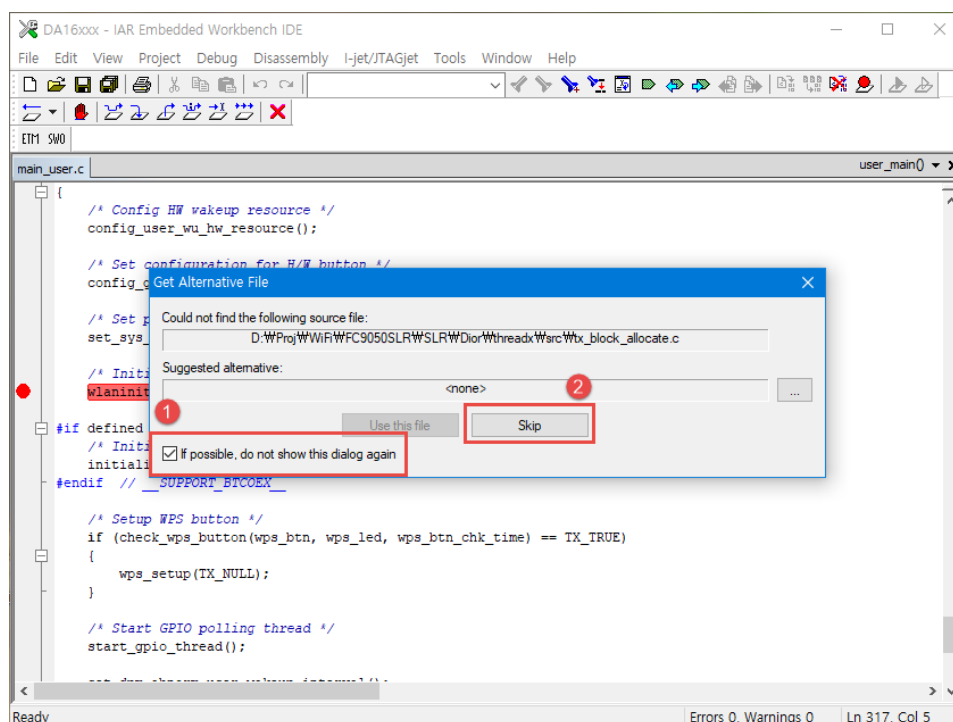
4. Setup a breakpoint (1~2), and then click **Download and Debug** (3). See [Figure 34](#).

## SDK Programmer Guide



**Figure 34: Download and Debug**

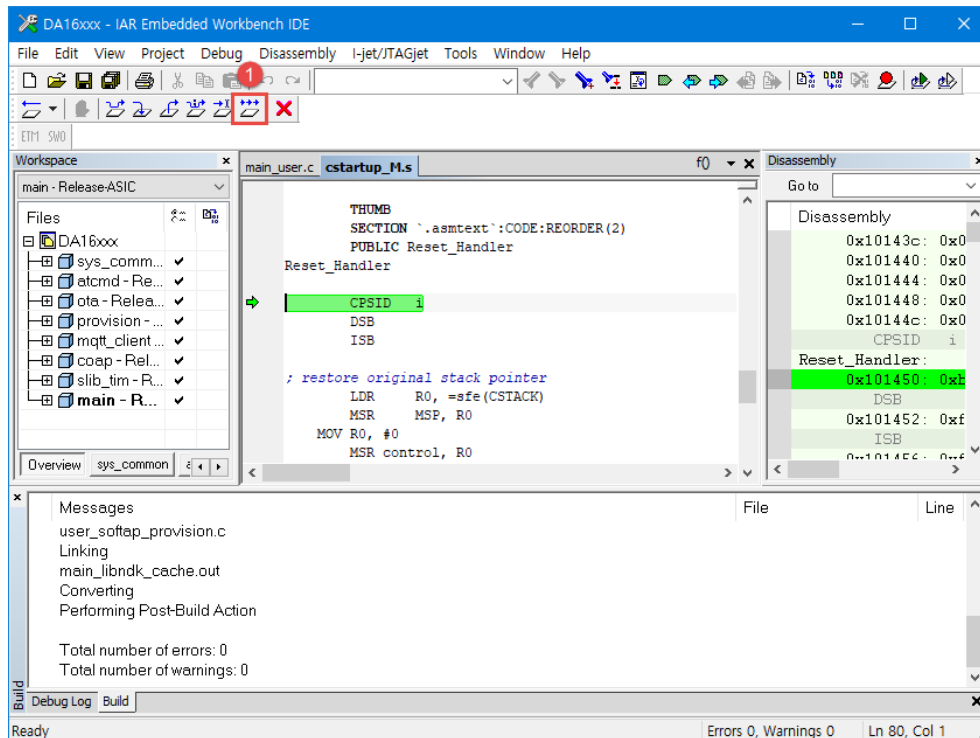
- Click the checkbox and Skip button in pop-up window. See Figure 35.



**Figure 35: Pop-up Message**

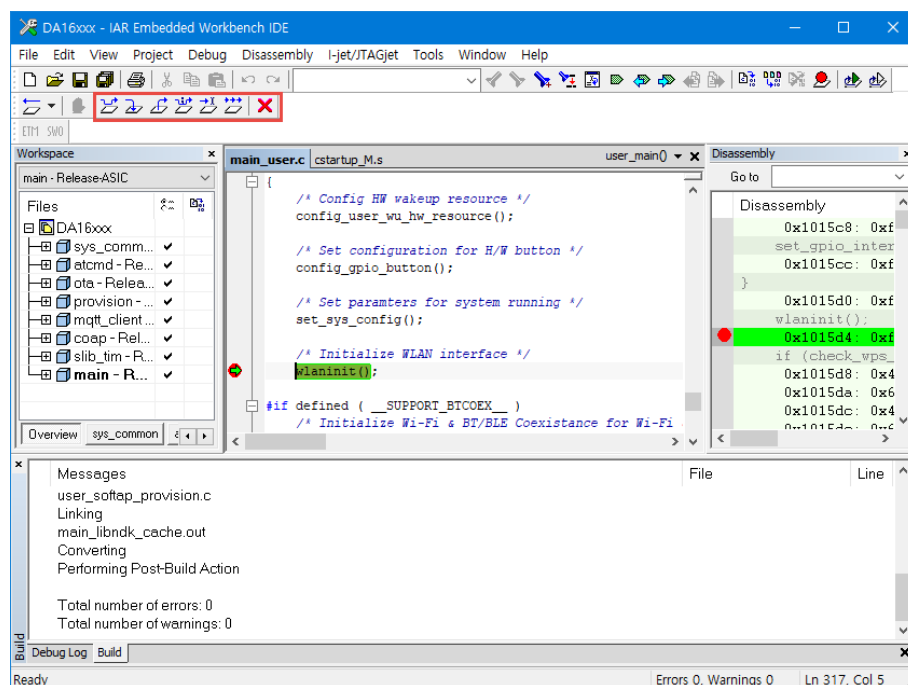
## SDK Programmer Guide

6. Click the **Go** button (1). See [Figure 36](#).



**Figure 36: Download and Debug Windows**

7. Pause Window on Break Pointer. See [Figure 37](#).



**Figure 37: Break Point Window**

## SDK Programmer Guide

### D.3 J-Link Debug Setting

The SEGGER J-Link Debugger proceeds by downloading and debugging an image at a temporary SFlash address. So, after the normal image is generated, the SFlash downloader or command `loady` is used to load the image to the formal address.

1. Download J-Link Software ([https://www.segger.com/downloads/jlink/JLink\\_Windows.exe](https://www.segger.com/downloads/jlink/JLink_Windows.exe)).
2. Install JLink\_Windows.
3. Connect the J-Link Debugger to the DA16200 EVB. See Figure 38.



Figure 38: Connect J-Link Debugger to the DA16200 EVB

4. Debugger set-up:
  - a. In the IAR Embedded Workbench IDE window, right-click on **main – Release-ASIC** and select **Options**. See Figure 39.

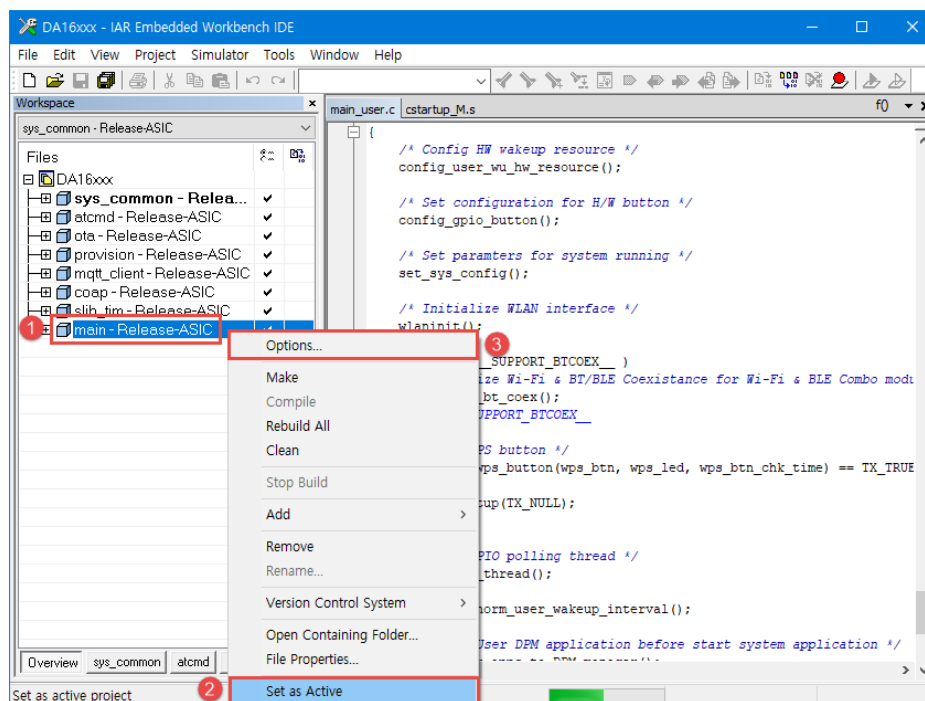
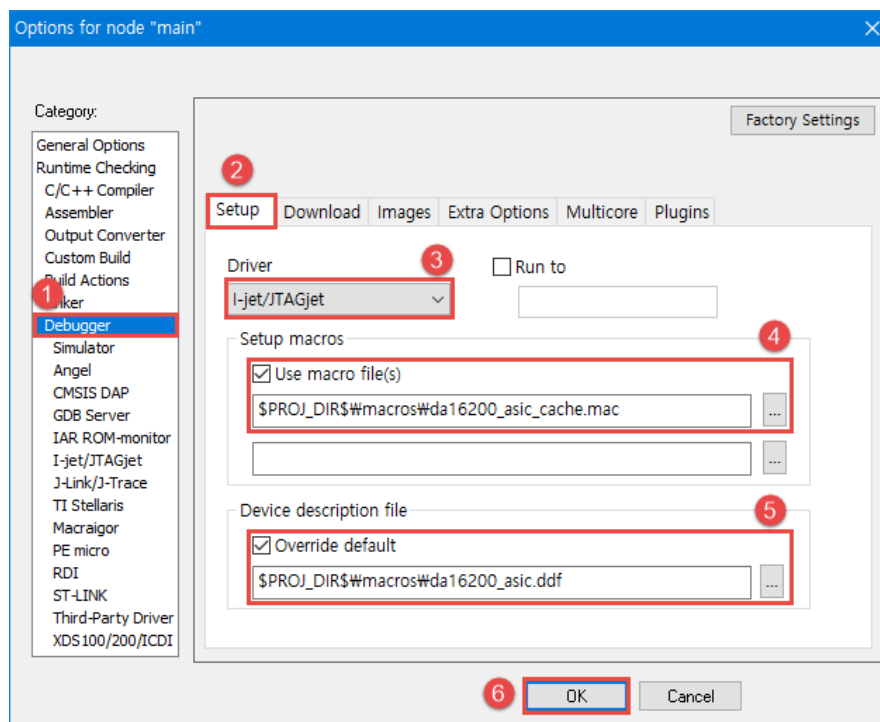


Figure 39: Select Debugger Option

- b. In the Options for node "main" window, in the Category list, select Debugger (1). See Figure 40.

## SDK Programmer Guide

- c. Select the Setup tab (2).
- d. In the Driver drop-down list, select J-Link/J-Trace (3).
- e. In the Setup macros area (4), select the checkbox Use macro file(s) and set the filename to \$PROJ\_DIR\$\macros\da16200\_asic\_cache.mac..
- f. In the Device description file area (5), select the checkbox Override default and set the filename to \$PROJ\_DIR\$\macros\da16200\_asic.ddf.



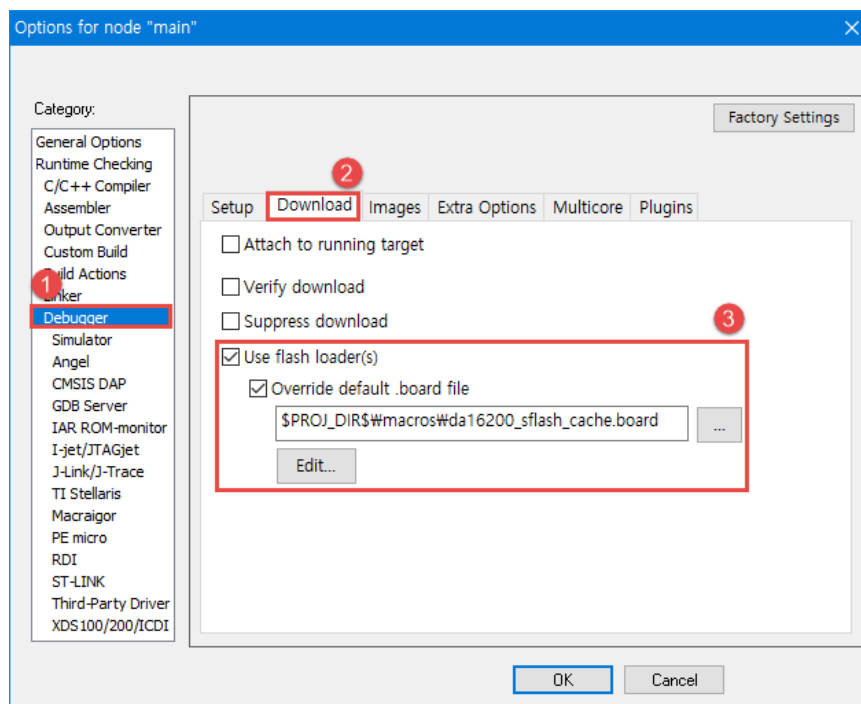
**Figure 40: Debugger Setup Setting**

- g. Select the Download tab (2). See [Figure 41](#).
- h. Select the checkbox Use flash loader(s) (3).
- i. Select the checkbox Override default .board file (3).



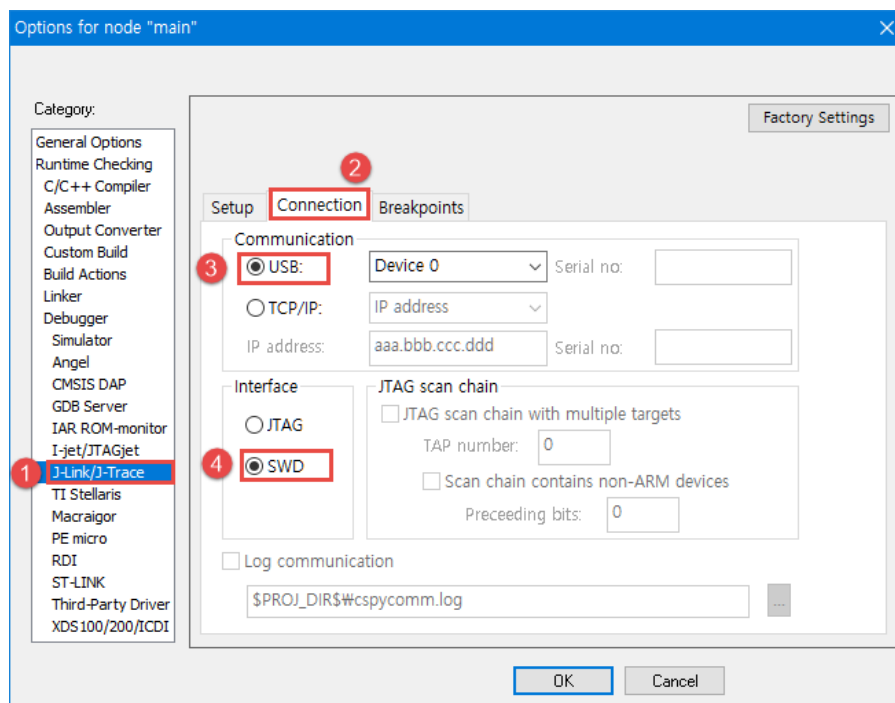
## SDK Programmer Guide

- j. Set the filename to \$PROJ\_DIR\$macros\da16200\_sflash\_cache.board (3).



**Figure 41: Debugger Download Setting**

- k. In the Category list, select J-Link/J-Trace (1). See Figure 42.



**Figure 42: J-Link/J-Trace JTAG/SWD Setting**

- l. Select the Connection tab (2).  
m. In the Interface area, select the SWD radio button (4).

## SDK Programmer Guide

5. Rebuild the SDK. See [Figure 43](#).

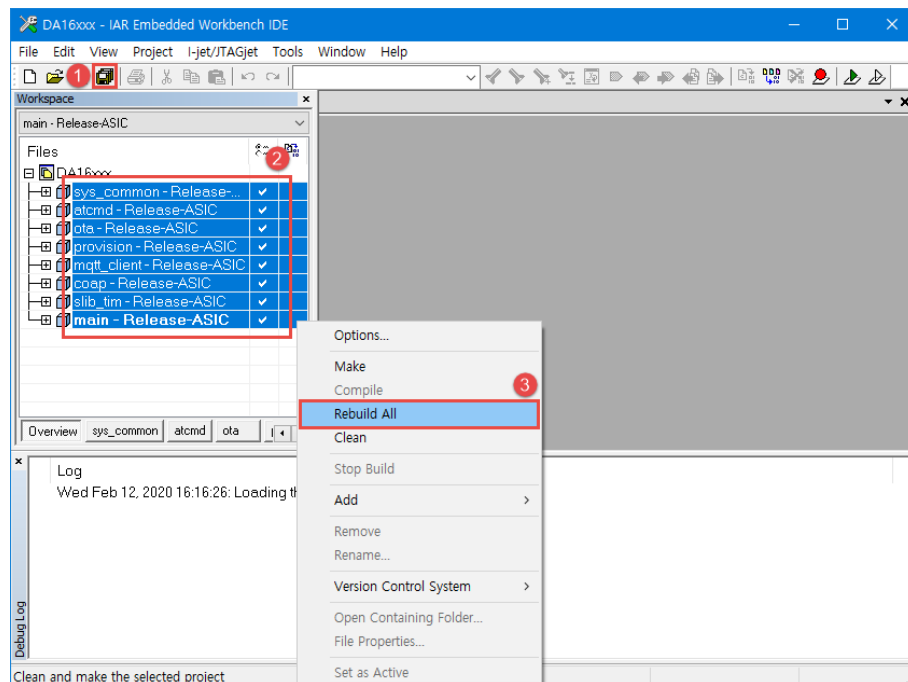


Figure 43: Rebuild SDK

6. Setup a breakpoint (1), and then click **Download and Debug** (2). See [Figure 44](#).

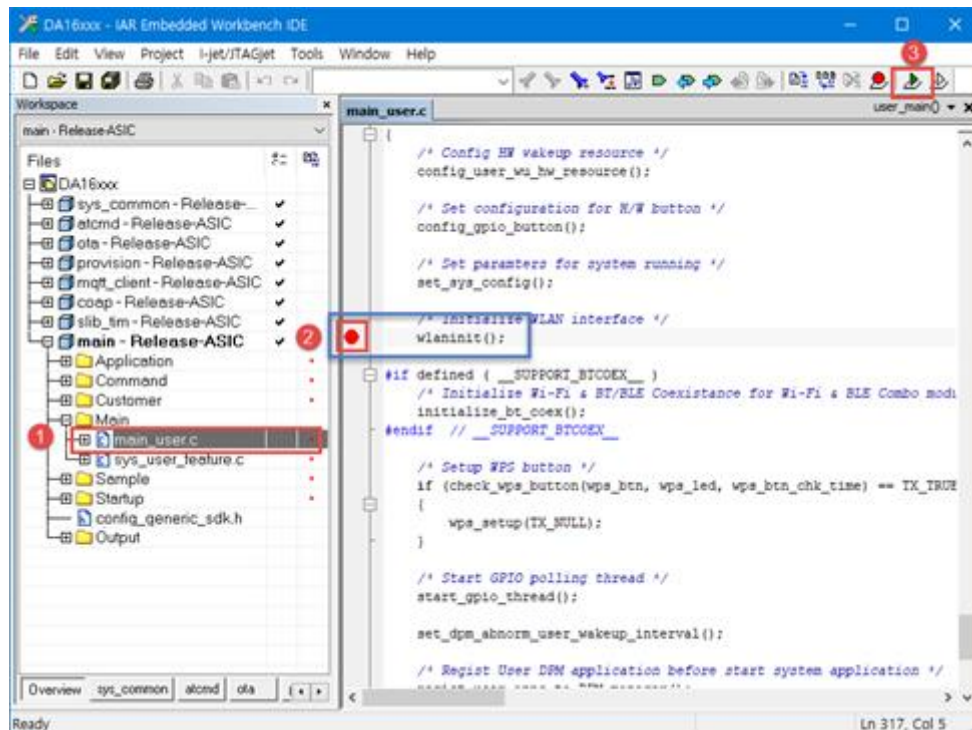


Figure 44: Download and Debug

## SDK Programmer Guide

- Click the check button in error pop-up window. See [Figure 45](#).

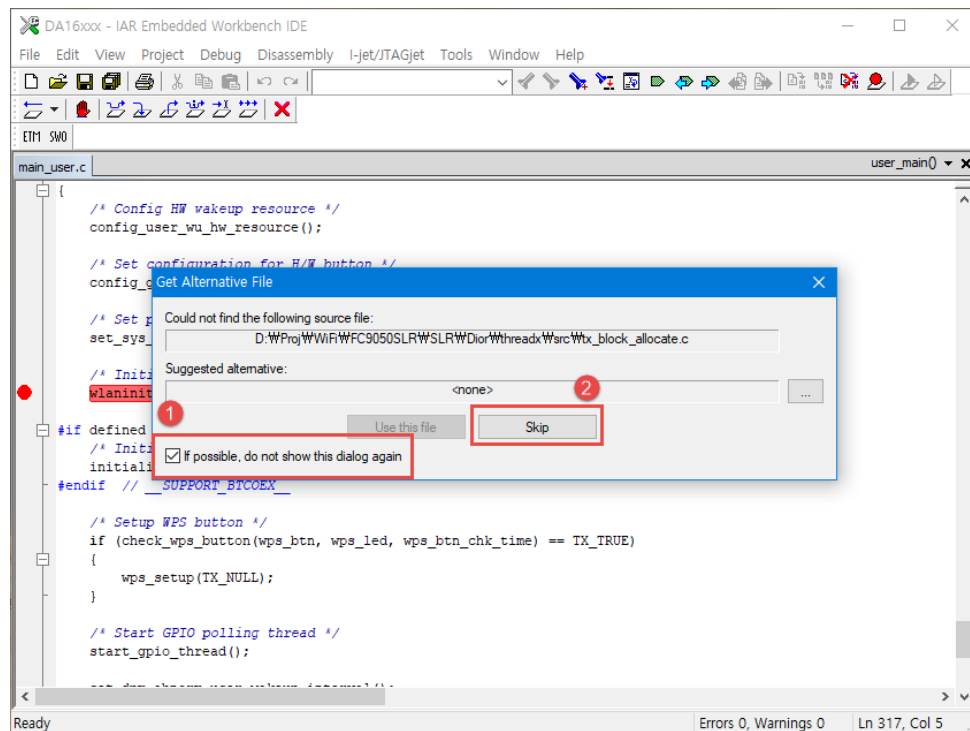


Figure 45: Pop-Up Message

- Click the **Go** button (1). See [Figure 46](#).

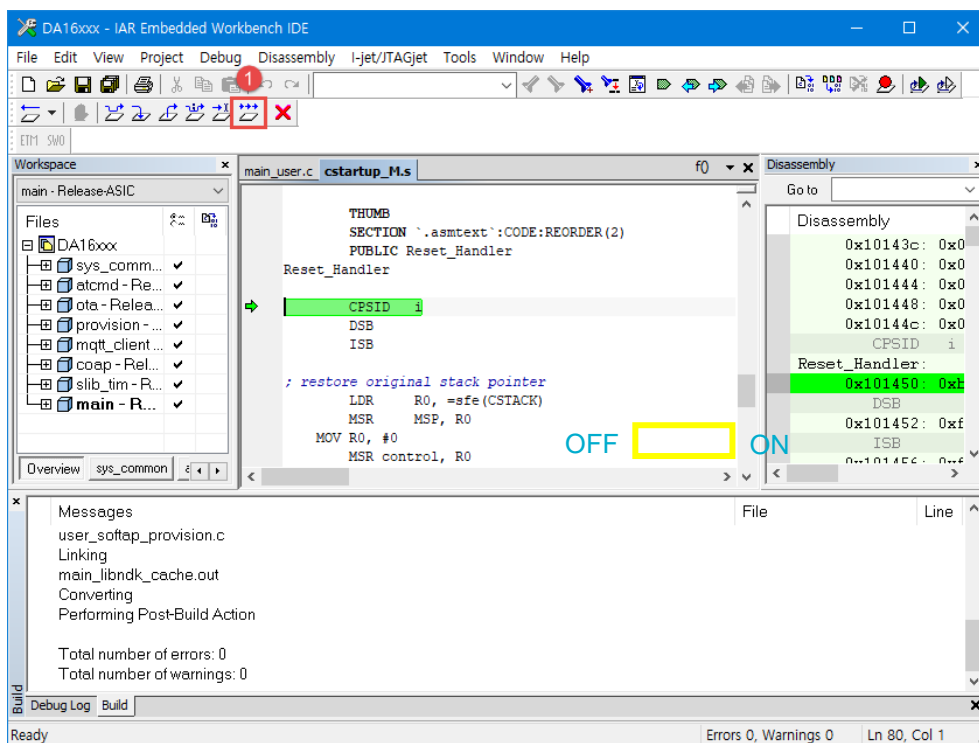


Figure 46: Download and Debug Windows

- Pause Window on Break Pointer. See [Figure 47](#).

## SDK Programmer Guide

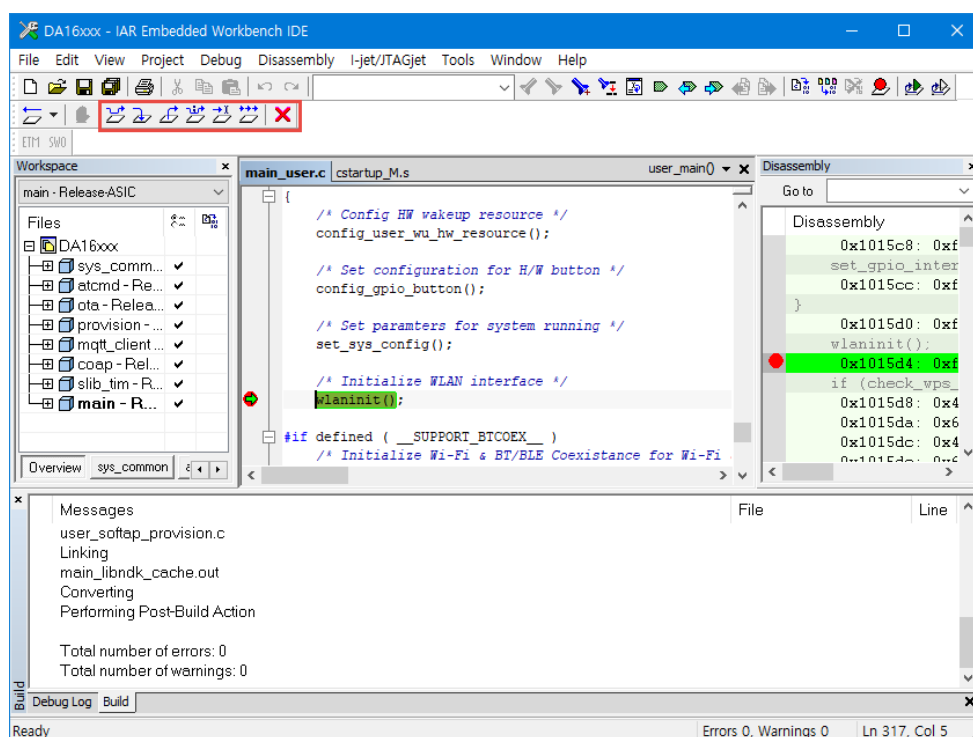


Figure 47: Break Point Window

### D.4 IAR build Setting

1. Some environments for the customer's PC needs to install a windows patch because one of tools to make an executable image in our SDK uses windows 32 bits DLL redistributables. For running 32bit dll, please install the following patch.
  - <https://www.microsoft.com/en-us/download/details.aspx?id=52685>

## SDK Programmer Guide

### Revision History

Revision	Date	Description
2.3	18-Mar-2021	Added appendix D.4 for IAR build setting
2.2	15-Dec-2020	Added Pulse Counter API Added ADC sleep mode API Delete OTA FW Update Paragraph
2.1	30-Nov-2020	Added 4.7 ADC Interrupt API descriptions
2.0	04-Nov-2020	Added Section 6.4 Memory copy using DMA
1.10	15-Sep-2020	Added "TLS" protocol to 7.3 Soft-AP Provisioning Protocol Changed figure image for Figure 11: Compile SDK on IAR Workbench Added 2.8 Make 4 MB SFLASH Images Changed 512 MB to 512 kB in 3.2 Memory Types
1.9	13-Aug-2020	Modification of "8. OTA FW Update" according to Generic SDK 2.2.0.0 update Add flow chart Add result codes Add API description
1.8	18-May-2020	Update for Generic SDK 2.2.0.0 Add description for "2.3 Startup Main" Add detail description for "3.3 Serial Flash Memory Map" Change unmatched pictures with SDK
1.7	20-Apr-2020	Add Appendix D.1 Notice to use Debugger on IAR workbench Updated contents for Generic SDK V2.0.0.0
1.6	06-Apr-2020	Add J-Link Debugger Setting
1.5	28-Oct-2019	Add 4.8.2 GPIO Retention API
1.4	17-Oct-2019	Finalized, removed draft status
1.3	03-Oct-2019	Editorial Review
1.2	30-Aug-2019	Add Chapter 8. OTA
1.1	19-Jul-2019	Add I-Jet Debug Setting
1.0	03-Jul-2019	Preliminary DRAFT Release

## SDK Programmer Guide

### Status Definitions

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

### Disclaimer

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

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

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

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

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

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

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

© 2021 Dialog Semiconductor. All rights reserved.

## Contacting Dialog Semiconductor

**United Kingdom (Headquarters)**  
Dialog Semiconductor (UK) LTD  
Phone: +44 1793 757700

**Germany**  
Dialog Semiconductor GmbH  
Phone: +49 7021 805-0

**The Netherlands**  
Dialog Semiconductor B.V.  
Phone: +31 73 640 8822

**Email:**  
[enquiry@diasemi.com](mailto:enquiry@diasemi.com)

**North America**  
Dialog Semiconductor Inc.  
Phone: +1 408 845 8500

**Japan**  
Dialog Semiconductor K. K.  
Phone: +81 3 5769 5100

**Taiwan**  
Dialog Semiconductor Taiwan  
Phone: +886 281 786 222

**Web site:**  
[www.dialog-semiconductor.com](http://www.dialog-semiconductor.com)

**Hong Kong**  
Dialog Semiconductor Hong Kong  
Phone: +852 2607 4271

**Korea**  
Dialog Semiconductor Korea  
Phone: +82 2 3469 8200

**China (Shenzhen)**  
Dialog Semiconductor China  
Phone: +86 755 2981 3669

**China (Shanghai)**  
Dialog Semiconductor China  
Phone: +86 21 5424 9058