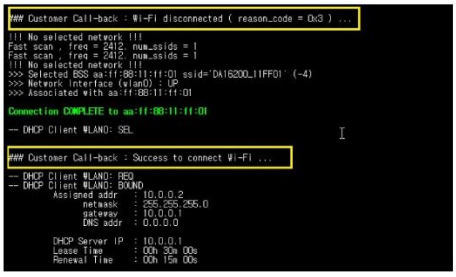
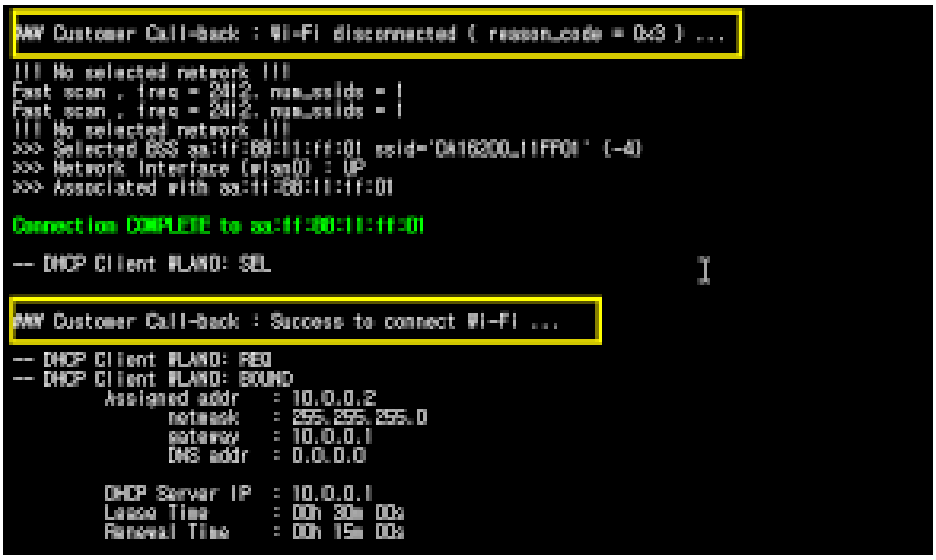


dialog DA16200 Wi-Fi Connection Notification User Manual

Contents

dialog DA16200 Wi-Fi Connection Notification



Abstract
This document describes how to use the Wi-Fi connection notification function of the DA16200.

Terms and Definitions

MCU Microcontroller Unit

References

[1] DA16200, SDK Programmer Guide, Dialog Semiconductor

Introduction

This document describes how to implement Wi-Fi Connection Notification in the customer’s program in the DA16200.

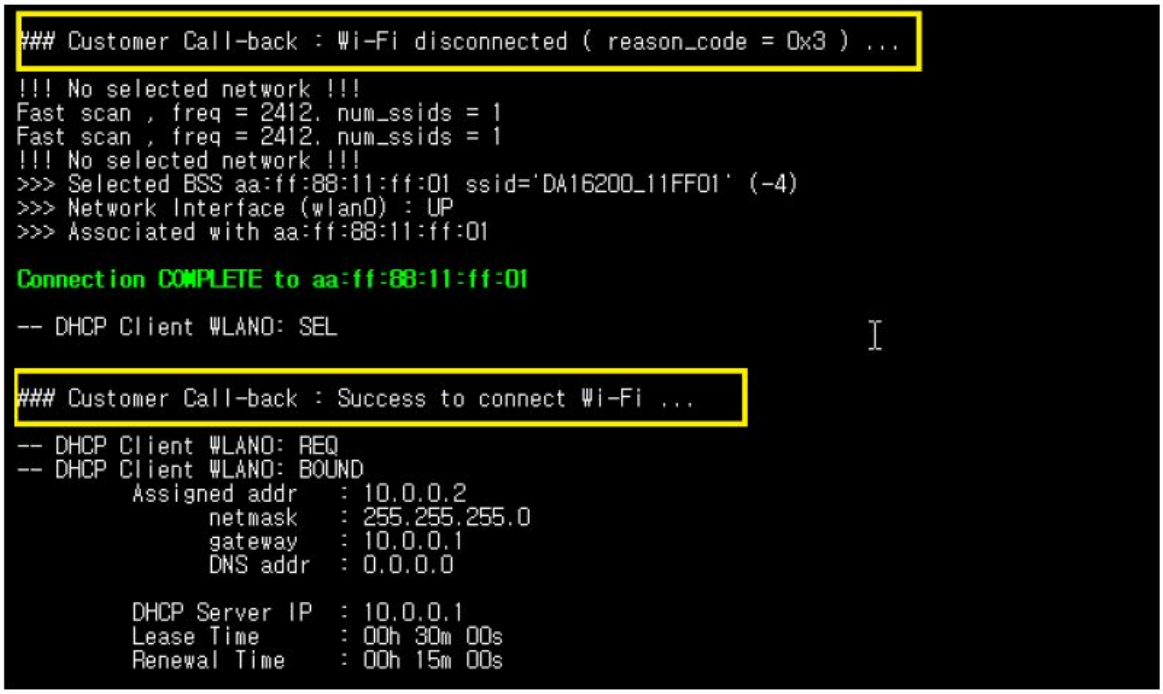
The DA16200 SDK provides the Wi-Fi connection status notification function when running STA mode. Using this function, customers/developer can implement their own operation to let MCU or other hardwired devices know the notified Wi-Fi connection status.

How to Create New F/W with Wi-Fi Notification Function

The DA16200 SDK provides a Wi-Fi connection status notification function as a compiled feature.
~/SDK/apps/da16200/get_started/inc/config_generic_sdk.h #undef __SUPPORT_WIFI_CONN_CB__

The customer or developer can change the compiled feature mentioned above to #define in order to create a new F/W.

Also, users should implement their own operations. For example, sending a specified event or data to the MCU or other hardwired device by using the defined protocol.
The released generic DA16200 SDK only shows a notification message on the console terminal.



Sequence for Wi-Fi Connection Status Notification

The DA16200 Wi-Fi connection status notification function runs as a library and the customer or developer needs to add their own code to send the result to the MCU or another hardwired device.
To use this function, the DA16200 SDK should include all source codes as shown in the following sequences.

- **Register call-back functions to use this feature**

  ~/SDK/apps/da16200/get_started/src/system_start.c

```
Int system_start(void)
{
      … …

      /* Regist Wi-Fi connect/disconnect status notify call-back functions */
      regist_wifi_notify_cb()
     … …
}
```

- Create a mutex-sema flags to prevent simultaneous access and register connect/disconnect callback function

  ~/SDK/core/common/main/util_api.c

```
void regist_wifi_notify_cb(void)
{
    wifi_conn_notify_mutex = malloc(sizeof(TX_MUTEX));
    if (wifi_conn_notify_mutex == NULL)
    {
        PRINTF("\n>>> Failed to allocate wifi_conn_notify_mutex buffer !\n");
        return;
    }
    memset(wifi_conn_notify_mutex, 0, sizeof(TX_MUTEX));

    status = tx_mutex_create(wifi_conn_notify_mutex,
                             "wifi_conn_cb_mutex", TX_NO_INHERIT);
    if (status != TX_SUCCESS)
    {
        PRINTF("\n>>> Failed to create Wi-Fi connection notify cb mutex !\n");
        return;
    }

    /* Wi-Fi connection call-back */
    wifi_conn_notify_cb_regist(wifi_conn_cb);

    /* Wi-Fi connection-fail call-back */
    wifi_conn_fail_notify_cb_regist(wifi_conn_fail_cb);

    /* Wi-Fi disconnection call-back */
    wifi_disconn_notify_cb_regist(wifi_disconn_cb);
}
```

- **Wi-Fi connection function**

  When Wi-Fi is connected, this function sends the notification “wifi_conn_flag = TRUE” and the Customer/Developer can use this flag for their function.

```
static void wifi_conn_cb(void)
{
    /* Wait until 3 seconds to get mutex */
    status = tx_mutex_get(wifi_conn_notify_mutex, 300);
    if (status != TX_SUCCESS)
    {
        PRINTF("\nFailed to get wifi_conn_notify_mutex during 3 secs !\n");
        return;
    }

    wifi_conn_flag = TRUE;

    tx_mutex_put(wifi_conn_notify_mutex);
}
```

- **Wi-Fi connection-fail function**

  When Wi-Fi is connected, this function sends the notification “wifi_conn_fail_flag = TRUE” and the Customer/Developer can use this flag for their function.

```
static void wifi_conn_fail_cb(ULONG reason_code)
{
    /* Wait until 3 seconds to get mutex */
    status = tx_mutex_get(wifi_conn_notify_mutex, 300);
    if (status != TX_SUCCESS)
    {
        PRINTF("\nFailed to get wifi_conn_notify_mutex during 3 seconds !!!\n");
        return;
    }

    wifi_conn_fail_flag     = TRUE;
    wifi_conn_fail_reason   = reason_code;

    tx_mutex_put(wifi_conn_notify_mutex);
}
```

- **Wi-Fi disconnection function**

  When Wi-Fi is disconnected, this function notifies “wifi_disconn_flag = TRUE” and sends the reason_code “wifi_disconn_reason”.

  These flag and reason_code are checked and used in the customer/developer function.

```
static void wifi_disconn_cb(ULONG reason_code)
{
    /* Wait until 3 seconds to get mutex */
    status = tx_mutex_get(wifi_conn_notify_mutex, 300);
    if (status != TX_SUCCESS)
    {
        PRINTF("\nFailed to get wifi_conn_notify_mutex during 3 sess !\n");
        return;
    }

    wifi_disconn_flag   = TRUE;
    wifi_disconn_reason = reason_code;

    tx_mutex_put(wifi_conn_notify_mutex);
}
```

NOTE: No other changes should be made to the sequences and functions listed in the above Section 5 Sequence for Wi-Fi Connection Status Notification.

User Function for Wi-Fi Connection Status

For the Wi-Fi connection status notification function, customers/developer should add their own operation in the DA16200 SDK before creating a new image.
In the DA16200 SDK, the customer/developer Wi-Fi connection status notification operation to MCU or hardwired device is created as independent threads to avoid affecting the basic Wi-Fi module operation.
~/SDK/apps/da16200/get_started/src/user_apps.c

```
const app_thread_info_t user_apps_table[] = {
/* name, func, stack_size, priority, net_chk_flag, dpm_flag, port_no, run_sys_mode
*/

   … …

#if defined ( __SUPPORT_WIFI_CONN_CB__ )
  { WIFI_CONN,      customer_wifi_conn,     1024,   USER_PRI_APP(0), FALSE, FALSE,
UNDEF_PORT, RUN_ALL_MODE    },
  { WIFI_CONN_FAIL, customer_wifi_conn_fail,1024,   USER_PRI_APP(0), FALSE, FALSE,
UNDEF_PORT, RUN_ALL_MODE    },
  { WIFI_DISCONN,   customer_wifi_disconn,  1024,   USER_PRI_APP(0), FALSE, FALSE,
UNDEF_PORT, RUN_ALL_MODE    },
#endif  // __SUPPORT_WIFI_CONN_CB__

   … …

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

NOTE: No other changes should be made to the above two thread creation items in user_apps_table except for the stack_size and the thread running priority.

- Wi-Fi connection status function:

  ~/SDK/apps/da16200/get_started/user_apps.c

  In the provided source code, the customer/developer should only change the Event/Data TX part to send a notification to MCU or hardwired device. Don’t clear the flag or change the mutex handling.

```
static void customer_wifi_conn(ULONG arg)
{
    while (1)
    {
        if (wifi_conn_flag == TRUE)
        {
#if 0
            //
            // Need customer's code about this event
            //
#else
            PRINTF("\n\n");
            PRINTF("### Customer Call-back : Success to connect Wi-Fi ...\n");
            PRINTF("\n");
#endif  // 0

            /*
             * Customer tunning value :
             *  Wait 100msec until sync with MCU
             */
```

```
            tx_thread_sleep(10);

            /* Clear event flag */
            tx_mutex_get(wifi_conn_notify_mutex, 300);

            wifi_conn_flag = FALSE;

            tx_mutex_put(wifi_conn_notify_mutex);
        }

        /* loop time delay : 10 msec */
        tx_thread_sleep(1);
    }
}
```

- Wi-Fi connection-fail status function:

  ~/SDK/apps/da16200/get_started/user_apps.c

  In the provided source code, the customer/developer should only change the Event/Data TX part to send a notification to MCU or hardwired device. Don't clear the flag or change the mutex handling.

```
static void user_wifi_conn_fail(ULONG arg)
{
    while (1)
    {
        if (wifi_conn_fail_flag == TX_TRUE)
        {
            /*
             * Customer tunning value :
             *  Wait 100msec until sync with MCU
             */
            tx_thread_sleep(10);

#if defined ( __SUPPORT_ATCMD__ )

#define WLAN_REASON_TIMEOUT                          39
#define WLAN_REASON_PEERKEY_MISMATCH                 45
#define WLAN_REASON_AUTHORIZED_ACCESS_LIMIT_REACHED 46

            switch (wifi_conn_fail_reason) {
                case WLAN_REASON_TIMEOUT :
                    PRINTF_ATCMD("\r\n+WFJAP:0,TIMEOUT\r\n"); break;
                case WLAN_REASON_PEERKEY_MISMATCH :
                    PRINTF_ATCMD("\r\n+WFJAP:0,WRONGPWD\r\n"); break;
                case WLAN_REASON_AUTHORIZED_ACCESS_LIMIT_REACHED :
                    PRINTF_ATCMD("\r\n+WFJAP:0,ACCESSLIMIT\r\n"); break;
                default :
                    PRINTF_ATCMD("\r\n+WFJAP:0,OTHER,%d\r\n", wifi_disconn_reason);
break;
            }
#else
            PRINTF("\n### User Call-back : Failed to connect Wi-Fi ( reason_code =
%d ) ...\n", wifi_conn_fail_reason);

#endif  // __SUPPORT_ATCMD__

            /* Clear event flag */
            tx_mutex_get(wifi_conn_notify_mutex, 300);

            wifi_conn_fail_reason = 0;
            wifi_conn_fail_flag = TX_FALSE;
```

```
            tx_mutex_put(wifi_conn_notify_mutex);
        }

        /* loop time delay : 10 msec */
        tx_thread_sleep(1);
    }
}
```

- Wi-Fi connection-fail status function:

  ~/SDK/apps/da16200/get_started/user_apps.c

  In the provided source code, the customer/developer should only change the Event/Data TX part to send a notification to MCU or hardwired device. Don't clear the flag or change the mutex handling.

```
static void customer_wifi_disconn(ULONG arg)
{
    while (1)
    {
        if (wifi_disconn_flag == TRUE)
        {
#if 0
            //
            // Need customer's code about this event
            //
#else
            PRINTF("\n\n");
            PRINTF("### Customer Call-back : Wi-Fi disconnected ( reason_code =
0x%x ) ...\n", wifi_disconn_reason);
            PRINTF("\n");
#endif  // 0

            /*
             * Customer tunning value :
             *  Wait 100msec until sync with MCU
             */
            tx_thread_sleep(10);

            /* Clear event flag */
            tx_mutex_get(wifi_conn_notify_mutex, 300);

            wifi_disconn_reason = 0;
            wifi_disconn_flag = FALSE;

            tx_mutex_put(wifi_conn_notify_mutex);
        }

        /* loop time delay : 10 msec */
        tx_thread_sleep(1);
    }
}
```

Revision History

Revision	Date	Description
1.1	18-Jun-2021	Add connection-fail call-back function SDK source path, Figures, ETC modified by SDK folder structure change
1.0	02-Sep-2020	Initial version.

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

RoHS Compliance

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

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

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

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

Documents / Resources

	dialog DA16200 Wi-Fi Connection Notification [pdf] User Manual DA16200 Wi-Fi Connection Notification, DA16200, WiFi Connection SoC Pptimized for Battery-Powered IoT Devices, Wi-Fi Connection Notification