

User Manual

DA16200 DA16600 DPM Manager

UM-WI-005

Abstract

This document describes how to set up and use the DPM Manager of DA16200 (DA16600).

Contents

Abstract	1
Contents	2
Figure	2
Tables	2
Terms and Definitions	4
References	4
1 Introduction	5
1.1 DPM Manager	5
1.2 DPM Manager Features.....	5
1.3 How DPM Manager Works.....	5
1.4 DPM Manager Operation	6
1.4.1 DPM Manager Operation.....	6
1.4.2 DPM Manager Workflow.....	6
1.4.3 The Threads of DPM Manager	9
2 Develop an Application for DPM Manager	10
2.1 Initialization	10
2.2 DPM Configuration.....	10
2.3 Callback Functions.....	11
2.4 DPM Manager API List.....	12
3 Example Code of DPM Manager	14
3.1 Enable DPM Manager.....	14
3.2 Start DPM Manager	14
3.3 Define DPM Configuration Callback Function.....	15
3.4 Option Definition of DPM Configuration	16
3.5 Define Callback Function Type	18
4 Process Abnormal DPM Operation	20
4.1 Abnormal DPM Operation	20
4.2 Implementation.....	20
Revision History	23

Figure

Figure 1: DPM Manager Structure	6
Figure 2: Abnormal DPM	20

Tables

Table 1: Initialization.....	10
Table 2: Configuration of DPM Manager for User Applications	10
Table 3: Callback Functions	12
Table 4: DPM Manager API List.....	12
Table 5: Enable DPM Manager	14
Table 6: Start DPM Manager.....	14

DA16200 DA16600 DPM Manager

Table 7: Configuration for Callback	15
Table 8: Option Definition	16
Table 9: Callback Declaration	18
Table 10: Enable the Abnormal DPM.....	20
Table 11: Set Wake-Up Interval	21
Table 12: Default Value of Library	22

Terms and Definitions

DPM	Dynamic Power Management
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
API	Application Programming Interface
POR	Power on Reset

References

- [1] DA16200, Datasheet, Dialog Semiconductor
- [2] DA16200 DA16600, SDK Programmer Guide, User Manual, Dialog Semiconductor

DA16200 DA16600 DPM Manager

1 Introduction

VirtualZero™ is a synthesis of breakthrough ultra-low-power technologies, which enables extremely low power operation in the DA16200 (DA16600) SoC. VirtualZero™ shuts down every microelement of the chip that is not in use, which allows a near-zero level of power consumption when the DA16200 (DA16600) SoC does not actively transmit or receive data. Such low power operation can deliver a year or more of battery life depending on the application. VirtualZero™ also enables ultra-low-power operation to transmit and receive data when the SoC needs to be awake to exchange information with other devices. Advanced algorithms enable one to stay asleep until the exact moment required to wake up to transmit or receive. DPM (Dynamic Power Management) Manager APIs make it easy to develop a DPM application.

1.1 DPM Manager

DPM Manager is developed for users to easily develop DPM applications. It has a simple interface; all that users need to do is to write the necessary callback functions and register them to the DPM Manager, which then takes care of all the DPM relevant jobs internally.

1.2 DPM Manager Features

- Provides a callback interface for application initialization
- Supports Timer registration interface
 - Periodic timer supported. Possibility to register up to four Timer callbacks
- Session (TCP/UDP) management
 - Possibility to register up to four TCP/UDP sessions (either server/client); up to four sessions for a TCP client. One TCP Server is assigned two sessions, so the use of two TCP servers is possible
 - After registration, the User Application can transmit and receive data in a session using a callback (Rx) and a Send API
 - When a “Connect/Accept” event happens, registered callbacks are invoked (TCP only)
- Non-volatile memory space (aka DPM Memory/Retention Memory) is supported for applications in-between DPM Sleep
 - The User Application lets DPM Manager know what address and size of DPM Memory to use, and then DPM Manager takes care of saving and loading the non-volatile memory when the system enters sleep or wakes up.
- Callback on an external signal is supported
- DPM Manager control APIs for User Applications
- All User DPM Manager configuration information is written in a header file

1.3 How DPM Manager Works

- Define and create a DPM Manager configuration header: `xxx_dpm_config.h` (see Section 3.4)
- In user main, invoke the configuration callback and start DPM Manager
- The user implements callbacks that the DPM Manager invokes when registered
- Once the user's job in a callback is finished, the User Application invokes the 'job done' API, and then the DPM Manager makes the system enter DPM Sleep
- Once callbacks are registered successfully on any event (timer/external wake-up/and so forth) during DPM Sleep, the registered user callbacks should be invoked when needed and the DPM Manager takes care of changing the power state of the system

DA16200 DA16600 DPM Manager

1.4 DPM Manager Operation

1.4.1 DPM Manager Operation

Figure 1 shows the DPM Manager architecture and operation. DPM manager offers easy control of the DPM function between User Applications and the Sleep Daemon. To enter DPM Sleep mode, the User Application should finish all active threads. After the User Application requests DPM Sleep mode, the DPM manager saves the data of the User Application that need to be maintained during DPM Sleep mode in retention memory and controls the callback function timer of the User Application, and manages the TCP/UDP session that processes the transmission and receipt of a packet.

To repeatedly process Sleep and Wake-up mode, the DPM manager offers to maintain a Wi-Fi connection without the reconnection process. When the User Application requests DPM Sleep mode, the DPM manager commands the Sleep Daemon to enter Sleep mode. Sleep Daemon is a system thread that operates the actual DPM sleep function. When the Register and Set bits of all threads are set to 1, the System is in DPM Sleep mode.

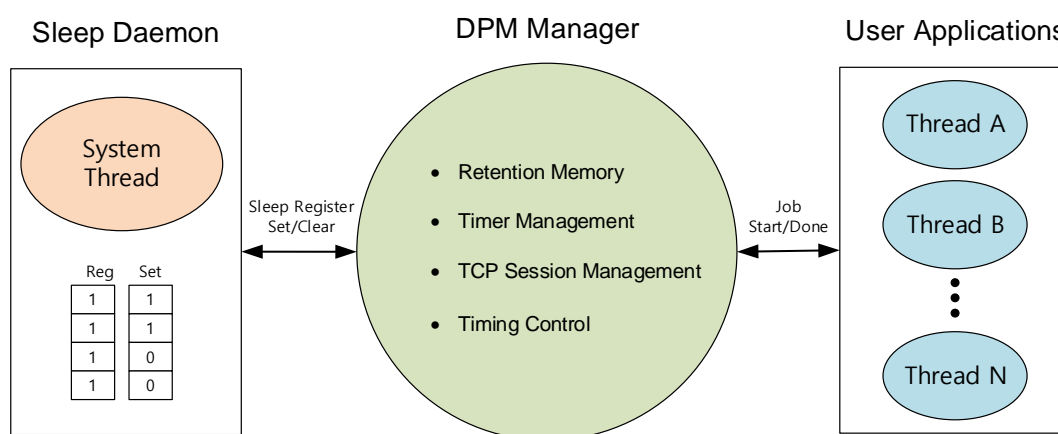


Figure 1: DPM Manager Structure

1.4.2 DPM Manager Workflow

DPM Manager works with the workflow mentioned below between System and User Application.

1.4.2.1 POR Boot

The System executes a Power on Reset (POR) boot.

1.4.2.2 DPM Configuration Request

DPM Manager requests DPM configuration information from the User Application:

- Boot initial callback function information
- Wake up initial callback function information
- Timer information, Retention Memory information
- TCP/UDP session information such as IP/port number/Callback function information
- External Wake-up callback function information

1.4.2.3 DPM Configuration Response

The User Application responds with DPM configuration information to the DPM Manager.

1.4.2.4 Retention Memory Allocation

Retention memory is allocated to save DPM configuration information.

DA16200 DA16600 DPM Manager

1.4.2.5 Call Boot Initial Callback Function

DPM Manager calls the Boot Initial Callback function to boot a device for the first time.

1.4.2.6 Timer Registration and Start TCP/UDP Session Thread

DPM Manager registers Timer information and starts the TCP/UDP Session Thread.

1.4.2.7 Processing Request Event

DPM Manager calls the Callback function of the target Thread to process the request Event.

1.4.2.8 Response Job Done of Request Event

The User Application's response signal for Job done of the processing request Event in the target Thread.

1.4.2.9 Save DPM Configuration

DPM Manager saves DPM configuration information of the device in Retention memory. Go to DPM Sleep mode.

1.4.2.10 Enter DPM Sleep Mode

The system enters DPM Sleep mode.

1.4.2.11 Wake Up Device

The system wakes up a device by Timer, Unicast signal, and external wake-up sources such as the HW button.

1.4.2.12 Restore DPM Configuration

DPM Manager restores the DPM configuration from Retention memory.

1.4.2.13 Call Wake Up Initial Callback Function

DPM Manager calls the Wake-up Initial Callback function for a quick system reboot.

1.4.2.14 Start TCP/UDP Session Thread

DPM Manager starts a TCP/UDP Session Thread to operate a Network Layer.

1.4.2.15 Processing Request Event

DPM Manager calls the Callback function of the target Thread to process the request Event.

1.4.2.16 Response Job Done of Request Event

User application responds by signaling "Job Done" in the callback function on the request Event.

1.4.2.17 Save DPM Configuration

DPM Manager saves the DPM configuration of the device in Retention memory. Go to DPM Sleep mode.

1.4.2.18 Enter DPM Sleep Mode

The system enters the DPM Sleep mode.

DA16200 DA16600 DPM Manager

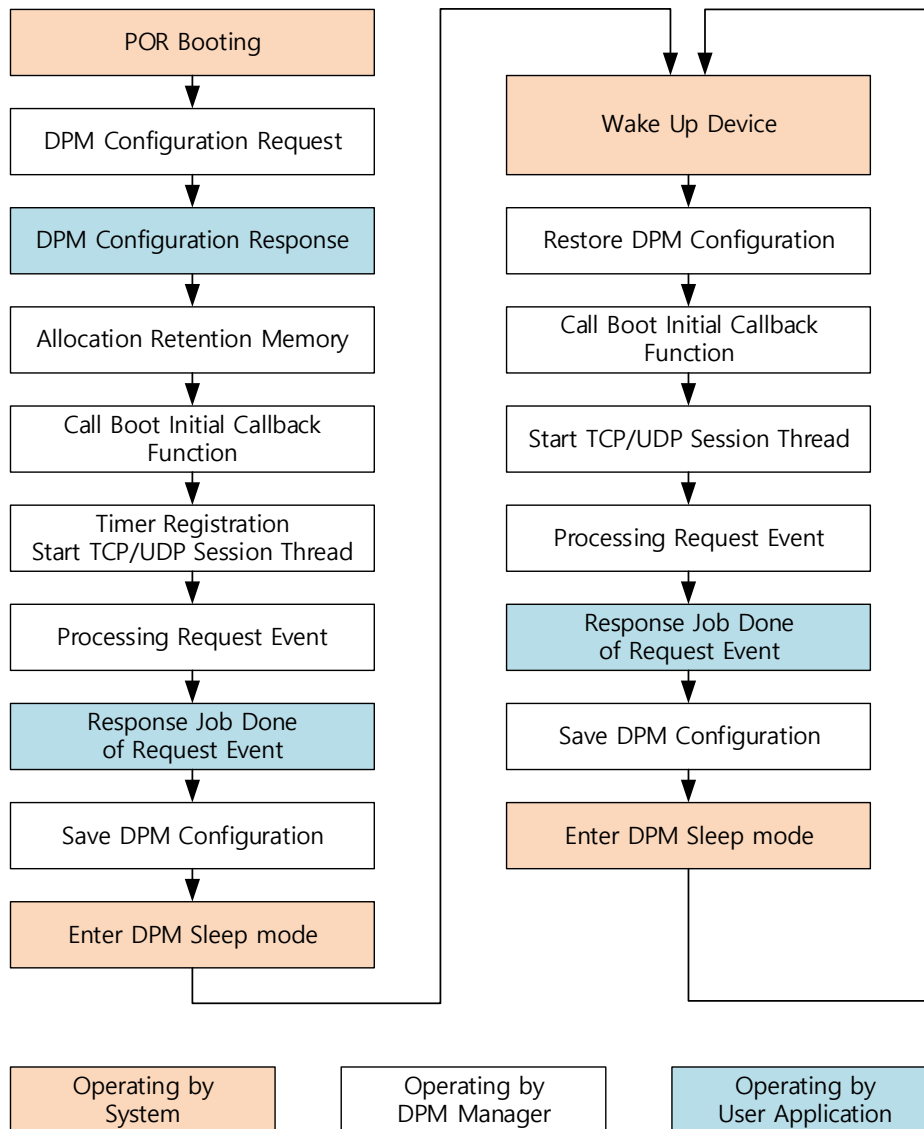


Figure 2: DPM Manager Workflow

1.4.3 The Threads of DPM Manager

DPM Manager creates three threads to process DPM Sleep mode as mentioned below.

1.4.3.1 dpmControlManager Thread

The `dpmControlManager` thread processes the following tasks:

- Get User DPM Configuration
- Allocate, restore, and save DPM configuration in Retention Memory
- Call initial function for Boot and Wake up device
- Register the Timer function and create `dpmEventManager` thread
- Create TCP/UDP Session thread
- Manage external wake up sources
- Control Timing

1.4.3.2 dpmEventManager

- Call Timer callback function
- Call external wake-up function

1.4.3.3 dpmSessionManager

The `dpmSessionManager` has four managers depending on the Network session:

- `tcpSvrSessionManager` – TCP server session manager
- `tcpCliSessionManager` – TCP client session manager
- `udpSvrSessionManager` – UDP server session manager
- `udpCliSessionManager` – UDP client session manager

The `dpmSessionManager` process has the following functions:

- Manage initial session resources such as Type, IP address, Port Number, and Socket Pool
- Call Callback function for connection and disconnection
- Call Receive Callback function
- Execute Send function

2 Develop an Application for DPM Manager

2.1 Initialization

The user should register a callback function to initialize the configuration for the DPM manager in the User Application through the `dpm_mng_regist_config_cb()` function. See [Table 1](#).

Table 1: Initialization

```
dpm_mng_regist_config_cb(initDpmUserConfig);
```

2.2 DPM Configuration

DPM Manager can define the DPM configuration of four threads.

[Table 2](#) shows an example of the DPM configuration for a User Application.

Table 2: Configuration of DPM Manager for User Applications

```
void init_DPM_user_config (dpm_user_config_t *dpmUserConf)
{
    /* Define Boot Initial Callback and Wakeup Callback */
    dpmUserConf->bootInitCallback = BOOT_INIT_FUNC;
    dpmUserConf->wakeupInitCallback = WAKEUP_INIT_FUNC;

    /* Define Timer Registration for Thread 0 of User Application */
    dpmUserConf->timerConfig[0].timerType = TIMER1_TYPE;
    dpmUserConf->timerConfig[0].timerInterval = TIMER1_INTERVAL;
    dpmUserConf->timerConfig[0].timerCallback = TIMER1_FUNC;

    /* Define Timer Registration for Thread 1 of User Application */
    dpmUserConf->timerConfig[1].timerType = TIMER2_TYPE;
    dpmUserConf->timerConfig[1].timerInterval = TIMER2_INTERVAL;
    dpmUserConf->timerConfig[1].timerCallback = TIMER2_FUNC;

    /* Define Timer Registration for Thread 2 of User Application */
    dpmUserConf->timerConfig[2].timerType = TIMER3_TYPE;
    dpmUserConf->timerConfig[2].timerInterval = TIMER3_INTERVAL;
    dpmUserConf->timerConfig[2].timerCallback = TIMER3_FUNC;

    /* Define Timer Registration for Thread 2 of User Application */
    dpmUserConf->timerConfig[3].timerType = TIMER4_TYPE;
    dpmUserConf->timerConfig[3].timerInterval = TIMER4_INTERVAL;
    dpmUserConf->timerConfig[3].timerCallback = TIMER4_FUNC;

    /* Define Network Session Configuration for Thread 0 of User Application */
    dpmUserConf->sessionConfig[0].session_type = REGIST_SESSION_TYPE1;
    dpmUserConf->sessionConfig[0].session_myPort = REGIST_MY_PORT_1;
    memcpy(dpmUserConf->sessionConfig[0].session_serverIp, REGIST_SERVER_IP_1,
           sizeof(REGIST_SERVER_IP_1));
    dpmUserConf->sessionConfig[0].session_serverPort = REGIST_SERVER_PORT_1;
    dpmUserConf->sessionConfig[0].session_ka_interval = SESSION1_KA_INTERVAL;
    dpmUserConf->sessionConfig[0].session_connectCallback = SESSION1_CONN_FUNC;
    dpmUserConf->sessionConfig[0].session_rcvCallback = SESSION1_RECV_FUNC;
    dpmUserConf->sessionConfig[0].supportSecure = SESSION1_SECURE_SETUP;
    dpmUserConf->sessionConfig[0].session_setupSecureCallback =
        SESSION1_SECURE_SETUP_FUNC;

    /* Define Network Session Configuration for Thread 1 of User Application */
```

DA16200 DA16600 DPM Manager

```

dpmUserConf->sessionConfig[1].session_type = REGIST_SESSION_TYPE2;
dpmUserConf->sessionConfig[1].session_myPort = REGIST_MY_PORT_2;
memcpy(dpmUserConf->sessionConfig[1].session_serverIp, REGIST_SERVER_IP_2,
        sizeof(REGIST_SERVER_IP_2));
dpmUserConf->sessionConfig[1].session_serverPort = REGIST_SERVER_PORT_2;
dpmUserConf->sessionConfig[1].session_ka_interval = SESSION2_KA_INTERVAL;
dpmUserConf->sessionConfig[1].session_connectCallback = SESSION2_CONN_FUNC;
dpmUserConf->sessionConfig[1].session_rcvCallback = SESSION2_RECV_FUNC;
dpmUserConf->sessionConfig[1].session_conn_retry_cnt =
    SESSION2_CONNECT_RETRY_COUNT; /* Only TCP Client */
dpmUserConf->sessionConfig[1].session_conn_wait_time =
    SESSION2_CONNECT_WAIT_TIME; /* Only TCP Client */
dpmUserConf->sessionConfig[1].session_auto_reconn =
    SESSION2_AUTO_RECONNECT; /* Only TCP Client */
dpmUserConf->sessionConfig[1].supportSecure = SESSION2_SECURE_SETUP;
dpmUserConf->sessionConfig[1].session_setupSecureCallback =
    SESSION2_SECURE_SETUP_FUNC;

/* Define Network Session Configuration for Thread 2 of User Application */
dpmUserConf->sessionConfig[2].session_type = REGIST_SESSION_TYPE3;
dpmUserConf->sessionConfig[2].session_myPort = REGIST_MY_PORT_3;
memcpy(dpmUserConf->sessionConfig[2].session_serverIp, REGIST_SERVER_IP_3,
        sizeof(REGIST_SERVER_IP_3));
dpmUserConf->sessionConfig[2].session_serverPort = REGIST_SERVER_PORT_3;
dpmUserConf->sessionConfig[2].session_ka_interval = SESSION3_KA_INTERVAL;
dpmUserConf->sessionConfig[2].session_connectCallback = SESSION3_CONN_FUNC;
dpmUserConf->sessionConfig[2].session_rcvCallback = SESSION3_RECV_FUNC;
dpmUserConf->sessionConfig[2].supportSecure = SESSION3_SECURE_SETUP;
dpmUserConf->sessionConfig[2].session_setupSecureCallback =
    SESSION3_SECURE_SETUP_FUNC;

/* Define Network Session Configuration for Thread 3 of User Application */
dpmUserConf->sessionConfig[3].session_type = REGIST_SESSION_TYPE4;
dpmUserConf->sessionConfig[3].session_myPort = REGIST_MY_PORT_4;
memcpy(dpmUserConf->sessionConfig[3].session_serverIp, REGIST_SERVER_IP_4,
        sizeof(REGIST_SERVER_IP_4));
dpmUserConf->sessionConfig[3].session_serverPort = REGIST_SERVER_PORT_4;
dpmUserConf->sessionConfig[3].session_ka_interval = SESSION4_KA_INTERVAL;
dpmUserConf->sessionConfig[3].session_connectCallback = SESSION4_CONN_FUNC;
dpmUserConf->sessionConfig[3].session_rcvCallback = SESSION4_RECV_FUNC;
dpmUserConf->sessionConfig[3].supportSecure = SESSION4_SECURE_SETUP;
dpmUserConf->sessionConfig[3].session_setupSecureCallback =
    SESSION4_SECURE_SETUP_FUNC;

/* Allocation Retention Memory for saving and restoring DPM Configuration */
dpmUserConf->ptrDataFromRetentionMemory = NON_VOLITALE_MEM_ADDR;
dpmUserConf->sizeOfRetentionMemory = NON_VOLITALE_MEM_SIZE;

/* Define Error state of Callback function */
dpmUserConf->externWakeupCallback = EXTERN_WU_FUNCTION;
dpmUserConf->errorCallback = ERROR_FUNCTION;
}

```

2.3 Callback Functions

Table 3 shows the callback function to register a timer.

DA16200 DA16600 DPM Manager

Table 3: Callback Functions

```

void timer1_callback()
{
    extern long iptolong(char *ip);
    char    txBuf[100];
    ULONG   ip;
    UINT    size;
    memset(txBuf, 0, 100);
    strcpy(txBuf, "Hello");
    ip = (ULONG)iptolong(REGIST_SERVER_IP_3);
    size = strlen(txBuf);
    /* tcp client */
    sendToSession(SESSION3, ip, REGIST_SERVER_PORT_3, txBuf, size);
    PRINTF(" [%s] Called by timer1...\n", __func__);
}

```

2.4 DPM Manager API List

DPM Manager provides the APIs mentioned in [Table 4](#) to get a callback. All functions should return 0 if done successfully.

Table 4: DPM Manager API List

API	Description
int dpm_mng_regist_config_cb(void (*regConfigFunction)())	Ask the DPM Manager to start after a register callback function
int dpm_mng_send_to_session(UINT sessionNo, ULONG ip, ULONG port, char *buf, UINT size)	Ask packet transmission
int dpm_mng_set_session_info_my_port_no(UINT sessionNo, ULONG port)	Register own port number for this session (only for server)
int dpm_mng_set_session_info_peer_port_no(UINT sessionNo, ULONG port)	Register peer's port number for this session (only for Server)
int dpm_mng_set_session_info_peer_ip_addr(UINT sessionNo, char *ip)	Register the peer's IP address for this session (only for Server)
int dpm_mng_set_session_info_server_ip_addr(UINT sessionNo, char *ip)	Register the server's IP address for this session (only for Client)
int dpm_mng_set_session_info_server_port_no(UINT sessionNo, ULONG port)	Register the server's port number for this session (only for Client)
int dpm_mng_set_session_info_local_port(UINT sessionNo, ULONG port)	Register own port number for this session (only for Client)

DA16200 DA16600 DPM Manager

API	Description
<pre>int dpm_mng_set_session_info(UINT sessionNo, ULONG type, ULONG myPort, char *peerIp, ULONG peerPort, ULONG kaInterval, void (*connCb)(), void (*recvCb)())</pre>	<p>Type: Set all the config info in one go to the session specified type</p> <p>1: TCP Server 2: TCP Client 3: UDP Server 4: UDP Client</p> <p>kaInterval: in seconds</p>
<pre>int dpm_mng_set_DPM_timer(UINT timerId, UINT timerType, UINT interval, void (*timerCallback)())</pre>	<p>timerId: from 1 to 4, up to four timers can be registered in total</p> <p>timerType: 1(periodic), 2(one-shot)</p> <p>interval: in seconds</p> <p>timerCallback: invoked when the timer is expired</p>
<pre>int dpm_mng_unset_DPM_timer(UINT timerId)</pre>	<p>timerId: from 1 to 4, up to 4 timers can be unregistered in total</p>
<pre>int dpm_mng_start_session(UINT sessionNo)</pre>	<p>Start the session</p>
<pre>int dpm_mng_stop_session(UINT sessionNo)</pre>	<p>Stop the session</p>
<pre>int dpm_mng_set_session_info_window_size(UINT sessionNo, UINT windowSize)</pre>	<p>Register Window size to the session (only for a TCP session. Session restart (stop/start) is needed to take effect</p>
<pre>int dpm_mng_set_session_info_conn_retry_count(UINT sessionNo, UINT connRetryCount)</pre>	<p>Set connection retry count for the session (only for TCP Client session)</p>
<pre>int dpm_mng_set_Session_info_conn_wait_time(UINT sessionNo, UINT connWaitTime)</pre>	<p>Set connection wait time for the session (only for TCP Client session)</p> <p>connWaitTime: in seconds</p>
<pre>int dpm_mng_set_Session_info_auto_reconnect(UINT sessionNo, UINT autoReconnect)</pre>	<p>You can specify the auto reconnect behavior of your session when the session is disconnected for some reason. If autoReconnect is 1, this function is activated. Only for TCP Client</p>
<pre>int dpm_mng_save_to_RTM()</pre>	<p>Store data in non-volatile memory. (It is also stored by the dpm_mng_job_done function.)</p>
<pre>int dpm_mng_init_done()</pre>	<p>Informs whether the initialization process of DPM Manager has been completed. return value (1: Done 0: Incomplete)</p>
<pre>int dpm_mng_job_done()</pre>	<p>A function to invoke when a job is done and to ask the DPM Manager for sleep</p>
<pre>int dpm_mng_job_start()</pre>	<p>A function to invoke when a job starts. When the DPM Manager gets this request, it keeps the system from entering the sleep mode. In case of a callback, this function is invoked by the DPM Manager before a callback is invoked, so the User Application does not need to call this function inside a callback</p>

3 Example Code of DPM Manager

3.1 Enable DPM Manager

In the SDK, enable the DPM Manager as shown in [Table 5](#).

Table 5: Enable DPM Manager

```
[\src\application\inc\sample_features.h]

#undef  __ALL_USED_DPM_MANAGER_SAMPLE__    /* definition */
#define __ALL_USED_LIGHT_DPM_MANAGER_SAMPLE__ /* Define to use DPM Manager */
```

Note that the recommendation is to choose the `__ALL_USED_DPM_MANAGER_SAMPLE__` or `__ALL_USED_LIGHT_DPM_MANAGER_SAMPLE__` feature depending on application requirement.

3.2 Start DPM Manager

The callback registration functions `dpm_mng_regist_config_cb()` and `dpm_mng_start()` are found in `all_used_dpm_manager_sample.c` and `all_used_light_dpm_manager_sample.c`. And, will be executed depending on the definition that you enable. [Table 6](#) gives an example of how to register the callback function and start the DPM manager.

Table 6: Start DPM Manager

```
int user_main(void)
{
    ...
    /* Initialize WLAN interface */
    wlaninit();
#ifdef __SUPPORT_DPM_MANAGER__
    dpm_mng_regist_config_cb(initDpmUserConfig);
    dpm_mng_start();
#endif /* __SUPPORT_DPM_MANAGER__ */
    ...
    start_user_apps();
    return TRUE;
}
```

3.3 Define DPM Configuration Callback Function

Table 7: Configuration for Callback

```

void initDpmUserConfig (dpm_user_config_t *dpmUserConf)
{
    dpmUserConf->bootInitCallback = BOOT_INIT_FUNC;
    dpmUserConf->wakeupInitCallback = WAKEUP_INIT_FUNC;
    dpmUserConf->timerConfig[0].timerType = TIMER1_TYPE;
    dpmUserConf->timerConfig[0].timerInterval = TIMER1_INTERVAL;
    dpmUserConf->timerConfig[0].timerCallback = TIMER1_FUNC;
    dpmUserConf->timerConfig[1].timerType = TIMER2_TYPE;
    dpmUserConf->timerConfig[1].timerInterval = TIMER2_INTERVAL;
    dpmUserConf->timerConfig[1].timerCallback = TIMER2_FUNC;
    dpmUserConf->timerConfig[2].timerType = TIMER3_TYPE;
    dpmUserConf->timerConfig[2].timerInterval = TIMER3_INTERVAL;
    dpmUserConf->timerConfig[2].timerCallback = TIMER3_FUNC;
    dpmUserConf->timerConfig[3].timerType = TIMER4_TYPE;
    dpmUserConf->timerConfig[3].timerInterval = TIMER4_INTERVAL;
    dpmUserConf->timerConfig[3].timerCallback = TIMER4_FUNC;

    dpmUserConf->sessionConfig[0].session_type = REGIST_SESSION_TYPE1;
    dpmUserConf->sessionConfig[0].session_myPort = REGIST_MY_PORT_1;
    memcpy(dpmUserConf->sessionConfig[0].session_serverIp,
           REGIST_SERVER_IP_1, sizeof(REGIST_SERVER_IP_1));
    dpmUserConf->sessionConfig[0].session_serverPort = REGIST_SERVER_PORT_1;
    dpmUserConf->sessionConfig[0].session_ka_interval = SESSION1_KA_INTERVAL;
    dpmUserConf->sessionConfig[0].session_connectCallback = SESSION1_CONN_FUNC;
    dpmUserConf->sessionConfig[0].session_rcvCallback = SESSION1_RECV_FUNC;
    dpmUserConf->sessionConfig[0].supportSecure = SESSION1_SECURE_SETUP;
    dpmUserConf->sessionConfig[0].session_setupSecureCallback =
        SESSION1_SECURE_SETUP_FUNC;

    dpmUserConf->sessionConfig[1].session_type = REGIST_SESSION_TYPE2;
    dpmUserConf->sessionConfig[1].session_myPort = REGIST_MY_PORT_2;
    memcpy(dpmUserConf->sessionConfig[1].session_serverIp,
           REGIST_SERVER_IP_2, sizeof(REGIST_SERVER_IP_2));
    dpmUserConf->sessionConfig[1].session_serverPort = REGIST_SERVER_PORT_2;
    dpmUserConf->sessionConfig[1].session_ka_interval = SESSION2_KA_INTERVAL;
    dpmUserConf->sessionConfig[1].session_connectCallback = SESSION2_CONN_FUNC;
    dpmUserConf->sessionConfig[1].session_rcvCallback = SESSION2_RECV_FUNC;
    dpmUserConf->sessionConfig[1].session_conn_retry_cnt =
        SESSION2_CONNECT_RETRY_COUNT; /* Only TCP Client */
    dpmUserConf->sessionConfig[1].session_conn_wait_time =
        SESSION2_CONNECT_WAIT_TIME; /* Only TCP Client */
    dpmUserConf->sessionConfig[1].session_auto_reconn =
        SESSION2_AUTO_RECONNECT; /* Only TCP Client */
    dpmUserConf->sessionConfig[1].supportSecure = SESSION2_SECURE_SETUP;
    dpmUserConf->sessionConfig[1].session_setupSecureCallback =
        SESSION2_SECURE_SETUP_FUNC;

    dpmUserConf->sessionConfig[2].session_type = REGIST_SESSION_TYPE3;
    dpmUserConf->sessionConfig[2].session_myPort = REGIST_MY_PORT_3;
    memcpy(dpmUserConf->sessionConfig[2].session_serverIp,
           REGIST_SERVER_IP_3, sizeof(REGIST_SERVER_IP_3));
    dpmUserConf->sessionConfig[2].session_serverPort = REGIST_SERVER_PORT_3;
    dpmUserConf->sessionConfig[2].session_ka_interval = SESSION3_KA_INTERVAL;
    dpmUserConf->sessionConfig[2].session_connectCallback = SESSION3_CONN_FUNC;
    dpmUserConf->sessionConfig[2].session_rcvCallback = SESSION3_RECV_FUNC;
    dpmUserConf->sessionConfig[2].supportSecure = SESSION3_SECURE_SETUP;
}

```

DA16200 DA16600 DPM Manager

```

dpmUserConf->sessionConfig[2].session_setupSecureCallback =
    SESSION3_SECURE_SETUP_FUNC;

dpmUserConf->sessionConfig[3].session_type = REGIST_SESSION_TYPE4;
dpmUserConf->sessionConfig[3].session_myPort = REGIST_MY_PORT_4;
memcpy(dpmUserConf->sessionConfig[3].session_serverIp,
    REGIST_SERVER_IP_4, sizeof(REGIST_SERVER_IP_4));
dpmUserConf->sessionConfig[3].session_serverPort = REGIST_SERVER_PORT_4;
dpmUserConf->sessionConfig[3].session_ka_interval = SESSION4_KA_INTERVAL;
dpmUserConf->sessionConfig[3].session_connectCallback = SESSION4_CONN_FUNC;
dpmUserConf->sessionConfig[3].session_rcvCallback = SESSION4_RECV_FUNC;
dpmUserConf->sessionConfig[3].supportSecure = SESSION4_SECURE_SETUP;
dpmUserConf->sessionConfig[3].session_setupSecureCallback =
    SESSION4_SECURE_SETUP_FUNC;

dpmUserConf->ptrDataFromRetentionMemory = NON_VOLITALE_MEM_ADDR;
dpmUserConf->sizeOfRetentionMemory = NON_VOLITALE_MEM_SIZE;

dpmUserConf->externWakeupCallback = EXTERN_WU_FUNCTION;
dpmUserConf->errorCallback = ERROR_FUNCTION;
}

```

3.4 Option Definition of DPM Configuration

All these configurations are defined in the `all_used_dpm_manager_sample.c` or `all_used_light_dpm_manager_sample.c` file as an example. So, users can define these definitions in a header file named `xxx_dpm_config.h`.

Table 8: Option Definition

```

#define TIMER_TYPE_NONE                0
#define TIMER_TYPE_PERIODIC            1
#define TIMER_TYPE_ONETIME            2
#define REG_TYPE_NONE                  0
#define REG_TYPE_TCP_SERVER            1
#define REG_TYPE_TCP_CLIENT            2
#define REG_TYPE_UDP_SERVER            3
#define REG_TYPE_UDP_CLIENT            4
#define DISABLE                         0
#define ENABLE                          1

/* Boot initial callback function */
#define BOOT_INIT_FUNC                  initConfigSampleByBoot
/* Wakeup initial callback function */
#define WAKEUP_INIT_FUNC                initConfigSampleByWakeup
/* timer1 type */
#define TIMER1_TYPE                     TIMER_TYPE_PERIODIC
/* timer1 interval */
#define TIMER1_INTERVAL                  10
/* timer1 callback function */
#define TIMER1_FUNC                     timer1_callback
/* timer2 type */
#define TIMER2_TYPE                     TIMER_TYPE_PERIODIC
/* timer2 interval */
#define TIMER2_INTERVAL                  15
/* timer2 callback function */
#define TIMER2_FUNC                     timer2_callback
/* timer3 type */
#define TIMER3_TYPE                     TIMER_TYPE_PERIODIC

```


DA16200 DA16600 DPM Manager

```

/* timer3 interval */
#define TIMER3_INTERVAL          10
/* timer3 callback function */
#define TIMER3_FUNC              timer3_callback
/* timer4 type */
#define TIMER4_TYPE              TIMER_TYPE_PERIODIC
/* timer4 interval */
#define TIMER4_INTERVAL          5
/* timer4 callback function */
#define TIMER4_FUNC              timer4_callback

/* Session Type (TCP Server) */
#define REGIST_SESSION_TYPE1    REG_TYPE_TCP_SERVER
/* My port no */
#define REGIST_MY_PORT_1        10197
/* Server ip : Client only */
#define REGIST_SERVER_IP_1      "0.0.0.0"
/* Server port : Client only */
#define REGIST_SERVER_PORT_1    0
/* Keep alive interval:TCP only, Sec */
#define SESSION1_KA_INTERVAL    0
/* Connect callback function */
#define SESSION1_CONN_FUNC      connect_callback_1
/* Receive callback function */
#define SESSION1_RECV_FUNC      recvPacket_callback_1
/* TLS enable/disable */
#define SESSION1_SECURE_SETUP   ENABLE
/* setup tls function */
#define SESSION1_SECURE_SETUP_FUNC setup_secure_callback_1

/* Session Type (TCP Client) */
#define REGIST_SESSION_TYPE2    REG_TYPE_TCP_CLIENT
/* My port no */
#define REGIST_MY_PORT_2        0
/* Server ip : Client only */
#define REGIST_SERVER_IP_2      "192.168.0.24"
/* Server port : Client only */
#define REGIST_SERVER_PORT_2    10196
/* Keep alive interval:TCP only, Sec */
#define SESSION2_KA_INTERVAL    0
/* Connect callback function */
#define SESSION2_CONN_FUNC      connect_callback_2
/* Receive callback function */
#define SESSION2_RECV_FUNC      recvPacket_callback_2
/* connect wait time(SEC): TCP Cli Only */
#define SESSION2_CONNECT_WAIT_TIME 4
/* connect retry count : TCP Client Only */
#define SESSION2_CONNECT_RETRY_COUNT 3
/* auto reconnect : TCP Client Only */
#define SESSION2_AUTO_RECONNECT ENABLE
/* TLS enable/disable */
#define SESSION2_SECURE_SETUP   ENABLE
/* setup tls function */
#define SESSION2_SECURE_SETUP_FUNC setup_secure_callback_2

/* Session Type (UDP Server) */
#define REGIST_SESSION_TYPE3    REG_TYPE_UDP_SERVER
/* My port no */
#define REGIST_MY_PORT_3        10197

```

DA16200 DA16600 DPM Manager

```

/* Server ip : Client only */
#define REGIST_SERVER_IP_3          "0.0.0.0"
/* Keep alive interval:TCP only, Sec */
#define REGIST_SERVER_PORT_3       0
/* Keep alive interval : TCP only */
#define SESSION3_KA_INTERVAL       0
/* Connect callback function */
#define SESSION3_CONN_FUNC         connect_callback_3
/* Receive callback function */
#define SESSION3_RECV_FUNC         recvPacket_callback_3
/* DTLS enable/disable */
#define SESSION3_SECURE_SETUP     ENABLE
/* setup tls function */
#define SESSION3_SECURE_SETUP_FUNC setup_secure_callback_3

/* Session Type (UDP Client) */
#define REGIST_SESSION_TYPE4       REG_TYPE_UDP_CLIENT
/* My port no */
#define REGIST_MY_PORT_4           0
/* Server ip : Client only */
#define REGIST_SERVER_IP_4        "192.168.0.24"
/* Server port : Client only */
#define REGIST_SERVER_PORT_4      10196
/* Keep alive interval:TCP only, Sec */
#define SESSION4_KA_INTERVAL       0
/* Connect callback function */
#define SESSION4_CONN_FUNC         connect_callback_4
/* Receive callback function */
#define SESSION4_RECV_FUNC         recvPacket_callback_4
/* DTLS enable/disable */
#define SESSION4_SECURE_SETUP     ENABLE
/* setup tls function */
#define SESSION4_SECURE_SETUP_FUNC setup_secure_callback_4

```

3.5 Define Callback Function Type

Table 9: Callback Declaration

```

void initConfigEn673ByBoot();
void initConfigEn673ByWakeup();
void timer1_callback();
void timer2_callback();
void timer3_callback();
void timer4_callback();
void connect_callback_1(void *sock, UINT conn_status);
void recvPacket_callback_1(void *sock, UCHAR *rx_buf,
                           UINT rx_len, ULONG rx_ip, ULONG rx_port);
void connect_callback_2(void *sock, UINT conn_status);
void recvPacket_callback_2(void *sock, UCHAR *rx_buf,
                           UINT rx_len, ULONG rx_ip, ULONG rx_port);
void connect_callback_3(void *sock, UINT conn_status);
void recvPacket_callback_3(void *sock, UCHAR *rx_buf, UINT rx_len,
                           ULONG rx_ip, ULONG rx_port);
void connect_callback_4(void *sock, UINT conn_status);
void recvPacket_callback_4(void *sock, UCHAR *rx_buf, UINT rx_len,
                           ULONG rx_ip, ULONG rx_port);
void connect_callback_5(void *sock, UINT conn_status);
void recvPacket_callback_5(void *sock, UCHAR *rx_buf, UINT rx_len,
                           ULONG rx_ip, ULONG rx_port);

```

DA16200 DA16600 DPM Manager

```
void connect_callback_6(void *sock, UINT conn_status);
void recvPacket_callback_6(void *sock, UCHAR *rx_buf, UINT rx_len,
                           ULONG rx_ip, ULONG rx_port);
void external_wu_callback();
void error_callback(UINT error_code, char *comment);
```

4 Process Abnormal DPM Operation

4.1 Abnormal DPM Operation

While DA16200 (DA16600) operates in DPM sleep, DA16200 (DA16600) executes an abnormal DPM mode if DA16200 (DA16600) is disconnected from the home AP. If DA16200 (DA16600) wakes up via an abnormal DPM mode, DA16200 (DA16600) tries to search the home AP within a predefined period and sleeps again for a predefined time. The DA16200 (DA16600) library provides a predefined value by default, but users can modify the related parameters based on their application.

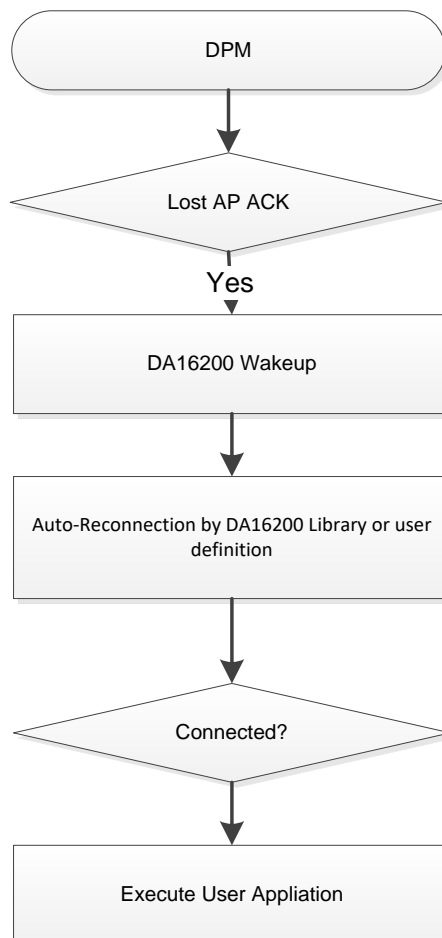


Figure 2: Abnormal DPM

4.2 Implementation

Table 10 gives an example to set `__USER_DPM_ABNORM_WU_INTERVAL__` to enable in `apps/da16200/get_started/inc/sys_common_features.h`.

Table 10: Enable the Abnormal DPM

```
apps/da16200/get_started/inc/sys_common_features.h
#define __USER_DPM_ABNORM_WU_INTERVAL__
```

Table 11 shows how to set the wake-up interval in `src/customer/main_user.c`.

DA16200 DA16600 DPM Manager
Table 11: Set Wake-Up Interval

```

apps/da16200/get_started/src/user_system_feature.c

#ifdef __USER_DPM_ABNORM_WU_INTERVAL__
/*
 * Format of dpm abnormal wakeup interval
 *   unsigned long long dpm_abnorm_wakeup_interval[10];
 *   {
 *       -1,                // Initial value : -1
 *       10, 10, 10, 10, 10,
 *       60,
 *       3600,
 *       3600,
 *       3600 * 24
 *   }
 */

unsigned long long _user_defined_wakeup_interval[DPM_MON_RETRY_CNT] =
{
    -1,                // Initial value : -1
    60,                // 1st Wakeup
    60,                // 2nd Wakeup : 0xdeadbeaf is no wakeup
    60,                // 3rd Wakeup : 0xdeadbeaf is no wakeup
    60 * 30,           // 4th Wakeup : 0xdeadbeaf is no wakeup
    60 * 30,           // 5th Wakeup : 0xdeadbeaf is no wakeup
    60 * 30,           // 6th Wakeup : 0xdeadbeaf is no wakeup
    60 * 60,           // 7th Wakeup : 0xdeadbeaf is no wakeup
    60 * 60,           // 8th Wakeup : 0xdeadbeaf is no wakeup
    0xDEADBEEF        // 9th Wakeup : 0xdeadbeaf is no wakeup
};

static void set_dpm_abnorm_user_wakeup_interval(void)
{
    extern unsigned long long *dpm_abnorm_user_wakeup_interval;

    dpm_abnorm_user_wakeup_interval =
        (unsigned long long *)_user_defined_wakeup_interval;
}
#endif /* __USER_DPM_ABNORM_WU_INTERVAL__ */

```

- The user can modify `_user_defined_wakeup_interval[10]` with the millisecond unit defined in `user_system_feature.c`
- If the parameter setting value is `0xdeadbeaf`, DA16200 (DA16600) executes the Power-off mode to not do the upcoming wake-up
- If the compile option is not defined, DA16200 (DA16600) operates based on the default setting (library). See [Table 12](#).

Table 12: Default Value of Library

```
unsigned long long _user_defined_wakeup_interval[DPM_MON_RETRY_CNT] =
{
    -1,                // Initial value : -1
    60,                // 1st Wakeup
    60,                // 2nd Wakeup : 0xdeadbeaf is no wakeup
    60,                // 3rd Wakeup : 0xdeadbeaf is no wakeup
    60 * 30,          // 4th Wakeup : 0xdeadbeaf is no wakeup
    60 * 30,          // 5th Wakeup : 0xdeadbeaf is no wakeup
    60 * 30,          // 6th Wakeup : 0xdeadbeaf is no wakeup
    60 * 60,          // 7th Wakeup : 0xdeadbeaf is no wakeup
    60 * 60,          // 8th Wakeup : 0xdeadbeaf is no wakeup
    0xDEADBEAF        // 9th Wakeup : 0xdeadbeaf is no wakeup
};
```

Revision History

Revision	Date	Description
1.7	28-Mar-2022	Update logo, disclaimer, copyright.
1.6	29-Nov-2021	Title was changed.
1.5	17-Jun-2021	Modified SDK source path and SDK folder structure
1.4	07-Apr-2020	Small changes in DPM Manager features, DPM Manager API list
1.3	31-Oct-2019	Removed Draft status; finalized for publication
1.2	21-Oct-2019	Editorial review
1.1	30-Aug-2019	Error correction
1.0	03-Jul-2019	First Release.

DA16200 DA16600 DPM Manager**Status Definitions**

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

RoHS Compliance

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

DA16200 DA16600 DPM Manager

Important Notice and Disclaimer

RENESAS ELECTRONICS CORPORATION AND ITS SUBSIDIARIES ("RENESAS") PROVIDES TECHNICAL SPECIFICATIONS AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES "AS IS" AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for developers skilled in the art designing with Renesas products. You are solely responsible for (1) selecting the appropriate products for your application, (2) designing, validating, and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. Renesas grants you permission to use these resources only for development of an application that uses Renesas products. Other reproduction or use of these resources is strictly prohibited. No license is granted to any other Renesas intellectual property or to any third party intellectual property. Renesas disclaims responsibility for, and you will fully indemnify Renesas and its representatives against, any claims, damages, costs, losses, or liabilities arising out of your use of these resources. Renesas' products are provided only subject to Renesas' Terms and Conditions of Sale or other applicable terms agreed to in writing. No use of any Renesas resources expands or otherwise alters any applicable warranties or warranty disclaimers for these products.

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu

Koto-ku, Tokyo 135-0061, Japan

www.renesas.com

Contact Information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:

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

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.