User Manual

DA16200 DA16600 DPM User Manual

UM-WI-030

Abstract

This document describes how to use DPM feature in DA16200 and DA16600.

Contents

Figures

Tables

1 Terms and Definitions

AP	Access Point
API	Application Programming Interface
BC	Broadcast Packet
BCN	Beacon
BUFP	Buffering Probe
BSS	Basic Service Set
DDPS	DPM Dynamic Period Setting
DHCP	Dynamic Host Configuration Protocol
DPM	Dynamic Power Management
DTIM	Delivery Traffic Indication Message
FFM	Fully Functional Mode
LPM	Low Power Mode
MCU	Micro Controller Unit
POR	Power on Reset
RTM	Retention Memory
RTOS	Real Time Operating System
SSID	Service Set Identifier
TCP	Transmission Control Protocol
TIM	Traffic Indication Map
UC	Unicast Packet
UDP	User Datagram Protocol
WAP	Wireless Application Protocol

2 References

[1] UM-WI-056, DA16200 DA16600 FreeRTOS Getting Started Guide, User Manual, Renesas Electronics
[2] UM-WI-003, DA16200 DA16600 Host Interfaces and AT Command, User Manual, Renesas Electronics
[3] UM-WI-006, DA16200 DA16600 Hardware Design Guide, User Manual, Renesas Electronics
[4] UM-WI-046, DA16200 DA16600 FreeRTOS SDK Programmer Guide, Renesas Electronics

3 DPM

Dynamic Power Management (DPM) is a technology to achieve low power consumption while connecting to access point (AP) or peer for a long time. If device has no actions for sending data to peer devices or communicating with external devices, the device can keep a low power state before receiving any data from peer. When DPM function is enabled, it is called DPM mode, otherwise is Non-DPM mode.

3.1 DPM Modes

DPM provides two different sub modes: FFM and LPM. DPM fully functional mode (FFM) allows a device to communicate over the network and with external devices, and DPM low power mode (LPM) enables a device to receive data from AP only.

In the DPM mode, DPM service starts and monitors the state of applications, and manages transition from DPM FFM to DPM LPM. In addition, DPM service controls timer function and transfers the received data to the related applications.

Figure 1 shows how the DPM mode works. DPM can be enabled only on Wireless Application Protocol (WAP) station mode and Wi-Fi connected state. Therefore, provisioning is required to configure AP profiles before enabling DPM, and a device reboot is required to run the DPM service and set up settings. To enable or disable DPM, use `dpm_mode_enable()` and `dpm_mode_disable()` functions. It might take a while to process various tasks like switching DPM FFM to DPM LPM (see Figure 4). For monitoring applications, register applications to DPM service first. To register applications, use `dpm_app_register()` or setting `dpm_flag` variable to TRUE in `user_app_table` in DA16200/DA16600 SDK. Each time the device enters DPM FFM, DPM service and applications are restarted and then, the applications must be registered to DPM service again. DPM service monitors the state of applications by receiving ready/not ready notifications `(dpm_app_sleep_ready_set()` and `dpm_app_sleep_ready_clear()` from the registered applications.

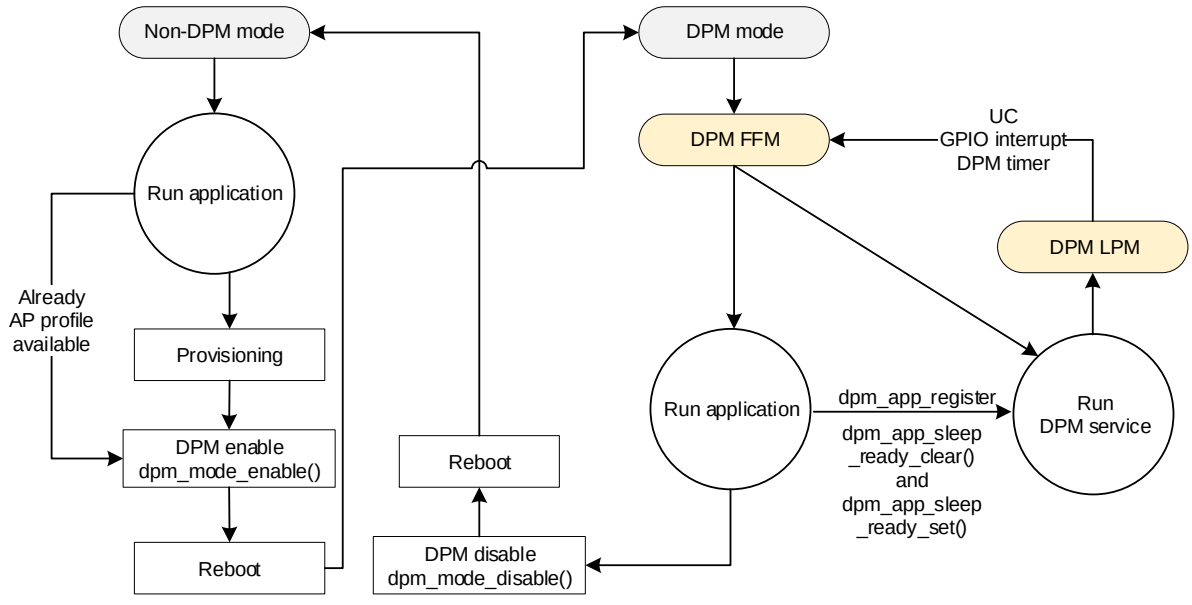

Figure 1: DPM Modes

Figure 2 shows power states in the DPM mode. The device can send data to peer devices over the network and communicate with external devices like peripherals or host device only in DPM FFM. Once the device enters DPM LPM, a firmware for Delivery Traffic Indication Message (DTIM) runs on retention RAM (or retention memory, RTM) with DTIM interval periodically. The PTIM is a tiny firmware image only for checking data from AP such as UC, BC/MC, or BCN. The device stays in sleep mode 3 when PTIM is not active in DPM LPM.

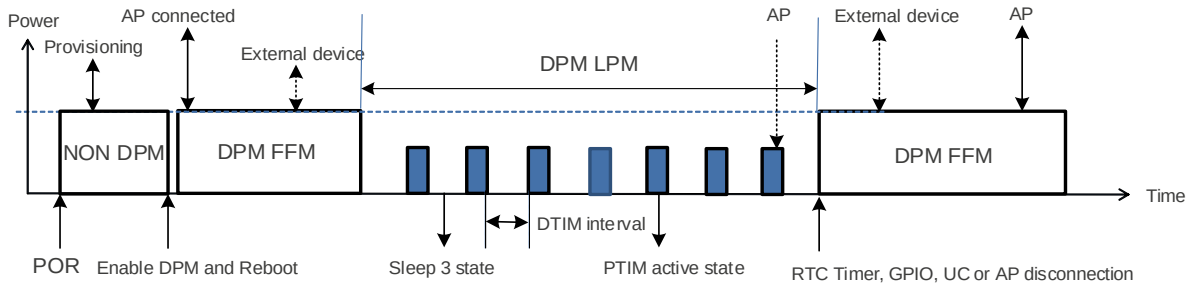

Figure 2: Power States in DPM Mode

3.2 DPM Service and Application

When data from AP or peer device are received, registered DPM timer is expired or GPIO interrupts occur in DPM LPM, the device wakes up from DPM LPM and DPM service runs in DPM FFM. Figure 3 shows how DPM service manages the received data and callback of DPM timer, and monitors applications in DPM FFM.

If network session is connected with peer devices and there are received data from peer, the application needs to be registered with the port number within **200 ms** because DPM service checks the port number where the data is received from peer, and then transfers the data to the application with the same port number. Also, the application has to notify that it is in ready state for receiving the data to DPM service within **500 ms** using `dpm_app_data_rcv_ready_set()` after waking up from DPM LPM. Otherwise DPM service drops the data because network stack cannot keep the data for a long time.

DPM timer is registered by the application and can be expired in DPM FFM or DPM LPM. When the timer is expired, DPM service checks whether the application is registered and ready to get the callback. Therefore, the application has to be registered at every wake-up from DPM LPM and notify DPM service that it is ready using `dpm_app_wakeup_done`.

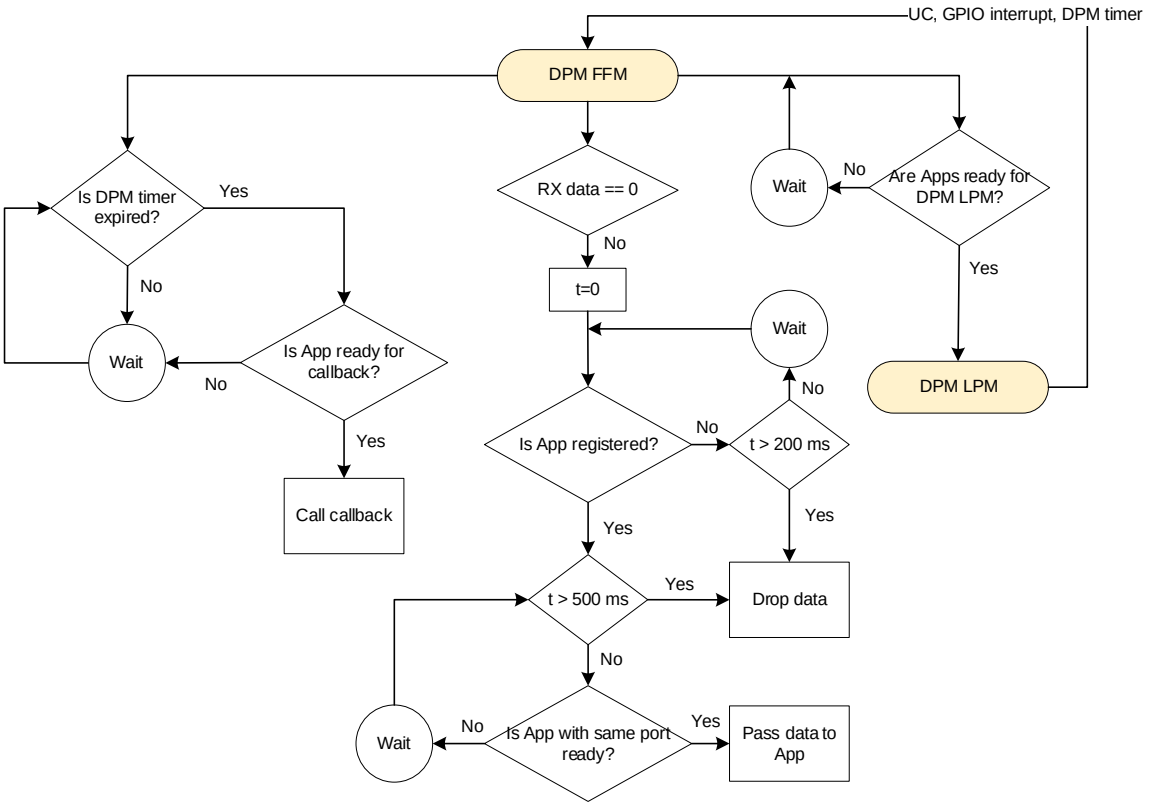

Figure 3: DPM Service in DPM FFM

Figure 4 shows how to register applications and notify the state of applications to DPM service.

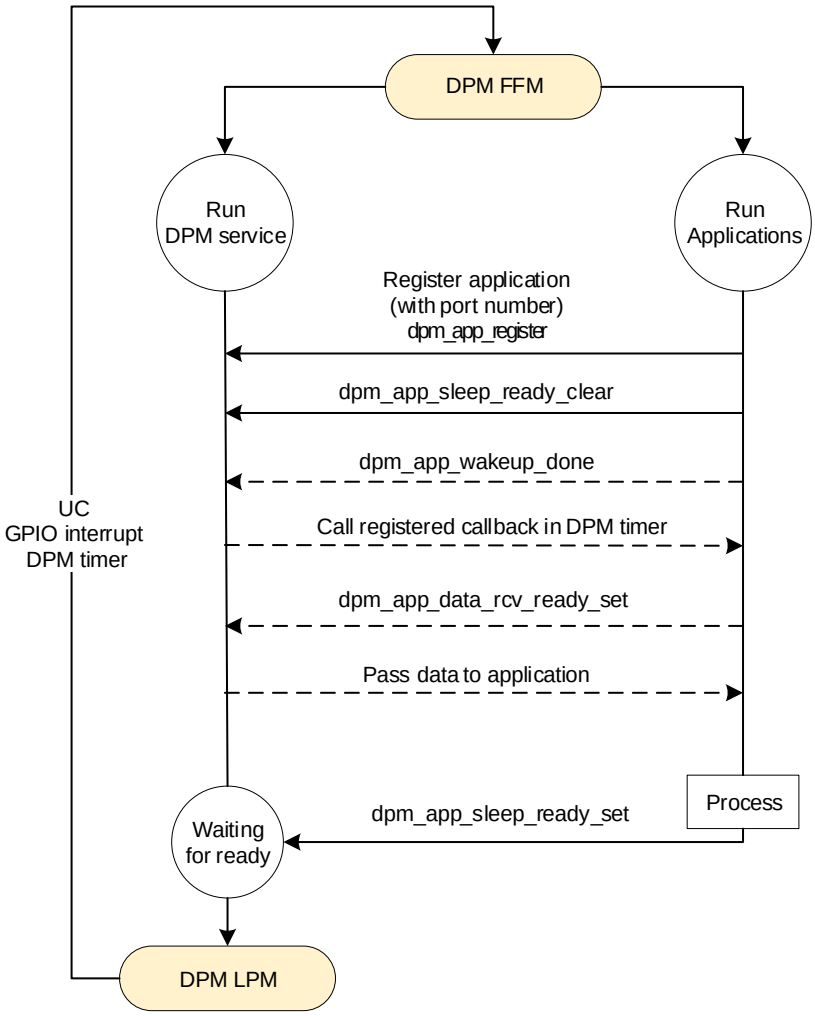

Figure 4: Application in DPM FFM

3.3 Application Programming Interface

3.3.1 DPM Management

The APIs in Table 1 are for managing DPM as shown in Figure 1.

Table 1: APIs for DPM Management

void dpm_mode_enable(void)	
Description	Enable DPM
void dpm_mode_disable(void)	
Description	Disable DPM
int dpm_mode_is_enabled(void)	
Return	1 (pdTRUE): DPM is enabled 0 (pdFALSE): DPM is disabled
Description	Return DPM is enabled or disabled

void dpm_mode_enable(void)		
int dpm_mode_is_wakeup(void)		
Return		1 (pdTRUE): When DA16200/DA16600 wakes up from DPM LPM 0 (pdFALSE): When DA16200/DA16600 wakes up by other cases like POR
Description		Return whether device wakes up from DPM LPM
int dpm_mode_get_wakeup_source(void)		
Return		0x00 (WAKEUP_RESET): Internal reset 0x01 (WAKEUP_SOURCE_EXT_SIGNAL): Boot by external wake-up signal 0x02 (WAKEUP_SOURCE_WAKEUP_COUNTER): Boot by wake-up counter 0x03 (WAKEUP_EXT_SIG_WAKEUP_COUNTER): Boot by wake-up counter or external wake-up signal 0x04 (WAKEUP_SOURCE_POR): Boot by power on reset 0x08 (WAKEUP_WATCHDOG): Boot by watchdog Others: Declared in the enumeration WAKEUP_SOURCE
Description		Return wake-up source
int dpm_mode_get_wakeup_type(void)		
Return		DPM_RTCTIME_WAKEUP: Boot by DPM timer DPM_PACKET_WAKEUP: Boot by receiving data Others: Declared in the enumeration DPM_WAKEUP_TYPE
Description		Return wake-up source when device wakes up from DPM LPM
int dpm_sleep_is_started(void)		
Return		0 (WAIT_DPM_SLEEP): Wait for entering DPM LPM 1 (RUN_DPM_SLEEP): Run for entering DPM LPM 2 (DONE_DPM_SLEEP): Done for entering DPM LPM
Description		Return state of DPM service
int dpm_sleep_start_mode_2(unsigned long long usec, unsigned char retention)		
Return		0: Succeed
Parameter	usec	Wake-up time; how long DA16200 is in sleep mode 2 or sleep mode 3 If 0, DA16200 wakes up only by external GPIO signal
	retention	Power on/off RTM in sleep
Description		Make device enter sleep mode 2 or sleep mode 3. Device can be woke up only by external GPIO signal or DPM timer

3.3.2 Application Registration and Status Notification

The APIs in Table 2 are for registering applications to DPM service and sending the status of applications to DPM service as shown in Figure 3 and Figure 4.

Table 2: APIs for Application Registration and Status Notification

int dpm_app_register(char *mod_name, unsigned int port_number)	
Return	0 (DPM_REG_OK): Succeeded 9999 (DPM_REG_DUP_NAME): Failed due to the duplicated name of application in DPM service Others: Failed due to other causes

int dpm_app_register(char *mod_name, unsigned int port_number)		
Parameter	mod_name	Name of application to be registered to DPM service. Note ● The name must be less than 19 characters and unique ● The maximum number of registered applications is 11 ● The port number of TCP and UDP must be unique
	port_number	Port number of applications. If not required, the value can be 0.
Description		Register the application to DPM service. DPM service identifies a registered application with a name
void dpm_app_unregister(char *mod_name)		
Parameter	mod_name	Name of registered application
Description		Deregister the application from DPM service
int dpm_app_is_register(char *mod_name)		
Return		9999 (DPM_REG_DUP_NAME): Registered application Others: Failed due to other causes
Parameter	mod_name	Name of registered application
Description		Return whether the application is registered to DPM service
char *dpm_app_is_register_port(unsigned int port)		
Return		Pointer of name of application if the port number is registered
Parameter	mod_name	Port number of registered applications
Description		Check the registered name of application is registered with the port number
int dpm_app_sleep_ready_set(char *mod_name)		
Return		0 (DPM_SET_OK): Succeeded Other: Failed due to other causes
Parameter	mod_name	Name of registered application
Description		Set the application to be ready for entering DPM LPM
int dpm_app_is_sleep_ready_set(char *mod_name)		
Return		1: Application is set as ready for DPM LPM 0: Application is not set as ready for DPM LPM
Parameter	mod_name	Name of registered application
Description		Return the set state by application
int dpm_app_sleep_ready_clear(char *mod_name)		
Return		0 (DPM_SET_OK): Succeeded Others: Failed due to other causes
Parameter	mod_name	Name of registered application
Description		Set the application not to be ready for DPM LPM
int dpm_app_data_rcv_ready_set(char *mod_name)		
Return		0 (DPM_SET_OK): Succeeded Others: Failed due to other causes
Parameter	mod_name	Name of registered application

int dpm_app_register(char *mod_name, unsigned int port_number)		
Description		Set the application to be ready for receiving data
int dpm_app_data_rcv_ready_set_by_port(unsigned int port)		
Return		0 (DPM_SET_OK): Succeeded Others: Fail due to other causes
Parameter	port	Port number of registered application
Description		Set the application's port number to be ready for receiving data
int dpm_app_wakeup_done(char *mod_name)		
Return		0 (DPM_SET_OK): Success Others: Fail due to other causes
Parameter	mod_name	Name of registered application
Description		Set the application to be ready for callback of DPM timer
bool dpm_app_is_wakeup_done(char *mod_name)		
Return		1: registered application is ready to get callback of DPM timer
Parameter	mod_name	Name of registered application
Description		Check the registered application is ready for callback of DPM timer

3.3.3 User Data in Retention RAM

There is 64 kB retention RAM in DA16200 and the data is kept during DPM mode. 8 kB is assigned to user area and each data can be allocated with specific name.

The APIs in Table 3 are for adding or removing user data to/from retention RAM in the DPM mode.

Table 3: APIs for Handling User Data in Retention Memory

unsigned int dpm_user_mem_alloc(char *name, void **memory_ptr, unsigned long memory_size, unsigned long wait_option)		
Return		0: Succeeded
Parameter	name	Specified name for memory allocation
	memory_ptr	Pointer of allocated memory
	memory_size	Size of allocated memory (bytes)
	wait_option	Suspension option. Deprecated in FreeRTOS SDK
Description		Allocate memory for user data in retention RAM
unsigned int dpm_user_mem_free(char *name)		
Return		0: Succeeded
Parameter	name	Specified name of allocated memory
Description		Release allocated memory
unsigned int dpm_user_mem_get(char *name, unsigned char **data)		
Return		Length of allocated memory (bytes)
Parameter	name	Specified name of allocated memory
	memory_ptr	Pointer of allocated memory
Description		Get data from allocated memory

unsigned int dpm_user_mem_alloc(char *name, void **memory_ptr, unsigned long memory_size, unsigned long wait_option)		
int dpm_user_mem_init_check(void)		
Return		1 (pdTRUE): access to user data in retention RAM is ready
Description		Check user data in retention RAM can be accessed

3.3.4 DPM Timer

The APIs in Table 4 are for using DPM Timer (RTC Timer) which can be used for periodically or one-time wake-up from DPM LPM.

Table 4: DPM Timer APIs

int dpm_timer_create(char *task_name, char *timer_name, void (* callback_func)(char *timer_name), unsigned int msec, unsigned int reschedule_msec)		
Return		5 ~ 15: Assigned Timer ID
Parameter	task_name	Name of the registered application
	timer_name	Timer name within 7 bytes as a unique character to distinguish timer
	callback_func	Function pointer to be called when timeout occurs NULL means no callback function is registered
	msec	Timeout time (milli seconds)
	reschedule_msec	Periodic timeout time (milli seconds) If it is set to 0, only one timeout occurs according to timeout time
Description		Register DPM timer
int dpm_timer_delete(char *task_name, char *timer_name)		
Return		5 ~ 15: Assigned Timer ID
Parameter	task_name	Name of the registered application
	timer_name	Timer name
Description		Delete the registered DPM timer.
int dpm_timer_change(char *task_name, char *timer_name, unsigned int msec)		
Return		5 ~ 15: Assigned Timer ID
Parameter	task_name	Name of the registered application
	timer_name	Timer name
	msec	Time value to change (milli seconds)
Description		Change the timeout time of DPM timer
int dpm_timer_remaining_msec_get(char *thread_name, char *timer_name)		
Return		Remained time to timeout (milli seconds)
Parameter	task_name	Name of the registered application
	timer_name	Timer name
Description		Get remained timeout time of DPM timer

3.3.5 Port Filtering

The APIs listed in Table 5 are for filtering a specific port number of TCP/UDP or IP multicast address. To achieve low power consumption, the device wakes up from DPM LPM by responding only to the registered TCP/UDP port number or IP multicast address in DPM LPM.

Table 5: APIs for Port Filtering

void dpm_udp_filter_enable(unsigned char en_flag)		
Parameter	en_flag	1: Enable UDP filter functionality 0: Disable UDP filter functionality
Description		Enable/Disable UDP filter functionality
void dpm_udp_port_filter_set(unsigned short d_port)		
Parameter	d_port	Port number of UDP. The maximum is DPM_MAX_UDP_FILTER (8)
Description		Set port number of UDP to allow to receive UDP packet in DPM LPM
void dpm_udp_port_filter_delete(unsigned short d_port)		
Parameter	d_port	Port number of UDP
Description		Delete port number of UDP
void dpm_tcp_filter_enable(unsigned char en_flag)		
Parameter	en_flag	1: Enable TCP filter functionality 0: Disable TCP filter functionality
Description		Enable/Disable TCP filter functionality
void dpm_tcp_port_filter_set(unsigned short d_port)		
Parameter	d_port	Port number of TCP. The maximum is DPM_MAX_TCP_FILTER (8)
Description		Set port number of TCP
void dpm_tcp_port_filter_delete(unsigned short d_port)		
Parameter	d_port	Port number of TCP
Description		Delete registered port number of TCP
void dpm_mc_filter_set(unsigned long mc_addr)		
Parameter	mc_addr	IP multicast address
Description		Set IP multicast address to allow receiving packet

3.3.6 Wake-Up Types of DPM LPM

When device wakes up from DPM LPM, wake-up types (see Table 6) can be checked by calling `da16x_get_wakeup_source()` API. The information about general wake-up sources is available in Ref. [4].

Table 6: Wake-up Types of DPM LPM

DPM_WAKEUP_TYPE	Description
DPM_UC	Wake up when there is Unicast data to process from the AP
DPM_BC_MC	Wake up when there is BC (Broadcast)/MC (Multicast) data to process from the AP
DPM_BCN_CHANGED	Wake up when BCN (beacon) frame is changed such as SSID and channel number
DPM_NO_BCN	Wake up when BCN is not received continuously

DPM_WAKEUP_TYPE	Description
DPM_FROM_FAST	Wake up when there is an RTC sleep counter expired wake-up set by a user application (Previous status is that no beacon was received in DPM FFM)
DPM_KEEP_ALIVE_NO_ACK	Wake up when Keep-alive ACK is not received from AP
DPM_DEAUTH	Wake up when De-authentication frame is received
DPM_FROM_FULL	Wake up when there is an RTC sleep counter expired wake-up set by a user application (Previous status is that no beacon was received in PTIM)

3.4 DPM Connection Retry State

If the device loses connection to AP in the DPM mode, initiate the connection again for a specified period in Connection Retry state, which is called DPM abnormal state in SDK. If the connection cannot be re-established within the specified period, the device enters into sleep mode 3. If duration in sleep is defined, the device wakes up from sleep mode 3 after the duration and tries to establish the connection again. If the duration is not defined, the device stays in sleep mode 3 until external GPIO event occurs. Figure 6 shows the flow of the connection retry process.

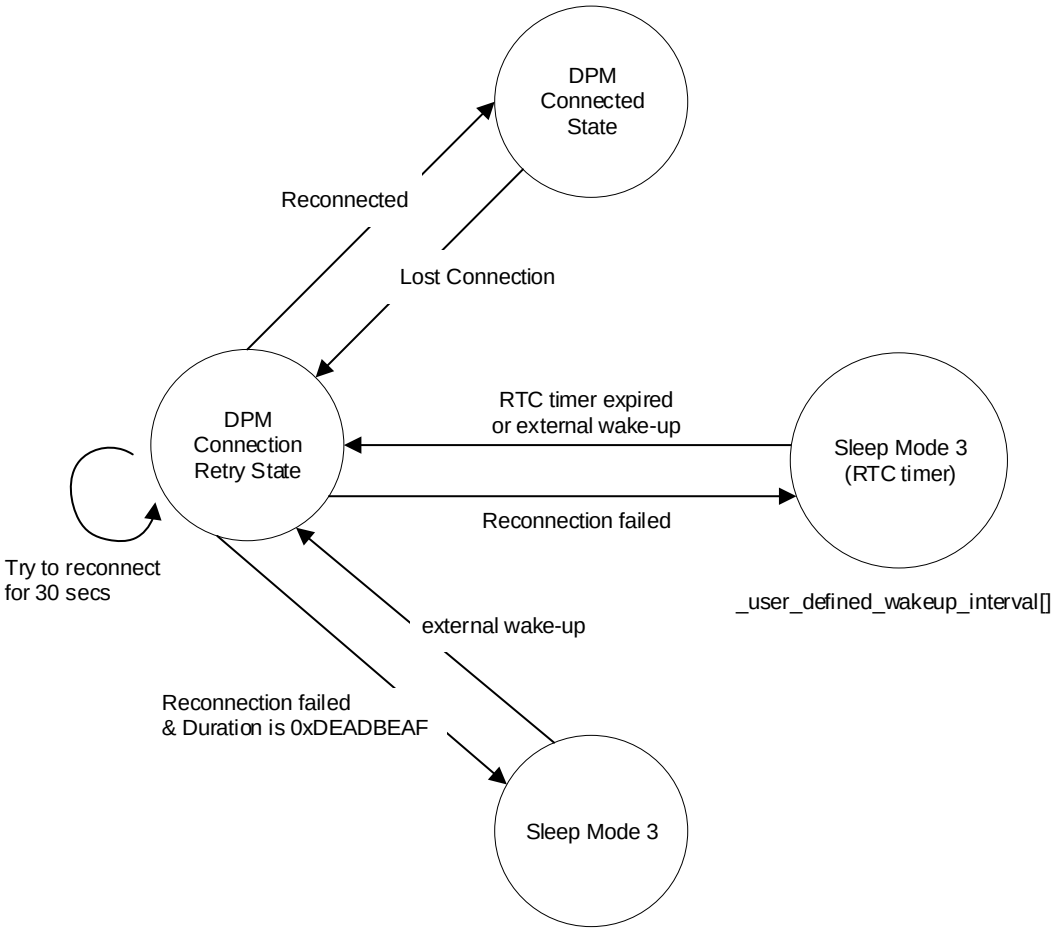

Figure 5: DPM Connection Retry Process

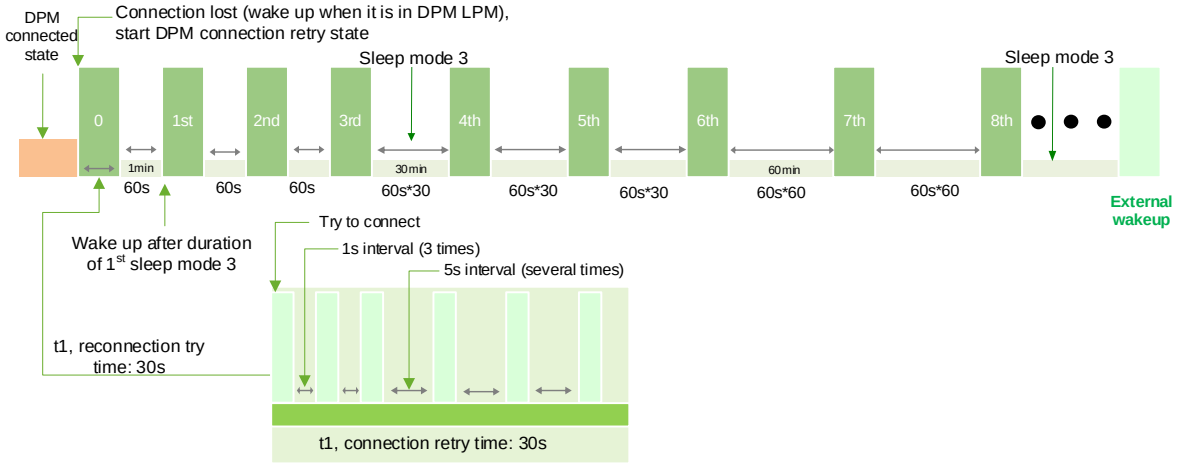

Figure 6: Connection Retry with Default Duration and Interval

The default duration and intervals are defined in a library in SDK as below.

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

Different duration and intervals can be applied by definition and table in SDK. The definition is as follows.

```
apps/da16200/get_started/inc/sys_common_features.h

#define __USER_DPM_ABNORM_WU_INTERVAL__
```

The table for duration and interval are as shown below.

```
// reconnection try time (t1 in the Figure 5)
core/system/src/dpm/da16x_dpm_abnorml_chk.c

#define MAX_INIT_WIFI_CONN_TIME     30 /* Seconds, t1 */

apps/da16200/get_started/src/user_system_feature.c

#ifdef  __USER_DPM_ABNORM_WU_INTERVAL__
/*
 * Format of dpm abnormal wake up 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 Wake up
       60,           // 2nd Wake up       : 0xdeadbeaf is no wake up
       60,           // 3rd Wake up       : 0xdeadbeaf is no wake up
       60 * 30,      // 4th Wake up       : 0xdeadbeaf is no wake up
       60 * 30,      // 5th Wake up       : 0xdeadbeaf is no wake up
       60 * 30,      // 6th Wake up       : 0xdeadbeaf is no wake up
       60 * 60,      // 7th Wake up       : 0xdeadbeaf is no wake up
       60 * 60,      // 8th Wake up       : 0xdeadbeaf is no wake up
       0xDEADBEAF    // 9th Wake up : 0xdeadbeaf is no wake up
};

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 step number of parameters in the table is increased whenever device wakes up from sleep mode 3. If the value of the parameter is `0xdeafbeaf`, the device stops the connection retry, and enters and stays in sleep mode 3 until external GPIO event occurs.

Below console log shows reason connection retry state occurs.

```
…
!!! No selected network !!!
>> Abnormal DPM(1) operation after 1 second.
…
```

The logs and the reasons are:

- Abnormal DPM(1): AP disconnected
- Abnormal DPM(2): DHCP renew failed
- Abnormal DPM(3): ARP response failed
- Abnormal DPM(4/5/6): For debug purpose

3.5 AT Commands with Host Interface in DPM Mode

This section describes how to use AT commands with host interface in the DPM mode.

3.5.1 Enable DPM in Host Interface

Figure 7 shows how to enable DPM using AT commands. Before enabling DPM, AP configuration must be completed in advance. Then, AP connection is done automatically after reboot. GPIO signal to the device can wake up the device from DPM LPM, sleep mode 2 or sleep mode 3. GPIO signal to MCU is also needed for waking up MCU from sleep state. If there is no further AT command communication between the device and MCU, the MCU has to send the device `AT+SETDPMSLPEXT` command for DPM service making the device enter DPM LPM.

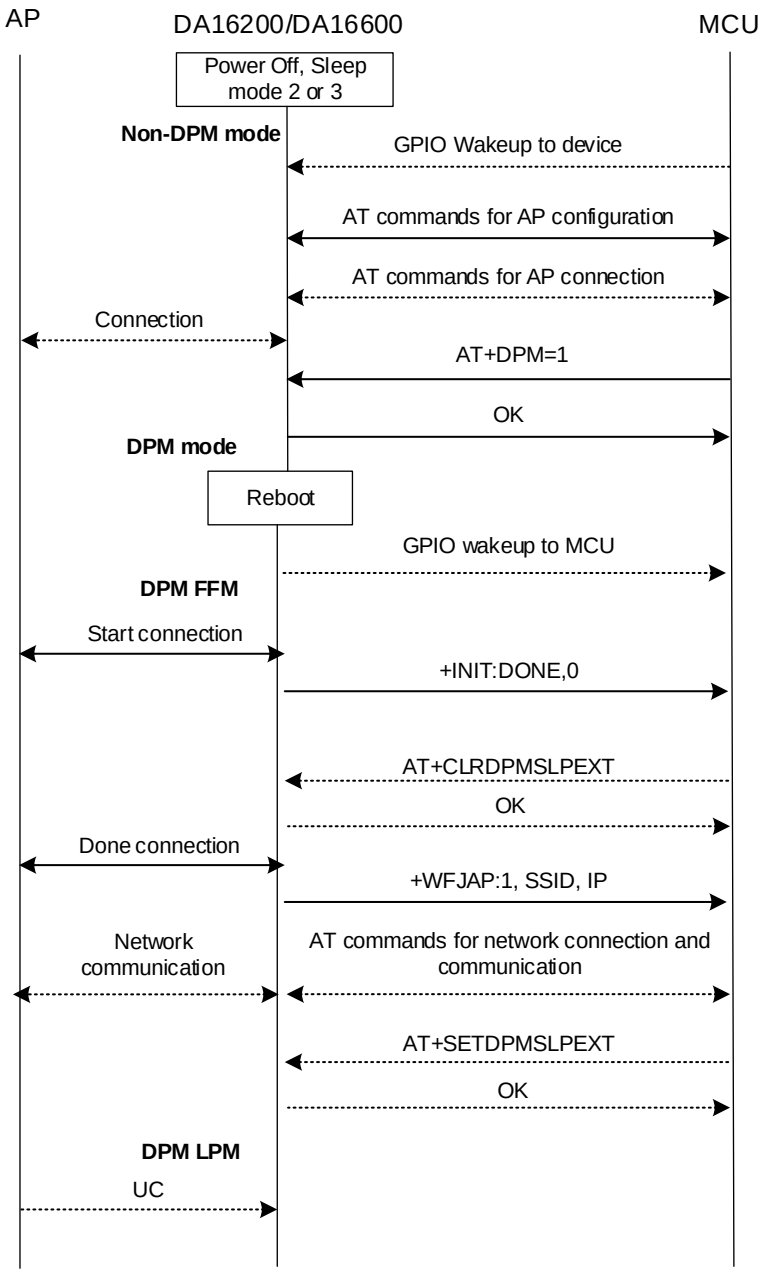

Figure 7: Flow of Enabling DPM Mode in Host Interface

3.5.2 Wake Up from DPM Low Power Mode

The device can be awake by following events in DPM LPM.

- GPIO from MCU
- UC from AP
- RTC timer event
- Disconnection from AP

Figure 8 shows the flow of AT commands when MCU wakes up the device using GPIO.

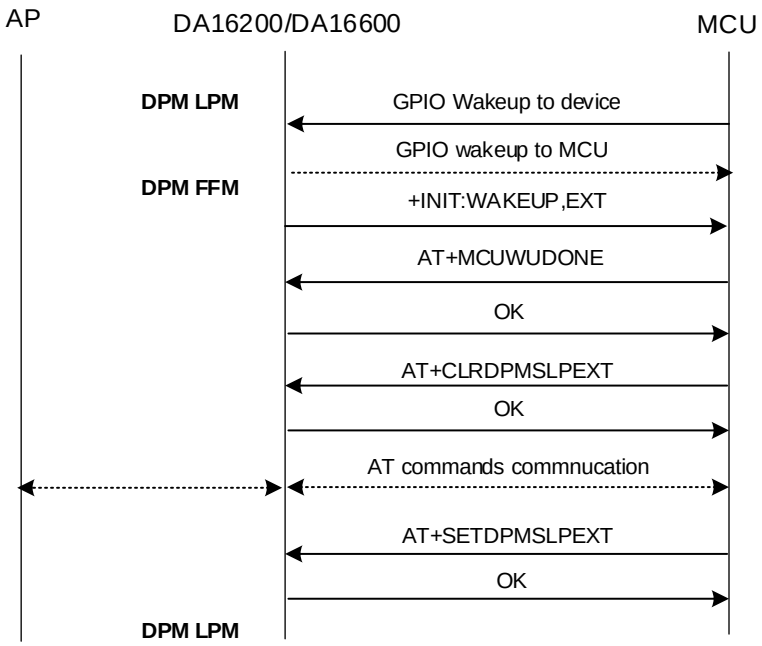

Figure 8: Flow of AT Commands for Waking Up by GPIO of MCU

Figure 9 shows the flow of AT commands when the device receives UC from AP in DPM LPM. When a device receives the UC, the device wakes up immediately, and MCU needs to do responsive process according to the UC.

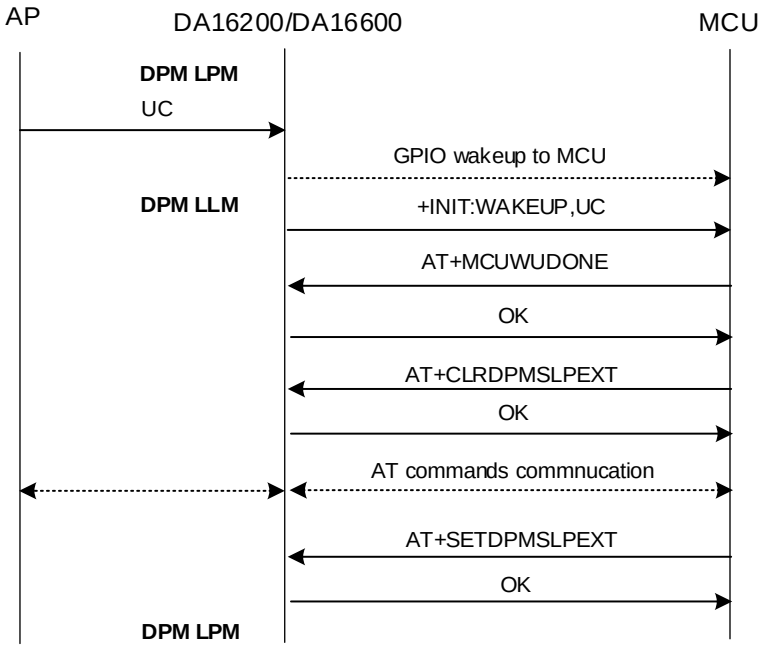

Figure 9: Flow of AT Commands for Waking Up by UC from AP

Figure 10 shows the flow of AT commands when the device wakes up by RTC timer.

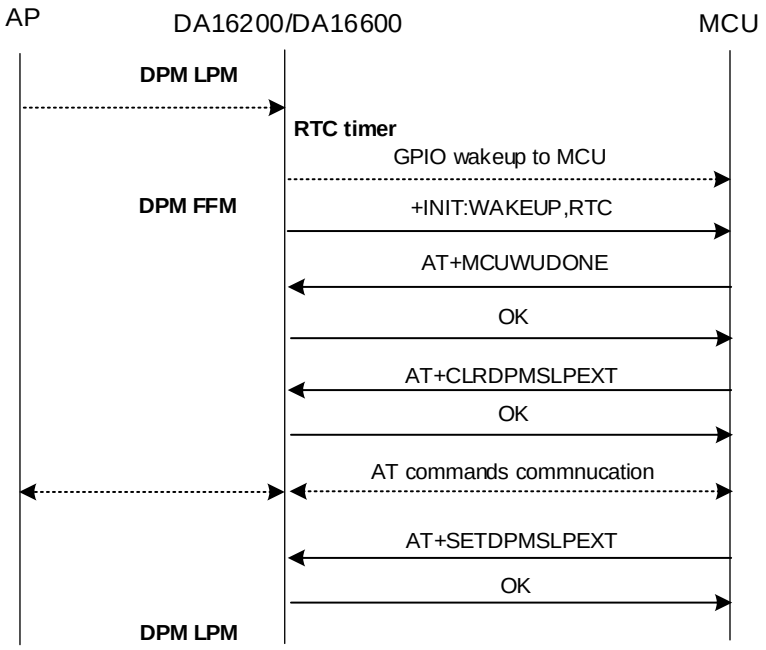

Figure 10: Flow of AT Commands for Waking Up by RTC Timer

Figure 11 shows how the device wakes up using AT commands when it is disconnected with AP. The device tries to connect AP with the same AP information automatically in DPM reconnection state and enters into DPM LPM when it reconnects. In this case, `AT+CLRDPMSLPEXT` and `AT+SETDPMSLPEXT` are not necessarily required because DPM service tries to reconnect by itself. But if MCU needs to communicate with the device, both AT commands are required.

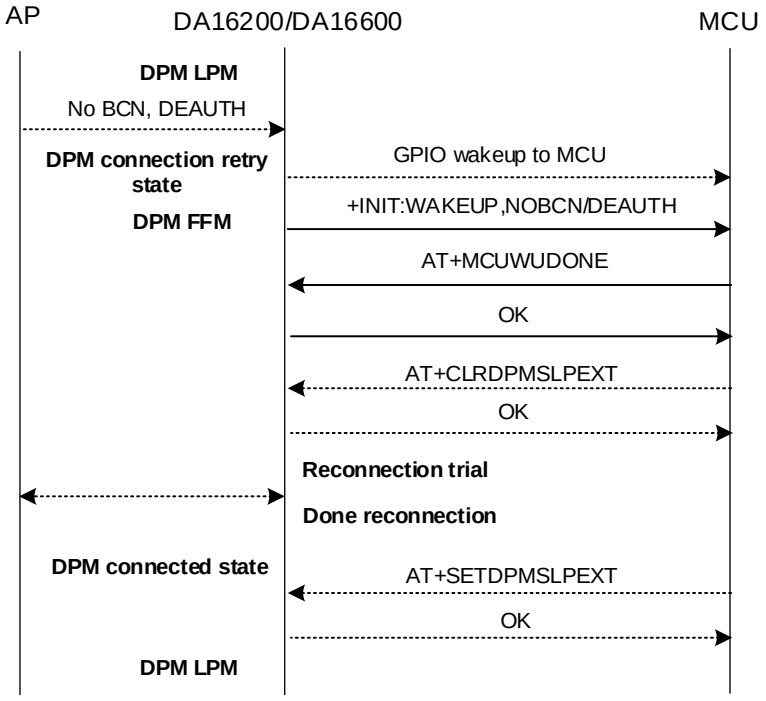

Figure 11: Flow of AT Commands for Waking Up by Disconnection from AP

4 DPM Manager

4.1 DPM Manager Introduction

DPM manager is consisted of high-level APIs for applications with DPM mode being developed easily in case only simple TCP or UDP sessions are used for network. Figure 12 shows the role of DPM manager. And main features supported by DPM manager are as follows:

- TCP/UDP Session Management
  - DPM manager manages TCP/UDP socket registered by user application. It provides callback function to user application to handle events like connection/data reception.
- RTC Timer
  - DPM manager provides 4 x timer and callback to User application. User application can use it according to use case.
- User Data Area in Retention Memory (RTM)
  - DPM manager provides maximum 8 kB user data area in RTM to store user application data. User data in RAM which is not stored in the RTM will be disappeared during DPM LPM because the power of RAM except of RTM is turned off during sleep mode 3 of DPM LPM.

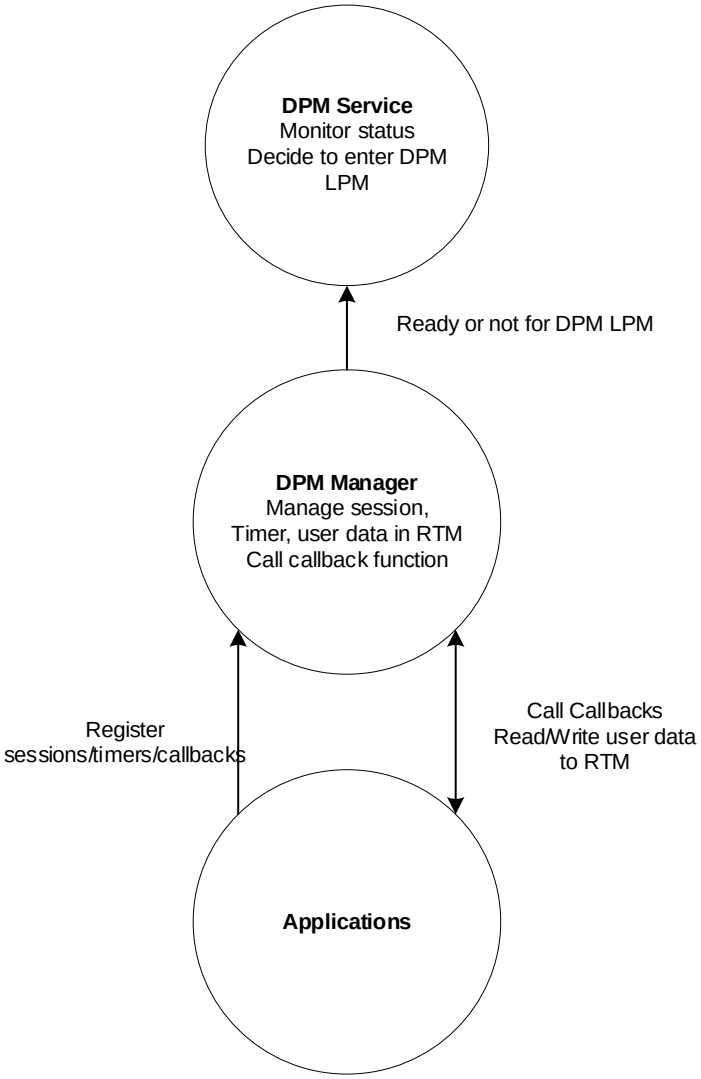

Figure 12: Role of DPM Manager

4.2 Getting Started with DPM Manager

There is an example application using DPM manager in `<SDK ROOT>\apps\common\examples\DPM\All_Used_DPM_Manager`.

- Define configurations and functions

`init_DPM_sample_config` includes all configurations and callback functions which are used and called by DPM manager as Table 7. All things must be defined in `init_DPM_sample_config` as shown in Table 8 according to use cases.

Table 7: Definition of Configurations and Callback Functions

Defined Function Name	Description
BOOT_INIT_FUNC	Function to be called in Wake-up from POR
WAKEUP_INIT_FUNC	Function to be called in Wake-up from external wake-up
TIMER1/2/3/4_TYPE	Timer 1/2/3/4
TIMER1/2/3/4_INTERVAL	Interval of Timer 1/2/3/4
TIMER1/2/3/4_FUNC	Callback function of Timer 1/2/3/4
REGIST_SESSION_TYPE1/2/3/4	Session1/2/3/4 type
REGIST_MY_PORT_1/2/3/4	Port number of session 1/2/3/4
REGIST_SERVER_IP_1/2/3/4	Server IP address for session 1/2/3/4
SESSION1/2/3/4_KA_INTERVAL	Keep alive interval of session 1/2/3/4
SESSION1/2/3/4_CONN_FUNC	Callback function when connected to server
SESSION1/2/3/4_RECV_FUNC	Callback function when received data from server
SESSION1/2/3/4_CONNECT_RETRY_COUNT	Retry count for connection
SESSION1/2/3/4_CONNECT_WAIT_TIME	TCP connection timeout (sec). ● Only for TCP client ● Default is 1 second
SESSION1/2/3/4_AUTO_RECONNECT	Enable/disable Auto reconnection (Only for TCP client)
SESSION1/2/3/4_SECURE_SETUP	TLS enable/disable
SESSION1/2/3/4/_SECURE_SETUP_FUNC	TLS setup function
NON_VOLITALE_MEM_ADDR	Address of user defined data
NON_VOLITALE_MEM_SIZE	Size of user defined data
EXTERN_WU_FUNCTION	Callback when wake-up by external pin ● This callback is disabled by default
ERROR_FUNCTION	Error message callback. ● It's called when error case occurs in the DPM manager

Table 8: Example Function

```
void init_DPM_sample_config (dpm_user_config_t *dpmUserConf)
{
    dpmUserConf->bootInitCallback = BOOT_INIT_FUNC;
    dpmUserConf->wakeupInitCallback = WAKEUP_INIT_FUNC;

    dpmUserConf->timerConfig[0].timerType = TIMER1_TYPE;
    if (sampleParams.tcpClientSendPeriod) {
```

```
        dpmUserConf->timerConfig[0].timerInterval =
sampleParams.tcpClientSendPeriod;
    } else {
        dpmUserConf->timerConfig[0].timerInterval = TIMER1_INTERVAL;
    }
    dpmUserConf->timerConfig[0].timerCallback = TIMER1_FUNC;

    dpmUserConf->timerConfig[1].timerType = TIMER2_TYPE;
    if (sampleParams.udpClientSendPeriod) {
        dpmUserConf->timerConfig[1].timerInterval =
sampleParams.udpClientSendPeriod;
    } else {
        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;

#if defined ( TCP_CLIENT_TEST )
    dpmUserConf->sessionConfig[0].sessionType = REGIST_SESSION_TYPE1;
    dpmUserConf->sessionConfig[0].sessionMyPort = REGIST_MY_PORT_1;
    memcpy(dpmUserConf->sessionConfig[0].sessionServerIp, REGIST_SERVER_IP_1,
sizeof(REGIST_SERVER_IP_1));
    dpmUserConf->sessionConfig[0].sessionServerPort = REGIST_SERVER_PORT_1;
    dpmUserConf->sessionConfig[0].sessionKaInterval = SESSION1_KA_INTERVAL;
    dpmUserConf->sessionConfig[0].sessionConnectCallback = SESSION1_CONN_FUNC;
    dpmUserConf->sessionConfig[0].sessionRecvCallback = SESSION1_RECV_FUNC;
    dpmUserConf->sessionConfig[0].sessionConnRetryCnt =
SESSION1_CONNECT_RETRY_COUNT;      // Only TCP Client
    dpmUserConf->sessionConfig[0].sessionConnWaitTime = SESSION1_CONNECT_WAIT_TIME;
     // Only TCP Client
    dpmUserConf->sessionConfig[0].sessionAutoReconn = SESSION1_AUTO_RECONNECT;
       // Only TCP Client
    dpmUserConf->sessionConfig[0].supportSecure = SESSION1_SECURE_SETUP;
    dpmUserConf->sessionConfig[0].sessionSetupSecureCallback =
SESSION1_SECURE_SETUP_FUNC;
#endif // TCP_CLIENT_TEST

#if defined ( UDP_CLIENT_TEST )
    dpmUserConf->sessionConfig[1].sessionType = REGIST_SESSION_TYPE2;
    dpmUserConf->sessionConfig[1].sessionMyPort = REGIST_MY_PORT_2;
    memcpy(dpmUserConf->sessionConfig[1].sessionServerIp, REGIST_SERVER_IP_2,
sizeof(REGIST_SERVER_IP_2));
    dpmUserConf->sessionConfig[1].sessionServerPort = REGIST_SERVER_PORT_2;
    dpmUserConf->sessionConfig[1].sessionKaInterval = SESSION2_KA_INTERVAL;
    dpmUserConf->sessionConfig[1].sessionConnectCallback  = SESSION2_CONN_FUNC;
    dpmUserConf->sessionConfig[1].sessionRecvCallback = SESSION2_RECV_FUNC;
    dpmUserConf->sessionConfig[1].supportSecure = SESSION2_SECURE_SETUP;
    dpmUserConf->sessionConfig[1].sessionSetupSecureCallback =
SESSION2_SECURE_SETUP_FUNC;
#endif // UDP_CLIENT_TEST


#if !defined ( __LIGHT_DPM_MANAGER__ )
```

```
#if defined ( TCP_SERVER_TEST )
    dpmUserConf->sessionConfig[2].sessionType = REGIST_SESSION_TYPE3;
    dpmUserConf->sessionConfig[2].sessionMyPort = REGIST_MY_PORT_3;
    memcpy(dpmUserConf->sessionConfig[2].sessionServerIp, REGIST_SERVER_IP_3,
sizeof(REGIST_SERVER_IP_3));
    dpmUserConf->sessionConfig[2].sessionServerPort = REGIST_SERVER_PORT_3;
    dpmUserConf->sessionConfig[2].sessionKaInterval = SESSION3_KA_INTERVAL;
    dpmUserConf->sessionConfig[2].sessionConnectCallback  = SESSION3_CONN_FUNC;
    dpmUserConf->sessionConfig[2].sessionRecvCallback = SESSION3_RECV_FUNC;
    dpmUserConf->sessionConfig[2].supportSecure = SESSION3_SECURE_SETUP;
    dpmUserConf->sessionConfig[2].sessionSetupSecureCallback =
SESSION3_SECURE_SETUP_FUNC;
#endif // TCP_SERVER_TEST

#if defined ( UDP_SERVER_TEST )
    dpmUserConf->sessionConfig[3].sessionType = REGIST_SESSION_TYPE4;
    dpmUserConf->sessionConfig[3].sessionMyPort = REGIST_MY_PORT_4;
    memcpy(dpmUserConf->sessionConfig[3].sessionServerIp, REGIST_SERVER_IP_4,
sizeof(REGIST_SERVER_IP_4));
    dpmUserConf->sessionConfig[3].sessionServerPort = REGIST_SERVER_PORT_4;
    dpmUserConf->sessionConfig[3].sessionKaInterval = SESSION4_KA_INTERVAL;
    dpmUserConf->sessionConfig[3].sessionConnectCallback  = SESSION4_CONN_FUNC;
    dpmUserConf->sessionConfig[3].sessionRecvCallback = SESSION4_RECV_FUNC;
    dpmUserConf->sessionConfig[3].supportSecure = SESSION4_SECURE_SETUP;
    dpmUserConf->sessionConfig[3].sessionSetupSecureCallback =
SESSION4_SECURE_SETUP_FUNC;
#endif // UDP_SERVER_TEST
#endif // !__LIGHT_DPM_MANAGER__

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

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

- Register `init_DPM_sample_config` using `dpm_mng_regist_config_cb`
- Start DPM manager using `dpm_mng_start`
- DPM manager calls all registered callback functions. User applications has to call `dpm_mng_job_done()` after task in user callback function is done because DPM manager calls `dpm_mng_job_start()` before the registered callback function is called
- DPM manager sends ready signal to DPM service for DPM LPM after the registered callback functions are called and related tasks are done
- The `dpm_mng_regist_config_cb` and `dpm_mng_start` functions are called whenever DPM wake up

4.2.1 APIs in DPM Manager

Table 9 shows user APIs which can be used in application.

Table 9: User APIs

API	Description
`int dpm_mng_regist_config_cb(` `  void (*regConfigFunction)())`	Register configurations and callback functions in DPM manager

API	Description
int dpm_mng_send_to_session(UINT sessionNo, ULONG ip, ULONG port, char *buf, UINT size)	Send data to the session
int dpm_mng_set_session_info_my_port_no(UINT sessionNo, ULONG port)	Change own port number of the session (only for server)
int dpm_mng_set_session_info_peer_port_no(UINT sessionNo, ULONG port)	Change peer's port number of the session (only for Server)
int dpm_mng_set_session_info_peer_ip_addr(UINT sessionNo, char *ip)	Change peer's IP address of the session (only for Server)
int dpm_mng_set_session_info_server_ip_addr(UINT sessionNo, char *ip)	Change the server's IP address of the session (only for Client)
int dpm_mng_set_session_info_server_port_no(UINT sessionNo, ULONG port)	Change the server's port number of the session (only for Client)
int dpm_mng_set_session_info_local_port(UINT sessionNo, ULONG port)	Change own port number of the session (only for Client)
int dpm_mng_set_session_info(UINT sessionNo, ULONG type, ULONG myPort, char *peerIp, ULONG peerPort, ULONG kaInterval, void (*connCb)(), void (*recvCb)())	Set all the configs of the session Type 1: TCP Server 2: TCP Client 3: UDP Server 4: UDP Client kaInterval: in seconds
int dpm_mng_set_DPM_timer(UINT timerId, UINT timerType, UINT interval, void (*timerCallback)())	Change timers timerId: 1~4 timerType: 1 (periodic), 2 (one-shot) interval: timer interval in seconds timerCallback of each timer
int dpm_mng_unset_DPM_timer(UINT timerId)	Unregister timer timerId: 1~4
int dpm_mng_start_session(UINT sessionNo)	Start the session
int dpm_mng_stop_session(UINT sessionNo)	Stop the session
int dpm_mng_set_session_info_window_size(UINT sessionNo, UINT windowSize)	Change window size of the session (only for a TCP session. Session restart (stop/start) is needed to take effect
int dpm_mng_set_session_info_conn_retry_count(UINT sessionNo, UINT connRetryCount)	Change connection retry count of the session (only for TCP Client session)
int dpm_mng_set_Session_info_conn_wait_time(UINT sessionNo, UINT connWaitTime)	Change connection wait time for the session (only for TCP Client session) connWaitTime: in seconds

API	Description
int dpm_mng_set_Session_info_auto_reconnect(UINT sessionNo, UINT autoReconnect)	Set auto reconnection after disconnection. (Only for TCP Client) autoReconnection 1: try for reconnection 0: no retry for reconnection
int dpm_mng_save_to_RTM()	Write user data to RTM
int dpm_mng_init_done()	Return whether the initialization process of DPM manager is complete. return value 1: Done 0: Incomplete
int dpm_mng_job_done()	Notify callback function is done
int dpm_mng_job_start()	This function is invoked by the DPM manager before a callback is called, so the application does not need to call this function inside a callback

5 DDPS

5.1 DDPS Introduction

AP with Wi-Fi Basic Server Set (BSS) provides a method to configure the power saving options of each connected station device. To support a station's power saving feature, the AP must maintain the packets for that station when the station is in a power saving state. The DPM Dynamic Period Setting (DDPS) algorithm checks the required buffering time of the AP and decides optimal DTIM interval.

5.2 Enable DDPS

DDPS can be enabled using `setup` console command during the DPM configuration as below:

```
Dialog DPM (Dynamic Power Management) ? [Yes/No/Quit] : y

DPM factors : Defaults ? [Yes/No/Quit] : n

DDPS Enable : Default ? [No/Yes/Quit] : y

DPM Keep Alive Time(0~600000 ms) ? [Quit] (Default 30000 ms) :

DPM User Wakeup Time(0~86400000 ms) ? [Quit] (Default 0 ms) :

DPM TIM Wakeup Count(1~30 dtim) ? [Quit] (Default 10) :
========================================
DPM MODE           : Enable
Dynamic Period Set : Enable
Keep Alive Time    : 30000 ms
User Wakeup Time   : 0 ms
TIM Wakeup Count   : 10 dtim
========================================
DPM CONFIGURATION CONFIRM ? [Yes/No/Quit] : y
```

For more information on the `setup` console command, see the getting started guide, Ref. [1]. DDPS can be also enabled or disabled by below DPM API.

Table 10: DPM API

unsigned char setup_apply_dpm(unsigned char dpm_mode, unsigned char dpm_Dynamic_Period_Set, int dpm_KeepAlive_time, int dpm_User_Wakeup_time, int dpm_TIM_wakeup_count)		
Parameter	dpm_mode	Enable/Disable DPM Mode
	dpm_Dynamic_Period_Set	Enable/Disable Dynamic Period Set
	dpm_KeepAlive_time	Keep Alive time (0~600000 ms) default: 30000 ms Time to wake up periodically to sync with the AP
	dpm_User_Wakeup_time	User Wake up Time (0~86400 sec) default: 0 sec This is used when the user needs to wake up periodically.
	dpm_TIM_wakeup_count	TIM Wake up Count (1~65535 dtim) default : 10 dtim This is the interval to check the AP's beacon. It is recommended to use 30 when using DDPS.
Return		E_ERROR(254): Error
		Others (E_CONTINUE): Success

5.3 AP Test Report for DDPS

Table 11 shows the DDPS test results with APs. DDPS function probes the buffering time of AP and sets almost the same value to the probed minimum or the lowest value 1 as the interval between PTIM active state of Figure 2.

Table 11: DDPS Test Result

AP Model	Probed Buffering Time of AP	
	Max	Min
360 F5C	5	5
360 F5S	5	4
360 P1	5	4
360 P4	5	5
360 V5S	5	5
AMPED ALLY-0091K	5	5
ANTIBANG A3	5	4
ASUS ACRH13	5	5
ASUS RT-AC1200GU	0	0
ASUS RT-AC1750	5	5
ASUS RT-AC3200	5	5
ASUS RT-AC51UPLUS	0	0
ASUS RT-AC5300	5	5
ASUS RT-AC58U	5	5
ASUS RT-AC66U	5	5
ASUS RT-AC87U	5	5
ASUS RT-AC88U	5	5
ASUS RT-N14UHP	5	5
ASUS TM-AC1900	5	5
BELKIN F7D6301	5	4
BELKIN F9K1002	5	3
BUFFALO WHR-300HP2D	4	2
BUFFALO WSR-1166DHP3	5	5
BUFFALO WSR-2533DHPL	5	4
CISCO RV110W-ECN	5	5
DLINK 605L	5	5
DLINK 616	5	5
DLINK 619L	5	5
DLINK 822	5	5
DLINK DIR-806A	5	5
DLINK DIR-820L	5	4
DLINK DIR-822P	5	5
DLINK DIR-823PRO	5	4

AP Model	Probed Buffering Time of AP	
	Max	Min
DLINK DIR-828	5	5
DLINK DIR-842	5	5
DLINK DIR850LW	5	4
DLINK DIR-880L	5	5
DLINK DIR-890L	5	5
ELECOM WRC-1167GEBKS	5	5
EZVIZ CS-X3C-8E	5	5
FASTCOM FAC1200R	5	4
FASTCOM FAC2100R	0	0
FASTCOM FW313R	5	4
FASTCOM FW450R	5	5
FASTCOM FWR200	5	4
H3CMAGIC R100	5	5
H3CMAGIC R300	5	5
HIWIFI E30	5	4
HIWIFI HC5861B	5	4
HUAWEI GLORY-ROUTINGPRO	5	5
HUAWEI HONOR-X2	5	5
HUAWEI WS5100	5	5
HUAWEI WS5102	5	5
HUAWEI WS5200	5	5
HUAWEI WS550	5	4
HUAWEI WS832	5	5
HUAWEI WS851	5	5
HUMAX QUANTUM-T3Av2	5	5
HUMAX T10X	5	4
IODATA WNAC583R	0	0
IODATA WNAC733GR	0	0
IODATA WNAX1167	0	0
IODATA WNPR2600G	5	5
IPTIME A1004	0	0
IPTIME A2004NSR	5	4
IPTIME A300NS-BCM	5	5
IPTIME A7004M	5	4
IPTIME A3004NS-BCM	5	5
IPTIME A3004NS-BCM	5	5
IPTIME A8004ITL	5	4
IPTIME A804NS	5	4

AP Model	Probed Buffering Time of AP	
	Max	Min
IPTIME N604	5	4
IPTIME A604R	5	5
IPTIME N702BCM	5	5
IPTIME N704BCM	5	4
IPTIME N804V	5	5
LBLINK BL-AC1200D	5	4
LBLINK WR9000	5	4
LBLINK WR4000	5	4
LINKSYS E1200	5	4
LINKSYS EA6900	5	5
LINKSYS EA7500	5	5
LINKSYS EA8300	5	5
LINKSYS WRT1900AC	5	5
LINKSYS WRT300N	5	5
LINKSYS WRT3200ACM	5	4
LINKSYS WRT54GL	5	5
MERCURY C12G	0	0
MERCURY D196G	5	5
MERCURY D19G	5	4
MERCURY D26GPro	5	5
MERCURY MW300R	5	4
MERCURY MW313R	5	4
MERCURY MW316R	5	5
MIKROTIK RB751U-2H	2	0
MOTOROLA MR1900	5	1
MERCURY RUSH-1537N	5	5
NETCORE 360_P2	5	5
NETGEAR JWNR2000v2	5	5
NETGEAR ORBI	5	5
NETGEAR R6120	5	4
NETGEAR R6220	5	3
NETGEAR R7000	5	4
NETGEAR R8000	5	4
NETGEAR RAX120	5	5
NETGEAR RAX40	5	4
NETGEAR RAX80	5	5
NETGEAR WNDR3400v3	5	1
NETGEAR X10	5	5

AP Model	Probed Buffering Time of AP	
	Max	Min
NETIS M3200N	5	5
NETIS MF1200AC	5	5
NETIS WF2770	0	0
NETIS WF2785	5	5
NETIS WF302	5	4
NEXT 504N	5	5
NEXT 7004N	5	5
NEXT 8004N	5	4
PHICOMM PSG1218	0	0
PIXLINK WR07	5	4
SAMSUNG SWW3100BG	5	3
SAMSUNG SWW-3400RW	5	5
SAMSUNG ET-WV525	5	5
SEMA SAP-H310SR	1	0
SYNOLOGY MR2200AC	5	5
SYNOLOGY RT2600AC	5	5
TENDA AC15	5	5
TENDA FH304	5	5
TENDA N318	5	3
TOTOLINK A2500R	5	5
TOTOLINK A3100R	5	5
TOTOLINK A780R	5	4
TOTOLINK A800R	5	5
TOTOLINK A850R	5	5
TOTOLINK N350RP	5	5
TOTOLINK N600R	5	5
TPLINK AD7200	5	4
TPLINK ARCHER-AX10	5	4
TPLINK ARCHER-C2600	5	5
TPLINK TL-WAR1200L	5	5
TPLINK TL-WDR8610	5	5
TPLINK TL-WDR8690	5	5
TPLINK WDR5600	5	4
TPLINK WDR5660	5	5
TPLINK WDR6500	5	5
TPLINK WDR7660	5	4
TPLINK WR2041	5	5
TPLINK WR842N	5	4

AP Model	Probed Buffering Time of AP	
	Max	Min
TPLINK WR880N	5	5
TPLINK WR940N	5	5
TRENDNET TEW-812DRU	5	1
TRENDNET TEW-827DRU	5	4
UNICORN AW	5	4
UTT A310	0	0
UTT A655W	0	0
UTT A755W	5	0
VOLANS G1	5	4
WAVLINK A33	0	0
WAVLINK N300	5	4
WAVLINK WN521N2A	0	3
WEVO 11AC-NASROUTER	5	4
WEVO HI1200AC	5	0
XIAOMI DVB4218CN	5	5
XIAOMI MIWIFI3	1	0
XIAOMI MIWIFIPRO	5	5
XIAOMI R1CM	0	5
XIAOMI R3AC	5	
ZIO 2520N	5	
ZIO 5500AC	5	5
ZIO FREEZIO	5	5

Revision History

Revision	Date	Description
1.6	Apr. 12, 2024	Added DPM wake-up type
1.5	Oct. 06, 2023	● Changed DPM Timer API argument from seconds to milliseconds ● Updated descriptions about DPM with additional figures ● Added details for abnormal working flow
1.4	Jan. 12, 2023	Merged documents listed below and added DPM API descriptions ● UM-WI-005_DA16200_DA16600_DPM_Manager ● UM-WI-034_DA16200_DA16600_DPM_Over_AT-CMD
1.3	Sep. 27, 2022	Updated DPM API
1.2	Mar. 28, 2022	Updated logo, disclaimer, and copyright
1.1	Nov. 25, 2021	Changed the title
1.0	Oct. 29, 2020	Initial release

Status Definitions

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

RoHS Compliance

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

Important Notice and Disclaimer

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