A company of SIM Tech

SIM800_Series_Embedded AT_Sleep Note_V1.00

Document Title	SIM800 Series_ Embedded AT_ Sleep Note
Version	1.00
Date	2013-11-10
Status	Filing
Document Control ID	SIM800 Series_ Embedded AT_ Sleep Note _V1.00

General Notes

SIMCom offers this information as a service to its customers, to support application and engineering efforts that use the products designed by SIMCom. The information provided is based upon requirements specifically provided to SIMCom by the customers. SIMCom has not undertaken any independent search for additional relevant information, including any information that may be in the customer's possession. Furthermore, system validation of this product designed by SIMCom within a larger electronic system remains the responsibility of the customer or the customer's system integrator. All specifications supplied herein are subject to change.

Copyright

Contents

Version History

Date	Version	What is new	Author
2012-10-10	1.00	New version	Mao bin
2013-11-10	1.01	Updated scope of application	Mao bin

Scope

This document can apply to Embedded AT modules of SIM800 series, including SIM800W64, SIM840W64, SIM800W128, SIM840W128, SIM800V128, SIM800H, SIM800.

This document presents the operations and notes of Sleep mode for Embedded AT.

1 Interface

Below is the description of interface in the file named “eat_interface.h”.

```
/*****************************************************************************
* Function :eat_sleep_enable
* Description:
*     enable module to enter sleep mode
* Parameters:
*     param1 eat_bool en[IN]: EAT_TRUE      enable module to enter sleep mode
*                             EAT_FALSE    disable module to enter sleep mode
* Returns:
*    eat_bool en: EAT_TRUE
*****************************************************************************/
extern eat_bool (*const eat_sleep_enable)(eat_bool en);
```

2 Sleep Mode

2.1 Sleep Mode Setting

The parameter eat_sleep_enable is used to set whether the system can enter sleep mode. When the value of eat_sleep_enable is EAT_TRUE, the system is allowed to enter sleep mode. When the value of eat_sleep_enable is EAT_FALSE, getting into sleep mode is forbidden.

Notes:

1. Ensure backlight has been closed before setting sleep mode parameter. If the state of backlight is open, the system can’t enter sleep mode. The parameters used to set backlight state are eat_lcd_light_sw and eat_kpled_sw. When the values of eat_lcd_light_sw and eat_kpled_sw are both KAL_TURE, the backlight state is open and the system can not enter sleep mode. When the values of eat_lcd_light_sw and eat_kpled_sw are both KAL_FALSE, the backlight is closed, then the system can be set to enter sleep mode.
2. After being allowed to enter sleep mode, the system may not start to sleep immediately. Before enter sleep mode, the system must check current network state and executing states of other tasks. Only when the system is free, it can enter sleep mode. If the system is busy, it will wait for all tasks finished before enter sleep mode. For example: if a sleep command is sponsored during a call, system will get into sleep until the call finished.

2.2 Wake Up from Sleep Mode

When the module is in sleep mode, only specific actions can wake up it. The specific actions are displayed in the list. They are coming call, coming SMS, timer time-out, keying and GPIO interrupt. Any other actions can’t make the module leave sleep state.

After being waked up by listing actions, the module is not leave sleep mode absolutely. When the system is free again, the module will back to sleep mode. So if module was waked up and need to leave sleep mode, user must set sleep mode parameter actively to forbid module enter sleep mode again. That is to say, the value of eat_sleep_enable must be set to EAT_FALSE if user want module to leave sleep mode after it was waked up by some reasons.

Wake Up Reason	Report Message	Related Information
Coming Call	EAT_EVENT_MDM_READY_RD	Report ”\r\nRING\r\n”
Coming SMS	EAT_EVENT_MDM_READY_RD	Report ”\r\n+CMTI: xxx\r\n”
Timer Time-out	EAT_EVENT_TIMER	event.data.timer.timer_id
Keying	EAT_EVENT_KEY	event.data.key
GPIO Interrupt	EAT_EVENT_INT	event.data. interrupt

- When module is waked up by Coming Call and Coming SMS: system will report EAT_EVENT_MDM_READY_RD message and the related AT command data. Interface function eat_modem_read(buf, len) will be used to obtain message parameters’ values.
- When module is waked up by timer: system will report EAT_EVENT_TIMER message, and the timer’s ID will be contained in the message’s parameter event.
- When module is waked up by keying: system will report AT_EVENT_KEY message,

and the key's value and state will be contained in the message's parameter event.

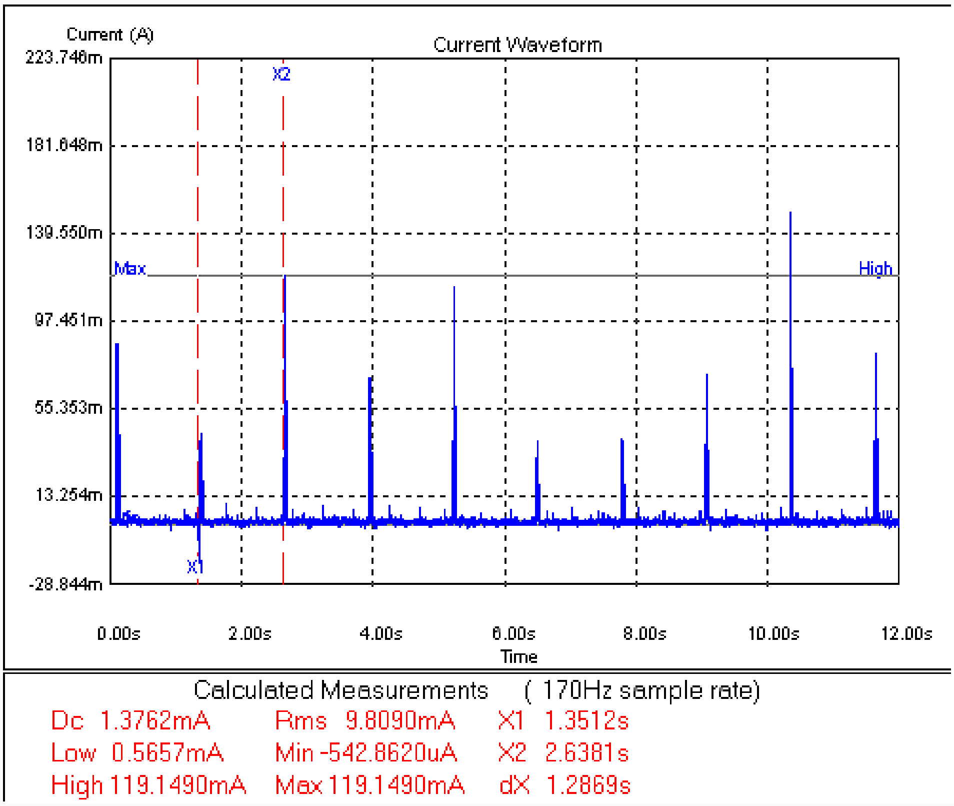
- When module is waked up by GPIO interrupt: system will report EAT_EVENT_INT message, and the pin's value and state will be contained in the message's parameter event.

2.3 Current Consumption in Sleep Mode

Once the module entering the sleep mode, the sleeping current will be less than 1mA. In the experiments, when the module is registering network, the mean consumption of current detected by instruments is about 1.4mA. While in the actual network, the current may be a little higher.

When the module is in sleep mode, it can be waked up by itself periodically in order to communicate with the network. This action is automatic and users don't need to do anything. The automatic wake-up time is about tens of milliseconds which is short-lived. If the automatic wake-up time is out, the module will enter sleep mode again.

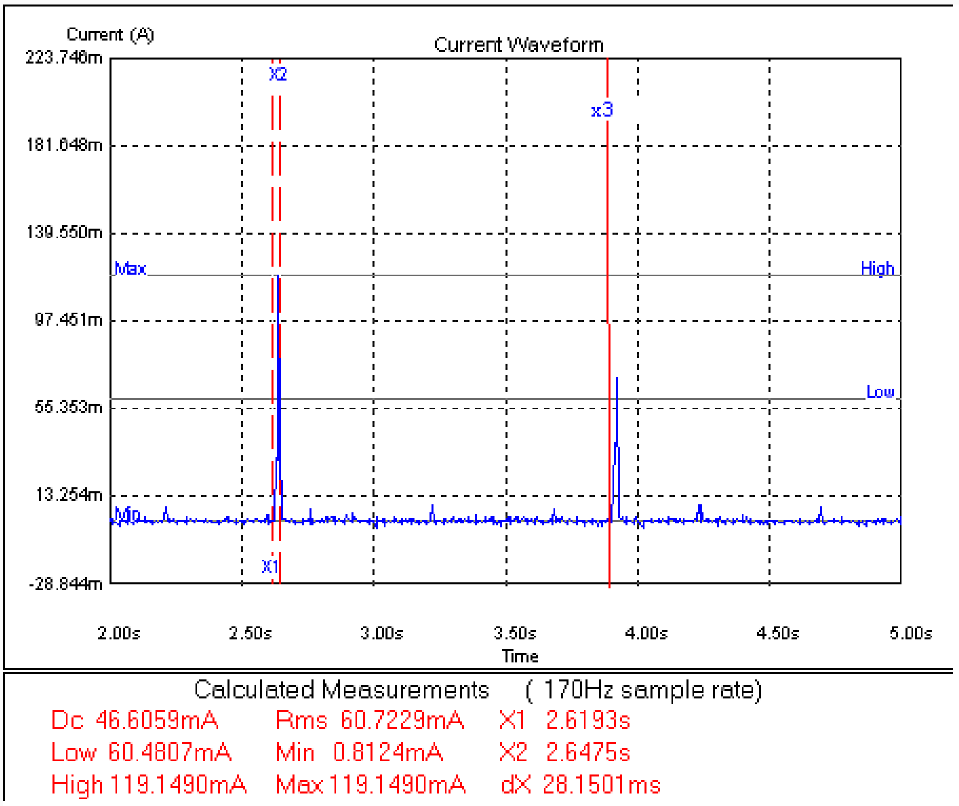
The consumption of current in sleep mode is present in the following figure1.



```
Calculated Measurements     ( 170Hz sample rate)
Dc  1.3762mA        Rms  9.8090mA       X1  1.3512s
Low  0.5657mA       Min -542.8620uA     X2  2.6381s
High 119.1490mA     Max 119.1490mA      dX  1.2869s
```

Figure 1 Consumption current of module in sleep mode

The following figure2 is the partial enlarged view of figure1.



Calculated Measurements (170Hz sample rate)

Dc 46.6059mA Rms 60.7229mA X1 2.6193s

Low 60.4807mA Min 0.8124mA X2 2.6475s

High 119.1490mA Max 119.1490mA dX 28.1501ms

Figure 2 Partial enlarged view of module's consumption current in sleep mode

2.4 Serial Port State in Sleep Mode

In the sleep state, serial port is not working. So it can't wake up module by transmitting data through serial port.

During the period of automatic wake-up time, module can receive data as normal. For example, during the time from x1 to x2 showed in figure2, module can receive data from serial port. As mentioned in chapter 2.3, this period is just tens of milliseconds. If user transmits data through serial port during the short period, data received by module may be incomplete. It is impossible to ensure data integrity if transmitting occurs during automatic wake-up time. In figure2, during the time between x2 and x3, module is in sleep mode and can't communicate through serial port.

If serial port needs to be used when the module is in sleep mode, there are 3 methods available. User can take the related method according to different conditions. For example: if serial port was used to transport AT command, the first method is the right one to take. While if serial port was used by APP, the second and third methods can both work. Below is the detailed information of the three methods.

1) Sending "AT+CSCLK=0\r\n" continuously: If serial port wants to be used to send AT command when the module is in sleep mode, user can send "AT+CSCLK=0\r\n" continuously. When the module responses "OK", it means serial port can be used to transmit AT command as normal.
2) Sending specific data continuously: In sleep mode, if serial port was used by APP, the APP will search specific data in data received from serial port. So, when peripherals need

to send data through serial port, it should firstly send specific data continuously. After the APP receives and recognizes the specific data, it will call interface function eat_sleep_enable to forbid system to enter sleep mode again. In the same time, the APP will return response data to serial port. Then the peripheral can start to send data after receiving response data from APP.

3) Using external interrupt pins: When serial port needs to be used to send data in sleep mode, user can give a high or low level to an interrupt pin of the module. After APP receives the interrupt signal, it will decide whether to forbid system to enter sleep mode again based on the level of the pin. Then the APP will call the interface function eat_sleep_enable() to realize the action of forbidding or allowing system to enter sleep mode again.