# SIM8200 Series_Open Linux _Sleep&Wakeup_Application Note

5G Module

**SIMCom Wireless Solutions Limited**
Building B, SIM Technology Building, No.633, Jinzhong Road
Changning District, Shanghai P.R. China
Tel: 86-21-31575100
support@simcom.com
www.simcom.com

| | |
|---|---|
| **Document Title:** | SIM8200 Series_Open Linux_Sleep&Wakeup_Application Note |
| **Version:** | 1.00 |
| **Date:** | 2020.8.17 |
| **Status:** | Released |

## 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



**SIMCom Wireless Solutions Limited**
Building B, SIM Technology Building, No.633 Jinzhong Road, Changning District, Shanghai P.R. China
Tel: +86 21 31575100
Email: simcom@simcom.com

**For more information, please visit:**
https://www.simcom.com/download/list-863-en.html

**For technical support, or to report documentation errors, please visit:**
https://www.simcom.com/ask/ or email to: support@simcom.com

# Version History

| Version | Date | Owner | What is new |
|---|---|---|---|
| V1.00 | 2020.8.17 | Xiang.sun | First Release |

# Contents

# 1 Introduction

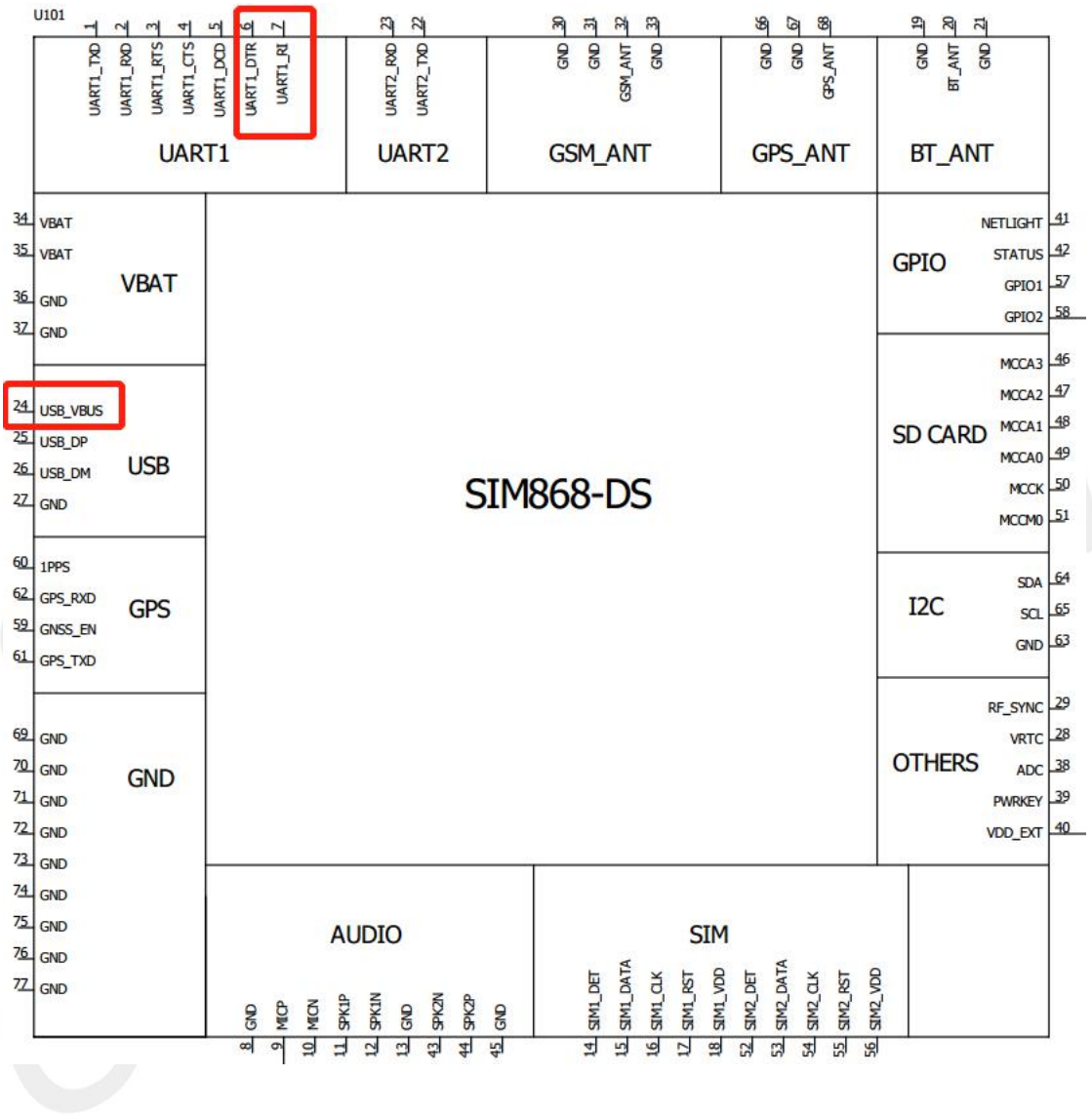
## 1.1 Purpose of the document

This document describes the SIM8200 module sleep wake mechanism and operation process, applied to the module secondary development of open Linux system, the documentation mainly describes the module Linux system sleep and wake.

## 1.2 Related documents

[1] SIM8200 Series_AT Command Manual

# 2 Hardware Interface




## 1.3 Pin Explain

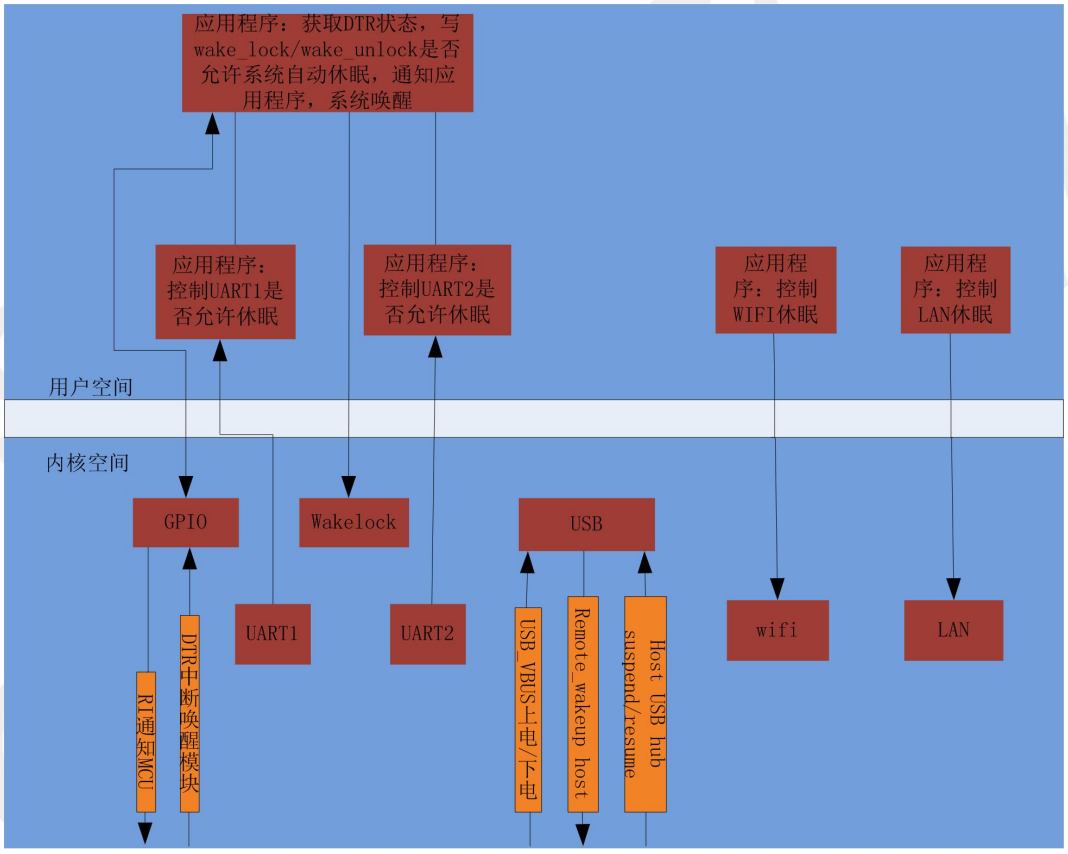
| Pin_name | Pin_num | In/Out | Explain |
|---|---|---|---|
| UART1_DTR | 6 | In | DTR is mainly used for module development application twice to manage system sleep/wake. The application program can detect the interruption of DTR in real time and determine the system sleep/wake state according to the level of GPIO. |
| UART1_RI | 7 | Out | RI is mainly used for the module to develop the application twice |

| | | | |
|---|---|---|---|
| | | | to wake up the HOST. The application can pull the RI up/down in the way of GPIO operation. |
| USB_VBUS | 24 | In | USB_VBUS power on and power off will affect the USB sleep/wake, USB sleep/wake directly related to the system sleep/wake. |

# 3 Module Sleep

## 3.1 Flow Chart of Sleep

The following is the system sleep flow chart:




## 3.2 Introduction to Linux sleep process

Devices are used differently in Linux systems due to different hardware versions, so Linux system sleep depends on device usage and application use of the wakelock. The following conditions:

1. USB dormancy requires the HOST module USB or VBUS to power off.
2. To enter sleep, the system must turn off WIFI. WIFI can be turned off by AT command

AT+CWMAP=0 or by calling the API interface of turning off WIFI.

3. If there is a LAN port, call the driver unloading API of the LAN to let the LAN port sleep

4. UART hibernation requires application management. If UART does not hibernate, the system will not enter hibernation mode, AT+CSCLK=1.

5. All applications release the wakelock.

So Linux systems go to suspend only when all devices are in the suspend state and no applications hold the wakelock lock.

## 3.3 The use of wakelock

Wakelock locks are used when an application needs the system to remain out of suspend for as long as it wants to perform a particular task.

When the application wants the system not to sleep, open /sys/power/wake_lock and write a specific string to the file (string naming rule: [process name]_[reason for not sleeping]).

When the application completes the task, it needs to release the corresponding wakelock. Open the /sys/power/wake_unlock file and write the corresponding string to the file.

When all applications have released the wakelock, the system goes to suspend.

## 3.4 Module Sleep Methods

1. Close the WIFI
2. The application releases all wakelock
3. HOST raises DTR, HOST allows the module to sleep
4. The application stops sending data to UART and sets the following adb command to allow UART to sleep and the serial port to sleep:

```
echo auto > /sys/class/tty/ttyHS0/device/power/control
echo auto > /sys/class/tty/ttyHS1/device/power/control
echo auto > /sys/class/tty/ttyHS2/device/power/control
```

5. Suspend USB or disconnect VBUS from HOST.

## 3.5 Hibernate DEBUG Methods

1. Check whether WIFI is off, AT+CWMAP?Zero is close.
2. Check the DTR level status.
3. Check wake_lock usage, adb shell cat/sys/power/wakelock.
4. See if UART sleep is allowed and auto sleep is allowed.
    1) adb shell cat /sys/class/tty/ttyHS0/device/power/control
    2) adb shell cat /sys/class/tty/ttyHS1/device/power/control
    3) adb shell cat /sys/class/tty/ttyHS2/device/power/control
5. Check the device sleep state.
    1) adb shell cat /sys/kernel/debug/wakeup_sources > wake.log
    2) adb shell cat /sys/kernel/debug/wakeup_sources > wake1.log

Comparing wake. Log and wake1.log files, the difference is that the device cannot sleep.

# 4 Wake up the module

## 4.1 Self-awakening Of The System

The system self - awakening, generally has the following several cases
1) incoming calls, text messages
2) the network server sends data to the module
3) the RTC timer After all the self-awakening event tasks are completed, the system automatically goes to sleep.

## 4.2 The HOST Wakeup Module

Host wake-up module, mainly via USB, GPIO:
1) the host wakes up the system via USB in two cases:
   a) when the module USB is suspended, the host wakes up the USB bus and the module USB
   b) when the module USB is in the state of Vbus power failure, the host will electrify the Vbus and enumerate the USB device to wake up the system.
2) DTR can support wake-up interrupts.

For example: to configure the DTR drop edge wake-up, take the following parameters:
   a) Echo falling > /sys/class/gpio/gpio74/edge
   b) Echo 1 > /sys/class/gpio/gpio74/can_wakeup

As long as one of the above conditions occurs, it can wake up the Linux system.

## 4.3 The Module Wakeup HOST

There are generally two methods of module wake host RI and USB, GPIO through the pulse wake host, USB through REMOTE_WAKEUP wake host.

1) RI wakes up the HOST method
   The HOST on the hardware can wake up the RI foot of the GPIO connection module. The HOST interrupt detects the RI. When the module has a message to notify HSOT, the module application program changes the state of the RI to notify the HOST.


2) USB remote wake host principle

When the host into the sleep, the host of USB bus in the suspend state, if the module has URC reported to host, request get communication module in a pending state, resume signal, the USB host controller by USB host controller into resume and keep 20 ms, the 20 ms will receive resume signal to ensure all the devices on the bus, after 20 ms began to recover the USB bus, USB bus into idle state and must be within the 3 ms start sending SOF package, or USB module will hang up again.

Description: the USB controller on the host side needs to support Remote Wakeup. This feature enables the USB controller to accept the resume signal from the module. When the USB controller is awakened remotely, the host will also be awakened.

The module also needs to support the Remote Wakeup feature, which is set in the bm Attributes field of the device descriptor. The host side enables the Remote Wakeup feature of the module through SET_FEATURE, and can also close the Remote Wakeup feature of the module through the CLEAR_FEATURE. When the host goes to sleep, you need the Remote Wakeup feature of the enable module.