=

A76xx_Series_APIs_ Programming_User_Guide

LTE Module

SIMCom Wireless Solutions Limited
SIMCom Headquarters Building, Building 3, No. 289 Linhong Road, Changning District, Shanghai P.R. China
Tel: 86-21-31575100
support@simcom.com
www.simcom.com

Document Title:	A7600_Series_APIs_Programming_User_Guide
Version:	1.00.16
Date:	2021.5.26
Status:	Release

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
SIMCom Headquarters Building, Building 3, No. 289 Linhong 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**

About Document

Version History

Version	Date	Chapter	What is new
V1.00.00	2019.4.27		New version
V1.00.01	2020.4.5	3.2.15 sAPI_NetworkSetCfun	Modify this API
	2020.5.4	8.1 Overview of APIs for PMU	Modify this section
	2020.5.6	6.2.1 sAPI _UartRead 6.2.3 sAPI _UartSetConfig 6.3 Default Configuration for UART	Modify these sections
	2020.5.6	2.2.1 sAPI _TaskCreate 2.2.8 sAPI _MsgQCreate 2.2.13 sAPI _SemaphoreCreate 2.2.18 sAPI _MutexCreate 2.2.21 sAPI _MutexUnlock 2.2.22 sAPI _FlagCreate 2.2.24 sAPI _FlagSet 2.2.26 sAPI _Malloc 2.2.27 sAPI _Free 2.2.28 sAPI _TimerCreate	Modify these APIs
	2020.5.6	14. APIs for MQTT(S);	Modify this chapter
	2020.5.6	7.2 Module GPIO Configuration Table	Add this section
	2020.5.6	7.3.3 sAPI _GpioConfig	Modify this API
	2020.5.6	19. APIs for Debug	Add this chapter
	2020.5.6	5.3 Demo for SMS	Modify this section
	2020.5.6	2.2.13 sAPI_SemaphoreCreate 2.2.18 sAPI _MutexCreate 2.2.25 sAPI _FlagWait	Modify these APIs
	2020.5.6	6.2.3 sAPI _UartSetConfig 6.2.4 sAPI _UartGetConfig	Modify these APIs
	2020.5.7	4.2.1 sAPI _SimPinset 4.2.2 sAPI _SimHotPlug	Modify these APIs
	2020.5.8	11.2 Detailed Description of APIs	Modify these sections

		for TCP/IP 11.3 demo for TCP/IP APIs	
	2020.5.8	9. APIs for File System 10. APIs for Internet Service 14. APIs for MQTT(S);	Modify these chapters
	2020.5.8	7. APIs for GPIO	Modify this chapter
V1.00.02	2020.5.11	3. APIs for Network	Modify this chapter
	2020.5.13	1.5.1 GPIO	Add A7670C
	2020.5.14	13.2.4 sAPI _FtpsLogout	Modify this API
	2020.5.14	15.2.2 sAPI _SsLSetContextIDMsg	Modify this API
	2020.5.20	17.2.1 sAPI _Debug	Modify this API
	2020.5.20	19. Appendixes	Modify this chapter
V1.00.03	2020.5.22	6.2.7 sAPI _UartPrintf 6.2.8 sAPI _SendATCMDWaitResp	Add these APIs
	2020.5.26	12.2.1 sAPI _TcpipPdpActive 12.2.2 sAPI _TcpipPdpDeactive	Add these APIs
	2020.6.11	4.2.4 sAPI _SysGetIccid	Add this API
	2020.6.11	2.2.11 sAPI _MsgQSendSuspend 2.2.33 sAPI _TimerGetStauts 2.2.34 sAPI _GetTicks 11.2.1 sAPI _TcpipPdpActive 11.2.2 sAPI _TcpipPdpDeactive	Add these APIs
	2020.6.11	1.6 Memory	Modify this section
	2020.6.11	9 APIs for I2C	Add this chapter
V1.00.04	2020.6.16	10.3 Result codes and demo for file system API	Modify this section
	2020.6.17	12.2.3 sAPI _TcpipPdpDeactiveNotifyDect	Add this chapter
		12.2.4 sAPI _TcpipGetSocketPdpAddr	Add this chapter
	2020.6.18	6.2.3 sAPI _UartWriteString	Add this chapter
	2020.6.22	11.2.3 sAPI _NtpUpdate	Modify this section
	2020.6.23	12.3 Demo for TCP/IP	Modify this section
	2020.7.3	6.2.1 sAPI _UartWrite modify Examples	Modify this section
		6.3.1 default configuration for uart	Modify this section

		6.3.2 received data from rx	Modify this section
	2020.7.6	11.2.3 Sapi_NtpUpdate	Modify this section
	2020.7.7	20 APIs for OTA	Add this chapter
		21.1 How to add user codes in OpenSDK	Add this chapter
		1.6 Memory	Modify this section
V1.00.05	2020.7.13	7 APIs for USB 20 APIs for Audio	Add these chapters
	2020.7.13	All	
V1.00.06	2020.7.15	20.2.1 sAPI _AudioPlaySampleRate 20.2.5 sAPI _AudioPcmPlay	Add these APIs
V1.00.07	2020.7.22	sAPI _UartSetFlowControlState sAPI _UartGetFlowControlState	Delete these APIs
		6.2.2 sAPI _UartWriteString 6.3 Description for UART	Add a API and the UART3
	2020.7.23	1.5.1 GPIO	Modify this section
	2020.7.23	5.2.4 sAPI_SmsReadMsg 5.2.5 sAPI_SmsDelAllMsg	Add a new API
	2020.7.23	13.2.27 sAPI_TcpipInetNtop	Add a new API
V1.00.08	2020.7.31	22.2.5 sAPI _FotaDownload	Add a new API
	2020.7.27	2.2.5 sAPI_TaskSleep	Modify this section
	2020.7.31	22.1 Overview of URC processor	Add the description of the SMS and the Network.
	2020.8.4	1.4 Software Framework	Modify this section
	2020.8.6	21 APIs for TTS	Modify this chapter
	2020.8.7	6 APIs for UART	Modify this chapter
	2020.8.12	22.2.5 sAPI _FotaDownload	Rename to sAPI_AppDownload
V1.00.09	2020.8.12	25 APIs for OTA	Modify this chapter
	2020.8.18	8.APIs for System Sleep	Add this chapter
	2020.8.28	11 APIs for SPI	Add this chapter
V1.00.10	2020.9.2	24 APIs for WIFI	Add this chapter
	2020.9.3	27 APIs for Call	Add this chapter
V1.00.11	2020.9.22	13 API for Flash 28 APIs for OTA	Add these chapters
	2020.9.24	2.2.1 sAPI_TaskCreate 2.2.30 sAPI_TimerStart	Modify these sections

		3.2.2 sAPI_NetworkGetCreg 3.2.3 sAPI_NetworkGetCgreg	
	2020.9.25	4.2.5 sAPI_SysGetimsi 4.2.6 Sapi_SysGetHplmn	Add these APIs
	2020.9.27	26 APIs for GNSS	Add this chapter
	2020.9.27	14.2.12 sAPI_FsFileTraverse 15.2.4 sAPI_GetSysLocalTime	Add these APIs
V1.00.12	2020.9.29	15.2.5 sAPI_SetSysLocalTime	Add this API
	2020.10.19	23.2.5 sAPI_AudioPcmPlay	Modify this API
	2020.10.28	6.2.7 sAPI_UartRxStatus	Add this API
	2020.10.30	4.2.2 sAPI _SimHotPlug	Modify this API
	2020.10.30	4.2.4 sAPI_SimIccidGet	Delete this API
V1.00.13	2020.11.5	23.2.2 sAPI_AudioPlay	Modify this API
	2020.11.5	25.2.1 sAPI_WifiScanStart 26.2.1 sAPI_GnssPowerStatusSet	Add the description of parameters
	2020.11.5	8.2.1 sAPI_SystemSleepSet	Add a Note
	2020.11.10	4.2.6 sAPI_SimcardSwitchMsg	Add this API
	2020.11.10	15.2.13 sAPI_FsIsFileNameValid	Add this API
	2020.11.10	All	Adjust the content structure
	2020.11.11	25.2.16 sAPI_AGPS	Add this API
	2020.11.12	7.2.8 sAPI_UartControl 3.2.17 sAPI_NetworkSetCtzu	Add these APIs
V1.00.14	2020.11.17	3.2.9 sAPI_NetworkGetCgdcont 3.2.11 sAPI_NetworkGetCgact 4.2.3 sAPI_SimPinGet 15.2.14 sAPI_FsNameSeparate 22.2.6 sAPI_AudioSetVolume 22.2.7 sAPI_AudioGetVolume 22.2.8 sAPI_PocInitLib 22.2.9 sAPI_PocPlaySound 22.2.10 sAPI_PocStopSound 22.2.11 sAPI_PocGetPcmAvail 22.2.10 sAPI_PocCleanBufferData 22.2.13 sAPI_PocStartRecord 22.2.14 sAPI_PocStopRecord 24.2.6 sAPI_AppPackageCrc 24.2.6 sAPI_AppPackageCrc	Add these APIs
	2020.12.24	3.2.18	Modify these APIs

		sAPI_NetworkGetCgpaddr 17.2.10 sAPI_TcpipSend 17.2.11 sAPI_TcpipRecv 17.2.13 sAPI_TcpipRecvfrom 25.2.3 sAPI_GnssNmeaDataGet 25.2.11 sAPI_GnssPowerStatusSet	
	2020.12.25	3.2.19 sAPI_NetworkGetcnetci	Add this API
	2021.1.10	2.2.35 sAPI_Gettimeofday 2.2.36 sAPI_Time	Add these APIs
	2021.1.11	12.2.6 sAPI_SPIReadBytes 12.2.7 sAPI_SPIWriteBytes 12.2.8 sAPI_SPIConfigInit	Add these APIs
	2021.1.12	18.2.4 sAPI_HttpAction 18.2.5 sAPI_HttpData	Modify these APIs
	2021.1.15	3.2.20 sAPI_NetworkGetCGAUTH 3.2.21 sAPI_NetworkSetCgauth	Add these APIs
	2021.1.18	9.2.3 sAPI_SystemAlarmClock2Wakeup	Add this API
	2021.3.23	15.2.15 sAPI_FsSync	Add this API
	2021.04.08	15.2.16 sAPI_FsFileTell	Add this API
	2021.4.20	1.6 Module list	Add this API
	2021.4.20	22.2.3 sAPI_AudioPlayMp3Cont	Add this API
	2021.4.21	25.2.4 sAPI_GnssStartMode 25.2.5 sAPI_GnssBaudRateSet 25.2.6 sAPI_GnssBaudRateGet 25.2.7 sAPI_GnssModeSet 25.2.8 sAPI_GnssModeGet 25.2.9 sAPI_GnssNmeaRateSet 25.2.10 sAPI_GnssNmeaRateGet	Modify these Notes

		25.2.11 sAPI_GnssNmeaSentenceSet 25.2.12 sAPI_GnssNmeaSentenceGet 25.2.13 sAPI_GPSInfoGet 25.2.14 sAPI_GNSSInfoGet	
	2021.04.22	31 APIs for File System of ASR1603 & ASR1803 Platforms	Add this chapter
V1.00.15	2021.04.23	10.3.3 sAPI_GpioConfig 12.2.8 sAPI_SPIConfigInit	Modify these Notes
	2021.04.23	API chapters for UART, USB and system sleep	Remove the extra semicolons from these chapters
	2021.04.23	4.2.2 sAPI_SimcardHotSwapMsg 4.2.6 sAPI_SimcardSwitchMsg	Add Examples
	2021.04.23	4.2.3 sAPI_SimPinGet 4.2.4 sAPI_SysGetimsi 4.2.5 sAPI_SimGetHplmn	Modify the header
	2021.04.23	25.2.14 sAPI_GNSSInfoGet	Modify this Note
	2021.04.26	17.2.1 sAPI_TcpipGetErrno 17.2.29 sAPI_TcpipGetSocketErrno 17.2.30 sAPI_TcpipHtonl 17.2.31 sAPI_TcpipGetaddrinfo 17.2.32 sAPI_TcpipGetaddrinfo_with_pcid 17.2.33 sAPI_TcpipFreeaddrinfo 17.2.34 sAPI_TcpipSocketWithCallback 17.2.35 sAPI_Tcpipnet_pton	Add this API
	2021.04.28	25.2.4 sAPI_GnssStartMode 25.2.5 sAPI_GnssBaudRateSet 25.2.6 sAPI_GnssBaudRateGet	Modify these Notes

		25.2.7 sAPI_GnssModeSet 25.2.8 sAPI_GnssModeGet 25.2.9 sAPI_GnssNmeaRateSet 25.2.10 sAPI_GnssNmeaRateGet 25.2.11 sAPI_GnssNmeaSentenceSet 25.2.12 sAPI_GnssNmeaSentenceGet 25.2.13 sAPI_GpsInfoGet 25.2.14 sAPI_GnssInfoGet 25.2.15 sAPI_SendCmd2Gnss 25.2.16 sAPI_GnssAgpsSeviceOpen	
	2021.04.28	10.3.6 sAPI_GpioGetDirection 10.3.7 sAPI_GpioWakeupEnable 13.2.3 sAPI_I2CreadEx	Add these APIs
	2021.04.28	10.3.5 sAPIGpioSetDirection	Modify this Note
	2021.04.28	15.2.17 sAPI_FsFileSave	Add this API
	2021.04.28	3.2.19 sAPI_NetworkGetCnetci 3.2.20 sAPI_NetworkGetCgauth 3.2.21 sAPI_NetworkSetCgauth 6.2.3 sAPI_SmsSendStorageMsg	Modify these Notes
	2021.04.28	22.2.7 sAPI_AudioPcmStop 22.2.21 sAPI_AudioSetMicGain 22.2.22 sAPI_AudioGetMicGain	Add these APIs
	2021.05.6	17.2.24 sAPI_TcpipInetAddr 17.2.27 sAPI_TcpipInetNtoa 17.2.38 sAPI_TcpipInetNtop 17.2.32 sAPI_TcpipGetaddrinfoWithPcid 17.2.34	Modify these sectionss

		sAPI_TcpipSocketWithCallback 17.2.35 sAPI_TcpipnetPton	
	2021.05.6	7.2.9 sAPI_UartRefRegister 8.2.2 sAPI_UsbVcomRefRegister	Add these APIs
V1.00.16	2021.05.11	27.2.15 sAPI_BleScan	Add the api
	2021.05.13	11.2.6 sAPI_GetPowerUpEvent	Add the api
	2021.05.13	11.2.7 sAPI_GetPowerDownEvent	Add the api
	2021.05.25	2.2.38 sAPI_ContextLock 2.2.39 sAPI_ContextUnlock	Add these APIs
	2021.05.26	Modify version etc.	
	2021.05.31	7.2.3 sAPI_UartSetConfig	Modified API example
	2021.06.08	5.2.5 sAPI_CallAutoAnswer	Add the api
	2021.07.03	2.2.40 sAPI_GettimeofdaySyncRtc	Add the api
	2021.07.05	22.2.23 sAPI_AudioMp3StreamPlay 22.2.24 sAPI_AudioMp3StreamStop 22.2.25 sAPI_AudioAmrStreamPlay 22.2.26 sAPI_AudioAmrStreamStop	Add these apis
	2021.07.07	22.2.27 sAPI_AudioPlayMp3Cont	Add the api
	2021.07.29	22.2.28 sAPI_AudioWavFilePlay 22.2.29 sAPI_AudioSetPlayPath 22.2.30 sAPI_AudioGetPlayPath	Add these apis
	2021.08.27	2.2.19 sAPI_MutexCreate	Modify API example & parametric description
	2021.09.01	9.2.3 sAPI_SystemSleepExSet 9.2.4 sAPI_SystemSleepExGet	Add these apis

Scope

This document presents the APIs for SIMCom A76xx Series modules of CAT1, including A76xx(x)(-xxxx), and A7620.

Contents

1 Introduction

1.1 Purpose of the document

Based on module AT command manual, this document will introduce the APIs for OpenSDK.

Developers could understand and develop application quickly and efficiently based on this document.

1.2 Related documents

[1] A76xx_Series_Open_SDK_编译指南_V1.xx.xx.pdf
[2] A76xx_Series_Open_SDK_Debug_and_Download_Application_Note_V1.xx.xx.pdf
[3] A76xx_Series_Open_SDK_用户开发与 DEMO 使用指南_V1.00.01.pdf
[4]分离式二次开发 Flash 和 RAM 资源表_V1.xx.xx.pdf

1.3 Conventions and abbreviations

In this document, the GSM engines are referred to as following term:
ME(Mobile Equipment);
MS(Mobile Station);
TA(Terminal Adapter);
DCE(Data Communication Equipment); or facsimile DCE(FAX modem, FAX board);

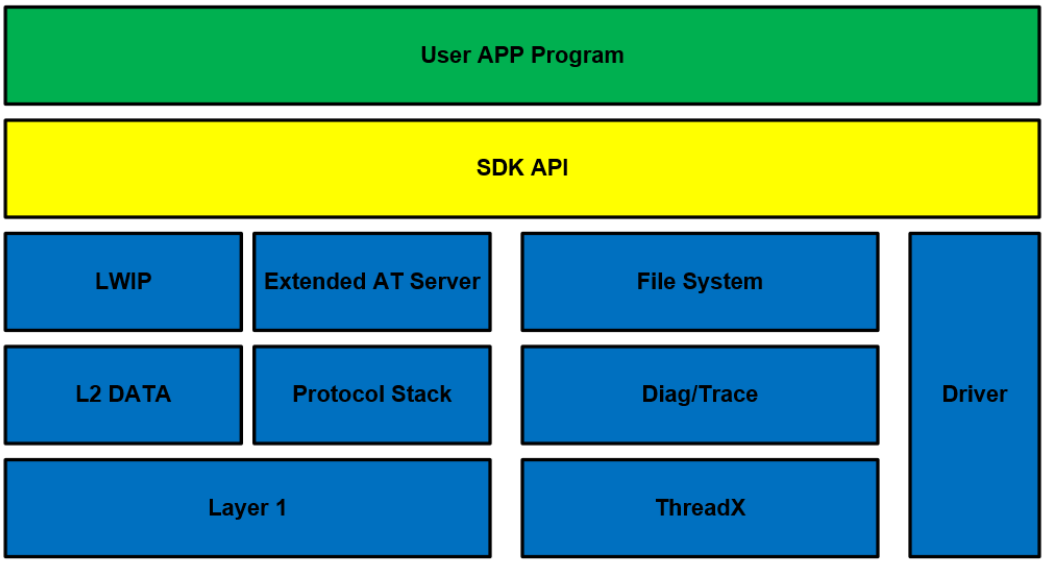
In application, controlling device controls the GSM engine by sending AT Command via its serial interface.
The controlling device at the other end of the serial line is referred to as following term:
TE(Terminal Equipment);
DTE(Data Terminal Equipment); or plainly "the application" which is running on an embedded system;

The following table lists the description of type names in the codes.

Type Name	Description
BOOL	unsigned char

UINT8	unsigned char
UINT16	unsigned short
UINT32	unsigned long
UINT64	unsigned long long
INT8	signed char
INT16	signed short
INT32	signed long
INT64	signed long long

1.4 Software Framework



1.5 Hardware Resource

1.5.1 Module Pin Resources

Please refer to following documents:

[1] A7600C1 系列_二次开发设计手册.pdf
[2] A7620 二次开发硬件设计手册_V1.01.pdf
[3] A7670 系列二次开发硬件设计手册_V1.02.pdf
[4] A7600C1-MNSE_二次开发硬件设计手册_V1.00.pdf

[5] A7630C 二次开发设计手册 V1.02.pdf

1.6 Module list

Model name	PN	Platform
A7600C1-LNSE	S2-109BG	ASR1601
	S2-109XW	
	S2-109X6	
A7600C1-MNSE	S2-109LQ	ASR1601
A7670C	S2-109G7	ASR1601
	S2-109XY	
A7670C-LAAL	S2-109S1	ASR1601
	S2-109ZB	ASR1601
A7670C-LAAE	S2-10A28	ASR1601
A7630C-LAAL	S2-109X4	ASR1601
A7670E	S2-109GE	ASR1601
A7600E-LNSE	S2-109BH	ASR1601
	S2-109X8	ASR1601
	S2-109XX	ASR1601
A7670C-FASL	S2-109ZN	ASR1603
	S2-10A2S	
A7670C-LASL	S2-109ZL	ASR1603
	S2-10A6S	
A7670C-MASL	S2-10A6Q	ASR1603
A7670C-BASL	S2-10A2L	ASR1603
	S2-10A6N	

module list

2 APIs for OS

2.1 Overview of APIs for OS

Interface	Description
sAPI_TaskCreate	Create a task, this task will be started automatically.
sAPI_TaskDelete	Delete the task
sAPI_TaskSuspend	Suspend a task
sAPI_TaskResume	Resume a suspended task
sAPI_TaskSleep	Suspend the current executing task for specified time
sAPI_TaskGetCurrentRef	Get the currently executing task
sAPI_TaskTerminate	Terminate a task.
sAPI_MsgQCreate	Create a message queue
sAPI_MsgQDelete	Delete a message queue
sAPI_MsgQSend	Send a message to the message queue.
sAPI_MsgQSendSuspend	Send a message to the message queue in blocking mode.
sAPI_MsgQRecv	Receive a message from the message queue
sAPI_MsqQPoll	Get the current message count in message queue
sAPI_SemaphoreCreate	Create a counting semaphore
sAPI_SemaphoreDlete	Delete the semaphore
sAPI_SemaphoreAcquire	Retrieves an instance from the semaphore, its count is decreased by one.
sAPI_SemaphoreRelease	Put the instance to the semaphore, its count is increased by one.
sAPI_SemaphorePoll	Get the current semaphore's count
sAPI_MutexCreate	Create a mutex
sAPI_MutexDelete	Delete a mutex
sAPI_MutexLock	Get the ownership of this mutex
sAPI_MutexUnlock	Releases the previously obtained mutex
sAPI_FlagCreate	Create a group of 32 event flags.
sAPI_FlagDelete	Delete the flag group
sAPI_FlagSet	Set or clear the event flag in the event group
sAPI_FlagWait	Set or clear the event flag in the event group
sAPI_malloc	Allocate a block of Size bytes of memory
sAPI_free	Free the memory to the defaultmemory pool
sAPI_TimerCreate	Create a software timer

sAPI_TimerStart	Start the software timer with its specified expiration function and periodic
sAPI_TimerStop	Stop the software timer
sAPI_TimerDelete	Delete the software timer
sAPI_TimerGetStatus	This function requests that the status of the specified timer be returned in the sTimerStatus structure
sAPI_GetTicks	This function requests the elapsed time, in OS clock tick, since the last system start-up
sAPI_Gettimeofday	This function can get time, and gives the number of seconds and microseconds since the Epoch.
sAPI_Time	This function returns the time as the number of seconds since the Epoch
sAPI_GetSystemInfo	Output cpu usage rate and remaining heap space size

2.2 Detailed Description of APIs for OS

2.2.1 sAPI_TaskCreate

Create a task, this task will be started automatically.

sAPI_TaskCreate	
Prototype	SC_STATUS **sAPI_TaskCreate**(sTaskRef *taskRef, void* stackPtr, UINT32 stackSize, UINT8 priority, char *taskName, void(*taskStart);(void*), void* argv);
Parameters	[in] **stackPtr**: Pointer to the low address of the stack. [in] **stackSize**: Maximum size of the stack. [in] **priority**: The priority of the proposed task ranges from 80 to 150. [in] **taskName**: Pointer to an 8 character name for the task. The name does not have to be null-terminated. [in] **taskStart:** Entry function of the task. [in] **argv:** Argument to be passed into task entry function. [out] **taskRef:** OS assigned reference to the task.
Return value	SC_SUCCESS: Successful completion of the service. SC_INVALID_REF: Task reference is NULL. SC_INVALID_PTR: Task's entry function pointer is NULL. SC_INVALID_MEMORY: Pointer to stack memory is NULL. SC_INVALID_SIZE: Stack size is insufficient. SC_INVALID_PRIORITY: Priority is invalid. SC_NO_TASKS: No available task references. SC_FAIL: OS specific error.
Header	#include "simcom_os.h"

Examples

```
sTaskRef taskRef;
SC_STATUS status;
status = sAPI_TaskCreate(&taskRef, stack_ptr, 2000, 20, "TASK_1”, task_entry, NULL);
```

2.2.2 sAPI_TaskDelete

This function requests that the specified task be deleted.

sAPI_TaskDelete	
Prototype	SC_STATUS **sAPI_TaskDelete**(sTaskRef taskRef);
Parameters	[in] **taskRef**: Reference to the task.
Return value	SC_SUCCESS: Successful completion of the service. SC_INVALID_REF: Invalid task reference. SC_FAIL: Task could not be deleted because it was not in the Suspended state or due to other OS specific error.
Header	#include "simcom_os.h"

Examples

```
sTaskRef taskRef;
SC_STATUS status;
status = sAPI_TaskDelete(taskRef);
```

2.2.3 sAPI_TaskSuspend

This function requests that a specific task be suspended including the current task.

sAPI_TaskSuspend	
Prototype	SC_STATUS **sAPI_TaskSuspend**(sTaskRef taskRef);
Parameters	[in] **taskRef**: Reference to the task.
Return value	SC_SUCCESS: Successful completion of the service. SC_INVALID_REF: Task reference is NULL. SC_FAIL: OS specific error.
Header	#include "simcom_os.h"

Examples

```
sTaskRef taskRef;
SC_STATUS status;
status = sAPI_TaskSuspend(taskRef);
```

2.2.4 sAPI_TaskResume

This function requests that a specified task be resumed.

sAPI_TaskResume	
Prototype	SC_STATUS **sAPI_TaskResume**(sTaskRef taskRef);
Parameters	[in] **taskRef**: Reference to the task.
Return value	SC_SUCCESS: Successful completion of the service. SC_INVALID_REF: Invalid task reference. SC_FAIL: Task could not be deleted because it was not in the Suspended state or due to other OS specific error.
Header	#include "simcom_os.h"

Examples

```
sTaskRef taskRef;
SC_STATUS status;
status = sAPI_TaskResume(taskRef);
```

2.2.5 sAPI_TaskSleep

Suspend the current executing task for specified time. In our OS, 1 ticks = 5ms.

sAPI_TaskResume	
Prototype	void **sAPI_TaskSleep**(UINT32 ticks);
Parameters	[in] **ticks**: Number of OS clock ticks to sleep.
Return value	None
Header	#include "simcom_os.h"

Examples

sAPI_TaskSleep(200);

2.2.6 sAPI_TaskGetCurrentRef

Get the currently executing task

sAPI_TaskGetCurrentRef	
Prototype	SC_STATUS **sAPI_ TaskGetCurrentRef**(sTaskRef *taskRef);
Parameters	[out] **taskRef:** OS assigned reference to the task.
Return value	SC_SUCCESS:Successful completion of the service. SC_INVALID_REF: Invalid task reference. SC_FAIL: Task could not be deleted because it was not in the Suspended state or due to other OS specific error.
Header	#include "simcom_os.h"

Examples

```
sTaskRef taskRef;
SC_STATUS status;
status = sAPI_ TaskGetCurrentRef(&taskRef);
```

2.2.7 sAPI_TaskTerminate

Terminate a task. Please use this API carefully. A task in a terminated state cannotexecute again

sAPI_TaskTerminate	
Prototype	SC_STATUS **sAPI_TaskTerminate**(sTaskRef taskRef);
Parameters	[in] **taskRef**: Reference to the task.
Return value	SC_SUCCESS: Successful completion of the service. SC_INVALID_REF: Invalid task reference. SC_FAIL: Task could not be deleted because it was not in the Suspended state or due to other OS specific error.
Header	#include "simcom_os.h"

Examples

```
sTaskRef taskRef;
SC_STATUS status;
status = sAPI_TaskTerminate(taskRef);
```

2.2.8 sAPI_MsgQCreate

This function requests that a message queue be created. All memory used to store messages on the message queue is allocated by the operating system.

sAPI_MsgQCreate	
Prototype	SC_STATUS **sAPI_MsgQCreate**(sMsgQRef *msgQRef, char *queueName, UINT32 maxSize, UINT32 maxNumber, UINT32 waitingMode);
Parameters	[in] **queueName**: 8 character name of queue. . The name does not have to be null-terminated. [in] **maxSize**: Maximum size of a message on the queue. This is used for error checking by sAPI_MsgQSend();. [in] **maxNumber**: Maximum number of messages on the queue [in] **waitingMode**: Defines scheduling of waiting events: SC_FIFO, or SC_PRIORITY. [out] **msgQRef:** Point to location to hold message queue reference allocated by the operating system.
Return value	SC_SUCCESS: Successful completion of the service. SC_INVALID_REF: Invalid queue reference. SC_INVALID_MODE: Invalid waiting mode. SC_INVALID_SIZE: Invalid queue size. SC_NO_QUEUES: No available queues left in the system. SC_FAIL: OS specific error.
Header	#include "simcom_os.h"

Examples

```
sMsgQRef msgQRef;
SC_STATUS status;
status = sAPI_MsgQCreate(&msgQRef, “testQ”, 32, 200, SC_PRIORITY);
```

2.2.9 sAPI_MsgQDelete

This function requests that the specified message queue be deleted.

sAPI_MsgQDelete	
Prototype	SC_STATUS **sAPI_MsgQDelete**(sMsgQRef msgQRef);
Parameters	[in] **msgQRef**: Identifier that uniquely identifies the message queue.
Return value	SC_SUCCESS: Successful completion of the service. SC_INVALID_REF: Invalid task reference. SC_FAIL OS specific error.
Header	#include "simcom_os.h"

Examples

```
sMsgQRef msgQRef;
SC_STATUS status;
status = sAPI_MsgQDelete(msgQRef);
```

2.2.10 sAPI_MsgQSend

This function requests that a message be sent to the specified message queue.

sAPI_MsgQSend	
Prototype	SC_STATUS **sAPI_MsgQSend**(sMsgQRef msgQRef, SIM_MSG_T *msgPtr);
Parameters	[in] **msgQRef**: Identifier that uniquely identifies the message queue. [in] **msgPtr**: Starting address of the data.
Return value	SC_SUCCESS: Successful completion of the service. SC_INVALID_REF: Invalid queue reference. SC_INVALID_POINTER: Message pointer is NULL. SC_QUEUE_FULL: Indicates the message queue is full. SC_INVALID_SIZE: Indicates the message size is incompatible with the message size supported by the queue. The maximum size of message that may be sent to the queue is specified in sAPI_MsgQCreate(); SC_FAIL: OS specific error.
Header	#include "simcom_os.h"

Examples

```
sMsgQRef msgQRef;
SC_STATUS status;
SIM_MSG_T msg;
status = sAPI_MsgQSend(msgQRef, &msg);
```

2.2.11 sAPI_MsgQSendSuspend

This function requests that a message be sent to the specified message queue in blocking mode.

sAPI_MsgQSendSuspend	
Prototype	SC_STATUS **sAPI_MsgQSendSuspend**(sMsgQRef msgQRef, SIM_MSG_T *msgPtr, UINT32 timeout);
Parameters	[in] **msgQRef**: Identifier that uniquely identifies the message queue. [in] **msgPtr**: Starting address of the data. [in] **timeout**: Specified ticks. 1 ticks = 5ms.If timeout = SC_SUSPEND, the function will block until there is space available on the queue.
Return value	SC_SUCCESS: Successful completion of the service. SC_INVALID_REF: Invalid queue reference. SC_INVALID_POINTER: Message pointer is NULL. SC_QUEUE_FULL: Indicates the message queue is full. SC_INVALID_SIZE: Indicates the message size is incompatible with the message size supported by the queue. The maximum size of message that may be sent to the queue is specified in sAPI_MsgQCreate(); SC_FAIL: OS specific error.
Header	#include "simcom_os.h"

Examples

```
sMsgQRef msgQRef;
SC_STATUS status;
SIM_MSG_T msg;
status = sAPI_MsgQSend(msgQRef, &msg);
```

2.2.12 sAPI_MsgQRecv

This function requests that a message be received from the specified message queue. If the queue is empty, the blocking behavior of the call is determined by the value of the "timeout" argument.

sAPI_MsgQRecv	
Prototype	SC_STATUS **sAPI_MsgQRecv**(sMsgQRef msgQRef, SIM_MSG_T *recvMsg, UINT32 timeout);
Parameters	[in] **msgQRef**: Identifier that uniquely identifies the message queue. [in] **timeout**: If timeout is set to SC_NO_SUSPEND, this call will not block. If timeout is set to SC_SUSPEND, this call will block until a message is available on the queue. If a timeout value between 1 and 4, 294, 967, 293 is specified, the call

	will block until a message is available or until the timeout period, in number of OS clock ticks, elapses. [out] **recvMsg**: Pointer to application supplied buffer to which the received message should be copied.
Return value	SC_SUCCESS: Successful completion of the service.It indicates a message has been copied to the receiving task's"recvMsg" buffer. SC_INVALID_REF: Invalid queue reference. SC_INVALID_POINTER: "recvMsg" pointer is NULL. SC_QUEUE_EMPTY: Indicates the message queue is empty. SC_TIMEOUT: A timeout has occurred while suspended waiting for a message on an empty queue. SC_INVALID_SIZE: Indicates that the actual message size is larger than the size of the application receive buffer as indicated by the 'size' parameter SC_FAIL OS: specific error.
Header	#include "simcom_os.h"

Examples

```
sMsgQRef msgQRef;
SC_STATUS status;
SIM_MSG_T recvMsg[20] ;
status = sAPI_MsgQRecv(msgQRef, recvMsg, SC_SUSPEND);
```

2.2.13 sAPI_MsgQPoll

This function checks the number of messages on the message queue.

sAPI_MsgQPoll	
Prototype	SC_STATUS **sAPI_MsgQPoll**(sMsgQRef msgQRef, UINT32* msgCount);
Parameters	[in] **msgQRef**: Identifier that uniquely identifies the message queue. [out] **msgCount:** On return from this function, msgCount contains the number of messages on the queue.
Return value	SC_SUCCESS: Successful completion of the service. SC_INVALID_POINTER: msgCount pointer is NULL. SC_INVALID_REF: Invalid queue reference
Header	#include "simcom_os.h"

Examples

```
sMsgQRef msgQRef;
```

```
SC_STATUS status;
UINT32 msgCount;
status = sAPI_MsgQPoll(msgQRef, &msgCount);
```

2.2.14 sAPI_SemaphoreCreate

This function requests that a counting semaphore be created.

sAPI_SemaphoreCreate	
Prototype	SC_STATUS **sAPI_SemaphoreCreate**(sSemaRef *semaRef, UINT32 initialCount, UINT32 waitingMode);
Parameters	[in] **initialCount**: Initial count of the semaphore. [in] **waitingMode**: SC_FIFO or SC_PRIORITY. "waitingMode" specifies how tasks are added to a semaphore's Wait queue. They may be added in first-in-first-out order(SC_FIFO); or in priority order(SC_PRIORITY); with the highest priority waiting task at the front on the queue. [out] **semaRef:** OS assigned reference to the semaphore
Return value	SC_SUCCESS: Successful completion of the service. SC_INVALID_REF: Invalid semaphore reference. SC_INVALID_MODE: Invalid mode. SC_NO_SEMAPHORES: No available semaphores
Header	#include "simcom_os.h"

Examples

```
sSemaRef semaRef;
SC_STATUS status;
status = sAPI_SemaphoreCreate(&semaRef, 2, SC_FIFO);
```

2.2.15 sAPI_SemaphoreDelete

This function requests that a counting semaphore be deleted.

sAPI_SemaphoreDlete	
Prototype	SC_STATUS **sAPI_ SemaphoreDelete**(sSemaRef semaRef);
Parameters	[in] **semaRef:** OS assigned reference to the semaphore.
Return value	SC_SUCCESS: Successful completion of the service. SC_INVALID_REF: Invalid semaphore reference.

	SC_FAIL: OS specific error.
Header	#include "simcom_os.h"

Examples

```
sSemaRef semaRef;
SC_STATUS status;
status = sAPI_ SemaphoreDelete(semaRef);
```

2.2.16 sAPI_SemaphoreAcquire

This function requests that the specified semaphore be decremented. If the semaphore count is zero before this call, the service cannot be satisfied immediately. In that case, the blocking behavior is specified by the "timeout" parameter.

sAPI_SemaphoreAcquire	
Prototype	SC_STATUS **sAPI_SemaphoreAcquire**(sSemaRef semaRef, UINT32 timeout);
Parameters	[in] **semaRef**: OS assigned reference to the semaphore. [in] **timeout**: If timeout is set to SC_NO_SUSPEND, this call will not block. If timeout is set to SC_SUSPEND, this call will block until the semaphore count is greater than zero. If a timeout value between 1 and 4, 294, 967, 293 is specified, the call will block until the semaphore count is greater than zero or the timeout period, in number of OS clock ticks, elapses.
Return value	SC_SUCCESS: Semaphore has been decremented. SC_INVALID_REF: Invalid semaphore reference. SC_UNAVAILABLE: The semaphore is not available. SC_TIMEOUT: A timeout has occurred while waiting for the semaphore count to be greater than zero. SC_FAIL: OS specific error.
Header	#include "simcom_os.h"

Examples

```
sSemaRef semaRef;
SC_STATUS status;
status = sAPI_SemaphoreAcquire(semaRef, 200);
```

2.2.17 sAPI_SemaphoreRelease

If there are any tasks waiting for the semaphore, the first waiting task is made ready to run. If there are no tasks waiting, the value of the semaphore is incremented by one.

sAPI_SemaphoreRelease	
Prototype	SC_STATUS **sAPI_SemaphoreRelease**(sSemaRef semaRef);
Parameters	[in] **semaRef**: OS assigned reference to the semaphore.
Return value	SC_SUCCESS: Successful completion of the service. SC_INVALID_REF: Invalid semaphore reference.
Header	#include "simcom_os.h"

Examples

```
sSemaRef semaRef;
SC_STATUS status;
status = sAPI_SemaphoreRelease(semaRef);
```

2.2.18 sAPI_SemaphorePoll

This function requests that a counting semaphore be created.

sAPI_SemaphorePoll	
Prototype	SC_STATUS **sAPI_SemaphorePoll**(sSemaRef semaRef, UINT32 *count);
Parameters	[in] **semaRef**: OS assigned reference to the semaphore. [out] **count**: Current value of the semaphore.
Return value	SC_SUCCESS: Successful completion of the service. SC_INVALID_REF: Invalid semaphore reference.
Header	#include "simcom_os.h"

Examples

```
sSemaRef semaRef;
SC_STATUS status;
UINT32 currentCount;
status = sAPI_SemaphorePoll(semaRef, &currentCount);
```

2.2.19 sAPI_MutexCreate

This function requests that a mutex be created. Mutexes use the priority inheritance protocol to bound the time spent in priority inversions.

sAPI_MutexCreate	
Prototype	SC_STATUS **sAPI_MutexCreate**(sMutexRef *mutexRef, UINT32 waitingMode);
Parameters	[in] **waitingMode**: SC_FIFO or SC_PRIORITY. [out] **mutexRef**: OS assigned reference to the mutex
Return value	SC_SUCCESS: Successful completion of the service. SC_INVALID_REF: Invalid mutex reference. SC_INVALID_MODE: Invalid mode. SC_NO_MUTEXES: No available mutexes in the system. SC_FAIL: Mutex already locked by the task.
Header	#include "simcom_os.h"

Examples

```
sMutexRef mutexRef;
SC_STATUS status;
status = sAPI_MutexCreate(&mutexRef, waitingMode);
```

2.2.20 sAPI_MutexDelete

This function requests that the specified mutex be deleted.

sAPI_MutexDelete	
Prototype	SC_STATUS **sAPI_MutexDelete**(sMutexRef mutexRef);
Parameters	[in] **mutexRef**: OS assigned reference to the semaphore.
Return value	SC_SUCCESS: Successful completion of the service. SC_INVALID_REF: Invalid mutex reference. SC_FAIL: OS specific error.
Header	#include "simcom_os.h"

Examples

```
sMutexRef mutexRef;
SC_STATUS status;
status = sAPI_MutexDlete(mutexRef);
```

2.2.21 sAPI_MutexLock

This function requests that the specified mutex be locked. If the mutex is locked by another task before this call, the service cannot be satisfied immediately. In this case, the blocking behavior is specified by the "timeout" parameter.

sAPI_MutexLock	
Prototype	SC_STATUS **sAPI_MutexLock**(sMutexRef mutexRef, UINT32 timeout);
Parameters	[in] **mutexRef**: OS assigned reference to the mutex. [in] **timeout**: If timeout is set to SC_NO_SUSPEND, this call will not block. If timeout is set to SC_SUSPEND, this call will block until the semaphore count is greater than zero. If a timeout value between 1 and 4, 294, 967, 293 is specified, the call will block until the mutex is unlocked or the timeout period, in number of OS clock ticks, elapses.
Return value	SC_SUCCESS: Mutex has been acquired. SC_INVALID_REF: Invalid mutex reference. SC_UNAVAIABLE: The mutex is not available. It is locked by another task. SC_TIMEOUT: A timeout has occurred while waiting for the mutex. SC _FAIL: The calling task already has locked the mutex.
Header	#include "simcom_os.h"

Examples

```
sMutexRef mutexRef;
SC_STATUS status;
status = sAPI_MutexCreate(mutexRef, 200);
```

2.2.22 sAPI_MutexUnlock

If there are any tasks waiting for the mutex, the task at the front of the Wait queue is made ready to run. Only the mutex owner may unlock a mutex.

sAPI_MutexUnlock	
Prototype	SC_STATUS **sAPI_MutexUnlock**(sMutexRef mutexRef);
Parameters	[in] **mutexRef**: OS assigned reference to the mutex.

Return value	SC_SUCCESS: Successful completion of the service. SC_INVALID_REF: Invalid mutex reference. SC_FAIL: The calling task is not the mutex owner.
Header	#include "simcom_os.h"

Examples

```
sMutexRef mutexRef;
SC_STATUS status;
status = sAPI_MutexUnlock(mutexRef);
```

2.2.23 sAPI_FlagCreate

This function requests that a flag group be created.

sAPI_FlagCreate	
Prototype	SC_STATUS **sAPI_FlagCreate**(sFlagRef *flagRef);
Parameters	[out] **flagRef**: OS assigned reference to the flag.
Return value	SC_SUCCESS: Successful completion of the service. SC_INVALID_REF: Pointer to "flagRef " is NULL. SC_NO_FLAGS: No available flags left in the system.
Header	#include "simcom_os.h"

Examples

```
sFlagRef flagRef;
SC_STATUS status;
status = sAPI_FlagCreate(&flagRef);
```

2.2.24 sAPI_FlagDelete

This function requests that a flag group be deleted

sAPI_FlagDlete	
Prototype	SC_STATUS **sAPI_FlagDlete**(sFlagRef flagRef);
Parameters	[in] **flagRef**: OS assigned reference to the flag.
Return value	SC_SUCCESS: Successful completion of the service.

	SC_INVALID_REF: Invalid flag reference. SC_FAIL: OS specific error.
Header	#include "simcom_os.h"

Examples

```
sFlagRef flagRef;
SC_STATUS status;
status = sAPI_FlagCreate(&flagRef);
```

2.2.25 sAPI_FlagSet

This function executes a logical OR or AND of the flag group with the input mask.

sAPI_FlagSet	
Prototype	SC_STATUS **sAPI_FlagSet**(sFlagRef flagRef, UINT32 mask, UINT32 operation);
Parameters	[in] **flagRef**: OS assigned reference to the flag. [in] **mask**: Mask specifying which bits need to be set. A 1 in a certain bit position will set the same bit position in the flag. [in] **operation**: Logical operation to be executed on the flag group. SC_FLAG_AND executes a logical AND and SC_FLAG_OR executes a logical OR.
Return value	SC_SUCCESS: Successful completion of the service. SC_INVALID_REF: Flag reference is NULL. SC_INVALID_MODE: Operation is invalid.
Header	#include "simcom_os.h"

Examples

```
sFlagRef flagRef;
SC_STATUS status;
status = sAPI_FlagSet(flagRef, 0x00000094, SC_OR);
```

2.2.26 sAPI_FlagWait

This function waits for the specified operation on a flag group to complete. The operation is defined by the "operation" input parameter. If the timeout input parameter is SC_NO_SUSPEND the operation completes immediately and the current value of the flags is returned in the output parameter "flags". By specifying

SC_NO_SUSPEND, an application can read the current value of the flags without blocking.

sAPI_FlagWait	
Prototype	SC_STATUS **sAPI_FlagWait**(sFlagRef flagRef, UINT32 mask, UINT32 operation, UINT32* flags, UINT32 timeout););
Parameters	[in] **flagRef**: OS assigned reference to the flag. [in] **mask**: Mask of flags to wait for. [in] **operation**: May be one of the following: **SC_FLAG_AND**: Wait for all of the bits in the input mask **SC_FLAG_AND_CLEAR**: Wait for all of the bits in the input mask to be set, clear all event flags on successful **SC_FLAG_OR**: Wait for any of the bits in the input mask to be set, don't clear the event flags **SC_FLAG_OR_CLEAR**: Wait for any of the bits in the input mask to completion to be set, don't clear the event flags [in] **timeout**: If timeout is set to SC_NO_SUSPEND, the operation completes immediately and the current value of the flags is returned in the output parameter "flags". If timeout is set to SC_SUSPEND, this call will block until the condition specified by the "mask" and "operation" inputs is satisfied. If a timeout value between 1 and 4, 294, 967, 293 is specified, the call will block until the wait condition is satisfied or until the timeout period, in number of OS clock ticks, elapses. [out] **flags**: The current value of all flags
Return value	SC_SUCCESS: Successful completion of the service. SC_INVALID_REF: Flag reference is NULL. SC_INVALID_MODE: Invalid operation. SC_INVALID_POINTER: Pointer to 'flags' is NULL. SC_TIMEOUT: A timeout has occurred while suspended waiting for a Wait operation to complete. SC_FLAG_NOT_PRESENT: Flag combination not met when using SC_NO_SUSPEND.
Header	#include "simcom_os.h"

Examples

```
sFlagRef flagRef;
SC_STATUS status;
UINT32 flags;
status = sAPI_FlagWait(flagRef, 0x0000094, SC_FLAG_AND_CLEAR, &flags, SC_SUSPEND);
```

2.2.27 sAPI_Malloc

Allocate a block of Size bytes of memory from the default memory pool, returning a pointer to the beginning of the block

sAPI_ malloc	
Prototype	void ***sAPI_Malloc**(UINT32 Size);
Parameters	[in] **size**: Number of bytes to be allocated.
Return value	Points to the first address of an allocated memory space.If return is NULL, description failed to allocate memory
Header	#include "simcom_os.h"

Examples

```
void * p = sAPI_Malloc(8);
```

2.2.28 sAPI_Free

Free the memory to the defaultmemory pool

sAPI_free	
Prototype	void **sAPI_Free**(void *p);
Parameters	[in] **p**: The first address of the memory that needs to be released.
Return value	None
Header	#include "simcom_os.h"

Examples

```
sAPI_Free(p);
```

2.2.29 sAPI_TimerCreate

This function allocates a timer. The state of the allocated timer is inactive. sAPI_TimerStart(); is used to activate the timer.

sAPI_TimerCreate	
Prototype	SC_STATUS **sAPI_TimerCreate**(sTimerRef *timerRef);

Parameters	[in] **timerRef**: Address to store a reference to the timer allocated by the operating system.
Return value	SC_SUCCESS: Successful completion of the service. SC_NO_TIMERS: No available timers in the system. SC_INVALID_REF: Input argument "timerRef" is a NULL pointer.
Header	#include "simcom_os.h"

Examples

```
sTimerRef timerRef;
SC_STATUS status;
status = sAPI_TimerCreate(&timerRef);
```

2.2.30 sAPI_TimerStart

This function requests that an inactive timer be started and the callback function executed at the expiration of the timer.

sAPI_TimerStart	
Prototype	SC_STATUS **sAPI_TimerStart**(sTimerRef timerRef, UINT32 initialTime, UINT32 rescheduleTime, void(*callBackRoutine);(UINT32), UINT32 timerArgc););
Parameters	[in] **initialTime**: Initial expiration time in OS clock ticks. [in] **rescheduleTime**: If 0, cyclic timing is disabled and the timer only expires once. If not zero, it indicates the period, in OS clock ticks, of a cyclic timer. 1 tick = 5ms. [in] **callBackRoutine**: Specifies the application routine to execute each time the timer expires. The callback function must not invoke any "blocking" operating system calls. [in] **timerArgc**: Argument to be passed to callback routine on expiration [out] **timerRef**: OS supplied timer reference
Return value	SC_SUCCESS: Successful completion of the service. SC_INVALID_REF: Input argument "timerRef" is not a valid timer reference. SC_INVALID_PTR: Input argurment "callBackRoutine" is a NULL pointer SC_FAIL: Timer is still active.
Header	#include "simcom_os.h"

NOTE

Please don’t perform blocking operations in the timer callback function.

Examples

```
sTimerRef timerRef;
SC_STATUS status;
void timerRoutine(UINT32);
status = sAPI_TimerStart(timerRef, 20, 56, timerRoutine(UINT32), 0x1234);
```

2.2.31 sAPI_TimerStop

This function requests that the state of an active timer be changed to inactive. Calling this function when the state of the timer is inactive has no effect.

sAPI_TimerStop	
Prototype	SC_STATUS **sAPI_TimerStop(**sTimerRef timerRef);
Parameters	[in] **timerRef**: OS assigned reference to the timer.
Return value	SC_SUCCESS: Successful completion of the service. SC_INVALID_REF: Invalid timer reference.
Header	#include "simcom_os.h"

Examples

```
sTimerRef timerRef;
SC_STATUS status;
status = sAPI_TimerStop(timerRef);
```

2.2.32 sAPI_TimerDelete

This function requests that the specified timer be deleted. The timer must be inactive when this function is called.

sAPI_TimerDelete	
Prototype	SC_STATUS **sAPI_TimerDelete**(sTimerRef timerRef);
Parameters	[in] **timerRef**: Timer reference that was allocated when the timer was created.
Return value	SC_SUCCESS: Successful completion of the service. SC_INVALID_REF: Invalid timer reference. SC_FAIL: Timer is still active.

Header	#include "simcom_os.h"

Examples

```
sTimerRef timerRef;
SC_STATUS status;
status = sAPI_TimerDelete(timerRef);
```

2.2.33 sAPI_TimerGetStatus

This function requests that the status of the specified timer be returned in the sTimerStatus structure.

sAPI_TimerGetStatus	
Prototype	SC_STATUS **sAPI_TimerGetStatus**(sTimerRef timerRef, sTimerStatus* timerStatus);
Parameters	[in] **timerRef**: Timer reference that was allocated when the timer was created. [out] **timerStatus**: Pointer to the structure that will be filled in.
Return value	SC_SUCCESS: Successful completion of the service. SC_INVALID_REF: Invalid timer reference.
Header	#include "simcom_os.h"

Examples

```
sTimerRef timerRef;
sTimerStatus timerStatus;
SC_STATUS status;
status = sAPI_TimerGetStatus(timerRef, &timerStatus);
```

2.2.34 sAPI_GetTicks

This function requests the elapsed time, in OS clock tick, since the last system start-up.

sAPI_GetTicks	
Prototype	UINT32 **sAPI_GetTicks**(void);
Parameters	None
Return value	Time elapsed in OS clock ticks
Header	#include "simcom_os.h"

Examples

```
UINT32 ticks;
ticks = sAPI_GetTicks();
```

2.2.35 sAPI_Gettimeofday

This function can get time, and gives the number of seconds and microseconds since the Epoch.

sAPI_Gettimeofday	
Prototype	INT32 **sAPI_Gettimeofday**(sTimeval *tv,void* dummy);
Parameters	[out] **tv**: gives the number of seconds and microseconds since the Epoch [out] **dummy**: reserved parameters
Return value	return 0 for success, or -1 for failure
Header	#include "simcom_os.h"

Examples

```
INT32 ret;
sTimeval tv={0,0};
ret = sAPI_Getimeofday(&tv,NULL);
```

2.2.36 sAPI_Time

This function returns the time as the number of seconds since the Epoch, 1970-01-01 00:00:00 +0000 (UTC).

sAPI_Time	
Prototype	unsigned long **sAPI_Time**(unsigned long *t);
Parameters	[out] **t**: If t is non-NULL, the return value is also stored in the memory pointed to by t
Return value	the value of time in seconds since the Epoch is returned.
Header	#include "simcom_os.h"

Examples

```
unsigned long t;
t = sAPI_Time(NULL);
```

2.2.37 sAPI_GetSystemInfo

This function output cpu usage rate and remaining heap space size .

sAPI_GetSystemInfo	
Prototype	Void sAPI_GetSystemInfo(unsignedint* cpuUsedRate, unsigned int *heapFree Size)
Parameters	[in] **cpuUsedRate**: Used to save cpu used rate [in] **heapFreeSize**: Used to save remaining heap size
Return value	None
Header	#include "simcom_system.h"

Examples

```
UINT32 cpuUsedRate, heapFreeSize;
sAPI_GetSystemInfo(&cpuUsedRate,&heapFreeSize)
```

2.2.38 sAPI_ContextLock

This function is used to lock context - No interrupts and no preemptions.

sAPI_ContextLock	
Prototype	SC_STATUS **sAPI_ContextLock**(void);
Parameters	None
Return value	SC_SUCCESS: Successful completion of the service. other: OS specific error.
Header	#include "simcom_os.h"

Examples

```
sAPI_ContextLock();
```

2.2.39 sAPI_ContextUnlock

This function is used to restore the context.

sAPI_ContextUnlock	
Prototype	SC_STATUS **sAPI_ContextUnlock**(void);
Parameters	None
Return value	SC_SUCCESS: Successful completion of the service. other: OS specific error.
Header	#include "simcom_os.h"

Examples

sAPI_ContextUnlock ();

2.2.40 sAPI_GettimeofdaySyncRtc

This function can get time, and gives the number of seconds and microseconds since the Epoch. After the RTC time changes, the obtained time will change synchronously.

sAPI_GettimeofdaySyncRtc	
Prototype	INT32 **sAPI_GettimeofdaySyncRtc**(sTimeval *tv,void* dummy);
Parameters	[out] **tv**: gives the number of seconds and microseconds since the Epoch [out] **dummy**: reserved parameter,
Return value	return 0 for success, or -1 for failure
Header	#include "simcom_os.h"

Examples

```
INT32 ret;
sTimeval tv={0,0};
ret = sAPI_GetimeofdaySyncRtc(&tv,NULL);

;
```

3 APIs for Network

3.1 Overview of APIs for Network

Interface	Description
sAPI_NetworkGetCsq	Get signal value
sAPI_NetworkGetCreg	Gets the value of creg :whether the network has currently indicated the registration of the ME
sAPI_NetworkGetCgreg	Gets the value of cgreg: Whether the network has currently indicated the registration of the MT
sAPI_NetworkGetCpsi	Gets the the UE system information
sAPI_NetworkGetCnmp	Gets the state of the mode preference
sAPI_NetworkSetCnmp	Sets the state of the mode preference
sAPI_NetworkGetCops	Gets the current mode and the currently selected operator
sAPI_NetworkSetCops	Forces an attempt to select and register the GSM/UMTS network
sAPI_NetworkGetCgdcont	Get PDP context parameter values for a PDP context
sAPI_NetworkSetCgdcont	Set PDP context parameter values for a PDP context
sAPI_NetworkGetCgact	Get state of PDP context
sAPI_NetworkSetCgact	Set state of PDP context
sAPI_NetworkGetCgatt	Get the current Packet Domain service state
sAPI_NetworkSetCgatt	Set the current Packet Domain service state
sAPI_NetworkSetCfun	This command is used to select the level of functionality in the ME
sAPI_NetworkGetCfun	Query the value of CFUN
sAPI_NetworkSetCtzu	Automatic time and time zone update
sAPI_NetworkGetCgpaddr	Get IP address
sAPI_NetworkGetCnetci	Get adjacent base station information
sAPI_NetworkGetCgauth	Get type of authentication for PDP-IP connections of GPRS
sAPI_NetworkSetCgauth	Set type of authentication for PDP-IP connections of GPRS

3.2 Detailed Description of APIs for Network

3.2.1 sAPI_NetworkGetCsq

This application interface is to get the current signal value.

sAPI_NetworkGetCsq	
Prototype	unsigned int **sAPI_NetworkGetCsq**(UINT8 *pCsq);
Parameters	[out] **pCsq:** signal strength indication 0: 113 dBm or less 1: 111 dBm 2...30: 109... -53 dBm 31: 51 dBm or greater 99: not known or not detectable
Return value	SC_NET_SUCCESS: Get csq success. SC_NET_FAIL: Get csq falied.
Header	#include "simcom_network.h"

Examples

```
#include "simcom_network.h"

unsigned int NetWork_Test(void);
{
    ret = sAPI_NetworkGetCsq(&csq);
    if(ret == 0);
        sAPI_Debug("Get csq success. csq=%d!", csq);
    else
        sAPI_Debug("Get csq falied!");
}
```

3.2.2 sAPI_NetworkGetCreg

This application interface is used to show whether the network has currently indicated the registration of the ME.

sAPI_NetworkGetCreg	
Prototype	unsigned int **sAPI_NetworkGetCreg**(int* pReg);
Parameters	[out] **pReg**: 0: not registered, ME is not currently searching a new operator to register to. 1: registered, home network. 2: not registered, but ME is currently searching a new operator to register to.

	3: registration denied. 4: unknown. 5: registered, roaming. 11: only emergency services are available.
Return value	SC_NET_SUCCESS: Get creg success. SC_NET_FAIL: Get creg falied.
Header	#include "simcom_network.h"

Examples

```
#include "simcom_network.h"

unsigned int NetWork_Test(void);
{
    sAPI_Debug("Get creg !");
    ret = sAPI_NetworkGetCreg(&creg);
    if(ret == 0);
        sAPI_Debug("Get creg success. creg=%d!", creg);
    else
        sAPI_Debug("Get creg falied!");
}
```

3.2.3 sAPI_NetworkGetCgreg

This application interface is used to get the MT's GPRS network registration status.

sAPI_NetworkGetCgreg	
Prototype	unsigned int **sAPI_NetworkGetCgreg**(int* pGreg);
Parameters	[out] **pGreg:** 0: not registered, ME is not currently searching an operator to register to 1: registered, home network 2: not registered, but ME is currently trying to attach or searching an operator to register to 3: registration denied 4: unknown 5: registered, roaming 11: attached for emergency bearer services only
Return value	SC_NET_SUCCESS: get the cgreg value success. SC_NET_FAIL: get the cgreg value failure.
Header	#include "simcom_network.h"

Examples

```
#include "simcom_network.h"

unsigned int NetWork_Test(void);
{
    ret = sAPI_NetworkGetCgreg(&cgreg);
    if(ret == 0);
        sAPI_Debug("Get cgreg success. cgreg=%d!", cgreg);
    else
        sAPI_Debug("Get cgreg falied!");
}
```

3.2.4 sAPI_NetworkGetCpsi

This application interface is used to get the UE system information.

sAPI_NetworkGetCpsi	
Prototype	unsigned int **sAPI_NetworkGetCpsi**(SCcpsiParm *pStr);
Parameters	[out] **pStr**: 1) If camping on a gsm cell: Networkmode: System mode, values: "NO SERVICE", "GSM", "LTE" Mnc_Mcc: Mobile Country Code and Mobile Network Code LAC: Location Area Code CellID: Service-cell Identify GSMBandStr: Frequency Band of active set 2) If camping on a lte cell: Networkmode: System mode, values: "NO SERVICE", "GSM", "LTE" Mnc_Mcc: Mobile Country Code and Mobile Network Code TAC: Tracing Area Code CellID: Service-cell Identify LTEBandStr: Frequency Band of active set 3) If no service: +CPSI: NO SERVICE, Online
Return value	SC_NET_SUCCESS: get UE system information success. SC_NET_FAIL: get UE system information fail.
Header	#include "simcom_network.h"

Examples

```
#include "simcom_network.h"

unsigned int NetWork_Test(void);
{
        SCcpsiParm cpsi;
        ret = sAPI_NetworkGetCpsi(&Scpsi);
        if(ret == SC_NET_SUCCESS);
            sAPI_Debug("Get cpsi success. NEmode=%s, MM=%s, LAC=%d, CELL=%d, Gband=%s,
            Lband=%s, TAC=%d!", Scpsi.networkmode, Scpsi.Mnc_Mcc, Scpsi.LAC, Scpsi.CellID,
            Scpsi.GSMBandStr, Scpsi.LTEBandStr, Scpsi.TAC);
        else
            sAPI_Debug("Get cpsi falied!");
}
```

3.2.5 sAPI_NetworkGetCnmp

This application interface is used to select the state of the mode preference.

sAPI_NetworkGetCnmp	
Prototype	unsigned int **sAPI_NetworkGetCnmp**(int *pCnmp);
Parameters	[out] **pCnmp:** 2: Automatic 13: GSM Only 38: LTE Only
Return value	SC_NET_SUCCESS: get the state of the mode preference success. SC_NET_FAIL: get the state of the mode preference fail.
Header	#include "simcom_network.h"

Examples

```
#include "simcom_network.h"

unsigned int NetWork_Test(void);
{
      ret = sAPI_NetworkGetCnmp(&cnmp);
      if(ret == 0);
            sAPI_Debug("Get cnmp success. cnmp=%d!", cnmp);
      else
            sAPI_Debug("Get cnmp falied!");
}
```

3.2.6 sAPI_NetworkSetCnmp

This application interface is used to set the state of the mode preference.

sAPI_NetworkSetCnmp	
Prototype	unsigned int **sAPI_NetworkSetCnmp**(int CnmpValue);
Parameters	[int] **CnmpValue**: state of the mode preference. 2: Automatic 13: GSM Only 38: LTE Only
Return value	SC_NET_SUCCESS: set the state of the mode preference success. SC_NET_FAIL: set the state of the mode preference fail.
Header	#include "simcom_network.h"

Examples

```
#include "simcom_network.h"

unsigned int NetWork_Test(void);
{
    ret = sAPI_NetworkGetCnmp(2);
    if(ret == 0);
        sAPI_Debug("Set cnmp success. cnmp!");
    else
        sAPI_Debug("Set cnmp falied!");
}
```

3.2.7 sAPI_NetworkGetCops

This application interface is used to current mode and the currently selected operator.

sAPI_NetworkGetCops	
Prototype	unsigned int **sAPI_NetworkGetCops**(char* pCops);
Parameters	[out] **pCops:** +COPS: <mode>[, <format>, <oper>[, <AcT>]]
Return value	SC_NET_SUCCESS: get the current mode success. SC_NET_FAIL: get the current mode fail.
Header	#include "simcom_network.h"

Defined Values

<mode>	0 automatic 1 manual 2 force deregister 3 set only <format> 4 manual/automatic **NOTE:** if <mode> is set to 1, 4 in write command, the <oper> is needed.
<format>	0 long format alphanumeric <oper> 1 short format alphanumeric <oper> 2 numeric <oper>
<oper>	string type, <format> indicates if the format is alphanumeric or numeric.
<AcT>	Access technology selected 0 GSM 1 GSM Compact 2 UTRAN 3 GSM w/EGPRS 4 UTRAN w/HSDPA 5 UTRAN w/HSUPA 6 UTRAN w/HSDPA and HSUPA 7 EUTRAN 8 UTRAN HSPA+

Examples

```
#include "simcom_network.h"

unsigned int NetWork_Test(void);
{
    ret = sAPI_NetworkGetCops(cops);
    if(ret == 0);
        sAPI_Debug("Get cops success. cops=%s!", cops);
    else
        sAPI_Debug("Get cops falied:%d!", ret);
}
```

3.2.8 sAPI_NetworkSetCops

This application interface is used to select and register the GSM/UMTS network operator.

sAPI_NetworkSetCops	
Prototype	unsigned int **sAPI_NetworkSetCops**(int modeVal, int formatVal, char *networkOperator, int accTchVal);
Parameters	[int] **modeVal**: 0: automatic 1: manual 2: force deregister 3: set only <format> 4: manual/automatic **NOTE:** if <mode> is set to 1, 4 in write command, the networkOperator is needed. [int] **formatVal:** 0: long format alphanumeric networkOperator 1: short format alphanumeric networkOperator 2: numeric networkOperator [int] **networkOperator:** string type [int] **accTchVal** Access technology selected 0: GSM 1: GSM Compact 2: UTRAN 3: GSM w/EGPRS 4: UTRAN w/HSDPA 5: UTRAN w/HSUPA 6: UTRAN w/HSDPA and HSUPA 7: EUTRAN 8: UTRAN HSPA+
Return value	SC_NET_SUCCESS:select and register the network operator success. SC_NET_FAIL:select and register the network operator fail.
Header	#include "simcom_network.h"

Examples

```
#include "simcom_network.h"

unsigned int NetWork_Test(void);
{
    int ret = CIRC_FAIL;

    ret = sAPI_NetworkSetCops(0, 2, "46001", 7);
    return ret;
}
```

3.2.9 sAPI_NetworkGetCgdcont

This application interface is used to get PDP context parameter values for a PDP context.

sAPI_NetworkGetCgdcont	
Prototype	unsigned int **sAPI_NetworkGetCgdcont**(SCApnParm * pCgdcont);
Parameters	[out] **pCgdcont:** typedef struct{ UINT8 cid; UINT8 iptype; char ApnStr[40]; }SCApnParm;
Return value	SC_NET_SUCCESS:get PDP context parameter values for a PDP context success. SC_NET_FAIL:get PDP context parameter values for a PDP context fail.
Header	#include "simcom_network.h"

Defined Values

<cid>	(PDP Context Identifier); a numeric parameter which specifies a particular PDP context definition. The parameter is local to the TE-MT interface and is used in other PDP context-related commands. The range of permitted values(minimum value = 1); is returned by the test form of the command. 1…15
<PDP_type>	(Packet Data Protocol type): a string parameter which specifies the type of packet data protocol. IP Internet Protocol PPP Point to Point Protocol IPV6 Internet Protocol Version 6 IPV4V6 Dual PDN Stack
<APN>	(Access Point Name); a string parameter which is a logical name that is used to select the GGSN or the external packet data network.
<PDP_addr>	A string parameter that identifies the MT in the address space applicable to the PDP. This parameter will be omitted when PDP_type is PPP type.

Examples

```
#include "simcom_network.h"
```

```
unsigned int NetWork_Test(void);
{
        ret = sAPI_NetworkGetCgdcont(cgdcont);
        if(ret == 0);
            sAPI_Debug("Get Cgdcont success. Cgdcont=%s!", cgdcont);
        else
            sAPI_Debug("Get Cgdcont falied:%d!", ret);
}
```

3.2.10 sAPI_NetworkSetCgdcont

This application interface is used to set PDP context parameter values for a PDP context.

sAPI_NetworkSetCgdcont	
Prototype	unsigned int **sAPI_NetworkSetCgdcont**(int primCid, char *type, char *APNstr);
Parameters	[int] **primCid**: 1-15 a numeric parameter which specifies a particular PDP context definition. The parameter is local to the TE-MT interface and is used in other PDP context-related commands. The range of permitted values(minimum value = 1); is returned by the test form of the command. [int] **type:** a string parameter which specifies the type of packet data protocol. IP Internet Protocol PPP Point to Point Protocol IPV6 Internet Protocol Version 6 [int] **APNstr:** a string parameter which is a logical name that is used to select the GGSN or the external packet data network
Return value	SC_NET_SUCCESS:set PDP context parameter values for a PDP context success. SC_NET_FAIL:set PDP context parameter values for a PDP context fail.
Header	#include "simcom_network.h"

Examples

```
#include "simcom_network.h"

unsigned int NetWork_Test(void);
{
    int ret = CIRC_FAIL;
```

```
    ret = sAPI_NetworkSetCgdcont(2, "IP", "cmnet");
    return ret;
}
```

3.2.11 sAPI_NetworkGetCgact

This application interface is used to activate or deactivate the specified PDP context.

sAPI_NetworkGetCgact	
Prototype	unsigned int **sAPI_NetworkGetCgact**(SCAPNact * pCgact);
Parameters	[out] **pCgact:** typedef struct{ UINT8 cid; UINT8 isActived; UINT8 isdefine; }SCAPNact;
Return value	SC_NET_SUCCESS:get the state of the specified PDP context success. SC_NET_FAIL:get the state of the specified PDP context fail.
Header	#include "simcom_network.h"

Defined Values

<state>	Indicates the state of PDP context activation: 0 deactivated 1 activated
<cid>	A numeric parameter which specifies a particular PDP context definition. 1…15

Examples

```
#include "simcom_network.h"

unsigned int NetWork_Test(void);
{
    ret = sAPI_NetworkGetCgact(cgact);
    if(ret == 0);
        sAPI_Debug("Get Cgact success. Cgact=%s!", cgact);
    else
        sAPI_Debug("Get Cgact falied!");
```

```
}
```

3.2.12 sAPI_NetworkSetCgact

This application interface is used to set the status of PDP context.

sAPI_NetworkSetCgact	
Prototype	unsigned int **sAPI_NetworkSetCgact**(int pdpState, int cid);
Parameters	[int] **pdpState**: Indicates the state of PDP context activation 0: deactivated 1: activated [int] **cid: 1-15** A numeric parameter which specifies a particular PDP context definition
Return value	SC_NET_SUCCESS: set the state of the specified PDP context success. SC_NET_FAIL: set the state of the specified PDP context fail.
Header	#include "simcom_network.h"

Examples

```
#include "simcom_network.h"

unsigned int NetWork_Test(void);
{
    int ret = CIRC_FAIL;

    ret = sAPI_NetworkSetCgact(1, 2);       //Activate the second PDP context
    return ret;
}
```

3.2.13 sAPI_NetworkSetCgatt

This application interface is used to attach the MT to, or detach the MT from, the Packet Domain service.

sAPI_NetworkSetCgatt	
Prototype	unsigned int **sAPI_NetworkSetCgatt**(int CgattValue);
Parameters	[int] **CgattValue**: Indicates the state of Packet Domain attachment 0: detached

	1: attached
Return value	SC_NET_SUCCESS:set the state of the mode preference success. SC_NET_FAIL:set the state of the mode preference fail.
Header	#include "simcom_network.h"

Examples

```
#include "simcom_network.h"

unsigned int NetWork_Test(void);
{
    int ret = CIRC_FAIL;

    ret = sAPI_NetworkSetCgatt(1);
    return ret;
}
```

3.2.14 sAPI_NetworkGetCgatt

This application interface is used to get the current Packet Domain service state.

sAPI_NetworkGetCgatt	
Prototype	unsigned int **sAPI_NetworkGetCgatt**(int* pCgatt);
Parameters	[out] **pCgatt**: Indicates the state of Packet Domain attachment 0: detached 1: attached
Return value	SC_NET_SUCCESS:get the state of the mode preference success. SC_NET_FAIL:get the state of the mode preference fail.
Header	#include "simcom_network.h"

Examples

```
#include "simcom_network.h"

unsigned int NetWork_Test(void);
{
    ret = sAPI_NetworkGetCgatt(&cgatt);
    if(ret == 0);
        sAPI_Debug("Get Cgatt success. Cgatt=%d!", cgatt);
    else
```

```
        sAPI_Debug("Get Cgatt falied!");
}
```

3.2.15 sAPI_NetworkSetCfun

This application interface is used to select the level of functionality in the ME.

sAPI_NetworkSetCfun	
Prototype	unsigned int **sAPI_NetworkSetCfun**(int CfunValue);
Parameters	[int] **CfunValue**: 0: minimum functionality 1: full functionality, online mode 4: disable phone both transmit and receive RF circuits
Return value	SC_NET_SUCCESS:select the level of functionality in the ME success. SC_NET_FAIL:select the level of functionality in the ME fail.
Header	#include "simcom_network.h"

Examples

```
#include "simcom_network.h"

unsigned int NetWork_Test(void);
{
    int ret = CIRC_FAIL;

    ret = sAPI_NetworkSetCfun(0);
    return ret;
}
```

3.2.16 sAPI_NetworkGetCfun

This application interface is used to get the value of CFUN.

sAPI_NetworkGetCfun	
Prototype	unsigned int **sAPI_NetworkGetCfun**(UINT8 *pCfun);
Parameters	[out] **pCfun**: the value of CFUN
Return value	SC_NET_SUCCESS:Get the value of CFUN success. SC_NET_FAIL:Get the value of CFUN fail.

Header	#include "simcom_network.h"

Examples

```
#include "simcom_network.h"

unsigned int NetWork_Test(void);
{
    ret = sAPI_NetworkGetCfun(&cfun);
    if(ret == SC_NET_SUCCESS);
    {
        sAPI_Debug("Get cfun success. cfun=%d!", cfun);
    }
    else
    {
        sAPI_Debug("Get cfun falied!");
    }
}
```

3.2.17 sAPI_NetworkSetCtzu

This application interface is used for automatic time and time zone update.

sAPI_NetworkSetCtzu	
Prototype	unsigned int **sAPI_NetworkSetCtzu**(int CtzuValue);
Parameters	[int] **CtzuValue**: 0: Disable automatic time and time zone update via NITZ 1: Enable automatic time and time zone update via NITZ
Return value	SC_NET_SUCCESS: Set automatic update successful SC_NET_FAIL: Set automatic update failed.
Header	#include "simcom_network.h"

3.2.18 sAPI_NetworkGetCgpaddr

This application interface is used to PDP addresses for the specified context identifiers.

sAPI_NetworkGetCgpaddr	
Prototype	unsigned int **sAPI_NetworkGetCgpaddr**(int Cid,SCcgpaddrParm *Pstr);

Parameters	[in] **Cid**: PDP cid [out] PDP addresses
Return value	SC_NET_SUCCESS: Get cgpaddr success. SC_NET_FAIL: Get cgpaddr falied.
Header	#include "simcom_network.h"

Examples

```
#include "simcom_network.h"

unsigned int NetWork_Test(void);
{
    sAPI_Debug("Get creg !");
    ret = sAPI_NetworkGetCgpaddr(opt,&cgpaddrParm);
    if(ret == SC_NET_SUCCESS)
    {
            sAPI_Debug("Get Ipaddr success. cid=%d,type=%d,ipv4=%s,ipv6=%s!",cgpaddrParm.cid,
            cgpaddrParm.iptype,cgpaddrParm.ipv4addr,cgpaddrParm.ipv6addr);
            sprintf(NetResp,"\r\nGet Ipaddr success. cid=%d,type=%d,ipv4=%s,ipv6=%s!",cgpaddrPar
            m.cid,cgpaddrParm.iptype,cgpaddrParm.ipv4addr,cgpaddrParm.ipv6addr);
            PrintfResp(NetResp);
            break;
    }
    else
    {
            sAPI_Debug("Get Ipaddr falied!");
            PrintfResp("\r\nGet Ipaddr falied!\r\n");
            break;
    }
}
```

3.2.19 sAPI_NetworkGetCnetci

This application interface is used to get adjacent base station information.

sAPI_NetworkGetCpsi	
Prototype	unsigned int **sAPI_NetworkGetCnetci(**SCcnetciParm *pStr);
Parameters	[out] **pStr**: Mnc_Mcc: Mobile Country Code and Mobile Network Code

	TAC: Tracing Area Code CellID: Service-cell Identify RSRP: Reference signal received power RSRQ: Reference signal received quality RXSIGLEVEL: Receive signal level
Return value	SC_NET_SUCCESS: get adjacent base station information success. SC_NET_FAIL: get adjacent base station information fail.
Header	#include "simcom_network.h"

Examples

```
#include "simcom_network.h"

unsigned int NetWork_Test(void);
{
    SCcnetciParm Snetci[12];
    ret = sAPI_NetworkGetcnetci(Snetci);
    if(ret == SC_NET_SUCCESS)
    {
        int i = 0;
        for(i = 0;i < 12;i++)/*Read it ten times*/
        {
            sAPI_Debug("Get cnetci
        success.i=%d:MM=%s,TAC=%d,CELL=%d,RSRP=%d,RSRQ=%d,RXSIGLEVEL=%d",i,Snetci[i].
        Mnc_Mcc,Snetci[i].TAC,Snetci[i].CellID,Snetci[i].Rsrp,Snetci[i].Rsrq,Snetci[i].RXSIGLEVEL);
        }
    }
    else
        sAPI_Debug("Get cnetci falied!");
}
```

3.2.20 sAPI_NetworkGetCgauth

Get type of authentication for PDP-IP connections of GPRS.

sAPI_NetworkGetCgauth	
Prototype	unsigned int sAPI_NetworkGetCgauth(SCCGAUTHParm *pCgauth,int cid)
Parameters	[out] **pStr**: pCgauth: Cgauth parm cid: for APN
Return value	SC_NET_SUCCESS: get adjacent base station information success.

	SC_NET_FAIL: get adjacent base station information fail.
Header	#include "simcom_network.h"

Examples

```
#include "simcom_network.h"

unsigned int NetWork_Test(void);
{
        SCCGAUTHParm cgauthParm = {0,0,{0},{0}};
        ret = sAPI_NetworkGetCGAUTH(&cgauthParm,cid);
        if(SC_NET_SUCCESS == ret)
        {
            sprintf(cgauthresp,"\r\ncid=%d,type=%d,usr=%s,passwd=%s
            \r\n",cgauthParm.cid,cgauthParm.authtype,cgauthParm.user,cgauthParm.passwd);
                        PrintfResp(cgauthresp);
        }
        else
        {
            sprintf(cgauthresp,"\r\n GET FAIL \r\n");
            PrintfResp(cgauthresp);
        }
}
```

3.2.21 sAPI_NetworkSetCgauth

Set type of authentication for PDP-IP connections of GPRS.

sAPI_NetworkSetCgauth	
Prototype	unsigned int sAPI_NetworkSetCgauth(SCCGAUTHParm *pCgauth,BOOL delflag)
Parameters	[out] **pStr**: pCgauth: Cgauth parm delflag: 1:delete 0:set
Return value	SC_NET_SUCCESS: get adjacent base station information success. SC_NET_FAIL: get adjacent base station information fail.
Header	#include "simcom_network.h"

Examples

```
#include "simcom_network.h"

unsigned int NetWork_Test(void);
{
    auth.cid = 1;
    auth.authtype = 2;
    sprintf(auth.user, "simcom");
    sprintf(auth.passwd, "simcom");
    ret = sAPI_NetworkSetCGAUTH(&auth,0);
}
```

4 APIs for SIM Card

4.1 Overview of APIs for SIM Card

Interface	Description
sAPI_SimcardPinSet	Set the pin and puk of simcard
sAPI_SimcardHotSwapMsg	Enable hot-swap or set the trigger mode of hot-swap
sAPI_SimcardPinGet	Get state of simcard
sAPI_SysGetImsi	Get the imsi
sAPI_SysGetHplmn	Get the Hplmn
sAPI_SimcardSwitchMsg	Switch the SIM1 or SIM2
sAPI_SysGetIccid	Get the iccid

4.2 Detailed Description of APIs for SIM Card

4.2.1 sAPI_SimcardPinSet

This API is used to set the pin and puk of simcard

sAPI_SimcardPinSet	
Prototype	SC_simcard_err_e **sAPI_SimcardPinSet**(char * oldpin, char * newpin, int new);
Parameters	[in] **oldpin:** old pin string, lenth is more than 4 [in] **newpin:**new pin string, lenth is more than 4 [in] **new:** need new pin or not 1:need, 2:no need.
Return value	SC_SIM_RETURN _SUCCESS: set the pin of simcard success SC_SIM_RETURN _FAIL: set the pin of simcard fail
Header	#include "simcom_simcard.h"

Examples

```
#include "simcom_simcard.h"
```

```
unsigned int SimCardApiTestFunc (void)
{
    SC_simcard_err_e ret;
    char oldpin[20] ={"1234"};
    char newpin[20] ={"4321"};
    int new = 1;
    ret = sAPI_SimcardPinSet(oldpin, newpin, new);
    if(ret == SC_ SIM_ RETURN _SUCCESS);
    {
        ……//The action after success
    }
    else if(ret == SC_ SIM_ RETURN_FAIL);
    {
        ……//The action after fail
    }
    return 0;
}
```

4.2.2 sAPI_SimcardHotSwapMsg

This API is used to enable hot-swap or set the trigger mode of hot-swap.

sAPI_ SimcardHotSwapMsg()	
Prototype	SC_simcard_err_e **sAPI_SimcardHotSwapMsg** (SC_HotSwapCmdType_e opt, UINT8 param, sMsgQRef msgQ);
Parameters	[in] **opt:** SC_HOTSWAP_QUERY_STATE:Query the hot-swap enable state; SC_HOTSWAP_QUERY_LEVEL: Query the hot-swap trigger mode; SC_HOTSWAP_SET_SWITCH:Set the hot-swap enable switch; SC_HOTSWAP_SET_LEVEL: Set the hot-swap trigger mode; [in] **param:** when **opt** is SC_HOTSWAP_SET_SWITCH,1: enable hot-swap function 0: disabled hot-swap function;when **opt** is SC_HOTSWAP_SET_LEVEL, 1:high level trigger mode 0:Low level trigger mode;**opt** is any other,this param value is invalid [in] **msgQ:** Results will be returned by msgQ
Return value	SC_SIM_RETURN _SUCCESS: execute success SC_SIM_RETURN _FAIL: execute fail
Header	#include "simcom_simcard.h"

Examples

```
#include "simcom_simcard.h"
```

```
unsigned int SimCardApiTestFunc(void)
{
   SC_simcard_err_e ret = SC_SIM_RTEURN_UNKNOW;
   SC_HotSwapCmdType_e opt = SC_HOTSWAP_QUERY_STATE;
   sMsgQRef simcard_test_msgQ;
   ret = sAPI_SimcardHotSwapMsg(opt,0, simcard_test_msgQ);
   if(SC_ SIM_ RETURN _SUCCESS == ret);
   {
      ……//The action after success
   }
   else if(SC_ SIM_ RETURN_FAIL == ret);
   {
      ……//The action after fail
   }
   return 0;
}
```

4.2.3 sAPI_SimcardPinGet

This API is used to get the state of simcard

sAPI_SimcardPinGet()	
Prototype	SC_simcard_err_e **sAPI_SimcardPinGet**(UINT8 *gcpin);

Parameters	[out] **gcpin:**
	enum SCsimcardstate
	{
	SC_SIM_READY = 0,
	SC_SIM_PIN,
	SC_SIM_PUK,
	SC_SIM_BLOCKED,
	SC_SIM_REMOVED,
	SC_SIM_CRASH,
	SC_SIM_NOINSRETED,
	SC_SIM_UNKNOW
	};

Return value	SC_SIM_RETURN _SUCCESS: execute success SC_SIM_RETURN _FAIL: execute fail
Header	#include "simcom_simcard.h"

Examples

```
#include "simcom_simcard.h"

unsigned int NetWork_Test(void);
{
        ret = sAPI_SimcardPinGet(cpin);
        if(ret == SC_SIM_RETURN _SUCCESS);
            sAPI_Debug("Get Cpin success. Cpin=%s!", cpin);
        else
            sAPI_Debug("Get Cpin falied!");
      return 0;
}
```

4.2.4 sAPI_SysGetImsi

This API is used to get the imsi.

sAPI_SysGetImsi()	
Prototype	SC_simcard_err_e **sAPI_SysGetImsi**(char *ImsiValue);
Parameters	[out] **ImsiValue:** the Imsi from SIM card
Return value	SC_SIM_RETURN _SUCCESS: execute success SC_SIM_RETURN _FAIL: execute fail
Header	#include "simcom_simcard.h"

Examples

```
#include "simcom_ simcard.h"

unsigned int NetWork_Test(void);
{
        ret = sAPI_SysGetImsi(imsi);
        if(ret == SC_SIM_RETURN _SUCCESS);
            sAPI_Debug("Get iccid success. imsi=%s ", imsi);
        else
            sAPI_Debug("Get imsi falied!");
}
```

4.2.5 sAPI_SysGetHplmn

This API is used to get the Hplmn.

sAPI_SysGetHplmn()	
Prototype	SC_simcard_err_e **sAPI_SysGetHplmn**(char *HplmnValue);
Parameters	[out] **HplmnValue:** the HPLMN from SIM card
Return value	SC_SIM_RETURN _SUCCESS: execute success SC_SIM_RETURN _FAIL: execute fail.
Header	#include "simcom_simcard.h"

Examples

```
#include "simcom_ simcard.h"

unsigned int NetWork_Test(void);
{
        ret = sAPI_SysGetHplmn(Hplmn);
        if(ret == SC_SIM_RETURN _SUCCESS);
            sAPI_Debug("Get iccid success. iccid=%s ", Hplmn);
        else
            sAPI_Debug("Get hplmn falied!");
        return 0;
}
```

4.2.6 sAPI_SimcardSwitchMsg

This API is used to switch SIM1 or SIM2.

sAPI_SimcardSwitchMsg()	
Prototype	SC_simcard_err_e **sAPI_SimcardSwitchMsg**(UINT8 status, sMsgQRef msgQ);
Parameters	[in] **status:** 0/SIM1 1/SIM2 [in] **msgQ:** Results will be returned by msgQ
Return value	SC_SIM_RETURN_SUCCESS switch successful SC_SIM_RETURN_FAIL switch failed
Header	#include "simcom_simcard.h"

Examples

```
#include "simcom_simcard.h"

unsigned int SimCardApiTestFunc(void)
{
   SC_simcard_err_e ret = SC_SIM_RTEURN_UNKNOW;
   UINT8 TestStatus = 1;
   sMsgQRef simcard_test_msgQ;
   ret = sAPI_SimcardSwitchMsg(TestStatus, simcard_test_msgQ);
   if(SC_ SIM_ RETURN _SUCCESS == ret);
   {
      ……//The action after success
   }
   else if(SC_ SIM_ RETURN_FAIL == ret);
   {
      ……//The action after fail
   }
   return 0;
}
```

4.2.7 sAPI_SysGetIccid

This application interface is to get the Iccid.

sAPI_SysGetIccid	
Prototype	SC_simcard_err_e **sAPI_SysGetIccid**(char *IccidValue);
Parameters	[out] **IccidValue:** The value of the ICCID
Return value	SC_SIM_RETURN_SUCCESS switch successful SC_SIM_RETURN_FAIL switch failed
Header	#include "simcom_simcard.h"

Examples

```
#include "simcom_simcard.h"

unsigned int version_Test(void);
{
      char iccid[32] = {0};
      SC_simcard_err_e ret = sAPI_SysGetIccid(iccid);
      if(ret == SC_SIM_RETURN_SUCCESS);
          sAPI_Debug("Get iccid success. iccid=%s ", iccid);
```

```
    else
        sAPI_Debug("Get iccid falied!");
    return ret;
}
```

5 APIs for Call

5.1 Overview of APIs for Call

Interface	Description
sAPI_CallDialMsg	Dial a call
sAPI_CallAnswerMsg	Answer a call
sAPI_CallEndMsg	End a call
sAPI_CallStateMsg	Get call state
sAPI_CallAutoAnswer	Set auto answer

5.2 Detailed Description of APIs for Call

5.2.1 sAPI_CallDialMsg

This function is used to dial a call.

sAPI_CallDialMsg	
Prototype	SC_CALLReturnCode **sAPI_CallDialMsg**(UINT8* dialstring, sMsgQRef msgQ);
Parameters	[in] **dialstring**: The number you want to dial [in] **msgQ**: Results will be returned by msgQ
Return value	SC_CALL_SUCESS: dail sucessfull -: dail unsucessfull
Header	#include "simcom_call.h"

Examples

Refer to demo of APIs for Call

5.2.2 sAPI_CallAnswerMsg

This function is used to answer a call

sAPI_CallAnswerMsg	
Prototype	SC_CALLReturnCode **sAPI_CallAnswerMsg**(sMsgQRef msgQ);
Parameters	[in] **msgQ**: Results will be returned by msgQ
Return value	SC_CALL_SUCESS: answer sucessfull -: answer unsucessfull
Header	#include "simcom_call.h"

Examples

Refer to demo of APIs for Call

5.2.3 sAPI_CallEndMsg

This function is used to end a call

sAPI_CallEndMsg	
Prototype	SC_CALLReturnCode **sAPI_CallEndMsg**(sMsgQRef msgQ);
Parameters	[in] **msgQ**: Results will be returned by msgQ
Return value	SC_CALL_SUCESS: end sucessfull -: end unsucessfull
Header	#include "simcom_call.h"

Examples

Refer to demo of APIs for Call

5.2.4 sAPI_CallStateMsg

This function is used to get call state.

sAPI_SmsSendStorageMsg	
Prototype	UINT8 **sAPI_CallStateMsg**(void);

Parameters	NULL
Return value	CICC_CURRENT_CSTATE: -: 0 CICC_CURRENT_CSTATE_ACTIVE -: 3 CICC_CURRENT_CSTATE_ALERTING -: 4 CICC_CURRENT_CSTATE_INCOMING -: 9 CICC_NUM_CURRENT_CSTATES
Header	#include "simcom_call.h"

Examples

Refer to demo of APIs for Call

5.2.5 sAPI_CallAutoAnswer

This function is used to set auto answer a call.

sAPI_CallDialMsg	
Prototype	SC_CALLReturnCode **sAPI_CallAutoanswer**(INT32 seconds, sMsgQRef msgQ);
Parameters	[in] **seconds**: The seconds of auto answer [in] **msgQ**: Results will be returned by msgQ
Return value	SC_CALL_SUCESS: set sucessfull -: set unsucessfull
Header	#include "simcom_call.h"

Examples

Refer to demo of APIs for Call

5.3 Demo for Call

```
#include "simcom_call.h"

#include "simcom_debug.h"

sMsgQRef g_call_msgq;
```

```
int call_testMain(void);
{
    SC_CALLReturnCode ret;
    Int state;
    SC_STATUS status = OS_FAIL;
    SIM_MSG_T msg;

    status = sAPI_MsgQCreate(&g_call_msgq, "g_call_msgq", sizeof(SIM_MSG_T), 4, SC_FIFO);

    ret = sAPI_CallDialMsg(10086, g_call_msgQ);
    if(ret == SC_CALL_SUCESS);
    {
        memset(&msg, 0, sizeof(msg););
        status = sAPI_MsgQRecv(g_call_msgq, &msg, SC_SUSPEND);
        sAPI_Debug("rc[%d] ", msg.arg2);
    }
    else
    {
        sAPI_Debug("dial fail");
    }

    ret = sAPI_CallAnswerMsg(g_call_msgq);
    if(ret == SC_CALL_SUCESS);
    {
        memset(&msg, 0, sizeof(msg););
        status = sAPI_MsgQRecv(g_call_msgq, &msg, SC_SUSPEND);
        sAPI_Debug("rc[%d] ", msg.arg2);
    }
    else
    {
        sAPI_Debug("answer fail");
    }
```

```
    ret = sAPI_CallEndMsg(g_call_msgq);
    if(ret == SC_CALL_SUCESS);
    {
        memset(&msg, 0, sizeof(msg););
        status = sAPI_MsgQRecv(g_call_msgq, &msg, SC_SUSPEND);
        sAPI_Debug("rc[%d] ", msg.arg2);
    }
    else
    {
        sAPI_Debug("end fail");
    }

    state = sAPI_CallStateMsg();
    sAPI_Debug("state[%d] ", state);

    ret = sAPI_CallAutoAnswer(4, g_call_msgq);
    if(ret == SC_CALL_SUCESS);
    {
        memset(&msg, 0, sizeof(msg););
        status = sAPI_MsgQRecv(g_call_msgq, &msg, SC_SUSPEND);
        sAPI_Debug("rc[%d] ", msg.arg2);
    }
    else
    {
        sAPI_Debug("set auto answer fail");
    }
}
```

6 APIs for SMS

6.1 Overview of APIs for SMS

Interface	Description
sAPI_SmsWriteMsg	Write message to memory
sAPI_SmsSendMsg	Send message
sAPI_SmsSendStorageMsg	Send message from storage
sAPI_SmsReadMsg	Read a stored message
sAPI_SmsDelAllMsg	Delete all atored message in SIM cared

6.2 Detailed Description of APIs for SMS

6.2.1 sAPI_SmsWriteMsg

This function is used to return message with location value <index> from message storage <mem1> to the TE.

sAPI_SmsWriteMsg	
Prototype	SC_SMSReturnCode **sAPI_SmsWriteMsg**(SCsmsFormatMode fmatmode, UINT8* sms, UINT16 smslen, UINT8* srcAddr, UINT8 stat, sMsgQRef msgQ);
Parameters	[in] **fmatmode**: The input and output format of the short messages [in] **sms**: Message body [in] **smslen**: The length of the message body [in] **srcAddr**: Destination-Address [in] **stat**: 1: stored unsent message 2: stored sent message [in] **msgQ**: Results will be returned by msgQ
Return value	SC_SMS_SUCESS: write was sucessfull -: read was unsucessfull
Header	#include "simcom_sms.h"

Examples

Refer to demo of APIs for SMS

6.2.2 sAPI_SmsSendMsg

This function is used to send message from a TE to the network(SMS-SUBMIT);

sAPI_SmsSendMsg	
Prototype	SC_SMSReturnCode **sAPI_SmsSendMsg**(SCsmsFormatMode fmatmode, UINT8* sms, UINT16 smslen, UINT8* addr, sMsgQRef msgQ);
Parameters	[in] **fmatmode**: The input and output format of the short messages [in] **sms**: Message body [in] **smslen**: The length of the message body [in] **addr**: Destination-Address [in] **msgQ**: Results will be returned by msgQ
Return value	SC_SMS_SUCESS: read was sucessfull -: read was unsucessfull
Header	#include "simcom_sms.h"

Examples

Refer to demo of APIs for SMS

6.2.3 sAPI_SmsSendStorageMsg

This function is used to send message with location value <index> from preferred message storage <mem2> to the network(SMS-SUBMIT or SMS-COMMAND);

sAPI_SmsSendStorageMsg	
Prototype	SC_SMSReturnCode **sAPI_SmsSendStorageMsg**(INT32 msgIndex, UINT8* addr, sMsgQRef msgQ);
Parameters	[in] **msgIndex**: Value in the range of location numbers supported by the associated memory and start with one [in] **addr**: Destination-Address [in] **msgQ**: Results will be returned by msgQ

Return value	SC_SMS_SUCESS: send stroge was sucessfull -: send stroge was unsucessfull
Header	#include "simcom_sms.h"

Examples

Refer to demo of APIs for SMS

6.2.4 sAPI_SmsReadMsg

This function is used to read a stored message in SIM card, and the msg will send to user by msgQ.

sAPI_SmsReadMsg	
Prototype	SC_SMSReturnCode **sAPI_SmsReadMsg**(SCsmsFormatMode fmatmode, INT32 msgIndex, sMsgQRef msgQ);
Parameters	[in] **fmatmode**: The input and output format of the short messages [in] **msgIndex**: Value in the range of location numbers supported by the associated memory and start with one [in] **msgQ**: Results will be returned by msgQ
Return value	SC_SMS_SUCESS: send stroge was sucessfull -: send stroge was unsucessfull
Header	#include "simcom_sms.h"

Examples

Refer to demo of APIs for SMS

6.2.5 sAPI_SmsDelAllMsg

This function is used to delete all messages which is stored in SIM card.

sAPI_SmsDelAllMsg	
Prototype	SC_SMSReturnCode **sAPI_SmsDelAllMsg**(sMsgQRef msgQ);
Parameters	[in] **msgQ**: Results will be returned by msgQ
Return value	SC_SMS_SUCESS: send stroge was sucessfull -: send stroge was unsucessfull
Header	#include "simcom_sms.h"

Examples

Refer to demo of APIs for SMS

6.2.6 sAPI_SmsDelOneMsg

This function is used to delete one messages which is stored in SIM card.

sAPI_SmsDelOneMsg	
Prototype	SC_SMSReturnCode **sAPI_SmsDelOneMsg**(INT32 msgIndex, sMsgQRef msgQ);
Parameters	[in] **msgIndex**: Value in the range of location numbers supported by the associated memory and start with one [in] **msgQ**: Results will be returned by msgQ
Return value	SC_SMS_SUCESS: delete stroge was sucessfull -: delete stroge was unsucessfull
Header	#include "simcom_sms.h"

Examples

Refer to demo of APIs for SMS

6.3 Demo for SMS

```
#include "simcom_sms.h"
#include "simcom_debug.h"
sMsgQRef g_sms_msgq;

int sms_testMain(void);
{
    SC_SMSReturnCode ret;
    SC_STATUS status = OS_FAIL;
    SIM_MSG_T msg;

    status = sAPI_MsgQCreate(&g_sms_msgq, "g_sms_msgq", sizeof(SIM_MSG_T), 4, SC_FIFO);
```

```
ret = sAPI_SmsWriteMsg(1, "123456", strlen("123456"), "18225165872", 0, g_sms_msgq);
if(ret == SC_SMS_SUCESS);
{
    memset(&msg, 0, sizeof(msg));;
    status = sAPI_MsgQRecv(g_sms_msgq, &msg, SC_SUSPEND);
    sAPI_Debug("rc[%d] reference[%d] ", msg.arg1, msg.arg2);
}
else
{
    sAPI_Debug("write fail");

}

ret = sAPI_SmsWriteMsg(0, "0011000B913188735695F60000FF0631D98C56B301", strlen("0011000B913188735695F60000FF0631D98C56B301"), NULL, 0, g_sms_msgq);
if(ret == SC_SMS_SUCESS);
{
    memset(&msg, 0, sizeof(msg));;
    status = sAPI_MsgQRecv(g_sms_msgq, &msg, SC_SUSPEND);
    sAPI_Debug("rc[%d] reference[%d] ", msg.arg1, msg.arg2);
}
else
{
    sAPI_Debug("write fail");
}

ret = sAPI_SmsWriteMsg(0, "039111F011000B913188735695F60000FF0631D98C56B301", strlen("039111F011000B913188735695F60000FF0631D98C56B301"), NULL, 0, g_sms_msgq);
if(ret == SC_SMS_SUCESS);
{
    memset(&msg, 0, sizeof(msg));;
    status = sAPI_MsgQRecv(g_sms_msgq, &msg, SC_SUSPEND);
```

```
        sAPI_Debug("rc[%d] reference[%d] ", msg.arg1, msg.arg2);
}
else
{
        sAPI_Debug("write fail");
  }

ret = sAPI_SmsSendMsg(1, "hello world", strlen("hello world"), "13883765596", g_sms_msgq);
if(ret == SC_SMS_SUCESS);
{
        memset(&msg, 0, sizeof(msg));;
        status = sAPI_MsgQRecv(g_sms_msgq, &msg, SC_SUSPEND);
        sAPI_Debug("rc[%d] reference[%d] ", msg.arg1, msg.arg2);
}
else
{
        sAPI_Debug("send fail");
}

ret = sAPI_SmsSendStorageMsg(1, "10086", g_sms_msgq);
if(ret == SC_SMS_SUCESS);
{
        memset(&msg, 0, sizeof(msg));;
        status = sAPI_MsgQRecv(g_sms_msgq, &msg, SC_SUSPEND);
        sAPI_Debug("rc[%d] reference[%d] ", msg.arg1, msg.arg2);
}
else
{
        sAPI_Debug("send fail");
  }

ret = sAPI_SmsReadMsg(1, g_sms_msgq);
if(ret == SC_SMS_SUCESS);
```

```
{
    memset(&msg, 0, sizeof(msg));;
    status = sAPI_MsgQRecv(g_sms_msgq, &msg, SC_SUSPEND);
    sAPI_Debug("rc[%d] reference[%d] ", msg.arg1, msg.arg2);
}
else
{
    sAPI_Debug("read fail");
 }

ret = sAPI_SmsDelAllMsg(g_sms_msgq);
if(ret == SC_SMS_SUCESS);
{
    memset(&msg, 0, sizeof(msg));;
    status = sAPI_MsgQRecv(g_sms_msgq, &msg, SC_SUSPEND);
    sAPI_Debug("rc[%d] reference[%d] ", msg.arg1, msg.arg2);
}
else
{
    sAPI_Debug("DelAll fail");
 }

ret = sAPI_SmsDelOneMsg(1, g_sms_msgq);
if(ret == SC_SMS_SUCESS);
{
    memset(&msg, 0, sizeof(msg));;
    status = sAPI_MsgQRecv(g_sms_msgq, &msg, SC_SUSPEND);
    sAPI_Debug("rc[%d] reference[%d] ", msg.arg1, msg.arg2);
}
else
{
    sAPI_Debug("DelOne fail");
}
```

```
}
```

7 APIs for UART

7.1 Overview of APIs for UART

Interface	Description
sAPI_UartWrite	Sends data to the uart.
sAPI_UartRead	Reads data from the uart.
sAPI_UartWriteString	Sends string directly to the uart.
sAPI_UartSetConfig	Sets the configuration of the uart, that includes baud-rate and the format of the frame.
sAPI_UartGetConfig	Gets the configuration of the uart, that includes baud-rate and the format of the frame.
sAPI_UartPrintf	Print the log by uart2(debug uart).
sAPI_SendATCMDWaitResp	Send / receive AT command internally
sAPI_UartRxStatus	Gets the status of Rx.
sAPI_UartControl	Control UART opening or closing.
sAPI_UartRegisterCallback	Bind a callback function for UART.
sAPI_UartRegisterCallbackEX	Bind a callback function for UART, And passes a pointer argument to the callback function.
sAPI_UartRs485DePinAssign	Assign a Rs485 dePin to a UART.

7.2 Detailed Description of APIs for UART

NOTE

The A7600 series supports a total of three serial ports, which include Enhanced UART, UART3 and standard UART. Enhanced UART is called **SC_UART** in code, UART3 is called **SC_UART3** in code, standard UART is used to print out some logs at the boot stage, called **SC_UART2** in code. When the module has GPS chip, GPS will occupy **UART3.**

7.2.1 sAPI_UartWrite

This application interface is used to send data to the **UART**.

sAPI_UartWrite	
Prototype	SC_Uart_Return_Code **sAPI_UartWrite**(SC_Uart_Port_Number port, UINT8 *data, UINT32 length);
Parameters	[in] **port**: the port number of UART. Three UART port are supported: SC_UART,SC_UART2 and SC_UART3. [in] **data**: Pointer to the data address. [in] **length**: The Maximum length of data.
Return value	SC_Uart_Return_Code_OK: send done. SC_Uart_Return_Code_ERROR: fail.
Header	#include "simcom_uart.h"

Examples

```
#include "simcom_api.h"
#include "simcom_uart.h"
#include "simcom_debug.h"
#include "simcom_common.h"

if(SC_Uart_Return_Code_OK != sAPI_UartWrite(SC_UART, buf, length))
{
    sAPI_Debug("uart send data error!!");
}
if(SC_Uart_Return_Code_OK != sAPI_UartWrite(SC_UART2, buf, length))
{
    sAPI_Debug("uart2 send data error!!");
}
if(SC_Uart_Return_Code_OK != sAPI_UartWrite(SC_UART3, buf, length))
{
    sAPI_Debug("uart3 send data error!!");
}
```

7.2.2 sAPI_UartRead

This application interface is used to read data to the **UART**.

sAPI_UartWrite

Prototype	int **sAPI_UartRead**(SC_Uart_Port_Number port, unsigned char *data, int len);
Parameters	[in] **port**: the port number of UART. Three UART port are supported: SC_UART,SC_UART2 and SC_UART3. [in] **data**: Pointer to the memory address. [in] **length**: The Maximum length of the data read.
Return value	The actual length of data read.
Header	#include "simcom_uart.h"

Examples

```
#include "simcom_api.h"
#include "simcom_uart.h"
#include "simcom_debug.h"
#include "simcom_common.h"

readLenght = sAPI_UartRead(SC_UART, buf, length);
sAPI_UartPrintf("Read data length:%d\n Read data:%s", readLenght,buf);
```

7.2.3 sAPI_UartWriteString

This application interface is used to send string directly to the **UART**.

sAPI_UartWrite	
Prototype	SC_Uart_Return_Code **sAPI_UartWrite**(SC_Uart_Port_Number port, UINT8 *data);
Parameters	[in] **port**: the port number of UART. Three UART port are supported: SC_UART,SC_UART2 and SC_UART3. [in] **data**: Pointer to the data address.
Return value	SC_Uart_Return_Code_OK: send done. SC_Uart_Return_Code_ERROR: fail.
Header	#include "simcom_uart.h"

Examples

```
#include "simcom_api.h"
#include "simcom_uart.h"
#include "simcom_debug.h"
#include "simcom_common.h"

if(SC_Uart_Return_Code_OK != sAPI_ UartWriteString (SC_UART, buf))
{
```

```
    sAPI_Debug("uart send string error!!");
}
if(SC_Uart_Return_Code_OK != sAPI_ UartWriteString (SC_UART2, buf))
{
    sAPI_Debug("uart2 send string error!!");
}
if(SC_Uart_Return_Code_OK != sAPI_ UartWriteString (SC_UART3, buf))
{
    sAPI_Debug("uart3 send string error!!");
}
```

7.2.4 sAPI_UartSetConfig

This application interface is used to set the configuration of the **UART**, that includes baud-rate and the format of the frame. If you want to change the initialization configuration, you can modify three structs uartConfig , uart2Config and uart3Config in **cus_uart.c.**

sAPI_UartSetConfig	
Prototype	SC_Uart_Return_Code **sAPI_UartSetConfig**(SC_Uart_Port_Number port, const SCuartConfiguration *config);
Parameters	[in] **port**: the port number of UART. Three UART port are supported: SC_UART,SC_UART2 and SC_UART3. [in] **config**: pointer to the configuration block of the UART. This parameters are specified in a SCuartConfiguration structure type.
	SC_Uart_Return_Code_OK: set done. SC_Uart_Return_Code_ERROR: fail.
Header	#include "simcom_uart.h"

NOTE

```
typedef struct SCuartConfiguration
{
    SC_UART_BaudRates BaudRate;    //the baudrate of the uart(default - 115200)
    SC_UART_WordLen DataBits;      // 7 or 8 number of data bits in the UART data frame (default - 8).
    SC_UART_StopBits StopBits;    // 1, 1.5 or 2 stop bits in the UART data frame (default - 1).
    SC_UART_ParityTBits ParityBit; // Even, Odd or no-parity bit type in the UART data frame (default - Non).
}SCuartConfiguration;
```

Examples

```
#include "simcom_api.h"
```

```
#include "simcom_uart.h"
#include "simcom_debug.h"
#include "simcom_common.h"

SC_Uart_Return_Code ret = SC_Uart_Return_Code_OK;
SCuartConfiguration uartConfig;

uartConfig.DataBits = SC_UART_WORD_LEN_8;
uartConfig.ParityBit = SC_UART_NO_PARITY_BITS;
uartConfig.StopBits = SC_UART_ONE_STOP_BIT;
uartConfig.BaudRate = SC_UART_BAUD_115200;
if(SC_Uart_Return_Code_OK == sAPI_UartSetconfig(SC_UART, &uartConfig))
{
    sAPI_Debug("uart setting configuration done!!");
}
```

7.2.5 sAPI_UartGetConfig

This application interface is used to get the configuration of the **UART**, that includes baud-rate and the format of the frame.

sAPI_UartGetConfig	
Prototype	SC_Uart_Return_Code **sAPI_UartGetConfig**(SC_Uart_Port_Number port, SCuartConfiguration *config);
Parameters	[in] **port**: the port number of UART. Three UART port are supported: SC_UART,SC_UART2 and SC_UART3. [out] **config**: Pointer to the configuration block of the **uart**. This parameters are specified in a SCuartConfiguration structure type.
Return value	SC_Uart_Return_Code_OK: get success. SC_Uart_Return_Code_ERROR: fail.
Header	#include "simcom_uart.h"

Examples

```
#include "simcom_api.h"
#include "simcom_uart.h"
#include "simcom_debug.h"
#include "simcom_common.h"

SC_Uart_Return_Code ret = SC_Uart_Return_Code_OK;
SCuartConfiguration getconfig;
```

```
if(SC_Uart_Return_Code_OK == sAPI_UartGetConfig(SC_UART, &getconfig))
{
    sAPI_Debug("getting baudrate: %d databits: %d paritybit: %d stopbits: %d", getconfig.BaudRate, getconfig.DataBits, getconfig.ParityBit, getconfig.StopBits);
}
```

7.2.6 sAPI_UartPrintf

This application interface is used to print the log by **UART** 2(debug uart).

sAPI_UartPrintf	
Prototype	int **sAPI_UartPrintf**(const char *fmt, ...);
Parameters	The Format String that needs to be output.
Return value	The actual length of string.
Header	#include "simcom_uart.h"

Examples

```
sAPI_UartPrintf("%s: print test", __func__);
```

7.2.7 sAPI_SendATCMDWaitResp

This application interface is used to send/receive AT command internally.

sAPI_SendATCMDWaitResp	
Prototype	int **sAPI_SendATCMDWaitResp**(int sATPInd, char *in_str, int timeout, char *ok_fmt, int ok_flag, char *err_fmt, char *out_str, int outLen);
Parameters	**sATPInd**: channel index, Channel 10 is recommended. **inStr**: a string for AT command. **timeOut**: time. **okFmt**: The return format of the instruction when executed correctly. **okFlag**: flag. **errFmt**: The return format of the instruction when executed incorrectly. **outStr**: return string about AT command. **outLen**: the length of the string.
Return value	0: success. else: fail.
Header	#include "simcom_uart.h"

Examples

```
char respStr[20] ;
int ret = 0;

ret = sAPI_SendATCMDWaitResp(10, "AT+IPR?\r", 150, "+IPR", 1, NULL, respStr, sizeof(respStr));
if(ret == 0);       //success
{
    sAPI_Debug("%s: respStr:%s", __func__, respStr);
}
else
{
    sAPI_Debug("%s: fail !!", __func__);
}
```

7.2.8 sAPI_UartRxStatus

This application interface is used to get the status of the Rx. If Rx exceeds the specified time and no data is received, Rx is idle, otherwise Rx is busy.

sAPI_UartRxStatus	
Prototype	SC_Uart_Rx_Status **sAPI_UartRxStatus**(SC_Uart_Port_Number port, int timeout);
Parameters	[in] **port**: the port number of UART. Three UART port are supported: SC_UART,SC_UART2 and SC_UART3. [in] **timeout**: A threshold that If Rx exceeds the threshold and no data is received, Rx is idle, otherwise Rx is busy. Unit is ms.
Return value	SC_UART_RX_IDEL: 0. SC_UART_RX_BUSY: 1.
Header	#include "simcom_uart.h"

Examples

```
#include "simcom_api.h"
#include "simcom_uart.h"
#include "simcom_debug.h"
#include "simcom_common.h"

SC_Uart_Rx_Status status;

if(SC_UART_RX_IDEL == sAPI_UartRxStatus(SC_UART, 50))
```

```
{
    sAPI_Debug("the Rx of UART is idel.");
}
if(SC_UART_RX_BUSY == sAPI_UartRxStatus(SC_UART, 50))
{
    sAPI_Debug("the Rx of UART is busy.");
}
```

7.2.9 sAPI_UartControl

This application interface is used to control **UART** opening or closing.

sAPI_UartControl	
Prototype	SC_Uart_Return_Code **sAPI_UartControl**(SC_Uart_Port_Number port, SC_Uart_Control ctl);
Parameters	[in] **port**: the port number of UART. Three UART port are supported: SC_UART,SC_UART2 and SC_UART3. [in] **Ctl**: the type of control, SC_UART_OPEN or SC_UART_CLOSE.
Return value	SC_Uart_Return_Code_OK: set success. SC_Uart_Return_Code_ERROR: fail.
Header	#include "simcom_uart.h"

Examples

```
#include "simcom_api.h"
#include "simcom_uart.h"
#include "simcom_debug.h"
#include "simcom_common.h"

if(SC_Uart_Return_Code_OK == sAPI_UartControl(SC_UART3, SC_UART_CLOSE))
{
    sAPI_Debug("Control UART3 success.");
}
if(SC_UART_RETURN_CODE_ERROR == sAPI_UartControl(SC_UART3, SC_UART_OPEN))
{
    sAPI_Debug("Control UART3 failed.");
}
```

7.2.10 sAPI_UartRegisterCallback

This application interface is used to bind a callback function for **UART**.

sAPI_ UartRefRegister	
Prototype	SC_Uart_Return_Code sAPI_UartRegisterCallback(SC_Uart_Port_Number port, SC_Uart_Callback cb);
Parameters	[in] **port**: the port number of UART. Three UART port are supported: SC_UART,SC_UART2 and SC_UART3. [in] **cb**: Functions that need to be bound to UART.
Return value	SC_UART_RETURN_CODE_OK: bind success. SC_UART_RETURN_CODE_ERROR: bind fail.
Header	#include "simcom_uart.h"

Examples

```
#include "simcom_api.h"
#include "simcom_uart.h"

Void CallbackFun(void);
if(SC_UART_RETURN_CODE_ OK == sAPI_ UartRegisterCallback(SC_UART3, CallbackFun))
{
    sAPI_Debug("Callback function bind success.");
}

if(SC_UART_RETURN_CODE_ERROR == sAPI_ UartRegisterCallback(SC_UART3, CallbackFun))
{
    sAPI_Debug("Callback function bind fail.");
}
```

7.2.11 sAPI_UartRegisterCallbackEX

This application interface is used to bind a callback function for **UART**, And passes a pointer argument to the callback function.

sAPI_ UartRefRegister	
Prototype	SC_Uart_Return_Code sAPI_UartRegisterCallbackEX(SC_Uart_Port_Number port, SC_Uart_CallbackEX cb, void *reserve);
Parameters	[in] **port**: the port number of UART. Three UART port are supported: SC_UART,SC_UART2 and SC_UART3.

	[in] **cb**: Functions that need to be bound to UART. [in] **reserve**: The pointer argument need to t be passed in the callback function.
Return value	SC_UART_RETURN_CODE_OK: bind success. SC_UART_RETURN_CODE_ERROR: bind fail.
Header	#include "simcom_uart.h"

Examples

```
#include "simcom_api.h"
#include "simcom_uart.h"

Void CallbackFun(void * reserve);
Void * Reserve;
if(SC_UART_RETURN_CODE_ OK==sAPI_ UartRegisterCallback(SC_UART3,CallbackFun, Reserve))
{
    sAPI_Debug("Callback function bind success.");
}

if(SC_UART_RETURN_CODE_ERROR==sAPI_UartRegisterCallback(SC_UART3,CallbackFun,Reserve))
{
    sAPI_Debug("Callback function bind fail.");
}
```

7.2.12 sAPI_UartRs485DePinAssign

This application interface is used to assign a Rs485 dePin to a **UART**.

sAPI_ UartRefRegister	
Prototype	GPIOReturnCode sAPI_UartRs485DePinAssign(SC_Uart_Port_Number port, Module_GPIONumbers GpinNum);
Parameters	[in] **port**: the port number of UART. Three UART port are supported: SC_UART,SC_UART2 and SC_UART3. [in] **GpinNum**: GPIO PIN assigned.
Return value	GPIORC_OK: Assign a Rs485 dePin success. else: Assign a Rs485 dePin fail.
Header	#include "simcom_uart.h"

Examples

```
#include "simcom_api.h"
```

```
#include "simcom_uart.h"

if(GPIORC_OK ==sAPI_ UartRs485DePinAssign (SC_UART3, MODULE_GPIO_0))
{
    sAPI_Debug("Assign a Rs485 dePin success.");
}
if(GPIORC_OK !=sAPI_ UartRs485DePinAssign (SC_UART3, MODULE_GPIO_0))
{
    sAPI_Debug("Assign a Rs485 dePin fail.");
}
```

7.3 Description for UART

7.3.1 Default configuration

The customer can modifies the initialization configuratin of the **UART** that includes the baud rate and the format of the frame in **sAPP_UartTask**. The function is in **cus_uart.c**.
For Examples: the structure **uartConfig , uart2Config or uart3Config** can be modified.

```
SCuartConfiguration uartConfig;
/************************Configure UART again*********************************/
/*******The user can modify the initialization configuratin of UART in here.***/
/*****************************************************************************/
uartConfig.BaudRate    =    SC_UART_BAUD_115200;
uartConfig.DataBits    =    SC_UART_WORD_LEN_8;
uartConfig.ParityBit   =    SC_UART_NO_PARITY_BITS;
uartConfig.StopBits    =    SC_UART_ONE_STOP_BIT;
if(sAPI_UartSetConfig(SC_UART, &uartConfig) == SC_UART_RETURN_CODE_ERROR)
{
    sAPI_Debug("%s: Configure UART failure!!", __func__);
}

SCuartConfiguration uart2Config
/************************Configure UART2 again*********************************/
/*******The user can modify the initialization configuratin of UART2 in here.***/
/*****************************************************************************/
uart2Config.BaudRate    =    SC_UART_BAUD_115200;
uart2Config.DataBits    =    SC_UART_WORD_LEN_8;
uart2Config.ParityBit =    SC_UART_NO_PARITY_BITS;
uart2Config.StopBits    =    SC_UART_ONE_STOP_BIT;
if(sAPI_UartSetConfig(SC_UART2,&uart2Config) == SC_UART_RETURN_CODE_ERROR)
```

```
{
    sAPI_Debug("%s: Configure UART2 failure!!",__func__);
}

SCuartConfiguration uart3Config
/************************Configure UART3 again*********************************/
/*******The user can modify the initialization configuratin of UART3 in here.***/
/*******************************************************************************/
uart3Config.BaudRate   =   SC_UART_BAUD_115200;
uart3Config.DataBits    =   SC_UART_WORD_LEN_8;
uart3Config.ParityBit    =   SC_UART_NO_PARITY_BITS;
uart3Config.StopBits    =   SC_UART_ONE_STOP_BIT;
if(sAPI_UartSetConfig(SC_UART3, &uart3Config) == SC_UART_RETURN_CODE_ERROR)
{
    sAPI_Debug("%s: Configure UART3 failure!!", __func__);
}
```

7.3.2 Received data from Rx

The customer can receive data from **UART** by Callback function, customer need to bind a Callback function by **sAPI_UartRegisterCallback** or **sAPI_UartRegisterCallbackEX,**Then use **sAPI_UartRead** in the callback function to receive data.

For Examples: the usage of **all uart** is the same.

```
#include "simcom_api.h"
#include "simcom_uart.h"

unsigned char * buf;
int length;
Void CallbackFun(void)
{
    if(sAPI_UartRead(SC_UART, buf, length) > 0)
    {
        sAPI_UartPrintf("receive data:%s", buf);
    }
};
Void Customer(void)
{
    sAPI_ UartRegisterCallback(SC_UART3, CallbackFun));
}
```

8 APIs for USB

8.1 Overview of APIs for USB

Interface	Description
sAPI_UsbVcomWrite	Sends data to Tx of the virtual USB port
sAPI_UsbVcomRefRegister	The registerd message reference will be receive the data from usb_vcom.

8.2 Detailed Description of APIs for USB

8.2.1 sAPI_UsbVcomWrite

This application interface is used to send data to Tx of the virtual USB port.

sAPI_UsbVcomWrite	
Prototype	void **sAPI_UsbVcomWrite**(unsigned char* data, unsigned long length);
Parameters	[in] **data**: Pointer to the data address. [in] **length**: The length of data.
Return value	None.
Header	#include "simcom_usb_vcom.h"

Examples

Reference 7.3.1

8.2.2 sAPI_UsbVcomRefRegister

This application interface is used to send data to Tx of the virtual USB port.

sAPI_UsbVcomRefRegister	
Prototype	int sAPI_UsbVcomRefRegister(sMsgQRef ref);
Parameters	[in] **ref**: message reference that will be receive the URC.
Return value	0: OK. -1: ref invalied. -2: ref already exit.
Header	#include "simcom_usb_vcom.h"

Examples

```
#include "simcom_api.h"
#include " simcom_usb_vcom.h "

int ret = 0;
ret = sAPI_UsbVcomRefRegister (gUsbVcom_msgq);
if (0 != ret)
{
    sAPI_MsgQDelete(gUsbVcom_msgq);
    return;
}
```

8.3 Description for USB

8.3.1 Received data from Rx

USB AT port no longer supports AT function, The customer can receive data from rx by function **sTask_UsbVcomRxProcesser**. The function is in **cus_usb_vcom.c**.

For Examples:

```
while(1);
    {
        /*now pend for ever to msgq*/
        osStatus = sAPI_MsgQRecv(gUsbVcom_msgq, &UsbVcomMsg, SC_SUSPEND);
        if(SRV_USB_VCOM != UsbVcomMsg.msg_id)
        {
            sAPI_Debug("%s, msg_id is error!!", __func__);
            break;
        }
```

```
        readsize=UsbVcomMsg.arg1;
        rxbuf = UsbVcomMsg.arg3;

        sAPI_Debug("%s, rxbuf:%s readsize:%d", __func__, rxbuf, readsize);

        /*   user can get data from usb_vcom rx                    */
        /*   readsize: the length of data                      */
        /*   rxbuf: the header address of data space       */


        sAPI_UsbVcomWrite(rxbuf, readsize); //for test

        sAPI_Free(UsbVcomMsg.arg3);
        UsbVcomMsg.arg1 = 0;
        UsbVcomMsg.arg3 = NULL;
        rxbuf = NULL;
}
```

9 APIs for System Sleep

9.1 Overview of APIs for System Sleep

Interface	Description
sAPI_SystemSleepSet	Module can enter sleep mode or exit from sleep mode by setting the flag.
sAPI_SystemSleepGet	Get the flag of the sleep mode.
sAPI_SystemAlarmClock2Wakeup	This interface is designed to periodically wake up the system.

9.2 Detailed Description of APIs for System Sleep

9.2.1 sAPI_SystemSleepSet

This application interface is used to set a flag to put the module into or out of sleep mode.

sAPI_SystemSleepSet	
Prototype	int **sAPI_SystemSleepSet**(SC_SYSTEM_SLEEP_FLAG flag);
Parameters	[in] **flag**: the flag of the system sleep mode.
Return value	0: set done. -1: fail.
Header	#include "simcom_system.h"

NOTE

Please unplug the USB to put the module into sleep mode and pull down the DTR pin to wake up the module.

Examples

```
#include "simcom_system.h"

int ret = 0;

ret = sAPI_SystemSleepSet(SC_SYSTEM_SLEEP_ENABLE);
if(-1 == ret)
{
    sAPI_Debug("%s, flag sets invalid!", __func__);
}
```

9.2.2 sAPI_SystemSleepGet

This application interface is used to get a flag about current sleep mode.

sAPI_SystemSleepFlagGet	
Prototype	SC_SYSTEM_SLEEP_FLAG **sAPI_SystemSleepGet**(void);
Parameters	None.
Return value	flag
Header	#include "simcom_system.h"

Examples

```
#include "simcom_system.h"

SC_SYSTEM_SLEEP_FLAG flag;

flag = sAPI_SystemSleepGet();
sAPI_Debug("%s, flag is %d!", __func__, flag);
```

9.2.3 sAPI_SystemSleepExSet

This application interface is used to ehance the module into or out of sleep mode.

sAPI_SystemSleepExSet	
Prototype	int **sAPI_SystemSleepExSet**(SC_SYSTEM_SLEEP_FLAG flag, unsigned char time);

Parameters	[in] **flag**: the flag of the system sleep mode. [in]**time**: the time into sleep
Return value	0: set done. -1: fail.
Header	#include "simcom_system.h"

NOTE

Please unplug the USB to put the module into sleep mode and pull down the DTR pin to wake up the module.

Examples

```
#include "simcom_system.h"

int ret = 0;

ret = sAPI_SystemSleepExSet(SC_SYSTEM_SLEEP_ENABLE,3);
if(-1 == ret)
{
    sAPI_Debug("%s, flag sets invalid!", __func__);
}
```

9.2.4 sAPI_SystemSleepExGet

This application interface is used to get a flag about current sleep mode.

sAPI_SystemSleepFlagExGet	
Prototype	SC_SleepEx_str **sAPI_SystemSleepExGet**(void);
Parameters	None.
Return value	flag
Header	#include "simcom_system.h"

Examples

```
#include "simcom_system.h"

SC_SleepEx_str sleepExStatus;
```

```
sleepExStatus = sAPI_SystemSleepExGet();
sAPI_Debug("%s, flag is %d, %d!", __func__, sleepExStatus.sleep_flag, sleepExStatus.time);
```

9.2.5 sAPI_SystemAlarmClock2Wakeup

This interface is designed to periodically wake up the system.

sAPI_SystemAlarmClock2Wakeup	
Prototype	int **sAPI_SystemAlarmClock2Wakeup**(unsigned long time);
Parameters	[in] **time**: The sleep time of the module
Return value	0: OK. -1: fail.
Header	#include "simcom_system.h"

Examples

```
#include "simcom_system.h"

int ret;

ret = sAPI_SystemAlarmClock2Wakeup(5000);
sAPI_Debug("%s, ret %d!", __func__, ret);
```

10 APIs for GPIO

10.1 Overview of APIs for GPIO

Interface	Description
sAPI_GpioSetValue	Set a specified GPIO's value in output mode
sAPI_GpioGetValue	Get a specified GPIO's value in input mode
sApi_GpioConfig	Configure all attributes of a specified GPIO
sAPI_GpioConfigInterrupt	Configure common gpio to interrupt gpio
sAPI_GpioSetDirection	Configure the direction of gpio
sAPI_GpioGetDirection	Get a specified GPIO's direction
sAPI_GpioWakeupEnable	Configure wakeup enable of GPIO

10.2 Module Gpio Configuration Table

The working state of all gpios of the whole module is initialized through **Module GPIO Configuration Table** in *cus_gpio.c*. This table will be automatically read and configured into the system when the module is started. It is an array of multiple **SC_GPIOConfiguration** structures. A structure consists of six properties, representing a GPIO.

The struct and six properties as follows:

```
typedef struct
{
    SC_GPIOPinDirection pinDir;
    UINT32              initLv;
    SC_GPIOPullUpDown   pinPull;
    SC_GPIOTransitionType pinEd;
    SC_GPIOCallback isr;
    GPIOCallback wu;
} SC_GPIOConfiguration;
```

1. The first property means pin direction

2. The second property means initial level you want.

3. The third property means pull up or pull down when it is input mode.

4. The fourth property means triger type of interrupt.

5. ISR

6. wake up routine.

If you want to configure a GPIO as interrupt, refer to *A7600E gpio1* shown above to set it. You need to set the input mode at least, select the interrupt trigger mode such as rising edge trigger(SC_GPIO_RISE_EDGE), falling edge trigger(SC_GPIO_FALL_EDGE); or bilateral trigger(SC_GPIO_TWO_EDGE), and provide an ISR.

10.3 Detailed Description of APIs for GPIO

Please refer to **Section 1.5.1** to get all gpio resources and MACRO.

10.3.1 sAPI_GpioSetValue

Get a specified GPIO's value on output mode.

sAPI_GpioSetValue	
Prototype	SC_GPIOReturnCode **sAPI_GpioSetValue**(unsigned int gpio, unsigned int value);
Parameters	[in] **gpio**: module gpio number. [in] **value**: 0 is low level, 1 is high level.
Return value	SC_GPIORC_OK: ok. SC_GPIORC_INVALID_PIN_NUM: invalid number.
Header	simcom_gpio.h

NOTE

Before setValue, please comfirm that the gpio is in the output direction.

Examples

```
...
sAPI_GpioSetValue(SC_MODULE_GPIO_77, 0);
…
```

10.3.2 sAPI_GpioGetValue

Get a specified GPIO's value on input mode.

sAPI_GpioGetValue	
Prototype	SC_GPIOReturnCode **sAPI_GpioGetValue**(unsigned int gpio);
Parameters	[in] **gpio**: module gpio number.
Return value	0: low level. 1: high level. SC_GPIORC_INVALID_PIN_NUM: invalid number.
Header	simcom_gpio.h

NOTE

Before getvalue, please ensure the direction of the pin is input.

Examples

```
...
Int value = sAPI_GpioGetValue(SC_MODULE_GPIO_77);
…
```

10.3.3 sAPI_GpioConfig

This API is used to configure all GPIO's contributes

sAPI_GpioConfig	
Prototype	SC_GPIOReturnCode **sAPI_GpioConfig**(unsigned int gpio,SC_GPIOConfiguration GpioConfig);
Parameters	[in] **gpio**: module gpio number. [in] **GpioConfig:** As explained in section 7.2
Return value	SC_GPIORC_OK: ok. SC_GPIORC_INVALID_PIN_NUM: invalid number.
Header	simcom_gpio.h

Examples

```
SC_GPIOReturnCode ret;
GPIOConfiguration newGpioConfig = {
…
…
};
ret = sAPI_GpioConfig(SC_MODULE_GPIO_1,newGpioConfig);
```

10.3.4 sAPI_GpioConfigInterrupt

This API is used to configure common gpio to interrupt gpio.

sAPI_ GpioConfigInterrupt	
Prototype	SC_GPIOReturnCode **sAPI_GpioConfigInterrupt**(unsigned int gpio, SC_GPIOtransitionType type, GPIOCallBack handler);
Parameters	[in] **gpio**: module gpio number. [in] **type**: interrupt trigger way. [in] **handler**: gpio interrupt routine.
Return value	SC_GPIORC_OK: ok. SC_GPIORC_INVALID_PIN_NUM: invalid number.
Header	simcom_gpio.h

Examples

```
SC_GPIOReturnCode ret;
void handle(void);{
…..
}
ret = sAPI_GpioConfig(SC_MODULE_GPIO_1, SC_TWO_EDGE, handle);
```

10.3.5 sAPI_GpioSetDirection

This API is used to configure the direction of gpio.

sAPI_ GpioSetDirection	
Prototype	SC_GPIOReturnCode **sAPI_GpioSetDirection(**unsigned int gpio, unsigned int direction);
Parameters	[in] **gpio**: module gpio number. [in] **derection**: 0 means input, 1 means output.

Return value	SC_GPIORC_OK: ok. SC_GPIORC_INVALID_PIN_NUM: invalid number.
Header	simcom_gpio.h

Examples

```
SC_GPIOReturnCode ret;
Ret = sAPI_GpioSetDirection(SC_MODULE_GPIO_1, 1);
```

10.3.6 sAPI_GpioGetDirection

This API is used to get a specified GPIO's direction.

sAPI_ GpioGetDirection	
Prototype	SC_GPIOReturnCode **sAPI_GpioGetDirection**(unsigned int gpio);
Parameters	[in] **gpio**: module gpio number.
Return value	0: input 1: output SC_GPIORC_INVALID_PIN_NUM: invalid number.
Header	simcom_gpio.h

Examples

```
SC_GPIOReturnCode ret;
ret = sAPI_GpioGetDirection(SC_MODULE_GPIO_1);
```

10.3.7 sAPI_GpioWakeupEnable

This API is used to configure wakeup enable of GPIO.

sAPI_ GpioWakeupEnable	
Prototype	SC_GPIOReturnCode **sAPI_GpioWakeupEnable**(unsigned int gpio, SC_GPIOTransitionType type);
Parameters	[in] **gpio**: module gpio number. [in] **type**: Gpio edge detection type.
Return value	SC_GPIORC_OK: ok. SC_GPIORC_INVALID_PIN_NUM: invalid number.
Header	simcom_gpio.h

Examples

```
SC_GPIOReturnCode ret;
ret = sAPI_GpioWakeupEnable(SC_MODULE_GPIO_1, SC_GPIO_FALL_EDGE);
```

11 APIs for PMU

11.1 Overview of APIs for PMU

Interface	Description
sAPI_ReadAdc	Read ADC Voltage
sAPI_ReadVbat	Read Battery Voltage
sAPI_SysPowerOff	Module PowerOff
sAPI_SysReset	Module Reset
sAPI_SetVddAux	Set VddAux's Volatge

11.2 Detailed Description of APIs for PMU

11.2.1 sAPI_ReadAdc

Currently the api only applies to ASR1601
Read ADC voltage

sAPI_ReadAdc	
Prototype	unsigned int **sAPI_ReadAdc**(int channel);
Parameters	[in] **Channel**: 0 is read ADC0; 1 is read ADC1;
Return value	voltage: mv.
Header	simcom_pm.h

NOTE

Minimum voltage: 0 mv
Maximum voltage : 1300 mv

Examples

…
…
unsigned int value1 = sAPI_ReadAdc(0);
unsigned int value2 = sAPI_ReadAdc(1);
…

11.2.2 sAPI_ReadVbat

Read the module's Battery Voltage(Supply voltage);

sAPI_ ReadVbat	
Prototype	unsigned int **sAPI_ReadVbat**(void);
Parameters	None
Return value	Voltage: read voltage(mv);
Header	simcom_pm.h

Examples

…
unsigned int value = sAPI_ReadVbat();
…

11.2.3 sAPI_SysPowerOff

Make module PowerOff

sAPI_SysPowerOff	
Prototype	void **sAPI_SysPowerOff**(void);
Parameters	None
Return value	None
Header	simcom_pm.h

Examples

…
sAPI_SysPowerOff();

…

11.2.4 sAPI_SysReset

Make module Reset

sAPI_SysReset	
Prototype	void **sAPI_SysReset**(void);
Parameters	None
Return value	None
Header	simcom_pm.h

Examples:

…
sAPI_SysReset();
…

11.2.5 sAPI_SetVddAux

This API is used to set VddAux Volatge.

sAPI_SetVddAux	
Prototype	int **sAPI_SetVddAux**(unsigned int voltage);
Parameters	[in] **voltage**: voltage setting value(mv);
Return value	0: ok. -1: error.
Header	simcom_pm.h

NOTE

The voltage can be set as 1200, 1250, 1700, 1800, 1850, 1900, 2500, 2600, 2700, 2750, 2800, 2850, 2900, 3000, 3100, 3300.

Examples

…
Int ret;
ret = sAPI_ SetVddAux(2750);
…

11.2.6 sAPI_GetPowerUpEvent

Make module Reset

sAPI_SysReset	
Prototype	int **sAPI_GetPowerUpEvent** (void);
Parameters	None
Return value	1: Press the reset key to power up 2: power key exton1 power up 3: soft reset fault wake up 4: rtc_alarm wake up
Header	simcom_pm.h

Examples

Int ret = 0;
Ret = sAPI_GetPowerUpEvent ();

11.2.7 sAPI_GetPowerDownEvent

Make module Reset

sAPI_SysReset	
Prototype	int **sAPI_GetPowerDownEvent**(void)
Parameters	None
Return value	1: software power down ,long press power key 2: VRTC Fall to VRTC_MIN_TH 3: an internal overtemperature in the internal PMIC temperature detector. 4: VINLDO going lower than UV_VSYS1_DETECT (like battery removal without aconnected charger). 5: PMIC watch dog expiry event 6: long press of ONKEY 7: VINLDO going above than VSYS_OVER_TH.

Header	simcom_pm.h

Examples:

```
…
Int ret = 0;
ret = sAPI_GetPowerDownEvent ();
```

12 APIs for SPI

12.1 Overview of APIs for SPI

Interface	Description
sAPI_ExtNorFlashBlock64kErase	Nor flash block 64k erase
sAPI_ExtNorFlashSectorErase	Nor flash sector erase
sAPI_ExtNorFlashWrite	Nor flash write
sAPI_ExtNorFlashRead	Nor flash read
sAPI_ExtNorFlashReadID	Nor flash read ID
sAPI_SPIReadBytes	SPI read bytes
sAPI_SPIWriteBytes	SPI write bytes
sAPI_SPIConfigInit	Configure the clock and mode for SPI

12.2 Detailed Description of APIs for SPI

12.2.1 sAPI_ExtNorFlashBlock64kErase

Nor flash block 64k erase.

sAPI_ExtNorFlashBlock64kErase	
Prototype	SC_SPI_ReturnCode **sAPI_ExtNorFlashBlock64kErase**(int offset, int size);
Parameters	[in] **offset**: Nor flash erase begin address; [in] **size**: the data lenghth which will be erase;
Return value	SC_SPI_RC_OK: spi erase ok. SC_SPI_RC_ERROR: spi erase error.
Header	simcom_spi.h

Examples

…
…
SC_I2C_ReturnCode ret;
ret = sAPI_ExtNorFlashBlock64kErase(0x000000, 0x10000);
…

12.2.2 sAPI_ExtNorFlashSectorErase

Nor flash sector erase.

sAPI_ ExtNorFlashSectorErase	
Prototype	SC_SPI_ReturnCode **sAPI_ExtNorFlashSectorErase**(int offset, int size);
Parameters	[in] **offset**: Nor flash erase begin address; [in] **size**: the data lenghth which will be erase;
Return value	SC_SPI_RC_OK: spi erase ok. SC_SPI_RC_ERROR: spi erase error.
Header	simcom_spi.h

Examples

…
SC_I2C_ReturnCode ret;
ret = sAPI_ExtNorFlashSectorErase(0x000000, 0x1000);
…

12.2.3 sAPI_ExtNorFlashWrite

Nor flash write.

sAPI_ExtNorFlashWrite	
Prototype	SC_SPI_ReturnCode **sAPI_ExtNorFlashWrite**(unsigned int Address, int Size, unsigned int Buffer);
Parameters	[in] **Address**: Nor flash write begin address; [in] **Size**: the data lenghth which will be write; [in] **Buffer**: the data cache which will be write;
Return value	SC_SPI_RC_OK: spi write ok. SC_SPI_RC_ERROR: spi write error.
Header	simcom_spi.h

Examples

```
…
char Buffer[300] = {0};
SC_I2C_ReturnCode ret;
ret = sAPI_ExtNorFlashSectorErase(0x000000, 0x100, (unsigned int); Buffer);
…
```

12.2.4 sAPI_ExtNorFlashRead

Nor flash read.

sAPI_ExtNorFlashRead	
Prototype	SC_SPI_ReturnCode **sAPI_ExtNorFlashRead**(unsigned int FlashOffset, int Size, unsigned int Buffer);
Parameters	[in] **FlashOffset**: Nor flash write begin address; [in] **Size**: the data lenghth which will be read; [in] **Buffer**: the data cache which will be read;
Return value	SC_SPI_RC_OK: spi read ok. SC_SPI_RC_ERROR: spi read error.
Header	simcom_spi.h

Examples:

```
…
char BufRev[300] = {0};
SC_I2C_ReturnCode ret;
ret = sAPI_ExtNorFlashSectorErase(0x000000, 0x100, (unsigned int); BufRev);
…
```

12.2.5 sAPI_ExtNorFlashReadID

Nor flash read ID.

sAPI_ExtNorFlashReadID	
Prototype	SC_SPI_ReturnCode **sAPI_ExtNorFlashReadID**(unsigned char *id);
Parameters	[in] **id**: the id cache which will be read;

Return value	SC_SPI_RC_OK: spi read id ok. SC_SPI_RC_ERROR: spi read id error.
Header	simcom_spi.h

Examples

```
…
Unsigned char id[4] = {0};
SC_I2C_ReturnCode ret;
ret = sAPI_ExtNorFlashReadID(id);
if(ret != 0);
      sAPI_Debug("read id fail\r\n");
sAPI_Debug("read id =0x%x\r\n", id[1] );
…
```

12.2.6 sAPI_SPIReadBytes

SPI reads bytes.

sAPI_SPIReadBytes	
Prototype	SC_SPI_ReturnCode **sAPI_SPIReadBytes**(unsigned int *SendData,unsigned int *RevData,int Size);
Parameters	[in] **SendData**: data array which will be sent; [in] **RevData**: data array which will be receive; [in] **Size**: Size of the data array to be received;
Return value	SC_SPI_RC_OK: spi read ok. SC_SPI_RC_ERROR: spi read error.
Header	simcom_spi.h

Examples

```
…
char SendData[4] = {0x9F,0,0,0};
char RevData[4] = {0};
sAPI_SPIReadBytes((unsigned int *)SendData,(unsigned int *)RevData,4);
if(ret != 0)
sAPI_Debug("ReadBytes fail\r\n");
…
```

12.2.7 sAPI_SPIWriteBytes

SPI write bytes.

sAPI_SPIWriteBytes	
Prototype	SC_SPI_ReturnCode **sAPI_SPIWritesBytes**(unsigned int *SendData,int Size);
Parameters	[in] **SendData**: data array which will be sent; [in] **Size**: Size of the data array to be sent;
Return value	SC_SPI_RC_OK: spi write ok. SC_SPI_RC_ERROR: spi write error.
Header	simcom_spi.h

Examples

```
…
char senddata[4] = {0x66,0x12,0x32,0x66};
sAPI_SPIWriteBytes((unsigned int *)senddata,4);
if(ret != 0)
sAPI_Debug("WriteBytes fail\r\n");
…
```

12.2.8 sAPI_SPIConfigInit

Configure the clock and mode for SPI.

sAPI_SPIConfigInit	
Prototype	void sAPI_SPIConfigInit(int ssp_clk, int ssp_mode);
Parameters	[in] **ssp_clk**: SPI clock; [in] **ssp_mode**: SPI polarity and phase;
Return value	NO
Header	simcom_spi.h

Examples

```
…
sAPI_SPIConfigInit(SPI_CLOCK_6MHz,SPI_MODE_PH0_PO0);
…
```

13 APIs for I2C

13.1 Overview of APIs for I2C

Interface	Description
sAPI_I2CRead	Receive I2C data
sAPI_I2CWrite	Send I2C data
sAPI_I2CReadEx	Receive I2C data(repeat start)

13.2 Detailed Description of APIs for I2C

13.2.1 sAPI_I2CRead

Receive i2c data.

sAPI_I2CRead	
Prototype	SC_I2C_ReturnCode **sAPI_I2CRead**(int i2cChannel, UINT8 slaveAddress, UINT8 regAddress, UINT8 *receiveDataBuffer, UINT16 datasize);
Parameters	[in] **i2cChannel**: choose i2c controller [in] **slaveAddress** : The meaning of this parameter is to shift the 7-bit device address one bit to the left to get the 8bit address. Note: both read and write operations use the 7-bit real device address to shift one bit to the left. For example, the 7-bit address is 0x1a, and the parameter value is 0x34. [in] **regAddress**: 8bits register address of slave address. [out] **receiveDataBuffer**: data array which used to receive data. [in] **datasize**: the data lenghth which will be received.
Return value	SC_I2C_RC_OK: i2c read ok. SC_I2C_RC_NOT_OK: i2c read error.
Header	simcom_i2c.h

Examples

```
…
…
    UINT8 data_2[2] = {0x55, 0x56};
    if(SC_I2C_RC_OK == sAPI_I2CWrite(0, 0x34, (UINT8);(0x41<<1), data_2, 2));
        uart_printf("I2C write suc(1);\r\n");
    else
        uart_printf("I2C write error(1);\r\n");
    sAPI_I2CRead(0, 0x34, (UINT8);(0x41<<1), receiveData, 2);
        uart_printf("I2C read 0x%x     0x%x\r\n", receiveData[0] , receiveData[1] );
…
```

13.2.2 sAPI_I2CWrite

Send i2c data to specified destination.

sAPI_I2CWrite	
Prototype	SC_I2C_ReturnCode **sAPI_I2CWrite**(int i2cChannel, UINT8 slaveAddress, UINT8 regAddress, UINT8 *data, UINT16 datasize);
Parameters	[in] **i2cChannel**: choose i2c controller [in] **slaveAddress** : The meaning of this parameter is to shift the 7-bit device address one bit to the left to get the 8bit address. Note: both read and write operations use the 7-bit real device address to shift one bit to the left. For example, the 7-bit address is 0x1a, and the parameter value is 0x34. [in] **regAddress**: 8bits register address of slave address. [in] **data**: data array which will be sent. [in] **datasize**: the data lenghth.
Return value	SC_I2C_RC_OK: i2c write ok. SC_I2C_RC_NOT_OK: i2c write error.
Header	simcom_i2c.h

Examples

```
…
…
    UINT8 data_2[2] = {0x55, 0x56};
    if(SC_I2C_RC_OK == sAPI_I2CWrite(0, 0x34, (UINT8);(0x41<<1), data_2, 2));
        uart_printf("I2C write suc(1);\r\n");
    else
        uart_printf("I2C write error(1);\r\n");
    sAPI_I2CRead(0, 0x34, (UINT8);(0x41<<1), receiveData, 2);
        uart_printf("I2C read 0x%x     0x%x\r\n", receiveData[0] , receiveData[1] );
…
```

13.2.3 sAPI_I2CReadEx

Receive i2c data(repeat start).

sAPI_I2CRead	
Prototype	SC_I2C_ReturnCode **sAPI_I2CReadEx**(int i2cChannel,UINT8 slaveAddress,UINT8 regAddress,UINT8 *receiveDataBuffer,UINT16 datasize);
Parameters	[in] **i2cChannel**: choose i2c controller [in] **slaveAddress** : The meaning of this parameter is to shift the 7-bit device address one bit to the left to get the 8bit address. Note: both read and write operations use the 7-bit real device address to shift one bit to the left. For example, the 7-bit address is 0x1a, and the parameter value is 0x34. [in] **regAddress**: 8bits register address of slave address. [out] **receiveDataBuffer**: data array which used to receive data. [in] **datasize**: the data lenghth which will be received.
Return value	SC_I2C_RC_OK: i2c read ok. SC_I2C_RC_NOT_OK: i2c read error.
Header	simcom_i2c.h

Examples

```
…
    if(SC_I2C_RC_OK != sAPI_I2CReadEx(0,SLAVE_ADDR,0x18,&reg_data,1))
    {
        sAPI_Debug("0x18 register read error");
    }
```

14 APIs for Flash

14.1 Overview of APIs for Flash

Interface	Description
sAPI_EraseFlashSector	Erase a specified Sector called "cus_data"
sAPI_WriteFlash	Write a specified Sector called "cus_data"
sAPI_ReadFlash	Read a specified Sector called "cus_data"

Note: We've expanded cus_data partition ranges from 8KB to 256KB.

14.2 Detailed Description of APIs for Flash

14.2.1 sAPI_EraseFlashSector

sAPI_EraseFlashSector	
Prototype	int **sAPI_EraseFlashSector**(unsigned int offset, unsigned int size);
Parameters	[in] offset: Where to start erasing (4K aligned);. [in] size: the number of erase bytes. Flash takes 4KB as a sector, and erasing is performed in the unit of sector. If the size is larger than 4KB, the next sector will be erased. For example, if you choose to erase 4097 bytes, you will actually erase the contents of 2 sectors, totaling 8KB bytes. Note: offset + size < 256KB.
Return value	0: ok Other: failed.
Header	simcom_flash.h

Examples

```
void sAPP_FlashRWdemo(void);
{
    int ret, ret1, ret2;
    ret1 =sAPI_EraseFlashSector(0, 4096); //sector_1
```

```
    ret2 =sAPI_EraseFlashSector(4096, 4096);    //sector_2

    if(ret1 != 0 || ret2 != 0); return;
    for(int i = 0; i < 10; i++);
    {
        ret = sAPI_WriteFlash(SECTOR_1 + i * sizeof(flash_test), (INT8*);&ftest, sizeof(flash_test););
        if(ret != 0); return;
    }

    flash_test fread_test[10] ;
    for(int i = 0; i < 10; i++);
    {
        ret = sAPI_ReadFlash(SECTOR_1 + i * sizeof(flash_test), (INT8*);&fread_test[i] , sizeof(flash_test));
        if(ret != 0); return;
        int rls = memcmp(&ftest, &fread_test[i] , sizeof(flash_test));
        sAPI_Debug("cmp %s\r\n", rls==0?"ok":"err");

    }
}
```

14.2.2 sAPI_WriteFlash

sAPI_WriteFlashSector	
Prototype	int **sAPI_WriteFlash**(unsigned int offset, char *buff, unsigned int size);
Parameters	[in] offset: Where to start writing. [in] buff: the data will to be writen. [in] size: the number of bytes to be writen. Note: offset + size < 256KB.
Return value	0: ok Other: failed.
Header	simcom_flash.h

Examples

```
void sAPP_FlashRWdemo(void);
{
    int ret, ret1, ret2;
    ret1 =sAPI_EraseFlashSector(0, 4096); //sector_1

    ret2 =sAPI_EraseFlashSector(4096, 4096);    //sector_2
```

```
    if(ret1 != 0 || ret2 != 0); return;
    for(int i = 0; i < 10; i++);
    {
        ret = sAPI_WriteFlash(SECTOR_1 + i * sizeof(flash_test), (INT8*);&ftest, sizeof(flash_test););
        if(ret != 0); return;
    }

    flash_test fread_test[10] ;
    for(int i = 0; i < 10; i++);
    {
        ret = sAPI_ReadFlash(SECTOR_1 + i * sizeof(flash_test), (INT8*);&fread_test[i] , sizeof(flash_test));
        if(ret != 0); return;
        int rls = memcmp(&ftest, &fread_test[i] , sizeof(flash_test));
        sAPI_Debug("cmp %s\r\n", rls==0?"ok":"err");

    }
}
```

14.2.3 sAPI_ReadFlash

sAPI_WriteFlashSector	
Prototype	int **sAPI_ReadFlash**(unsigned int offset, char *buff, unsigned int size);
Parameters	[in] offset: Where to start reading. [out] buff: data will to be received. [in] size: the number of bytes to be read. Note: offset + size < 256KB.
Return value	0: ok Other: failed.
Header	simcom_flash.h

Examples

```
void sAPP_FlashRWdemo(void);
{
    int ret, ret1, ret2;
    ret1 =sAPI_EraseFlashSector(0, 4096); //sector_1

    ret2 =sAPI_EraseFlashSector(4096, 4096);    //sector_2
```

```
    if(ret1 != 0 || ret2 != 0); return;
    for(int i = 0; i < 10; i++);
    {
        ret = sAPI_WriteFlash(SECTOR_1 + i * sizeof(flash_test), (INT8*);&ftest, sizeof(flash_test));
        if(ret != 0); return;
    }

    flash_test fread_test[10] ;
    for(int i = 0; i < 10; i++);
    {
        ret = sAPI_ReadFlash(SECTOR_1 + i * sizeof(flash_test), (INT8*);&fread_test[i] ,
sizeof(flash_test));
        if(ret != 0); return;
        int rls = memcmp(&ftest, &fread_test[i] , sizeof(flash_test));
        sAPI_Debug("cmp %s\r\n", rls==0?"ok":"err");

    }
}
```

15 APIs for File System of ASR1601 Platform

15.1 Overview of APIs for File System

Interface	Description
sAPI_FsDirProcessor	Make new folder or delete a folder
sAPI_FsFileOpen	Flie open
sAPI_FsFileRead	File read
sAPI_FsFileSeek	File seek
sAPI_FsFileWrite	File write
sAPI_FsFileClose	File close
sAPI_FsFileDelete	Delete a file
sAPI_FsFileRename	File rename
sAPI_FsFindFileAndGetSize	Find a file and get file size, If the file exists
sAPI_FsGetDiskSize	Get the total size and free size of file system
sAPI_FsIsPathExist	Check file path exists or not
sAPI_FsFileTraverse	List directories/files in current directory
sAPI_FsIsFileNameValid	Check if the file name is valid
sAPI_FsNameSeparate	Split the file path and file name in the string
sAPI_FsSync	Synchronize a file
sAPI_FsFileTell	Get the file position
sAPI_FsFileSave	Save data to a file fastly.

15.2 Detailed Description of APIs for File System

The file system is used to store files in a hierarchical(tree); structure, and there are some definitions and conventions to use the APIs.
Local storage space is mapped to "C:", "D:" for SD card.

NOTE

General rules for naming(both directories and files):

a): The length of actual fully qualified names of files(C:/); can not exceed 112.

b): The length of actual fully qualified names of directories and files(D:/); can not exceed 250.

c): Directory and file names can not include the following characters:

\ : * ? " < > | , ;

d): Between directory name and file/directory name, use character "/" as list separator, so it can not appear in directory name or file name.

If the last character of names is period "."; the flash(C:/); will auto delete this character; the SD card can support this character, but the compatibility is not good.

15.2.1 sAPI_FsDirProcessor

This command is used to create a new directory or delete a directory. Support "D:".

sAPI_FsDirProcessor	
Prototype	SCfsReturnCode **sAPI_FsDirProcessor**(int operation_num, SCfileNameInfo *pName_info);
Parameters	[in] **operation_num**: choose the opreation 0 make a new folder 1 delete a folder [in] **pName_info**: The structure pointer with folder name and path.
Return value	SC_FS_OK:Process successfully executed SC_FS_PATH_INVALID:Incorrect path SC_FS_PARAMETER_INVALID:Incorrect operation
Header	#include "simcom_file_system.h"

15.2.2 sAPI_FsFileOpen

This API is used to open or create a file and associate it with a stream, Support "C:", "D:".

sAPI_FsFileOpen	
Prototype	SCfileHandle **sAPI_FsFileOpen**(const SCfileNameInfo *pName_info, const char *pMode);
Parameters	[in] **pName_info**: The structure pointer with folder name and path. [in] **pMode**: const character string for type specification, r/w/a/b;

	rb:read only wb/ab:write only rb+/wb+/ab+: readwrite Note: "rb" and "wb" modes are usually used
Return value	A structure that contains the file stream and the disk symbol on which the file resides .
Header	#include "simcom_file_system.h"

15.2.3 sAPI_FsFileRead

Read some data to a buffer with specific length, Support "C:", "D:".

sAPI_FsFileRead	
Prototype	unsigned int **sAPI_FsFileRead**(char *buf, unsigned int len, SCfileHandle handle);
Parameters	[out] **buf**: pointer to buffer to place data read [in] **len**: the data length try to read from the file [in] **handle**: stream index for the file to be read.
Return value	actualRead:The data length of actually read
Header	#include "simcom_file_system.h"

15.2.4 sAPI_FsFileSeek

Sets the file position indicator of the file specified by stream. The new position, measured in bytes from the beginning of the file is obtained by adding offset to the position specified by wherefrom. Support "C:", "D:".

sAPI_FsFileSeek	
Prototype	SCfsReturnCode **sAPI_FsFileSeek**(SCfileHandle handle, long offset, int whereFrom);
Parameters	[in] **handle**: stream index for the file to be read. [in] **offset**: move size from the start address [in] **whereFrom**: argument can be FS_SEEK_BEGIN 0 FS_SEEK_CURREN 1 FS_SEEK_END 2
Return value	SC_FS_OK:Process successfully executed SC_FS_ERROR: Process failly executed
Header	#include "simcom_file_system.h"

15.2.5 sAPI_FsFileWrite

This API is used to write data to a file. Support "C:", "D:".

sAPI_FsFileWrite	
Prototype	unsigned int **sAPI_FsFileWrite(**const char *buf, unsigned int len, SCfileHandle handle);
Parameters	[in] **buf**: pointer to buffer to place data need write [in] **len**: the data length try to write to file [in] **handle**: stream index for the file to be read.
Return value	actualWrite :The data length of actually write
Header	#include "simcom_file_system.h"

15.2.6 sAPI_FsFileClose

This API is used to close a file stream. Support "C:", "D:".

sAPI_FsFileClose	
Prototype	SCfsReturnCode **sAPI_FsFileClose**(SCfileHandle handle);
Parameters	[in] **handle**: stream index for the file to be closed.
Return value	SC_FS_OK :Process successfully executed SC_FS_ERROR: Process failly executed
Header	#include "simcom_file_system.h"

15.2.7 sAPI_FsFileRename

This API is used to rename a file. Support "C:", "D:".

sAPI_FsFileRename	
Prototype	SCfsReturnCode **sAPI_FsFileRename(**const SCfileNameInfo *pFileOld, const SCfileNameInfo *pFileNew);
Parameters	[in] **pFileOld**: the pointer of old file name info [in] **pFileNew**: the pointer of new file name info.
Return value	SC_FS_OK : Process successfully executed SC_FS_ERROR: Process failly executed SC_FS_PATH_INVALID: The path error SC_FS_PARAMETER_INVALID: The file is not on Disk "C" or "D"
Header	#include "simcom_file_system.h"

NOTE

On 1802S series module SD card operating not supported for this API.

15.2.8 sAPI_FsFindFileAndGetSize

This API is used to find a file and get file size. Support "C:", "D:".

sAPI_FsFindFileAndGetSize	
Prototype	SCfsReturnCode **sAPI_FsFindFileAndGetSize**(const SCfileNameInfo * pName_info, unsigned long * size);
Parameters	[in] **pName_info**: The structure pointer with folder name and path. [out] **size**: the found file size, if failed the value is 0.
Return value	SC_FS_OK :Process successfully executed SC_FS_ERROR: Process failly executed SC_FS_FILE_NAME_INVALID: File name is too long
Header	#include "simcom_file_system.h"

15.2.9 sAPI_FsFileDelete

This API is used to remove the file from existing files list. Support "C:", "D:".

sAPI_FsFileDelete	
Prototype	SCfsReturnCode **sAPI_FsFileDelete**(const SCfileNameInfo *pName_info);
Parameters	[in] **pName_info**: The structure pointer with folder name and path.
Return value	SC_FS_OK : Process successfully executed SC_FS_ERROR: Process failly executed SC_FS_FILE_NAME_INVALID: File name is too long
Header	#include "simcom_file_system.h"

15.2.10 sAPI_FsGetDiskSize

This API is used to Get file system space size. Support "C:", "D:".

sAPI_FsGetDiskSize	
Prototype	SCfsReturnCode **sAPI_FsGetDiskSize**(const char *pPath_in, INT64 *total_size, INT64 *free_size);
Parameters	[in] **pPath_in**: the path("C:/" or "D:/"); [out] **total_size**: the total storage capacity [out] **free_size**: the free storage capacity
Return value	SC_FS_OK : Process successfully executed SC_FS_ERROR: Process failly executed
Header	#include "simcom_file_system.h"

15.2.11 sAPI_FsIsPathExist

This API is used to Check the file path exiting or not. Support "C:", "D:".

sAPI_FsIsPathExist	
Prototype	BOOL **sAPI_FsIsPathExist**(const char *pPath_in);
Parameters	[in] **pPath_in**: the path("C:/" or "D:/ * ");
Return value	TRUE: File path exists FALSE: File path does not exist
Header	#include "simcom_file_system.h"

15.2.12 sAPI_FsFileTraverse

This API is used to list informations of directories and/or files in current directory. Support "C:", "D:".

sAPI_ FsFileTraverse	
Prototype	SCfsReturnCode **sAPI_FsFileTraverse**(const char *pPath_in, char *fileinfo, char *dirinfo);
Parameters	[in] **pPath_in**: the path("C:/" or "D:/"); [out] **fileinfo**: pointer to a buffer to place the traversal filename [out] **dirinfo**: pointer to a buffer to place the traversal subdirectory name
Return value	SC_FS_OK : Process successfully executed SC_FS_ERROR: Process failly executed
Header	#include "simcom_file_system.h"

15.2.13 sAPI_FsIsFileNameValid

This API is used to check if the file name is valid. Support "C:", "D:".

sAPI_ FsIsFileNameValid	
Prototype	SCfsReturnCode **sAPI_ FsIsFileNameValid(**const SCfileNameInfo *pName_info);
Parameters	[in] **pName_info**: The structure pointer with folder name and path.
Return value	SC_FS_OK : File name is valid other: File name is invalid
Header	#include "simcom_file_system.h"

15.2.14 sAPI_FsNameSeparate

This API is used to split the file path and file name in the string. Support "C:", "D:".

sAPI_ FsNameSeparate	
Prototype	SCfsReturnCode **sAPI_FsNameSeparate**(SCfileNameInfo *pName, char *pFull);
Parameters	[in] **pFull**: A string containing the path and filename [out] **pName_info**: The structure pointer with folder name and path.
Return value	SC_FS_OK : Process successfully executed other: Process failly executed
Header	#include "simcom_file_system.h"

15.2.15 sAPI_FsSync

This API is used to synchronize a file. Support "C:".

sAPI_FsSync	
Prototype	SCfsReturnCode sAPI_FsSync(ScfileHandle handle)
Parameters	[in] **handle**: stream index for the file.
Return value	SC_FS_OK :Process successfully executed SC_FS_ERROR: Process failly executed
Header	#include "simcom_file_system.h"

15.2.16 sAPI_FsFileTell

This API is used to get the number of bytes offset from the beginning of the file from the current position of the file position pointer. Support "C:", "D:".

sAPI_FsFileTell	
Prototype	int **sAPI_FsFileTell**(SCfileHandle handle);
Parameters	[in] **handle**: stream index for the file
Return value	file position
Header	#include "simcom_file_system.h"

15.2.17 sAPI_FsFileSave

This API is used to save data to a file fastly. Support "C:", "D:".

sAPI_FsFileSave	
Prototype	unsigned int **sAPI_FsFileSave**(BOOL covering, const char *filename, const char *data, unsigned int length)
Parameters	[in] **covering**: whether to overwrite a file of the same name [in] **filename**: full file name [in] **data**: pointer to buffer to place data need write [in] **length**: the data length try to write to file
Return value	actualWrite :The data length of actually write
Header	#include "simcom_file_system.h"

15.3 Result codes

15.3.1 Description of <err>s

<err>	Description
0	Operation succeeded
1	Failed
2	Invalid path

3	Invalid path head
4	Invalid file name
5	Empty parameter
6	Invalid parameter
7	Not support non-ascii code

15.4 Demo for File System

```
#include "simcom_file_system.h"


#define BUFF_LEN 100

int ret = 0;

SCfileHandle file_hdl = {0};

SCfileNameInfo file_name = {0};

char buff[BUFF_LEN] = {0};

char pTemBuffer[MAX_SD_FILE_PATH_LENGTH] ;

UINT32 buff_data_len = 0;

UINT32 actul_write_len = 0;

UINT32 actul_read_len = 0;

INT64 total_size = 0;

INT64 free_size = 0;

memcpy(file_name.name, "test_file.txt", strlen("test_file.txt"));

memcpy(file_name.path, "c:/", strlen("c:/"));


sAPI_FsGetDiskSize(file_name.path, &total_size, &free_size);

sAPI_Debug("total_size= %d, free_size= %d", total_size, free_size);


file_hdl = sAPI_FsFileOpen(&file_name, "wb");

sAPI_Debug("flie_hdl.loc= %d", file_hdl.loc);

memcpy(buff, "12345678", 8);

buff_data_len = 8;

actul_write_len = sAPI_FsFileWrite(buff, buff_data_len, file_hdl);
```

```
ret = SIM_FileClose(file_hdl);


buff_data_len = 0;
ret = sAPI_FsFindFileAndGetSize(&file_name, &buff_data_len);
file_hdl = sAPI_FsFileOpen(&file_name, "rb");
memset(buff, 0, BUFF_LEN);
actul_read_len = sAPI_FsFileRead(buff, buff_data_len, file_hdl);
ret = sAPI_FsFileClose(file_hdl);


actul_write_len = 0;
char full_name[MAX_SD_FILE_PATH_LENGTH] = {0};
memset(file_name.name, 0, MAX_SD_FILE_PATH_LENGTH);
memset(file_name.path, 0, MAX_SD_FILE_PATH_LENGTH);
memcpy(file_name.path, "c:/", strlen("c:/"););
memcpy(file_name.name, "file_seek_test", strlen("file_seek_test"););
sprintf(full_name, "%s%s", file_name.path, file_name.name);
buff_data_len =    strlen("123456789abcdef");
memset(buff, 0, BUFF_LEN);
memcpy(buff, "123456789abcdef", buff_data_len);
actul_write_len = sAPI_FsFileSave(1, full_name, buff, buff_data_len); //input full file name


file_hdl = sAPI_FsFileOpen(&file_name, "rb");
ret = sAPI_FsFileSeek(file_hdl, 9, SIM_SEEK_BEGIN);
memset(buff, 0, BUFF_LEN);
actul_read_len = sAPI_FsFileRead(buff, 4, file_hdl);
ret = sAPI_FsFileClose(file_hdl);
ret = sAPI_FsFileDelete(&file_name);
```

16 APIs for Internet Service

16.1 Overview of APIs for HTP and NTP

Interface	Description
sAPI_HtpSrvConfig	Add or remove htp server informotion
sAPI_HtpUpdate	Updata system time by htp protocol
sAPI_NtpUpdate	Update system time by ntp protocal
sAPI_GetSysLocalTime	Get system local time
sAPI_SetSysLocalTime	Set system local time

16.2 Detailed Description of APIs for HTP and NTP

16.2.1 sAPI_HtpSrvConfig

This API is used to add or delete HTP server information. There are maximum 16 HTP servers.

sAPI_HtpSrvConfig	
Prototype	SChtpReturnCode **sAPI_HtpSrvConfig**(SChtpOperationType commad_type, char *return_string, char *cmd, char* host_or_idx, int host_port, int http_version, char *proxy, int proxy_port);
Parameters	[in] **commad_type**: SC_HTP_OP_SET set parameters, add or remove addr SC_HTP_OP_GET get current addrs in the list [out] **return_string**: get the addr list, availiable only for SC_HTP_OP_GET [in] **cmd**: This command to operate the HTP server list. "ADD" add a HTP server item to the list "DEL" delete a HTP server item from the list [in] **host_or_idx**: If the <cmd> is "ADD", this field is the same as <host>, length is 0-255, needs quotation marks; If the <cmd> is "DEL", this field is the index of the HTP server item to be deleted from the list, does not need quotation marks. [in] **port**: the HTP server port, the range is(1-65535); .

	[in] **http_version**: The HTTP version of the HTP server: 0 HTTP 1.0 1 HTTP 1.1 [in] **proxy**: The proxy address, length is 1-255. [in] **proxy_port**: The port of the proxy, the range is(1-65535);.
Return value	SC_HTP_OK: Process successfully executed SC_HTP_ERROR: Htp process is busy SC_HTP_INVALID_PARAM: parameter error
Header	#include "simcom_htp_client.h"

Examples

```
…
sAPI_HtpSrvConfig(SC_HTP_OP_SET, NULL, "ADD", "www.baidu.com", 80, 1, NULL, 0);
sAPI_HtpSrvConfig(SC_HTP_OP_GET, buff, NULL, NULL, 0, 0, NULL, 0);
…
```

16.2.2 sAPI_HtpUpdate

This API is used to updating date time using HTP protocol.

sAPI_HtpUpdate	
Prototype	SChtpReturnCode **sAPI_HtpUpdate**(sMsgQRef magQ_urc);
Parameters	[in] **magQ_urc**: the message queue, the final result will send to this message queue as urc.
Return value	SC_HTP_OK: Process successfully executed SC_HTP_ERROR: Htp process is busy SC_HTP_INVALID_PARAM: Parameter error SC_HTP_NETWORK_ERROR: Pdp active failed
Header	#include "simcom_htp_client.h"

Examples

```
…
sAPI_HtpUpdate(urc_htp_msgq);
…
```

16.2.3 sAPI_NtpUpdate

This API is used to update system time with NTP server.

sAPI_NtpUpdate	
Prototype	SCntpReturnCode **sAPI_NtpUpdate**(SCntpOperationType commad_type, char* host_addr, int time_zone, sMsgQRef magQ_urc);
Parameters	[in] **commad_type**: SC_NTP_OP_SET, set host addr SC_NTP_OP_GET, get current host addr SC_NTP_OP_EXC, update sys time by ntp protocol [in/out] **host_addr**: NTP server addr, for SC_NTP_OP_GET, a string of server addr will copy to this para. [in] **time_zone**: the local time zone, (-47 to 48);default is 32 [in] **magQ_urc**: the message queue, the final result will send to this message queue as urc.
Return value	SC_NTP_OK: Process successfully executed SC_NTP_ERROR_NETWORK_FAIL: Pdp active failed SC_NTP_ERROR_INVALID_PARAM: Parameter error
Header	#include "simcom_ntp_client.h"

NOTE

Part of application, some parameters in this API are unnecessary, it shoule be set as NULL or 0.

16.2.4 sAPI_GetSysLocalTime

This API is used to get system time, you can use currUtcTime.tm_year, currUtcTime.tm_mon...

sAPI_GetSysLocalTime	
Prototype	void **sAPI_GetSysLocalTime**(tm_rtc *currUtcTime);

Parameters	[in/out] **currUtcTime**: typedef struct sys_time {
	int tm_sec; //seconds [0, 59]
	int tm_min; //minutes [0, 59]
	int tm_hour; //hour [0, 23]
	int tm_mday; //day of month [1, 31]
	int tm_mon; //month of year [1, 12]
	int tm_year; // since 1970
	int tm_wday; // sunday = 0
	}tm_rtc;
	tm_rtc currUtcTime;

Return value	None
Header	#include "simcom_ntp_client.h"

Examples

```
…
sAPI_NtpUpdate(SC_NTP_OP_SET, "120.25.108.11", 32, NULL);
sAPI_NtpUpdate(SC_NTP_OP_EXC, NULL, 0, urc_ntp_msgq);
…
```

16.2.5 sAPI_SetSysLocalTime

This API is used to set system local time.

sAPI_SetSysLocalTime	
Prototype	void **sAPI_SetSysLocalTime**(char* timeStr);
Parameters	[in/out] timeStr String format is "yy/MM/dd, hh:mm:ss" yy:year(two last digits); MM:month dd:day hh:hour mm:minute ss:second;
Return value	None
Header	#include "simcom_ntp_client.h"

Examples

```
char *timeStr="14/01/01, 02:14:36";
sAPI_GetSysLocalTime(timeStr);
…
sAPI_NtpUpdate(SC_NTP_OP_SET, "120.25.108.11", 32, NULL);
sAPI_NtpUpdate(SC_NTP_OP_EXC, NULL, 0, urc_ntp_msgq);
…
```

16.3 Result Codes

16.3.1 Description of <err>s of HTP

<err>	Description
0	Operation succeeded
1	Unknown error
2	Wrong parameter
3	Wrong date and time calculated
4	Network error

16.3.2 Description of <err>s of NTP

<err>	Description
0	Operation succeeded
1	Unknown error
2	Wrong parameter
3	Wrong date and time calculated
4	Network error
5	Time zone error
6	Time out error

17 APIs for TCP/IP

17.1 Overview of APIs for TCP/IP

Interface	Description
sAPI_TcpipSocket	Create an endpoint for communication
sAPI_TcpipBind	Bind a name to a socket
sAPI_TcpipListen	Listen for connections on a socket
sAPI_TcpipAccept	Accept a connection on a socket
sAPI_TcpipConnect	Initiate a connection on a socket
sAPI_TcpipSend	Send a message on a socket
sAPI_TcpipRecv	Receive a message from a socket
sAPI_TcpipSendto	Send a message on a socket
sAPI_TcpipRecvfrom	Receive a message from a socket
sAPI_TcpipClose	Close a file descriptor
sAPI_TcpipShutdown	Shut down part of a full-duplex connection
sAPI_TcpipGetsockname	Get socket name
sAPI_TcpipGetpeername	Get name of connected peer socket
sAPI_TcpipGetsockopt	Get options on sockets
sAPI_TcpipSetsockopt	Set options on sockets
sAPI_TcpipIoctlsocket	Control device
sAPI_TcpipGethostbyname	Returns a structure of type hostent for the given host name
sAPI_TcpipInetAddr	Converts the Internet host address cp from ipv4 numbers-and-dots notation into binary data in network byte order
sAPI_TcpipHtons	Converts the unsigned short integer from host byte order to network byte order
sAPI_TcpipNtohs	Converts the unsigned short integer netshort from network byte order to host byte order
sAPI_TcpipInetNtoa	Converts the Internet host address in, given in network byte order, to a string in ipv4 dotted-decimal notation

17.2 Detailed Description of APIs for TCP/IP

17.2.1 sAPI_TcpipGetErrno

Get the tcpip errno.

sAPI_TcpipGetErrno	
Prototype	INT32 **sAPI_TcpipGetErrno**(void);
Parameters	[in] **void**
Return value	lwip_errno
Header	#include "simcom_tcpip.h"

Examples

Refer to demo for TCP/IP.

17.2.2 sAPI_TcpipPdpActive

Activate the PDP context for TCPIP.

sAPI_TcpipPdpActive	
Prototype	INT32 **sAPI_TcpipPdpActive**(int **cid**, int **channel**);
Parameters	[in] **cid**: A numeric parameter which specifies a particular PDP context definition. TCPIP cid is 1. [in] **channel**: The data transmission channel.
Return value	0: success to activate the PDP context for TCPIP -1: fail to activate the PDP context for TCPIP
Header	#include "simcom_tcpip.h"

Examples

Refer to demo for TCP/IP.

17.2.3 sAPI_TcpipPdpDeactive

Deactive the PDP context for TCPIP.

sAPI_TcpipPdpDeactive

Prototype	INT32 **sAPI_TcpipPdpDeactive**(int **cid**, int **channe**);
Parameters	[in] **cid**: A numeric parameter which specifies a particular PDP context definition. TCPIP cid is 1. [in] **channel**: The data transmission channel.
Return value	0: success to deactive the PDP context for TCPIP -1: fail to deactive the PDP context for TCPIP
Header	#include "simcom_tcpip.h"

Examples

Refer to demo for TCP/IP.

17.2.4 sAPI_TcpipPdpDeactiveNotify

Notifies when the PDP is deactivated.

sAPI_TcpipPdpDeactiveNotifyDect	
Prototype	INT32 **sAPI_TcpipPdpDeactiveNotify**(sMsgQRef **msgQ**);
Parameters	[in] **msgQ**: results will be returned by msgQ
Return value	0: success to notifies when the PDP is deactivated -1: fail to notifies when the PDP is deactivated
Header	#include "simcom_tcpip.h"

Examples

```
sAPI_TcpipPdpDeactiveNotify(tcp_msgq);
```

17.2.5 sAPI_TcpipGetSocketPdpAddr

Get the PDP address for TCPIP.

sAPI_TcpipGetSocketPdpAddr	
Prototype	INT32 **sAPI_TcpipGetSocketPdpAddr**(int **cid**, int **channel**, struct simcom_ip_info ***info**);
Parameters	[in] **cid**: A numeric parameter which specifies a particular PDP context

	definition.TCPIP cid is 1. [in] **channel**: The data transmission channel. [in] **info**: A pointer to a structure containing IP address information.
Return value	0: dectivate the PDP context for TCPIP success -1: Dectivate the PDP context for TCPIP fail
Header	#include "simcom_tcpip.h"

Examples

Refer to demo for TCP/IP.

17.2.6 sAPI_TcpipSocket

Creates an endpoint for communication and returns a descriptor

sAPI_TcpipSocket	
Prototype	INT32 **sAPI_TcpipSocket**(INT32 domain, INT32 type, INT32 protocol);
Parameters	[in] **domain**: specifies a communication domain. [in] **type**: specifies the communication semantics [in] **protocol**: specifies a particular protocol to be used with the socket
Return value	On success, a file descriptor for the new socket is returned. -1: fail to receive a message from a socket, and errnao is set appropriately.
Header	#include "simcom_tcpip.h"

Examples

Refer to demo for TCP/IP.

17.2.7 sAPI_TcpipBind

Bind a name to a socket

sAPI_TcpipBind	
Prototype	INT32 **sAPI_TcpipBind**(INT32 sockfd, const SCsockAddr *addr, UINT32 addrlen);
Parameters	[in] **sockfd**: file descriptor. [in] **addr** assigns the address specified by addr

	[in] **addrlen**: specifies the size of addr, in bytes
Return value	0: success to bind a name to a socket. -1: fail to bind the name to the socket.
Header	#include "simcom_tcpip.h"

Examples

Refer to demo for TCP/IP.

17.2.8 sAPI_TcpipListen

Listen for connections on a socket

sAPI_TcpipListen	
Prototype	INT32 **sAPI_TcpipListen**(INT32 sockfd, INT32 backlog);
Parameters	[in] **scokfd**: file descriptor. [in] **backlog**: defines the maximum length to which the queue of pending connections
Return value	0: success to listen for connections on a socket -1: fail to listen for connections on a socket
Header	#include "simcom_tcpip.h"

Examples

Refer to demo for TCP/IP.

17.2.9 sAPI_TcpipAccept

Accept a connection on a socket

sAPI_TcpipAccept	
Prototype	INT32 **sAPI_TcpipAccept**(INT32 sockfd, SCsockAddr *addr, UINT32 *addrlen);
Parameters	[in] **scokfd**: file descriptor. [in] **addr**: address [in] **addrlen**: specifies the size of addr, in bytes
Return value	On success, return a nonnegative integer that is a descriptor for the accepted socket. On error, -1 is returned, and errno is set appropriately

Header	#include "simcom_tcpip.h"

Examples

Refer to demo for TCP/IP.

17.2.10 sAPI_TcpipConnect

Initiate a connection on a socket

sAPI_TcpipConnect	
Prototype	INT32 **sAPI_TcpipConnect**(INT32 sockfd, const SCsockAddr *addr, UINT32 addrlen);
Parameters	[in] **scokfd**: file descriptor. [in] **addr**: address [in] **addrlen**: specifies the size of addr, in bytes
Return value	0: success to initiate a connection on a socket -1: fail to initiate a connection on a socket
Header	#include "simcom_tcpip.h"

Examples

Refer to demo for TCP/IP.

17.2.11 sAPI_TcpipSend

Send a message on a socket.The api may be used only when the socket is in a connected state

NOTE

When the message does not fit into the send buffer of the socket, sAPI_TCPIPSend normally blocks, unless the socket has been placed in nonblocking I/O mode.

sAPI_TcpipSend

Prototype	INT32 **sAPI_TcpipSend**(INT32 sockfd, const void *buf, INT32 len, INT32 flags);
Parameters	[in] **scokfd**: file descriptor. [in] **buf**: specifies the send buf [in] **len**: specifies the size of buf, in bytes [in] **flags**: specifies a particular protocol to be used with the socket.It is generally set to 0
Return value	On success, return the number of characters sent. 0: the peer has performed an orderly shutdown. -1: fail to success to send a message on the socket.
Header	#include "simcom_tcpip.h"

Examples

Refer to demo for TCP/IP.

17.2.12 sAPI_TcpipRecv

Receive a message from a socket

sAPI_TcpipRecv	
Prototype	INT32 **sAPI_TcpipRecv**(INT32 sockfd, void *buf, INT32 len, INT32 flags);
Parameters	[in] **scokfd**: file descriptor. [in] **buf**: specifies the send buf [in] **len**: specifies the size of buf, in bytes [in] **flags**: specifies a particular protocol to be used with the socket.It is generally set to 0
Return value	On success,return the number of bytes received. 0: the peer has performed an orderly shutdown. -1: fail to receive a message from a socket, and errnao is set appropriately.
Header	#include "simcom_tcpip.h"

Examples

Refer to demo for TCP/IP.

17.2.13 sAPI_TcpipSendto

Send a message on a socket

sAPI_TcpipSendto	
Prototype	INT32 **sAPI_TcpipSendto**(INT32 sockfd, const void *buf, INT32 len, INT32 flags, const SCsockAddr *dest_addr, UINT32 addrlen);
Parameters	[in] **scokfd**: file descriptor. [in] **buf**: specifies the send buf [in] **len**: specifies the size of buf, in bytes [in] **flags**: specifies a particular protocol to be used with the socket.It is generally set to 0 [in] **dest_addr**: address [in] **addrlen**: specifies the size of addr, in bytes
Return value	On success, return the number of characters sent. -1: fail to send a message on a socket, and errnao is set appropriately.
Header	#include "simcom_tcpip.h"

Examples

Refer to demo for TCP/IP.

17.2.14 sAPI_TcpipRecvfrom

Receive a message from a socket

sAPI_TcpipRecvfrom	
Prototype	INT32 **sAPI_TcpipRecvfrom**(INT32 sockfd, void *buf, INT32 len, INT32 flags, SCsockAddr *src_addr, UINT32 *addrlen);
Parameters	[in] **scokfd**: file descriptor. [in] **buf**: specifies the send buf [in] **len**: specifies the size of buf, in bytes [in] **flags**: specifies a particular protocol to be used with the socket.It is generally set to 0 [in] **src_addr**: address [in] **addrlen**: specifies the size of addr, in bytes
Return value	On success,return the number of bytes received. -1: fail to receive a message from a socket, and errnao is set appropriately.
Header	#include "simcom_tcpip.h"

Examples

Refer to demo for TCP/IP.

17.2.15 sAPI_TcpipClose

Close a file descriptor

sAPI_TcpipClose	
Prototype	INT32 **sAPI_TcpipClose**(INT32 fd);
Parameters	[in] **sockfd**: file descriptor.
Return value	0: success to close a file descriptor -1: fail to close a file descriptor, and errnao is set appropriately.
Header	#include "simcom_tcpip.h"

Examples

Refer to demo for TCP/IP.

17.2.16 sAPI_TcpipShutdown

Shut down part of a full-duplex connection

sAPI_TcpipShutdown	
Prototype	INT32 **sAPI_TcpipShutdown**(INT32 sockfd, INT32 how);
Parameters	[in] **sockfd**: file descriptor. [in] **how**: specifies the communication semantics
Return value	0: success to shut down part of a full-duplex connection -1: fail to shut down part of a full-duplex connection, and errnao is set appropriately.
Header	#include "simcom_tcpip.h"

Examples

Refer to demo for TCP/IP.

17.2.17 sAPI_TcpipGetsockname

Get socket name

sAPI_TcpipGetsockname	
Prototype	INT32 **sAPI_TcpipGetsockname**(INT32 sockfd, SCsockAddr *addr, UINT32 *addrlen);
Parameters	[in] **sockfd**: file descriptor. [in] **addr**: address [in] **addrlen**: specifies the size of addr, in bytes
Return value	0: success to get socket name -1: fail to get socket name
Header	#include "simcom_tcpip.h"

Examples

Refer to demo for TCP/IP.

17.2.18 sAPI_TcpipGetpeername

Get name of connected peer socket

sAPI_TcpipGetpeername	
Prototype	INT32 **sAPI_TcpipGetpeername**(INT32 sockfd, SCsockAddr *addr, UINT32 *addrlen);
Parameters	[in] **sockfd**: file descriptor. [in] **addr**: address [in] **addrlen**: specifies the size of addr, in bytes
Return value	0: success to get name of connected peer socket -1: fail to get name of connected peer socket
Header	#include "simcom_tcpip.h"

Examples

```
struct sockaddr_in sa;
int len = sizeof(sa);
if( !getpeername(sockfd, (struct sockaddr *);&sa, &len););
{
    printf( " opposite IP:%s ", inet_ntoa(sa.sin_addr););
    printf( " opposite PORT:%d ", ntohs(sa.sin_port););
}
```

17.2.19 sAPI_TcpipGetsockopt

Get the current value of an option for any type, any state socket interface, and store the result in OptVAL.

sAPI_TcpipGetsockopt	
Prototype	INT32 **sAPI_TcpipGetsockopt**(INT32 sockfd, INT32 level, INT32 optname, void *optval, UINT32 *optlen);
Parameters	[in] **scokfd**: file descriptor. [in] **level**: When manipulating socket options, the level at which the option resides and the name of the option must be specified. [in] **optname**: specifies a particular protocol to be used with the socket [in] **optval**: specifies a particular protocol to be used with the socket [in] **optlen**: specifies a particular protocol to be used with the socket
Return value	0: success to get options on sockets -1: fail to get options on sockets
Header	#include "simcom_tcpip.h"

Examples

Refer to demo for TCP/IP.

17.2.20 sAPI_TcpipSetsockopt

Set options on sockets

sAPI_TcpipSetsockopt	
Prototype	INT32 **sAPI_TcpipSetsockopt**(INT32 sockfd, INT32 level, INT32 optname, const void *optval, UINT32 optlen);
Parameters	[in] **scokfd**: file descriptor. [in] **level**: When manipulating socket options, the level at which the option resides and the name of the option must be specified. [in] **optname**: specifies a particular protocol to be used with the socket [in] **optval**: specifies a particular protocol to be used with the socket [in] **optlen**: specifies a particular protocol to be used with the socket
Return value	0: success to set options on sockets -1: fail to set options on sockets
Header	#include "simcom_tcpip.h"

Examples

Refer to demo for TCP/IP.

17.2.21 sAPI_TcpipIoctlsocket

Control the mode of the socket interface.Any set of interfaces that can be used in any state.It is used to obtain operational parameters related to the socket interface, regardless of the specific protocol or communication subsystem.

sAPI_TcpipIoctlsocket	
Prototype	INT32 **sAPI_TcpipIoctlsocket**(INT32 fd, INT32 level, INT32 value);
Parameters	[in] **fd**: specifies a communication domain. [in] **level**: specifies the communication semantics [in] **value**: specifies a particular protocol to be used with the socket
Return value	0: success to control device -1: fail to control device
Header	#include "simcom_tcpip.h"

Examples

```
int iResult;
u_long iMode = 0;
iResult = ioctlsocket(socket_fd, FIONBIO, &iMode);
if(0 != iResult);
{
    printf("ioctlsocket failed with error: %ld\n", iResult);
}
```

17.2.22 sAPI_TcpipGethostbyname

Returns a structure of type hostent for the given host name

sAPI_TcpipGethostbyname	
Prototype	SChostent * **sAPI_TcpipGethostbyname**(const INT8 *name);
Parameters	[in] **name**: specifies a communication domain.
Return value	On success, returns a structure of type hostent. On error, a NULL pointer is returned
Header	#include "simcom_tcpip.h"

Examples

Refer to demo for TCP/IP.

17.2.23 sAPI_TcpipSelect

The function allow a program to monitor multiple file descriptors, waiting until one or more of the file descriptors become "ready" for some class of I/O operation.

sAPI_TcpipSelect	
Prototype	INT32 **sAPI_TcpipSelect**(INT32 nfds, SCfdSet *readfds, SCfdSet *writefds, SCfdSet *exceptfds, SCtimeval *timeout);
Parameters	[in] **nfds**: specifies a communication domain. [in] **readfds**: specifies a communication domain. [in] **writefds**: specifies a communication domain. [in] **exceptfds**: specifies a communication domain. [in] **timeout**: specifies a communication domain.
Return value	a file descriptor for the new socket is returned On success, return the number of file descriptors contained in the three returned descriptor sets(that is, the total number of bits that are set in readfds, writefds, exceptfds); which may be zero if the timeout expires before anything interesting happens. -1: fail to monitor multiple file descriptors, and errno is set appropriately; the sets and timeout become undefined, so do not rely on their contents after an error.
Header	#include "simcom_tcpip.h"

Examples

Refer to demo for TCP/IP.

17.2.24 sAPI_TcpipInetAddr

Converts the Internet host address cp from IPv4 numbers-and-dots notation into binary data in network byte order

sAPI_TcpipInetAddr	
Prototype	UINT32 **sAPI_TcpipInetAddr**(const INT8 *cp);

Parameters	[in] **cp**: Internet host address.
Return value	0: success to convert the Internet host address cp -1: fail to convert the Internet host address cp
Header	#include "simcom_tcpip.h"

Examples

Refer to demo for TCP/IP.

17.2.25 sAPI_TcpipHtons

Converts the unsigned short integer hostshort from host byte order to network byte order

sAPI_TcpipHtons	
Prototype	UINT16 **sAPI_TcpipHtons**(UINT16 hostshort);
Parameters	[in] **hostshort**: specifies a communication domain.
Return value	0: success to convert the unsigned short integer hostshort -1: fail to convert the unsigned short integer hostshort
Header	#include "simcom_tcpip.h"

Examples

Refer to demo for TCP/IP.

17.2.26 sAPI_TcpipNtohs

Converts the unsigned short integer netshort from network byte order to host byte order

sAPI_TcpipNtohs	
Prototype	UINT16 **sAPI_TcpipNtohs**(UINT16 hostshort);
Parameters	[in] **nhostshort**: hostshort.
Return value	0: success to convert the unsigned short integer netshort -1: fail to convert the unsigned short integer netshort
Header	#include "simcom_tcpip.h"

Examples

Refer to demo for TCP/IP.

17.2.27 sAPI_TcpipInetNtoa

Converts the Internet host address in, given in network byte order, to a string in IPv4 dotted-decimal notation. The string is returned in a statically allocated buffer, which subsequent calls will overwrite.

sAPI_TcpipInetNtoa	
Prototype	INT8 ***sAPI_TcpipInetNtoa**(UINT32 in);
Parameters	[in] **in**: internet host address
Return value	NULL: fail to converts the Internet host address Other:success to converts the Internet host address
Header	#include "simcom_tcpip.h"

Examples

Refer to demo for TCP/IP.

17.2.28 sAPI_TcpipInetNtop

Converts network binary structures to ASCII type addresses.

sAPI_TcpipInetNtop	
Prototype	INT8 ***sAPI_TcpipInetNtop**(INT32 af, const void *src, INT8 *dst, UINT32 size);
Parameters	[in] **af**: address cluster [in] **src**:the source address [in] **dst**: converted data [in] **size**: the size of the dst
Return value	NULL: fail to converts the Internet host address Other:success to converts the Internet host address
Header	#include "simcom_tcpip.h"

Examples

Refer to demo for TCP/IP.

17.2.29 sAPI_TcpipGetSocketErrno

Get the errno of sockfd.

sAPI_TcpipGetSocketErrno	
Prototype	INT32 **sAPI_TcpipGetsocketErrno**(int sockfd);
Parameters	[in] **sockfd**: socket id
Return value	errno
Header	#include "simcom_tcpip.h"

Examples

Refer to demo for TCP/IP.

17.2.30 sAPI_TcpipHtonl

Converts the host byte order to network byte order.

sAPI_TcpipHtonl	
Prototype	UINT32 **sAPI_TcpipHtonl**(INT32 hostlong);
Parameters	[in] **hostlog**: 32 bit host byte data
Return value	32 bit network byte data
Header	#include "simcom_tcpip.h"

Examples

Refer to demo for TCP/IP.

17.2.31 sAPI_TcpipGetaddrinfo

Host name to address resolution.

sAPI_TcpipGetaddrinfo	
Prototype	int ***sAPI_TcpipGetaddrinfo**(const char *nodename, const char *servname, const struct addrinfo *hints, struct addrinfo **res);

Parameters	[in] **nodename**: address name [in] **servname**: address port [in] **hints**: null pointer or a pointer point to addrinfo struct [in] **res**: a pointer point to return value of addrinfo struct
Return value	0: success others: socket error code
Header	#include "simcom_tcpip.h"

Examples

Refer to demo for TCP/IP.

17.2.32 sAPI_TcpipGetaddrinfoWithPcid

Host name to address resolution.

sAPI_TcpipGetaddrinfoWithPcid	
Prototype	int ***sAPI_TcpipGetaddrinfoWithPcid**(const char *nodename, const char *servname, const struct addrinfo *hints, struct addrinfo **res, u8_t pcid);
Parameters	[in] **af**: address cluster [in] **src**:the source address [in] **dst**: converted data [in] **size**: the size of the dst [in] **pcid:** value pcid
Return value	0: success others: socket error code
Header	#include "simcom_tcpip.h"

Examples

Refer to demo for TCP/IP.

17.2.33 sAPI_TcpipFreeaddrinfo

Free the addrinfo pointer.

sAPI_TcpipFreeaddrinfo

Prototype	void **sAPI_TcpipFreeaddrinfo**(struct addrinfo *ati);
Parameters	[in] **ati**: addrinfo pointer
Return value	NULL
Header	#include "simcom_tcpip.h"

Examples

Refer to demo for TCP/IP.

17.2.34 sAPI_TcpipSocketWithCallback

Creates an endpoint for communication and returns a descriptor.

sAPI_TcpipSocketWithCallback	
Prototype	Int **sAPI_TcpipSocketWithCallback**(int domain, int type, int protocol, socket_callback callback);
Parameters	[in] **domain**: specifies a communication domain. [in] **type**: specifies the communication semantics [in] **protocol**: specifies a particular protocol to be used with the socket [in] **callback**: callback function
Return value	On success, a file descriptor for the new socket is returned. -1: fail to receive a message from a socket, and errnao is set appropriately.
Header	#include "simcom_tcpip.h"

Examples

Refer to demo for TCP/IP.

17.2.35 sAPI_TcpipInetPton

Convert IPv4 and IPv6 addresses from text to binary.

sAPI_TcpipInetPton	
Prototype	int **sAPI_TcpipInetPton**(int af, const char *src, void *dst);
Parameters	[in] **af**: address cluster [in] **src**:the source address [in] **dst**: converted data

Return value	NULL: fail to converts the Internet host address Other:success to converts the Internet host address
Header	#include "simcom_tcpip.h"

Examples

Refer to demo for TCP/IP.

17.3 Demo for TCP/IP

```
#include "simcom_tcpip.h"
#include "simcom_debug.h"

INT32 sAPI_TCPIPConnectTcpServerTestMain(INT8*ipstr, UINT16 port, UINT16 localPort);
{
    INT32 sockfd = -1;
    INT32 ret = -1;
    SChostent *host_entry = NULL;
    SCsockAddrIn sa;
    SCsockAddrIn server;
    INT32 keepalive = 1, keepidle = 10, keepcount = 2, keepINT32erval = 10;
    INT8 recvbuf[100] = {0};
    INT32 len = 0;
      INT8 ip[100] ;
    struct simcom_ip_info ip_info = {INVALID, 0, {0}};

    if(-1 == sAPI_TcpipPdpActive(1, 1););
    {
       sAPI_Debug("PDP active err");
       return ret;
    }
```

```
if(-1 == sAPI_TcpipGetSocketPdpAddr(1, 1, &ip_info););
{
   sAPI_Debug("PDP active err");
   sAPI_TcpipPdpDeactive(1, 1);
   return ret;
}

sAPI_Debug("PDP type = %d, ipv4 = %d, ipv6 = %s", ip_info.type, ip_info.ip4, ip_info.ip6);
sockfd = sAPI_TcpipSocket(SC_AF_INET, SC_SOCK_STREAM, 0);
if(sockfd < 0);
{
      sAPI_Debug("create socket err");
      goto err;
}

host_entry = sAPI_TcpipGethostbyname(ipstr);
if(host_entry == NULL);
{
      sAPI_Debug("DNS gethostbyname fail");
      goto err;
}

sAPI_Debug("DNS gethostbyname, Get %s ip %d.%d.%d.%d\n",
            ipstr, host_entry->h_addr_list[0] [0] & 0xff,
            host_entry->h_addr_list[0] [1] & 0xff, host_entry->h_addr_list[0] [2] & 0xff,
            host_entry->h_addr_list[0] [3] & 0xff);

ret = sAPI_TcpipSetsockopt(sockfd, SC_SOL_SOCKET, SC_SO_KEEPALIVE, (void *);&keepalive, sizeof(keepalive););
if(ret < 0);
      goto err;

ret = sAPI_TcpipSetsockopt(sockfd, SC_IPPROTO_TCP, SC_TCP_KEEPIDLE, (void*);&keepidle, sizeof(keepidle););
if(ret < 0);
```

```
        goto err;

    ret = sAPI_TcpipSetsockopt(sockfd, SC_IPPROTO_TCP, SC_TCP_KEEPCNT, (void *);&keepcount, sizeof(keepcount););
    if(ret < 0);
        goto err;

    ret = sAPI_TcpipSetsockopt(sockfd, SC_IPPROTO_TCP, SC_TCP_KEEPINTVL, (void *);&keepINT32erval, sizeof(keepINT32erval););
    if(ret < 0);
        goto err;


    ret = sAPI_TcpipGetsockopt(sockfd, SC_IPPROTO_TCP, SC_TCP_KEEPINTVL, (void *);&keepINT32erval, sizeof(keepINT32erval););
    if(ret < 0);
        goto err;

    if(localPort> 0);
    {
        sa.sin_family           = SC_AF_INET;
        sa.sin_port             = htons(localPort);
        sa.sin_addr.s_addr    = 0;
        if(sAPI_TcpipBind(sockfd, (const SCsockAddr *);&sa, sizeof(sa);); < 0);
        {
            sAPI_Debug("Bind local port fail errno[%d] ", sAPI_TcpipGetErrno(););
            goto err;
        }
    }

    server.sin_family = SC_AF_INET;
    server.sin_port = htons(port);
    server.sin_addr.s_addr= *(UINT32 *);host_entry->h_addr_list[0] ;
```

```
ret = sAPI_TcpipConnect(sockfd, &server, sizeof(SCsockAddr););
if(ret != 0);
{
    sAPI_Debug("connect server fail errno[%d] ", sAPI_TcpipGetErrno(););
    goto err;
}

sAPI_Debug("ret[%d] connect server sucess", ret);

//send hello to server

ret = sAPI_TcpipSend(sockfd, "hello", 5, 0);
if(ret < 0);
{
    sAPI_Debug("send hello to server    fail errno[%d] ", sAPI_TcpipGetErrno(););
    goto err;
}

memset(recvbuf, 0x0, sizeof(recvbuf););

ret = sAPI_TcpipRecv(sockfd, recvbuf, sizeof(recvbuf), 0);
if(ret < 0);
{
    sAPI_Debug("recv from server fail errno[%d] ", sAPI_TcpipGetErrno(););
    goto err;
}

sAPI_Debug("recv SUCESS byte:[%d]    recvbuf[%s] ", ret, recvbuf);

len = sizeof(sa);
ret = sAPI_TcpipGetsockname(sockfd, (SCsockAddr *);&sa, &len);
if(ret < 0);
```

```
        goto err;


    memset(ip, 0x0, sizeof(ip));
    if(sAPI_TcpipInetNtop(SC_AF_INET, &sa.sin_addr, ip, sizeof(ip)); == NULL);
        goto err;


    sAPI_Debug("host ip[%s] port[%d] ", ip, sAPI_TcpipNtohs(sa.sin_port));
err:
    if(sockfd>= 0);
    {
        sAPI_TcpipClose(sockfd);
        sAPI_TcpipPdpDeactive(1, 1);
    }


    return ret;


}
void sAPI_TCPIPListenTestMain();
{
    INT32 ret = -1;
    INT32 serverfd = 0, newfd = 0;
    SChostent *host_entry;
    SCsockAddrIn sa;
    SCsockAddrIn cliaddr;
    INT32 reuseport = 1;
    INT32 fdmax = -1;
    SCfdSet read_fds;
    INT32 i = 0;
    INT32 len = 0;


    INT32 serverport = 8888;
    INT32 backlog = 5;
    struct simcom_ip_info ip_info = {INVALID, 0, {0}};
```

```
if(-1 == sAPI_TcpipPdpActive(1, 1););
{
   sAPI_Debug("PDP active err");
   return ret;
}

if(-1 == sAPI_TcpipGetSocketPdpAddr(1, 1, &ip_info););
{
   sAPI_Debug("PDP active err");
   sAPI_TcpipPdpDeactive(1, 1);
   return ret;
}
serverfd = sAPI_TcpipSocket(SC_AF_INET, SC_SOCK_STREAM, 0);
if(serverfd < 0);
{
      sAPI_Debug("create socket err");
      goto err;
}

memset(&(sa), 0, sizeof(sa));
sa.sin_addr.s_addr = sAPI_TcpipInetAddr("127.0.0.1");

sAPI_Debug("s_addr[%d] ", sa.sin_addr.s_addr);

if(sa.sin_addr.s_addr == SC_IPADDR_None);
{
      goto err;
}
sa.sin_family          = SC_AF_INET;
sa.sin_port            =   sAPI_TcpipHtons(serverport);

ret = sAPI_TcpipSetsockopt(serverfd, SC_SOL_SOCKET, SC_SO_REUSEADDR, &reuseport, sizeof(reuseport));
```

```
if(ret < 0);
    goto err;



if(sAPI_TcpipBind(serverfd, (const SCsockAddr *);&sa, sizeof(sa);); < 0);
{
    sAPI_Debug("Bind local port fail errno[%d] ", sAPI_TcpipGetErrno(););
    goto err;
}

if(sAPI_TcpipListen(serverfd, backlog); < 0);
{
    sAPI_Debug("listen fail errno[%d] ", sAPI_TcpipGetErrno(););
    goto err;
}

SC_FD_ZERO(&read_fds);
SC_FD_SET(serverfd, &read_fds);
fdmax = serverfd+1;
ret = sAPI_TcpipSelect(fdmax, &read_fds, NULL, NULL, NULL);
if(ret == -1 || ret == 0);
{
    sAPI_Debug("select fail or timeout[%d] ", sAPI_TcpipGetErrno(););
    goto err;
}

sAPI_Debug("--- ret[%d] ", ret);

for(i = 0;i < fdmax;i++);
{
    if(!SC_FD_ISSET(i, &read_fds););
    {
        continue;
```

```
}

if(i == serverfd);
{
    char ipstr[100] ;
    UINT16 clientport = 0;



    len = sizeof(cliaddr);
    newfd = sAPI_TcpipAccept(serverfd, (SCsockAddr *);&cliaddr, (UINT32 *);&len);
    sAPI_Debug("--- newfd[%d] ", newfd);

    if(newfd < 0);
    {
        sAPI_Debug("accept fail [%d] ", sAPI_TcpipGetErrno(););
        goto err;
    }

    clientport = sAPI_TcpipNtohs(cliaddr.sin_port);
    memset(ipstr, 0x0, sizeof(ipstr););
    snprintf(ipstr, sizeof(ipstr), "%s", sAPI_TcpipInetNtoa(cliaddr.sin_addr.s_addr););

    sAPI_Debug("new client connect port[%d] ip[%d] newfd[%d] ", clientport, ipstr, newfd);

    ret = sAPI_TcpipSend(newfd, "hello client", strlen("hello client"), 0);
    if(ret < 0);
    {
        sAPI_Debug("send to client fail");
        goto err;
    }

    sAPI_TcpipClose(newfd);
    sAPI_TcpipPdpDeactive(1, 1);
```

```
                ret = 0;
            }

        }
 err:

        if(serverfd> 0);
        {
            sAPI_TcpipClose(serverfd);
            sAPI_TcpipPdpDeactive(1, 1);
        }

        return;
}

INT32 sAPI_UDPTestMain(INT32 localport, char *addr, UINT16 port);
{
        INT32 ret = -1;
        INT32 sockfd = 0;
        SCsockAddrIn sa;
        SCsockAddrIn server;
        SChostent *host_entry;
        UINT32 addrlen = 0;
        INT8 recvbuf[100] ;
        INT8 ipstr[100] ;
          struct simcom_ip_info ip_info = {INVALID, 0, {0}};

        host_entry = sAPI_TcpipGethostbyname(addr);
        if(host_entry == NULL);
        {
            sAPI_Debug("DNS gethostbyname fail");
            return -1;
         }
```

```
if(-1 == sAPI_TcpipPdpActive(1, 1););
{
   sAPI_Debug("PDP active err");
   return ret;
}
if(-1 == sAPI_TcpipGetSocketPdpAddr(1, 1, &ip_info););
{
   sAPI_Debug("PDP active err");
   sAPI_TcpipPdpDeactive(1, 1);
   return ret;
}

sAPI_Debug("PDP type %d, ipv4 = %d, ipv6 = %s", ip_info.type, ip_info.ip4, ip_info.ip6);
sAPI_Debug("localport[%d] port[%d] ", localport, port);

sockfd = sAPI_TcpipSocket(SC_AF_INET, SC_SOCK_DGRAM, 0);
if(sockfd < 0);
{
      sAPI_Debug("create socket err");
      goto err;
}

sa.sin_family = SC_AF_INET;
sa.sin_port = sAPI_TcpipHtons(port);
sa.sin_addr.s_addr= *(UINT32 *);host_entry->h_addr_list[0] ;

ret = sAPI_TcpipSendto(sockfd, "hello", 5, 0, (SCsockAddr*);&sa, sizeof(SCsockAddrIn););
if(ret < 0);
{
      sAPI_Debug("udp send data fail [%d] ", sAPI_TcpipGetErrno(););
      goto err;
}
```

```
    memset(recvbuf, 0x0, sizeof(recvbuf));;
    ret = sAPI_TcpipRecvfrom(sockfd, recvbuf, sizeof(recvbuf), 0, &server, &addrlen);
    if(ret < 0);
    {
        sAPI_Debug("udp recv data fail [%d] ", sAPI_TcpipGetErrno());;
        goto err;
    }
    memset(ipstr, 0x0, sizeof(ipstr));;
    snprintf(ipstr, "%s", sAPI_TcpipInetNtoa(server.sin_addr.s_addr));;

    sAPI_Debug("udp recv size ret[%d] [%s] ", ret, ipstr);
    sAPI_Debug("recvbuf[%s] ", recvbuf);
err:
    if(sockfd>= 0);
    {
        sAPI_TcpipClose(sockfd);
        sAPI_TcpipPdpDeactive(1, 1);
    }

    return ret;
}
```

18 APIs for HTTP(S)

18.1 Overview of APIs for HTTP(S)

Interface	Description
sAPI_HttpInit	Start HTTP service
sAPI_HttpTerm	Stop HTTP Service
sAPI_HttpPara	Set HTTP Parameters value
sAPI_HttpAction	HTTP Method Action
sAPI_HttpHead	Read the HTTP Header Information of Server Response
sAPI_HttpRead	Read the response information of HTTP Server
sAPI_HttpData	Trans the file's filename to sAPI_HttpAction
sAPI_HttpPostfile	Send HTTP Request to HTTP(S); server by File

18.2 Detailed Description of APIs for HTTP(S)

18.2.1 sAPI_HttpInit

This application interface is used to start HTTP service by activating PDP context.

sAPI_HttpInit	
Prototype	SC_HTTP_RETURNCODE **sAPI_HttpInit**(int channel, OSMsgQRef magQ_urc);
Parameters	[in] **channel**: The data transmission channel. [in] **magQ_urc**: The message queue for urc report, it is valid duiring whole http service.
Return value	SC_HTTPS_SUCCESS: start http service success SC_HTTPS_FAIL: start http service fail
Header	#include "simcom_https.h"

Examples

```
#include "simcom_https.h"

void HTTPINIT_APITest(void);
{
    static sMsgQRef ghttps_msgq_ret;
    SC_HTTP_RETURNCODE ret;

    ret = sAPI_HttpInit(1, ghttps_msgq_ret);        //channel : 0 - serial port, 1 - usb
    if(ret == SC_HTTP_SUCCESS);
    {
        ……                                          //The action after success
    }
    Else if(ret == SC_HTTP_FAIL);
    {
        ……                                          //The action after fail
    }
}
```

18.2.2 sAPI_HttpTerm

This application interface is used to stop HTTP service.

sAPI_HttpTerm	
Prototype	SC_HTTP_RETURNCODE **sAPI_HttpTerm**(void);
Parameters	None.
Return value	SC_HTTPS_SUCCESS: stop http service success SC_HTTPS_FAIL: stop http service fail
Header	#include "simcom_https.h"

Examples

```
#include "simcom_https.h"

void HTTPINIT_APITest(void);
{
    static sMsgQRef ghttps_msgq_ret;
    SC_HTTP_RETURNCODE ret;

    ret = sAPI_HttpTerm();                          //Stop http service
    if(ret == SC_HTTP_SUCCESS);
    {
```

```
        ……                                  //The action after success
    }
    Else if(ret == SC_HTTP_FAIL);
    {
        ……                                  //The action after fail
    }
}
```

18.2.3 sAPI_HttpPara

This application interface is used to set HTTP parameters value.

sAPI_HttpPara	
Prototype	SC_HTTP_RETURNCODE **sAPI_HttpPara**(char *command_type, char *parameters);
Parameters	[in] **command_type**: the different types of commands.such as:"URL", "CONNEC--TO", "RECVTO", "CONTENT", "ACCEPT", "SSLCFG", "USERDATA", "READMODE" refer to Defined Value. [in] **parameters**: the parameters value matched with its command
Return value	SC_HTTPS_SUCCESS: set http parameters success. SC_HTTPS_FAIL: set http parameters fail
Header	#include "simcom_https.h"

Defined Values

<url>	URL of network resource.String, start with "http://" or"https://" a);http://'server'/'path':'tcpPort'. b);https://'server'/'path':'tcpPort' "server": DNS domain name or IP address "path": path to a file or directory of a server "tcpPort": http default value is 80, https default value is 443.(can be omitted);
<conn_timeout>	Timeout for accessing server, Numeric type, range is 20-120s, default is 120s.
<recv_timeout>	Timeout for receiving data from server, Numeric type range is 2s-120s, default is 20s.
<content_type>	This is for HTTP "Content-Type" tag, String type, max length is 256, default is "text/plain".
<accept-type>	This is for HTTP "Accept-type" tag, String type, max length is 256, default is "*/*".
<sslcfg_id>	This is setting SSL context id, Numeric type, range is 0-9. Default is

	0.Please refer to Chapter 19 of this document.
<user_data>	The customized HTTP header information. String type, max length is 256.
<readmode>	For HTTPREAD, Numeric type, it can be set to 0 or 1. If set to 1, you can read the response content data from the same position repeatly. The limit is that the size of HTTP server response content should be shorter than 1M.Default is 0.

Examples

```
 HTTP_RETURNCODE ret = SC_HTTPS_FAIL;
char command_type ={0};
char parameters ={0};

strcpy(command_type, "URL");
strcpy(parameters, "https://www.baidu.com");

if(SC_HTTPS_SUCCESS == sAPI_HttpPara(command_type, parameters);       //set URL parameters
{
        ……                                          //The action after success
}
else
{
        ……                                          //The action after fail
}
```

18.2.4 sAPI_HttpAction

This application interface is used to perform a HTTP Method

sAPI_HttpAction	
Prototype	SC_HTTP_RETURNCODE **sAPI_HttpAction**(int methods);
Parameters	[in] **methods**: 0: GET 1: POST 2: HEAD 3: DELETE
Return value	SC_HTTPS_SUCCESS: perform the HTTP Method success SC_HTTPS_FAIL: perform the HTTP Method fail SC_HTTPS_STATUS_CODE: please refer to chapter 18.3
Header	#include "simcom_https.h"

Examples

```
HTTP_RETURNCODE ret = HTTPS_FAIL;
Int method = 0;
typedef struct
{
    INT32 result;           /*process result for request msg*/
    INT32 status_code;   /*process result for request msg*/
    INT32 method;         /*for sAPI_HttpAction*/
    INT32 action_content_len;      /*the recv lenth of sAPI_HttpAction and sAPI_HttpPostfile*/
    CHAR *data;        /*the data trans from API*/
    INT32 dataLen;    /*the datalen of data*/
    INT32 httpcmd;         /*the API's name where the msg from*/
}SChttpApiTrans;
SChttpApiTrans * httptransdata;
ret = sAPI_HttpAction(method);
if(ret == SC_HTTP_SUCCESS);
{
    sAPI_MsgQRecv(ghttp_api_msgq, &recvMsg, OS_SUSPEND);
    httptransdata   =(SChttpApiTrans *);recvMsg.arg3;              //The action after success
}
else if(ret == SC_HTTP_FAIL);
{
    ……                                                  //The action after fail
}
```

18.2.5 sAPI_HttpData

This application interface is used to save the post data.

sAPI_HttpData	
Prototype	SC_HTTP_RETURNCODE **sAPI_HttpData**(char *buffer, int len);
Parameters	[in] **buffer**: the data for http post. [in] **len**: the length of the buffer.
Return value	SC_HTTPS_SUCCESS: set the data success SC_HTTPS_FAIL: set the data fail
Header	#include "simcom_https.h"

Examples

```
SC_HTTP_RETURNCODE ret = HTTPS_FAIL;
char *buffer =”http post request”;
ret = sAPI_HttpData(buffer, strlen(buffer);
if(ret == SC_HTTP_SUCCESS);
{
    ……                                          //The action after success
}
else if(ret == SC_HTTP_FAIL);
{
    ……                                          //The action after fail
}
```

18.2.6 sAPI_HttpHead

This application interface is used to read the HTTP header information of server response when module receives the response data from server.

sAPI_HttpHead	
Prototype	SC_HTTP_RETURNCODE **sAPI_HttpHead**(void);
Parameters	None.
Return value	SC_HTTPS_SUCCESS: get the header success SC_HTTPS_FAIL: get the header fail
Header	#include "simcom_https.h"

Examples

```
HTTP_RETURNCODE ret = HTTPS_FAIL;

ret = sAPI_HttpHead();
if(ret == HTTP_SUCCESS);
{
    sAPI_MsgQRecv(ghttp_api_msgq, &recvMsg, SC_SUSPEND);
    httptransdata   =(SChttpApiTrans *);recvMsg.arg3;          //The action after success
}
else if(ret == SC_HTTP_FAIL);
{
    ……                                          //The action after fail
}
```

18.2.7 sAPI_HttpRead

This application interface is used to read the data which from server.

sAPI_HttpHead	
Prototype	SC_HTTP_RETURNCODE **sAPI_HttpRead**(int commd_type, int start_offset, int byte_size);
Parameters	[in] **commd_type:** 0: get total size of data saved in buffer, get the size by para: byte_size 1: set the read data offset and size [in] **start_offset:** the start position of reading [in] **byte_size:** the length of data to read
Return value	SC_HTTPS_SUCCESS:get the header success SC_HTTPS_FAIL: get the header fail
Header	#include "simcom_https.h"

Examples

```
 SC_HTTP_RETURNCODE ret = SC_HTTPS_FAIL;
int commd_type = 1;
int start_offset = 0;
int byte_size = 100;

ret = sAPI_HttpRead(commd_type, start_offset, byte_size);
if(ret == SC_HTTP_SUCCESS);
{
    sAPI_MsgQRecv(ghttp_api_msgq, &recvMsg, OS_SUSPEND);
    httptransdata   =(SChttpApiTrans *);recvMsg.arg3;                    //The action after success
}
else if(ret == SC_HTTP_FAIL);
{
    ……                                                    //The action after fail
}
```

18.2.8 sAPI_HttpPostfile

This application interface is used to post file from client to server.

sAPI_HttpPostfile	
Prototype	SC_HTTP_RETURNCODE **sAPI_HttpPostfile**(char * filename, int dir);

Parameters	[in] **filename**: the file for post data. [in] **dir**: the path of the file. 1: C:/ 2: D:/
Return value	SC_HTTPS_SUCCESS: get the header success SC_HTTPS_FAIL: get the header fail SC_HTTPS_STATUS_CODE: please refer to chapter 18.3
Header	#include "simcom_https.h"

Examples

```
SC_HTTP_RETURNCODE ret = SC_HTTPS_FAIL;
char filename = {0};
int dir = 1;
strcpy(filename, "baidu_get.txt");

ret = sAPI_HttpPostfile(filename, dir);
if(ret == SC_HTTP_SUCCESS);
{
    sAPI_MsgQRecv(ghttp_api_msgq, &recvMsg, OS_SUSPEND);
    httptransdata =(SChttpApiTrans *);recvMsg.arg3;                    //The action after success
}
else if(ret == SC_HTTP_FAIL);
{
    ……                                                    //The action after fail
}
```

18.3 Result Codes

18.3.1 Description of <statuscode>s

<statuscode>	Description
100	Continue
101	Switching Protocols
200	OK
201	Created
202	Accepted

203	Non-Authoritative Information
204	No Content
205	Reset Content
206	Partial Content
300	Multiple Choices
301	Moved Permanently
302	Found
303	See Other
304	Not Modified
305	Use Proxy
307	Temporary Redirect
400	Bad Request
401	Unauthorized
402	Payment Required
403	Forbidden
404	Not Found
405	Method Not Allowed
406	Not Acceptable
407	Proxy Authentication Required
408	Request Timeout
409	Conflict
410	Gone
411	Lenth Required
412	Precondition Failed
413	Request Entity Too Large
414	Request-URI Too Large
415	Unsupported Media Type
416	Requested range not satisfiable
417	Expectation Failed
500	Internal Server Error
501	Not Implemented
502	Bad Gateway
503	Service Unavailable
504	Gateway timeout
505	HTTP Version not supported

600	Not HTTP PDU
601	Network Error
602	No memory
603	DNS Error
604	Stack Busy

18.3.2 Description of <errcode>s

<errcode>	Description
0	Success
701	Alert state
702	Unknown error
703	Busy
704	Connection closed error
705	Timeout
706	Receive/send socket data failed
707	File not exists or other memory error
708	Invalid parameter
709	Network error
710	start a new ssl session failed
711	Wrong state
712	Failed to create socket
713	Get DNS failed
714	Connect socket failed
715	Handshake failed
716	Close socket failed
717	No network error
718	Send data timeout
719	CA missed

18.4 Unsolicited Result Codes

URC	Description
+HTTP_PEER_CLOSED	It's a notification message. While received, it means the connection has been closed by server.
+HTTP_NoneT_EVENT	It's a notification message. While received, it means now the network is unavailable.

19 APIs for FTP(S)

19.1 Overview of APIs for FTP(S)

Interface	Description
sAPI_FtpsInit	Start FTP(S); service
sAPI_FtpsDeInit	Stop FTP(S); Service
sAPI_FtpsLogin	Login to a FTP(S);server
sAPI_FtpsLogout	Logout a FTP(S); server
sAPI_FtpsList	List the items in the directory on FTP(S); server
sAPI_FtpsCreateDirectory	Create a new directory on FTP(S); server
sAPI_FtpsDeleteDirectroy	Delete a directory on FTP(S); server
sAPI_FtpsChangeDirectory	Change the current directory on FTP(S); server
sAPI_FtpsGetCurrentDirectory	Get the current directory on FTP(S); server
sAPI_FtpsDeleteFile	Delete a file on FTP(S); server
sAPI_FtpsDownloadFile	Download a file from FTP(S); server to module
sAPI_FtpsUploadFile	Upload a file from module to FTP(S); server
sAPI_FtpsDownloadFileToBuffer	Get a file from FTP(S); server to serial port
sAPI_FtpsGetFileSize	Get the file size on FTP(S); server
sAPI_FtpsDataSocketIpConfig	Set FTP(S); data socket address type
sAPI_FtpsGetDataSocketIpConfig	Get the FTP(S); data socket address type
sAPI_FtpsTransferType	Set the transfer type on FTP(S); server
sAPI_FtpsGetTransferType	Get the transfer type on FTP(S); server
sAPI_FtpsSslConfig	Set the SSL context id for FTPS session

19.2 Detailed Description of APIs for FTP(S)

19.2.1 sAPI_FtpsInit

sAPI_FtpsInit is used to start FTP(S); service by activating PDP context. You must execute sAPI_FtpsInit before any other FTP(S); related operations.

sApi_FTPSInit	
Prototype	void **sAPI_FtpsInit(**SC_PDP_ACTIVE_TYPE type, sMsgQRef msgQRef);
Parameters	[in] **type**: PDP activation currently has two channels: USB and UART. [in] **msgQRef**: All API results will be returned by msgQRef.
Return value	**SC_FTPS_ERROR_CODE**: return by msgQRef, please refer to chapter 17.3 to get more information about error codes
Header	#include "simcom_ftps.h"

Examples

```
sMsgQRef MSGQREF;
SIM_MSG_T ftpsMsg;
sAPI_FtpsInit(FTPS_USB, MSGQREF);
sAPI_MsgQRecv(MSGQREF, &ftpsMsg, SC_SUSPEND);
if(SC_FTPS_RESULT_OK == ftpsMsg.arg2);
{
    ……
    //ftps init successful
}
else
{
    ……
    //ftps init error
}
```

19.2.2 sAPI_FtpsDeInit

sAPI_FtpsDeInit is used to stop FTP(S); service by deactivating PDP context When you are no longer using the FTP(S); service, use this command.

SAPI_FTPSDeInit	
Prototype	void **sAPI_FtpsDeInit**(SC_PDP_ACTIVE_TYPE type);
Parameters	[in] **type**: PDP deactivation currently has two channels: USB and UART.
Return value	**SC_FTPS_ERROR_CODE:** return by msgQRef, please refer to chapter 17.3 to get more information about error codes
Header	#include "simcom_ftps.h"

Examples

```
sMsgQRef MSGQREF;
SIM_MSG_T ftpsMsg;
sAPI_FtpsDeInit(FTPS_USB);
sAPI_MsgQRecv(MSGQREF, &ftpsMsg, SC_SUSPEND);
if(SC_FTPS_RESULT_OK == ftpsMsg.arg2);
{
    ……
    //ftps deInit successful
}
else
{
    ……
    //ftps deInit error
}
```

19.2.3 sAPI_FtpsLogin

sAPI_FtpsLogin is used to login to a FTP(S); server.

sAPI_FtpsLogin	
Prototype	void **sAPI_FtpsLogin**(SCftpsLoginMsg msg);
Parameters	[in] **msg**: SCftpsLoginMsg:Contains information needed to log on to the FTP(S); server.Include server IP address, username, password, port number, server type. IP address: Host address, string type, maximum length is 128. Username: FTP(S); user name, string type, maximum length is 128. Password: The user password, string type, maximum length is 128. Port: The host listening port for FTP(S), the range is from 1 to 65535. Server type: FTP(S); server type, numeric, from 0-3, default is 3 0: FTP server. 1: Explicit FTPS server with AUTH SSL. 2: Explicit FTPS server with AUTH TLS. 3: Implicit FTPS server.
Return value	**SC_FTPS_ERROR_CODE**: return by msgQRef, please refer to chapter 17.3 to get more information about error codes
Header	#include "simcom_ftps.h"

Examples

```
sMsgQRef MSGQREF;
SIM_MSG_T ftpsMsg;
SCftpsLoginMsg msg;
```

```
sAPI_FtpsLogin(msg);
sAPI_MsgQRecv(MSGQREF, &ftpsMsg, SC_SUSPEND);
if(SC_FTPS_RESULT_OK == ftpsMsg.arg2);
{
    ……
    //login successful
}
else
{
    ……
    //login error
}
```

19.2.4 sAPI_FtpsLogout

sAPI_FtpsLogout is used to logout a FTP(S); sever, make sure you login a FTP(S); sever before you execute sAPI_FtpsLogout.

sAPI_FtpsLogout	
Prototype	void **sAPI_FtpsLogout**();
Parameters	None.
Return value	**SC_FTPS_ERROR_CODE:** return by msgQRef, please refer to chapter 17.3 to get more information about error codes
Header	#include "simcom_ftps.h"

Examples

```
sMsgQRef MSGQREF;
SIM_MSG_T ftpsMsg;
sAPI_FtpsLogout();
sAPI_MsgQRecv(MSGQREF, &ftpsMsg, SC_SUSPEND);
if(SC_FTPS_RESULT_OK == ftpsMsg.arg2);
{
    ……
    //logout successful
}
else
{
    ……
    //logout error
}
```

NOTE

When you want to stop the FTP(S); service, please use sAPI_FtpsLogout to log out of the FTP(S); server, then use sAPI_FtpsDeInit to stop FTP, if you only use sAPI_FtpsDeInit, it will report ERROR.

19.2.5 sAPI_FtpsList

This API is used to list the items in the specified directory on FTP(S); server. Make sure that you have login to FTP(S); server successfully.

sAPI_FtpsList	
Prototype	void **sAPI_FtpsList**(char* dir);
Parameters	[in] **dir**: The directory to be listed, string type, maximum length is 112.
Return value	**SC_FTPS_ERROR_CODE**: return by msgQRef, please refer to chapter 17.3 to get more information about error codes. **apiFtpsData**: return by msgQRef, the structure contains the length of the data, content, and whether the end of the flag.
Header	#include "simcom_ftps.h"

Examples

```
sMsgQRef MSGQREF;
SIM_MSG_T ftpsMsg;
SCapiFtpsData* ftpsData;
sAPI_FtpsList(dir);
While(1);
{
    sAPI_MsgQRecv(MSGQREF, &ftpsMsg, SC_SUSPEND);
    if(SC_FTPS_RESULT_OK == ftpsMsg.arg2);
    {
        ftpsData =(SCapiFtpsData*);ftpsMsg.arg3;
        if(SC_DATA_COMPLETE == ftpsData->flag);
        {
            …….
        }
        else if(SC_DATA_RESUME == ftpsData->flag);
        {
            ……
        }
```

```
        else
        {
            ……
        }
    }

}
```

19.2.6 sAPI_FtpsCreateDirectory

sAPI_FtpsCreateDirectory is used to create a new directory on a FTP(S); server. Please make sure login to the FTP(S); server successfully before create a directory.

sAPI_FtpsCreateDirectory	
Prototype	void **sAPI_FtpsCreateDirectory**(char* dir);
Parameters	[in] **dir**: The directory to be created, string type, maximum length is 112.
Return value	**SC_FTPS_ERROR_CODE**: return by msgQRef, please refer to chapter 17.3 to get more information about error codes
Header	#include "simcom_ftps.h"

Examples

```
sMsgQRef MSGQREF;
SIM_MSG_T ftpsMsg;
sAPI_FtpsCreateDirectory(dir);
sAPI_MsgQRecv(MSGQREF, &ftpsMsg, SC_SUSPEND);
if(SC_FTPS_RESULT_OK == ftpsMsg.arg2);
{
    ……
    //create directory successful
}
else
{
    ……
    //create directory error
}
```

19.2.7 sAPI_FtpsDeleteDirectroy

sAPI_FtpsDeleteDirectroy is used to delete a directory on FTP(S); server, please make sure login to the FTP(S);server successfully before delete a directory.

sAPI_FtpsDeleteDirectroy	
Prototype	void **sAPI_FtpsDeleteDirectroy**(char* dir);
Parameters	[in] **dir**: The directory to be deleted, string type, maximum length is 112.
Return value	**SC_FTPS_ERROR_CODE:** return by msgQRef, please refer to chapter 17.3 to get more information about error codes
Header	#include "simcom_ftps.h"

Examples

```
sMsgQRef MSGQREF;
SIM_MSG_T ftpsMsg;
sAPI_FtpsDeleteDirectroy(dir);
sAPI_MsgQRecv(MSGQREF, &ftpsMsg, SC_SUSPEND);
if(SC_FTPS_RESULT_OK == ftpsMsg.arg2);
{
    ……
    //change directory successful
}
else
{
    ……
    //create directory failed
}
```

19.2.8 sAPI_FtpsChangeDirectory

You can use this API to change the current directory on FTP(S); sever. Make sure you have login to FTP(S); server successfully.

sAPI_FtpsChangeDirectory	
Prototype	void **sAPI_FtpsChangeDirectory**(char* dir);
Parameters	[in] **dir**: The directory to be changed, string type, maximum length is 112.
Return value	**SC_FTPS_ERROR_CODE:** return by msgQRef, please refer to chapter 17.3 to get more information about error codes
Header	#include "simcom_ftps.h"

Examples

```
sMsgQRef MSGQREF;
SIM_MSG_T ftpsMsg;
sAPI_FtpsChangeDirectory(dir);
sAPI_MsgQRecv(MSGQREF, &ftpsMsg, SC_SUSPEND);
if(SC_FTPS_RESULT_OK == ftpsMsg.arg2);
{
    ……
    //change directory successful
}
else
{
    ……
    //change directory failed
}
```

19.2.9 sAPI_FtpsGetCurrentDirectory

This API is used to get the current directory on FTPS server. Before use this API, please make sure you have login to FTP(S); server successfully

sAPI_FtpsGetCurrentDirectory	
Prototype	void **sAPI_FtpsGetCurrentDirectory**();
Parameters	None.
Return value	**SC_FTPS_ERROR_CODE:** return by msgQRef, please refer to chapter 17.3 to get more information about error codes. **char*:** return by msgQRef, directory name.
Header	#include "simcom_ftps.h"

Examples

```
sMsgQRef MSGQREF;
SIM_MSG_T ftpsMsg;
char * dir;
sAPI_FtpsGetCurrentDirectory();
sAPI_MsgQRecv(MSGQREF, &ftpsMsg, SC_SUSPEND);
if(SC_FTPS_RESULT_OK == ftpsMsg.arg2);
{
    dir =(char *);ftpsMsg.arg3;
    ……
    //get directory successful
}
```

```
else
{
    ……
    //get directory failed
}
```

19.2.10 sAPI_FtpsDeleteFile

You can use sAPI_FtpsDeleteFile to delete a file on FTP(S); server, please make sure login to the FTP(S); server successfully before delete a file.

sAPI_FtpsDeleteFile	
Prototype	void **sAPI_FtpsDeleteFile**(char* fileName);
Parameters	[in] **fileName**: The name of the file to be deleted. String type, the maximum length is 112
Return value	**SC_FTPS_ERROR_CODE:** return by msgQRef, please refer to chapter 17.3 to get more information about error codes
Header	#include "simcom_ftps.h"

Examples

```
sMsgQRef MSGQREF;
SIM_MSG_T ftpsMsg;
FTPS_delete(fileName);
sAPI_MsgQRecv(MSGQREF, &ftpsMsg, SC_SUSPEND);
if(SC_FTPS_RESULT_OK == ftpsMsg.arg2);
{
    ……
    //delete directory successful
}
else
{
    ……
    //delete directory failed
}
```

19.2.11 sAPI_FtpsDownloadFile

You can download a file from FTP(S); server to module, and you can select the directory where to save the

downloaded file. Make sure that you have login to FTP(S); server successfully before download file.

sAPI_FtpsDownloadFile	
Prototype	void **sAPI_FtpsDownloadFile**(char* fileName, SC_FTPS_FILE_LOCATION loc);
Parameters	[in] **fileName**: The remote file path. String type, maximum length is 112 [in] **loc**: The directory to save the downloaded file. Numeric type, range is 1-2 1: C:/(local storage); 2: D:/(sd card);
Return value	**SC_FTPS_ERROR_CODE:** return by msgQRef, please refer to chapter 17.3 to get more information about error codes
Header	#include "simcom_ftps.h"

Examples

```
sMsgQRef MSGQREF;
SIM_MSG_T ftpsMsg;
sAPI_FtpsDownloadFile(fileName);
sAPI_MsgQRecv(MSGQREF, &ftpsMsg, SC_SUSPEND);
if(SC_FTPS_RESULT_OK == ftpsMsg.arg2);
{
    ……
    //download file successful
}
else
{
    ……
    //download file failed
}
```

19.2.12 sAPI_FtpsUploadFile

You can use this command to upload a file to FTP(S); server from module. By setting parameter loc you can select the directory that contains the file to be uploaded. Make sure that you have login to the FTP(S); server successfully before upload file.

sAPI_FtpsUploadFile	
Prototype	void **sAPI_FtpsUploadFile**(char* fileName, SIMCOM_FTPS_FILE_LOCATION loc, int startPos);
Parameters	[in] **fileName**: The remote file path. String type, maximum length is 112 [in] **loc**: The directory to save the downloaded file. Numeric type, range is 1-2 1: C:/(local storage);

	2: D:/(sd card); [in] **startPos**: The value for FTP "REST" command which is used for broken transfer when transferring failed last time. Numeric type, the range is from 0 to 2147483647.
Return value	**SC_FTPS_ERROR_CODE:** return by msgQRef, please refer to chapter 17.3 to get more information about error codes
Header	#include "simcom_ftps.h"

Examples

```
sMsgQRef MSGQREF;
SIM_MSG_T ftpsMsg;
sAPI_FtpsUploadFile(fileName);
sAPI_MsgQRecv(MSGQREF, &ftpsMsg, SC_SUSPEND);
if(SC_FTPS_RESULT_OK == ftpsMsg.arg2);
{
    ……
    //upload file successful
}
else
{
    ……
    //upload file failed
}
```

19.2.13 sAPI_FtpsDownloadFileToBuffer

You can use this command to get a file from FTP(S); server to buffer.

sAPI_FtpsDownloadFileToBuffer	
Prototype	void **sAPI_FtpsDownloadFileToBuffer**(char* fileName, int startPos);
Parameters	[in] **fileName**: The remote file path. String type, maximum length is 112. [in] **startPos**: The value for FTP "REST" command which is used for broken transfer when transferring failed last time. Numeric type, the range is from 0 to 2147483647.
Return value	**SC_FTPS_ERROR_CODE**: return by msgQRef, please refer to chapter 17.3 to get more information about error codes. **apiFtpsData**: return by msgQRef, the structure contains the length of the data, content, and whether the end of the flag.
Header	#include "simcom_ftps.h"

Examples

```
sMsgQRef MSGQREF;
FS_MSG_T ftpsMsg;
SCapiFtpsData* ftpsData;
sAPI_FtpsList(dir);
While(1);
{
    sAPI_MsgQRecv(MSGQREF, &ftpsMsg, SC_SUSPEND);
    if(SC_FTPS_RESULT_OK == ftpsMsg.arg2);
    {
        ftpsData =(SCapiFtpsData*);ftpsMsg.arg3;
        if(SC_DATA_COMPLETE == ftpsData->flag);
        {
            …….
        }
        else if(SC_DATA_RESUME == ftpsData->flag);
        {
            ……
        }
        else
        {
            ……
        }
    }

}
```

19.2.14 sAPI_FtpsGetFileSize

You can use this command to get the file size on FTP(S); server. Please make sure you have login to FTP(S); server before use sAPI_FtpsGetFileSize.

sAPI_FtpsGetFileSize	
Prototype	void **sAPI_FtpsGetFileSize**(char* fileName);
Parameters	[in] **fileName**: The remote file path on FTP(S); server. String type, max length is 112.
Return value	**SC_FTPS_ERROR_CODE:** return by msgQRef, please refer to chapter 17.3 to get more information about error codes. **char*:** return by msgQRef, file size.
Header	#include "simcom_ftps.h"

Examples

```
sMsgQRef MSGQREF;
SIM_MSG_T ftpsMsg;
int * size;
sAPI_FtpsGetFileSize(fileName);
sAPI_MsgQRecv(MSGQREF, &ftpsMsg, SC_SUSPEND);
if(SC_FTPS_RESULT_OK == ftpsMsg.arg2);
{
    size =(int *);ftpsMsg.arg3;
    ……
    //get file size successful
}
else
{
    ……
    //get file size fail
}
```

19.2.15 sAPI_FtpsTransferType

This api is used to set the transfer type on FTP(S); server, please make sure you have login to FTP(S); server before use sAPI_FtpsTransferType.

sAPI_FtpsTransferType	
Prototype	void **sAPI_FtpsTransferType**(char* type);
Parameters	[in] **type**: The type of transferring: A: ASCII. I: Binary
Return value	**SC_FTPS_ERROR_CODE:** return by msgQRef, please refer to chapter 17.3 to get more information about error codes
Header	#include "simcom_ftps.h"

Examples

```
sMsgQRef MSGQREF;
SIM_MSG_T ftpsMsg;
sAPI_FtpsTransferType(type);
sAPI_MsgQRecv(MSGQREF, &ftpsMsg, SC_SUSPEND);
```

```
if(SC_FTPS_RESULT_OK == ftpsMsg.arg2);
{
    ……
    //set type successful
}
else
{
    ……
    //set type failed
}
```

19.2.16 sAPI_FtpsGetTransferType

This api is used to get the transfer type on FTP(S); server, please make sure you have login to FTP(S); server before use sAPI_FtpsGetTransferType

sAPI_FtpsGetTransferType	
Prototype	void **sAPI_FtpsGetTransferType**();
Parameters	None
Return value	**SC_FTPS_ERROR_CODE:** return by msgQRef, please refer to chapter 17.3 to get more information about error codes. **char*:** return by msgQRef, transfer type.
Header	#include "simcom_ftps.h"

Examples

```
sMsgQRef MSGQREF;
SIM_MSG_T ftpsMsg;
char * type;
sAPI_FtpsGetTransferType();
sAPI_MsgQRecv(MSGQREF, &ftpsMsg, SC_SUSPEND);
if(SC_FTPS_RESULT_OK == ftpsMsg.arg2);
{
    type =(char *);ftpsMsg.arg3;
    ……
    //get type successful
}
else
{
    ……
    //get type failed
```

}

19.2.17 sAPI_FtpsSslConfig

You can use this api to set the SSL context id for FTPS session.

sAPI_FtpsSslConfig	
Prototype	void **sAPI_FtpsSslConfig**(int session, int sslId);
Parameters	[in] **session**: 0 for control session, 1 for data session. [in] **sslId**: SSL context ID during 0-9.
Return value	**SC_FTPS_ERROR_CODE:** return by msgQRef, please refer to chapter 17.3 to get more information about error codes.
Header	#include "simcom_ftps.h"

Examples

```
sMsgQRef MSGQREF;
SIM_MSG_T ftpsMsg;
sAPI_FtpsSslConfig(0, 0);
sAPI_MsgQRecv(MSGQREF, &ftpsMsg, SC_SUSPEND);
if(SC_FTPS_RESULT_OK == ftpsMsg.arg2);
{
    ……
    //configured ssl successful
}
else
{
    ……
    //configured ssl fail
}
```

19.3 Result Codes

19.3.1 Description of <errcode>s

For more details, please refer to simcom_ftps.h.

<errcode>	Description
0	Success
1	SSL alert
2	Unknown error
3	Busy
4	Connection closed by server
5	Timeout
6	Transfer failed
7	File not exists or any other memory error
8	Invalid parameter
9	Operation rejected by server
10	Network error
11	State error
12	Failed to parse server name
13	Create socket error
14	Connect socket failed
15	Close socket failed
16	SSL session closed
17	File error, file not exist or other error.
421	Server response connection time out, while received error code 421, you need do AT+CFTPSLOGOUT to logout server then AT+CFTPSLOGIN again for further operations.

20 APIs for MQTT(S)

20.1 Overview of APIs for MQTT(S)

Interface	Description
sAPI_MqttStart	Start MQTT service
sAPI_MqttStop	Stop MQTT service
sAPI_MqttAccq	Acquire a client
sAPI_MqttRel	Release a client
sAPI_MqttSslCfg	Set the SSL context(only for SSL/TLS MQTT);
sAPI_MqttWillTopic	Input the topic of will message
sAPI_MqttWillMsg	Input the will message
sAPI_MqttConnect	Connect to MQTT server
sAPI_MqttDisConnect	Disconnect from server
sAPI_MqttTopic	Input the topic of publish message
sAPI_MqttPayload	Input the publish message
sAPI_MqttPub	Publish a message to server
sAPI_MqttSubTopic	Input the topic of subscribe message
sAPI_MqttSub	Subscribe a message to server
sAPI_MqttUNSubTopic	Input the topic of unsubscribe message
sAPI_MqttUnsub	Unsubscribe a message to server
sAPI_MqttCfg	Configure the MQTT Context

20.2 Detailed Description of APIs for MQTT(S)

Part of API for MQTT containing a few parameters for different status, if it is unnecessary for a specific condition these parameters should be set as NULL or 0. There are some unimportant parameters that can be set as the default values, which is defined in API parameters description.
All API functions are blocking functions and will return only after the function is completely executed, some API, such as sAPI_MqttConnect, may take few seconds.

20.2.1 sAPI_MqttStart

sAPI_MqttStart is used to start MQTT service by activating PDP context. You must use this API before any other MQTT related operations.

sAPI_MqttStart	
Prototype	SCmqttReturnCode **sAPI_MqttStart**(int channel);
Parameters	[in] **channel**: urc transmission channel(to Serial port or other channel); 0: serial port 1: usb -1: ignor it
Return value	SCmqttReturnCode, succeed or the error code
Header	#include "simcom_mqtts_client.h"

20.2.2 sAPI_MqttStop

sAPI_MqttStop is used to stop MQTT service.

sAPI_MqttStop	
Prototype	SCmqttReturnCode **sAPI_MqttStop**(void);
Parameters	None
Return value	SCmqttReturnCode, succeed or the error code
Header	#include "simcom_mqtts_client.h"

NOTE

This API is used to stop MQTT service. You can use it after disconnect and release.

20.2.3 sAPI_MqttAccq

sAPI_MqttAccq is used to acquire a MQTT client. It must be called before all commands about MQTT connect and after mqtt service start.

sAPI_MqttAccq	
Prototype	SCmqttReturnCode **sAPI_MqttAccq**(SCmqttOperationType commad_type, char *string_return, int client_index, char *clientID, int server_type, sMsgQRef

	msgQ_urc);
Parameters	[in] **commad_type**: SC_MQTT_OP_SET: set parameters by this command SC_MQTT_OP_GET: get the acquired parameters set by this command command [out] **string_return**: the return string for commad_type=1; [in] **client_index**: A numeric parameter that identifies a client. The range of permitted values is 0 to 1. [in] **clientID**: The UTF-encoded string. It specifies a unique identifier for the client. The string length is from 1 to 128 bytes. [in] **server_type**: A numeric parameter that identifies the server type. The default value is 0. 0: MQTT server with TCP 1: MQTT server with SSL/TLS [in] **msgQ_urc**: the massage queue for urc message of mqtt service.
Return value	SCmqttReturnCode, succeed or the error code
Header	#include "simcom_mqtts_client.h"

NOTE

This is used to acquire a MQTT client. It must be called before all commands about MQTT connect and after start MQTT service.
For MQTT message queue(**msgQ_urc);** only receive subscribed message data which is from MQTT server.

20.2.4 sAPI_MqttRel

sAPI_MqttRel is used to release a MQTT client. It must be called after disconnect and before stop.

UART_writeData	
Prototype	SCmqttReturnCode **sAPI_MqttRel**(int client_index);
Parameters	[in] **client_index:** A numeric parameter that identifies a client. The range of permitted values is 0 to 1.
Return value	SCmqttReturnCode, succeed or the error code
Header	#include "simcom_mqtts_client.h"

NOTE

This is used to release a MQTT client. It must be called after disconnect and before stop.

20.2.5 sAPI_MqttSslCfg

sAPI_MqttSslCfg is used to set the SSL context which to be used in the SSL connection when it will connect to a SSL/TLS MQTT server. It must be called before connect and after start. The setting will be cleared after connect failed or disconnect.

sAPI_MqttSslCfg	
Prototype	SCmqttReturnCode **sAPI_MqttSslCfg**(SCmqttOperationType commad_type, char *string_return, int client_index, int ssl_ctx_index);
Parameters	[in] **commad_type**: SC_MQTT_OP_SET: set parameters by this command SC_MQTT_OP_GET: get the parameters seted by this command [out] **string_return**: the return string for commad_type == SC_MQTT_OP_GET; [in] **client_index**: A numeric parameter that identifies a client. The range of permitted values is 0 to 1. [in] **ssl_ctx_index**: The SSL context ID which will be used in the SSL connection. **Refer to the SSL related APIs**.
Return value	SCmqttReturnCode, succeed or the error code
Header	#include "simcom_mqtts_client.h"

20.2.6 sAPI_MqttWillTopic

sAPI_MqttWillTopic is used to input the topic of will message.

sAPI_MqttWillTopic	
Prototype	SCmqttReturnCode **sAPI_MqttWillTopic**(int client_index, char *topic_data, int topic_length);
Parameters	[in] **client_index**: A numeric parameter that identifies a client. The range of permitted values is 0 to 1. [in] **topic_data**: a topic string of publish message, it should be UTF-encoded string. The range is from 1 to 1024 bytes. [in] **topic_length**: The length of input topic data.
Return value	SCmqttReturnCode, succeed or the error code
Header	#include "simcom_mqtts_client.h"

20.2.7 sAPI_MqttWillMsg

sAPI_MqttWillMsg is used to input the message body of will message.

sAPI_MqttWillMsg	
Prototype	SCmqttReturnCode **sAPI_MqttWillMsg**(int client_index, char *msg_data, int msg_length);
Parameters	[in] **client_index**: A numeric parameter that identifies a client. The range of permitted values is 0 to 1. [in] **msg_data**: a topic string of will message, it should be UTF-encoded string. The range is from 1 to 1024 bytes. [in] **msg_length**: The length of will message data.
Return value	SCmqttReturnCode, succeed or the error code
Header	#include "simcom_mqtts_client.h"

20.2.8 sAPI_MqttConnect

sAPI_MqttConnect is used to connect to a MQTT server.

sAPI_MqttConnect	
Prototype	SCmqttReturnCode **sAPI_MqttConnect**(SCmqttOperationType commad_type, char *string_return, int client_index, char *server_addr, int keepalive_time, int clean_session, char *user_name, char *pass_word);
Parameters	[in] **commad_type**: SC_MQTT_OP_SET: set parameters by this command SC_MQTT_OP_GET: get the parameters seted by this command [out] **string_return**: the return string for commad_type= SC_MQTT_OP_GET; [in] **client_index**: A numeric parameter that identifies a client. The range of permitted values is 0 to 1. [in] **server_addr**: The string that described the server address and port. The range of the string length is 9 to 256 bytes. The string should be like this "tcp://116.247.119.165:5141", must begin with "tcp://". If the <server_addr> not include the port, the default port is 1883. [in] **keepalive_time**: The time interval between two messages received from a client. The client will send a keep-alive packet when there is no message sent to server after song long time. The range is from 1s to 64800s(18 hours);. [in] **clean_session**: The clean session flag. The value range is from 0 to 1, and default value is 0. [in] **user_name**: The user name identifies the name of the user which can be used for authentication when connecting to server. The string length is from 1 to 256 bytes.

	[in] **pass_word**: The password corresponding to the user which can be used for authentication when connecting to server. The string length is from 1 to 256 bytes.
Return value	SCmqttReturnCode, succeed or the error code
Header	#include "simcom_mqtts_client.h"

Examples

Refer to Demo for MQTT(S).

NOTE

If you don't set the SSL context by sAPI_MqttSslCfg before connecting a SSL/TLS MQTT server, it will use the <client_index> SSL context when connecting to the server.

20.2.9 sAPI_MqttDisConnect

sAPI_MqttDisConnect is used to disconnect from the server.

sAPI_MqttDisConnect	
Prototype	SCmqttReturnCode **sAPI_MqttDisConnect**(SCmqttOperationType commad_type, char *string_return, int client_index, int timeout);
Parameters	[in] **commad_type**: SC_MQTT_OP_SET: set parameters by this command SC_MQTT_OP_GET: get the parameters seted by this command [out] **string_return**: the return string for commad_type=1; [in] **client_index**: A numeric parameter that identifies a client. The range of permitted values is 0 to 1. [in] **timeout**: The timeout value for disconnection. The unit is second. The range is 60s to 180s. The default value is 0s(not set the timeout value);.
Return value	SCmqttReturnCode, succeed or the error code
Header	#include "simcom_mqtts_client.h"

20.2.10 sAPI_MqttTopic

sAPI_MqttTopic is used to input the topic of a publish message.

sAPI_MqttTopic	
Prototype	SCmqttReturnCode **sAPI_MqttTopic**(int client_index, char *topic_data, int topic_length);
Parameters	[in] client_index: A numeric parameter that identifies a client. The range of permitted values is 0 to 1. [in] topic_data: a topic string of publish message. The publish message topic should be UTF-encoded string. [in] topic_length: The length of input topic data. The range is from 1 to 1024 bytes.
Return value	SCmqttReturnCode, succeed or the error code
Header	#include "simcom_mqtts_client.h"

NOTE

The topic will be clean after published to server.

20.2.11 sAPI_MqttPayload

sAPI_MqttPayload is used to input the message body of a publish message.

sAPI_MqttPayload	
Prototype	SCmqttReturnCode **sAPI_MqttPayload**(int client_index, char *payload_data, int payload_length);
Parameters	[in] **client_index**: A numeric parameter that identifies a client. The range of permitted values is 0 to 1. [in] **payload_data**: a buffer for payload data. The publish message should be UTF-encoded string. [in] **payload_length**: The length of input message data. The range is from 1 to 10240 bytes.
Return value	SCmqttReturnCode, succeed or the error code
Header	#include "simcom_mqtts_client.h"

NOTE

The topic will be clean after published to server.

20.2.12 sAPI_MqttPub

sAPI_MqttPub is used to publish a message to MQTT server.

sAPI_MqttPub	
Prototype	SCmqttReturnCode **sAPI_MqttPub**(int client_index, int qos, int pub_timeout, int ratained, int dup);
Parameters	[in] **client_index**: A numeric parameter that identifies a client. The range of permitted values is 0 to 1. [in] **qos**: The publish message's qos. The range is from 0 to 2. [in] **pub_timeout**: The timeout value for disconnection. The unit is second. The range is 60s to 180s. The default value is 0s(not set the timeout value);. [in] **ratained**: The retain flag of the publish message. The value is 0 or 1. The default value is 0. [in] **dup**: The dup flag to the message. The value is 0 or 1. The default value is 0. The flag is set when the client or server attempts to re-deliver a message.
Return value	SCmqttReturnCode, succeed or the error code
Header	#include "simcom_mqtts_client.h"

NOTE

The topic and payload will be clean after publish.

20.2.13 sAPI_MqttSubTopic

sAPI_MqttSubTopic is used to input the topic of a subscribe message.

sAPI_MqttSubTopic	
Prototype	SCmqttReturnCode **sAPI_MqttSubTopic**(int client_index, char *sub_topic_data, int sub_topic_length, int qos);
Parameters	[in] **client_index**: A numeric parameter that identifies a client. The range of permitted values is 0 to 1. [in] **sub_topic_data**: the topic of a subscribe message. The publish message topic should be UTF-encoded string. [in] **sub_topic_length**: The length of input topic data. The range is from 1 to 10240 bytes. [in] **qos**: The publish message's qos. The range is from 0 to 2.
Return value	SCmqttReturnCode, succeed or the error code
Header	#include "simcom_mqtts_client.h"

NOTE

The topic will be clean after subscribe successfully.

20.2.14 sAPI_MqttSub

sAPI_MqttSub is used to subscribe a message to MQTT server.

sAPI_MqttSub	
Prototype	SCmqttReturnCode **sAPI_MqttSub**(int client_index, char *topic_data, int topic_length, int qos, int dup);
Parameters	[in] **client_index**: A numeric parameter that identifies a client. The range of permitted values is 0 to 1. [in] **topic_data**: a topic string of publish message. The publish message topic should be UTF-encoded string. [in] **topic_length**: The length of input topic data. The range is from 1 to 1024 bytes. [in] **dup**: The dup flag to the message. The value is 0 or 1. The default value is 0. The flag is set when the client or server attempts to re-deliver a message. [in] **qos**: The publish message's qos. The range is from 0 to 2.
Return value	SCmqttReturnCode, succeed or the error code
Header	#include "simcom_mqtts_client.h"

20.2.15 sAPI_MqttUNSubTopic

sAPI_MqttUNSubTopic is used to input the topic of a unsubscribe message.

sAPI_MqttUNSubTopic	
Prototype	SCmqttReturnCode **sAPI_MqttUNSubTopic**(int client_index, char *unsub_topic_data, int unsub_topic_length);
Parameters	[in] **client_index**: A numeric parameter that identifies a client. The range of permitted values is 0 to 1. [in] **unsub_topic_data**: the topic of a unsubscribe message. The publish message topic should be UTF-encoded string. [in] **unsub_topic_length**: The length of input topic data. The range is from 1 to 1024 bytes.

Return value	SCmqttReturnCode, succeed or the error code
Header	#include "simcom_mqtts_client.h"

NOTE

The topic will be clean after unsubscribe successfully.

20.2.16 sAPI_MqttUnsub

sAPI_MqttUnsub is used to unsubscribe a message to MQTT server.

sAPI_MqttUnsub	
Prototype	SCmqttReturnCode **sAPI_MqttUnsub**(int client_index, char *topic_data, int topic_length, int dup);
Parameters	[in] **client_index**: A numeric parameter that identifies a client. The range of permitted values is 0 to 1. [in] **topic_data**: A topic string of Unsubscribe message. The publish message topic should be UTF-encoded string. [in] **topic_length**: The length of input topic data. The range is from 1 to 1024 bytes. [in] **dup**: The dup flag to the message. The value is 0 or 1. The default value is 0. The flag is set when the client or server attempts to re-deliver a message.
Return value	SCmqttReturnCode, succeed or the error code
Header	#include "simcom_mqtts_client.h"

20.2.17 sAPI_MqttCfg

sAPI_MqttCfg is used to configure the MQTT context. It must be called before connect to server and after acquire a client. The setting will be cleared after client release.

sAPI_MqttCfg	
Prototype	SCmqttReturnCode **sAPI_MqttCfg**(SCmqttOperationType commad_type, char *string_return, int client_index, int config_type, int related_value);
Parameters	[in] **commad_type**: SC_MQTT_OP_SET: set parameters by this command SC_MQTT_OP_GET: get the parameters seted by this command [out] **string_return**: the return string for commad_type=1.

	[in] **client_index**: A numeric parameter that identifies a client. The range of permitted values is 0 to 1. [in] **config_type**: choose one of the configuration type 0: "checkUTF8" 1: "optimeout" [in] **related_value**: (1); if config_type = 0, related_value set as the flag to indicate whether to check the string is UTF8 coding or not, the default value is 1. 0: Not check UTF8 coding. 1: Check UTF8 coding. (2); if config_type = 1, related_value set as time out value, it is used to set max timeout interval of sending or receiving data operation. The range is from 20 seconds to 120 seconds, the default value is 120 seconds.
Return value	SCmqttReturnCode, succeed or the error code
Header	#include "simcom_mqtts_client.h"

NOTE

The setting will be cleared after sAPI_MqttRel.

20.2.18 sAPI_MqttConnLostCb

sAPI_MqttConnLostCb is used to used to handle abnormal disconnection of mqtt client.

sAPI_MqttConnLostCb	
Prototype	void sAPI_MqttConnLostCb(mqtt_connlost_cb cb);
Parameters	[in] **cb**: The function that will be called when the network is interrupted abnormally
Return value	None
Header	#include "simcom_mqtts_client.h"

NOTE

The setting will be cleared after sAPI_MqttRel.

20.3 Result Codes

20.3.1 Description of <err>s

<err>	Meaning
0	Operation succeeded
1	Failed
2	Bad utf-8 string
3	Sock connect fail
4	Sock create fail
5	Sock close fail
6	Message receive fail
7	Network open fail
8	Network close fail
9	Network not opened
10	Client index error
11	No connection
12	Invalid parameter
13	Not supported operation
14	Client is busy
15	Require connection fail
16	Sock sending fail
17	Timeout
18	Topic is empty
19	Client is used
20	Client not acquired
21	Client not released
22	Length out of range
23	Network is opened
24	Packet fail
25	Dns error
26	Socket is closed by server
27	Connection refused: unaccepted protocol version
28	Connection refused: identifier rejected
29	Connection refused: server unavailable
30	Connection refused: bad user name or password
31	Connection refused: not authorized
32	Handshake fail

33	Not set certificate
34	Open session failed
35	Disconnect from server failed

20.4 Demo for MQTT(S)

```
#include "simcom_mqtts_client.h"

int ret = -1;

int error_code = 0;

mqtt_client_thread();     //client thread to control mqtt process

{

    ret = sAPI_MqttStart(-1);          //start mqtt service, no urc transmission channel

    if(0 != ret);{

        sAPI_Debug("start fail, error_code = %d", ret);

    }

    sAPI_MqttAccq(0, NULL, 0, "test0", 0, urc_mqtt_msgq_1); //accquir a client-0


    sAPI_MqttCfg(0, NULL, 0, 0, 0);     //config client-0, not check UTF-8 coding


    ret = sAPI_MqttConnect(0, NULL, 0, "tcp://120.27.2.154:1883", 60, 1, NULL, NULL);

    //connect to srv

    if(0 != ret);{

        sAPI_Debug("connect fail, error_code = %d", ret);

    }

    sAPI_MqttSubTopic(0, "apitest", 7, 1);    //set the multi topic message(no more than 10);

    sAPI_MqttSubTopic(0, "apitest_2", 9, 1);

    sAPI_MqttSubTopic(0, "apitest_3", 9, 1);

    . . .


    ret = sAPI_MqttSub(0, NULL, 0, 0, 0);    //subscribe more than one input topic

    if(0 != ret);{

        sAPI_Debug("sub fail, error_code = %d", ret);
```

```
}
ret = sAPI_MqttSub(0, "apitest1", 8, 0, 0);    //just sub a tpoic
if(0 != ret);{
    sAPI_Debug("sub one topic fail, error_code = %d", ret);
}

sAPI_MqttTopic(0, "apitest", 7);    //input pub topic message

sAPI_MqttPayload(0, "come here, it is the test msg", strlen("come here, it is the test msg"));
// input the payload of pub message(no more than 10240 byte);

ret = sAPI_MqttPub(0, 1, 60, 0, 0);    //pub the message to server
if(0 != ret);{
    sAPI_Debug("pub message fail, error_code = %d", ret);
}

Sleep(30);
//sleep 30 sec before unsub, before unsubscribe the receive task could receive message about subscribed topic

ret = sAPI_MqttUnsub(0, "apitest", 8, 0);    //unsubscribe a topic
if(0 != ret);{
    sAPI_Debug("unsub a topic fail, error_code = %d", ret);
}

sAPI_MqttUNSubTopic(0, "apitest2", 8);    //set multi unsubscribe topic(no more than 10);
. . .

ret = sAPI_MqttUnsub(0, NULL, 0, 0);    //unsubscribe multi topic to server
if(0 != ret);{
    sAPI_Debug("unsub multi topic fail, error_code = %d", ret);
}

Sleep(20);
```

```
    //sleep 20 sec before disconn, receive task can not receive messge from unsubed topic
    Ret = sAPI_MqttDisConnect(0, NULL, 0, 60);    //disconnect from server
    if(0 != ret);{
        sAPI_Debug("disconnect fail, error_code = %d", ret);
   }
    sAPI_MqttRel(0);    //release the client-0

    sAPI_MqttStop();    //stop mqtt service
}

mqtt_receive_thread();        //the thread prepared to receive message about subscrobed topic
{
   mqtt_data *sub_data = NULL;
   FS_MSG_T msgQ_data_recv = {FS_MSG_INIT, 0, -1, NULL};
   simMsgRecv(urc_mqtt_msgq_1, &msgQ_data_recv, OS_SUSPEND);
   sub_data =(mqtt_data *);(msgQ_data_recv.arg3);    //get subscribed tpoic message data here

}
```

21 APIs for SSL

21.1 Overview of APIs for SSL

Interface	Description
sAPI_SslGetContextIDMsg	Get the SSL Context Configuration
sAPI_SslSetContextIDMsg	Configure the SSL Context
sAPI_SslHandShake	Handshake with server
sAPI_SslRead	Read data from server
sAPI_SslSend	Send data to server
sAPI_SslClose	Free the ssl context

21.2 Detailed Description of APIs for SSL

21.2.1 sAPI_SsLGetContextIDMsg

sAPI_SsLGetContextIDMsg	
Prototype	sslContent **sAPI_SsLGetContextIDMsg**(int sslId);
Parameters	[in] **sslId**: The SSL context ID. The range is 0-9.
Return value	**sslContent**: Include sslversion, authmode, ignoreltime, negotiatetime, ca_file, clientcert_file, clientkey_file, enalbeSNI_flag, err. **err**: 0:get content success -1: get content fail **sslversion**: The SSL version, the default value is 4. 0 – SSL3.0 1 – TLS1.0 2 – TLS1.1 3 – TLS1.2 4 – All The configured version should be support by server. So you should use the default value if you are not sure that the version which the server supported. **authmode**: The authentication mode, the default value is 0. 0 – no authentication. 1–server authentication. It needs the root CA of the server. 2–server and client authentication. It needs the root CA of the server, the cert and key of the client. 3–client authentication and no server authentication. It needs the cert and key of

	the client. **Ignoreltime**: The flag to indicate how to deal with expired certificate, the default value is 1. 0 – care about time check for certification. 1 – ignore time check for certification When set the value to 0, it need to set the right current date and time by AT+CCLK when need SSL certification. **ca_file**: The root CA file name of SSL context. The file name must have type like ".pem" or ".der". The length of filename is from 5 to 108 bytes. If the filename contains non-ASCII characters, the file path parameter should contain a prefix of {non-ascii} and the quotation mark(The string in the quotation mark should be hexadecimal of the filename's UTF8 code);. **clientcert_file:** Same as ca_file. **clientkey_file:** Same as ca_file. **enalbeSNI_flag:** The flag to indicate that enable the SNI flag or not, the default value is 0. 0 – not enable SNI. 1 – enable SNI.
Header	#include "simcom_ssl.h"

Examples

```
#include "simcom_ssl.h"
Int ssl_testMain(void);
{
    sslContent content = {0};
    content = sAPI_SsLGetContextIDMsg(0);
    if(0 == content.err);
    {
        ……    //get content success
    }
    else
    {
        return -1;
    }
}
```

21.2.2 sAPI_SsLSetContextIDMsg

sAPI_SsLSetContextIDMsg	
Prototype	int **sAPI_SsLSetContextIDMsg**(char* op, int sslId, char* value);

Parameters	[in] **sslId**: The SSL context ID. The range is 0-9. [in] **op**: I Include sslversion, authmode, ignoreltime, negotiatetime, ca_file, clientcert_file, clientkey_file, enalbeSNI_flag, err. [in] **value**: configure op value, please refer to 15.2.1 return value.
Return value	**0:** configure successfully. **-1:** configure fail.
Header	#include "simcom_ssl.h"

Examples

```
#include "simcom_ssl.h"
Int ssl_testMain(void);
{
    int rest = 0;
    ret = sAPI_SsLSetContextIDMsg("sslversion", 0, 4);
    if(0 == ret);
    {
        ……    //configure successful.
    }
    else
    {
        return -1;
    }
}
```

21.2.3 sAPI_SslHandShake

sAPI_SslHandShake	
Prototype	INT32 **sAPI_SslHandShake** SCSslCtx_t *ctx);
Parameters	[in] **ctx**: The context of ssl, please consider the struct SCSslCtx_t
Return value	**0:** handshake success **other:** handshake fail.
Header	#include "simcom_ssl.h"

Examples

```
#include "simcom_ssl.h"
void demo_ssl_test(void);
{
    INT32 ret = 0;
```

```
    SCSslCtx_t ctx;
    ctx.fd = sockfd;
    ctx.ssl_version = SC_SSL_CFG_VERSION_ALL;


    ret = sAPI_SslHandShake(&ctx);
    sAPI_Debug("sAPI_SslHandShake ret[%d] ", ret);

    if(ret == 0);
    {
        sAPI_Debug("sAPI_SslHandShake success");
    }
    else
    {
        sAPI_Debug("sAPI_SslHandShake fail");
    }
}
```

21.2.4 sAPI_SslRead

sAPI_SslRead	
Prototype	INT32 **sAPI_SslRead**(UINT32 ClientID, INT8 *buf, INT32 len);
Parameters	[in] **ClientID**: The session id of ssl [in] **buf**:The data recv from network [in] **len**:The lenth you want to read
Return value	the number of bytes received, or a non-zero error code
Header	#include "simcom_ssl.h"

Examples

```
#include "simcom_ssl.h"
void demo_ssl_test(void);
{
    memset(recvbuf, 0x0, 1024);
    ret = sAPI_SslRead(0, recvbuf, 1024);
    sAPIebug("ret [%d] recvbuf[%s] ", ret, recvbuf);
    if(ret> 0);
    {
        sAPI_Debug("sAPI_ SslRead success");
    }
    else
```

```
    {
        sAPI_Debug("sAPI_ SslRead errono:%d", ret);
    }
}
```

21.2.5 sAPI_SslSend

sAPI_SslSend	
Prototype	INT32 **sAPI_SslSend**(UINT32 ClientID, INT8 *buf, INT32 len);
Parameters	[in] **ClientID**: The session id of ssl [in] **buf**:The data send to network [in] **len**:The lenth you want to send
Return value	the number of bytes actual send
Header	#include "simcom_ssl.h"

Examples

```
#include "simcom_ssl.h"
void demo_ssl_test(void);
{
    ret = sAPI_SslSend(0, sendbuf, sizeof(sendbuf));
    sAPI_Debug("ret [%d] sendbuf[%s] ", ret, sendbuf);
    if(ret> 0);
    {
        sAPI_Debug("sAPI_ SslSend success");
    }
    else
    {
        sAPI_Debug("sAPI_ SslSend fail);
    }
}
```

21.2.6 sAPI_SslClose

sAPI_SslSend	
Prototype	void **sAPI_SslClose**(UINT32 ClientID);
Parameters	[in] **ClientID**: The session id of ssl
Return value	None

Header	#include "simcom_ssl.h"

Examples

```
#include "simcom_ssl.h"
void demo_ssl_test(void);
{
    sAPI_SslClose(0);
}
```

22 APIs for Audio

22.1 Overview of APIs for Audio

Interface	Description
sAPI_AudioPlaySampleRate	Set sample rate of audio play
sAPI_AudioPlay	Play Audio file
sAPI_AudioPlayMp3Continuously	Play mp3 files continuously
sAPI_AudioStop	Stop Audio file playback
sAPI_AudioRecord	Start or Stop recording
sAPI_AudioPcmPlay	Play pcm stream directly
sAPI_AudioPcmStop	Stop pcm stream directly
sAPI_AudioSetVolume	Set system volume
sAPI_AudioGetVolume	Get system volume
sAPI_PocInitLib	Init poc function
sAPI_PocPlaySound	Play poc PCM buffer
sAPI_PocStopSound	Stop playing poc PCM buffer
sAPI_PocGetPcmAvail	Get available free size
sAPI_PocCleanBufferData	Clean buffer data
sAPI_PocStartRecord	Start recording
sAPI_PocStopRecord	Stop recording
sAPI_PocPcmRead	Get record data
sAPI_AudioSetMicGain	Set microphone gain
sAPI_AudioGetMicGain	Get microphone gain

22.2 Detailed Description of APIs for Audio

22.2.1 sAPI_AudioPlaySampleRate

This application interface is to set sample rate of audio play.

sAPI_AudioPlaySampleRate	
Prototype	void **sAPI_AudioPlaySampleRate**(AUD_SampleRate rate);
Parameters	[in] [**rate**] SAMPLE_RATE_8K: sample rate 8K SAMPLE_RATE_16K: sample rate 16K
Return value	None
Header	#include "simcom_audio.h"

Examples

```
#include "simcom_ audio.h"

void taskMain();
{
    sAPI_AudioPlaySampleRate(SAMPLE_RATE_16K);
}
```

```
#include "simcom_ audio.h"

void taskMain();
{
    sAPI_AudioPlaySampleRate(SAMPLE_RATE_16K);
}
```

22.2.2 sAPI_AudioPlay

This application interface is to audio file playback

sAPI_AudioPlay	
Prototype	BOOL **sAPI_AudioPlay**(char *file, BOOL direct, BOOL isSingle);
Parameters	[in] [**file**] Path and name of file, supporting PCM, WAV, AMR, MP3. [in] [**direct**] whether to play directly.(if set "True", stop current playback and play the new PCM data); [in] [**isSingle**] whether to play a single file("FALSE" means to play multiple files in a row.);
Return value	**true:** playback success. **false:** playback failed.
Header	#include "simcom_audio.h"

NOTE

For more details about the access to view the logs, please refer to Chapter 3 of the document A7600_Series_Open_SDK_Debug_and_Download_Application_Note.

Examples

```
#include "simcom_ audio.h"

void taskMain();
{
    char filename[25] = "C:/test.wav";
    BOOL result = false;

    //play C:/test.wav
    result = sAPI_AudioPlay(filename, 1,1);
    if(result);
      sAPI_Debug("%s start playing", filename);
    else
      sAPI_Debug("%s failed to play ", filename);
}
```

22.2.3 sAPI_AudioPlayMp3Cont

This application interface is to Mp3 continuous playback.

sAPI_AudioPlayMp3Cont	
Prototype	BOOL **sAPI_AudioPlayMp3Cont** (char *file, BOOL startplay, BOOL frame);
Parameters	[in] [**file**] Path and name of file, supporting MP3. [in] [**startplay**] whether to start playing.(if set "True",start playing mp3 filetable); [in] [**frame**] delete the first few frames of mp3 file(the number of frames deleted ranges from 0 to 2.);
Return value	**true:** playback or load success. **false:** playback or load failed.
Header	#include "simcom_audio.h"

Examples

```
#include "simcom_ audio.h"

void taskMain();
```

```
{
    char filename1[25] = "C:/test1.mp3";
    char filename2[25] = "C:/test2.mp3";
    char filename3[25] = "C:/test3.mp3";
    //load test1,test2 into filetable
    sAPI_AudioPlayMp3Cont (filename1, 0,2);
    sAPI_AudioPlayMp3Cont (filename2, 0,2);
    //load test3 and play all the mp3 files in the filetable
    sAPI_AudioPlayMp3Cont (filename3, 1,2);
}
```

22.2.4 sAPI_AudioStop

This application interface is to stop audio file playback

sAPI_AudioStop	
Prototype	BOOL **sAPI_AudioStop(**void**);**
Parameters	None
Return value	**true:** playback success. **false:** playback failed.
Header	#include "simcom_audio.h"

NOTE

For more details about the access to view the logs, please refer to Chapter 3 of the document A7600_Series_Open_SDK_Debug_and_Download_Application_Note.

Examples

```
#include "simcom_ audio.h"

void taskMain();
{
    BOOL result = false;

    //stop playing C:/test.wav
    result = sAPI_AudioStop();
    if(result);
       sAPI_Debug("audio file is stopped");
```

```
    else
        sAPI_Debug("audio file stop failed ");
}
```

22.2.5 sAPI_AudioRecord

This application interface is to audio file playback

sAPI_AudioRecord	
Prototype	BOOL **sAPI_AudioRecord**(BOOL enable, AUD_RecordSrcPath path, char *file);
Parameters	[in] [**enable**] recording or not, 0 is stop recording and 1 is starting to record. [in] [**path**] local recording, REC_PATH_LOCAL is local recording. [in] [**file**] Path and name of file.just saving to "C:/", just WAV and AMR.
Return value	**true:** start recording. **false:** failed to record.
Header	#include "simcom_audio.h"

NOTE

For more details about the access to view the logs, please refer to Chapter 3 of the document A7600_Series_Open_SDK_Debug_and_Download_Application_Note.

Examples

```
#include "simcom_ audio.h"

void taskMain();
{
    BOOL result = false;
    char filename[25] = "C:/ recording.wav";

    //record to C:/ recording.wav
    result = sAPI_AudioRecord(1, 0, filename);

    if(result);
        sAPI_Debug("recording is starting");
    else
        sAPI_Debug("recording failed ");
}
```

```
void taskStopRecording();
{
    BOOL result = false;

    //record to C:/ recording.wav
    result = sAPI_AudioRecord(0, 0, NULL);

    if(result);
      sAPI_Debug("recording is stopped");
    else
      sAPI_Debug("recording failed to stop ");
}
```

22.2.6 sAPI_AudioPcmPlay

This application interface is to play pcm stream directly.

sAPI_AudioPcmPlay	
Prototype	BOOL **sAPI_AudioPcmPlay**(char *buffer, UINT32 len, BOOL direct);
Parameters	[in] [**buffer**] PCM stream buffer. [in] [**len**] PCM stream buffer length. [in] [**direct**] whether to play directly.(if set "True", stop current playback and play the new PCM data);
Return value	**true:** start play success. **false:** start play fail.
Header	#include "simcom_audio.h"

Examples

```
#include "simcom_ audio.h"

void taskMain();
{
    char *buffer = NULL;
    UINT32 len;
    buffer = sAPI_Malloc(100*1024);
    len = 100*1024;
    …
    sAPI_AudioPcmPlay(buffer, len);
}
```

22.2.7 sAPI_AudioPcmStop

This application interface is to stop pcm stream directly.

sAPI_AudioPcmStop	
Prototype	BOOL **sAPI_AudioPcmStop(**void**);**
Parameters	None
Return value	**true:** playback success. **false:** playback failed.
Header	#include "simcom_audio.h"

NOTE

For more details about the access to view the logs, please refer to Chapter 3 of the document A7600_Series_Open_SDK_Debug_and_Download_Application_Note.

Examples

```
#include "simcom_ audio.h"

void taskMain();
{
    BOOL result = false;

    //stop playing pcm stream
    result = sAPI_AudioPcmStop();
    if(result);
      sAPI_Debug("pcm stream is stopped");
    else
      sAPI_Debug("pcm stream stop failed ");
}
```

22.2.8 sAPI_AudioSetVolume

This application interface is to set system volume.

sAPI_AudioSetVolume	
Prototype	void **sAPI_AudioSetVolume**(AUD_Volume volume);
Parameters	[in] [**volume**] system volume ,range is 0-11.
Return value	None
Header	#include "simcom_audio.h"

Examples

```
#include "simcom_ audio.h"
void taskMain();
{
    sAPI_AudioSetVolume(AUDIO_VOLUME_11);
}
```

22.2.9 sAPI_AudioGetVolume

This application interface is to get volume.

sAPI_AudioGetVolume	
Prototype	AUD_Volume **sAPI_AudioGetVolume**(void);
Parameters	None
Return value	volume level.
Header	#include "simcom_audio.h"

Examples

```
#include "simcom_ audio.h"
void taskMain();
{
    AUD_Volume volume = 0;
    …
    volume = sAPI_AudioGetVolume();
}
```

22.2.10 sAPI_PocInitLib

This application interface is to init poc function.

sAPI_PocInitLib	
Prototype	int **sAPI_PocInitLib**(sAPI_NotifyBufStateCb callback);
Parameters	[in] [**callback**] registers the callback to report the remaining data.
Return value	0: init success. -1: init fail.
Header	#include "simcom_audio.h"

Examples

```
#include "simcom_ audio.h"
sAPI_NotifyBufStateCb simcom_report(UINT32 dataLen);
{
    //print datalen info
}
void taskMain();
{
    Int result = 0;
    …
    result = sAPI_PocInitLib(simcom_report);
}
```

22.2.11 sAPI_PocPlaySound

This application interface is to play poc PCM buffer.

sAPI_PocPlaySound	
Prototype	int **sAPI_PocPlaySound**(const char *data, int length);
Parameters	[in] [**data**] PCM data buffer. [in] [**length**] PCM data length.
Return value	**0:** play success. **-1:** play fail.
Header	#include "simcom_audio.h"

Examples

```
#include "simcom_ audio.h"

void taskMain();
{
    UINT16 pcmBuffer[320] ;
    Int result = 0;
    memset(pcmBuffer, 0xff, sizeof(pcmBuffer););
    …
    result = sAPI_PocPlaySound(pcmBuffer, 320);
}
```

22.2.12 sAPI_PocStopSound

This application interface is to stop playing poc PCM buffer.

sAPI_PocStopSound	
Prototype	int **sAPI_PocStopSound**(void**);**
Parameters	None
Return value	**0:** stop success. **-1:** stop fail.
Header	#include "simcom_audio.h"

Examples

```
#include "simcom_ audio.h"

void taskMain();
{
    Int result = sAPI_PocStopSound();
}
```

22.2.13 sAPI_PocGetPcmAvail

This application interface is to get available free size.

sAPI_PocGetPcmAvail	
Prototype	int **sAPI_PocGetPcmAvail**(void**);**
Parameters	None

Return value	The size of remaining cache.
Header	#include "simcom_audio.h"

Examples

```
#include "simcom_ audio.h"

void taskMain();
{
    Int freeSize = sAPI_PocGetPcmAvail();
}
```

22.2.14 sAPI_PocCleanBufferData

This application interface is to clean buffer data.

sAPI_PocCleanBufferData	
Prototype	int **sAPI_PocCleanBufferData**(void**);**
Parameters	None
Return value	**0:** stop success. **-1:** stop fail.
Header	#include "simcom_audio.h"

Examples

```
#include "simcom_ audio.h"

void taskMain();
{
    Int result = sAPI_PocCleanBufferData();
}
```

22.2.15 sAPI_PocStartRecord

This application interface is to start recording.

sAPI_PocStartRecord

Prototype	int **sAPI_PocStartRecord**(sAPI_ProcessRecordCb callback,int mode);
Parameters	[in] [**callback**] registers the callback to report the recording data. [in] [**mode**] active or passive reporting of data.
Return value	**0:** start success. **-1:** stop fail.
Header	#include "simcom_audio.h"

Examples

```
#include "simcom_ audio.h"

sAPI_ProcessRecordCb simcom_get_record(char *buffer, int len);
{
    //get recording data
}
void taskMain();
{
    char
    //active
    Int result = sAPI_poc_start_record(simcom_get_record,1);

    //passive
    Int result = sAPI_poc_start_record(NULL,2);
    //get record data
    While(1)
    {
        result = sAPI_PocPcmRead(buffer,320);
    }
}
```

22.2.16 sAPI_PocStopRecord

This application interface is to stop recording.

sAPI_PocStopRecord	
Prototype	int **sAPI_PocStopRecord**(void);
Parameters	None
Return value	**0:** start success. **-1:** stop fail.
Header	#include "simcom_audio.h"

Examples

```
#include "simcom_ audio.h"

void taskMain();
{
    Int result = sAPI_PocStopRecord();
}
```

22.2.17 sAPI_PocPcmRead

This application interface is to get record data.

sAPI_PocPcmRead	
Prototype	int **sAPI_PocPcmRead**(char *buffer,int dataLen);
Parameters	[out] [buffer] recording data buffer. [in] [**dataLen**] just 320(8K16bit).
Return value	**0:** start success. **-1:** stop fail.
Header	#include "simcom_audio.h"

Examples

```
#include "simcom_ audio.h"

void taskMain();
{
    Char *buffer
    //start recod function before.
    Int result = sAPI_PocPcmRead(buffer,320);
}
```

22.2.18 sAPI_AudioStatus

This application interface is to get audio status.

sAPI_AudioStatus

Prototype	UINT8 **sAPI_AudioStatus**(void);
Parameters	None
Return value	**0:** audio is idle **1:** audio is working
Header	#include "simcom_audio.h"

Examples

```
#include "simcom_ audio.h"

void taskMain();
{
    UINT8 status;
    //start recod function before.
    status = sAPI_AudioStatus();
}
```

22.2.19 sAPI_AudRec

This application interface is to record with time duration.

sAPI_AudRec	
Prototype	BOOL **sAPI_AudRec**(UINT8 oper,char *file,UINT8 duration, sAPI_AudRecCallback callback);
Parameters	[in] [**oper**] recording operation:1(start),0(stop). [in] [**file**] file name and path like C:/test.wav. [in] [**duration**] recording time duration.recording wav duration is 1 to 15s and recording amr duaration is 1 to 180s. [in] [**callback**] get recording status (just stop recording).
Return value	**0:** start successfully. **1:** start failed.
Header	#include "simcom_audio.h"

Examples

```
#include "simcom_ audio.h"

Void RecStatus(UINT8 status)
{
    sAPI_Debug(“recording status is %d”,status);
```

```
}
void taskMain();
{
    Char filename[20] = “C:/test.wav”;
    sAPI_AudRec(1, filename,15, RecStatus);
    sAPI_TaskSleep(2000);//recording 10s
    sAPI_ AudRec(0, NULL,0, NULL);
}
```

22.2.20 sAPI_AudioSetAmrEncodeRate

This application interface is to set recording amr encode rate(this api just for sAPI_AudioRecord).

sAPI_AudioSetAmrEncodeRate	
Prototype	void **sAPI_AudioSetAmrEncodeRate**(AudioAmrEncodeRate rate);
Parameters	[in] [**rate**] amr encode rate.default is REC_AMR_MR122. typedef enum{ REC_AMR_MR475, REC_AMR_MR515, REC_AMR_MR59, REC_AMR_MR67, REC_AMR_MR74, REC_AMR_MR795, REC_AMR_MR102, REC_AMR_MR122, }AudioAmrEncodeRate;
Return value	**None**
Header	#include "simcom_audio.h"

Examples

```
#include "simcom_ audio.h"
void taskMain();
{
    sAPI_AudioSetAmrEncodeRate (REC_AMR_MR475);
}
```

22.2.21 sAPI_AudioSetMicGain

This application interface is to set microphone gain.

sAPI_AudioSetMicGain	
Prototype	void **sAPI_AudioSetMicGain**(UINT32 micgain);
Parameters	[in] [**micgain**] microphone ,range is 0-7.
Return value	None
Header	#include "simcom_audio.h"

Examples

```
#include "simcom_ audio.h"
void taskMain();
{
    sAPI_AudioSetVolume(7);
}
```

22.2.22 sAPI_AudioGetMicGain

This application interface is to get microphone gain.

sAPI_AudioGetMicGain	
Prototype	int **sAPI_AudioGetMicGain**(void);
Parameters	None
Return value	microphone level.
Header	#include "simcom_audio.h"

Examples

```
#include "simcom_ audio.h"
void taskMain();
{
    UINT32 micphone = 0;
    …
    micphone = sAPI_AudioGetVolume();
}
```

22.2.23 sAPI_AudioMp3StreamPlay

This application interface is to play mp3 data buffer,just for 1603 OpenSDK.

sAPI_AudioMp3StreamPlay	
Prototype	BOOL **sAPI_AudioMp3StreamPlay**(char *buffer, UINT32 len, BOOL direct);
Parameters	[in] [**buffer**] MP3 stream buffer. [in] [**len**] MP3 stream buffer length. [in] [**direct**] whether to play directly.(if set "True", stop current playback and play the new MP3 data);
Return value	**true:** start play success. **false:** start play fail.
Header	#include "simcom_ audio.h"

Examples

```
#include "simcom_ audio.h"
void taskMain();
{
    char *buffer = NULL;
    UINT32 len;
    buffer = sAPI_Malloc(100*1024);
    len = 100*1024;
    …
    sAPI_AudioMp3StreamPlay(buffer, len);
}
```

22.2.24 sAPI_AudioMp3StreamStop

This application interface is to stop mp3 data buffer playback,just for 1603 OpenSDK.

sAPI_AudioMp3StreamStop	
Prototype	BOOL **sAPI_AudioMp3StreamStop**(void);
Parameters	None
Return value	**true:** start play success. **false:** start play fail.
Header	#include "simcom_audio.h"

Examples

```
#include "simcom_ audio.h"
void taskMain();
{
    BOOL result = false;

    //stop playing mp3 stream
    result = sAPI_AudioMp3StreamStop();
    if(result);
      sAPI_Debug("mp3 stream is stopped");
    else
      sAPI_Debug("mp3 stream stop failed ");
}
```

22.2.25 sAPI_AudioAmrStreamPlay

This application interface is to play amr data buffer,just for 1603 OpenSDK.

sAPI_AudioAmrStreamPlay	
Prototype	BOOL **sAPI_AudioAmrStreamPlay**(char *buffer, UINT32 len, BOOL direct);
Parameters	[in] [**buffer**] AMR stream buffer. [in] [**len**] AMR stream buffer length. [in] [**direct**] whether to play directly.(if set "True", stop current playback and play the new AMR data);
Return value	**true:** start play success. **false:** start play fail.
Header	#include "simcom_ audio.h"

Examples

```
#include "simcom_ audio.h"
void taskMain();
{
    char *buffer = NULL;
    UINT32 len;
    buffer = sAPI_Malloc(100*1024);
    len = 100*1024;
    …
    sAPI_AudioAmrStreamPlay(buffer, len);
}
```

22.2.26 sAPI_AudioAmrStreamStop

This application interface is to stop amr data buffer playback,just for 1603 OpenSDK.

sAPI_AudioAmrStreamStop	
Prototype	BOOL **sAPI_AudioAmrStreamStop**(void);
Parameters	None
Return value	**true:** start play success. **false:** start play fail.
Header	#include "simcom_audio.h"

Examples

```
#include "simcom_ audio.h"
void taskMain();
{
    BOOL result = false;

    //stop playing amr stream
    result = sAPI_AudioAmrStreamStop();
    if(result);
      sAPI_Debug("amr stream is stopped");
    else
      sAPI_Debug("amr stream stop failed ");
}
```

22.2.27 sAPI_AudioPlayAmrCont

This application interface is to Amr continuous playback.

sAPI_AudioPlayAmrCont	
Prototype	BOOL **sAPI_AudioPlayAmrCont** (char *file, BOOL startplay);
Parameters	[in] [**file**] Path and name of file, supporting AMR. [in] [**startplay**] whether to start playing.(if set "True",start playing amr filetable);
Return value	**true:** playback or load success. **false:** playback or load failed.
Header	#include "simcom_audio.h"

Examples

```
#include "simcom_ audio.h"

void taskMain();
{
    char filename1[25] = "C:/test1.amr";
    char filename2[25] = "C:/test2.amr";
    char filename3[25] = "C:/test3.amr";
    //load test1,test2 into filetable
    sAPI_AudioPlayAmrCont (filename1, 0);
    sAPI_AudioPlayAmrCont (filename2, 0);
    //load test3 and play all the amr files in the filetable
    sAPI_AudioPlayAmrCont (filename3, 1);
}
```

22.2.28 sAPI_AudioWavFilePlay

This application interface is to play wav files with other sampling rates except 8k and 16k.

sAPI_AudioWavFilePlay	
Prototype	BOOL **sAPI_AudioWavFilePlay** (char *file, BOOL direct);
Parameters	[in] [**file**] Path and name of file, supporting wav. [in] [**direct**] whether to play directly.(if set "True", stop current playback and play the new wav file);
Return value	**true:** playback success. **false:** playback failed.
Header	#include "simcom_audio.h"

Examples

```
#include "simcom_ audio.h"

void taskMain();
{
    char filename[25] = "C:/test.wav";
    BOOL result = false;

    //play C:/test.wav (the sampling rate is 44.1k)
    result = sAPI_AudioWavFilePlay(filename, 1);
```

```
    if(result);
      sAPI_Debug("%s start playing", filename);
    else
      sAPI_Debug("%s failed to play ", filename);
}
```

22.2.29 sAPI_AudioSetPlayPath

This application interface is to set local playback or remote playback.

sAPI_AudioSetPlayPath	
Prototype	BOOL **sAPI_AudioSetPlayPath**(UINT8 path);
Parameters	[in] [**path**]playback path , 0 is local and 1 is remote.
Return value	**true:** set successfully. **false:** setting failed.
Header	#include "simcom_audio.h"

Examples

```
#include "simcom_ audio.h"
#include “simcom_call.h”

void taskMain();
{
    char filename[25] = "C:/test.wav";
    BOOL result = false;

    //call 199*********
    sAPI_CallDialMsg(“199********”, msgQ);
    ……
    //call status is active.
    //set playback path
    sAPI_AudioSetPlayPath(1);
    //play C:/test.wav
    result = sAPI_AudioPlay(filename, 1,1);
    if(result);
      sAPI_Debug("%s start playing", filename);
    else
      sAPI_Debug("%s failed to play ", filename);
}
```

22.2.30 sAPI_AudioGetPlayPath

This application interface is to get playback path value.

sAPI_AudioGetPlayPath	
Prototype	UINT8 **sAPI_AudioGetPlayPath** (void);
Parameters	NONE
Return value	0 is local. 1 is remote.
Header	#include "simcom_audio.h"

Examples

```
#include "simcom_ audio.h"

void taskMain();
{
    UINT8 path = sAPI_AudioGetPlayPath().
    sAPI_Debug("play path is %d",path);
}
```

23 APIs for TTS

23.1 Overview of APIs for TTS

Interface	Description
sAPI_TTSPlay	Play TTS text
sAPI_TTSStop	Stop TTS text playback
sAPI_TTSSetParameters	Set TTS parameters
sAPI_TTSSetStatusCallBack	Set TTS playback status callback function.

23.2 Detailed Description of APIs for TTS

23.2.1 sAPI_TTSPlay

This application interface is to play TTS text.

sAPI_TTSPLay	
Prototype	BOOL **sAPI_TTSPlay**(UINT8 option, char *inputText, UINT8 playMode);
Parameters	[in] [**option**] text code, 1 is to UNICODE and 2 is to ASCII or GBK. [in] [**inputText**] text to play.Max length is 508 byte. Please segment the longer English text so that the data cannot be read in its entirety. [in] [**playMode**] play path, 0 is local playing and 1 is remote playing.
Return value	**true:** playback success. **false:** playback failed.
Header	#include "simcom_tts_api.h"

NOTE

For more details about the access to view the logs, please refer to Chapter 3 of the document A7600_Series_Open_SDK_Debug_and_Download_Application_Note.

Examples

```
#include "simcom_tts_api.h"

void taskMain();
{
    BOOL result = false;

    //play "1234567890"
    result = sAPI_TTSPlay(2, "1234567890", 0);
    if(result);
      sAPI_Debug("TTS start playing");
    else
      sAPI_Debug("TTS failed to play ");
}
```

23.2.2 sAPI_TTSStop

This application interface is to stop tts playback

sAPI_AudioStop	
Prototype	BOOL **sAPI_TTSStop(**void);
Parameters	None
Return value	**true:** stop success. **false:** stop failed.
Header	#include "simcom_tts_api.h"

NOTE

For more details about the access to view the logs, please refer to Chapter 3 of the document A7600_Series_Open_SDK_Debug_and_Download_Application_Note.

Examples

```
#include "simcom_ tts_api.h"
```

```
void taskMain();
{
    BOOL result = false;

    //stop tts playing
    result = sAPI_TTSStop();
    if(result);
      sAPI_Debug("tts is stopped");
    else
      sAPI_Debug("tts stop failed ");
}
```

23.2.3 sAPI_TTSSetParameters

This application interface is to set TTS parameters.

sAPI_TTSSetParameters	
Prototype	BOOL **sAPI_TTSSetParameters**(UINT8 volume, UINT8 sysVolume, UINT8 digitMode, UINT8 pitch, UINT8 speed);
Parameters	[in] [**volume**] system volume, range is 0-2, default is 1. [in] [**sysVolume**] tts volume, range is 0-3, default is 3. [in] [**digitMode**] digit read mode, range is 0-3, default is 0. [in] [**pitch**] voice tone, range is 0-2, default is 1. [in] [**speed**] voice speed, range is 0-2, default is 1.
Return value	**true:** start recording. **false:** failed to record.
Header	#include "simcom_tts_api.h"

NOTE

For more details about the access to view the logs, please refer to Chapter 3 of the document A7600_Series_Open_SDK_Debug_and_Download_Application_Note.

Examples

```
#include "simcom_ tts_api.h"

void taskMain();
{
```

```
    BOOL result = false;

    //set parameters
    result = sAPI_ TTSSetParameters(2, 2, 1, 0, 0);

    if(result);
      sAPI_Debug("set parameters success");
    else
      sAPI_Debug("set parameters failed ");
}
```

23.2.4 sAPI_TTSSetStatusCallBack

This application interface is to set TTS status callback

sAPI_TTSSetStatusCallBack	
Prototype	BOOL **sAPI_TTSSetStatusCallBack**(sAPI_TTSStatussCb ttsCb);
Parameters	[in] [**ttsCb**] registers the callback to report TTS status.
Return value	**true:** register successfully. **false:** failed to register.
Header	#include "simcom_tts_api.h"

NOTE

For more details about the access to view the logs, please refer to Chapter 3 of the document A7600_Series_Open_SDK_Debug_and_Download_Application_Note.

Examples

```
#include "simcom_ tts_api.h"
sAPI_TTSStatussCb simcom_get_TTSStatus(TTSStatus flag)
{
    //get tts status flag
}

void taskMain();
{
    BOOL result = false;
```

```
    //set status callback
    result = sAPI_ TTSSetStatusCallBack(simcom_get_TTSStatus);
    //start TTS function

    if(result);
      sAPI_Debug("set parameters success");
    else
      sAPI_Debug("set parameters failed ");
}
```

24 APIs for OTA

24.1 Overview of APIs for OTA

Interface	Description
sAPI_AppPackageOpen	Open OTA partition
sAPI_AppPackageWrite	Write data to OTA partition
sAPI_AppPackageRead	Read data from OTA partition
sAPI_AppPackageClose	Close OTA partition
sAPI_AppDownload	Download firmware from server
sAPI_AppPackageCrc	Check CRC for app ota pacakage
sAPI_FOTAServiceBegin	Start kernel's OTA service

24.2 Detailed Description of APIs for OTA

24.2.1 sAPI_AppPackageOpen

This application interface is to open OTA partition, you should use this api before write or read.

sAPI_AppPackageOpen	
Prototype	int **sAPI_AppPackageOpen**(char *mode);
Parameters	[in] mode: Open partition mode. w: Open OTA partition in write mode. r: Open OTA partition in read mode
Return value	SC_SUCCESS: Open OTA partition successfully. SC_FAIL: Fail to open OTA partition.
Header	#include "simcom_app_updater.h"

Examples

```
#include "simcom_ app_updater.h"
```

```
SC_STATUS status = SC_SUCCESS;
char mode[2] = {0};
mode[0] = 'w';
sAPI_Debug("mode[0] = %c", mode[0] );
status = sAPI_AppPackageOpen(mode);
if(SC_FAIL == status);
{
    sAPI_Debug("open appPackage fail!");
    return SC_FAIL;
}
```

24.2.2 sAPI_AppPackageWrite

This application interface is to write data to OTA partition.

sAPI_AppPackageWrite	
Prototype	int **sAPI_AppPackageWrite**(char * data, unsigned int size);
Parameters	[in] data: OTA data. [in] size: The size of data.
Return value	SC_SUCCESS: write OTA partition successfully.. SC_FAIL: Fail to write OTA partition.
Header	#include "simcom_app_updater.h"

Examples

```
#include "simcom_ app_updater.h"

SC_STATUS status = SC_SUCCESS;
UINT8 buff[1024] = {0};
UINT32 actul_read_len = 0;
SCfileNameInfo file_name = {0};
memcpy(file_name.name, "test.bin", strlen("test.bin"));
memcpy(file_name.path, "c:/", strlen("c:/"));
file_hdl = sAPI_FsFileOpen(&file_name, "rb");
actul_read_len = sAPI_FsFileRead(buff, 1024, file_hdl);
status = sAPI_AppPackageWrite(buff, actul_read_len);
if(SC_FAIL == status);
{
    sAPI_Debug("write appPackage fail!");
}
```

24.2.3 sAPI_AppPackageRead

This application interface is to read data from OTA partition.

sAPI_AppPackageRead	
Prototype	int **sAPI_AppPackageRead**(char * data, unsigned int size);
Parameters	[in] data: OTA data. [in] size: The size of data.
Return value	SC_SUCCESS: read OTA partition successfully.. SC_FAIL: Fail to read OTA partition.
Header	#include "simcom_app_updater.h"

Examples

```
#include "simcom_ app_updater.h"

SC_STATUS status = SC_SUCCESS;
UINT8 buff[1024] = {0};
UINT32 actul_read_len = 0;
status = sAPI_AppPackageRead(buff, actul_read_len);
if(SC_FAIL == status);
{
    sAPI_Debug("read appPackage fail!");
}
```

24.2.4 sAPI_AppPackageClose

This application interface is to close OTA partition.

sAPI_AppPackageClose	
Prototype	int **sAPI_AppPackageClose(**void);
Parameters	None
Return value	SC_SUCCESS: Close OTA partition successfully.. SC_FAIL: Fail to close OTA partition.
Header	#include "simcom_app_updater.h"

Examples

```
#include "simcom_ app_updater.h"

SC_STATUS status = SC_SUCCESS;
status = sAPI_AppPackageClose();
if(SC_FAIL == status);
{
    sAPI_Debug("close appPackage fail!");
}
```

24.2.5 sApi_AppDownload

This application interface is download firmware from server

sApi_AppDownload	
Prototype	SCAppDwonLoadReturnCode **sApi_AppDownload**(SCAppDownloadPram *pram);
Parameters	[in] pram: app download prameter.
Return value	SC_ APP_DOWNLOAD _SUCESSED: successfully.. other: Fail.
Header	#include "simcom_app_download.h"

Examples

```
#include "simcom_app_download.h"

SCAppDownloadPram pram;
SCAppDwonLoadReturnCode ret;

pram.mod = SC_APP_DOWNLOAD_HTTP_MOD;
pram.url = "http://183.230.174.137:80/bin/customer_app.bin";
ret = sApi_AppDownload(&pram);
if(ret == SC_APP_DOWNLOAD_SUCESSED);
{
    sAPI_Debug("http download app sucess");
}
```

NOTE

This interface is a blocking function

24.2.6 sAPI_AppPackageCrc

This application interface is to check crc for app ota pacakage.

sAPI_AppPackageCRC	
Prototype	int **sAPI_AppPackageCrc** (SCAppPackageInfo *pInfo);
Parameters	[in] pInfo: app package information.
Return value	SC_ APP_DOWNLOAD _SUCESSED: successfully.. other: Fail.
Header	#include "simcom_app_updater.h"

Examples

```
#include "simcom_app_download.h"

SCAppDownloadPram pram;
SCAppDwonLoadReturnCode ret;

pram.mod = SC_APP_DOWNLOAD_HTTP_MOD;
pram.url = "http://183.230.174.137:80/bin/customer_app.bin";
ret = sApi_AppDownload(&pram);
if(ret == SC_APP_DOWNLOAD_SUCESSED);
{
    sAPI_Debug("http download app sucess");
}
```

NOTE

This interface is a blocking function

24.2.7 sAPI_FotaServiceBegin

This application interface is used to start kernel's OTA service.(not app);

sApi_ FotaServiceBegin	
Prototype	int **sAPI_FotaServiceBegin(**void* pram);
Parameters	[in] pram: a struct SC_FotaApiParam pointer.
Return value	0: service begin successfully.. -1: param invalid. -2: malloc memory failed. -3: pdp active failed -4: send fota request failed.
Header	#include "simcom_fota.h"

Examples

```
int fotacb(int isok);
{
    if(isok);
    {
        PrintfResp("fota is successful, please reset.\r\n");
    }
    else
    {
        PrintfResp("fota failed, try again\r\n");
    }
    return 0;

}
        void Examples();{
            struct SC_FotaApiParam param = {0};
            strcpy(param.host, "183.230.174.137:6047/fbf_dfota.bin");
            strcpy(param.username, "huyujie");
            strcpy(param.password, "huyujie626");
            param.mode = 0;
            param.sc_fota_cb = fotacb;
            int ret = sAPI_FotaServiceBegin((void *);&param);
            if(ret == -1);
                PrintfResp("\r\nparam error\r\n");
            else if(ret == -2);
                PrintfResp("\r\nmalloc mem error\r\n");
            else if(ret == -3);
                PrintfResp("\r\nnet error\r\n");
            break;
}
```

NOTE

This interface is a blocking function. And fotacb will be called when fota service end.

25 APIs for GNSS

25.1 Overview of APIs for GNSS

Interface	Description
sAPI_GnssPowerStatusSet	Set GNSS's power status
sAPI_GnssPowerStatusGet	Get GNSS's power status
sAPI_GnssOutput2NmeaPort	Send nmea date from UART3 to NMEA port
sAPI_GnssStartMode	Set GNSS's start mode
sAPI_GnssBaudRateSet	Set the baud rate of GNSS
sAPI_GnssBaudRateGet	Get the baud rate of GNSS
sAPI_GnssModeSet	Set GNSS's mode
sAPI_GnssModeGet	Get GNSS's mode
sAPI_GnssNmeaRateSet	Set the update rate of GNSS's nmea data
sAPI_GnssNmeaRateGet	Get the update rate of GNSS's nmea data
sAPI_GnssNmeaSentenceSet	Set the nmea sentence of GNSS
sAPI_GnssNmeaSentenceGet	Get the nmea sentence of GNSS
sAPI_GpsInfoGet	Get the information of GPS
sAPI_GnssInfoGet	Get the information of GNSS
sAPI_SendCmd2Gnss	Send nmea command directly to GNSS
sAPI_GnssAgpsSeviceOpen	Get AGPS data from the AGNSS server for assisted positioning

25.2 Detailed Description of APIs for GNSS

25.2.1 sAPI_GnssPowerStatusSet

This application interface is used to set GNSS's power status, default value is SC_GNSS_POWER_OFF, to enable GNSS by setting SC_GNSS_POWER_ON.

sAPI_GnssPowerStatusSet		
Prototype	SC_Gnss_Return_Code	**sAPI_GnssPowerStatusSet**(SC_Gnss_Power_Status

	power);
Parameters	[in] **power**: set GNSS power on/off
Return value	SC_GNSS_RETURN_CODE_OK: set done. SC_GNSS_RETURN_CODE_ERROR: set fail.
Header	#include "simcom_gps.h"

Examples

```
#include "simcom_api.h"
#include "simcom_gps.h"
#include "simcom_common.h"

if(SC_GNSS_RETURN_CODE_OK != sAPI_GnssPowerStatusSet(SC_GNSS_POWER_ON);
{
    sAPI_Debug("GNSS power on error!");
}
if(SC_GNSS_RETURN_CODE_OK != sAPI_GnssPowerStatusSet(SC_GNSS_POWER_OFF);
{
    sAPI_Debug("GNSS power off error!");
}
```

25.2.2 sAPI_GnssPowerStatusGet

This application interface is used to get GNSS's power status.

sAPI_GnssPowerStatusGet	
Prototype	SC_Gnss_Power_Status **sAPI_GnssPowerStatusGet**(void);
Parameters	None
Return value	0: power off. 1: power on.
Header	#include "simcom_gps.h"

Examples

```
#include "simcom_api.h"
#include "simcom_gps.h"
#include "simcom_common.h"

SC_Gnss_Power_Status pwr;
pwr =   sAPI_GnssPowerStatusGet();
```

sAPI_Debug("GNSS power status is %d!", pwr);

25.2.3 sAPI_GnssNmeaDataGet

This application interface is used to get NMEA data by NMEA port or URC.

sAPI_GnssNmeaDataGet	
Prototype	SC_Gnss_Return_Code **sAPI_GnssNmeaDataGet**(SC_Gnss_Output_Control ctl, SC_Gnss_Nmea_Data_Get mode);
Parameters	[in] **ctl**: control whether the data is output. [in] **mode**: get NMEA data by NMEA port or URC report.
Return value	SC_GNSS_RETURN_CODE_OK: get done. SC_GNSS_RETURN_CODE_ERROR: get fail.
Header	#include "simcom_gps.h"

Examples

```
#include "simcom_api.h"
#include "simcom_gps.h"
#include "simcom_common.h"

if(SC_GNSS_RETURN_CODE_OK !=
sAPI_GnssNmeaDataGet(SC_GNSS_START_OUTPUT_NMEA_DATA,
SC_GNSS_NMEA_DATA_GET_BY_PORT);
{
    sAPI_Debug("start get nmea data by NMEA port error!");
}
if(SC_GNSS_RETURN_CODE_OK !=
sAPI_GnssNmeaDataGet(SC_GNSS_START_OUTPUT_NMEA_DATA,
SC_GNSS_NMEA_DATA_GET_BY_URC);
{
    sAPI_Debug("start get nmea data by URC error!");
}
```

25.2.4 sAPI_GnssStartMode

This application interface is used to start GNSS.

sAPI_GnssStartMode	
Prototype	SC_Gnss_Return_Code **sAPI_GnssStartMode**(SC_Gnss_Start_Mode mode);
Parameters	[in] **mode**: select GNSS's start mode.
Return value	SC_GNSS_RETURN_CODE_OK: set done. SC_GNSS_RETURN_CODE_ERROR: set fail.
Header	#include "simcom_gps.h"

NOTE

ASR1601: Support cold start and hot start
ASR1603: Support cold start, warm start and hot start

Examples

```
#include "simcom_api.h"
#include "simcom_gps.h"
#include "simcom_common.h"

if(SC_GNSS_RETURN_CODE_OK != sAPI_GnssStartMode(SC_GNSS_START_HOT);
{
    sAPI_Debug("GNSS hot start error!");
}
if(SC_GNSS_RETURN_CODE_OK != sAPI_GnssStartMode(SC_GNSS_START_COLD);
{
    sAPI_Debug("GNSS cold start error!");
}
```

25.2.5 sAPI_GnssBaudRateSet

This application interface is used to set the baud rate of GNSS.

sAPI_GnssBaudRateSet	
Prototype	SC_Gnss_Return_Code **sAPI_GnssBaudRateSet**(SC_Gnss_Baud_Rate baudrate);
Parameters	[in] **baudrate**: the baud rate of GNSS.
Return value	SC_GNSS_RETURN_CODE_OK: set done. SC_GNSS_RETURN_CODE_ERROR: set fail.
Header	#include "simcom_gps.h"

NOTE

ASR1601: Support 4800, 9600, 19200, 38400, 57600, 115200
ASR1603: Support 9600, 115200, 230400

Examples

```
#include "simcom_api.h"
#include "simcom_gps.h"
#include "simcom_common.h"

if(SC_GNSS_RETURN_CODE_OK != sAPI_GnssBaudRateSet(SC_GNSS_BAUD_RATE_19200);
{
    sAPI_Debug("set GNSS's baud rate error!");
}
```

25.2.6 sAPI_GnssBaudRateGet

This application interface is used to get GNSS's baud rate.

sAPI_GnssBaudRateGet	
Prototype	SC_Gnss_Baud_Rate **sAPI_GnssBaudRateGet**(void);
Parameters	None
Return value	SC_Gnss_Baud_Rate
Header	#include "simcom_gps.h"

NOTE

ASR1601: Support 4800, 9600, 19200, 38400, 57600, 115200
ASR1603: Support 9600, 115200, 230400

Examples

```
#include "simcom_api.h"
#include "simcom_gps.h"
#include "simcom_common.h"
```

```
SC_Gnss_Baud_Rate baudrate;
baudrate = sAPI_GnssBaudRateGet();
sAPI_Debug("GNSS's baud rate is %d!", baudrate);
```

25.2.7 sAPI_GnssModeSet

This application interface is used to set the mode of GNSS.

sAPI_GnssModeSet	
Prototype	SC_Gnss_Return_Code **sAPI_GnssModeSet**(SC_Gnss_Mode mode);
Parameters	[in] **mode**: the mode of GNSS.
Return value	SC_GNSS_RETURN_CODE_OK: set done. SC_GNSS_RETURN_CODE_ERROR: set fail.
Header	#include "simcom_gps.h"

NOTE

ASR1601: Support GPS, BDS, GPS+BDS, GLONASS, GPS+GLONASS, BDS+GLONASS, GPS+BDS+GLONASS
ASR1603: Support GPS+BDS+QZSS, BDS, GPS+QZSS

Examples

```
#include "simcom_api.h"
#include "simcom_gps.h"
#include "simcom_common.h"

if(SC_GNSS_RETURN_CODE_OK != sAPI_GnssModeSet(SC_GNSS_MODE_GPS);
{
    sAPI_Debug("set GNSS's mode error!");
}
```

25.2.8 sAPI_GnssModeGet

This application interface is used to get the mode supported by GNSS.

sAPI_GnssModeGet	
Prototype	SC_Gnss_Mode **sAPI_GnssModeGet**(void);
Parameters	None
Return value	SC_Gnss_Mode
Header	#include "simcom_gps.h"

NOTE

ASR1601: Support GPS, BDS, GPS+BDS, GLONASS, GPS+GLONASS, BDS+GLONASS, GPS+BDS+GLONASS
ASR1603: Support GPS+BDS+QZSS, BDS, GPS+QZSS

Examples

```
#include "simcom_api.h"
#include "simcom_gps.h"
#include "simcom_common.h"

SC_Gnss_Mode mode;
mode = sAPI_GnssModeGet();
sAPI_Debug("GNSS's mode is %d!", mode);
```

25.2.9 sAPI_GnssNmeaRateSet

This application interface is used to set the nmea's update rate of GNSS.

sAPI_GnssNmeaRateSet	
Prototype	SC_Gnss_Return_Code **sAPI_GnssNmeaRateSet**(SC_Gnss_Nmea_Rate rate);
Parameters	[in] **rate**: the nmea's update rate of GNSS.
Return value	SC_GNSS_RETURN_CODE_OK: set done. SC_GNSS_RETURN_CODE_ERROR: set fail.
Header	#include "simcom_gps.h"

NOTE

ASR1601: Support 1Hz, 2Hz, 4Hz, 5Hz, 10Hz
ASR1603: Support 1Hz, 2Hz, 5Hz, 10Hz

Examples

```
#include "simcom_api.h"
#include "simcom_gps.h"
#include "simcom_common.h"

if(SC_GNSS_RETURN_CODE_OK !=sAPI_GnssNmeaRateSet(SC_GNSS_NMEA_UPDATE_RATE_5HZ)
;
{
    sAPI_Debug("set nmea's update rate error!");
}
```

25.2.10 sAPI_GnssNmeaRateGet

This application interface is used to get the nmea's update rate of GNSS.

sAPI_GnssNmeaRateGet	
Prototype	SC_Gnss_Nmea_Rate **sAPI_GnssNmeaRateGet**(void);
Parameters	None
Return value	SC_Gnss_Nmea_Rate
Header	#include "simcom_gps.h"

NOTE

ASR1601: Support 1Hz, 2Hz, 4Hz, 5Hz, 10Hz
ASR1603: Support 1Hz, 2Hz, 5Hz, 10Hz

Examples

```
#include "simcom_api.h"
#include "simcom_gps.h"
#include "simcom_common.h"

SC_Gnss_Nmea_Rate    rate;
rate = sAPI_GnssNmeaRateGet();
sAPI_Debug("nmea's update rate is %d!", rate);
```

25.2.11 sAPI_GnssNmeaSentenceSet

This application interface is used to set the nmea's sentence type of GNSS.

sAPI_GnssNmeaSentenceSet	
Prototype	SC_Gnss_Return_Code **sAPI_GnssNmeaSentenceSet**(unsigned short mask);
Parameters	[in] **mask**: the mask of nmea's sentence type, 1 means enable the corresponding field, 0 means disable the corresponding, the ASR1601 platform range is 0-13311,the ASR1603 platform range is 0-63.
Return value	SC_GNSS_RETURN_CODE_OK: set done. SC_GNSS_RETURN_CODE_ERROR: set fail.
Header	#include "simcom_gps.h"

NOTE

ASR1601: 0 means no output, 1 means output this sentence. default mask is 255(00000011111111)
- bit0-GGA, default is 1
- bit1-GLL, default is 1
- bit2-GSA, default is 1
- bit3-GSV, default is 1
- bit4-RMC, default is 1
- bit5-VTG, default is 1
- bit6-ZDA, default is 1
- bit7-ANT, default is 1
- bit8-DHV, default is 0
- bit9-LPS, default is 0(nonsupport)
- bit10-res1, default is 0
- bit11-res2, default is 0
- bit12-UTC, default is 0(nonsupport)
- bit13-GST, default is 0

ASR1603: 0 means no output, 1 means output this sentence. default mask is 63(00111111)
- bit0-GGA, default is 1
- bit1-GLL, default is 1
- bit2-GSA, default is 1
- bit3-GSV, default is 1
- bit4-RMC, default is 1
- bit5-VTG, default is 1
- bit6-ZDA, default is 0
- bit7-GST, default is 0

Examples

```
#include "simcom_api.h"
#include "simcom_gps.h"
#include "simcom_common.h"

if(SC_GNSS_RETURN_CODE_OK !=sAPI_GnssNmeaSentenceSet(13310);
{
    sAPI_Debug("set nmea's sentence type error!");
}
```

25.2.12 sAPI_GnssNmeaSentenceGet

This application interface is used to get the nmea's sentence type of GNSS.

sAPI_GnssNmeaSentenceGet	
Prototype	UINT8* **sAPI_GnssNmeaSentenceGet**(void);
Parameters	None
Return value	The header address of array. arr[0] -GGA arr[1] -GLL arr[2] -GSA arr[3] -GSV arr[4] -RMC arr[5] -VTG arr[6] -ZDA arr[7] -ANT arr[8] -DHV arr[9] -LPS arr[10] -res1 arr[11] -res2 arr[12] -UTC arr[13] -GST
Header	#include "simcom_gps.h"

NOTE

ASR1601:
arr[0] -GGA
arr[1] -GLL
arr[2] -GSA
arr[3] -GSV
arr[4] -RMC

arr[5] -VTG
arr[6] -ZDA
arr[7] -ANT
arr[8] -DHV
arr[9] -LPS
arr[10] -res1
arr[11] -res2
arr[12] -UTC
arr[13] -GST
ASR1603:
arr[0] -GGA
arr[1] -GLL
arr[2] -GSA
arr[3] -GSV
arr[4] -RMC
arr[5] -VTG
arr[6] -ZDA
arr[7] -GST

Examples

```
#include "simcom_api.h"
#include "simcom_gps.h"
#include "simcom_common.h"

UINT8 *arr=NULL;
int I;
arr = sAPI_GnssNmeaSentenceGet();
for(i=0; i<14;i++);
{
      sAPI_Debug("bit[%d] is %d!", i, arr[i] );
}
```

25.2.13 sAPI_GpsInfoGet

This application interface is used to Get information from GPS.

sAPI_GpsInfoGet	
Prototype	SC_Gnss_Return_Code **sAPI_GpsInfoGet**(UINT8 period);
Parameters	[in] **period**: The rang is 0-255, unit is second. after set <period> will report *the GPS information every the seconds, 0 means stop reporting informations.
Return value	SC_GNSS_RETURN_CODE_OK: set done.

	SC_GNSS_RETURN_CODE_ERROR: set fail.
Header	#include "simcom_gps.h"

NOTE

GPS information is reported in **cus_urc.c**. the information is described as follows:

<**lat**>: Latitude of current position. Output format is ddmm.mmmmmm.
<**N/S**>: N/S Indicator, N=north or S=south.
<**log**>: Longitude of current position. Output format is dddmm.mmmmmm.
<**E/W**>: E/W Indicator, E=east or W=west.
<**date**>: Output format is ddmmyy.
<**UTC time**>: UTC Time. Output format is hhmmss.s.
<**alt**>: MSL Altitude. Unit is meters.
<**speed**>: Speed Over Ground. Unit is knots.
<**course**>: Course. Degrees.

Examples

```
#include "simcom_api.h"
#include "simcom_gps.h"
#include "simcom_common.h"

if(SC_GNSS_RETURN_CODE_OK != sAPI_GpsInfoGet(1));
{
    sAPI_Debug("set period error!");
}
```

25.2.14 sAPI_GnssInfoGet

This application interface is used to Get information from GNSS.

sAPI_GnssInfoGet	
Prototype	SC_Gnss_Return_Code **sAPI_GnssInfoGet**(UINT8 period);
Parameters	[in] **period**: The rang is 0-255, unit is second. after set <period> will report *the GNSS information every the seconds, 0 means stop reporting informations.
Return value	SC_GNSS_RETURN_CODE_OK: set done. SC_GNSS_RETURN_CODE_ERROR: set fail.
Header	#include "simcom_gps.h"

NOTE

GNSS information is reported in **cus_urc.c**. the information is described as follows:

<**mode**>: Fix mode, 2=2D fix, 3=3D fix
<**GPS-SVs**>: GPS satellite valid numbers, scope: 00-12
<**GLONASS-SVs**>: GLONASS satellite valid numbers, scope: 00-12. (ASR1603 does not support)
<**BEIDOU-SVs**>: BEIDOU satellite valid numbers, scope: 00-12
<**lat**>: Latitude of current position. Output format is ddmm.mmmmmm.
<**N/S**>: N/S Indicator, N=north or S=south.
<**log**>: Longitude of current position. Output format is dddmm.mmmmmm.
<**E/W**>: E/W Indicator, E=east or W=west.
<**date**>: Output format is ddmmyy.
<**UTC time**>: UTC Time. Output format is hhmmss.s.
<**alt**>: MSL Altitude. Unit is meters.
<**speed**>: Speed Over Ground. Unit is knots.
<**course**>: Course. Degrees.
<**PDOP**>: Position Dilution Of Precision.
<**HDOP**>: Horizontal Dilution Of Precision.
<**VDOP**>: Vertical Dilution Of Precision.

NOTE

ASR1603: <GLONASS-SVs> does not support

Examples

```
#include "simcom_api.h"
#include "simcom_gps.h"
#include "simcom_common.h"

if(SC_GNSS_RETURN_CODE_OK != sAPI_GnssInfoGet(1));
{
    sAPI_Debug("set period error!");
}
```

25.2.15 sAPI_SendCmd2Gnss

This application interface is used to send command directly to GNSS.

sAPI_SendCmd2Gnss	
Prototype	SC_Gnss_Return_Code **sAPI_SendCmd2Gnss**(char *string);
Parameters	[in] **string**: the command of neam format, like this: $PCAS02, 1000*2E
Return value	SC_GNSS_RETURN_CODE_OK: set done. SC_GNSS_RETURN_CODE_ERROR: set fail.
Header	#include "simcom_gps.h"

Examples

```
#include "simcom_api.h"
#include "simcom_gps.h"
#include "simcom_common.h"

char nmeaString = '$PCAS02, 1000*2E';
ret=sAPI_SendCmd2Gnss(nmeaString);
if(SC_GNSS_RETURN_CODE_ERROR == ret);
{
    sAPI_Debug("%s: send command to GNSS error!", __func__);
}
```

25.2.16 sAPI_ GnssAgpsSeviceOpen

This application interface is used to get AGPS data from the AGNSS server for assisted positioning.

sAPI_ GnssAgpsSeviceOpen	
Prototype	SC_Gnss_Return_Code **sAPI_ GnssAgpsSeviceOpen** (void);
Parameters	None
Return value	0 : AGPS ok -1: GNSS power off 101: open socket unsuccessfully. 102: get the AGNSS server unsuccessfully. 103: connect to AGNSS server unsuccessfully. 104: write information to socket unsuccessfully. 105: read AGPS data from socket unsuccessfully.
Header	#include "simcom_gps.h"

Examples

```
#include "simcom_api.h"
#include "simcom_gps.h"
#include "simcom_common.h"
INT32 pGreg = 0;

while(1);     //get network status
{
      sAPI_NetworkGetCgreg(&pGreg);
      if(1 != pGreg);
      {
          sAPI_Debug("[AGPS] NETWORK STATUS IS [%d] ", pGreg);
          sAPI_TaskSleep(10*300);
      }
      else
      {
           sAPI_Debug("[AGPS] NETWORK STATUS IS NORMAL");
           break;
       }
   }

   ret =(SC_Gnss_DEMO_Return_Code);sAPI_ GnssAgpsSeviceOpen ();
   if(SC_GNSS_RETURN_CODE_OK != ret);
   {
      sAPI_Debug("AGPS failed, error code is %d", ret);
   }
```

26 APIs for WIFI

26.1 Overview of APIs for WIFI

Interface	Description
sAPI_WifiScanStart	Start to scan WIFI network.
sAPI_WifiScanStop	Stop scanning WIFI network.
sAPI_WifiSignalShowSet	Set a flag to control the display of the signal.
sAPI_WifiSignalShowGet	Get a flag which can control the display of the signal.

26.2 Detailed Description of APIs for WIFI

26.2.1 sAPI_WifiScanStart

This application interface is to start scanning WIFI network.

sAPI_WifiScanStart	
Prototype	void **sAPI_WifiScanStart(**void);
Parameters	None.
Return value	None.
Header	#include "simcom_api.h"

NOTE

The scanning process takes some time, if the returned string contains "end of scan", WIFI scan is over.
The format of string: **[bssid] , [channel_num] , [rssi] \n**
[bssid] : The MAC address of external wireless network.
[channel_num] : The channel number of external wireless network.
[rssi] : The signal level of external wireless network.
For Examples:
BC:46:99:20:EB:A0, 1, -84

A0:BD:1D:E9:27:25, 3, -85
30:0D:9E:9F:ED:81, 6, -87
end of scan

Customer can receive the result of scanning in **cus_urc.c** after calling this interface.

Examples

```
#include"simcom_api.h"

sAPI_WifiScanStart();
```

26.2.2 sAPI_WifiScanStop

This application interface is to stop scanning WIFI network.

sAPI_WifiScanStop	
Prototype	void **sAPI_WifiScanStop(**void);
Parameters	None.
Return value	None.
Header	#include "simcom_api.h"

Examples

```
#include"simcom_api.h"

sAPI_WifiScanStop();
```

26.2.3 sAPI_WifiSignalShowSet

This application interface is to set a flag for controlling the display of signal.

sAPI_WifiSignalShowSet	
Prototype	int **sAPI_WifiSignalShowSet**(SC_WIFI_SCAN_SIGNAL_SHOW flag);
Parameters	[in] [flag] enable/disable to show signal.
Return value	0 is ok.
Header	#include "simcom_api.h"

	#include "simcom_wifi.h"

NOTE

Default value is SC_WIFI_SCAN_SIGNAL_SHOW_ENABLE.

Examples

```
#include"simcom_api.h"
#include "simcom_wifi.h"

int ret = 0;
ret = sAPI_WifiSignalShowSet(SC_WIFI_SCAN_SIGNAL_SHOW_DISABLE);
If(ret != 0);
{
    sAPI_Debug("WIFI signal show set error!");
}
```

26.2.4 sAPI_WifiSignalShowGet

This application interface is to get a flag for controlling the display of signal.

sAPI_WifiSignalShowSet	
Prototype	SC_WIFI_SCAN_SIGNAL_SHOW **sAPI_WifiSignalShowGet(**void);
Parameters	None.
Return value	SC_WIFI_SCAN_SIGNAL_SHOW_ENABLE: enable. SC_WIFI_SCAN_SIGNAL_SHOW_DISABLE: disable.
Header	#include "simcom_api.h" #include "simcom_wifi.h"

Examples

```
#include "simcom_api.h"
#include "simcom_wifi.h"

SC_WIFI_SCAN_SIGNAL_SHOW ret;
ret = sAPI_WifiSignalShowGet();
If(ret == SC_WIFI_SCAN_SIGNAL_SHOW_ENABLE);
```

```
{
    sAPI_Debug("WIFI signal will show!");
}
```

27 APIs for Bluetooth LE

27.1 Overview of APIs for Bluetooth LE

Interface	Description
sAPI_BleOpen	Open the ble device.
sAPI_BleClose	Close the ble device.
sAPI_BleSetAddress	Set the broadcast address for ble device.
sAPI_BleCreateAdvData	Create broadcast packet for ble device.
sAPI_BleSetAdvData	Set up broadcast packets for ble device.
sAPI_BleSetAdvParam	Set broadcast parameters for ble device.
sAPI_BleRegisterService	Register gatt service for ble device.
sAPI_BleUnregisterService	Destroy the registered service for ble device.
sAPI_BleRegisterEventHandle	Register event handler for ble device.
sAPI_BleEnableAdv	Turn on broadcast mode.
sAPI_BleDisableAdv	Turn off broadcast mode.
sAPI_BleDisconnect	Disconnect all connections.
sAPI_BleIndicate	Send a indicate.
sAPI_BleNotify	Send a notify.
sAPI_BleScan	Scan surrounding BLE device

27.2 Detailed Description of APIs for Bluetooth LE.

27.2.1 sAPI_BleOpen

Open the ble device..

sAPI_BleOpen	
Prototype	int **sAPI_BleOpen**(sMsgQRef msgQ, int flag);
Parameters	[in] [msgQ] The message queue that receives the result of the function. [in] [flag] Asynchronous flag. If flag=1, the function result is passed through the

	message queue. If flag=0, the function result is passed through the return value.
Return value	0: success. else: fail.
Header	#include "simcom_api.h" and #include “simcom_ble.h”

Examples

```
#include "simcom_api.h"
#include “simcom_ble.h”

Int result = sAPI_BleOpen(NULL, 0);
If (result == 0)
{
    sAPI_Debug(“open ble device success.”);
}
else
{
    sAPI_Debug(“open ble device fail. error code = %d”, result);
}
```

27.2.2 sAPI_BleClose

Close the ble device..

sAPI_BleClose	
Prototype	void **sAPI_BleClose**(void);
Parameters	None.
Return value	None
Header	#include "simcom_api.h" and #include “simcom_ble.h”

Examples

```
#include "simcom_api.h"
#include “simcom_ble.h”

sAPI_BleClose();
```

27.2.3 sAPI_BleSetAddress

Set the broadcast address for ble device.

sAPI_BleSetAddress	
Prototype	int **sAPI_BleSetAddress**(SC_BLE_ADDR_T *address);
Parameters	[in] [address] ble device address.
Return value	0: success. else: fail.
Header	#include "simcom_api.h" and #include “simcom_ble.h”

Examples

```
#include "simcom_api.h"
#include “simcom_ble.h”

SC_BLE_ADDR_T address;

address.bytes[0] = 0xdf;
address.bytes[1] = 0x45;
address.bytes[2] = 0xe6;
address.bytes[3] = 0x29;
address.bytes[4] = 0x65;
address.bytes[5] = 0xc0;

Int result = sAPI_BleSetAddress(&address);
If (result == 0)
{
    sAPI_Debug(“set address success.”);
}
else
{
    sAPI_Debug(“set address fail. error code = %d”, result);
}
```

27.2.4 sAPI_BleCreateAdvData

Create broadcast packet for ble device.

sAPI_BleCreateAdvData	
Prototype	int **sAPI_BleCreateAdvData**(int dataType, void *advData, int advDataSize, const void *data, int length);

Parameters	[in] [dataType] ble broadcast packet data type. [out] [advData] ble broadcast packet. [in] [advDataSize] ble broadcast packet size. [in] [data] ble broadcast packet data. [in] [length] ble broadcast packet data length.
Return value	0: success. else: fail.
Header	#include "simcom_api.h" and #include “simcom_ble.h”

Examples

```
#include "simcom_api.h"
#include “simcom_ble.h”

char advData[31];
int size = sAPI_BleCreateAdvData(BLE_ADV_DATA_TYPE_COMPLETE_NAME, advData, sizeof(advData), "SIMCOM BLE", strlen("SIMCOM BLE"));
```

27.2.5 sAPI_BleSetAdvData

Set broadcast packet for ble device.

sAPI_BleSetAdvData	
Prototype	int **sAPI_BleSetAdvData**(const void *advData, int length);
Parameters	[in] [advData] ble broadcast packet. [in] [length] ble broadcast packet length.
Return value	0: success. else: fail.
Header	#include "simcom_api.h" and #include “simcom_ble.h”

Examples

```
#include "simcom_api.h"
#include “simcom_ble.h”

char advData[31];
int size = sAPI_BleCreateAdvData(BLE_ADV_DATA_TYPE_COMPLETE_NAME, advData, sizeof(advData), "SIMCOM BLE", strlen("SIMCOM BLE"));
int result = sAPI_BleSetAdvData(advData, size);
if (result == 0)
{
```

```
    sAPI_Debug("set broadcast name success.");
}
else
{
    sAPI_Debug("set broadcast name fail. error code = %d", result);
}
```

27.2.6 sAPI_BleSetAdvParam

Set broadcast parameter.

sAPI_BleSetAdvParam	
Prototype	int **sAPI_BleSetAdvParam**(SC_BLE_ADV_PARAM_T *param);
Parameters	[in] [param] ble broadcast paramter.
Return value	0: success. else: fail.
Header	#include "simcom_api.h" and #include "simcom_ble.h"

Examples

```
#include "simcom_api.h"
#include "simcom_ble.h"

SC_BLE_ADV_PARAM_T param;

param.interval_min = 0x800; // 1.28s
param.interval_max = 0x800;
param.advertising_type = LE_ADV_TYPE_IND;
param.own_address_type = LE_ADDRESS_TYPE_RANDOM;

int result = sAPI_BleSetAdvParam(&param);;
if (result == 0)
{
    sAPI_Debug("set broadcast parameter success.");
}
else
{
    sAPI_Debug("set broadcast parameter fail. error code = %d", result);
}
```

27.2.7 sAPI_BleRegisterService

Register gatt service.

sAPI_BleRegisterService	
Prototype	int **sAPI_BleRegisterService**(SC_BLE_SERVICE_T *service, int length);
Parameters	[in] [service] ble broadcast service. [in] [length] ble broadcast service
Return value	0: success. else: fail.
Header	#include "simcom_api.h" and #include “simcom_ble.h”

27.2.8 sAPI_BleUnregisterService

Destroy the registered service for ble device.

sAPI_BleUnregisterService	
Prototype	int **sAPI_BleUnregisterService**(void);
Parameters	None.
Return value	0: success. else: fail.
Header	#include "simcom_api.h" and #include “simcom_ble.h”

Examples

```
#include "simcom_api.h"
#include “simcom_ble.h”

int result = sAPI_BleUnregisterService();
if (result == 0)
{
    sAPI_Debug(“destroy the registered service success.”);
}
else
{
    sAPI_Debug(“destroy the registered service fail. error code = %d”, result);
}
```

27.2.9 sAPI_BleRegisterEventHandle

Register ble event handler.

sAPI_BleRegisterEventHandle	
Prototype	int **sAPI_BleRegisterEventHandle**(SC_BLE_EVENT_HANDLE_T handle);
Parameters	[in] [handle] the ble event handler.
Return value	0: success. else: fail.
Header	#include "simcom_api.h" and #include “simcom_ble.h”

27.2.10 sAPI_BleEnableAdv

Start the ble broadcast.

sAPI_BleEnableAdv	
Prototype	int **sAPI_BleEnableAdv**(void);
Parameters	None.
Return value	0: success. else: fail.
Header	#include "simcom_api.h" and #include “simcom_ble.h”

Examples

```
#include "simcom_api.h"
#include “simcom_ble.h”

int result = sAPI_BleEnableAdv();
if (result == 0)
{
    sAPI_Debug(“start broadcast success.”);
}
else
{
    sAPI_Debug(“start broadcast fail. error code = %d”, result);
}
```

27.2.11 sAPI_BleDisableAdv

Stop the ble broadcast.

sAPI_BleDisableAdv	
Prototype	int **sAPI_BleDisableAdv**(void);
Parameters	None.
Return value	0: success. else: fail.
Header	#include "simcom_api.h" and #include “simcom_ble.h”

Examples

```
#include "simcom_api.h"
#include “simcom_ble.h”

int result = sAPI_BleDisableAdv();
if (result == 0)
{
    sAPI_Debug(“stop broadcast success.”);
}
else
{
    sAPI_Debug(“stop broadcast fail. error code = %d”, result);
}
```

27.2.12 sAPI_BleDisconnect

Disconnect all connections.

sAPI_BleDisableAdv	
Prototype	int **sAPI_BleDisconnect**(void);
Parameters	None.
Return value	0: success. else: fail.
Header	#include "simcom_api.h" and #include “simcom_ble.h”

Examples

```
#include "simcom_api.h"
#include “simcom_ble.h”

int result = sAPI_BleDisconnect();
if (result == 0)
```

```
{
    sAPI_Debug("close all connections success.");
}
else
{
    sAPI_Debug("close all connections fail. error code = %d", result);
}
```

27.2.13 sAPI_BleIndicate

Send a indicate.

sAPI_BleDisableAdv	
Prototype	int **sAPI_BleIndicate**(unsigned short att_handle, const void *data, int size);
Parameters	[in] [att_handle] Characteristic handle. [in] [data] The data to waiting send. [in] [size] The data's length.
Return value	0: success. else: fail.
Header	#include "simcom_api.h" and #include "simcom_ble.h"

Examples

```
#include "simcom_api.h"
#include "simcom_ble.h"

int result = sAPI_BleIndicate(handle, "first data", strlen("first data"));
if (result == 0)
{
    sAPI_Debug("send a indicate success.");
}
else
{
    sAPI_Debug("send a indicate fail. error code = %d", result);
}
```

27.2.14 sAPI_BleNotify

Send a notify.

sAPI_BleNotify	
Prototype	int **sAPI_BleNotify**(unsigned short att_handle, const void *data, int size);
Parameters	[in] [att_handle] Characteristic handle. [in] [data] The data to waiting send. [in] [size] The data's length.
Return value	0: success. else: fail.
Header	#include "simcom_api.h" and #include "simcom_ble.h"

Examples

```
#include "simcom_api.h"
#include "simcom_ble.h"

int result = sAPI_BleNotify(handle, "first data", strlen("first data"));
if (result == 0)
{
    sAPI_Debug("send a notify success.");
}
else
{
    sAPI_Debug("send a notify fail. error code = %d", result);
}
```

27.2.15 sAPI_BleScan

Scan surrounding BLE device.

sAPI_BleScan	
Prototype	int **sAPI_BleScan**(unsigned char type, unsigned short interval, unsigned short window, unsigned char own_address_type, void (*handler)(SC_BLE_SCAN_EVENT_T *scan));
Parameters	[in] [type] scan type. example: LE_ACTIVE_SCAN, LE_PASSIVE_SCAN. [in] [interval] scan interval. range: 0x0004-0x4000. [in] [window] scan window. range: 0x0004-0x4000. [in] [own_address_type] address type. example: LE_ADDRESS_TYPE_PUBLIC, LE_ADDRESS_TYPE_RANDOM. [in] [handler] the callback that get the scan result.
Return value	0: success. else: fail.
Header	#include "simcom_api.h" and #include "simcom_ble.h"

Examples

```
#include "simcom_api.h"
#include “simcom_ble.h”

void ble_handle_scan(SC_BLE_SCAN_EVENT_T *scan)
{
    sAPI_Debug(“rssi = %d”, scan->rssi);
}

int result = sAPI_BleScan(LE_ACTIVE_SCAN,0x0800,0x0400,LE_ADDRESS_TYPE_RANDOM,ble_handle_scan);
if (result == 0)
{
    sAPI_Debug(“scan success.”);
}
else
{
    sAPI_Debug(“scan fail. error code = %d”, result);
}
```

27.3 Result Codes

<err>	Meaning
0	Success.
1	Not know error.
2	Status alert.
3	Parameter error.
4	Open ble device error.

28 APIs for Debug

28.1 Overview of APIs for Debug

Interface	Description
sAPI_Debug	Log output

28.2 Detailed Description of APIs for Debug

28.2.1 sAPI_Debug

This application interface is to log output

sAPI_Debug	
Prototype	void **sAPI_Debug**(const char *format, ...);
Parameters	[in] [**format**] The format string is a character string
Return value	none
Header	#include "simcom_debug.h"

NOTE

For more details about the access to view the logs, please refer to Chapter 3 of the document A7600_Series_Open_SDK_Debug_and_Download_Application_Note.

Examples

```
#include "simcom_ debug.h"

void sAPI_DebugMain();
{
    char chararray[] = {"hello sAPI_Debug"};
    int a = 100;
    //Print const string
    sAPI_Debug("print const string");

    //print variable in char format
    sAPI_Debug("print variable in char format %c", chararray[0] );

    //print char array
    sAPI_Debug("print variable in char array %s", chararray);
```

```
    //print integer type
    sAPI_Debug("print variable integer type %d", a);
}
```

29 APIs for Version Information and Others

29.1 Overview of APIs for Version Information and Others

Interface	Description
sAPI_SysGetImei	Get Imei
sAPI_SysVersion	Gets simcomVersion Info
sAPI_SysGetCusVersion	Gets customer Version Info
sAPI_SysGetRFVersion	Gets RFVersion Info

29.2 Detailed Description of APIs for version information and Others

29.2.1 sAPI_SysGetImei

This application interface is to get the Imei.

sAPI_SysVersion	
Prototype	unsigned int **sAPI_SysGetImei**(char *ImeiValue);
Parameters	[out] **ImeiValue:** The value of the IMEI
Return value	SC_SYS_SUCCESS: success to get Imei SC_SYS_FAIL: fail to get Imei
Header	#include "simcom_system.h"

Examples

```
#include "simcom_ other.h"

unsigned int version_Test(void);
{
```

```
    int ret = CIRC_FAIL;
    char imei_value[16] ;
    ret = sAPI_SysGetImei(imei_value);
    return ret;
}
```

29.2.2 sAPI_SysGetVersion

This application interface is to get the simcomversion info.

sAPI_SysVersion	
Prototype	void **sAPI_SysGetVersion**(simcomversion *simcominfo);
Parameters	[out] **simcominfo:** the version info
Return value	-
Header	#include "simcom_system.h"

Examples

```
#include "simcom_ version.h"

void version_Test(void);
{
    simcomversion simcominfo;

    sAPI_SysGetVersion(&simcominfo);
}
```

29.2.3 sAPI_SysGetCusVersion

This application interface is to get the cusversion info.

sAPI_SysVersion	
Prototype	void **sAPI_SysGetCusVersion**(CUSVersion *CUSinfo);
Parameters	[out] **CUSinfo:** the customer version info
Return value	-
Header	#include "simcom_system.h"

Examples

```
#include "simcom_ version.h"

void version_Test(void);
{
    CUSversion CUSinfo;

    sAPI_SysGetCusVersion(&CUSinfo);
}
```

29.2.4 sAPI_SysGetRFVersion

This application interface is to get the RFversion info.

sAPI_SysVersion	
Prototype	void **sAPI_SysGetRFVersion**(RFVersion *RFinfo);
Parameters	[out] **RFinfo:** the RF version info
Return value	-
Header	#include "simcom_system.h"

Examples

```
#include "simcom_ version.h"

void version_Test(void);
{
    RFversion RFinfo;

    sAPI_SysGetRFVersion(&RFinfo);
}
```

30 URC Processor

30.1 Overview of URC Processor

Interface	Description
sTask_UrcProcesser	The task which used to process the URC.
sAPI_UrcRefRegister	Register the message queue which used to receive the URC.
sAPI_UrcRefRelease	Release the message queue which used to receive the URC in the past.

URC Type	Description
SC_URC_SMSDWON	SMS done
SC_URC_SMSFULL	SMS memery full
SC_URC_STATUS_REPORT	SMS status report(similar to +CDS);
SC_URC_NEW_MSG_IND	SMS new msg received, report msg index
SC_URC_FLASH_MSG	SMS new msg directly report(similar to +CMT);
SC_URC_PBDOWN	PB done
SC_URC_NETACTED	NETWORK ACT
SC_URC_NETDIS	NETWORK DISCONNECT
SC_URC_PDP_ACTIVE	PDP ACT
SC_URC_PDP_DEACT	PDP DEACT

30.2 Detailed Description of interface for URC processer

30.2.1 sTask_UrcProcesser

This task will work automatically, and all necessary URC will be processed at there.

sTask_UrcProcesser	
Prototype	void **sTask_UrcProcesser(**void *ptr);

Parameters	[in] ptr: not use.
Return value	void.

30.2.2 sAPI_UrcRefRegister

Register the message queue which used to receive the URC.

sAPI_UrcRefRegister	
Prototype	int **sAPI_UrcRefRegister**(sMsgQRef ref, UINT32 mask);
Parameters	[in] ref: The message queue used to receive the URC. [in] mask: The filter to indicate which module of URC you want.
Return value	0 is OK, otherwise return the error code.
Header	sim_common.h

30.2.3 sAPI_UrcRefRelease

Release the message queue which used to receive the URC in the past.

sAPI_UrcRefRelease	
Prototype	int **sAPI_UrcRefRelease**(sMsgQRef ref);
Parameters	[in] ref: The message queue used to receive the URC in the pase.
Return value	0 is OK, otherwise return the error code.
Header	sim_common.h

Examples

```
sMsgQRef gUrcMsgQueue = NULL;

void sTask_UrcProcesser(void *ptr);
{
    int ret = 0;
    SC_STATUS osStatus = OS_FAIL;
    SIM_MSG_T msg = {0};

    osStatus = sAPI_MsgQCreate(&gUrcMsgQueue);
    if(SC_SUCCESS != osStatus);
    {
        return;
```

```
    }

    ret = sAPI_UrcRefRegister(gUrcMsgQueue, SC_MODULE_PS | SC_MODULE_CM);

    if(0 != ret);
    {
        sAPI_MsgQDelete(gUrcMsgQueue);
        return;
    }

    while(1);
    {
        osStatus = sAPI_MsgQRecv(gUrcMsgQueue, &msg, SC_SUSPEND);
        if(SC_SUCCESS != osStatus);
        {
            continue;
        }

        if((SRV_URC != msg.msg_id); ||(NULL == msg.arg3););
        {
            sAPI_FREE(msg.arg3);
            continue;
        }

        switch(msg.arg1);
        {
            default:
            break;

            case SC_MODULE_PS:
            break;

            case SC_MODULE_CM:
            break;
        }

        sAPI_FREE(msg.arg3);
    }

    sAPI_UrcRefRelease(gUrcMsgQueue);
    sAPI_MsgQDelete(gUrcMsgQueue);
}
```

31 APIs for File System of ASR1603 & ASR1803 Platforms

31.1 Overview of APIs for File System

API	Description
sAPI_fopen	Flie open
sAPI_fclose	File close
sAPI_fwrite	File write
sAPI_fread	File read
sAPI_fseek	File seek
sAPI_ftell	File tell
sAPI_frewind	Move file pointer to the beginning of the file
sAPI_fsize	Gets the size of an open file
sAPI_fsync	File synchronization
sAPI_mkdir	Create directory
sAPI_opendir	Open directory
sAPI_closedir	Close directory
sAPI_readdir	Read directory
sAPI_seekdir	Sets the read location for directory
sAPI_telldir	The current location of the directory stream
sAPI_remove	Delete a file
sAPI_rename	File rename
sAPI_access	Check whether the file exists
sAPI_stat	Get infomation of file or folder
sAPI_GetSize	Get the total size of file system
sAPI_GetFreeSize	Get free size of file system
sAPI_GetUsedSize	Get used size of file system

31.2 Detailed Description of APIs for File System

The file system is used to store files in a hierarchical(tree); structure, and there are some definitions and conventions to use the APIs.
Local storage space is mapped to”C:”,”D” for SD card.(1603 platform not support SD card)

NOTE

General rules for naming(both directories and files):

a): The length of actual fully qualified names of files(C:/):
for 1803 platform, can not exceed 112;
for 1603 platform ,can not exceed 255.

b): The length of actual fully qualified names of directories and files(D:/); can not exceed 250.

c): Directory and file names can not include the following characters:

\ : * ? " < > | , ;

d): Between directory name and file/directory name, use character "/" as list separator, so it can not appear in directory name or file name.

e) File names on “C:/” drive cannot begin with '.' or ‘ ’ .

f) **sAPI_frewind(), sAPI_fsize(),sAPI_fsync(), sAPI_seekdir(),sAPI_telldir() not support 1803 platform.**

If the last character of names is period "."; the flash(C:/); will auto delete this character; the SD card can support this character, but the compatibility is not good.

31.2.1 sAPI_fopen

This API is used to open or create a file and associate it with a stream, Support "C:", "D:".

sAPI_fopen	
Prototype	SCFILE ***sAPI_fopen**(const char *fname, const char *mode);
Parameters	[in] **fname**: contains the full file name of the path. [in] **mode**: const character string for type specification, r/w/a/b; rb:read only wb/ab:write only rb+/wb+/ab+:read and write Note: "rb" and "wb" modes are usually used
Return value	A file pointer to the stream. Returns NULL on failure.
Header	#include "simcom_file_system.h"

31.2.2 sAPI_fclose

This API is used to close a file stream. Support "C:", "D:"

sAPI_fclose	
Prototype	int **sAPI_fclose**(SCFILE *fp);
Parameters	[in] **fp**: stream index for the file to be closed.
Return value	0 :Process successfully executed other: Process failly executed
Header	#include "simcom_file_system.h"

31.2.3 sAPI_fwrite

This API is used to write data to a file. Support "C:", "D:"

sAPI_fwrite	
Prototype	int **sAPI_fwrite**(const void *buffer, size_t size, size_t num, SCFILE *fp);
Parameters	[in] **buffer**: pointer to the data to be written to the file [in] **size**: the size of each element to be read, in bytes. [in] **num**: the number of elements, each of which is size bytes [in] **fp**: file pointer to the stream for the file to be read.
Return value	The data length of actually write, which is an integer data type
Header	#include "simcom_file_system.h"

31.2.4 sAPI_fread

Read some data to a buffer with specific length, Support "C:", "D:"

sAPI_fread	
Prototype	int **sAPI_fread**(void *buffer, size_t size, size_t num, SCFILE *fp);
Parameters	[out] **buffer**: pointer to buffer to place data read [in] **size**: the size of each element to be read, in bytes. [in] **num**: the number of elements, each of which is size bytes [in] **fp**: file pointer to the stream for the file to be read.
Return value	The total number of elements successfully read, which is an integer data type
Header	#include "simcom_file_system.h"

31.2.5 sAPI_fseek

Sets the file position indicator of the file specified by stream. The new position, measured in bytes from the beginning of the file is obtained by adding offset to the position specified by wherefrom. Support "C:", "D:".

sAPI_fseek	
Prototype	int sAPI_fseek(SCFILE *fp, long offset, int whence);
Parameters	[in] **fp**: stream the file handle [in] **offset**: move size from the start address [in] **whence**: argument can be FS_SEEK_BEGIN 0 FS_SEEK_CURREN 1 FS_SEEK_END 2
Return value	0:Process successfully executed other: Process failly executed
Header	#include "simcom_file_system.h"

31.2.6 sAPI_ftell

The function get the file position. Support "C:", "D:"

sAPI_ftell	
Prototype	long **sAPI_ftell**(SCFILE *fp);
Parameters	[in] **fp:** stream the file handle
Return value	the current file position
Header	#include "simcom_file_system.h"

31.2.7 sAPI_frewind

Move file pointer to the beginning of the file. Support "C:", "D:".

sAPI_frewind	
Prototype	void **sAPI_frewind**(SCFILE *fp);
Parameters	[in] **fp**: stream the file handle
Return value	No return value
Header	#include "simcom_file_system.h"

31.2.8 sAPI_fsize

This API is used to get the size of an open file. Support "C:".

sAPI_fsize	
Prototype	int **sAPI_fsize**(SCFILE *fp);
Parameters	[in] **fp**: stream the file handle
Return value	the size of a file
Header	#include "simcom_file_system.h"

31.2.9 sAPI_fsync

Synchronize a file. Support "C:".

sAPI_fsync	
Prototype	int sAPI_fsync(SCFILE *fp);
Parameters	[in] **handle**: stream the file handle
Return value	0 :Process successfully executed other: Process failly executed
Header	#include "simcom_file_system.h"

31.2.10 sAPI_mkdir

This command is used to create a new directory. Support "C:", "D:"

sAPI_mkdir	
Prototype	int **sAPI_mkdir**(const char *path, unsigned int mode);
Parameters	[in] **path**: folder name with path [in] **mode**: create mode.(no practical implications, just set it to 0)
Return value	0 :Process successfully executed other: Process failly executed
Header	#include "simcom_file_system.h"

31.2.11 sAPI_opendir

This command is used to open a directory. Support "C:", "D:"

sAPI_opendir	
Prototype	SCDIR ***sAPI_opendir**(const char *path);
Parameters	[in] **path**: directory name with path
Return value	A directory pointer to the stream. Returns NULL on failure.
Header	#include "simcom_file_system.h"

31.2.12 sAPI_closedir

This command is used to close a directory. Support "C:", "D:"

sAPI_closedir	
Prototype	int **sAPI_closedir**(SCDIR *dirp);
Parameters	[in] **dirp**: directory stream
Return value	0 :Process successfully executed other: Process failly executed
Header	#include "simcom_file_system.h"

31.2.13 sAPI_readdir

This command is used to read a directory. Support "C:", "D:".

sAPI_readdir	
Prototype	struct dirent ***sAPI_readdir**(SCDIR *dirp);
Parameters	[in] **handle**: directory stream [in] **info**: the structure that contains file information(filename,type,filesize)
Return value	A pointer to the structure that contains file information(filename,type,filesize). Returns NULL on failure.
Header	#include "simcom_file_system.h"

31.2.14 sAPI_seekdir

Sets the read location for the next call to sAPI_readdir(). Support “C:”.

sAPI_seekdir

Prototype	int **sAPI_seekdir**(SCDIR *dirp, unsigned long offset);
Parameters	[in] **dirp**: directory stream [in] **offset**: move size from the directory beginning
Return value	0:success other:fail
Header	#include "simcom_file_system.h"

31.2.15 sAPI_telldir

The current location of the directory stream. Support “C:”.

sAPI_telldir	
Prototype	int sAPI_telldir(SCDIR *dirp);
Parameters	[in] **dirp**: directory stream [in] **offset**: move size from the directory beginning
Return value	current location
Header	#include "simcom_file_system.h"

31.2.16 sAPI_remove

Remove a file or directory. Support "C:", "D:".

sAPI_remove	
Prototype	int **sAPI_remove**(const char *fname);
Parameters	[in] **fPath**: file name
Return value	0 : Process successfully executed other: Process failly executed
Header	#include "simcom_file_system.h"

31.2.17 sAPI_rename

This API is used to rename a file. Support "C:".

sAPI_rename	
Prototype	int **sAPI_rename**(const char *oldpath, const char *newpath);

Parameters	[in] **oldPath**: old file name [in] **newPath**: new file name
Return value	0: Process successfully executed other: Process failly executed
Header	#include "simcom_file_system.h"

31.2.18 sAPI_access

This API is used to check whether the file exists. Support "C:", "D:".

sAPI_access	
Prototype	int **sAPI_access**(const char *path, int mode);
Parameters	[in] **path**: file name [in] **mode**: mode.(no practical implications, just set it to 0)
Return value	0 : file exists other: file doesn't exist
Header	#include "simcom_file_system.h"

31.2.19 sAPI_stat

This API is used to get the information structure of an existing file/directory. Support "C:", "D:".

sAPI_stat	
Prototype	int **sAPI_stat**(const char *path, struct dirent *info);
Parameters	[in] **path**: file name [out] **info**: the structure that contains file information(filename,type,filesize)
Return value	0 : file exists other: file doesn't exist
Header	#include "simcom_file_system.h"

31.2.20 sAPI_GetSize

This API is used to Get file system space size. Support "C:", "D:".

sAPI_GetSize	
Prototype	long long **sAPI_GetSize**(char *disc);
Parameters	[in] **disc**: the disk (such as "C");

Return value	disk total size
Header	#include "simcom_file_system.h"

31.2.21 sAPI_GetFreeSize

This API is used to Get file system free space size. Support "C:", "D:".

sAPI_GetFreeSize	
Prototype	long long **sAPI_GetFreeSize**(char *disc);
Parameters	[in] **disc**: the disk (such as "C");
Return value	disk free size
Header	#include "simcom_file_system.h"

31.2.22 sAPI_GetUsedSize

This API is used to Get file system used space size. Support "C:", "D:".

sAPI_GetUsedSize	
Prototype	long long **sAPI_FsGetUsedSize**(char *disc);
Parameters	[in] **disc**: the disk (such as "C");
Return value	disk used size
Header	#include "simcom_file_system.h"

31.3 Result codes

31.3.1 Description of <err>s

<err>	Description
0	Operation succeeded
Other values	Operation failed

31.4 Demo for File System

```
void FsDemo(void)
{
    SCFILE *file_hdl = NULL;
    SCDIR *dir_hdl = NULL;
    char file_name[MAX_VALID_USER_NAME_LENGTH] = {0};
    char dir_name[MAX_VALID_USER_NAME_LENGTH]  = {0};
    //char copy_name[MAX_VALID_USER_NAME_LENGTH]  = {0};

    char operation_path[MAX_VALID_USER_NAME_LENGTH]={0};
    char  pTemBuffer[MAX_OUTPUT_LEN*5];
    UINT32 ret = 0;
    char buff[MAX_OUTPUT_LEN] = {0};

    unsigned long buff_data_len = 0;
    UINT32 actul_write_len = 0;
    UINT32 actul_read_len = 0;
    INT64 total_size = 0;
    //int total_size = 0;
    INT64 free_size = 0;
    INT64 used_size = 0;
    struct dirent *info_dir = NULL;
    struct dirent info_file = {0};

    SIM_MSG_T optionMsg ={0,0,0,NULL};


    UINT32 opt = 0;
    char *note = "\r\nPlease select an option to test from the items listed below.\r\n";
    char *options_list[] = {
        "1. Write file",
        "2. File seek",
        "3. Read file",
        "4. Delete file",
        "5. Get disk size",
        "6. Make directory",
        "7. Remove directory",
        "8. List file",
        "99. Back",
    };

    memset(file_name,0,MAX_VALID_USER_NAME_LENGTH);
    //memcpy(file_name, "c:/test_file.txt", strlen("c:/test_file.txt"));
```

```
    while(1)
    {
        PrintfResp(note);
        PrintfOptionMenu(options_list,sizeof(options_list)/sizeof(options_list[0]));
        sAPI_MsgQRecv(simcomUI_msgq,&optionMsg,SC_SUSPEND);
        if(SRV_UART != optionMsg.msg_id)
        {
            sAPI_Debug("%s,msg_id is error!!",__func__);
            break;
        }

        sAPI_Debug("arg3 = [%s]",optionMsg.arg3);
        opt = atoi(optionMsg.arg3);
        sAPI_Debug("opt %d",opt);
        sAPI_Free(optionMsg.arg3);

        switch(opt)
        {
            case SC_FS_DEMO_WRITEFILE:
            {
                memset(file_name,0,MAX_VALID_USER_NAME_LENGTH);
                PrintfResp("\r\nPlease input path:\r\n");
                optionMsg = GetParamFromUart();
                strcpy(file_name,optionMsg.arg3);
                sAPI_Free(optionMsg.arg3);

                sAPI_Debug("ready write file[%s]", file_name);
                sAPI_Debug("sAPI_GetFreeSize address, %p,%x",
sAPI_GetFreeSize,sAPI_GetFreeSize);
                ret = sAPI_GetFreeSize(cur_path_in);
                sAPI_Debug("free_size= %lld", ret);

                file_hdl = sAPI_fopen(file_name, "wb");
                sAPI_Debug("sAPI_fopen address, %p,%x", sAPI_fopen,sAPI_fopen);
                if(file_hdl == NULL)
                {
                    sAPI_Debug("sAPI_FsFileOpen err");
                    sprintf(pTemBuffer, "\r\nFILE SYSTEM Write file failed because open file
failed\r\n");
                    goto err;
                }
                memcpy(buff, "12345678", 8);
                buff_data_len = 8;
                actul_write_len = sAPI_fwrite(buff, buff_data_len,1,file_hdl);
                if(actul_write_len != buff_data_len)
```

```
                    {
                        sAPI_Debug("sAPI_fwrite err write length: %d\r\n", actul_write_len);
                        sprintf(pTemBuffer, "\r\nFILE SYSTEM Write file failed\r\n");
                        goto err;
                    }

                    ret = sAPI_fclose(file_hdl);
                    if(ret != 0)
                    {
                        sAPI_Debug("sAPI_fclose err");
                        sprintf(pTemBuffer, "\r\nFILE SYSTEM Write file failed because close file failed\r\nn");
                        goto err;
                    }else{
                        file_hdl = NULL;
                    }
                    sprintf(pTemBuffer, "\r\nFILE SYSTEM Write file successful\r\n\r\nFilename is %s \r\n",file_name);

                    ret = sAPI_stat(file_name,&info_file);
                    if(ret != 0)
                    {
                        sAPI_Debug("sAPI_FsStat make err,ERROR code: %d", ret);
                        strcat(pTemBuffer, "\r\nFILE SYSTEM stat fail\r\n");
                        goto err;
                    }

                    sAPI_Debug("sAPI_FsStat                                                    succ [%s]-[%d]-[%d]",info_file.name,info_file.size,info_file.type);
                    strcat(pTemBuffer, "\r\nFILE SYSTEM stat SUCC\r\n");

                    break;
                }
                case SC_FS_DEMO_FILESEEK:
                {
                    buff_data_len = 0;

                    file_hdl = sAPI_fopen(file_name, "rb");
                    if(file_hdl == NULL)
                    {
                        sAPI_Debug("sAPI_fopen err");
                        sprintf(pTemBuffer, "\r\nFILE SYSTEM File seek failed because open file failed\r\n");
                        goto err;
                    }
```

```
                ret = sAPI_fseek(file_hdl, 3, FS_SEEK_BEGIN);
                if(ret != 0)
                {
                        sAPI_Debug("sAPI_FsFileSeek fail! ERROR code: %d",ret);
                        sprintf(pTemBuffer, "\r\nFILE SYSTEM File seek failed!\r\n");
                        goto err;
                }
                ret = sAPI_ftell(file_hdl);
                if(ret < 0)
                {
                        sAPI_Debug("sAPI_ftell fail! ERROR code: %d",ret);
                        sprintf(pTemBuffer, "\r\nFILE SYSTEM File tell failed!\r\n");
                        goto err;
                }
                sAPI_Debug("sAPI_FsFileTell file position:    %d",ret);

                buff_data_len = sAPI_fsize(file_hdl);
                sAPI_Debug("sAPI_fsize buff_data_len: %d",    buff_data_len);

                memset(buff, 0, BUFF_LEN);
                actul_read_len = sAPI_fread(buff, buff_data_len,1,file_hdl);
                if(actul_read_len <= 0)
                {
                    sAPI_Debug("sAPI_fread err,data: %s, len: %d", buff, actul_read_len);
                    sprintf(pTemBuffer, "\r\nFILE SYSTEM File seek failed\r\n");
                    goto err;
                }
                sAPI_Debug("read data: %s, len: %d", buff, actul_read_len);

                ret = sAPI_fclose(file_hdl);
                if(ret != 0)
                {
                    sAPI_Debug("sAPI_fclose err");
                    sprintf(pTemBuffer, "\r\nFILE SYSTEM File seek failed\r\n");
                    goto err;
                }else{
                    file_hdl = NULL;
                }
                sprintf(pTemBuffer,  "\r\nFILE  SYSTEM  File  seek  successful\r\n\r\nFilename
is %s,data read after offset is %s\r\n",file_name,buff);

                break;
            }
            case SC_FS_DEMO_READFILE:
            {
                memset(file_name,0,MAX_VALID_USER_NAME_LENGTH);
```

```
                PrintfResp("\r\nPlease input path:\r\n");
                optionMsg = GetParamFromUart();
                strcpy(file_name,optionMsg.arg3);
                sAPI_Free(optionMsg.arg3);

                buff_data_len = 0;
                file_hdl = sAPI_fopen(file_name, "rb");
                if(file_hdl == NULL)
                {
                    sAPI_Debug("sAPI_fopen err");
                    sprintf(pTemBuffer, "\r\nFILE SYSTEM Read file failed\r\n");
                    goto err;
                }

                buff_data_len = sAPI_fsize(file_hdl);
                sAPI_Debug("sAPI_fsize buff_data_len: %d",    buff_data_len);

                memset(buff, 0, BUFF_LEN);
                actul_read_len = sAPI_fread(buff,buff_data_len, 1, file_hdl);
                if(actul_read_len <= 0)
                {
                    sAPI_Debug("sAPI_FsFileRead err,data: %s, len: %d", buff, actul_read_len);
                    sprintf(pTemBuffer, "\r\nFILE SYSTEM Read file failed\r\n");
                    goto err;
                }

                sAPI_Debug("read data: %s, len: %d", buff, actul_read_len);

                ret = sAPI_fclose(file_hdl);
                if(ret != 0)
                {
                    sAPI_Debug("sAPI_fclose err");
                    sprintf(pTemBuffer, "\r\nFILE SYSTEM Read file failed\r\n");
                    goto err;
                }else{
                    file_hdl = NULL;
                }

                sprintf(pTemBuffer, "\r\nFILE SYSTEM Read file successful\r\n\r\nFilename is %s,read data is %s\r\n",file_name,buff);


            #if 0
                /*test rename api*/
                memset(copy_name,0,MAX_VALID_USER_NAME_LENGTH);
                memcpy(copy_name, "c:/copy.txt", strlen("c:/copy.txt"));
```

```
                memset(pTemBuffer,0,sizeof(pTemBuffer));

                ret = sAPI_rename(file_name,copy_name);
                if(ret != 0){
                    sAPI_Debug("[FS] sAPI_rename err,ERROR code: %d", ret);
                    sprintf(pTemBuffer, "\r\nFILE SYSTEM Rename file failed\r\n");
                    goto err;
                }
                sAPI_Debug("[FS] sAPI_rename succ");
            #endif

                break;
            }
            case SC_FS_DEMO_DELETEFILE:
            {
                ret = sAPI_remove(file_name);
                if(ret != 0)
                {
                    sAPI_Debug("sAPI_FsFileDelete err,ERROR code: %d", ret);
                    sprintf(pTemBuffer, "\r\nFILE SYSTEM Delete file failed\r\n");
                    goto err;
                }

                sprintf(pTemBuffer, "\r\nFILE SYSTEM Delete file successful\r\n");
                break;
            }
            case SC_FS_DEMO_GETDISKSIZE:
            {
                total_size = sAPI_GetSize(cur_path_in);
                free_size = sAPI_GetFreeSize(cur_path_in);
                used_size = sAPI_GetUsedSize(cur_path_in);

                sAPI_Debug("total_size= %lld, free_size= %lld,used_size = %lld", total_size,
free_size,used_size); //sd card size may out of int type num range
                sprintf(pTemBuffer, "\r\nFILE SYSTEM Total_size= %lld, Free_size= %lld,used_size
= %lld\r\n",total_size, free_size,used_size);
                break;
            }
            case SC_FS_DEMO_MAKEDIR:
            {
                memset(dir_name,0,MAX_VALID_USER_NAME_LENGTH);
                //memcpy(dir_name, "c:/dir1", strlen("c:/dir1"));

                memset(dir_name,0,MAX_VALID_USER_NAME_LENGTH);
                PrintfResp("\r\nPlease input path:\r\n");
                optionMsg = GetParamFromUart();
```

```
                strcpy(dir_name,optionMsg.arg3);
                sAPI_Free(optionMsg.arg3);

                sAPI_Debug("sAPI_FsDirMake:dir[%s]",dir_name);

                ret = sAPI_mkdir(dir_name,0);
                if(ret != 0)
                {
                    sAPI_Debug("sAPI_FsDirProcessor make err,ERROR code: %d", ret);
                    sprintf(pTemBuffer, "\r\nFILE SYSTEM Make dir failed\r\n");
                    goto err;
                }

                sprintf(pTemBuffer, "\r\nFILE SYSTEM Make dir SUCC\r\n");

                ret = sAPI_stat(dir_name,&info_file);
                if(ret != 0)
                {
                    sAPI_Debug("sAPI_FsStat make err,ERROR code: %d", ret);
                    strcat(pTemBuffer, "\r\nFILE SYSTEM stat fail\r\n");
                    goto err;
                }
                sAPI_Debug("sAPI_FsStat                                                    succ
[%s]-[%ld]-[%d]",info_file.name,info_file.size,info_file.type);
                strcat(pTemBuffer, "\r\nFILE SYSTEM stat SUCC\r\n");

                break;
            }
            case SC_FS_DEMO_REMOVEDIR:
            {
                memset(dir_name,0,MAX_VALID_USER_NAME_LENGTH);
                memcpy(dir_name, "c:/dir1", strlen("c:/dir1"));

                ret = sAPI_remove(dir_name);
                if(ret != 0)
                {
                    sAPI_Debug("sAPI_FsDirRemove del err,ERROR code: %d", ret);
                    sprintf(pTemBuffer, "\r\nFILE SYSTEM REMOVE dir failed\r\n");
                    goto err;
                }
                sprintf(pTemBuffer, "\r\nFILE SYSTEM REMOVE dir SUCC\r\n");

                break;
            }
            case SC_FS_DEMO_OPENDIR:
            {
```

```
        memset(operation_path,0,MAX_VALID_USER_NAME_LENGTH);

        PrintfResp("\r\nPlease input path:\r\n");
        optionMsg = GetParamFromUart();
        strcpy(operation_path,optionMsg.arg3);
        sAPI_Free(optionMsg.arg3);

        sAPI_Debug("sAPI_FsDirProcessor:operation_path[%s]",operation_path);
        dir_hdl = sAPI_opendir(operation_path);
        if(dir_hdl == NULL)
        {
            sAPI_Debug("sAPI_FsDirOpen OPEN err");
            sprintf(pTemBuffer, "\r\nFILE SYSTEM Open dir failed\r\n");
            goto err;
        }
        sAPI_Debug("\r\nFILE SYSTEM Open dir SUCC\r\n");

        while((info_dir = sAPI_readdir(dir_hdl)) != NULL)
        {
            sprintf(pTemBuffer, "\r\n[name]-[filesize]-[type]\r\n");
            sprintf(buff, "[%s]-[%ld]-[%d]\r\n",info_dir->name,info_dir->size,info_dir->type);
            strcat(pTemBuffer,buff);
            sAPI_UartWrite(SC_UART,pTemBuffer,strlen(pTemBuffer));
            memset(pTemBuffer,0,sizeof(pTemBuffer));
        }

        ret = sAPI_closedir(dir_hdl);
        if(ret != 0)
        {
            sAPI_Debug("sAPI_FsDirClose make err,ERROR code: %d", ret);
            sprintf(pTemBuffer, "\r\nFILE SYSTEM Close dir Failed\r\n");
            goto err;
        }
        sAPI_Debug("sAPI_FsDirClose succ");
        sprintf(pTemBuffer, "\r\nFILE SYSTEM List File END\r\n");

        break;
    }
    case SC_FS_DEMO_MAX:
    {
        sAPI_Debug("Return to the previous menu!");
        PrintfResp("\r\nReturn to the previous menu!\r\n");
        memset(pTemBuffer,0,sizeof(pTemBuffer));
        return;
    }
    default:
```

```
                {
                    memset(pTemBuffer,0,sizeof(pTemBuffer));
                    break;
                }
            }

err:
            if(file_hdl != NULL)
            {
                ret = sAPI_fclose(file_hdl);
                if(ret != 0)
                {
                    sAPI_Debug("sAPI_FsFileClose err");
                    sprintf(pTemBuffer, "\r\nFILE SYSTEM close file failed\r\n");
                }else{
                    file_hdl = NULL;
                }
            }

            if(strlen(pTemBuffer) > 0)
            {
                sAPI_Debug("sAPI_Fs test_result [%s]", pTemBuffer);
                sAPI_UartWrite(SC_UART,pTemBuffer,strlen(pTemBuffer));
            }
        }
}
```

32 APIs for RTC ASR1601 Platforms

32.1 Overview of APIs for RTC

API	Description
sAPI_SetRealTimeClock	Set time
sAPI_GetRealTimeClock	Get time
sAPI_RtcSetAlarm	Set alarm time
sAPI_RtcGetAlarm	Get alarm time
sAPI_RtcEnableAlarm	Turn on or off alarm
sAPI_RtcRegisterCB	Register callback function for alarm

32.2 Detailed Description of APIs for RTC

The RTC is used to recording time also wake up modem through alarm.

NOTE

Wake up mode through alarm,you must set alarm befor poweroff.

32.2.1 sAPI_SetRealTimeClock

This API is used to set RTC time.You can use this api change RTC time.

sAPI_fopen	
Prototype	INT16 sAPI_RtcSetRealTime(t_rtc *rtcSetTime);

Parameters	[in] **rtcSetTime**: this time will instead of RTC time.
	Note:
	typedef struct rtc_time {
	int tm_sec; //seconds [0,59]
	int tm_min; //minutes [0,59]
	int tm_hour; //hour [0,23]
	int tm_mday; //day of month [1,31]
	int tm_mon; //month of year [1,12]
	int tm_year; // since 1970
	int tm_wday; // sunday = 0
	}t_rtc;

Return value	Returns value 0 on success, -1 on failure.
Header	#include "simcom_api.h"

32.2.2 sAPI_RtcGetRealTime

This API is used to get RTC time.

sAPI_fopen	
Prototype	INT16 sAPI_RtcGetRealTime(t_rtc *RtcTime);
Parameters	[in] **RtcTime**: out put vlaue.
Return value	Returns value 0 on success, -1 on failure.
Header	#include "simcom_api.h"

32.2.3 sAPI_RtcSetAlarm

This API is used to set alarm time.

sAPI_fopen	
Prototype	void sAPI_RtcSetAlarm(t_rtc *rtcSetTime);
Parameters	[in] **rtcSetTime**: this time will instead of current alarm time.
Return value	None.
Header	#include "simcom_ api.h"

32.2.4 sAPI_RtcGetAlarm

This API is used to get alarm time.

sAPI_fopen	
Prototype	void sAPI_RtcGetAlarm(t_rtc *rtcSetTime);
Parameters	[in] **rtcSetTime**: current alarm time.
Return value	None.
Header	#include "simcom_api.h"

32.2.5 sAPI_RtcEnableAlarm

This API is used to enable or disable alarm.

sAPI_fopen	
Prototype	void sAPI_RtcEnableAlarm(BOOL onoff);
Parameters	[in] **onoff**: Vlaue 0 is off, 1 is on.
Return value	None.
Header	#include "simcom_api.h"

32.2.6 sAPI_RtcRegisterCB

This API is used to register callback function. Callback function will be executed when alarm time equivalent to RTC time.

sAPI_fopen	
Prototype	void sAPI_RtcRegisterCB(AlCallback callback);
Parameters	[in] callback: callback function. Note: Don’t sleep in callback function.
Return value	None.
Header	#include "simcom_api.h"

32.3 Demo for RTC

```
#include “simcom_api”

void func1(void)
{
        ;
}

int main()
{
        t_rtc timeval;
        /* Get time */
    sAPI_GetRealTimeClock(&timeval);

        /* Set time */
        timeval.tm_min = 1;
        sAPI_SetRealTimeClock(&timeval);

        /* Set alarm */
        sAPI_GetRealTimeClock(&timeval);
        timeval.tm_min += 1;
        sAPI_RtcSetAlarm(&timeval);

        /* Get alarm */
        sAPI_RtcGetAlarm(&timeval);

        /* register callback */
        sAPI_RtcRegisterCB(func1);

        /* enable alarm */
        sAPI_RtcEnableAlarm(1);

        return 0;
}
```