Perfect Wireless Experience
完美无线体验

OpenCPU API User Manual

Version：V1.0.7

Date：2015.08.25

Copyright

Attention

The document is subject to update from time to time owing to the product version upgrade or other reasons. Unless otherwise specified, the document only serves as the user guide. All the statements, information and suggestions contained in the document do not constitute any explicit or implicit guarantee.

Trademark

Versions

Version	Date	Remarks
V1.0.0	2013-06-21	Initial Version
V1.0.1	2013-07-30	Update the name of the document Update chapter 4 and chapter 8
V1.0.2	2013-10-12	Page11, change “timer switch” to” timer Switch” Change “sys_snprintf” to “sys_vsnprintf”
V1.0.3	2013-11-13	Update the name of the document Add G610-A20-XX in applicability table
V1.0.4	2014-04-09	Update section 4 and section 8
V1.0.5	2014-06-17	Add part of description in section 5
V1.0.6	2015-02-07	Add the description of SSL interface

V1.0.7	2015-08-25	Update the logo.

Applicability Type

No.	Type	Note
1	G510-Q50-00	Standard model, it can be upgraded by integrated software
2	G510-Q50-90	Integrated model number, the external label is different from other models.
3	G510S-Q50-00	CE certification included, it can be upgraded by integrated software
4	G610-A20-XX	
5	G610-Q20-XX	

Content

1 Preface

This document mainly introduces the integration Application Programming Interface of G510/G610 Module (for FAE, testers and customers).

2 File System Interface

This is a FAT file system:

1) File name cannot exceed 8 characters (ASC II code), the file name extension are 3 characters, a total of 11 characters. Do not use file name which contains directory structure. The file name must be in ASC II format, it can contain letters (a-z), underscores (_) and numbers (0-9).

2) File reading and writing is operated via file handler

3) If you want to delete a file, make sure the file is not opened by other threads.

4) The file comes with buffer, before you restart the system, please close the file first and wait for 10 seconds

5) To make sure the file is completed written into FLASH, please use sys_file_flush, we suggest you configure the second parameter to 10000, which equals to 10 seconds.

6) The total size of the file system is 1M, as it shared with the module internal procedures, so the available size that can be used by user program is about 512K; there is no limit on the number of the files. Besides, the size of a single file should not exceed 100K, if the file is over 4K, please use sys_file_flush (the suggested value of the second parameter is 10000), waiting for the data be written to flash correctly..

The following table shows the interface function of file operation:

Interface Function	Input Parameter	Return	Description
INT32 sys_file_open (Const INT8 *name UINT32 opt)	<name> No directory structure file name <opt> Open file option	≥0: open (create) successfully, this is the file handle <0: failed to create	Create a specified name file, the file option contains basic options and extension option: Basic option: FS_O_RDONLY: opened read only FS_O_WRONLY: opened write only FS_O_RDWR: opened read and write Extended options can be mixed with the basic options using “I” in

			C language, for example: “FS_O_RDONLY | FS_O_CREATE” FS_O_CREAT, if this option is available, it means that there is a creating file operation when the specified file does not exist, just open the file if it existed. FS_O_EXCL and FS_O_CREAT can be used together to determine whether the specified file exist, if it exists, returns fail, otherwise, the file is created by FS_O_CREATE. FS_O_TRUNC, if the file exists, change the file length to 0. FS_O_APPEND, open the file in appending mode.
INT32 sys_file_close (INT32 fd)	<fd> the opened file handle	0: successfully ≤0: failed	Close file
INT32 sys_file_read (INT32 fd, UINT8 *buf, UINT16 buflen)	<fd> the opened file handle <buf> Pointer of data buffer <buflen> Length of data buffer	≥0: Bytes that have been read <0: error	Read data from current position of opened file, you can use sys_file_seek to set the pointer of current file; the data cannot exceed 1024 bytes
INT32 sys_file_write (INT32 fd,	<fd> the opened file handle <buf>	≥0: Bytes that have been write <0: error	Write data from current position of opened file, you can use sys_file_seek to set the pointer of current file; the data cannot

UINT8 *buf, UINT16 buflen)	Pointer of data buffer <buflen> Length of data buffer		exceed 1024 bytes.
INT32 sys_file_seek (INT32 fd INT32 offset UINT8 opt)	<fd> the opened file handle <offset> Offset from the beginning of file <opt> Original position of offset	≥0: return the position of pointer <0: failed	Move file pointer, and return the pointer position after operation successful. Offset can be null, opt means the original position of the pointer, there are three options: FS_SEEK_SET: move offset bytes from the start of the file. FS_SEEK_CUR: move offset bytes from the current position of the file pointer FS_SEEK_END: move offset bytes from the end of the file
INT32 sys_file_delete (Const INT8 *name)	<name>File name	0: delete successfully ≤0: failed	Close the file first, then delete the file, and make sure this file is not opened by other threads.
INT32 sys_file_clear (Void)	NULL	0: successful ≤0: failed	Delete all the files that are created by user program, make sure all the files are closed when you call this function.
INT32 sys_file_flush (INT32 fd， UINT32 ms)	<fd>File handle <ms> Wait for Refreshing operation	0: successful ≤0: failed	Save the corresponding data of file system buffer to FLASH. If the second parameter is 0, the function is an asynchronous operation, the function successfully returned doesn't mean the data is written in FLASH, it only means the data is

			written to underlying driver, the written time is decided by underlying driver. If the second parameter ms is not 0, it means the program waiting time, this calling will block until time out or the data is successfully written into FLASH.
INT32 sys_file_getSize (INT32 fd)	<fd>File handle	≥0: file length <0: failed	Obtain file size
INT32 sys_file_EOF (INT32 fd)	<fd>File handle	1: the end of the file 0: not the end of the file Null: error	Check the pointer reaches the end of the file or not.

3 Timer Interface

G510 provides a set of timer interface within the project which can be used directly.

Interface Function	Input Parameter	Return	Description
INT32 sys_timer_new (UINT32 ms, Void (*fn)(void *arg), Void *arg)	<ms>Time, the unit is ms <fd>callback function <arg>Parameters of callback function	≥0: Return timer ID Return NULL if it failed.	Create timer, the time and function cannot be 0 (null). The range of timer is 1ms to 10 minutes.
INT32 sys_timer_free (INT32 id)	<id>Timer ID	0: succeed <0: fail	Disable and release timer.

4 Hardware Interface

Interface Function	Input Parameter	Return	Description

INT32 sys_gpio_cfg
(
GAPP_GPIO_ID_T id,
GAPP_GPIO_CFG_T cfg
)
	<id>GPIO ID
	<cfg>GPIO configuration
		0: successfully
		<0: failed
			Configure GPIO port, G510 and G610 support the GPIO as listed below:
			Table 1 for GPIO supported by G510;
			Table 2 for GPIO supported by G610;

INT32 sys_gpio_set (GAPP_GPIO_ID_T id, UINT8 level)	<id>GPIO ID <level>Status value	0: successfully <0: failed	Set the status of output IO port. 0 -- low output 1 --high output
INT32 sys_gpio_get (GAPP_GPIO_ID_T id, UINT8 * level)	<id>IO number <level>Return status	0: successfully <0: failed	Read the status of input IO port and interrupt pin level: When ID is GAPP_IO_9(value 9) can read SIM_CD level; When ID is GAPP_IO_10(value 10) can read G510 WAKEUP or HS_DET PIN level of G610.
INT32 sys_setRTC (GAPP_RTC_T *set)	<set>Pointer of date data	0: successfully <0: failed	Set RTC time, including: Sec: second 0-59 Min: minute0-59 Hour:0-23 Day: 1-31 Month: 1-12 Year: 0-127 (2000-2127) Wday: 1-7

INT32 sys_getRTC (GAPP_RTC_T *get)	<get>Pointer of date data	0: successfully <0: failed	Obtain RTC time, the parameters can refer to the above.
INT32 sys_setRTC_timerMode (GAPP_RTC_MODE_T mode)	<mode> RTC interrupt mode	0: successfully <0: failed	Set interrupt mode: GAPP_RTC_INT_DISABLED: no interrupt GAPP_RTC_INT_PER_SEC: interrupts per second GAPP_RTC_INT_PER_MIN: interrupts per minute GAPP_RTC_INT_PER_HOUR: Interrupts per hour
INT32 sys_setRTC_timerSwitch (INT32 on)	<on>Enable or disable	0: successfully <0: failed	RTC interrupt switch 0—no interruption 1—a interruption, the interrupt mode is GAPP_RTC_INT_DISABLED
INT32 sys_setRTC_timerCB (Void (*cb)(void))	<cb> Timer callback function	0: successfully <0: failed	Set interrupt timers callback function.
INT32 sys_setRTC_alarm (GAPP_RTC_T *set)	<set>Pointer of date data	0: successfully <0: failed	Set clock alarm time
INT32	No parameter	0: successfully	Enable clock function

sys_seRTC_alarmON (Void)		<0: failed	
INT32 sys_seRTC_alarmOFF (Void)	No parameter	0: successfully <0: failed	Disable clock function
INT32 Sys_setRTC_alarmCB (Void (*cb)(void))	<cb> Clock callback function	0: successfully <0: failed	Set the callback function
Void sys_watchdog_enable (UINT32 ms)	<ms>Watch dog time	No return	Set and enable watch dog, if the dog is not feed before the specified time, system restarted. The waiting time range is from 1 second to 60 seconds.
Void sys_watchdog_disable (Void)	No parameter	No return	Disable watch dog function
Void sys_watchdog_feed (Void	No parameter	No return	Feed the dog

)			
UINT32i2c_open(void)	No parameter	0: successfully <0: failed	Open IIC
void i2c_close(void)	No parameter	No return	Close IIC
UINT32 i2c_send_byte(UINT32 slaveAddr,UINT32 memAddr,UINT8 data)	< slaveAddr>IIC Slave address of IIC device < memAddr> The address that need write into IIC device. <data> The data that need read in IIC device.	0: successfully <0: failed	Send a a byte of data by IIC.
UINT32 i2c_get_byte(UINT32 slaveAddr,UINT32 memAddr,UINT8* data)	< slaveAddr> Slave address of IIC device < memAddr> The address that need read from IIC device. <data> To save the data that read from IIC device.	0: successfully <0: failed	Receive a a byte of data by IIC.
UINT32 i2c_send_data(UINT32 slaveAddr,UINT32 memAddr,UINT8 *pData, UINT8 datalen)	< slaveAddr> Slave address of IIC device. < memAddr> The address that need write into IIC device.	0: successfully <0: failed	Send datas with certain length by IIC.

	<pData> The data address that need write into IIC device <datalen> The data length that need write into IIC device.		
UINT32 i2c_get_data(UINT32 slaveAddr,UINT32 memAddr,UINT8 *pData, UINT8 datalen)	< slaveAddr> Slave address of IIC device. < memAddr> The address that need write into IIC device. <pData> Save the address of read data. <datalen> Read the length of data of IIC device.	0: successfully <0: failed	Read datas with certain length by IIC

* Table 1:Here is the GPIO supported by G510

GPIO	Standard module	OpenCPU module	Description
IO0（pin 22）	UART1_RING	GAPP_IO_0	Output only
IO1（pin 17）	UART1_DCD	GAPP_IO_1	Output only
IO2（pin 16）	UART1_DSR	GAPP_IO_2	Output only
IO3（pin 15）	UART1_DTR	GAPP_IO_3	Input/Output. default low level
IO4（pin 38）	LPG	GAPP_IO_4，default is LPG	Input/Output, need set by sys_set, there is one LPG CONTROL in sys_set used for setting control power

			of LPG.
IO5（pin20）	UART1_RTS	GAPP_IO_5	Output only
IO6（pin21）	UART1_CTS	GAPP_IO_6	Intput only

* Table 2:Here is the GPIO supported by G610

GPIO	Standard module	OpenCPU module	Description
IO0（pin 39）	UART1_RING	GAPP_IO_0	Output only
IO1（pin 41）	UART1_DCD	GAPP_IO_1	Output only
IO2（pin 38）	UART1_DSR	GAPP_IO_2	Output only
IO3（pin 40）	UART1_ DTR	GAPP_IO_3	Input/Output. default low level
IO4（pin 49）	LPG	GAPP_IO_4，default is LPG	Input/Output, need set by sys_set, there is one LPG CONTROL in sys_set used for setting control power of LPG.
IO5（pin43）	UART1_CTS	GAPP_IO_5	Output only
IO6（pin42）	UART1_RTS	GAPP_IO_6	Intput only
IO11（pin31）	GPIO07	GAPP_IO_11	Input/Output
IO12（pin32）	GPIO04	GAPP_IO_12	Input/Output
IO13（pin33）	GPIO03	GAPP_IO_13	Input/Output
IO14（pin34）	GPIO02	GAPP_IO_14	Input/Output
IO15（pin35）	GPIO01	GAPP_IO_15	Input/Output
IO16（pin54）	GPIO36	GAPP_IO_16	Input/Output

5 OS Interface Function

Interface Function	Input Parameter	Return	Description
Void sys_taskSleep (UINT32 ms)	<ms>Sleep time, ms	No return	Let the current thread sleep for a while, so that the CPU can be used by other threads. The range of sleep time is from 1ms to 10 minutes.
INT32 sys_taskSend (UINT32 tid; UINT32 msgid; UINT32 n1, UINT32 n2, UINT32 n3)	<tid>Thread ID <msgid>Message ID <n1>Number 1 <n2>Number 2 <n3>Number 3	0: successfully <0: failed	Send a message to a specified thread, the thread ID cannot specify randomly, when the thread starts for the first time, it receives n1 value when MSGID is 0. MSGID cannot be 0.
UINT32 sys_getSysTick (Void)	NULL	System TICK	Obtain the accumulated TICK, system TICK increase 16384Hz
INT32 sys_sem_new (UINT8 v)	<v>Semaphores initial value	>0: successfully, Semaphores ID ≤0: failed	Apply for semaphores
INT32 sys_sem_free (INT32 semid)	<semid> Semaphores ID	0: successfully <0: failed	Release semaphores

Void sys_sem_wait (INT32 semid)	<semid> Semaphores ID	No return	Take the Semaphores, if it becomes 0, thread scheduling comes up.
Void sys_sem_signal (INT32 semid)	<semid> Semaphores ID	No return	Release the Semaphores. Note: Semaphore is a UNIT8 value, make sure it won't reflow when use.
Void * sys_malloc (UINT32 size)	<size> Request memory size	Requested memory starting address, if request failed, it returns NULL (0)	Request memory, the dynamic memory allocated by user program should not exceed 256K.
Void sys_free (void *p)	<p>sys_malloc Allocated memory starting address	No return	Free the memory
Void sys_softreset (Void)	NULL	No return	Restart the system
UINT32 sys_enterCS (Void)	NULL	The system is in interrupt status, sys_exitCS need this parameter	Critical section protection, it is available when it is interrupted
Void sys_exitCS (UINT32 status)	<status>sys_enterC Return value of the command	Null	Left Critical section, it is available when it is interrupted

int setjmp(jmp_buf env)	<env> jmp_buf ,global variable, used for saving the skip point	Return 0 while the first calling, return the second parameter of longjmp() while the second calling	Set the skip address
volatile void longjmp(jmp_buf env,int value)	<enb>, global variable, used for saving the skip point <value> Used for transmit the return value to setjmp(), and judge if skipping or not.	Null	Skip to the address set by setjmp()
void srand(unsigned int init)	<init> The basic value of random number	Null	Set the basic value of random number, not the first return value of the rand()
int rand(void)	Null	Random number	Get a random number

6 Serial Input/Output Interface

Interface Function	Input Parameter	Return	Description
INT32 sys_at_send (UINT8 *cmd, UINT16 len)	<cmd> AT command string (including 0x0d) <len> string length	≥0: the number of bytes which are successfully send out <0: failed	Send AT command to the module, it cannot exceed 1024 bytes; please do not send another command until you receive the return.
Void sys_uart_output (INT32 id, UINT8 *buff, UINT16 len)	<id> UART ID, 0 means UART1, 1 means UART2 <buff> data pointer <len> data length	No return	Output data from UART port, the length cannot exceed 1024 bytes, do not send the data frequently, the interval should higher than or equals to 100ms
INT32 sys_eventTrace (UINT32 value)	<value> 32 bytes data	0: successfully ≤0: failed	Output a event value in trace tools
INT32 sys_textTrace (INT8 *fmt, ...)	Parameters like printf	0: successfully ≤0: failed	Output format string in trace tools, the longest length of BUFF is 240 bytes.

7 Socket Interface

Maximally support 4 socket interfaces.

Interface Function	Input Parameter	Return	Description
INT32 sys_PDPActive (INT8 *apn, INT8 *user, INT8 *pwd, UINT32 *ip)	<apn> APN: Access Point Name <user> username (leave it blank if you don't have a username) <pwd> Password (leave it blank if you don't have a password) <ip>ip return	Return 0 and IP is not 0 means successfully, otherwise, it failed	Activate PDP connection, this is a blocking function.
INT32 sys_PDPRelease (Void)	NULL	0: successfully ≤0: failed	De-activate PDP connection, this is a blocking function.
INT32 sys_PDPStatus (UINT32 *ip)	<ip>returned IP address	Return 0 and IP is not 0 means successfully, otherwise, it failed to activate.	Obtain IP address, check the current status of PDP
INT32 sys_sock_create (UINT32 protocal)	<protocal> protocol type 0: TCP, 1:UDP, 2:SSL	<0: failed ≥0: successfully, this is the socket id	Create a socket
INT32 sys_sock_close	<sock>Socket id	0: successfully	Close a socket which is

(INT32 sock)		≤0: failed	already been opened.
INT32 sys_sock_connect (INT32 sock, GAPP_TCPIO_ADDR_T *dst)	<sock>Socket id <dst>the address information of network byte order	0: successfully ≤0: failed	Create connection with remote socket
INT32 sys_sock_listen (INT32 sock)	<sock>Socket id	0: successfully ≤0: failed	TCP listen
INT32 sys_sock_bind (INT32 sock, GAPP_TCPIP_ADDR_T *dst)	<sock>Socket id <dst>the address information of network byte order, IP is 0	0: successfully ≤0: failed	Bind the local port
INT32 sys_sock_accept (INT32 sock, GAPP_TCPIP_ADDR_T *src)	<sock>Socket id <src>obtain the remote address information	≥0: successfully create a socket ID with remote socket <0: failed	Accept the connect request from remote, return a new socket ID successfully.
INT32 sys_sock_send (INT32 sock,	<sock>Socket id <buff>data starting address	≥0: return the actual data which has been send out <0: failed	Send data, the data cannot exceed 1024 bytes

UINT8 *buff, UINT16 len)	<len>data length		
INT32 sys_sock_send2 (INT32 sock, UINT8 *buff, UINT16 len, GAPP_TCIP_ADD_T *dst)	<sock>Socket id <buff>data starting address <len>data length <dst>remote address information	≥0: return the actual data which has been send out <0: failed	UDP send data to the specified address, the data cannot exceed 1024 bytes.
INT32 sys_sock_recv (INT32 sock, UINT8 *buff, UINT116 len)	<sock>Socket id <buff>the buff starting address of data received <len>buff length	≥0: the actual data length <0: socket error	Receive TCP data function; the read data cannot exceed 2048 bytes.
INT32 sys_sock_recvfrom (INT32 sock, UINT8 *buff, UINT16 len, GAPP_TCPIP_ADDR_T *src)	<sock>Socket id <buff>the buff starting address of data received <len>buff length <src>the returned remote address information	≥0: the actual data length <0: socket error	Receive UDP data function, and return the sender information, the read data cannot exceed 2048 bytes.
INT32 sys_getHostByName (	<hostname> host name <addr> the returned host name corresponding IP	0: successfully ≤0: failed	Obtain the corresponding IP, the domain length cannot exceed 100 bytes, this

INT8 *hostname, Struct ip_addr *addr)			is a blocking function.
INT32 sys_sock_setOpt (INT32 sock, INT32 level, INT32 optname, Const void *optval, INT32 optlen)	<sock>Socket id <level>protocol level(fix number is 6) <optname>set option type <optval>option value pointer <optlen>option value length	0: successfully ≤0: failed	Set socket parameter, the interface is reserved currently.
INT32 sys_sock_getOpt (INT32 sock, INT32 level, INT32 optname, Const void *optval, INT32 optlen)	<sock>Socket id <level>protocol level(fix number is 6) <optname>set option type <optval>option value pointer <optlen>option value length	0: successfully ≤0: failed	Obtain socket parameter, the interface is reserved currently.

8 SSL Interface Function

Interface Function	Input Parameter	Return	Description
INT32 sys_write_ssl_file (INT8 *type_str, UINT8 *buff, UINT16 len)	< type_str > the character string of CA certification type ; "CAFILE" means the client certification; "CAKEY" means the client`s KEY filed; "TRUSTFILE" means the trust certification of server-side; <buff> means there are effective information in certification; <len> means the length of effective information of the certification.	0 means write successfully; 1 means write failed.	< type_str > only the shown three parameters are effective, the others will report error. ".cer" or other suffix types, the "-----xxx----" of CA certification`s tops and tails are invalid information, and cannot write into.
void sys_set_ssl_chkmode (UINT8 mode)	<mode> the certification`s check mode ; 0 : do not check certification from server 1: double sides check.	Null	Set the check mode of SSL certification.
INT32 sys_get_ssl_errcode (void)	Null	0 means normal; -1 means the parameter is error; -2 means the SSL connection is failed; -3 means the file read error; -4 means the connection can not finished, cause	Get the SSL error code (the last time the error happened in SSL answer)

		of the socket cannot write and read effectively. -5 means the write and read operation cannot finished, cause the socket cannot read. -6 means the write and read operation cannot finished, cause the socket cannot write. -7 means the SSL protocol error. -8 means the server-side doesn`t response the handshake of client-side. -9 means the SSL connection closed automatically by server-side. -10 means the unknown error. -11 means the certification`s check is failed. -12 means the information of the length of certification is unmatched. -13 means lack of the encrypted RSA certification. -14 means lack of the signed RSA certification. -15 means cannot find	

		the public key information. -16 means the unknown certification type. -17 means the client-side certification file error. -18 means the client-side key file error. -19 means the trusty server-side certification file error. -20 means get the data with timeout while SSL on a conversation.	

9 Interface of User Parameter Setting

API	Input parameter	Return	Description
INT32 sys_set (GAPP_OPTION_ID_T id, Void *arg, UINT16 len)	<id> operator ID <arg> Parameter pointer <len> Parameter length	0: succeed <0: failed	Miscellaneous Settings, for parameters description please see the table of ”sys_set”.
INT32 sys_get (GAPP_OPTION_ID_T id, Void *arg, UINT16 len)	<id> operator ID <arg> Parameter pointer <len> Parameter length	0: succeed <0: failed	Obtain system parameter, for parameters description please see the table of ”sys_set”.

9.1 sys_set Parameters Description

Operating ID	Corresponding Structure	Corresponding Module	Description
APP_OPT_UART_BAUDRATE_ID	GAPP_OPT_UART_BAUDREATE_T	G510-Q50 G610-A20	Set the baud rate of UART
GAPP_OPT_PIN41_IRQ_ID	GAPP_OPT_PIN_CFG_T	G510-Q50	Set G510 PIN41 interrupt
GAPP_OPT_PIN27_IRQ_ID	GAPP_OPT_PIN_CFG_T	G510-Q50	Set G510 PIN27 interrupt

GAPP_OPT_LPG_CONTROL_ID	GAPP_OPT_LPG_CONTROL_T	G510-Q50 G610-A20	Set LPG control power
GAPP_OPT_PIN47_IRQ_ID	GAPP_OPT_PIN_CFG_T	G610-A20	Set G610 PIN47 interrupt
GAPP_OPT_PIN3_IRQ_ID	GAPP_OPT_PIN_CFG_T	G610-A20	SetG610 PIN3 interrupt
GAPP_OPT_PIN40_IRQ_ID	GAPP_OPT_PIN_CFG_T	G610-A20	SetG610 PIN40 interrupt

9.2 sys_get Parameters Description

Operating ID	Corresponding Structure	Corresponding Module	Description
APP_OPT_UART_BAUDRATE_ID	GAPP_OPT_UART_BAUDREATE_T	G510-Q50 G610-A20	Get the baud rate of UART
GAPP_OPT_LPG_CONTROL_ID	GAPP_OPT_LPG_CONTROL_T	G510-Q50 G610-A20	Get the setting of LPG control power
GAPP_OPT_SYS_VERSION_ID	GAPP_OPT_SYS_VERSION_T	G510-Q50 G610-A20	Read software version of module and API version
GAPP_OPT_APP_UPDATA_ID	GAPP_OPT_APP_UPDATA_T	G510-Q50 G610-A20	Read the mapping filename of APP upgrade
GAPP_OPT_CPU_ID	GAPP_APP_CPUID_T	G510-Q50 G610-A20	Read CPU ID
GAPP_OPT_MMAD_ID	GAPP_APP_MMAD_T	G610-A20	Read ADC and temperature detection value

10 Other Interface

Interface Function	Input Parameter	Return	Description
INT32 sys_vsnprintf (INT8 *buff, UINT32 n, const INT8 *fmt, va_list ap)	Similar with snprintf	The length of data which has been written in Buff.	Formatted output to the buffer
INT32 sys_GB2UNI (UINT16 gb_char, UINT16 *uchar)	<gb_char> GB code pointer <uchar> Unicode pointer	0: successful <0: failed	Convert a GB code to a unicode
INT32 sys_UNI2GB (UINT16 ucode, UINT16 *pDst)	<ucode>Unicode <pDst> GB code pointer	After conversion succeeds, the actual bytes of GB code	Convert a Unicode to a GB code
INT32 sys_hookUart (INT32 id, INT32 op)	<id>UART id <op> whether hook UART	0: successful <0: failed	Hook UART function, id 0 means UART1; id1 means UART2, when op is 0 means free the UART, op 1 means hook UART. After op is set to 1, when there is data transferring in UART, sys_callback->uart_input will be called.

11 User Callback Function

When you write user program, you must define a global variable which is called sys_callback and SYS_CALLBACK_T type. The module will obtain this variable automatically when the user program is started, and calls the callback function at appropriate time.

Callback Function	Input Parameter	Return	Description
UINT8 (*init) (GAPP_TASK_T **ptl)	<ptl> the returned task(thread) list	The actual task number	When module is initializing, it calls init to initialize the respond data, read a thread list and start and configure the respond thread.
Void (*uart_input) (INT32 uid, UINT8 *data, UINT16 len)	<uid> UART ID <data> UART data pointer <len> UART data length	No return	After sys_hookuart is called, the uart input will be transferred to the function pointed by uart_input as a parameter, the max length is 2048 bytes.
Void (*at_resp) (UINT8 *rsp, UINT16 rsplen)	<rsp> AT port data starting address <rsplen> the data length	No return	Virtual AT command will be called after received data; the max length is 2048 bytes.
Void (*sys_signal) (GAPP_SIGNAL_ID_T sig, Va_list arg)	<sig> system signal (system event) ID <arg>variable parameter	No return	System signal corresponds to a system event, such as PDP and socket event, different events in different threads are issued and called this callback function.

12 Macro Definition

When the system event is be triggered, calls sys_callback->sys_signal to notify user program, user program can deal with issue or not.

Macro name	Value	Significance
GAPP_SIG_PDP_RELEASE_IND	0	After PDP is activated, if an exception occurs in the network or inside the system which causes PDP released, this signal will be triggered.
GAPP_SIG_SOCK_CONNECT_RSP	1	After TCP calls sys_sock_connect, it returns OK, and try to communicating with remote server, if it is successful, this signal will be triggered, this signal only has one parameter which is UINT32, and this parameter is a socket id (it can be transferred to INT32 by force).
GAPP_SIG_SOCK_TCPXON_IND	2	TCP flow control, there are two parameters, the first parameter is UINT32, which means socket id, the second parameter means the status of TCP flow control, if it is 0, it means it can send out data, otherwise, it means TCP cannot send more data, you must wait.
GAPP_SIG_SOCK_CLOSE_IND	3	TCP close event, normally the reason is that the remote side close TCP proactively, the first parameter UINT32 means socket id.
GAPP_SIG_SOCK_ERROR_IND	4	Socket abnormal, the first parameter means the wrong socket id.
GAPP_SIG_ACCEPT_IND	5	After TCP listening, if there is any connection request from client in the remote, this event will be triggered, the first parameter UINT32 means socket id, you will need this function when use sys_sock_accept.
GAPP_SIG_CLOSE_WITH_FIN_RSP	6	Reserved
GAPP_SIG_SOCK_SEND_RSP	7	Send respond to event, when socket send data and be processed by underlying processor, this event will be triggered, the first parameter is UINT32, which means socket id, you will need this function when use

		sys_sock_accept.
GAPP_SIG_SOCK_CLOSE_RSP	8	TCP close successful event, the first event UINT32 means socket id.
GAPP_SIG_SOCK_DATA_RECV_IND	9	Socket receives the data and triggers this event, the first parameter UINT32 means socket id.

13 API Return Code

Macro name	Value	Significance
GAPP_RET_OK	0	Succeed
GAPP_RET_PARAMS_ERROR	-1	Parameter error
GAPP_RET_NOT_INIT	-2	System is not initialized, or the system doesn't support this interface.
GAPP_RET_MEMORY_ERROR	-3	Memory error
GAPP_RET_OPTION_NO_SUPPORT	-4	Option not supported
GAPP_RET_TIMETOUT	-5	Function executed timeout
GAPP_RET_UNKNOW_ERROR	-6	Unknown error
GAPP_RET_TASK_ERR_BASE	-100	Task error code begins
GAPP_RET_TASK_MSG_TOO_MUCH	-101	Too much task message
GAPP_RET_THREAD_NOT_CREATED	-102	Thread is not created yet
GAPP_RET_THREAD_TOO_MUCH	-103	Too much thread
GAPP_RET_TASK_ERR_MAX	-150	Task error code ends
GAPP_RET_TIMER_ERR_BASE	-151	Timer error code begins
GAPP_RET_TIMER_TOO_MUCH	-152	Too much timer
GAPP_RET_TIMER_NOT_CREATE	-153	Timer is not created yet
GAPP_RET_TIMER_ERR_MAX	-200	Timer error code ends
GAPP_RET_PDP_ERR_BASE	-201	PDP error code begins

GAPP_RET_PDP_NOT_ACTIVE	-202	PDP is not activated
GAPP_RET_PDP_BUSY	-203	PDP is busy (activating or de-activating)
GAPP_RET_PDP_ERR_MAX	-250	PDP error code ends
GAPP_RET_TCPIP_ERR_BASE	-251	TCPIP error code begins
GAPP_RET_DNS_BUSY	-252	DNS is busy
GAPP_RET_DNS_ERROR	-253	DNS error
GAPP_RET_TCPIP_ERROR	-254	TCPIP internal error
GAPP_RET_TCPIP_ERR_MAX	-300	TCPIP error code ends
GAPP_RET_RTC_ERR_BASE	-301	RTC error code begins
GAPP_RET_RTC_ERR_0	-302	RTC hardware error
GAPP_RET_RTC_ERR_MAX	-350	RTC error code ends
GAPP_RET_TRACE_ERR_BASE	-351	TRACE error code begins
GAPP_RET_TRACE_ERR_TIMEOUT	-352	Output timed out
GAPP_RET_TRACE_ERR_MAX	-400	TRACE error code ends