SIM5360 BMP Demo AT Command Note V1.00

Document Title:	SIM5360 BMP Demo AT Command Demo Note
Version:	1.00
Date:	2013-12-13
Status:	Release
Document Control ID:	SIM5360_BMP_Demo_AT_Command_Note_V1.00

General Notes

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

Copyright

Version history

Date	Version	Description of change	Author
2013-12-13	1.00	Origin	qiujianhua

SCOPE

This document describes how to use AT command interface for develop.

CONTENTS

1 INTRODUCTION

This technology guide describes the Brew® Mobile Platform (Brew MP) AT Command, The AT command Interface is not included in Brew MP 1.0.4. It's an extended interface of BMP. It mainly contain the following interface:

• modem_IATCommand: Provides at command functions.

• IPort1: Provides functions to allow read and write at command value.

2 SAMECODE

The at command is a fundamental service provided by an operating system. The AT command provides the following services used by Brew MP-based applications and OS Services:

• Send an AT command to command processor.
• read AT command response data.
• To register with the ATCOP for unsolicited result codes

We provide a a sample code about how to use AT command interface. The sample code is available to help you build, modify, test and execute specific tasks. Our sample code is organized by Brew API family so you can add the features and functionality you want to your application. The detail code see the demo project.

2.1 Send an AT Command to command processor

To send an AT Command, an application needs to do the following:

- Call ISHELL_CreateInstance() to create an instance of the modem_IATCommand interface.

The following sample code shows how to create modem_IATCommand instance:

```
if (ISHELL_CreateInstance(pMe->piShell, modem_AEECLSID_ATCOMMAND,
                                (void**)&pMe->piATCmd) != AEE_SUCCESS)
{
        DBGPRINTF_HIGH("Create modem_IATComand fail");
}
```

- Call IQI_QueryInterface() to create an instance of the IPort1 interface.

The following sample code shows how to create IPort1 instance:

```
nErr = IQI_QueryInterface((IQI*)pMe->piATCmd, AEEIID_IPort1,
                                                (void**)&pMe->piPort1);

if(nErr != AEE_SUCCESS){
    DBGPRINTF_HIGH("query IPort1 fail");
}
```

- Call IPort1_Write () to send command to processor

```
nErr = IPort1_Write(pMe->piPort1, (const byte*)pMe->write_buf,
                        pMe->write_len,&writelen);
```

If return value is equal to AEE_SUCCESS, Means send command success,and then you can call IPort1_Read() to read the at command response data. Sometimes, if there is no data available, it will return AEE_EWOULDBLOCK, at this time, you need call

IPort1_Writeable to wait the sign signal of data available

```
nErr = IPort1_Write(pMe->piPort1, (const byte*)pMe->write_buf,
                    pMe->write_len,&writelen);
DBGPRINTF_HIGH("IPort1_Write nErr = %d, writelen = %d", nErr, writelen);
if(AEE_EWOULDBLOCK == nErr){
    ISignalCBFactory    *piSignalCBFactory = NULL;
    nErr = ISHELL_CreateInstance(pMe->piShell,
                        AEECLSID_SignalCBFactory,(void**)&piSignalCBFactory);
    if (AEE_SUCCESS != nErr)
    {
        DBGPRINTF_HIGH("create SCBFactory    fail");
        return;
    }
    nErr = ISignalCBFactory_CreateSignal(piSignalCBFactory,
                                         WriteToBuffer_CB,
                                         (void*)pMe,
                                         &pMe->piSignal,
                                         &pMe->piSignalCtl);
    IQI_RELEASEIF(piSignalCBFactory);
    if(AEE_SUCCESS == nErr){
        IPort1_Writeable(pMe->piPort1,pMe->piSignal);
    }
}else if(AEE_SUCCESS == nErr){
    read_at_buffer(pMe);       // read AT command response data
}
```

When the signal of data available is received, the callback function(WriteToBuffer_CB) will be called. The callback function：

```
static void WriteToBuffer_CB(void *pvCxt)
{
    ATCommand_Demo *pMe = (ATCommand_Demo*) pvCxt;
    int writelen, nErr;
    nErr = IPort1_Write(pMe->piPort1, (const byte*)pMe->write_buf,
                        pMe->write_len,&writelen);
    if(AEE_SUCCESS == nErr){
        DBGPRINTF_HIGH("WriteToBuffer_CB success IPort1_Write nErr = %d, writelen
        = %d", nErr, writelen);
        read_at_buffer(pMe);       // read AT command response data
    }else{
        DBGPRINTF_HIGH("qiu WriteToBuffer_CB error");
    }
}
```

2.2 Read AT command response data

When send AT command to processor success. We can read the response data. To Read AT command response data, an application needs to do the following:

- Call IPort1_Read() to read the response data.
- If the return value is AEE_SUCCESS, the prepared buf will be filled response data.
- Generally, IPort1_Read will not immediately return AEE_SUCCESS. It will return AEE_EWOULDBLOCK, We need to register a signal and wait for the response data signal. When the response data is available, the callback function will be called.
- In callback function, we call IPort1_Read() again to get the response data.

```
static void read_at_buffer(ATCommand_Demo *pMe){
    int nErr;
    int readlen;
    MEMSET(pMe->read_buf,0,sizeof(pMe->read_buf));
    nErr = IPort1_Read(pMe->piPort1, (byte *)pMe->read_buf, sizeof(pMe->read_buf),
                       &readlen);
    if(AEE_EWOULDBLOCK == nErr){
        ISignalCBFactory     *piSignalCBFactory = NULL;
        nErr = ISHELL_CreateInstance(pMe->piShell,
            AEECLSID_SignalCBFactory,(void**)&piSignalCBFactory);
        if (AEE_SUCCESS != nErr)
        {
            DBGPRINTF_HIGH("create SCBFactory    fail");
            return;
        }
        nErr = ISignalCBFactory_CreateSignal(piSignalCBFactory,
                                             ReadFromBuffer_CB,
                                             (void*)pMe,
                                             &pMe->piSignal,
                                             &pMe->piSignalCtl);
        IQI_RELEASEIF(piSignalCBFactory);
        if(AEE_SUCCESS != nErr){
            DBGPRINTF_HIGH("CreateSignal fail");
        }else{
            IPort1_Readable(pMe->piPort1,pMe->piSignal);
        }
    }else if(AEE_SUCCESS == nErr){
        DBGPRINTF_HIGH("IPort1_Read nErr = %d, writelen = %d", nErr, readlen);
        DBGPRINTF_HIGH("IPort1_Read buf = %s",pMe->read_buf);
    }
}
```

The callback function:

```
static void ReadFromBuffer_CB(void *pvCxt)
{
	ATCommand_Demo *pMe = (ATCommand_Demo*) pvCxt;
	int readlen;
	int nErr;
	nErr = IPort1_Read(pMe->piPort1,(byte *)pMe->read_buf,sizeof(pMe->read_buf),
&readlen);
	if(AEE_SUCCESS == nErr){
		pMe->read_len = readlen;
		DBGPRINTF_HIGH("qiu ReadFromBuffer_CB IPort1_Read nErr = %d, writelen
				= %d", nErr, readlen);
		DBGPRINTF_HIGH("qiu ReadFromBuffer_CB IPort1_Read buf =
				%s",pMe->read_buf);
	}else{
		DBGPRINTF_HIGH("ReadFromBuffer_CB error");
	}
}
```

2.3 To register with the ATCOP for unsolicited result codes

Sometimes, the application need to catch the unsolicited result codes, for example SMS notify modem_IATCommand provide an interface for registering the callback.

- Call ISignalCBFactory_CreateSignal () to create a signal and bind a callback function.
- Call modem_IATCommand_RegisterForURC() to register the signal.
- When the application catch the unsolicited result codes, the callback function will be called.
- In callback function, call modem_IATCommand_ReadURC() to read the unsolicited data.

```
nErr = ISHELL_CreateInstance(pMe->piShell,
                                AEECLSID_SignalCBFactory,(void**)&piSignalCBFactory);
if (AEE_SUCCESS != nErr)
{
    DBGPRINTF_HIGH("create SCBFactory    fail");
    return;
}
nErr = ISignalCBFactory_CreateSignal(piSignalCBFactory,
                                        ReadURC_CB,
                                        (void*)pMe,
                                        &pMe->piSignal,
                                        &pMe->piSignalCtl);
IQI_RELEASEIF(piSignalCBFactory);
if(AEE_SUCCESS != nErr){
    DBGPRINTF_HIGH("CreateSignal fail");
}else{
    nErr = modem_IATCommand_RegisterForURC(pMe->piATCmd,pMe->piSignal);
    if(AEE_SUCCESS != nErr){
        DBGPRINTF_ERROR("RegisterForURC Fail");
    }
}
```

Callback function:

```
static void ReadURC_CB(void *pvCxt)
{
    ATCommand_Demo *pMe = (ATCommand_Demo*) pvCxt;
    int nErr;
    int bufLenReq;
    char buf[100];
    ISignalCtl_Enable(pMe->piSignalCtl);
    nErr = modem_IATCommand_ReadURC(pMe->piATCmd, (byte *)&buf, 100,
                                        &bufLenReq);
    if(AEE_SUCCESS != nErr){
        DBGPRINTF_ERROR("ReadURC Fail");
    }else{
        DBGPRINTF_ERROR("ReadURC buf = %s bufLenReq = %d",buf,bufLenReq);
    }
}
```

Contact us:

Shanghai SIMCom Wireless Solutions Ltd.

Add: Building A, SIM Technology Building, No.633 Jinzhong Road, Changning District, Shanghai, P. R. China 200335

Tel: +86 21 3252 3300

Fax: +86 21 3252 3020

URL: www.sim.com/wm