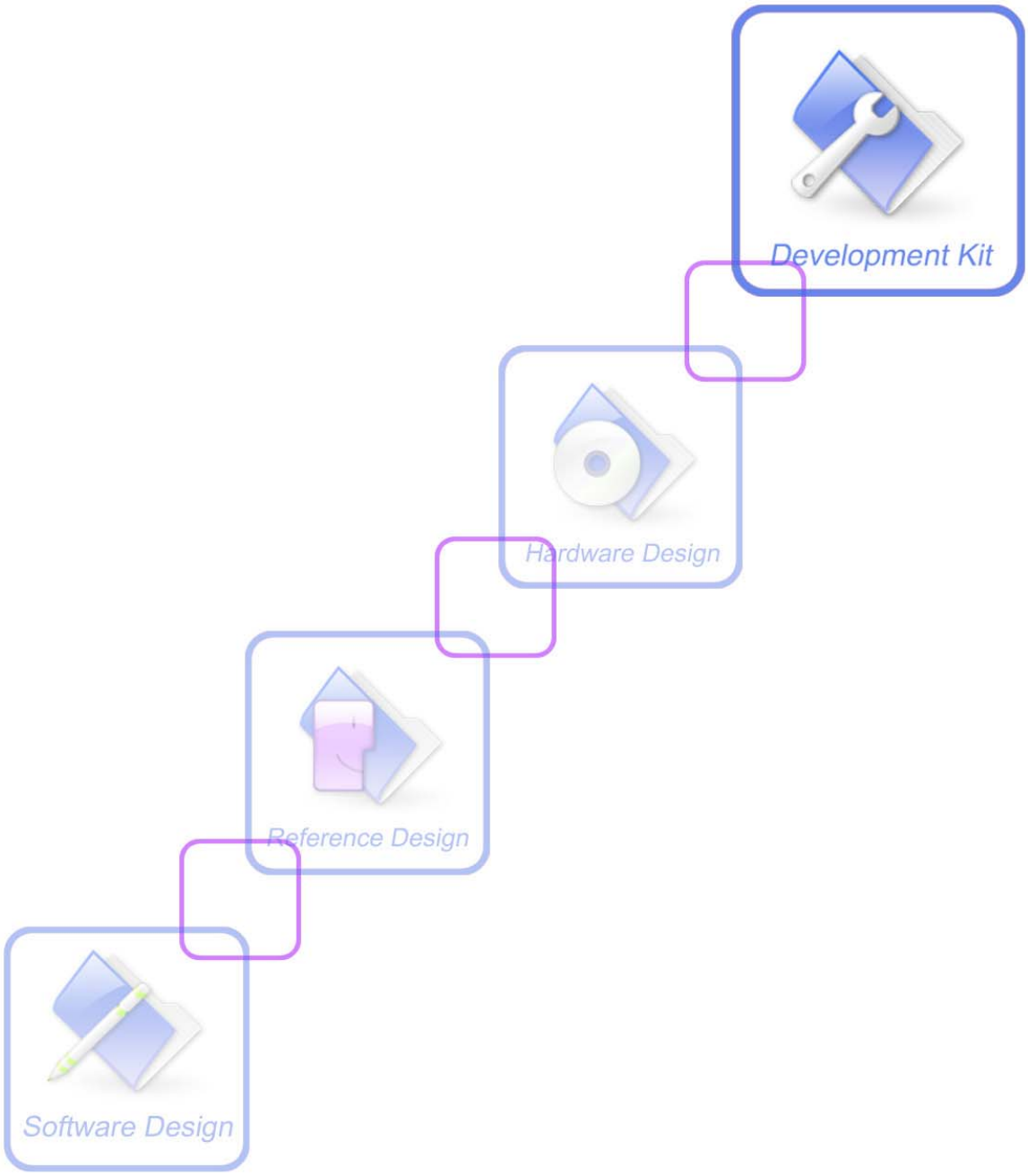
A company of SIM Tech

# SIM5360_GPIO_Demo Note_V1.00



| | |
|---|---|
| **Document Title:** | SIM5360 BMP Demo GPIO Note |
| **Version:** | 1.00 |
| **Date:** | 2013-12-13 |
| **Status:** | Release |
| **Document Control ID:** | SIM5360_BMP_Demo_GPIO_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 GPIO interface for developing.

# CONTENTS

# 1 INTRODUCTION

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

• device_IGPIO: Provides GPIO functions.

# 2 EXAMPLES

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

• Get and Set pin value.

• GPIO_Interrupt.

We provide a a sample code about how to use GPIO 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 Set pin value

To set gpio pin value, an application needs to do the following:

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

The following sample code shows how to create modem_IATCommand instance:

```
if(ISHELL_CreateInstance(pMe->piShell, device_AEECLSID_GPIO,
                                (void**)&pMe->piGPIO) != AEE_SUCCESS){
        DBGPRINTF("create GPIO instance fail");
}
```

- Call device_IGPIO_GetPinConfiguration () to get pin config. If the current GPIO direction is not equal device_GPIO_DIR_OUTPUT, Call device_IGPIO_SetPinConfiguration() to set the GPIO direction as device_GPIO_DIR_OUTPUT.

```
nErr = device_IGPIO_GetPinConfiguration(pMe->piGPIO, nGPIO, &nDir, &nPull);
if (nErr != AEE_SUCCESS) {
    DBGPRINTF_ERROR("device_IGPIO_GetPinConfiguration fail");
}
if(nDir !=  device_GPIO_DIR_OUTPUT){
    nDir = device_GPIO_DIR_OUTPUT;
    nErr = device_IGPIO_SetPinConfiguration(pMe->piGPIO, nGPIO, nDir);
    if (nErr != AEE_SUCCESS)
    {
        DBGPRINTF_ERROR("device_IGPIO_SetPinConfiguration fail");
    }
}
```

- Call device_IGPIO_SetPinValue() to set gpio pin value

```
nErr = device_IGPIO_SetPinValue(pMe->piGPIO, nGPIO, (device_GPIO_ValType)nValue);
if (nErr != AEE_SUCCESS)
{
    DBGPRINTF_ERROR("device_IGPIO_SetPinValue fail");
}
```

## 2.2 Get pin value

To set gpio pin value, an application needs to do the following:

- Call ISHELL_CreateInstance() to create an instance of the device_IGPIO interface.
- Call device_IGPIO_GetPinConfiguration () to get pin config. If the current GPIO direction is not equal device_GPIO_DIR_INPUT, Call device_IGPIO_SetPinConfiguration() to set the GPIO direction as device_GPIO_DIR_INPUT.

```
nErr = device_IGPIO_GetPinConfiguration(pMe->piGPIO, nGPIO, &nDir, &nPull);
if (nErr != AEE_SUCCESS) {
    DBGPRINTF_ERROR("device_IGPIO_GetPinConfiguration fail");
}
if(nDir !=  device_GPIO_DIR_INPUT){
    nDir = device_GPIO_DIR_INPUT;
    nErr = device_IGPIO_SetPinConfiguration(pMe->piGPIO, nGPIO, nDir);
    if (nErr != AEE_SUCCESS)
    {
        DBGPRINTF_ERROR("device_IGPIO_SetPinConfiguration fail");
    }
}
```

- Call device_IGPIO_GetPinValue() to set gpio pin value:

```
nErr = device_IGPIO_GetPinValue(pMe->piGPIO, nGPIO, &nValue);
if (nErr != AEE_SUCCESS)
{
    DBGPRINTF_ERROR("qiu device_IGPIO_GetPinValue fail");
}
```

## 2.3 GPIO interrupt

Each GPIO pin may be configured to generate an interrupt on Rising Edge, Falling Edge and Change From Read event. The interrupt behavior of GPIO pins are controlled by the registers. To generate an interrupt, an application needs to do the following:

- Call ISHELL_CreateInstance() to create an instance of the device_IGPIO interface.
- Setting the Setting the interrupt condition. There are some conditions:

```
//Falling Edge
device_GPIO_PolarityType nPolarity = device_GPIO_INT_ACTIVE_LOW;
device_GPIO_DetectType nDetect = device_GPIO_INT_DETECT_EDGE;

//Rising Edge:
device_GPIO_PolarityType nPolarity = device_GPIO_INT_ACTIVE_HIGH;
device_GPIO_DetectType nDetect = device_GPIO_INT_DETECT_EDGE;

//high level:
device_GPIO_PolarityType nPolarity = device_GPIO_INT_ACTIVE_HIGH;
device_GPIO_DetectType nDetect = device_GPIO_INT_DETECT_LEVEL;

//low level:
device_GPIO_PolarityType nPolarity = device_GPIO_INT_ACTIVE_LOW;
device_GPIO_DetectType nDetect = device_GPIO_INT_DETECT_LEVEL;
```

- Set direction type for a GPIO pin. Must be set as device_GPIO_DIR_INPUT. And set GPIO Index. Call device_IGPIO_SetPinConfiguration() to set the configuration of GPIO

```
device_GPIO_DirType          nDir = device_GPIO_DIR_INPUT;
pMe->nGPIO = TEST_GPIO_INTERRUPT_INDEX;
nErr = device_IGPIO_SetPinConfiguration(pMe->piGPIO, pMe->nGPIO, nDir);
if (nErr != AEE_SUCCESS)
{
      DBGPRINTF_ERROR("qiu device_IGPIO_SetPinConfiguration fail");
}
```

- Create an ISignalCBFactory instance . And then use the ISignalCBFactory instance to create a singal which bind a callback.

```
nErr = ISHELL_CreateInstance(pMe->piShell,
                                AEECLSID_SignalCBFactory,(void**)&piSignalCBFactory);
if (AEE_SUCCESS != nErr)
{
    DBGPRINTF_HIGH("create SCBFactory    fail");
        goto INTERRUPT_FAIL;
}
nErr = ISignalCBFactory_CreateSignal(piSignalCBFactory,
                                        Interrupt_CB,
                                        (void*)pMe,
                                        &pMe->piSignal,
                                        &pMe->piSignalCtl);
IQI_RELEASEIF(piSignalCBFactory);
if(AEE_SUCCESS != nErr){
    DBGPRINTF_HIGH("CreateSignal fail");
    goto INTERRUPT_FAIL;
}
```

- Call device_IGPIO_RegisterPinNotifier() to register a pin notify.

```
nErr = device_IGPIO_RegisterPinNotifier( pMe->piGPIO,
                                            pMe->nGPIO,
                                            nPolarity,
                                            nDetect,
                                            pMe->piSignal);
if(AEE_SUCCESS != nErr){
    DBGPRINTF_ERROR("RegisterPinNotify Fail nErr = %d",nErr);
}
```

- **A**nd then if there is a GPIO interrupt. The interrupt will call the callback function through the signal. The callback function as blew:

```
static void Interrupt_CB(void *pvCxt){
    GPIO_Demo *pMe = (GPIO_Demo *) pvCxt;
    //deregister notify.
    if(AEE_SUCCESS != device_IGPIO_DeregisterPinNotifier(pMe->piGPIO,
                                                        pMe->nGPIO ,pMe->piSignal)){
        DBGPRINTF_HIGH("DeregisterPinNotifier Fail");
    }
    IQI_RELEASEIF(pMe->piSignal);
    IQI_RELEASEIF(pMe->piSignalCtl);
}
```

**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