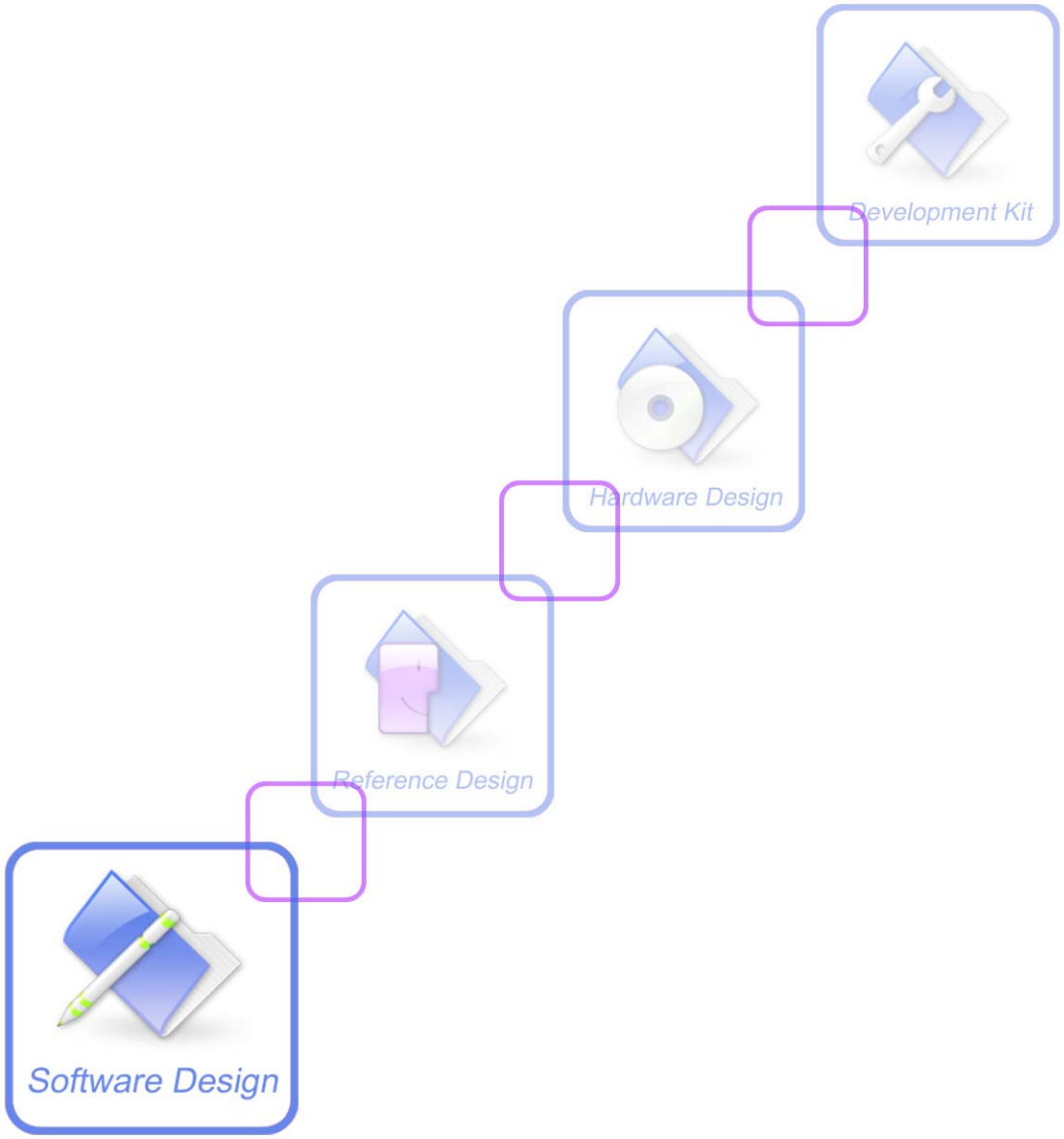
A company of SIM Tech

SIM900_DTMF EmbeddedAT® Application Note_V1.01



Document Title:	SIM900_DTMF Embedded AT® Application Note
Version:	1.01
Date:	2012-02-07
Status:	Release
Document Control ID:	SIM900_DTMF Embedded AT_Application Note_V1.01

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.

Contents

Version history

Date	Version	Description of change	Author
2012-02-07	V1.01	Origin	Chenyang

1 DTMF EVENT

EVENT is wrapped in structure FlEventBuffer, by which the core system communicates with the embedded applications. Only through eat1_02GetEvent(&flEventBuffer), EVENTs can be passed from the core system to the embedded applications. Structure FlEventBuffer consists of two parts. one is the event type, which defines the type of the EVENT, and the other is the event data.

```
typedef struct FlSignalBufferTag
{
    FlEventType   eventTyp;
    EventData     eventData;
}FlEventBuffer;
```

1.1 EVENT Type

1.1.1 FlEventType

DTMF EVENT is categorized as following:

```
typedef enum FlEventTypeTag
{
    ......
    EVENT_DTMF,
    ......
    EVENT_MAX = 0xFF
}FlEventType;
```

1.1.2 EVENT_DTMF

The event can be triggered when DTMF function is enabled (see chapter 2.1 for details).

1.1.3 Example

The following code skeleton demonstrates how events are captured in embedded application:

```
void fl_entry() //customer entrance
{
    /* some code here */
    switch(flEventBuffer.eventTyp) // deal with signals according to its type
    {
        …
```

```
            case EVENT_DTMF:
            /* add here to deal with signal associated to EVENT_DTMF*/
            break;
            …
            default:
            break;
        }
}
```

1.2 EVENT Data

1.2.1 EventData

Each DTMF related EVENT type has its corresponding EVENT data.

```
typedef union EventDataTag
{
    ......
    DTMF_EVENT          dtmf_evt;
    ......

}EventData;
```

Note EventData is not like EventType, EventData is a union, and each data type has its own structure, which is detailed in the following sections.

1.2.2 DTMF_EVENT

```
typedef struct DTMF_EVENTTag
{
    ascii demfChar;
    u8 reserve[3];
}DTMF_EVENT;
```

demfChar: The character of DTMF.

2 DTMF API

This chapter categorizes DTMF related API functions and describes their usages, including function prototype, parameters, and their return values.

2.1 ebdat10_06DTMFDetectEnable

This function is used to enable/disable DTMF detect function.

- **Prototype**

```
s32 ebdat10_06DTMFDetectEnable (bool isEnable);
```

- **Parameter**

isEnable: 0 disable
1 enable

- **Return values**

FL_OK: DTMF detection set successfully
FL_ERROR: Incorrect parameter

Appendix A: Example

The following example is used CRWP function to enable or disable DTMF detection. When DTMF detection is enabled, the character of the DTMF can be captured by EVENT_DTMF event as below:

```
void fl_entry()
{
    bool keepGoing = TRUE;
    FlEventBuffer flEventBuffer;
    u32 para1,i;
    ebdat7_00EnterDebugMode();
    while (keepGoing == TRUE)
    {
        memset((u8*)&flEventBuffer,0x00,sizeof(flEventBuffer));
        eat1_02GetEvent(&flEventBuffer);
        switch(flEventBuffer.eventTyp)
        {
            case EVENT_MODEMDATA:
            {
                flEventBuffer.eventData.modemdata_ev.data[flEventBuffer.eventData.mode
                                        mdata_evt.length]=0;
                ebdat7_01DebugTrace((const char*)
                flEventBuffer.eventData.modemdata_evt.data);
                if(MODEM_CRWP == flEventBuffer.eventData.modemdata_evt.type)
                {
                    /*get AT+CRWP parameters*/
```

```
                    i = sscanf(const char *)flEventBuffer.eventData.modemdata_evt.data,
                    “AT+CRWP=%d”,&para1);
                    if(para1>1)
                    {
                       ebdat7_01DebugTrace(“Parameter error”);
                    }
                    /*enable or disable DTMF detection 1:enable    0:disable*/
                    if(ebdat10_06DTMFDetectEnable(para1)==FL_OK)
                    {
                       ebdat7_01DebugTrace(“successfully”);
                    }
                    else
                    {
                       ebdat7_01DebugTrace(“fail”);
                    }
                }
                /*MODEM_CMD or MODEM_DATA*/
                else
                {}
            }
            break;
            case EVENT_DTMF:
            {
                /*show the character of the DTMF*/
                ebdat7_01DebugTrace((const char *) “DTMF:%c”,
                                    flEventBuffer.event.dtmf_evt.dtmfChar );
            }
            break;
            default:
            break;
        }
    }
}
```

Contact us:
Shanghai SIMCom Wireless Solutions Ltd
Add: SIM Technology Building A,
No. 633, Jinzhong Road, Shanghai, P. R. China 200335
Tel: +86 21 3252 3300
Fax: +86 21 3252 3020
URL: www.sim.com