BenQ

M25 GSM Wireless Module

Customer development package manual

Rev.0.1

2007/1/30

BenQ Corporation.

Networking & Communications BG

18 JiHu Road, Nei-Hu, Taipei 114, Taiwan

Tel: +886-2-2799-8800

Fax: +886-2-2799-8332

http://www.benq.com

DOCUMENT REVISION HISTORY			
ISSUE	**DATE**	**AUTHOR**	**COMMENTS**
[0.1]	2007/1/30	Kevin RY Cheng	Creation

1. M25 software architecture

This Module platform provides dual channels for MMI task to control protocol stack and hardware device respectively. These service interfaces are Internal AT command interface and driver service interface, as shown in Figure 1.

MMI task sends message which carries string based AT command to protocol stack. By this way, MMI task must register AT command call back function for handling occurred event. Hence, MMI task controls communication procedure via this “Internal AT command” directly without extra hardware device support. Moreover, previous BenQ module platform uses AT command to control hardware device. This platform provides simply and uniform specific driver API for operating hardware device.

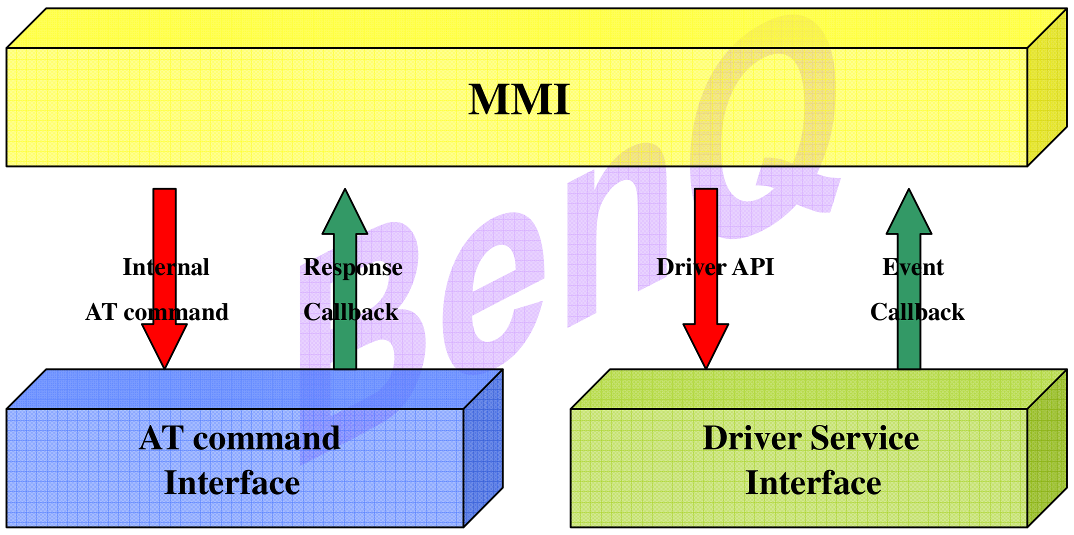

Figure 1 Service interfaces for MMI task

1.1. Internal AT command API

MMI task sends AT command string via internal AT command interface, and receives response message from registered callback function. The follow APIs are used to handle those procedures.

- If_at_init() : This function is used to register callback function.
- If_at_cmd() : This function is used to send internal AT command.

1.2. Driver service interface

BenQ provide complete specific driver API, which are:

- LCD driver API
  - 128*64 Black & White SPI control LCD.
- Keypad driver API
- Audio driver API
  - Three sound types: key-beep, tone, melody.
  - Two audio paths: hand-held, hand-free.
- GPIO driver API
  - 5 GPIO pins for customers.
- Real Time Clock API
- Flash File System driver API
- Power Management API
- Communication Channel API
  - Communicate with external device via UART or USB port.

1.3. Dual image architecture

This platform provides one advanced code building policy which is “dual image” architecture. Convention is compiling and link whole source files into one image file. Hence, customer must maintain lots of library files and cost much time to download the huge image file.

Dual image architecture is divide source code into two parts. The core procedures which contain system, drivers and protocol are linked into one image file, and MMI procedures of customer are built into another image file independently. By this method, code management is more convenient. After module product leaving the factory, the core image file is already downloaded into module and users only need to download the MMI image. Hence, it costs users less time for proceeding download procedure.

2. Console daemon

BenQ provides an MMI task prototype with development phase debug environment, which called “console daemon”. This software component simulates the MMI task to communicate with module system core. Customers will know how to implement MMI task in this platform easily and conveniently via this task prototype. The debug environment with command line interface is used to examine the correctness of procedure and export the debug message via the serial port.

3. Compile environment

3.1. Install M25 platform compiler

- TMS4701x_2.54 : Compile tool package which includes compiler, archiver, linker and some standard libraries.
  - Copy the whole TMS4701x_2.54 folder to C:\ (root directory).

3.2. Code base directory description

- Build folder : This folder includes one makefile (MMI.mak) and one batch file (MMIbuild.bat) for building MMI source code.
  - MMI.mak : The makefile contains rules for compiling console daemon (MMI prototype) source code, and user can directly modify it for compiling self-own source files.
  - MMIbuild.bat : When execute this batch file at command line mode of Windows XP (or Windows 2000) operating system, it will start compiling and linking MMI source code.
  - gnumake.exe : Standard compile tool (make).
- MMI folder : This folder contains console daemon source codes and three sub-folders.
  - cons_cmd.c, cons_cmd.h : console daemon command line function set.
  - cons_cmd_hdlr.c, cons_cmd_hdlr.h : console daemon command line parser.
  - cons_cmd_tbl.c, cons_cmd_tbl.h : console daemon command line fuction table.
  - cons_ker.c : console daemon MMI task body.
  - cons_ret_cb.c, cons_ret_cb.h : event callback functions.
  - Obj folder : After building MMI source code, the generated object code will store in this output folder.
  - If_obj folder : Service interface object code which contains driver API and AT command API must be linked with MMI object code.
    - atcmd_if.obj : AT command API object code.
    - drv_if.obj : driver API object code.
    - cons_tramp.obj : dual image policy object code.
  - Include folder : Contains service interface header files and MMI task includes these header file for using service APIs.
    - atcmd_if.h : AT command API header file.
    - drv_if.h : driver API header file.
    - Os_if.h : OS service API header file.

- Out folder : This folder includes image files.
    - M25[version_number].m0 : System core image file.
    - MMI.m0 : Generated MMI image file.
    - MMI.map : Generated MMI memory mapping file.
- Sys folder : This folder contains runtime support libraries and one command file (MMI.cmd) which contains memory mapping information for linking image file.
    - rts16le_flash.lib, rts16le_int_ram.lib : runtime support libraries.
    - mmi.cmd : memory mapping command file.
- Version folder : Version number text of system core image.

4. Download image

In development phase, customers must download both M25 system core image file and MMI image file. BenQ provides USB based GUI download tool for customer to download image files.

i. Download MMI image file.
ii. Then, download system core image.
iii. After rebooting the module device, module will start initialization procedures and communicating with network.

The download tool manual contains detailed download tool configuration and operating steps.

5. Command line interface of console daemon

Console daemon provides command line interface for development phase debugging. After downloading image files and resetting module device, the default setting of UART port is used for sending external AT command. Connect EVB board with PC terminal via UART port and execute Windows hyper-terminal application. Hyper-terminal application will show the follow message:

```
AT-Command Interpreter ready
_
```

This message shows that external AT command interface is ready. Input AT command string from hyper-terminal into UART port of module and this AT command string is directly sending to AT command interpreter which is component of ACI protocol. On the other hand, internal AT command is message with AT command string sent from MMI task to ACI.

5.1. Enter command line mode of console daemon

Input one specific AT command “at$cons” to enter into console daemon command line mode. If command line mode changes successfully, hyper-terminal will show “cmd%” string.

```
AT-Command Interpreter ready
at$cons
cmd%
```

In this mode, the inputted character string is sending to console daemon (MMI prototype). Then, console daemon task sends internal AT command to ACI or executes specific command line function. For example, we defined one command line function “cmd_if_audio_melody_start” to test correctness of audio driver API “if_audio_melody_start”. Input command “cmd_if_audio_melody_start” and its arguments (single space between arguments) for executing this command line function. If this command is inputted correctly, melody will start playing. Otherwise, it will shows “cmd error” string on hyper-terminal.

Correct

```
AT-Command Interpreter ready
at$cons
cmd%cmd_if_audio_melody_start 1 0
cmd%
```

Wrong

```
AT-Command Interpreter ready
at$cons
cmd%cmd_if_audio_melady_start 1 0
cmd error
cmd%
```

If we want to input internal AT command in command line mode, input “>” character and then it is available to input internal AT command. Input AT command character string and this string will be transmitted to console daemon task. Then, console daemon task will send Internal AT command to ACI. And input “<” character to return command function input mode.

```
AT-Command Interpreter ready
at$cons
cmd%>
at              ⇐   Internal AT command
ok
<
cmd%            ⇐   Command line function mode
```

5.2. Add new command line function

The command line interface is extendable for user adding or modifying command line functions. The adding function steps are:

i. Add new command line function body into cons_cmd.c.

```
Ex:
void cmd_hello
(       unsigned int iArgNum  ⇐  Number of argument
        char * module)
{
        c_printf("\n\rHello %s!", module);
}
```

ii. Add new data entry (contains function information) into command function table(cons_cmd_tbl.c).

```
const tCOMMAND atCmdTbl[] =
{       Data entry #n
        { "hello",                    ⇐   Command
          (MyFunc)cmd_hello,          ⇐   Function name
          "command line example",     ⇐   Description
          {  "module", (STRING_ARG | REQUIRED_ARG),  ⇐  Arguments and its
               NULL, 0   ⇐   Terminal                   property (string)
          }
        },
        NULL
};
```

After finishing these steps, we can type "hello" command to execute "cmd_hello" this command line function.

```
AT-Command Interpreter ready
at$cons
cmd%hello M25
Hello M25!
```

Argument input

5.1 Compliance with FCC Rules and Regulations

The FCC Equipment Authorization Certification for the M25 reference application described in Section 5.1 is listed under the
FCC identifier JVPM25
IC: 6175A-M25
granted to BenQ Corporate.

This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions:
1.this device may not cause harmful interference, and
2.this device must accept any interference received, including interference that may cause undesired operation.
Changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate the equipment.

Note:

1. The maximum antenna gain allowed for use with this device is 0 dBi.
2. When the module is installed in the host device, the FCC ID label must be visible through a window on the final device or it must be visible when an access panel, door or cover is easily removed. If not, a second label must be placed on the outside of the final device that contains the following text: “Contains FCC ID: JVPM25”.