Wireless Module Expert

M85 OpenCPU Solution Presentation

2013/09/22

OUTLINE

OpenCPU Summary

OpenCPU is an embedded development solution for M2M field. Based on it, customers can design their applications conveniently since it can enable customers to create innovative applications and embed them directly into Quectel modules and easily migrate the application software to different MCU platform. In the OpenCPU solution, GSM/GPRS module acts as a main processor. So, GSM/GPRS module with OpenCPU solution facilitates customer's product design and accelerates the application development.

M85 OpenCPU module is a powerful functional Quad-band GSM/GPS module in LCC castellation packaging. It integrates advanced eCall/Era-Glonass, UFS, recording and audio playing function into a compact form factor of 24.5×25.3×2.6mm, which fully meets developers' comprehensive requirements for large space storage and extends the functionality of the application at no additional cost.

OpenCPU module has been widely used in M2M field, such as tracker & tracing, automotive, energy, etc.

OUTLINE

Advantages – Low cost, Fast Develop

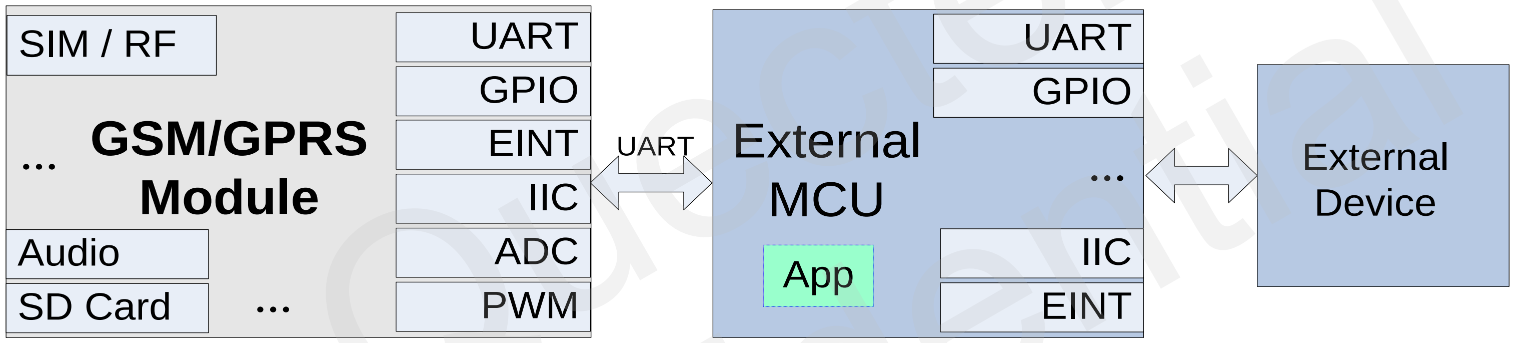
- Reduce product development time
- Simplify circuit design and reduce costs and power consumption
- Decrease the product’s size
- Upgrade firmware remotely via OpenCPU FOTA
- Improve the performance-to-price ratio and enhance the competitive advantages

Low cost, Fast develop

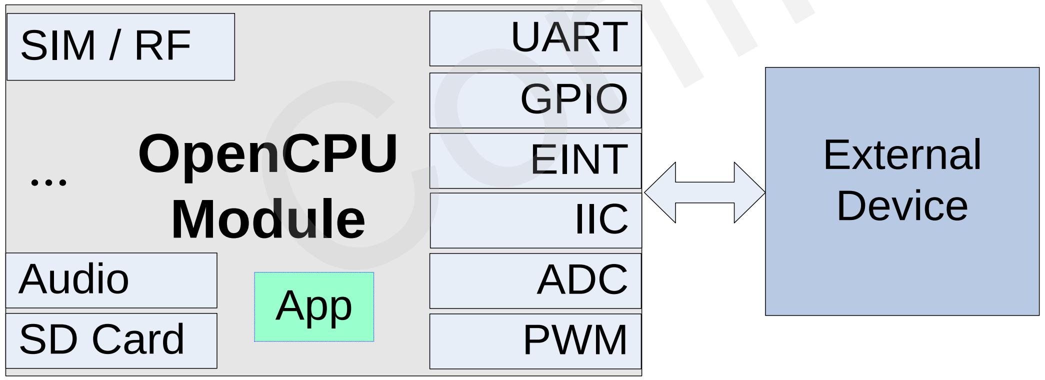
Advantages – Easy Hardware Design

Compared with traditional solution, OpenCPU solution can make hardware design easier for the developer. The figure below shows the differences between them.

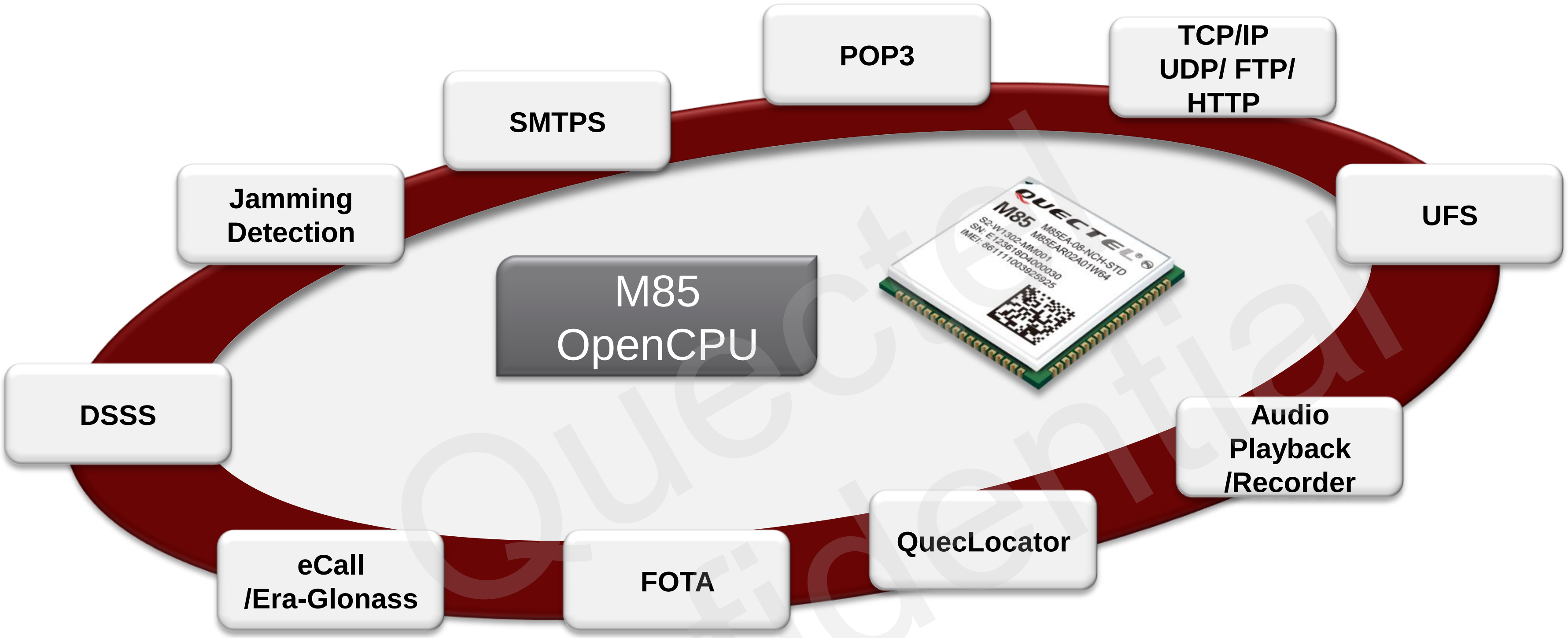
■ Traditional Solution



■ OpenCPU Solution



Advantages - Enhanced Technology



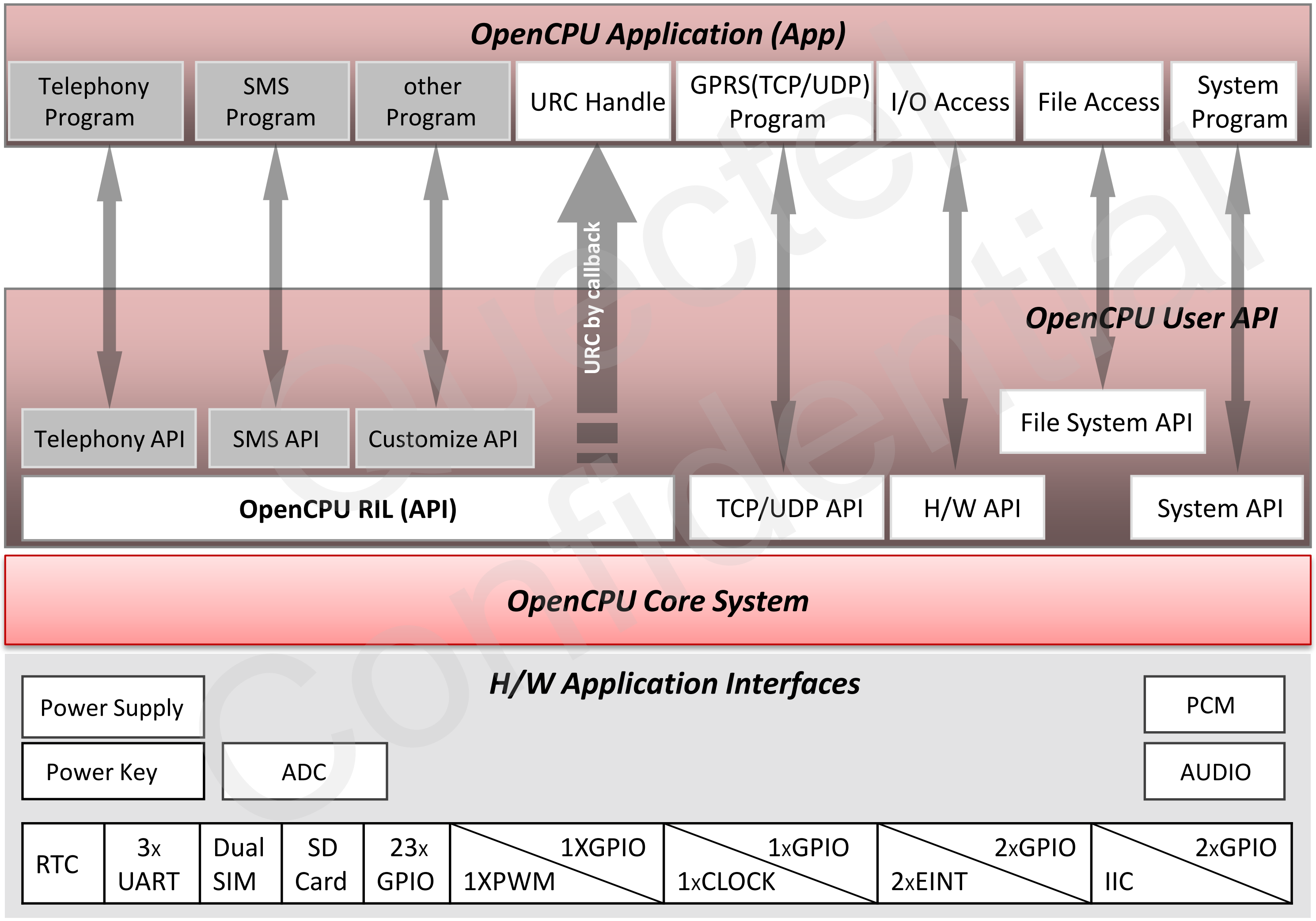
Quality Guarantee

- Reliable network protocol
- Steady flash protected mechanism
- Superior audio algorithms

OUTLINE

Software Architecture (1)

System software of M85 OpenCPU consists of 3 layers: Core system, User API, Application. The following block diagram is the software architecture of M85 OpenCPU.



Software Architecture (2)

- **Core System**

Core System is a combination of hardware and system software of GSM/GPRS module. It has a built-in ARM7EJ-S processor, and has been built over Nucleus operating system, which has the characteristics of micro-kernel, real-time, multi-tasking and etc.

- **OpenCPU RIL**

A open source layer, OpenCPU RIL are embedded into User API layer. Using the OpenCPU RIL, developer can simply call API to send AT and get the response when API returns.

AT commands related to SMS and telephone are packed to the RIL APIs by default. Besides, developer can also easily add a new API to implement an AT command according to the open source of RIL.

OUTLINE

What's New? (1)

■ OpenCPU RIL Support

Compared with the previous OpenCPU solution, the new M85 OpenCPU module provides RIL diver in user API layer, which makes software design easier. When the developer sends AT commands to core system, he or she can get the response directly by RIL layer. M85 OpenCPU can provide the Open Source of RIL as well. The following C code describes the differences in details.

➢ **M85 OpenCPU solution**

```
 s32  RIL_NW_GetSignalQuality(u32* rssi, u32* ber)
{
          s32 retRes = 0;
         char strAT[] = "AT+CSQ\0";
         ST_CSQ_Reponse pCSQ_Reponse;

          //  Here Sending "AT+CSQ" command
          retRes = Ql_RIL_SendATCmd(strAT,Ql_strlen(strAT), ATResponse_CSQ_Handler,(void*)&pCSQ_Reponse,0);

         // Now the response of "AT+CSQ" is just in pCSQ_Reponse, then you can get the CSQ value
         if(RIL_AT_SUCCESS == retRes)
         {
                    *rssi = pCSQ_Reponse.rssi;
                    *ber = pCSQ_Reponse.ber;
         }
          return retRes;
}
```

What’s New? (2)

➢ **The previous OpenCPU solution**

```
// Here, sending AT+CSQ
void SendAtCmd()
{
   s32 ret;
   ret = Ql_SendToModem(ql_md_port1, (u8*)"AT+CSQ\r", Ql_strlen("AT+CSQ\r"));
}
//Here, get the response of AT+CSQ in main loop body.
void ql_entry( )
{
   switch (g_cmd_idx)
   {
      case 1:// Echo mode off
       Ql_sprintf((char *)buffer, "ATE0\n");
   while(1)
   {
      Ql_GetEvent(&g_event);
      switch(g_event.eventType)
      {
         case EVENT_MODEMDATA:
         {
            //TODO: receive and handle data from CORE through virtual modem port
            PortData_Event* pPortEvt = (PortData_Event*)&g_event.eventData.modemdata_evt;
            // Here receive URC and the AT response
            break;
         } } }
}
```

What’s New? (3)

- **GCC Compiler Support**
  - M85 OpenCPU supports free-of-charge GCC compiler (Sorcery CodeBench Lite ARM EABI) and supports ARM RVCT as well.
  - The previous OpenCPU solution only supports ARM RVCT.

- **IDE Support**
  - M85 OpenCPU supports a plenty of tools, such as Source Insight, and Microsoft Visual Studio, helping you to manage or edit your program.
  - The former version of OpenCPU only supports Source Insight tool to manage or edit program.

What’s New? (4)

■ Easy to Set Initial Configuration of GPIO

M85 OpenCPU provides a very simple method to set the initial configuration of some GPIOs when these GPIOs need to be initialized after the early boot time. For example, GPIOs are used to control the power supply of peripheral circuit.

Here is an example code to configure the initial status of NETLIGHT pin. When the module powers up, NETLIGHT pin will be initialized to “output”, “low level” and “pull-down inside module”.

```
/*--------------------------------------------------------------------------------------------------------------------------------------
Function Name     Pin Name                  Direction(In or Out)           Level              Pull Selection(Down or Up)
--------------------------------------------------------------------------------------------------------------------------------------*/
GPIO_ITEM (  PINNAME_NETLIGHT,      PINDIRECTION_OUT,     PINLEVEL_LOW,      PINPULLSEL_PULLDOWN)
```

Besides, developer may call the GPIO-related APIs to reprogram the GPIO pin. For example, to call Ql_GPIO_SetLevel() may change the level of GPIO pin.

What’s New? (5)

■ Easy to Add New Task (Thread)

Developer can simply follow the procedures below to add a task in “custom_task_cfg.h” file to define a new task.

➢**Task Id Name:** Task Id Name is a totally customized name. Developer can define the name, and the system will automatically define and assign the value.

➢**Task Stack Size**: The range of task stack size is from 1KB to 10KB. If there are any files operations in the task, the task size must be set to at least 4KB, otherwise, the tack overflow probably happens.

➢**Default Value2**: Developer does not specify the value and set it to the default definition.

```
/*---------------------------------------------------------------------------------------------------------------------------
            Task Entry function      Task Id Name    Task Stack Size(Bytes)    Default Value1        Default Value2
----------------------------------------------------------------------------------------------------------------------------*/
TASK_ITEM (Proc_main_task,       main_task_id,           4*1024,           DEFAULT_VALUE1,    DEFAULT_VALUE2)
```

What's New? (6)

- **Programmable Power Key Pin for App**

  Here is an function example on how to power on or power off the module by Power Key pin. It is very easy to program Power Key pin for the developer.

```
static const ST_PowerKeyCfg pwrkeyCfg = {
{
   TRUE,  // working mode for power-on on PWRKEY pin
   /*
   Module automatically powers on when feeding a low level to POWER_KEY pin.
   When setting to FALSE, the callback that Ql_PwrKey_Register registers will be trigged. Application must
   call Ql_LockPower () to lock power supply, or module will lose power when the level of PWRKEY pin goes
    high.
   */
   TRUE,  // working mode for power-off on PWRKEY pin
   /*
   Module automatically powers off when feeding a low level to POWER_KEY pin.
   When setting  to FALSE, the callback that Ql PwrKey_Register registers will be trigged.
   Application may do post processing before switching  off the module.
   */
};
```

What’s New? (7)

■ Programmable Reset Pin for App

EMERG_OFF pin can work in two modes. One is “POWER DOWN” mode, the other is “RESET” mode, which is the default value.

```
static const ST_EmergOffCfg emergoffCfg = {
EMERGOFF_RESET };
```

In EMERGOFF_RESET mode, when a low level is fed to EMERG_OFF pin, the module resets.

In EMERGOFF_POWEROFF mode, when a low level is fed to EMERG_OFF pin, the module powers down.

OUTLINE

Open Resources (1)

System Resource on M85 OpenCPU Module

- **CPU**

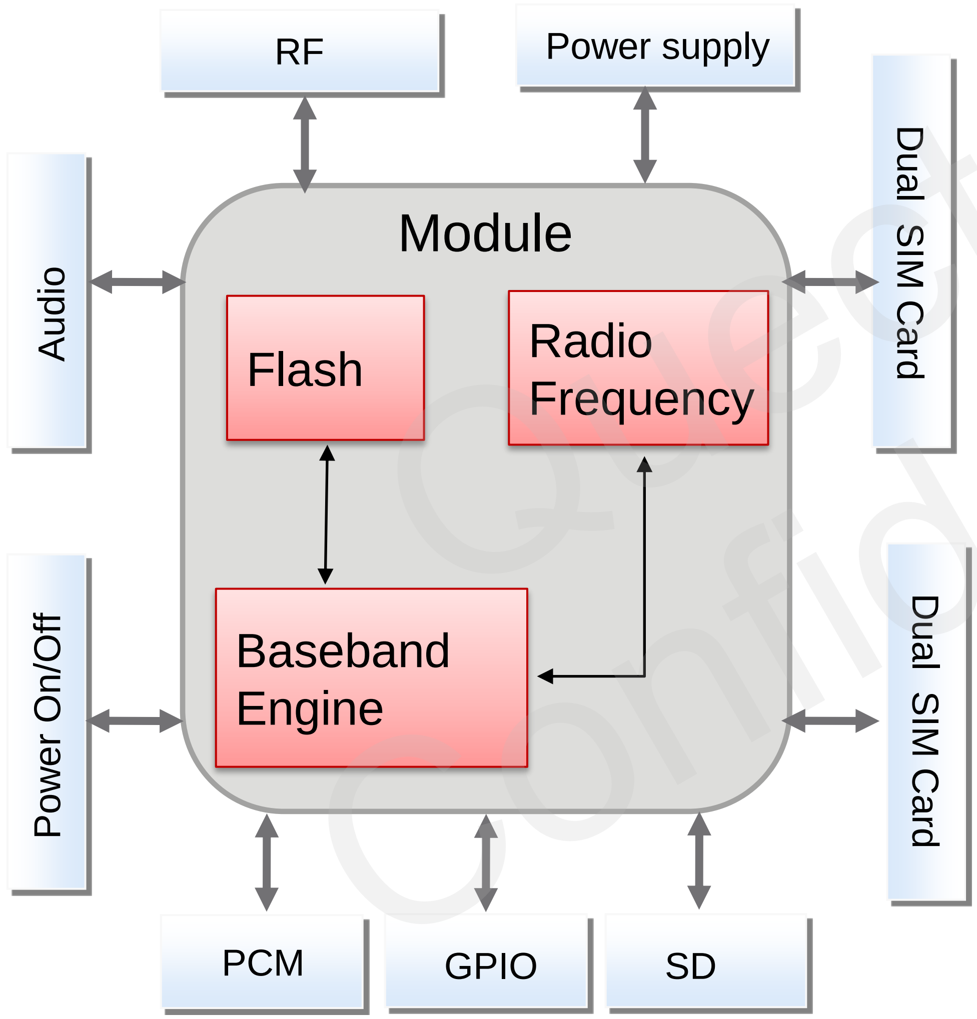
  32-BIT ARM7EJ-S^{TM} RISC 104MHz

- **MEMORY (64Mb Flash + 32Mb RAM)**

  - Code Region: 640KB space for App image bin
  - RAM: 2MB space for application program
  - File Region: 1MB space

Open Resources (2)

Hardware Architecture



Hardware Resource

- 3 × UARTs
- RTC
- EINT Interfaces
- PWM
- Dual-SIM Card Interface
- PCM
- SD Card Interface
- 3 × Audio Output Interfaces
- I^{2}C Bus
- ADC
- Power on/off

Open Resources (3)

Programmable Multifunctional Pins:

PIN No	PIN NAME	RESET	MODE1	MODE2	MODE3
3	PINNAME_GPIO0	I/PU	GPIO		
4	PINNAME_NETLIGHT	I/PD	NETLIGHT	GPIO	PWM_OUT
15	PINNAME_STATUS	I/PD	STATUS	GPIO	
18	PINNAME_PCM_IN	I/PD	PCM_IN	GPIO	
19	PINNAME_PCM_CLK	I/PD	PCM_CLK	GPIO	
20	PINNAME_PCM_OUT	I/PD	PCM_OUT	GPIO	
21	PINNAME_PCM_SYNC	I/PU	PCM_SYNC	GPIO	
24	PINNAME_GPIO1	I/PD	GPIO		
25	PINNAME_GPIO2	I/PD	GPIO		
26	PINNAME_GPIO3	I/PD	GPIO		
27	PINNAME_GPIO4	I/PD	GPIO		
28	PINNAME_GPIO5	I/PD	GPIO		
29	PINNAME_GPIO6	I/PU	GPIO		
30	PINNAME_GPIO7	I/PU	GPIO		
31	PINNAME_GPIO8	I/PU	GPIO		
32	PINNAME_GPIO9	I/PU	GPIO		
33	PINNAME_GPIO10	I/PU	GPIO		
45	PINNAME_DCD	I/PD	DCD	GPIO	CLOCK
46	PINNAME_RI	I/PD	RI	GPIO	
47	PINNAME_DTR	I/PU	DTR	GPIO	EINT
48	PINNAME_CTS	I/PD	CTS	GPIO	
49	PINNAME_RTS	I/PU	RTS	GPIO	
57	PINNAME_SIM_PRESENCE	I/PU	SIM_PRES	GPIO	EINT

OUTLINE

Development Requirements (1)

Host System Requirements

The following host operating systems and architectures should be supported:

- Microsoft Windows XP (SP1 or later)
- Windows Vista
- Windows 7 systems using IA32, AMD64, and Intel 64 processors.

Compiler & IDE Requirements

- GCC Compiler (Sorcery CodeBench Lite for ARM EABI)
- IDE: Eclipse or Microsoft Visual Studio (Optional)

Development Requirements (2)

Programming Language Requirement

- Basic C-language programming knowledge

SDK and Other Requirements

- Quectel GSM/GPRS Module with OpenCPU Feature
- Quectel EVB
- OpenCPU SDK
- Firmware Download Tool (included in SDK)
- FOTA Package Tool (included in SDK)

Wireless Module Expert

Q&A…

Thank you