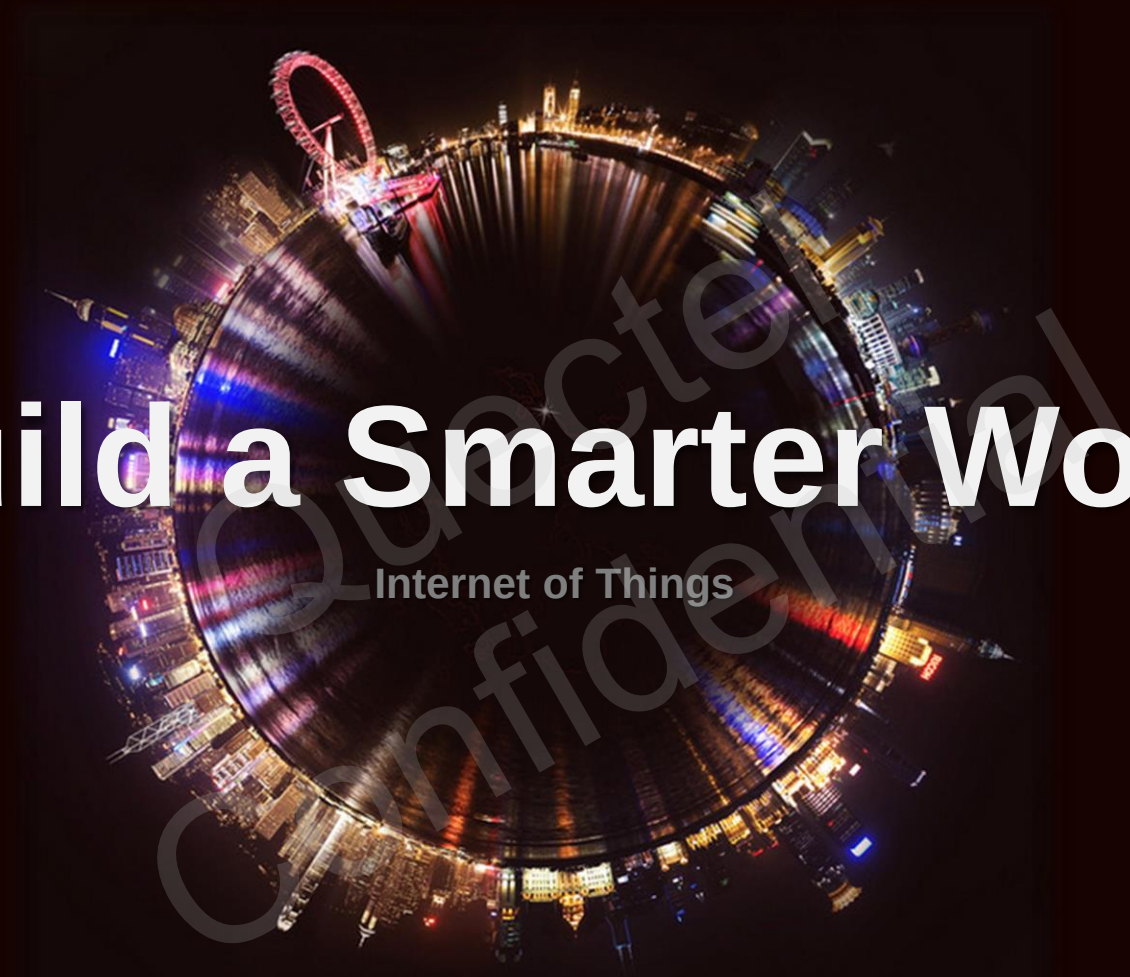
MC60-OpenCPU Solution Presentation

July, 2016

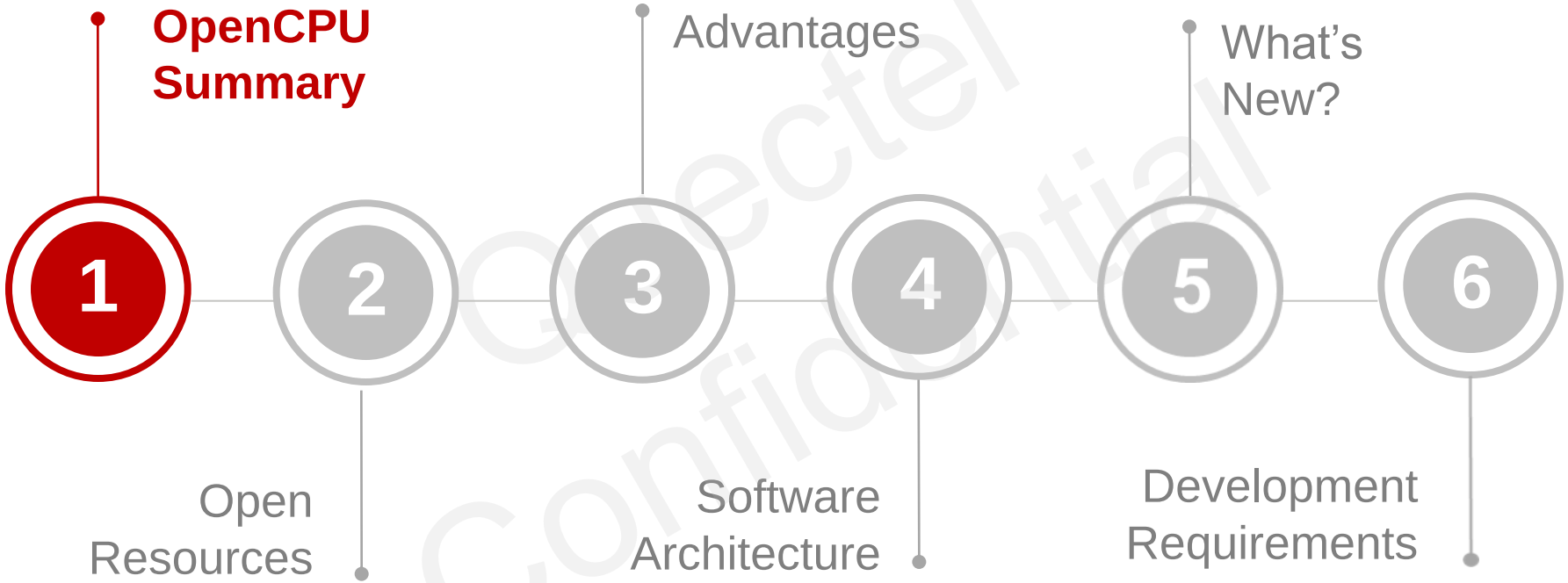
www.quectel.com



Build a Smarter World

Internet of Things

Contents



OpenCPU Summary

OpenCPU is an embedded development solution for M2M field. Based on it, customers can conveniently design embedded applications. It enables customers to create innovative applications and download it directly into Quectel modules to run. In the OpenCPU solution, GSM/GPRS module acts as a main processor. So, GSM/GPRS module with OpenCPU solution facilitates customers' product design and accelerates the application development.

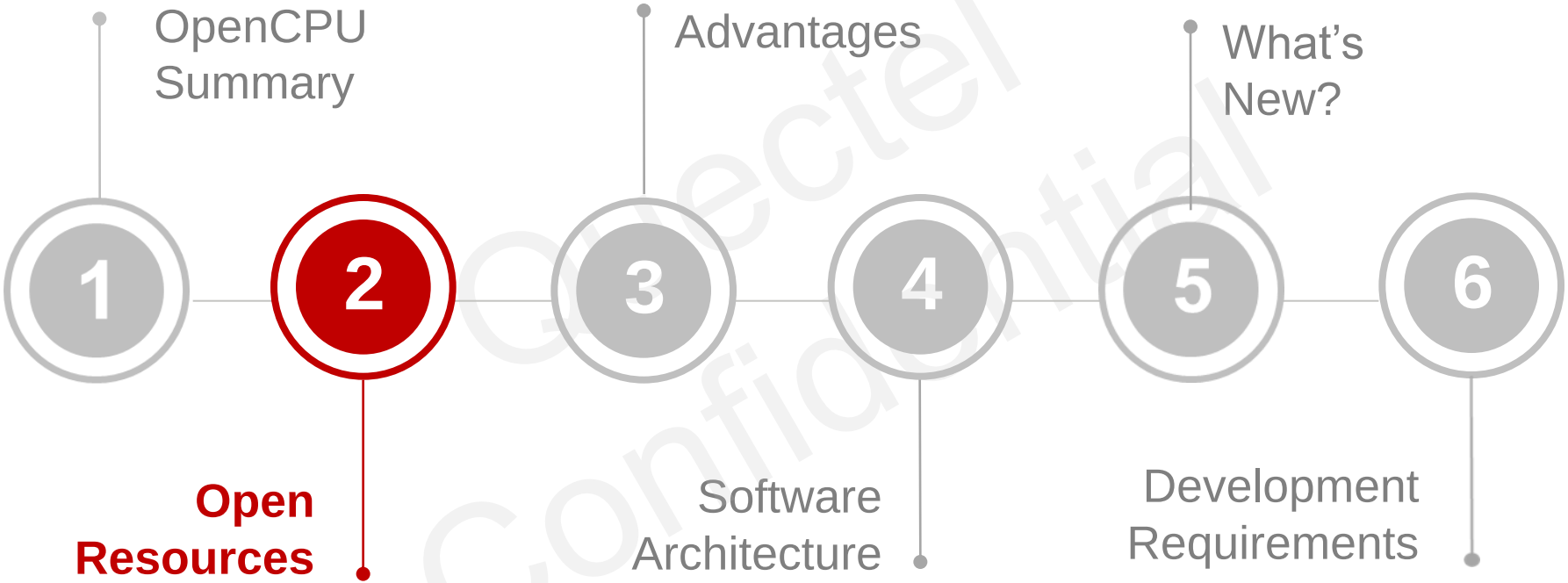
MC60-OpenCPU Module is a powerful functional quad-band GSM/GPRS module in LCC castellation packaging, featuring embedded GNSS function. It has an ultra-compact profile of 18.7×16.0×2.1mm. In addition to Internet, telephony, SMS, FOTA, and data storage functions, it builds in advanced audio player, recorder, QuecLocator, eCall, navigation system and the new featured "Bluetooth" function, which make this module a best choice for applications that have strict requirements on extended functions and cost-effectiveness.

OpenCPU module can be widely used in M2M field, such as tracker & tracing, automobile, energy, wearable devices, etc.





Contents



System Resources on MC60-OpenCPU Module

- **CPU**

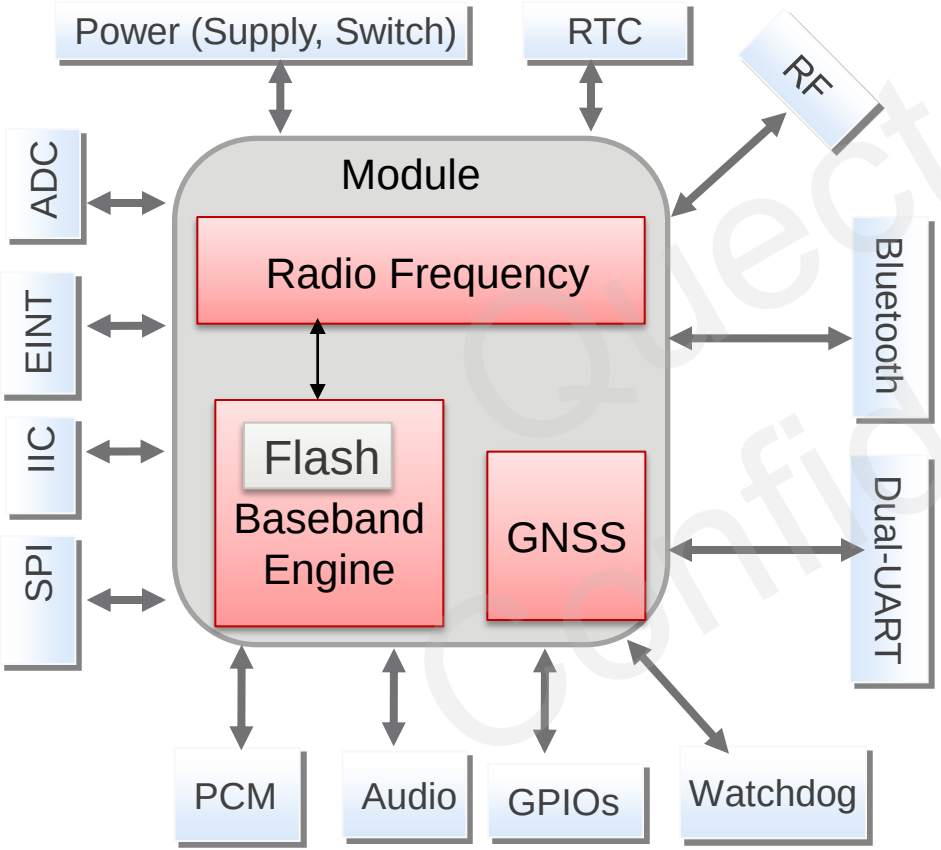
  32-BIT ARM7EJ-S™ RISC 260MHz

- **MEMORY (4MB Flash + 4MB RAM)**

  - Code Region: 320KB space for App image bin
  - RAM: 100KB static memory and 500KB dynamic memory
  - UFS Region: 120KB space

Open Resources (2)

Hardware Architecture



Hardware Resources

- Power supply
- Power switch
- RTC
- 2 UART ports
- ADC
- PCM
- Audio interfaces (2 output channels and 1 input channel)
- Bluetooth
- GPIO interfaces
- PWM output interface
- EINT interfaces
- IIC interface
- SPI interface
- Watchdog
- GNSS

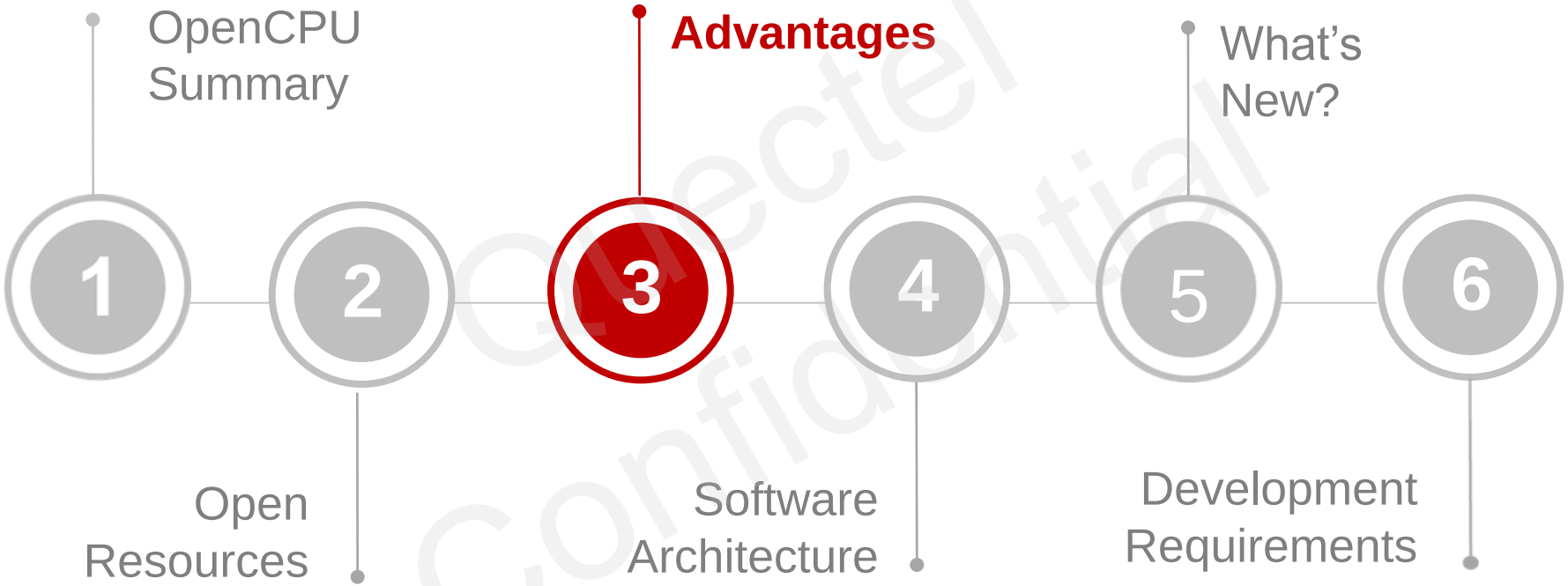
Open Resources (3)

Programmable Multiplexing Pins:

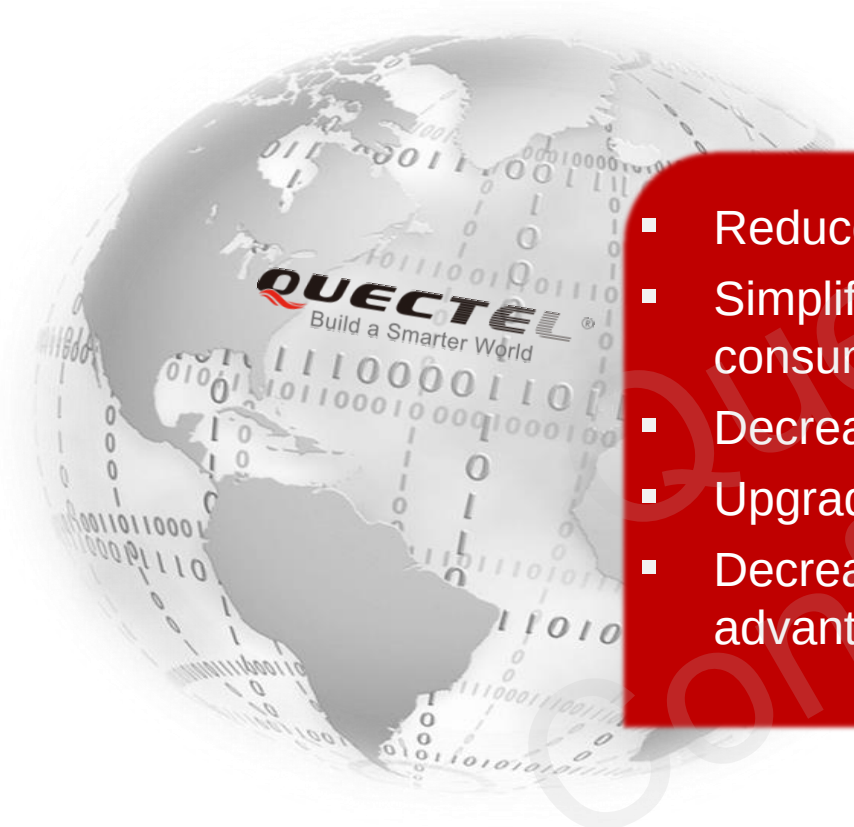
PIN No.	PIN NAME	MODE1 (default)	MODE2	MODE3	MODE4
47	PINNAME_NETLIGHT	NETLIGHT	GPIO	PWM_OUT	
37	PINNAME_DTR	DTR	GPIO	EINT	SIM_PRESENCE
35	PINNAME_RI	RI	GPIO	I2C_SCL	
36	PINNAME_DCD	DCD	GPIO	I2C_SDA	
38	PINNAME_CTS	CTS	GPIO	EINT	
39	PINNAME_RTS	RTS	GPIO		
59	PINNAME_PCM_CLK	PCM_CLK	GPIO	SPI_CS	
61	PINNAME_PCM_ SYNC	PCM_SYNC	GPIO	SPI_MISO	
62	PINNAME_PCM_IN	PCM_IN	GPIO	SPI_CLK	
60	PINNAME_PCM_OUT	PCM_OUT	GPIO	SPI_MOSI	
7	PINNAME_SD_CMD	SD_CMD	GPIO		
8	PINNAME_SD_CLK	SD_CLK	GPIO		
9	PINNAME_SD_DATA	SD_DATA	GPIO		

Note: The interrupt response time is 50ms by default, and can be re-programmed to a bigger value in OpenCPU. However, it's strongly recommended that the interrupt frequency cannot be more than 3Hz so as to ensure stable working of the module.

Contents



Advantages – Low Cost & Fast Development



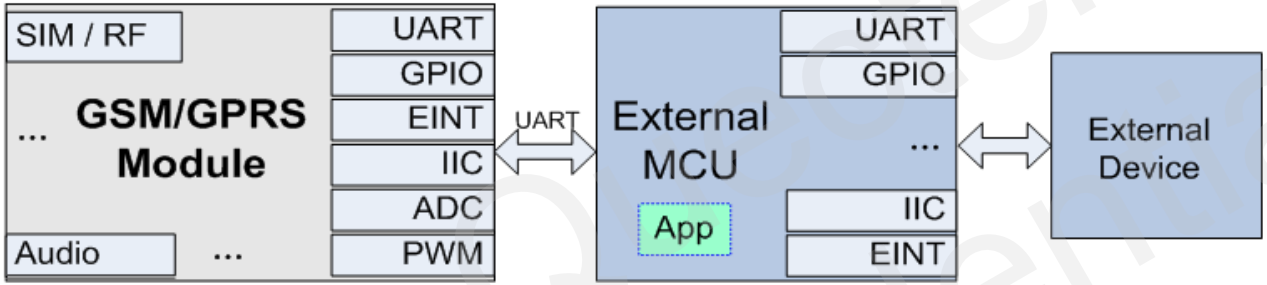
- Reduce product development time
- Simplify circuit design and reduce cost & power consumption
- Decrease product size
- Upgrade firmware remotely via OpenCPU FOTA
- Decrease the total cost and enhance the competitive advantages

Low Cost & Fast Development

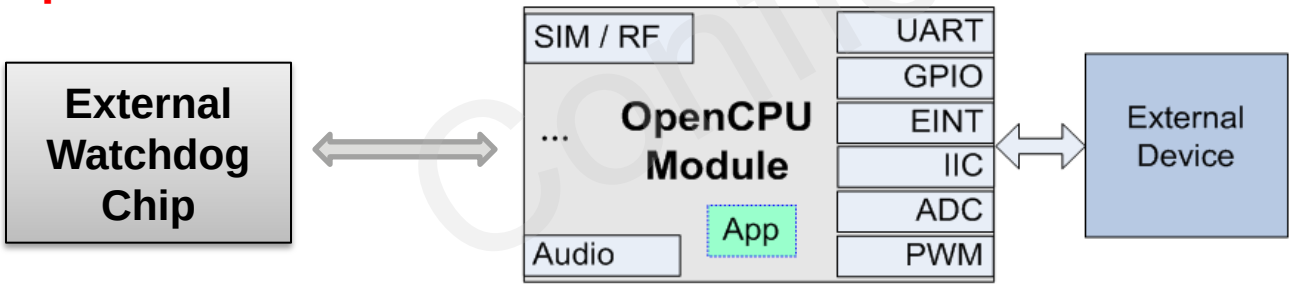
Advantages – Easy Hardware Design

Compared with traditional solutions, OpenCPU solution can make hardware design easier for developers. The following figures show the differences between them.

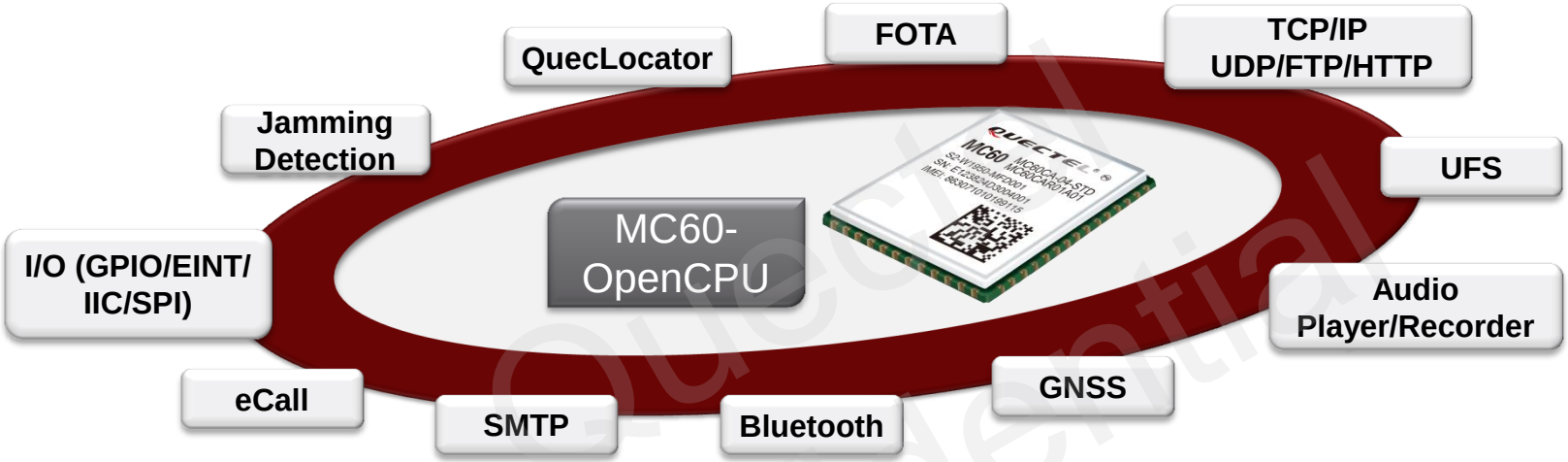
Traditional Solution



OpenCPU Solution



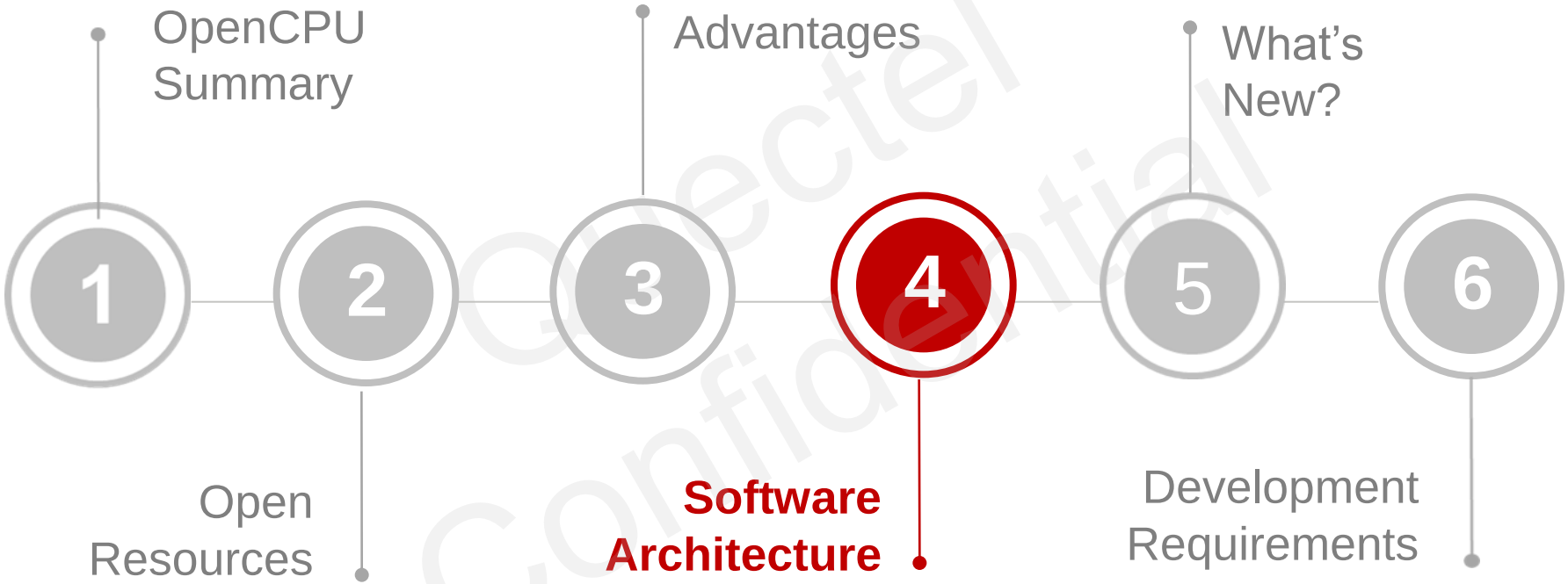
Advantages – Enhanced Technology



Quality Guarantee

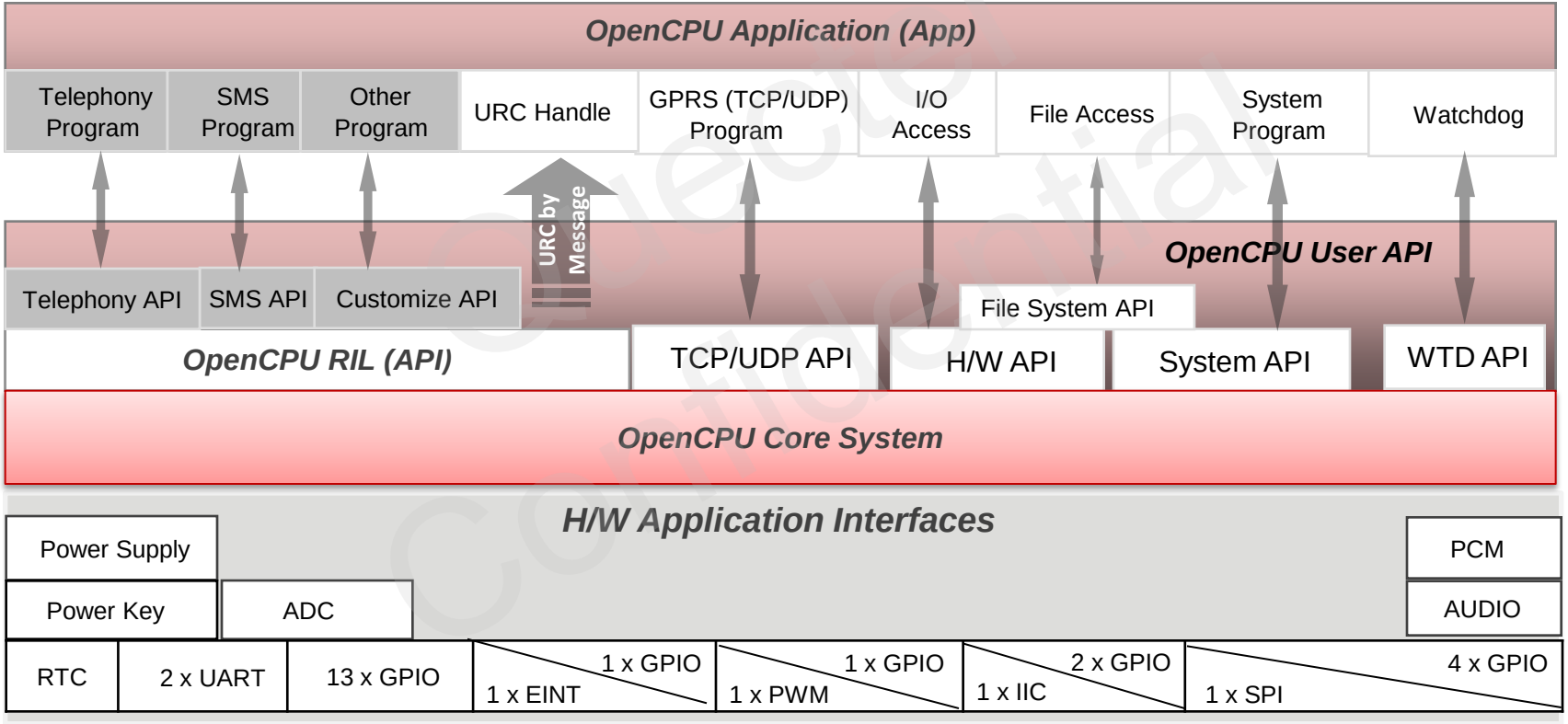
- Reliable network protocols
- Steady flash protection mechanism
- Superior audio algorithms
- Rich I/O interfaces
- Bluetooth 3.0
- Internal GNSS engine

Contents



Software Architecture (1)

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



Software Architecture (2)

Core System

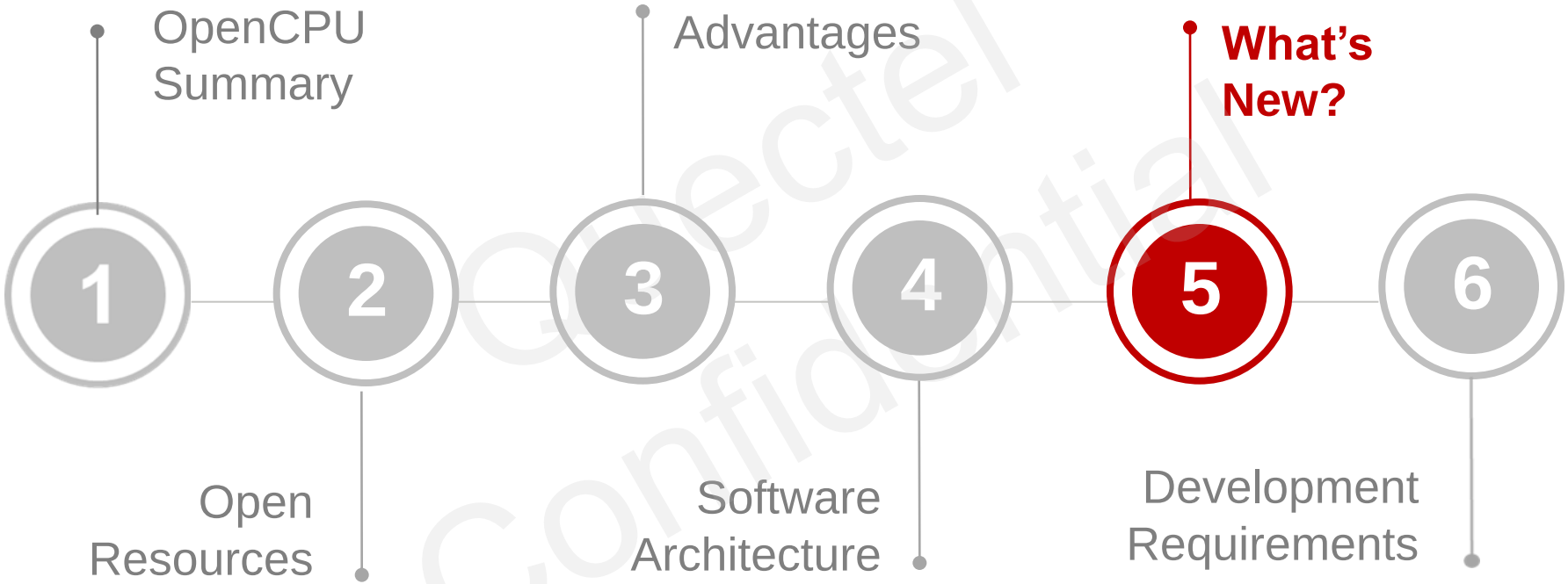
Core System is a combination of hardware and system software of GSM/GPRS module. It has a built-in ARM7EJ-S processor, and has been built over Nucleus operating system which has the characteristics of micro-kernel, real-time, multi-tasking, etc.

OpenCPU RIL

OpenCPU RIL, an open source layer, is embedded into User API layer. With OpenCPU RIL, developers can simply call API to send AT commands and immediately get the response when API returns.

By default, AT commands related to SMS and telephone are wrapped in the RIL APIs. Developers can easily develop some new API functions to implement the AT commands according to the requirements.

Contents



OpenCPU RIL Support

As compared with the previous OpenCPU solution, the new OpenCPU solution provides RIL driver in user API layer, which makes software design easier. When developers send AT commands to the core system, the responses are received and handled by RIL layer first, and then RIL returns the eventual data that application wants. Also the new OpenCPU provides the Open Source of RIL. The following C code demonstrates the differences in details.

What's New? (2)

➢ New 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? (3)

➢ 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? (4)

GCC Compiler Support

- MC60-OpenCPU supports free-of-charge GCC compiler (Sourcery CodeBench Lite for ARM EABI).
- The previous OpenCPU solution uses ARM RVCT as the compiler.

IDE Support

MC60-OpenCPU recommends two sets of tools to manage codes and compile program. One is command-line + Source Insight, and the other is Eclipse.

Easy to Configure the Initial Status of GPIO

MC60-OpenCPU provides a very simple method to configure the initial status of GPIOs when these GPIOs need to be configured during the early stage of module startup process. For instance, with OpenCPU solution, GPIOs can be configured to control the power supply of peripheral circuits.

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)
```

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

Easy to Add New Task (Thread)

Developers 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. Developers 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 file operations in the task, the task size must be set to at least 5KB. Otherwise, the stack overflow probably happens.

➢**Default Value2**: Developers don't need to specify the value.

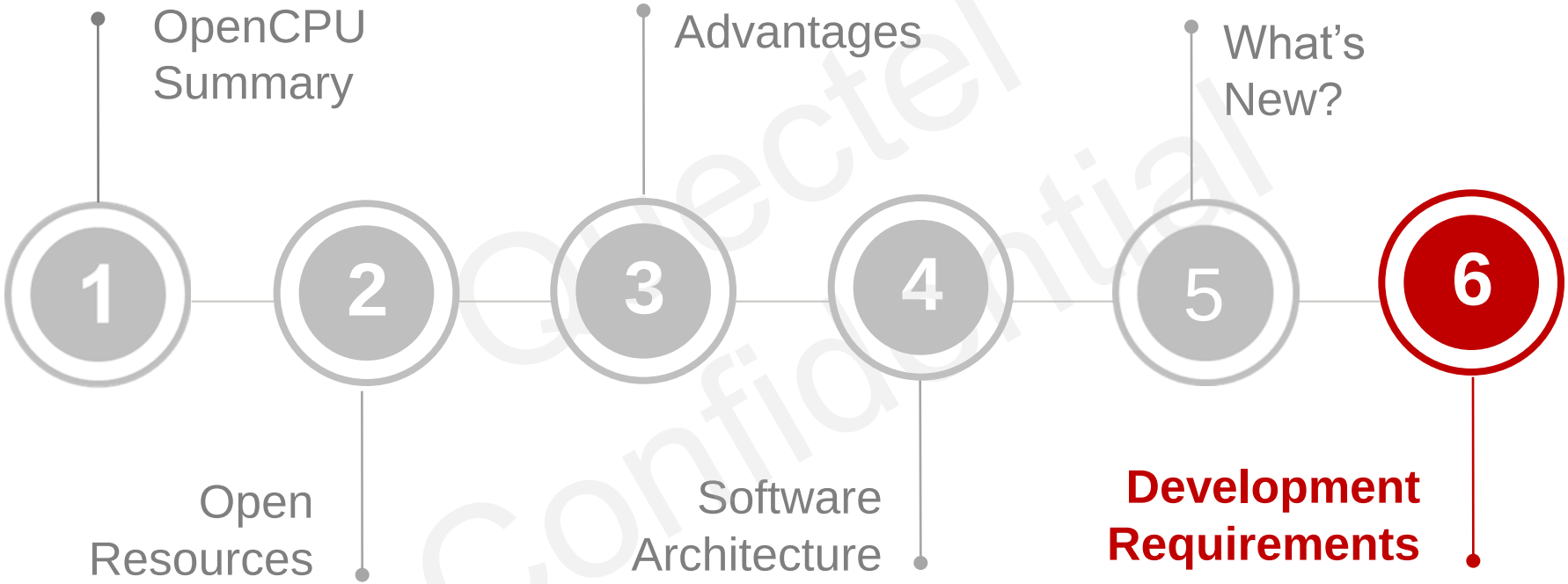
```
/*-------------------------------------------------------------------------------------------------------------------------------------
        Task Entry function      Task Id Name      Task Stack Size (Bytes)      Default Value1      Default Value2
-------------------------------------------------------------------------------------------------------------------------------------*/
TASK_ITEM (Proc_main_task,      main_task_id,      10*1024,                    DEFAULT_VALUE1,     DEFAULT_VALUE2)
```

Programmable Power Key Pin for App

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

```
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 triggered. Application must
   call Ql_LockPower () to lock power supply, Or else, 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 triggered.
   Application may do post processing before switching  off the module.
   */
};
```

Contents



Development Requirements (1)

Host System Requirements

The following host Operating Systems and architectures are supported:

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

Compiler & IDE Requirements

- GCC Compiler (Sourcery CodeBench Lite for ARM EABI)
- IDE: Eclipse (optional)

Development Requirements (2)

Programming Language Requirement

- Basic C-language programming knowledge

SDK and Other Requirements

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

Thank you!

July, 2016

www.quectel.com

Office # 501, Building 13, No. 99 Tianzhou Road, Shanghai, China 200233
Tel: +86-21-5108 6236 Fax: +86-21-5445 3668 Email: info@quectel.com