EC2x&AG35-Quecopen GPIO 配置方法

LTE Standard/Automotive 系列

版本：EC2x&AG35-Quecopen_GPIO_配置方法_V1.1

日期：2018-09-29

状态：临时文件

上海移远通信技术股份有限公司始终以为客户提供最及时、最全面的服务为宗旨。如需任何帮助，请随时联系我司上海总部，联系方式如下：

上海移远通信技术股份有限公司
上海市徐汇区虹梅路 1801 号宏业大厦 7 楼　　邮编：200233
电话：+86 21 51086236　　邮箱：info@quectel.com

或联系我司当地办事处，详情请登录：
http://www.quectel.com/cn/support/sales.htm

如需技术支持或反馈我司技术文档中的问题，可随时登陆如下网址：
http://www.quectel.com/cn/support/technical.htm
或发送邮件至：support@quectel.com

前言

上海移远通信技术股份有限公司提供该文档内容用以支持其客户的产品设计。客户须按照文档中提供的规范、参数来设计其产品。由于客户操作不当而造成的人身伤害或财产损失，本公司不承担任何责任。在未声明前，上海移远通信技术股份有限公司有权对该文档进行更新。

版权申明

文档历史

修订记录

版本	日期	作者	变更表述
1.0	2018-01-10	高飞虎	初始版本
1.1	2018-09-29	高飞虎	增加 User Space 的 GPIO/EINT API

目录

1 引言

本文档主要适用于 Global 市场，目前支持的 LTE Standard/Automotive 模块包括：

- EC2x: EC20 R2.1/EC25/EC21
- AG35

2 Bootloader 配置 GPIO

相关定义参考头文件：*ql-ol-sdk/ql-ol-bootloader/platform/mdm9607/include/platform/gpio.h*

void gpio_tlmm_config(uint32_t gpio, uint8_t func, uint8_t dir,
uint8_t pull, uint8_t drvstr, uint32_t enable);
函数功能：配置引脚功能，方向，上下拉，驱动能力，是否使能 function
参数：
gpio：芯片 GPIO 编号
func：0 表示 GPIO
dir：方向 GPIO_INPUT 或者 GPIO_OUTPUT
pull：上下拉 GPIO_NO_PULL，GPIO_PULL_DOWN，GPIO_KEEPER，GPIO_PULL_UP
enable：GPIO_ENABLE， GPIO_DISABLE

void gpio_set_val(uint32_t gpio, uint32_t val);
函数功能：配置 gpio 电平
参数：
gpio：芯片 gpio 编号
val：0, 1

uint32_t gpio_get_state(uint32_t gpio);
函数功能：获取 gpio 电平
参数：gpio：芯片 gpio 编号
返回值：0，1

3 Kernel Space 配置 GPIO

3.1. 使用内核接口

申请 GPIO
*int gpio_request(unsigned gpio, const char *label);*

标记 GPIO 的使用方向包括输入还是输出
int gpio_direction_input(unsigned gpio);
int gpio_direction_output(unsigned gpio, int value);

获得 GPIO 引脚的电平和设置 GPIO 引脚的电平(对于输出)
int gpio_get_value(unsigned gpio);
void gpio_set_value(unsigned gpio, int value);

转换 GPIO 为相应的 IRQ 值，返回中断号
int gpio_to_irq(unsigned gpio)；

配置中断
request_irq(unsigned int irq, irq_handler_t handler,
*unsigned long flags, const char *name, void *dev)；*

注销中断
*void free_irq(unsigned int irq, void *dev_id)；*

释放 GPIO
void gpio_free(unsigned gpio);

3.2. 使用 pinctrl 子系统

配置设备树：

```
ql-ol-kernel/arch/arm/boot/dts/qcom/mdm9607-pinctrl.dtsi
                gpio_default:   gpio_default{
                    mux {
                        pins = "gpioX", "gpioY";      //需要配置的管脚
```

```
                function = "gpio";        //配置为 GPIO function
            };
            config {
                pins = "gpioX", "gpioY";
                drive-strength = <2>;  //驱动能力
                bias-disable;          //上下拉可选值：bias-disable ；bias-pull-down ；
                                         bias-pull-up
                output-low;            //输出电平可选值：output-low ；  output-high
            };
        };

ql-ol-kernel/arch/arm/boot/dts/qcom/mdm9607.dtsi
        xxx{
            compatible = "xxx";              /* 匹配用户 driver */
            pinctrl-names = "default";       /*定义 pinctrl name，驱动中使用
                                               pinctrl_lookup_state()接口解析 */
            pinctrl-0 = <&gpio_xxx>;         /* 选中上面定义的 GPIO 配置 */
            status = "ok";                   /* 使能此设备节点 */
        };
```

设备树配置后，写驱动来解析使能以上配置，相关内核 API 如下：

A. 获取一个 pinctrl 句柄，参数是 dev 是包含这个 pin 的 device 结构体

```
/**
 * struct devm_pinctrl_get() - Resource managed pinctrl_get()
 * @dev: the device to obtain the handle for
 *
 * If there is a need to explicitly destroy the returned struct pinctrl,
 * devm_pinctrl_put() should be used, rather than plain pinctrl_put().
 */
struct pinctrl *devm_pinctrl_get(struct device *dev)
```

B. 获取这个 pin 对应 pin_state（引脚状态）

```
/**
 * pinctrl_lookup_state() - retrieves a state handle from a pinctrl handle
 * @p: the pinctrl handle to retrieve the state from
 * @name: the state name to retrieve
 */
struct pinctrl_state *pinctrl_lookup_state(struct pinctrl *p, const char *name)
```

C. 设置引脚为某个 state

```
/**
 * pinctrl_select_state() - select/activate/program a pinctrl state to HW
 * @p: the pinctrl handle for the device that requests configuration
 * @state: the state handle to select/activate/program
 */
int pinctrl_select_state(struct pinctrl *p, struct pinctrl_state *state)
```

4 User Space 配置 GPIO

用户层控制 GPIO，使用 QuecOpen API。

4.1. 返回值错误类型

```
enum {
        RES_OK = 0,
        RES_BAD_PARAMETER    = -1,
        RES_IO_NOT_SUPPORT   = -2,
        RES_IO_ERROR         = -3,
        RES_NOT_IMPLEMENTED = -4
    };
```

4.2. GPIO

枚举：

```
    typedef   enum {
        PINDIRECTION_IN   = 0,
        PINDIRECTION_OUT = 1
    }Enum_PinDirection;

    typedef   enum{
        PINLEVEL_LOW   = 0,
        PINLEVEL_HIGH   = 1
    }Enum_PinLevel;

    typedef   enum{
        PINPULLSEL_DISABLE      = 0,
        PINPULLSEL_PULLDOWN = 1,
        PINPULLSEL_PULLUP      = 3
    }Enum_PinPullSel;
```

int Ql_GPIO_Base_Init(Enum_PinName pinName);

函数功能：初始化 GPIO，仅初始化，不做任何设置，与 Ql_GPIO_Init 选一个使用；

参数：pinName：Pin 脚编号，见 ql_gpio.h Enum_PinName 枚举值；
返回值：错误类型，见 ***4.1 章***

```
int Ql_GPIO_Init( Enum_PinName    pinName,
                  Enum_PinDirection    dir,
                  Enum_PinLevel        level,
                  Enum_PinPullSel      pullSel
                );
```

函数功能：指定管脚初始化，该函数会配置方向，电平，上下拉；返回错误类型；
参数：
pinName：Pin 脚名字，见 ql_gpio.h Enum_PinName 枚举值；
dir：方向，枚举 Enum_PinDirection；
level：电平，枚举 Enum_PinLevel；
pullSel：上下拉，枚举 Enum_PinPullSel；
返回值：错误类型，见 ***4.1 章***

int Ql_GPIO_SetDirection(Enum_PinName pinName, Enum_PinDirection dir);
函数功能：指定管脚 direction 配置；返回错误类型；
参数：
pinName：Pin 脚名字，见 ql_gpio.h Enum_PinName 枚举值；
dir：方向，枚举 Enum_PinDirection
返回值：错误类型，见 ***4.1 章***

int Ql_GPIO_GetDirection(Enum_PinName pinName);
参数：pinName: Pin 脚名字，见 ql_gpio.h Enum_PinName 枚举值；
返回值：返回指定管脚 direction。

int Ql_GPIO_SetLevel(Enum_PinName pinName, Enum_PinLevel level);
当 GPIO 的 direction 已经为 out 时，可以直接调此接口配置输出的电平；返回错误类型；
参数：
pinName：Pin 脚名字，见 ql_gpio.h Enum_PinName 枚举值；
Level：电平，枚举 Enum_PinLevel；
返回值：错误类型，见 ***4.1 章***

int Ql_GPIO_GetLevel(Enum_PinName pinName);
参数：pinName：Pin 脚名字，见 ql_gpio.h Enum_PinName 枚举值；
返回值：返回指定管脚电平；

int Ql_GPIO_SetPullSelection(Enum_PinName pinName, Enum_PinPullSel pullSel);
函数功能：指定管脚上下拉的配置；返回错误类型；
参数：
pinName: Pin 脚名字，见 ql_gpio.h Enum_PinName 枚举值；
pullSel：上下拉，枚举 Enum_PinPullSel；

返回值：错误类型，见 ***4.1 章***

int Ql_GPIO_GetPullSelection(Enum_PinName pinName);
参数：pinName：Pin 脚名字，见 ql_gpio.h Enum_PinName 枚举值；
返回值：返回指定管脚当前上下拉状态；

int Ql_GPIO_Uninit(Enum_PinName pinName);
释放指定管脚的 GPIO 配置；返回错误类型；
参数：pinName：Pin 脚名字，见 ql_gpio.h Enum_PinName 枚举值；
返回值：错误类型，见 ***4.1 章***

参考：*ql-ol-extsdk/example/gpio*

4.3. EINT

枚举：

```
typedef enum {
    EINT_SENSE_NONE,
    EINT_SENSE_RISING,
    EINT_SENSE_FALLING,
    EINT_SENSE_BOTH
}Enum_EintType;

int Ql_EINT_Enable(   Enum_PinName    eintPinName,
Enum_EintType    eintType,
                      Ql_EINT_Callback      eint_callback
);
```

引脚中断使能，并注册用户回调函数；返回错误类型；
参数：
eintPinName：Pin 脚名字，见 ql_gpio.h Enum_PinName 枚举值；
eintType：边沿触发类型，枚举 Enum_EintType；
eint_callback：用户回调函数，中断触发回调；
返回值：返回值：错误类型，见 ***4.1 章***

int Ql_EINT_Disable(Enum_PinName eintPinName);
注销引脚中断功能；返回错误类型；
参数：eintPinName：Pin 脚名字，见 ql_gpio.h Enum_PinName 枚举值；
返回值：错误类型，见 ***4.1 章***

参考 *ql-ol-extsdk/example/eint*