

Lierda IC610 GPIO 应用开发指导

产品名称：ST-A35-IC610 工业核心板

产品型号：L-IDMIM0-AA185

版本：Rev1.0

日期：25/03/15

状态：受控版本

法律声明

若接收利尔达科技集团股份有限公司(以下称为“利尔达”)的此份文档,即表示您已经同意以下条款。若不同意以下条款,请停止使用本文档。

本文档版权归利尔达科技集团股份有限公司所有,保留任何未在本文档中明示授予的权利。文档中涉及利尔达的专有信息。未经利尔达事先书面许可,任何单位和个人不得复制、传递、分发、使用和泄漏该文档以及该文档包含的任何图片、表格、数据及其他信息。

本产品符合有关环境保护和人身安全方面的设计要求,产品的存放、使用和弃置应遵照产品手册、相关合同或者相关法律、法规的要求进行。

本公司保留在不预先通知的情况下,对此手册中描述的产品进行修改和改进的权利;同时保留随时修订或收回本手册的权利。



文件修订历史

| 文档版本 | 变更日期 | 修订人 | 审核人 | 变更内容 |
|--------|----------|-----|-----|------|
| Rev1.0 | 25-03-15 | YQA | | 初始版本 |

Lierda
利 尔 达

目录

| | |
|---------------------------------|---|
| 法律声明 | 1 |
| 文件修订历史 | 2 |
| 目录 | 3 |
| 1 引言 | 4 |
| 2 IC610 gpio 开发指导 | 5 |
| 2.1 IC610 gpio 简介 | 5 |
| 2.2 STM32CubeMX 端 gpio 操作 | 5 |
| 2.3 Gpio 上电初始化 | 6 |
| 2.4 板子端 gpio 复用查询及操作 | 7 |
| 2.5 用户层设置 gpio | 7 |
| 2.6 应用层控制 gpio | 9 |



1 引言

本文档依托 IC610 核心板，cpu 为 STM32MP255DAK3，STM32CubeMX 对应版本为 6.14.0。



2 IC610 gpio 开发指导

2.1 IC610 gpio 简介

STM32MP255DAK3 共 424 个 io，其中 BOOT 引脚、LVDS 引脚、DSI 引脚、CSI 引脚、USB 引脚、PCIE 引脚等为专用引脚不可复用为 gpio，其他引脚为普通 gpio，具有复用功能。

GPIO 共 10 组，分别为 PA0-PA15、PB0-PB15、PC0-PC13、PD0-PD15、PE0-PE15、PF0-PF15、PG0-PG15、PH2-PH15、PI0-PI11、PZ0-PZ9。

以上引脚标号为 cpu 唯一引脚号，可用于在 cubemx 中查询及复用引脚，及应用层操作。

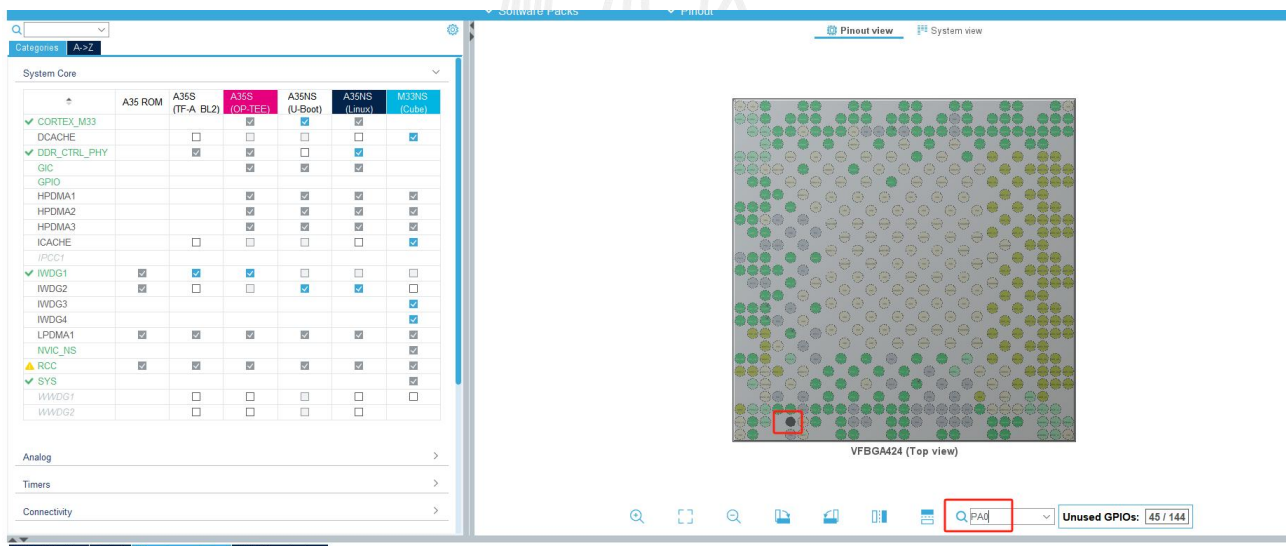
注意

PD14、PD15、PH5 为核心板内部使用，驱动及应用开发中以上三引脚不得改动，否则会导致系统无法启动。

2.2 STM32CubeMX 端 gpio 操作

使用 gpio 引脚号在 cubemx 端查询相关引脚使用情况，如查询 PA0

搜索窗口输入 PA0，搜索即出现 PA0 对应 gpio，





点击该引脚，即可查询该引脚的复用功能及当前复用状态，具体复用操作参考《Lierda+IC530&IC610+ST+cubemx 应用指导_Rev1.0》3.2 章节内容。

2.3 Gpio 上电初始化

特殊引脚可能需要上电后输出高 3.3v 或者输出低为 0V，u-boot dts 下无法初始化 gpio，可在 tf-a 代码中增加 gpio 初始化。具体操作如下：

```
tf-a-stm32mp-v2.10.5-stm32mp-r1-r0/tf-a-stm32mp-v2.10.5-stm32mp-r1$ vim
drivers/st/gpio/stm32_gpio.c

void lierda_setup_control_gpio(void)
{
    NOTICE("Set gpio for lierda\n");
    // gpiof 15 high: GPIO_BANK_F 15 1U
    //gpiod 5 low , GPIO_BANK_D 5 0U

set_gpio(GPIO_BANK_F,15,GPIO_MODE_OUTPUT,GPIO_TYPE_PUSH_PULL,GPIO_S
```

```
PEED_LOW,GPIO_PULL_UP,1U,GPIO_ALTERNATE_(0),DT_NON_SECURE);
```

```
set_gpio(GPIO_BANK_D,5,GPIO_MODE_OUTPUT,GPIO_TYPE_PUSH_PULL,GPIO_SPEED_LOW,GPIO_PULL_UP,0U,GPIO_ALTERNATE_(0),DT_NON_SECURE);

}
```

如上默认代码已经添加对 PF15 输出为高，PD5 输出为低。修改位置如下

```
void lierda_setup_control_gpio(void)
{
    NOTICE("Set gpio for lierda\n");
    // gpio 15 high: GPIO_BANK_F 15 1U
    set_gpio(GPIO_BANK_F,15,GPIO_MODE_OUTPUT,GPIO_TYPE_PUSH_PULL,GPIO_SPEED_LOW,GPIO_PULL_UP,1U,GPIO_ALTERNATE_(0),DT_NON_SECURE);
    set_gpio(GPIO_BANK_D,5,GPIO_MODE_OUTPUT,GPIO_TYPE_PUSH_PULL,GPIO_SPEED_LOW,GPIO_PULL_UP,0U,GPIO_ALTERNATE_(0),DT_NON_SECURE);
}
```

2.4 板子端 gpio 复用查询及操作

查看 PA-PK 复用情况

```
cat /sys/kernel/debug/pinctrl/soc@0:pinctrl@44240000/pinmux-pins
```

查看 pz 复用情况

```
cat /sys/kernel/debug/pinctrl/soc@0:pinctrl@46200000/pinmux-pins
```

复用状态共 3 类，具体如下：

pin 115 (PH3): UNCLAIMED : 表示该 gpio 未被复用，且未被驱动及应用层使用，用户可以使用 libgpiod 进行控制，如 gpioset gpioget

pin 116 (PH4): GPIO GPIOH:626 : 表示该 gpio 被驱动使用为 gpio，实际该引脚为 sd 卡驱动 cd 引脚 cd-gpios = <&gpioh 4 (GPIO_ACTIVE_LOW | GPIO_PULL_UP)>;

pin 122 (PH10): device 482c0000.eth1 function af10 group PH10 : 表示该引脚为 eth1 功能引脚，复用为 af10，查询该引脚为 ETH1_RGMII_TXD2

2.5 用户层设置 gpio

gpioinfo 查看全部 gpio

查看单组 gpio

```
gpioinfo -c gpiochip0
```

后面数字 0-11，分别对应 PA-PZ，具体如下

| GPIO 组 | gpiochip-number | GPIO 组 | gpiochip-number |
|--------|-----------------|--------|-----------------|
| PA | gpiochip0 | PG | gpiochip6 |
| PB | gpiochip1 | PH | gpiochip7 |
| PC | gpiochip2 | PI | gpiochip8 |
| PD | gpiochip3 | PG | gpiochip9 |
| PE | gpiochip4 | PK | gpiochip10 |
| PF | gpiochip5 | PZ | gpiochip11 |

注意

该版本 kernel gpio 驱动仅支持 gpiod，不再支持 sysfs 进行 gpio 控制。

用户层操作未被驱动使用的 gpio 拉高、拉低如：

```

line 9: "PZ9" input consumer="kernel"
gpiochip2 - 14 lines:
line 0: "PC0" input consumer="kernel"
line 1: "PC1" input consumer="kernel"
line 2: "PC2" input consumer="kernel"
line 3: "PC3" input
line 4: "PC4" input
line 5: "PC5" input
line 6: "PC6" input
line 7: "PC7" input
line 8: "PC8" input
line 9: "PC9" input
line 10: "PC10" input

```

如 PC4 未被 kernel 使用，用户层可以拉高、拉低及读取操作

拉高即输出 3.3v：

```
gpioset -c gpiochip2 4=1
```

设置后 ctrl+c 退出

或使用参数--hold-period 如下保持 20ms 自动退出。

```
gpioset --hold-period 20ms -t0 -c gpiochip2 4=1
```

拉低即输出 0v：

```
gpioset --hold-period 20ms -t0 -c gpiochip2 4=0
```

读取：

```
gpioget -c gpiochip2 4
```

inactive 表示低

active 表示高

```
line 14:      "PJ14"      input
line 15:      "PJ15"      input
root@stm32mp2:~# gpioget -c gpiochip2 4
"4"=inactive
root@stm32mp2:~# gpioset -c gpiochip2 4=1
^C
root@stm32mp2:~# gpioget -c gpiochip2 4
"4"=active
root@stm32mp2:~#
```

其他可参考官方 wiki:

https://wiki.stmicroelectronics.cn/stm32mpu/wiki/How_to_control_a_GPIO_in_userspace

ce

2.6 应用层控制 gpio

应用层控制 gpio 需要使用 libgpiod 的 api 进行操作，具体 api 可参考：

<https://libgpiod.readthedocs.io/en/latest/>

操作 PA14 引脚进行 led 控制，示例代码如下：

```
#include <errno.h>

#include <fcntl.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sys/ioctl.h>

#include <unistd.h>

#include <linux/gpio.h>

int main(int argc, char **argv)
{
    struct gpiohandle_request req;
    struct gpiohandle_data data;
    char chrdev_name[20];
    int fd, ret;
```

```

strcpy(chrdev_name, "/dev/gpiochip0");

/* Open device: gpiochip0 for GPIO bank A */
fd = open(chrdev_name, 0);
if (fd == -1) {
    ret = -errno;
    fprintf(stderr, "Failed to open %s\n", chrdev_name);

    return ret;
}

/* request GPIO line: GPIO_A_14 */
req.lineoffsets[0] = 14;
req.flags = GPIOHANDLE_REQUEST_OUTPUT;
memcpy(req.default_values, &data, sizeof(req.default_values));
strcpy(req.consumer_label, "led_gpio_a_14");
req.lines = 1;

ret = ioctl(fd, GPIO_GET_LINEHANDLE_IOCTL, &req);
if (ret == -1) {
    ret = -errno;
    fprintf(stderr, "Failed to issue GET LINEHANDLE IOCTL (%d)\n",
        ret);
}
if (close(fd) == -1)
    perror("Failed to close GPIO character device file");

```

```
/* Start led blinking */
while(1) {

    data.values[0] = !data.values[0];
    ret = ioctl(req.fd, GPIOHANDLE_SET_LINE_VALUES_IOCTL, &data);
    if (ret == -1) {
        ret = -errno;
        fprintf(stderr, "Failed to issue %s (%d)\n",
            "GPIOHANDLE_SET_LINE_VALUES_IOCTL", ret);
    }
    sleep(1);
}

/* release line */
ret = close(req.fd);
if (ret == -1) {
    perror("Failed to close GPIO LINEHANDLE device file");
    ret = -errno;
}
return ret;
}
```