D1-H Tina Linux 扩展 IO 开发指南

版本号: 1.0
发布日期: 2021.04.12

版本历史

版本号	日期	制/修订人	内容描述
1.0	2021.04.12	AWA1611	新建文档

目　录

1 前言

1.1 文档简介

本文介绍 Tina 平台 D1-H 方案扩展 IO 驱动的使用方法，方便扩展 IO 驱动维护和应用开发。

本文扩展 IO 采用的模块是 PCF8574，设备通过 I2C 与 SOC 通信。

1.2 目标读者

扩展 IO 驱动和应用开发人员。

1.3 适用范围

表 1-1：适用产品列表

产品名称	内核版本	驱动文件
D1-H	Linux-5.4	gpio-pcf857x.c

2 模块介绍

2.1 模块功能介绍

当主控芯片 SOC 的 IO 口数量不够使用的时候，就可以使用到扩展 IO 了。

本文介绍的扩展 IO 型号为 PCF8574，这是一款 I2C 并行口扩展电路。扩展 IO 这边有两根 I2C 与一根中断接线（INT）与主控 SOC 相连，以 I2C 读写寄存器的方式来操作 IO 的状态变化。

PCF8574 能够支持以下属性：

- I2C 控制电路状态。
- IO 支持输入输出功能。
- IO 支持输出高低电平。
- IO 支持开漏中断输出。
- 实现扩展 8 个 IO 口。
- 具有大电流驱动能力。

2.2 软件术语

表 2-1：软件术语列表

术语	解释说明
I2C	二线制同步串行总线
扩展 IO	扩展 IO 芯片或驱动

2.3 源码结构

本模块接触与标准 Linux gpio 子系统，代码路径为：

```
tina/lichee/linux-5.4/drivers/gpio/gpio-pcf8574.c
```

3 模块配置介绍

3.1 kernel menuconfig 配置说明

在 tina 根目录下，执行 make kernel_menuconfig，配置路径如下：

```
Device Drivers
    GPIO Support >
        I2C GPIO expanders >
            PCF857x, PCA{85,96}7x, and MAX732[89] I2C GPIO expanders
```

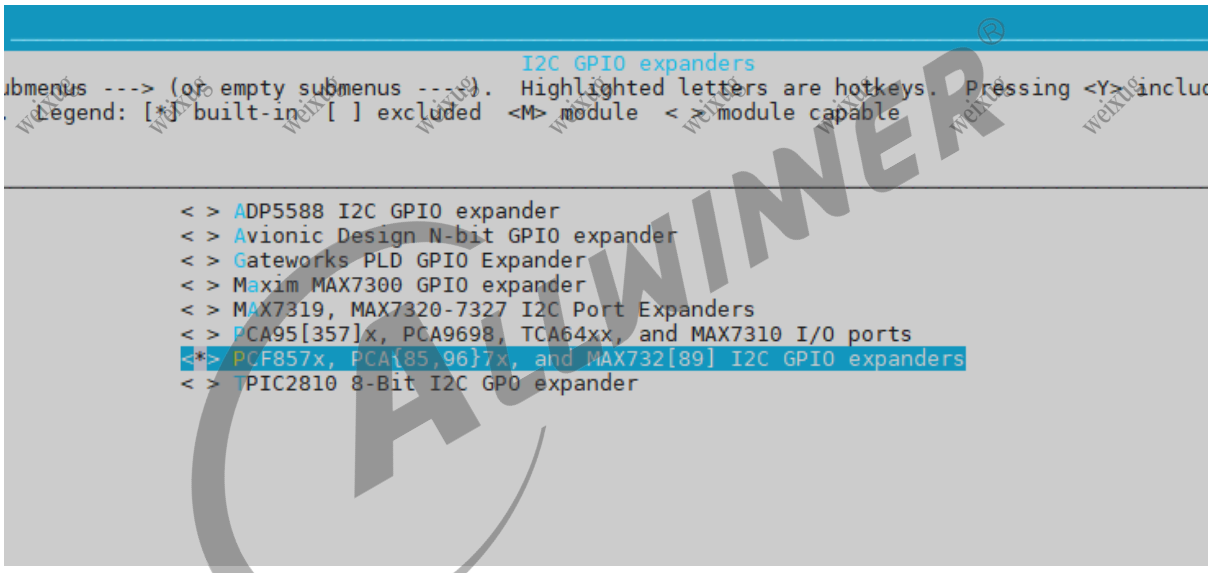

图 3-1: 扩展 IO 配置界面图

说明

这是 *PCF* 系列芯片的驱动，能够支持 *PCF/PCA/MAX732* 等系列扩展 *IO* 芯片，本文只针对 *PCF8574* 作介绍，文中的 “扩展 *IO*” 都特指 *PCF8574*。

3.2 Device Tree 配置说明

方案的 dts 所在位置可以通过命令 cdts 跳转过去，详细路径为：

```
tina/lichee/linux-5.4/arch/riscv/boot/dts/sunxi/sun20iw1p1.dtsi
```

板级的 dts 所在位置可以通过命令 cconfigs 转过去，详细路径为：

tina/device/config/chips/d1-h/configs/nezha/linux/board.dts

3.2.1 board.dts 板级配置

由于扩展 IO 是不属于 SOC 内部模块，而是一个 I2C 外设，因此 dts 建议是配置在板级 dts 里，配置如下所示：

```
&twi2 {
        clock-frequency = <400000>;
        pinctrl-0 = <&twi2_pins_a>;
        pinctrl-1 = <&twi2_pins_b>;
        pinctrl-names = "default", "sleep";
        status = "okay";

        /* pcf8574-usage:
         * only use gpio0~7, 0 means PP0.
         * pin set:
         * gpios = <&pcf8574 0 GPIO_ACTIVE_LOW>;
         * interrupt set:
         * interrupt-parent = <&pcf8574>;
         * interrupts = <0 IRQ_TYPE_EDGE_FALLING>;
         */
        pcf8574: gpio@38 {
                compatible = "nxp,pcf8574";
                reg = <0x38>;
                gpio_base = <2020>;
                gpio-controller;
                #gpio-cells = <2>;
                interrupt-controller;
                #interrupt-cells = <2>;
                interrupt-parent = <&pio>;
                interrupts = <PB 2 IRQ_TYPE_EDGE_FALLING>;
                status = "okay";
        };
};
```

扩展 IO 硬件上是连接 I2C2 总线上，扩展 IO 是 I2C2 总线下的一个 I2C 设，因此 dts 里需要写在 I2C2 的配置里。I2C 这里命名为 twi，详细配置方法查看《D1-H_Linux_TWI_ 开发指南》。

扩展 IO 配置含义如下：

- compatible：匹配扩展 IO 驱动。
- reg：扩展 IO 的 i2c 设备地址。
- gpio_base：扩展出来的 8 个 IO 的 gpio 初始编号，可用于调试和编码时识别对应的 GPIO。
- gpio-controller：表明自己的身份为 gpio 控制器。
- #gpio-cells：第一个 cell 表示 gpio 号，第二个 cell 表示 gpio 默认电平。
- interrupt-controller：表明自己的身份为中断控制器。
- #interrupt-cells：第一个 cell 表示中断号，第二个 cell 表示中断触发方式。
- interrupt-parent：中断源是 SOC 的 GPIO。

- interrupts：扩展 IO 的中断（INT）引脚设置，用于反馈中断给 SOC。
- status：状态，是否加载该设备。

4 使用示例

4.1 使用扩展 IO 的 pin 脚 dts 配置示例

当扩展 IO 驱动成功加载起来后，一共有 8 个引脚可以使用，这里命名为 PP0~PP7。

其他模块如果要用到扩展 IO 的 GPIO 口，配置方法如下：

```
gpios= <&pcf8574 1 GPIO_ACTIVE_LOW>;
```

这里代表的是使用第二个 IO 口，PP1。由于扩展 IO 的 gpio_base 设置为 2020，因此这个口的 gpio 号为 2021。当设备配置了这个 IO 口后，代码里就可以使用标准的 Linux 的 GPIO 子系统的接口来获取调用改 IO 口，详情使用方法请查看《D1-H_Linux_GPIO_ 开发指南》。

4.2 使用扩展 IO 作为按键引脚

首先，软件上需要配置 GPIO 中断方式的按键驱动，配置方法请查看《D1-H_Tina_Linux_Key_ 快速配置 _ 使用指南》。

然后在 board.dts 配置 gpio-keys:

```
&soc {
    ...
    gpiokey {
        device_type ="gpiokey";
        compatible ="gpio-keys";
        ok_key {
            device_type ="ok_key";
            label ="ok_key";
            gpios = <&pcf8574 1 GPIO_ACTIVE_LOW>;
            linux,input-type ="1>";
            linux,code = <0x1c>;
            wakeup-source = <0x1>;
        };
    };
    ...
};
```

当 GPIO 中断按键加载起来后，在小机端串口使用 getevent 验证该功能是否正常。在 D1-H 电路板，对应硬件原理图找到 PP1 引出来的排针接口，使用杜邦线连接，然后短接地，查看 getevent 是否会打印出对应的键值以及 input 相关的信息。

使用这种方法也可以验证扩展 IO 中断功能是可以正常使用的。

4.3 使用 gpio sysfs 节点

首先需要确保 gpio sysfs 节点已经配置上，在 tina 根目录运行 make kernel_menuconfig，查看下面该项：

```
Device Drivers
    GPIO Support >
        /sys/class/gpio/... (sysfs interface)
```

需要将上述的节点选成 [*] 的状态。

在系统正常加载 gpio sysfs 功能后，

```
cd sys/class/
ls
```

能够看到一个 gpio 的目录，进入后看到 gpio0 和 gpio2020 两个目录，前者是 SOC 的 gpio 信息，后者是扩展 IO 的 gpio 信息，进入后者目录可以查看一些扩展 IO 的信息。

```
root@TinaLinux:/sys/class/gpio# ls
export       gpiochip0    gpiochip2020  unexport
root@TinaLinux:/sys/class/gpio# cd gpiochip2020/
root@TinaLinux:/sys/devices/platform/soc@3000000/2502800.twi/i2c-2/2-0038/gpio/gpiochip2020# ls
base                 ngpio                uevent
device               power                waiting_for_supplier
label                subsystem
root@TinaLinux:/sys/devices/platform/soc@3000000/2502800.twi/i2c-2/2-0038/gpio/gpiochip2020# cat label
pcf8574
root@TinaLinux:/sys/devices/platform/soc@3000000/2502800.twi/i2c-2/2-0038/gpio/gpiochip2020# cat base
2020
root@TinaLinux:/sys/devices/platform/soc@3000000/2502800.twi/i2c-2/2-0038/gpio/gpiochip2020# cat ngpio
8
root@TinaLinux:/sys/devices/platform/soc@3000000/2502800.twi/i2c-2/2-0038/gpio/gpiochip2020#
```

图 4-1: 扩展 IO gpio 信息

label 是扩展 IO 的标签，base 是 dts 中配置的 gpio 的起始号，ngpio 是该设备一共有多少 gpio。

单 gpio 调试，可以通过 export 节点来调试。在/sys/class/gpio 目录下，如下图的操作方法：

```
root@TinaLinux:/sys/class/gpio# ls
export       gpiochip0    gpiochip2020  unexport
root@TinaLinux:/sys/class/gpio# echo 2021 > export
root@TinaLinux:/sys/class/gpio# ls
export       gpio2021     gpiochip0     gpiochip2020  unexport
root@TinaLinux:/sys/class/gpio# cd gpio2021
root@TinaLinux:/sys/devices/platform/soc@3000000/2502800.twi/i2c-2/2-0038/gpiochip1/gpio/gpio2021# ls
active_low           edge                 uevent
device               power                value
direction            subsystem            waiting_for_supplier
root@TinaLinux:/sys/devices/platform/soc@3000000/2502800.twi/i2c-2/2-0038/gpiochip1/gpio/gpio2021#
```

图 4-2: 单 gpio 调试

这里想要调试 gpio 号为 2021 的 gpio 引脚，2021 则是 PP1。进入到 gpio2021 目录后，可以对 IO 口的一些输入输出功能进行调试。

- active_low：查看该 gpio 是否为 active_low，1 代表是，0 代表否。active_low 代表低电平有效。
- edeg：中断触发方式，没有的话则是 none。
- value：IO 口当前的电平，如果是 output 模式的话，则可以设置 IO 口的逻辑电平。
- direciton：输入输出功能节点。

说明

***active_low* 与 *value* 两个值是息息相关的，当 *gpio* 是低电平有效时，*value* 为 *1* 则表示当前是有效电平（物理电平为低电平）；*value* 为 *0* 则表示当前是无效电平（物理电平为高电平）。相反的，当高电平有效时，有效电平的物理电平是高电平。**

IO 口输入输出调试方法：

```
echo "in" > direction   //控制该gpio变为输入功能
cat value       //获得当前gpio的电平值
echo "out" > direction  //控制gpio变为输出功能
echo 1 > active_low     //控制此时gpio低电平有效
echo 1 > value          //在output模式下，输出低电平
echo 0 > value          //在output模式下，输出高电平
echo 0 > active_low     //取消低电平有效
echo 1 > value          //在output模式下，输出高电平
echo 0 > value          //在output模式下，输出低电平
```

4.4 使用驱动自带的调试节点

修改扩展 IO 驱动源文件，新增宏定义，如下所示：

```
AwExdroid88:~/workspace/tina/lichee/linux-5.4/drivers/gpio$ git df .
diff --git a/drivers/gpio/gpio-pcf857x.c b/drivers/gpio/gpio-pcf857x.c
index 46d41c9..e81802f 100644
--- a/drivers/gpio/gpio-pcf857x.c
+++ b/drivers/gpio/gpio-pcf857x.c
@@ -19,6 +19,7 @@
 #include <linux/spinlock.h>
 #include <linux/of_gpio.h>

+#define IO_EXPAND_DEBUG 1
 static const struct i2c_device_id pcf857x_id[] = {
        { "pcf8574", 8 },
```

测试一个引脚的输入输出功能，第二个引脚的输入输出以及中断功能。

```
1 pr_info("------gpio:%d state test------\n", gpio);
2 ret = gpio_request(gpio, "gpio_test");
3 pr_info("gpio_request return %d\n", ret);
4
5 ret = gpio_direction_output(gpio, 0);
6 pr_info("gpio_direction_output return %d\n", ret);
7
8 ret = gpio_get_value_cansleep(gpio);
9 pr_info("gpio_get_value return %d\n", ret);
```

```
10
11  ret = gpio_direction_input(gpio);
12  pr_info("gpio_direction_input return %d\n", ret);
13
14
15  pr_info("------gpio:%d state test------\n", gpio2);
16  ret = gpio_request(gpio2, "gpio2_test");
17  pr_info("gpio2_request return %d\n", ret);
18
19  ret = gpio_direction_output(gpio2, 1);
20  pr_info("gpio2_direction_output return %d\n", ret);
21
22  ret = gpio_get_value_cansleep(gpio2);
23  pr_info("gpio2_get_value return %d\n", ret);
24
25  pr_info("------gpio:%d irq tese-------\n", gpio);
26  irq = gpio_to_irq(gpio);
27  pr_info("gpio to irq:%d\n", irq);
28
29  ret = request_irq(irq, gpio_test_handler, flags, dev_name(dev), dev);
30  pr_info("request irq return%d\n", ret);
31
32  disable_irq(irq);
33  pr_info("disable irq\n");
34  enable_irq(irq);
35  pr_info("enable irq\n");
```

进入小机端后，用 find 命令寻找 gpio_test 节点，然后 cat 该节点即可，简易合成一条命令如下所示：

```
cat `find -name gpio_test`
```

著作权声明

商标声明

免责声明