

# S720

## 应用配置指南（Linux）

智能模组

版本 1.0

日期 2023-03-31



## 版权声明

版权所有 © 深圳市有方科技股份有限公司 2023。深圳市有方科技股份有限公司保留所有权利。  
未经深圳市有方科技股份有限公司书面同意,任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部,并不得以任何形式传播。

neoway有方是深圳市有方科技股份有限公司所有商标。

本文档中出现的其他商标,由商标所有者所有。

## 说明

本文档对应产品为 **S720** 模组。

本文档的使用对象为系统工程师,开发工程师及测试工程师。

本文档为客户产品设计提供支持,客户须按照本文中的规范和参数进行产品设计和调试。如因客户操作不当造成的人身伤害和财产损失,有方概不承担责任。

由于产品版本升级或其它原因,本文档内容会在不预先通知的情况下进行必要的更新。

除非另有约定,本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

深圳市有方科技股份有限公司为客户提供全方位的技术支持,任何垂询请直接联系您的客户经理或发送邮件至以下邮箱:

Sales@neoway.com

Support@neoway.com

公司网址: <http://www.neoway.com>

# 目 录

关于本文档 .....	v
范围 .....	v
读者对象 .....	v
修订记录 .....	v
符号约定 .....	v
1 S720 介绍 .....	错误!未定义书签。
2 模组开机 .....	7
3 应用配置 .....	8
3.1 驱动添加 .....	8
3.1.1 编写驱动文件 .....	8
3.1.2 添加并修改 Kconfig 文件和 Makefile 文件 .....	8
3.1.3 检查配置 .....	9
3.2 GPIO 配置 (Linux) .....	10
3.2.1 GPIO 介绍 .....	10
3.2.2 Pinmap 配置 .....	10
3.3 I2C 配置 .....	11
3.3.1 I2C 介绍 .....	11
3.3.2 Pinmap 配置 .....	11
3.3.3 DTS 配置 .....	12
3.3.4 各路 I2C 配置 .....	13
3.4 SPI 配置 .....	16
3.4.1 SPI 介绍 .....	16
3.4.2 Pinmap 配置 .....	16
3.4.3 DTS 配置 .....	17
3.4.4 各路 SPI 配置 .....	19
3.5 UART 配置 .....	21
3.5.1 UART 介绍 .....	21
3.5.2 UART 使用情况 .....	22
3.5.3 AP UART 控制器配置 .....	22
3.5.4 dts 配置 .....	23
3.5.5 CM4 UART 控制器配置 .....	24
3.6 电池曲线配置 .....	25
3.6.1 介绍 .....	25
3.6.2 设备树配置 .....	25
3.6.3 电池曲线配置方法 .....	27

3.6.4 电池充电曲线配置 .....	28
----------------------	----

# 关于本文档

## 范围

本文档对应产品为 **S720** 模组。




## 读者对象

本文档的使用对象为系统工程师，开发工程师及测试工程师。

## 修订记录

版本	日期	变更	作者
1.0	2023-12	初始版本	Li Xiao Yun

## 符号约定

符号	含义
	危险或警告，用户必须遵从的规则，否则会造成模组或客户设备不可逆的故障损坏，甚至可能造成人员身体伤害。
	注意，警示用户使用模组时应该特别注意的地方，如不遵从，模组或客户设备可能出现故障。
	说明或提示，提供模组使用的意见或建议。

# 1 概述

S720 是一款基于紫光展锐平台的 LTE 无线通信智能模组，支持制式包含 FDD-LTE、TDD-LTE、WCDMA、GSM 四模通信，同时支持 GNSS、Bluetooth4.2、Wi-Fi2.4G。支持显示屏、摄像头、SD、MIC、Speaker、耳机、USB2.0 等。S720 支持 Android10 系统，适用于智能 POS、智能网关、视频监控、行车记录仪、DVR、车载支付设备、执法设备、智能手持设备、智能穿戴、售卖机、物流柜等终端，能够满足用户在工业、车载和消费类应用中对高速率和多媒体功能的需求。

本文主要介绍如何对各模组管脚功能进行配置、以及介绍电池曲线配置方法。

## 2 模组开机



1680234558344

在按键中，POWER ON 按键为开机/关机按键，S720 开发套件上电后，长按 POWER ON 键约 3 秒即可使模组开机。

## 3 应用配置

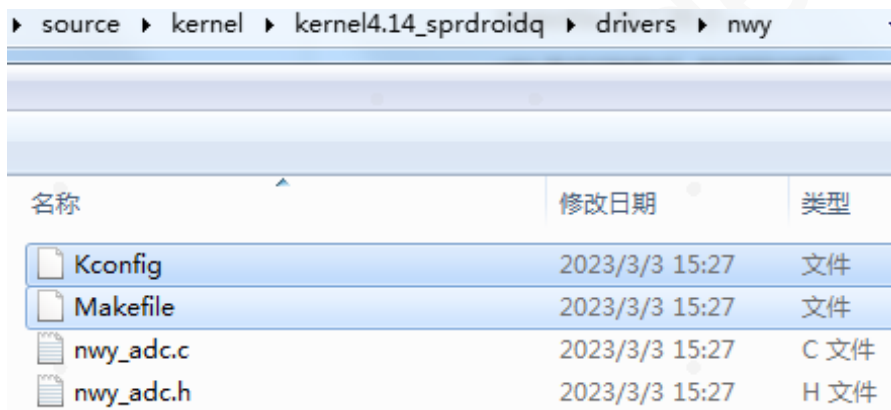
### 3.1 驱动添加

Linux 系统使用 defconfig 文件来配置驱动，本文以 adc 驱动为例，介绍如何添加一个新的驱动。

#### 3.1.1 编写驱动文件

编写相应的驱动文件，并放到对应的目录。对应的目录一般根据驱动的类型来区分，尽量跟同一类的驱动放到一起。

如需新增一个 nwy\_adc 文件，则需要新增对应目录  
source\kernel\kernel4.14\_sprdroidq\drivers\nwy



名称	修改日期	类型
Kconfig	2023/3/3 15:27	文件
Makefile	2023/3/3 15:27	文件
nwy_adc.c	2023/3/3 15:27	C 文件
nwy_adc.h	2023/3/3 15:27	H 文件

添加文件之后，还需要修改 source\kernel\kernel4.14\_sprdroidq\drivers 下的 Kconfig 和 Makefile

Kconfig 文件如下所示：

```
source "drivers/trusty/Kconfig" **\#neoway add for adc** **source "drivers/nwy/  
Kconfig" ** endmenu
```

Makefile 如下所示：

```
obj-$(CONFIG_PARPORT) += parport/ obj-$(CONFIG_NVM) += lightnvm/ obj-y += ba  
se/ block/ misc/ mfd/ nfc/ **nwy/** obj-$(CONFIG_LIBNVDIMM) += nvdimmm/
```

#### 3.1.2 添加并修改 Kconfig 文件和 Makefile 文件

1. 添加 source\kernel\kernel4.14\_sprdroidq\drivers\nwy 目录的 Kconfig 和 Makefile 文件，如下图所示：



source > kernel > kernel4.14_sprdroidq > drivers > nwy		
名称	修改日期	类型
Kconfig	2023/3/3 15:27	文件
Makefile	2023/3/3 15:27	文件
nwy_adc.c	2023/3/3 15:27	C 文件
nwy_adc.h	2023/3/3 15:27	H 文件

2. 修改 Makefile 文件 根据括号里面的宏的值决定是编译进内核，编译成模组，还是不参与编译。

```
obj-$(CONFIG_NWY_ADC_TEMP) += nwy_adc.o
```

3. 修改 Kconfig 文件

可以参照已有宏的格式来添加自己的控制宏。

**注意:** 这个里面的宏比 makefile 里面的少了一个 CONFIG\_，tristate 代表驱动可以配置为 y、m、n 三种，分别代表编译进 kernel，编译为模组和不参与编译

```
config NWY_ADC_TEMP tristate "ADC read driver" help if need adc function, say y.
```

4. 修改 config 文件

**注意:** 对于 S720\_L 项目，需要修改

source\kernel\kernel4.14\_sprdroidq\arch\arm\configs\S720\_L\_sprd\_sharkle\_defconfig: 添加  
**CONFIG\_NWY\_ADC\_TEMP=y** |

### 3.1.3 检查配置

检查最终编译出来的配置文件

```
build-unisoc-wayland\work\sl8541e_emmc_marlin2-unisoc-linux-gnueabi\linux-unisoc-4.14\4.14-r0\linux-unisoc-4.14-4.14\config,
```

确认是否包含了添加的 CONFIG 项。

按本文所示操作，.config 文件中会包含 **CONFIG\_NWY\_ADC\_TEMP=y**

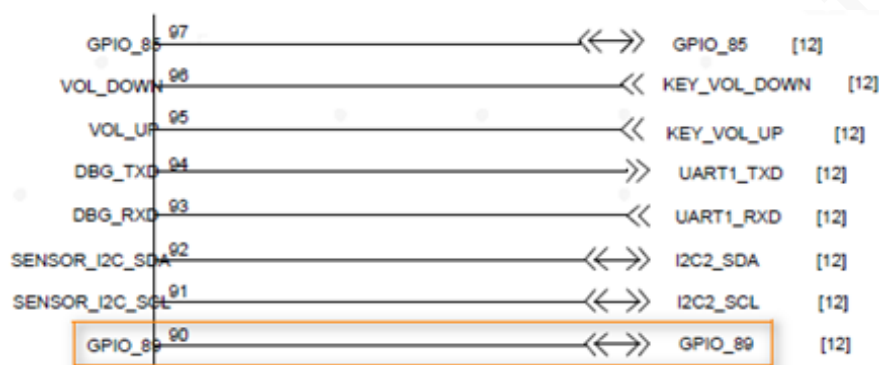
## 3.2 GPIO 配置 (Linux)

### 3.2.1 GPIO 介绍

GPIO 是通用输入输出端口的简称，简单来说就是可控制的引脚，芯片的 GPIO 引脚与外部设备连接起来，从而实现与外部设备进行通讯、控制以及数据采集的功能。

### 3.2.2 Pinmap 配置

在配置 pinmap 之前首先要通过原理图确认模组 PIN 脚对应的 GPIO 序号，本文以模组的 PIN90 脚为例，如图，对应的 gpio 为 gpio89。



如果要将该管脚配置成 GPIO 功能，首先要查找手册《S720-管脚定义文档-V1.1.xlsx》。

基带芯片	功能复用			
芯片管脚名称	Function0	Function1	Function2	Function3
RTCK_LTE	DRTCK_LTE	DRTCK_TWG	DBG_BUS31(G0)	GPIO89

由图中可以看出，gpio89 对应的 PinName 是 RTCK\_LTE，可以用作 4 种功能，Function 3 为 gpio 功能，我们要使用的是 gpio 功能，在 pinmap 文件中根据 Pin Name 找到对应的配置项进行修改即可。

Pinmap 文件位于 bspboot15\_sprdroidq720\_L-sl8541e\_1h10\_32b.c

```
{REG_PIN_RTCK_LTE,                                BITS_PIN_AF(3)},  
{REG_MISC_PIN_RTCK_LTE,                            BITS_PIN_DS(1)|BIT_PIN_NULL|BIT_PIN_WPD  
|BIT_PIN_SLP_AP|BIT_PIN_SLP_NUL|BIT_PIN_SLP_OE},
```

如上所示，找到 RTCK\_LTE 之后，将 BITS\_PIN\_AF 配置为 3，此时 pin 脚就可以当作普通 GPIO 功能使用。注意：BITS\_PIN\_AF 配置值为 0-3，分别对应 pin 的 Function 0-3。

dts 配置如下所示：

```
extcon_gpio: extcon-gpio {  
    compatible = "linux,extcon-usb-gpio";  
    vbus-gpio = <&pmic_eic 0 GPIO_ACTIVE_HIGH>;  
    id-gpio = <&ap_gpio 126 0>;  
    otg5v-gpio = <&ap_gpio 89 0>;  
};
```

## 3.3 I2C 配置

### 3.3.1 I2C 介绍

I2C 通讯协议(Inter-Integrated Circuit)是由 Philips 公司开发的，由于它引脚少，硬件实现简单，可扩展性强，不需要 USART、CAN 等通讯协议的外部收发设备，现在被广泛地使用在系统内多个集成电路(IC)间的通讯。

S720\_L 项目可供客户自由使用的 I2C 有 3 组：i2c2、i2c3、i2c4。

i2c 编号	模组 pin	gpio 编号	备注
I2c-2	pin-91、pin-92	gpio127、gpio128	默认 Sensor I2C 使用
I2c-3	pin-47、pin-48	gpio146、gpio147	默认触摸屏 I2C 使用
I2c-4	pin-168、pin-167	gpio154、gpio155	默认 SIM2 功能

对于 I2C 的配置，需要在 pinmap 和设备树中配置，具体配置方法请参考下面步骤。

### 3.3.2 Pinmap 配置

S720\_L 采用 pinmap 配置管脚功能，如果使用 I2C 功能需要对照下表，将对应 pin 配置为 I2C 功能。 pinmap 文件路径：

```
source\bsp\u-boot15_sprdroidq\board\spreadtrum\S720_L\pinmap-sl8541e_1h10_32  
b.c
```

基带芯片	功能复用			
芯片管脚名称	Function0	Function1	Function2	Function3
SCL2	SCL2	-	-	GPIO127
SDA2	SDA2	-	-	GPIO128
SIMDAT2	SIMDAT2	SDA4	SE_GPIO12	GPIO155
SIMCLK2	SIMCLK2	SCL4	SE_GPIO11	GPIO154
SCL3	SCL3	-	EXT_XTL_EN0	GPIO146
SDA3	SDA3	-	-	GPIO147

### 3.3.3 DTS 配置

1. 确认 aliases 节点 请在 aliases 节点下添加对于 i2c 节点的配置，如下以 i2c3 为例：

配置路径：

source\kernel\kernel4.14\_sprdroidq\arch\arm\boot\dts\S720\_L\_sharkle.dtsi

2. 添加如下代码

```
aliases {
    ...
+ i2c3 = &i2c3;
    ...
};
```

3. 确认 I2C 节点

在 soc 节点下添加 I2C 配置，以 i2c3 为例： 位置： source\_sprdroidq720\_L\_sharkle.dtsi 一般情况下，I2C 的节点都是配置完成的，这里只需要检查确认即可。

```
i2c3: i2c@70800000 {
    compatible = "sprd,sharkl3-i2c";
    reg = <0x70800000 0x100>; /*i2c 寄存器地址*/
    interrupts = <GIC_SPI 14 IRQ_TYPE_LEVEL_HIGH>;
    clock-names = "enable","i2c", "source"; /*i2c 时钟配置*/
    clock-frequency = <400000>; /*i2c 主模式时钟频率*/
    #address-cells = <1>;
    #size-cells = <0>;
    status = "disabled"; /*注意在使用中需要使能：okay*/
};
```

4. 添加 I2C 设备节点 i2c 节点配置过后，需要添加对应的 i2c 设备节点，以触摸屏节点举例。打开文件： source\_sprdroidq720\_L\_sl8541e-1h10-gofu.dts 配置如下代码：

```
&i2c3 {
    status = "okay"; /*打开 i2c 使能: okay*/
    goodix@14 {
        compatible = "goodix,gt1x";
        reg = <0x14>; /*从机 7 位地址*/
        goodix,irq-gpio = <&ap_gpio 144 GPIO_ACTIVE_HIGH>;
        goodix,reset-gpio = <&ap_gpio 145 GPIO_ACTIVE_HIGH>;
    };
};
```

### 3.3.4 各路 I2C 配置

#### I2C-2 配置

1. 配置 pinmap: 查看功能表, 功能 0 是 i2c-2 功能。 打开文件:

```
source\bsp\u-boot15_sprdroidq\board\spreadtrum\S720_L\pinmap-sl8541e_1h10_3
2b.c
```

配置如下代码:

```
// i2c-2, scl
{REG_PIN_SCL2,      BITS_PIN_AF(0)},
{REG_MISC_PIN_SCL2,
BITS_PIN_DS(1)|BIT_PIN_WPUS|BIT_PIN_WPU|BIT_PIN_SLP_CM4|BIT_PIN_SLP_WPU|BIT_P
IN_SLP_Z},
// i2c-2, sda
{REG_PIN_SDA2,      BITS_PIN_AF(0)},
{REG_MISC_PIN_SDA2,
BITS_PIN_DS(1)|BIT_PIN_WPUS|BIT_PIN_WPU|BIT_PIN_SLP_CM4|BIT_PIN_SLP_WPU|BIT_P
IN_SLP_Z},
```

注意: BITS\_PIN\_AF 配置值为 0-3, 分别对应 pin 的 Function 1-4。

2. 确认 aliases 节点 打开文件:

```
source\kernel\kernel4.14_sprdroidq\arch\arm\boot\dts\S720_L_sharkle.dtsi
```

确认如下代码:

```
aliases {
...
    i2c2 = &i2c2;
...
};
```

```
i2c2: i2c@70700000 {
    compatible = "sprd,sharkle-i2c";
    reg = <0x70700000 0x100>;
    interrupts = <GIC_SPI 13 IRQ_TYPE_LEVEL_HIGH>;
    clock-frequency = <100000>;
    #address-cells = <1>;
    #size-cells = <0>;
    status = "disabled";
};
```

## I2C-3 配置

1. 配置 pinmap: 查看功能表, 功能 0 是 I2C-3。 打开文件:

```
source\bsp\u-boot15_sprdroidq\board\spreadtrum\S720_L\pinmap-sl8541e_1h10_32
b.c
```

配置如下代码:

```
// i2c-3, scl
{REG_PIN_SCL3,      BITS_PIN_AF(0)},
{REG_MISC_PIN_SCL3,
BITS_PIN_DS(3)|BIT_PIN_WPUS|BIT_PIN_WPU|BIT_PIN_SLP_AP|BIT_PIN_SLP_WPU|BIT_PI
N_SLP_Z},
// i2c-3, sda
{REG_PIN_SDA3,      BITS_PIN_AF(0)},
{REG_MISC_PIN_SDA3,
BITS_PIN_DS(3)|BIT_PIN_WPUS|BIT_PIN_WPU|BIT_PIN_SLP_AP|BIT_PIN_SLP_WPU|BIT_PI
N_SLP_Z},
```

2. 确认 aliases 节点: 打开文件:

```
source\kernel\kernel14.14_sprdroidq\arch\arm\boot\dtb\S720_L_sharkle.dtsi
```

确认如下代码:

```
aliases {
...
    i2c3 = &i2c3;
...
};

i2c3: i2c@70800000 {
    compatible = "sprd,sharkle-i2c";
    reg = <0x70800000 0x100>;
```

```
interrupts = <GIC_SPI 14 IRQ_TYPE_LEVEL_HIGH>;
clock-frequency = <400000>;
#address-cells = <1>;
#size-cells = <0>;
status = "disabled";
};
```

## I2C-4 配置

1. 配置 pinmap: 查看功能表, 功能 1 是 I2C-4。打开文件:

```
source\bsp\u-boot15_sprdroidq\board\spreadtrum\S720_L\pinmap-sl8541e_1h10_3
2b.c
```

配置如下代码:

```
// I2C-4, scl
{REG_PIN_SIMCLK2,      BITS_PIN_AF(1)},
{REG_MISC_PIN_SIMCLK2,
BITS_PIN_DS(1)|BIT_PIN_WPUS|BIT_PIN_WPU|BIT_PIN_SLP_AP|BIT_PIN_SLP_WPU|BIT_PI
N_SLP_Z},
// I2C-4, sda
{REG_PIN_SIMDAT2,      BITS_PIN_AF(1)},
{REG_MISC_PIN_SIMDAT2,
BITS_PIN_DS(1)|BIT_PIN_WPUS|BIT_PIN_WPU|BIT_PIN_SLP_AP|BIT_PIN_SLP_WPU|BIT_PI
N_SLP_Z},
```

2. 确认 aliases 节点:

打开文件:

```
source\kernel\kernel4.14_sprdroidq\arch\arm\boot\dts\S720_L_sharkle.dtsi
```

确认如下代码:

```
aliases {
...
    i2c4 = &i2c4;
...
};

i2c4: i2c@70900000 {
    compatible = "sprd,sharkle-i2c";
    reg = <0x70900000 0x100>;
    interrupts = <GIC_SPI 15 IRQ_TYPE_LEVEL_HIGH>;
```

```
clock-frequency = <100000>;  
#address-cells = <1>;  
#size-cells = <0>;  
status = "disabled";  
};
```

## 3.4 SPI 配置

### 3.4.1 SPI 介绍

SPI 是串行外设接口(Serial Peripheral Interface)的缩写。是 Motorola 公司推出的一种同步串行接口技术，是一种高速的，全双工，同步的通信总线。

S720\_L 项目可供客户自由使用的 SPI 有 2 组，SPI0、SPI2。

SPI 编号	功能描述	模组 pin	gpio 编号	备注	默认功能
SPI0	SPI0_DO	pin-112	pin-113	GPIO-91	主机输出
	SPI0_DI	pin-123	GPIO-92	GPIO-93	主机接收
	SPI0_CS	pin-124	GPIO-90	GPIO-93	片选
	SPI0_CLK				时钟
SPI2	SPI2_DO	pin-108	pin-109	GPIO-54	主机输出
	SPI2_DI	pin-52	pin-110	GPIO-53	主机接收
	SPI2_CS			GPIO-52	片选
	SPI2_CLK			GPIO-55	时钟

对于 SPI 的配置，引脚功能配置在 pinmap 文件中，spi 节点和设备节点放在 Kernel 的 dts 中配置，具体配置方法请参照如下步骤。

### 3.4.2 Pinmap 配置

S720\_L 采用 pinmap 配置管脚功能。如果使用 SPI 功能需要对照下表，将对应 pin 配置为 SPI 功能。

pinmap 文件路径：

```
source\\bsp\\u-boot15_sprdroidq\\board\\spreadtrum\\S720_L\\pinmap-sl8541e_1h  
10_32b.c
```



芯片管脚名称	Function0	Function1	Function2	Function3
NF_DATA_2	NF_DATA_2	NF_DATA_2_T	-	GPIO143
SPI0_CLK	SPI0_CLK	-	EXTINT8	GPIO93
SPI0_CSN	SPI0_CSN	-	EXTINT5	GPIO90
SPI0_DI	SPI0_DI	-	EXTINT7	GPIO92
SPI0_DO	SPI0_DO	-	EXTINT6	GPIO91
SPI2_CSN	SPI2_CSN	-	CM4_GPIO5	GPIO52
SPI2_DI	SPI2_DI	-	CM4_GPIO1	GPIO54
SPI2_DO	SPI2_DO	-	CM4_GPIO0	GPIO53
SPI2_CLK	SPI2_CLK	-	CM4_GPIO2	GPIO55

以 spi0 及设备节点的配置方法举例如下。

### 3.4.3 DTS 配置

修改 pinmap 配置

举例，配置 spi0 的修改：

```
// spi0, cs
{REG_PIN_SPI0_CSN,          BITS_PIN_AF(0)},
{REG_MISC_PIN_SPI0_CSN,
BITS_PIN_DS(1)|BIT_PIN_NULL|BIT_PIN_WPU|BIT_PIN_SLP_AP|BIT_PIN_SLP_NUL|BIT_PIN_SLP_OE},
// spi0, DO
{REG_PIN_SPI0_DO,          BITS_PIN_AF(0)},
{REG_MISC_PIN_SPI0_DO,
BITS_PIN_DS(1)|BIT_PIN_NULL|BIT_PIN_WPD|BIT_PIN_SLP_AP|BIT_PIN_SLP_WPD|BIT_PIN_SLP_Z},
// spi0, DI
{REG_PIN_SPI0_DI,          BITS_PIN_AF(0)},
{REG_MISC_PIN_SPI0_DI,
BITS_PIN_DS(1)|BIT_PIN_NULL|BIT_PIN_WPD|BIT_PIN_SLP_AP|BIT_PIN_SLP_WPD|BIT_PIN_SLP_Z},
// spi0, CLK
{REG_PIN_SPI0_CLK,          BITS_PIN_AF(0)},
{REG_MISC_PIN_SPI0_CLK,
BITS_PIN_DS(1)|BIT_PIN_NULL|BIT_PIN_WPD|BIT_PIN_SLP_AP|BIT_PIN_SLP_WPD|BIT_PIN_SLP_Z},
```

确认 aliases 节点

请在 `aliases` 节点下添加对于 SPI 节点的配置，如下以 `spi0` 为例：

打开文件：

```
source\\kernel\\kernel4.14_sprdroidq\\arch\\arm\\boot\\dts\\S720_L_sharkle.dtsi
```

确认以下代码：

```
aliases {  
...  
spi0 = &spi0;  
...  
};
```

确认 SPI 节点

在 `soc` 节点下添加 SPI 配置，以 `SPI0` 为例：

打开文件：

```
source\\kernel\\kernel4.14_sprdroidq\\arch\\arm\\boot\\dts\\S720_L_sharkle.dtsi
```

并确认以下代码：

```
spi0: spi@70a00000 {  
    compatible = "sprd,sc9860-spi",  
        "sprd,sharkle-spi";  
    reg = <0x70a00000 0x1000>;  
    interrupts = <GIC_SPI 7 IRQ_TYPE_LEVEL_HIGH>;  
    #address-cells = <1>;  
    #size-cells = <0>;  
    status = "disabled"; /*在使用过程中请使能节点：okay*/  
};
```

一般情况下 SPI 的节点都是配置完成的，这里只需要检查确认即可。

添加 SPI 设备节点

SPI 节点配置过后，需要添加对应的 SPI 设备节点，以 `fpga` 节点举例。

打开文件：

```
source\\kernel\\kernel4.14_sprdroidq\\arch\\arm\\boot\\dts\\S720_L_sl8541e-1h10-gofu.dts
```

参考配置以下代码：

```
&spi0 {
    status = "okay"; /*打开 spi 使能: okay*/
    fpga: fpga {
        compatible = "lattice-spi";
        spi-max-frequency = <48000000>; /*spi 时钟频率*/
        crstn-gpio = <&ap_gpio 133 0>;
        rstn-gpio = <&ap_gpio 132 0>;
        reg = <0>;
    };
};
```

### 3.4.4 各路 SPI 配置

#### SPI-0 配置

##### 1. 配置 pinmap

查看功能表，功能 0 是 SPI-0 功能。

打开文件：

```
source\\bsp\\u-boot15_sprdroidq\\board\\spreadtrum\\S720_L\\pinmap-sl8541e_1h
10_32b.c
```

配置以下代码：

```
// spi0 cs
{REG_PIN_SPI0_CSN,          BITS_PIN_AF(0)},
{REG_MISC_PIN_SPI0_CSN,
BITS_PIN_DS(1)|BIT_PIN_NULL|BIT_PIN_WPU|BIT_PIN_SLP_AP|BIT_PIN_SLP_NUL|BIT_PI
N_SLP_OE},
// spi0 DO
{REG_PIN_SPI0_DO,          BITS_PIN_AF(0)},
{REG_MISC_PIN_SPI0_DO,
BITS_PIN_DS(1)|BIT_PIN_NULL|BIT_PIN_WPD|BIT_PIN_SLP_AP|BIT_PIN_SLP_WPD|BIT_PI
N_SLP_Z},
// spi0 DI
{REG_PIN_SPI0_DI,          BITS_PIN_AF(0)},
{REG_MISC_PIN_SPI0_DI,
BITS_PIN_DS(1)|BIT_PIN_NULL|BIT_PIN_WPD|BIT_PIN_SLP_AP|BIT_PIN_SLP_WPD|BIT_PI
N_SLP_Z},
// spi0 CLK
```

```
{REG_PIN_SPI0_CLK,          BITS_PIN_AF(0)},  
{REG_MISC_PIN_SPI0_CLK,  
BITS_PIN_DS(1)|BIT_PIN_NULL|BIT_PIN_WPD|BIT_PIN_SLP_AP|BIT_PIN_SLP_WPD|BIT_PIN_SLP_Z},
```

## 2. 确认 aliases 节点

打开文件:

```
source\\kernel\\kernel4.14_sprdroidq\\arch\\arm\\boot\\dts\\S720_L_sharkle.dtsi
```

确认以下代码:

```
aliases {  
...  
spi0 = &spi0;  
...  
};  
  
spi0: spi@70a00000 {  
    compatible = "sprd,sc9860-spi",  
        "sprd,sharkle-spi";  
    reg = <0x70a00000 0x1000>;  
    interrupts = <GIC_SPI 7 IRQ_TYPE_LEVEL_HIGH>;  
    #address-cells = <1>;  
    #size-cells = <0>;  
    status = "disabled";  
};
```

## SPI-2 配置

### 1. 配置 pinmap

查看功能表, 功能 0 是 SPI-2。

打开文件:

```
source\\bsp\\u-boot15_sprdroidq\\board\\spreadtrum\\S720_L\\pinmap-sl8541e_1h10_32b.c
```

配置以下代码:

```
{REG_PIN_SPI2_CSN,          BITS_PIN_AF(0)},  
{REG_MISC_PIN_SPI2_CSN,    BITS_PIN_DS(1)|BIT_PIN_NULL|BIT_PIN_WPU
```

```
|BIT_PIN_SLP_AP|BIT_PIN_SLP_NUL|BIT_PIN_SLP_OE},
{REG_PIN_SPI2_DO,                      BITS_PIN_AF(0)},
BITS_PIN_DS(1)|BIT_PIN_NULL|BIT_PIN_WPD|BIT_PIN_SLP_AP|BIT_PIN_SLP_WPD|BIT_PI
N_SLP_Z},
{REG_PIN_SPI2_DI,                      BITS_PIN_AF(0)},
{REG_MISC_PIN_SPI2_DI,                BITS_PIN_DS(1)|BIT_PIN_NULL|BIT_PIN_WPD
|BIT_PIN_SLP_AP|BIT_PIN_SLP_WPD|BIT_PIN_SLP_Z},
{REG_PIN_SPI2_CLK,                    BITS_PIN_AF(0)},
{REG_MISC_PIN_SPI2_CLK,                BITS_PIN_DS(1)|BIT_PIN_NULL|BIT_PIN_WPD
|BIT_PIN_SLP_AP|BIT_PIN_SLP_WPD|BIT_PIN_SLP_Z},
```

## 2. 确认 aliases 节点

打开文件：

```
source\\kernel\\kernel4.14_sprdroidq\\arch\\arm\\boot\\dts\\S720_L_sharkle.dtsi
```

确认以下代码：

```
aliases {
...
spi2 = \&spi2;
...
};

spi2: spi@70c00000 {
    compatible = "sprd,sc9860-spi",
        "sprd,sharkle-spi";
    reg = <0x70c00000 0x1000>;
    interrupts = <GIC_SPI 9 IRQ_TYPE_LEVEL_HIGH>;
    #address-cells = <1>;
    #size-cells = <0>;
    status = "disabled";
};
```

## 3.5 UART 配置

### 3.5.1 UART 介绍

通用异步收发传输器 (Universal Asynchronous Receiver/Transmitter)，通常称作 UART。它将要传输的资料在串行通信和并行通信之间加以转换。作为把并行输入信号转成串行输出信号的芯片，UART 通常被集成于其他通讯接口的连线上。

### 3.5.2 UART 使用情况

当前平台引出的 uart 管脚分为三组：

- DBG\_TXD/ DBG\_RXD，对应 pin93 和 pin94，
- UART0\_TXD/ UART0\_RXD，对应 pin34 和 pin35，
- UART2\_RXD/ UART2\_TXD，对应 pin153 和 pin154。

由于芯片内部只有两个 UART 控制器：AP UART 控制器和 CM4 UART 控制器，所以同时只能有两组 UART 起作用。

默认使用 DBG\_TXD/ DBG\_RXD 和 UART2\_RXD/ UART2\_TXD，管脚 DBG\_TXD/ DBG\_RXD 连接到 AP UART 控制器，管脚 UART2\_RXD/ UART2\_TXD 连接到 CM4 UART 控制器。

### 3.5.3 AP UART 控制器配置

#### I. pinmap 配置

如图，本配置需要使用模组的 Pin93 和 Pin94 引脚：



根据手册《S720-管脚定义文档-V1.1》找到它们对应的 GPIO 管脚：

芯片管脚名称	Function0	Function1	Function2	Function3
U1RXD	U1RXD	PPS(G1)	-	GPIO71
U1TXD	U1TXD	-	-	GPIO70

如上图所示，PIN93 和 PIN94 脚对应的 GPIO 是 GPIO70 和 GPIO71。

在 pinmap 文件中根据芯片管脚名称找到对应的配置项进行修改，Pinmap 文件位于：

```
source\\bsp\\u-boot15_sprdroidq\\board\\spreadtrum\\S720_L\\pinmap-sl8541e_1h10_32b.c
```

```
{REG_PIN_U1TXD, BITS_PIN_AF(0)},  
{REG_MISC_PIN_U1TXD, BITS_PIN_DS(1)|BIT_PIN_NULL|BIT_PIN_NUL|BIT_PIN_SLP_AP  
|BIT_PIN_SLP_NUL|BIT_PIN_SLP_OE},//BB_U1TXD  
{REG_PIN_U1RXD, BITS_PIN_AF(0)},
```

```
{REG_MISC_PIN_U1RXD,    BITS_PIN_DS(1)|BIT_PIN_NULL|BIT_PIN_WPU|BIT_PIN_SLP_AP  
|BIT_PIN_SLP_WPU|BIT_PIN_SLP_IE},//BB_U1RXD
```

找到对应的配置之后，对照文档《S720-管脚定义文档-V1.1》，可以看出，function 0 为 U1TXD/U1RXD 功能，将 BITS\_PIN\_AF 配置为 0，此时 pin 脚即作为 uart 功能使用。

### 3.5.4 dts 配置

pinmap 配置完成之后，下一步就要做 dts 配置。

#### 1. aliases 添加 打开文件

```
source\\kernel\\kernel4.14_sprdroidq\\arch\\arm\\boot\\dts\\ S720_L_sl8541e-1  
h10-gofu.dts
```

```
aliases {  
    serial0 = &uart0;  
    serial1 = &uart1;  
};
```

#### 2. 添加 uart 配置

打开文件

```
source\\kernel\\kernel4.14_sprdroidq\\arch\\arm\\boot\\dts\\S720_L_sharkle.dtsi
```

增加 uart1 驱动配置，包括 reg、interrupt、clock 这些基本都是配置好的，在使用的时候打开即可；

```
uart1: serial@70100000 {  
    compatible = "sprd,sc9836-uart";  
    reg = <0x70100000 0x100>;  
    interrupts = <GIC_SPI 3 IRQ_TYPE_LEVEL_HIGH>;  
    status = "disabled";  
};
```

当需要使用 uart 节点的时候，将 uart 配置 status 设为 ok。

在配置完成后，dev/下会生成一个名为 ttyS1 的节点，uart 的配置就完成了。



该串口默认用作内核的 console log 打印。

### 3.5.5 CM4 UART 控制器配置

#### pinmap 配置

如图，我们所要使用的脚 U2TXDG0 和 U2TXDG0；



据手册《S720-管脚定义文档-V1.1》，对应在 pinmap 中根据 PinName 找到对应的配置项进行修改；

芯片管脚名称	Function0	Function1	Function2	Function3
U2RXD	U2RXD	SE_GPIO5	DBG_BUS15(G1)	GPIO73
U2TXD	U2TXD	SE_GPIO4	DBG_BUS14(G1)	GPIO72

Pinmap 文件位于

```
source\\bsp\\u-boot15_sprdroidq\\board\\spreadtrum\\S720_L\\pinmap-sl8541e_1h10_32b.c
```

```
{REG_PIN_U2TXD, BITS_PIN_AF(0)}, ,  
{REG_MISC_PIN_U2TXD, BITS_PIN_DS(1)|BIT_PIN_NULL|BIT_PIN_NUL|BIT_PIN_SLP_AP  
|BIT_PIN_SLP_NUL|BIT_PIN_SLP_OE},  
{REG_PIN_U2RXD, BITS_PIN_AF(0)},  
{REG_MISC_PIN_U2RXD, BITS_PIN_DS(1)|BIT_PIN_NULL|BIT_PIN_WPU|BIT_PIN_SLP_AP  
|BIT_PIN_SLP_WPU|BIT_PIN_SLP_IE},
```

找到对应的配置之后，对照文档《S720-管脚定义文档-V1.1》，可以看出，function 0 为 U2TXD/U2RXD 功能，将 BITS\_PIN\_AF 配置为 0，此时 pin 脚即作为 uart 功能使用。

#### dtb 配置

pinmap 配置完成之后，下一步就要做 dtb 配置。

##### 1. aliases 添加

添加在配置目录位于 source\kernel\kernel4.14\_sprdroidq\arch\arm\boot\dtb\ S720\_L\_sl8541e-1h10-gofu.dts

```
S720_L_sl8541e-1h10-gofu.dts
```

```
aliases {  
    serial0 = &uart0; /*这里对应是 uart 控制器名字*/
```



```
serial1 = &uart1;  
};
```

## 2. 添加 uart 配置

位于 source\kernel\kernel4.14\_sprdroidq\arch\arm\boot\dtbs\S720\_L\_sharkle.dtsi

增加 uart0 驱动配置, 包括 reg、interrupt、clock 这些基本都是配置好的, 在使用的时候打开即可;

```
uart0: serial@508d0000 {  
    compatible = "sprd,sc9836-uart-ex";  
    reg = <0x508d0000 0x100>;  
    interrupts = <GIC_SPI 1 IRQ_TYPE_LEVEL_HIGH>;  
    sprd,aon-apb = <&aon_apb_regs>;  
    status = "disabled";  
};
```

当需要使用 uart0 节点的时候, 将 uart0 配置 status 设为 ok。

在配置完成后, dev/下会生成一个名为 ttySE0 的节点, uart0 的配置就完成了。

## 3.6 电池曲线配置

### 3.6.1 介绍

电池是智能模组中重要的供电部件, 其充电控制和放电时的电量显示需要依赖电池的充放电曲线数据。

本文以 JBT-D009 型号的电池添加到 S720\_L 项目为例, 介绍如何在 Linux kernel 中添加自定义的电池曲线文件。

### 3.6.2 设备树配置

充电芯片和电量计的配置在

source\kernel\kernel4.14\_sprdroidq\arch\arm\boot\dtbs\S720\_L\_sl8541e-1h10-gofu.dts, charger-manager 驱动为电池计量和充电的中间驱动, 具有电池参数获取, 充电参数获取和与用户层通信等功能。

```
charger-manager {  
    compatible = "charger-manager";  
    cm-name = "battery";  
    cm-poll-mode = <2>;  
    cm-poll-interval = <15000>;  
    cm-battery-stat = <2>;
```

```
cm-fullbatt-vchkdirp-ms = <30000>;
cm-fullbatt-vchkdirp-volt = <60000>;
cm-fullbatt-voltage = <4300000>;
cm-fullbatt-current = <120000>;
cm-fullbatt-capacity = <100>;

cm-num-chargers = <1>;
cm-chargers = "sc2721_charger";
cm-fuel-gauge = "sc27xx-fgu";

/* in deci centigrade */
cm-battery-cold = <200>;
cm-battery-cold-in-minus;
cm-battery-hot = <800>;
cm-battery-temp-diff = <100>;
```

其中：

- cm-chargers = “sc2721\_charger”是充电芯片的配置，示例中配置为sc2721\_charger 内部充电芯片。
- cm-fuel-gauge = “sc27xx-fgu”是电量计的配置，示例中配置为sc27xx-fgu 电量计。

对应 name 配置位于对应 charger 芯片中，搜索 sc27xx\_fgu\_desc 关键字：

```
2:
3: static const struct power_supply_desc sc27xx_fgu_desc = {
4:     .name           = "sc27xx-fgu",
5:     .type           = POWER_SUPPLY_TYPE_UNKNOWN,
6:     .properties      = sc27xx_fgu_props,
7:     .num_properties  = ARRAY_SIZE(sc27xx_fgu_props),
8:     .get_property    = sc27xx_fgu_get_property,
9:     .set_property    = sc27xx_fgu_set_property,
10:    .external_power_changed = sc27xx_fgu_external_power_changed,
11:    .property_is_writeable = sc27xx_fgu_property_is_writeable,
12: };
13:
```

论电量计的计量或充电都需要依赖电池参数的配置，S720\_L 中电量计和充电的电池配置均为monitored-battery = <&bat>，不建议修改。

```
&pmic_fgu {
    monitored-battery = <&bat>;
    sprd,calib-resistance-real = <20000>;
    sprd,calib-resistance-spec = <20000>;
};
```

```
&pmic_charger {
    status = "okay";
    phys = <&hsphy>;
    monitored-battery = <&bat>;
};
```

3.6.3 电池曲线配置方法

在配置电池对应的电池之前，会从电池厂商拿到一组电池曲线和电池的说明文档，以《A.11.002.008.004 JBT-D009 中英文电池规格书 B 版.pdf》为例。

电池组基本参数

电池规格书中一般提供电池组基本参数，包括标准容量、充电截至电压、最大充电电流、工作温度等内容，下图为 JBT-D009 示例：

2. Battery Pack Specification 电池组参数

NO	Items	Criteria	Remarks
2.1	Nominal Capacity 标称容量	5500mAh	0.2C discharge 0.2C 放电 cut-off voltage 3.0V 截止电压 3.0V
	Minimum Capacity 最小容量	5500mAh	
2.2	Nominal Voltage 标称电压	3.80V	
2.3	Shipment voltage 出货电压	≥3.75V	Within 10 days from Factory 在出厂 10 天内
2.4	Internal Impedance 内阻	≤180mΩ	
2.5	Charge cut-off voltage 充电截止电压	4.35V	
2.6	Standard charging Method 标准充电方式	0.2C CC to 4.35V, CV to 0.02C	
2.7	Max. Charge Current 最大充电电流	4A	@15~45℃
		0.2C	@0~15℃
2.8	Standard discharge Method 标准放电方式	0.2C CC to 3.0V	
2.9	Max. discharge current 最大放电电流	4.0A	@10~60℃
		0.2C	@-20~10℃
2.10	Discharge cut-off voltage 放电截止电压	3.0V	
2.11	Operating Temperature 工作温度	0~+45℃	Charging 充电
		-20~+60℃	Discharging 放电
2.12	Storage Temperature 贮存温度 (30%SOC)	-20℃~+50℃	Less than 1 month 小于一个月
			Recovery

为 S720\_L 电池相关配置：

```
bat: battery {
    compatible = "simple-battery";
```

```
charge-full-design-microamp-hours = <2780000>;
charge-term-current-microamp = <120000>;
constant_charge_voltage_max_microvolt = <4350000>;
factory-internal-resistance-micro-ohms = <320000>;
voltage-min-design-microvolt = <3450000>;
ocv-capacity-celsius = <20>;
ocv-capacity-table-0 = <4330000 100>, <4249000 95>, <4189000 90>,
    <4133000 85>, <4081000 80>, <4034000 75>,
    <3991000 70>, <3953000 65>, <3910000 60>,
    <3866000 55>, <3836000 50>, <3813000 45>,
    <3795000 40>, <3782000 35>, <3774000 30>,
    <3765000 25>, <3750000 20>, <3726000 15>,
    <3687000 10>, <3658000 5>, <3400000 0>;
voltage-temp-table = <1095000 800>, <986000 850>, <878000 900>,
    <775000 950>, <678000 1000>, <590000 1050>,
    <510000 1100>, <440000 1150>, <378000 1200>,
    <324000 1250>, <278000 1300>, <238000 1350>,
    <204000 1400>, <175000 1450>, <150000 1500>,
    <129000 1550>, <111000 1600>, <96000 1650>;
charge-sdp-current-microamp = <500000 500000>;
charge-dcp-current-microamp = <1150000 3000000>;
charge-cdp-current-microamp = <1150000 1150000>;
charge-unknown-current-microamp = <500000 500000>;
};
```

参数说明如下：

- charge-full-design-microamp-hours：标准容量 uah。
- charge-term-current-microamp：截至电流 ua（可根据情况配置）。
- constant\_charge\_voltage\_max\_microvolt：最大电压。
- factory-internal-resistance-micro-ohms：电池内阻。
- voltage-min-design-microvolt：最小电压，不建议修改。
- charge-sdp-current-microamp：充电类型为 sdp 时的充电电流，图示为 500ma—500ma。
- charge-dcp-current-microamp：充电类型为 dcp 时的充电电流，图示为 1150ma-3000ma。
- charge-cdp-current-microamp：充电类型为 cdp 时的充电电流，图示为 1150ma-1150ma。
- charge-unknown-current-microamp：充电类型未识别时的充电电流，图示为 500ma。

### 3.6.4 电池充电曲线配置

在配置之前，从电池厂会拿到一组基础电池数据，分温度分为 0°、-10°、25°、50°四组

ocv-capacity-celsius 和 ocv-capacity-table-0 两组数据是电池充电曲线的配置文件。

- **ocv-capacity-celsius** 代表驱动需要配置 20 组参数，不建议修改。
- **ocv-capacity-table-0** 为电池曲线配置，参数中代表电池电压和电池电量，例如，<3953000 65>，为电池电压在 3.95v 时的电池电量为 65%；配置中，一共分为 20 档，S726 中根据电池厂商提供的电池曲线表配置即可，对应的曲线温度 25°。

```
ocv-capacity-table-0 = <4330000 100>, <4249000 95>, <4189000 90>,  
                        <4133000 85>, <4081000 80>, <4034000 75>,  
                        <3991000 70>, <3953000 65>, <3910000 60>,  
                        <3866000 55>, <3836000 50>, <3813000 45>,  
                        <3795000 40>, <3782000 35>, <3774000 30>,  
                        <3765000 25>, <3750000 20>, <3726000 15>,  
                        <3687000 10>, <3658000 5>, <3400000 0>;
```

voltage-temp-table 为温度电压对照表，用于读取电池温度根据 NTC 变化，电池曲线无需进行修改。