Linux IR-RX
开发指南

版本号: 1.3
发布日期: 2024.11.26

版本历史

版本号	日期	制/修订人	内容描述
1.0	2022.11.1	XAA0248	初始版本
1.1	2023.3.20	XAA0311	调整文档结构，添加 FAQ
1.2	2023.12.13	XAA0311	优化内部接口及说明
1.3	2024.11.26	AWA2215	增加 Linux-6.6 内核版本支持，调整文档结构

目　录

插　图

1 前言

1.1 文档简介

介绍 Sunxi 平台上 IR-RX 驱动接口与调试方法，为 IR-RX 模块开发提供参考。

1.2 目标读者

IR-RX 模块内核层以及应用层的开发、维护人员。

1.3 适用范围

表 1-1: 适用产品列表

内核版本	驱动文件
Linux-5.10	sunxi-ir-rx.c
Linux-5.15	sunxi-ir-rx.c
Linux-6.6	sunxi-ir-rx.c

1.4 文档约定

1.4.1 标志说明

⚠ 注意

- **提醒操作中应注意的事项。不当的操作可能会损坏器件，影响可靠性、降低性能等。**

说明

为准确理解文中指令、正确实施操作而提供的补充或强调信息。

技巧

一些容易忽视的小功能、技巧。了解这些功能或技巧能帮助解决特定问题或者节省操作时间。

1.4.2 地址与数据描述方法约定

本文档在描述地址、数据时遵循如下约定：

表 1-2: 地址与数据描述方法约定

符号	例子	说明
0x	0x0200，0x79	地址或数据以 16 进制表示。
0b	0b010，0b00 000 111	数据采用二进制表示 (寄存器描述除外)。
X	00X，XX1	数据描述中，X 代表 0 或 1。 例如，00X 代表 000 或 001；XX1 代表 001，011，101 或 111。

1.4.3 数值单位约定

本文档在描述数据容量（如 NAND 容量）时，单位词头代表的是 1024 的倍数；描述频率、数据速率等时则代表的是 1000 的倍数。具体如下：

表 1-3: 数值单位约定

类型	符号	对应数值
数据容量（如 NAND 容量）	1 K	1024
	1 M	1 048 576
	1 G	1 073 741 824
频率，数据速率等	1 k	1000
	1 M	1 000 000
	1 G	1 000 000 000

1.5 相关术语介绍

1.5.1 硬件术语

表 1-4: 硬件术语

术语	解释说明
IR	Infrared Remote，红外模块

1.5.2 软件术语

表 1-5: 软件术语

术语	解释说明
Sunxi	指 Allwinner 的一系列 SOC 硬件平台。
RX	接收。
NEC 协议	一种标准的红外传输协议。

2 模块介绍

红外遥控的发射电路是采用红外发光二极管来发出经过调制的红外光波；红外接收电路由红外接收二极管、三极管或硅光电池组成，它们将红外发射器发射的红外光转换为相应的电信号，再送后置放大器。

鉴于家用电器的品种多样化和用户的使用特点，生产厂家对进行了严格的规范编码，这些编码各不相同，从而形成不同的编码方式，统一称为红外遥控器编码传输协议。到目前为止，红外遥控协议已多达十种，如：RC5、SIRCS、Sy、RECS80、Denon、NEC、Motorola、Japanese、SAMSWNG 和 Daewoo 等。我国家用电器的红外遥控器的生产厂家，其编码方式多数是按上述的各种协议进行编码的，而用得较多的有 NEC 协议。

2.1 模块功能介绍

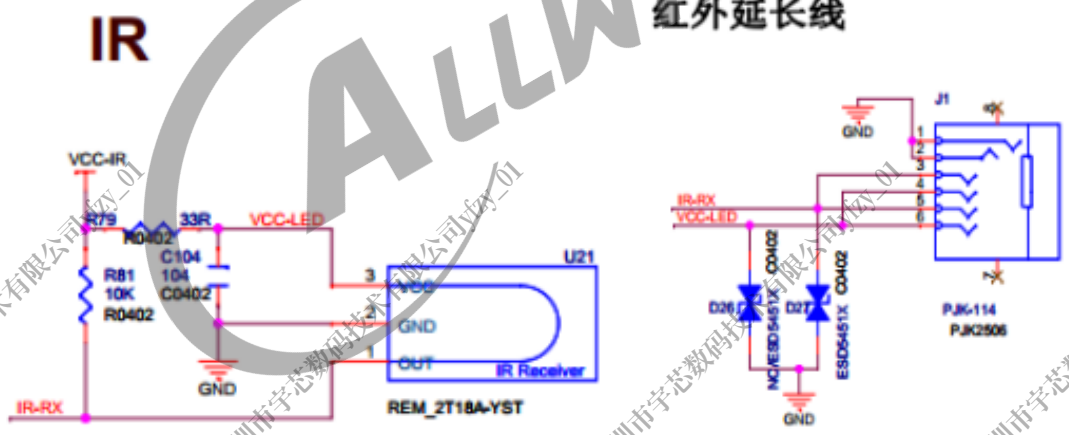

图 2-1: IR-RX 模块

IR-RX 接到主控的 IR-RX 模块的接收管脚。当 IR 接收到数据后，会产生中断，软件收到中断会进行数据读取。

2.2 结构框图

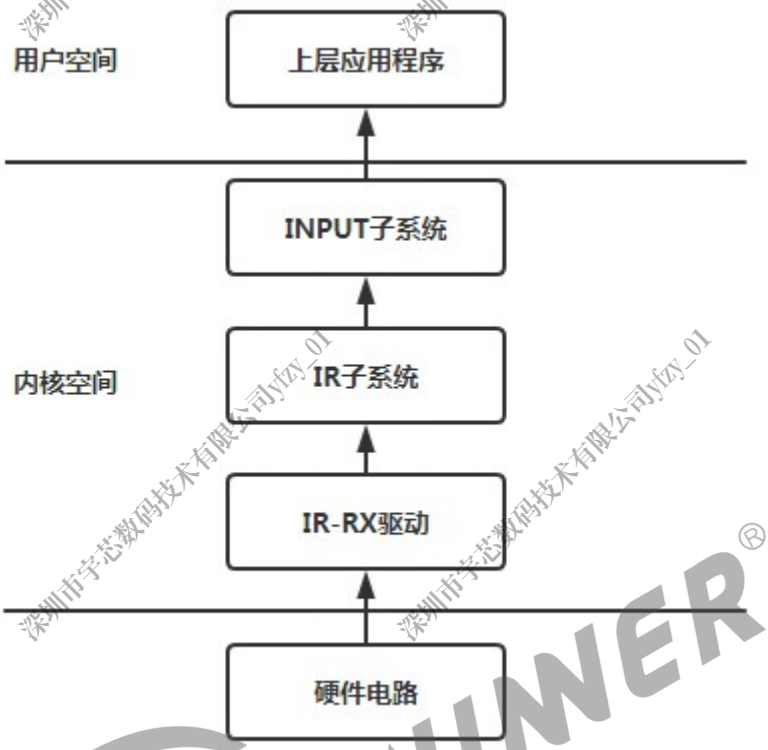

图 2-2: IR-RX 模块结构框图

IR-RX 红外接收模块通过读取红外接收模块接收到的红外遥控器发过来的信息，并进行解码，最后通过内核的 input 子系统将解码的按键信息上报给上层用户空间。

2.3 模块配置介绍

在不同的 Sunxi 硬件平台中，IR-RX 控制器的数目不同；但对于同一块板子上的每一个 IR-RX 控制器来说, 模块配置类似，本小节展示 Sunxi 平台上的 s_cir 控制器配置（其他 IR-RX 控制器配置类似）。

2.3.1 device tree 默认配置

设备树中存在的是该类芯片所有平台的模块配置，IR-RX 的设备树配置如下所示：

```
s_cir0: s_cir@7040000 {
    compatible = "allwinner,s_cir";             //具体的设备，用于驱动和设备的绑定
    reg = <0x0 0x07040000 0x0 0x400>;           //设备使用的地址
    interrupts = <GIC_SPI 151 IRQ_TYPE_LEVEL_HIGH>; //设备使用的中断
```

```
    clocks = <&r_ccu CLK_R_APB0_BUS_IRRX>, <&dcxo24M>, <&r_ccu CLK_R_APB0_IRRX>;  //设备使用的时钟
    clock-names = "bus", "pclk", "mclk";                    //使用的时钟名
    resets = <&r_ccu RST_R_APB0_BUS_IRRX>;      //设备使用的复位时钟
    status = "disabled";                    //设备是否使用，dtsi中设为disabled，会被board.dts中的配置覆盖
};
```

2.3.2 board.dts 板级配置

board.dts 用于保存每一个板级平台的设备信息（如 demo 板，perf1 板，ver1 板等等），里面的配置信息会覆盖上面的 device tree 默认配置信息。

对应 board.dts 里面 s_cir 的具体配置如下：

```
&pio {
    s_cir0_pins_a: s_cir0@0 {
        pins = "PB1";                   //使用的引脚,不同芯片不一样
        function = "ir";
    };

    s_cir0_pins_b: s_cir0@1 {
        pins = "PB1";
        function = "gpio_in";
    };
};

&s_cir0 {
        pinctrl-names = "default", "sleep";   //设备使用的pin脚名称
        pinctrl-0 = <&s_cir0_pins_a>;       //设备使用的pin脚配置（default）
        pinctrl-1 = <&s_cir0_pins_b>;       //设备使用的pin脚配置（sleep）
        status = "okay";                //设备是否使用
};
```

2.3.3 kernel menuconfig 配置

在 SDK 根目录中执行./build.sh menuconfig，选择 **Allwinner BSP** 选项进入下一级 **Device Drivers** 配置，如下图所示：

图 2-3: Device Drivers

选择 **IR-RX Drivers** 选项进入下一级配置，如下图所示：

图 2-4: IR-RX Drivers

选择 **IR-RX Support for Allwinner SoCs** 选项，可选择直接编译进内核，也可编译成模块。如下图所示：

图 2-5: IR-RX Support for Allwinner SoCs

2.4 源码模块结构

源代码位于 sdk/bsp/drivers/ir-rx/目录下：

```
bsp/drivers/ir-rx/
├── sunxi-ir-rx.c          // Sunxi平台的ir-rx驱动代码
```

3 接口设计

3.1 内部接口

3.1.1 sunxi_irrx_probe

- 函数原型：static int sunxi_irrx_probe(struct platform_device *pdev)。
- 功能：初始化 IR RX 模块。
- 参数：pdev: 设备结构体。
- 返回值：成功返回 0，失败返回错误码。

3.1.2 sunxi_irrx_reg_cfg

- 函数原型：static void sunxi_irrx_reg_cfg(void __iomem *reg_base)。
- 功能：irrx 寄存器配置。
- 参数：reg_base：irrx 的基地址。
- 返回值：无。

3.1.3 irqreturn_t sunxi_irrx_irq

- 函数原型：static irqreturn_t sunxi_irrx_irq(int irq, void *dev_id)。
- 功能：中断服务函数，读取 FIFO 数据，并解码。
- 参数：irqno 中断号；dev_id 保存 IR RX 数据结构体。
- 返回值：返回 IRQ_HANDLED。

3.1.4 sunxi_irrx_recv

- 函数原型：static void sunxi_irrx_recv(u32 reg_data, struct sunxi_ir_rx *chip)。
- 功能：将接收的 ir 数据上报至 input 子系统。
- 参数：reg_data：FIFO 寄存器中的数据；chip：irrx 数据结构体。
- 返回值：无。

3.2 外部接口

IR-RX 模块在 Linux 内核中是作为字符设备使用，所以可以使用相关字符设备接口来对 IR-RX 模块进行相应的读写和配置操作。相关定义在 evdev.c 文件里面。下面介绍几个比较有用的函数：

3.2.1 evdev_open()

- 函数原型：static int evdev_open(struct inode *inode, struct file *file)。
- 功能描述：程序（C 语言等）使用 open(file) 时调用的函数。打开一个 IR-RX 模块设备。
- 参数说明：inode：inode 节点；file：file 结构体。
- 返回值：文件描述符。

3.2.2 evdev_read()

- 函数原型：static ssize_t evdev_read(struct file *file, char __user buffer, size_t count, loff_t ppos)。
- 功能描述：程序（C 语言等）调用 read() 时调用的函数。读取 IR-RX 模块上报事件数据。
- 参数说明：file，file 结构体；buf，写数据 buf；offset, 文件偏移。
- 返回值：成功返回读取的字节数，失败返回负数。

3.2.3 evdev_write()

- 函数原型：static ssize_t evdev_write(struct file *file, const char __user buffer, size_t count, loff_t ppos)。
- 功能描述：程序（C 语言等）调用 write() 时调用的函数。向 IR-RX 模块写入上报事件。
- 参数说明：file，file 结构体；buf，读数据 buf；offset, 文件偏移。
- 返回值：成功返回 0，失败返回负数。

3.2.4 evdev_ioctl()

- 函数原型：static long evdev_ioctl(struct file *file, unsigned int cmd, unsigned long arg)。
- 功能描述：程序（C 语言等）调用 ioctl() 时调用的函数。管理相关的 IR-RX 模块功能。
- 参数说明：file，file 结构体，cmd，指令，arg，其他参数。
- 返回值：成功返回 0，失败返回负数。

找到 IR-RX 模块对应的 eventX(如 dev/input/event0) 文件，就可以使用 C 语言的文件读写，控制函数来调用上述的接口。

4 模块使用范例

4.1 获取 IR-RX 模块 event 事件

查看 IR-RX 模块对应设备的上报事件节点，查看/proc/bus/input/devices 这个文件，找到对应设备名字对应的 Handlers 节点。若没有节点信息则证明驱动加载失败，需要进一步排查。

```
/ # cat proc/bus/input/devices
I: Bus=0019 Vendor=0001 Product=0001 Version=0100
N: Name="sunxi-keyboard"
P: Phys=sunxikbd/input0
S: Sysfs=/devices/virtual/input/input0
U: Uniq=
H: Handlers=kbd event0
B: PROP=0
B: EV=3
B: KEY=800 c0040 0 0 10000000

I: Bus=0019 Vendor=0001 Product=0001 Version=0100
N: Name="sunxi-ir"
P: Phys=sunxi-ir/input0
S: Sysfs=/devices/platform/soc/ir0/rc/rc0/input1
U: Uniq=
H: Handlers=kbd event1
B: PROP=0
B: EV=100013
B: KEY=2
B: MSC=10
```

图 4-1: 查看 event 事件

4.2 hexdump 获取 event 数据

获取 IR-RX 模块上报数据, 进入 /dev/input 目录，hexdump 相关的 event 节点，一旦 IR-RX 模块接收到数据，就会显示在控制台上。

```
/ # hexdump dev/input/event1
0000000 0098 0000 0115 000e 0004 0004 404d 0040
0000010 0098 0000 0115 000e 0000 0000 0000 0000
0000020 0098 0000 e57e 000e 0004 0004 404d 0040
0000030 0098 0000 e57e 000e 0000 0000 0000 0000
0000040 0099 0000 4a4d 0001 0004 0004 404d 0040
0000050 0099 0000 4a4d 0001 0000 0000 0000 0000
```

图 4-2: hexdump 调试

第七列和第八列是厂商码和键值，其中 4d 代表该遥控器的键值，4040 代表该遥控器的厂商码。

4.3 Android 获取 event 数据

android 提供了 getevent 来获取输入设备的信息。具体用法如下：可以通过 cmd 进入 adb shell 直接输入 getevent 查看。有时候无法确定是内核按键判断出错，还是 android 层没有响应某个按键，可以在串口下输入 getevent 调试命令，该命令会打出驱动上报的所有 input 事件，如果按遥控器有打印，并且键值正确，那说明是 android 响应的问题。

```
apollo-p2:/ $ getevent dev/input/event0
0004 0004 01ff404d
0000 0000 00000000
0004 0004 00ff404d
0000 0000 00000000
```

图 4-3: getevent 调试

以上是调试平台输入 getevent 的效果，按的时候会上报事件，其中 4d 代表该遥控器的键值，ff40 代表该遥控器的厂商码。上面的 01 代表按下，下面的是 00 则代表抬起。

4.4 Android 增加新遥控器

如果只是增加自定义的按键，主要是要修改 kl 文件；如果是增加新遥控器，则需要新建该遥控器对应的 kl 文件（填充键值对），并且将该 kl 表添加到 multiir.mk 文件中，示例如下。

- 在vendor/aw/homlet/hardware/input/multi_ir/keylayout目录中添加新遥控的键值映射表，例如customer_ir_ff40.kl：

```
key 23   BROWSER         WAKE
key 26   HOME            WAKE
key 28   VOLUME_DOWN      WAKE
key 30   MEDIA_PREVIOUS    WAKE
key 31   MEDIA_NEXT       WAKE
key 68   F7           WAKE
key 66   BACK           WAKE
key 67   MUTE           WAKE
key 69   MENU           WAKE_DROPPED
key 71   SEARCH         WAKE_DROPPED
#key 72   PROG_BLUE        WAKE
key 73   PROG_YELLOW      WAKE
key 77   POWER          WAKE
key 78   PROG_GREEN       WAKE
key 79   BACK           WAKE_DROPPED
key 80   MEDIA_PLAY_PAUSE   WAKE
key 84    MOUSE          WAKE_DROPPED
```

- 在vendor/aw/homlet/hardware/input/multi_ir/multiir.mk里面加上新创建的customer_ir_ff40.kl表：

```
--- a/hardware/input/multi_ir/multiir.mk
+++ b/hardware/input/multi_ir/multiir.mk
@@ -21,6 +21,7 @@ BASE_KL_COPY_LIST += customer_ir_9f00.kl \
      customer_ir_4040.kl \
      customer_ir_7f00.kl \
      customer_ir_bf00.kl \
+      customer_ir_ff40.kl \

 SYSTEM_KL_COPY_LIST := $(BASE_KL_COPY_LIST) \
      Vendor_000d_Product_3838.kl \
```

著作权声明

本文档及内容受著作权法保护，其著作权由珠海全志科技股份有限公司（“全志”）拥有并保留一切权利。

本文档是全志的原创作品和版权财产，未经全志书面许可，任何单位和个人不得擅自摘抄、复制、修改、发表或传播本文档内容的部分或全部，且不得以任何形式传播。

商标声明





（不完全列举）均为珠海全志科技股份有限公司的商标或者注册商标。在本文档描述的产品中出现的其它商标，产品名称，和服务名称，均由其各自所有人拥有。

免责声明

您购买的产品、服务或特性应受您与珠海全志科技股份有限公司（“全志”）之间签署的商业合同和条款的约束。本文档中描述的全部或部分产品、服务或特性可能不在您所购买或使用的范围内。使用前请认真阅读合同条款和相关说明，并严格遵循本文档的使用说明。您将自行承担任何不当使用行为（包括但不限于如超压，超频，超温使用）造成的不利后果，全志概不负责。

本文档作为使用指导仅供参考。由于产品版本升级或其他原因，本文档内容有可能修改，如有变更，恕不另行通知。全志尽全力在本文档中提供准确的信息，但并不确保内容完全没有错误，因使用本文档而发生损害（包括但不限于间接的、偶然的、特殊的损失）或发生侵犯第三方权利事件，全志概不负责。本文档中的所有陈述、信息和建议并不构成任何明示或暗示的保证或承诺。

本文档未以明示或暗示或其他方式授予全志的任何专利或知识产权。在您实施方案或使用产品的过程中，可能需要获得第三方的权利许可。请您自行向第三方权利人获取相关的许可。全志不承担也不代为支付任何关于获取第三方许可的许可费或版税（专利税）。全志不对您所使用的第三方许可技术做出任何保证、赔偿或承担其他义务。