Rockchip
瑞芯微电子

Rockchip
红外遥控开发指南

发布版本:0.01

日期:2016.07

前言

概述

本文档主要介绍 Rockchip 红外遥控器的基本原理、添加遥控器的基本方法以及调试的一些基本手段。

产品版本

芯片名称	内核版本
RK312x	Linux3.10
RK3288	Linux3.10
RK3368	Linux3.10
RK322x	Linux3.10
RK3399	Linux4.4

读者对象

本文档（本指南）主要适用于以下工程师：
技术支持工程师
软件开发工程师

修订记录

日期	版本	作者	修改说明

目录

插图目录

1 红外遥控简介

1.1 概述

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。目前 RK 平台也只支持 NEC 编码的红外协议。

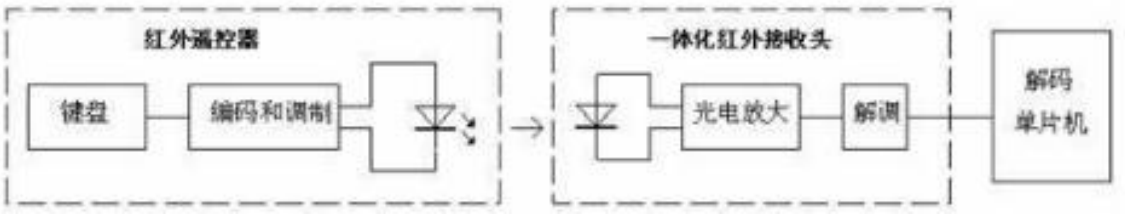


图 1-1 红外遥控系统框图

1.2 RK 平台上红外实现原理简介

PWM 有三种工作模式， reference mode, one-shot mode 和 continuousmode. 红外遥控器就采用 reference mode，这种模式下 PWM 可以捕获输入高低电平的宽度，并产生中断，CPU 接收到中断后去相应的寄存器读取。

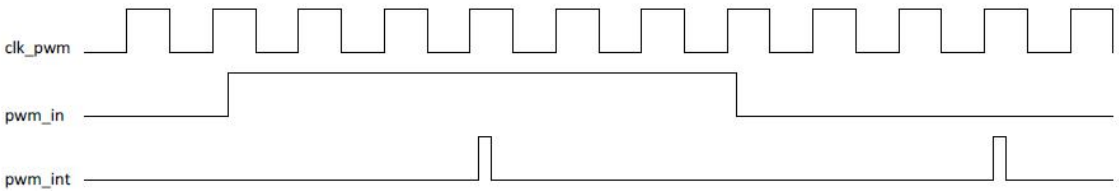
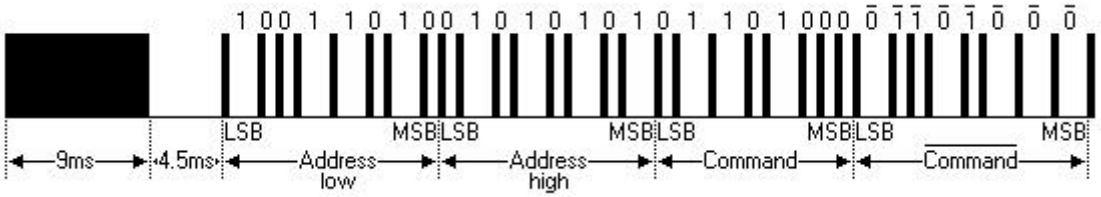


图 1-2 PWM Reference Mode 示意图

按下遥控的时候，红外接收头会产生一系列的高低电平，PWM 就会产生相应的中断，CPU 读取相应的寄存器就知道这些高低电平的时间，根据协议就可以解码出红外的用户码和键值码出来。下图是 NEC 红外编码协议的简单示意图，详细的协议附在最后。



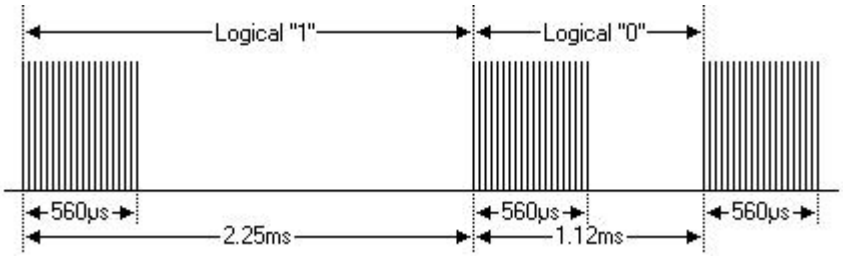


图 1-3 红外 NEC 编码协议简单图示

2 遥控器添加方法

添加一个新的遥控器支持比较简单，大概分为记录键值，添加键值进代码两个大的步骤。以下详细介绍这两大步。

2.1 记录键值

a) 打开打印键值的调试开关

```
echo 1 > sys/module/rockchip_pwm_remotectl/parameters/code_print
```

b) 按遥控器的按键，记录下对应的键值

例如按向下键，有如下打印

```
[19634.735833] GET USERCODE=0x4040
[19634.762463] RMC_GETDATA=e9
```

则，该遥控器的 usercode 是 0x4040，向下键的键值就是 0xe9

如此反复，直到打印完遥控器上的所有键值。

2.2 添加键值

为了和内核 dts 结构相符合，将添加遥控器的修改放到 dts 里面，驱动文件不要做任何修改。

具体方法：将上述记录下的每一个按键的键值以及遥控器的 usercode 添加到对应项目 dts 文件的&remotectl {}结构中。例如 3288 默认的 box 的 dts 文件为 arch/arm/boot/dts/rk3288-box.dts，其他的也是在各自的 dts 文件中修改。

例如：

```
&remotectl {
        handle_cpu_id = <2>;
        ir_key1{
                rockchip,usercode = <0x4040>;
                rockchip,key_table =
                        <0xf2    KEY_REPLY>,
            <0xba        KEY_BACK>,
            <0xf4        KEY_UP>,
            <0xf1        KEY_DOWN>,
            <0xef        KEY_LEFT>,
            <0xee        KEY_RIGHT>,
            <0xbd        KEY_HOME>,
            <0xea        KEY_VOLUMEUP>,
            <0xe3        KEY_VOLUMEDOWN>,
            <0xe2        KEY_SEARCH>,
            <0xb2        KEY_POWER>,
            <0xbc        KEY_MUTE>,
            <0xec        KEY_MENU>;
        };
};
```

注意：

（1）dts 文件的格式要求比较严格，不然会编译不过。“{”后面的“;”，以及最后一项后面

要用“;”都要注意。

（2）切记要烧写 resource.img，不然 dts 的修改没烧进系统。

ir_key1 是代码第一个，这个名字没有要求，新加一个就序号加 1，第二个的话一般就命名成 ir_key2，以此类推。

handle_cpu_id 项代表 ir 中断在哪一个 cpu 上处理，如果是 4 核系统可以是 0~3，如果是双核系统只能是 0~1，为了遥控中断更好的响应建议错开 cpu0 去处理。

3 红外按键定义

RK 平台红外键值参考

（1）	Down	（向下）	KEY_DOWN = 108
（2）	Up	（向上）	KEY_UP = 103
（3）	Left	（向左）	KEY_LEFT = 105
（4）	Right	（向右）	KEY_RIGHT = 106
（5）	Enter	（确定）	KEY_REPLY = 232
（6）	ESC	（返回）	KEY_BACK = 158
（7）	Menu	（菜单）	KEY_MENU = 139
（8）	Search	（搜索）	KEY_SEARCH = 217
（9）	Power	（待机键）	KEY_POWER = 116
（10）	Vol+	（音量加）	KEY_VOLUMEUP = 115
（11）	Vol-	（音量减）	KEY_VOLUMEDOWN = 114
（12）	Mute	（静音键）	KEY_MUTE = 113
（13）	Home	（主页）	KEY_HOME = 102
（14）	Zoom out	（放大）	185
（15）	Zoom in	（缩小）	186
（16）	Rotate left	（左旋）	183
（17）	Rotate right	（右旋）	184
（18）	0	（数字 0）	KEY_0 = 11
（19）	1	（数字 1）	KEY_1 = 2
（20）	2	（数字 2）	KEY_2 = 3
（21）	3	（数字 3）	KEY_3 = 4
（22）	4	（数字 4）	KEY_4 = 5
（23）	5	（数字 5）	KEY_5 = 6
（24）	6	（数字 6）	KEY_6 = 7
（25）	7	（数字 7）	KEY_7 = 8
（26）	8	（数字 8）	KEY_8 = 9
（27）	9	（数字 9）	KEY_2 = 10

字母和符号键都是 linux 的标准键值，在可以在 include/dt-bindings/input/input.h 中查找。

需要注意的是确定键，是 KEY_REPLY，不然会有问题。

4 常见问题调试

4.1 调试打印开关

与打印键值一样，系统也预留调试打印开关，需要的时候可以打开调试开关

```
echo 1 > sys/module/rk_pwm_remotectl/parameters/dbg_level
```

有时候需要配合

```
echo 1 > /sys/module/rockchip_pwm_remotectl/parameters/code_print
```

一起打印，然后看出错的时候，是哪一个或者几个 bit 引起的，有时候放宽一点判断的条件即可，一般是通过修改上下限来达到，具体可以参考代码里面 bit 值的判断地方。

4.2 getevent

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。

```
shell@rk3288:/ # getevent
add device 1: /dev/input/event0
name:       "ff680000.pwm"
/dev/input/event0: 0001 006c 00000001
/dev/input/event0: 0000 0000 00000000
/dev/input/event0: 0001 006c 00000000
/dev/input/event0: 0000 0000 00000000
```

以上是我调试平台输入 getevent 的效果，最前面会列出所有的 input 设备，按的时候会上报事件，其中 0x6c 是上报的 linux 键值，后面的 1 代表按下，如果是 0 则代表弹起。

4.3 增加自定义按键

增加自定义的按键，首先是驱动要增加相应按键的键值，然后 android 层要做修改，主要是 kl 等要修改， kl 没有加入该按键的对应关系也会导致该按键无法使用。

如果是新加一个 android 的全新按键可以参考网络上的方法，这里不再一一列出。

另外 RK 平台的 kl 文件一般是放在\device\rockchip\rksdk 目录下，与驱动名字一样的文件（驱动中的“.”要换成 kl 中的“_”）。每个芯片的 kl 文件名字不一样，需要注意。

kl 的作用和相关的说明可以参考 google 的开发文档。

```
http://source.android.com/devices/tech/input/index.html
```

开机的 logcat 可以看出驱动是跟哪一个 kl 配对，每一个 input 设备都有一个 kl 文件与之配对，如果没有自己写 kl 的话，系统会有一个默认的 kl 与之配对。

```
W/EventHub(  467): Unable to disable kernel key repeat for /dev/input/event0: Function not implemented
I/EventHub(  467): New device: id=1, fd=99, path='/dev/input/event0', name='ff680000.pwm', classes=0x421, configuration='', keyLayout='/system/usr/keylayout/ff680000_pwm.kl', keyCharacterMap='/system/usr/keychars/Generic.kcm', builtinKeyboard=false, usingSuspendBlockIoctl=true, usingClockIoctl=true
```

如上，则说明名为 ff680000.pwm'的驱动，与 ff680000_pwm.kl 的 kl 及 Generic.kcm 的

kcm 配对

4.4 dumpsys input

可以使用 dumpsys input 命令 dump 出 input 的一些信息

5 附录 NEC 编码标准

NEC 格式的特征：

1：使用 38 kHz 载波频率

2：引导码间隔是 9 ms + 4.5 ms

3：使用 16 位客户代码

4：使用 8 位数据代码和 8 位取反的数据代码

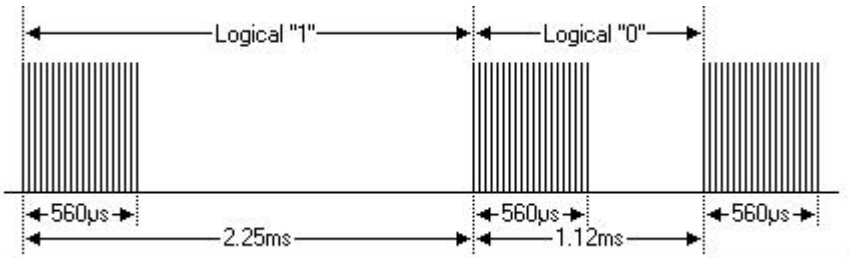
To my knowledge the protocol I describe here was developed by NEC. I've seen very similar protocol descriptions on the internet, and there the protocol is called Japanese Format.

I do admit that I don't know exactly who developed it. What I do know is that it is used in my late VCR produced by Sanyo and was marketed under the name of Fisher. NEC manufactured the remote control IC.

This description was taken from the VCR's service manual. Those were the days, when service manuals were fulled with useful information!

Features

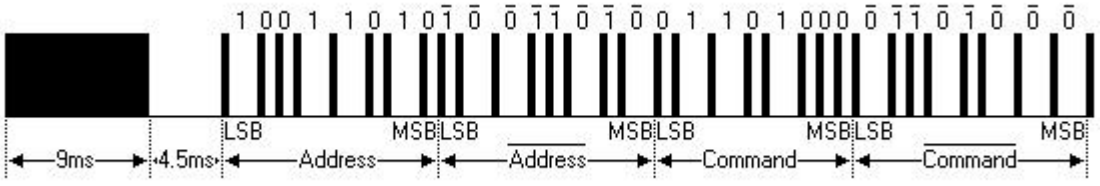
- 8 bit address and 8 bit command length
- Address and command are transmitted twice for reliability
- Pulse distance modulation
- Carrier frequency of 38kHz
- Bit time of 1.125ms or 2.25ms



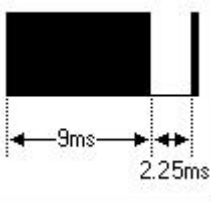
Modulation

The NEC protocol uses pulse distance encoding of the bits. Each pulse is a 560µs long 38kHz carrier burst (about 21 cycles). A logical "1" takes 2.25ms to transmit, while a logical "0" is only half of that, being 1.125ms. The recommended carrier duty-cycle is 1/4 or 1/3.

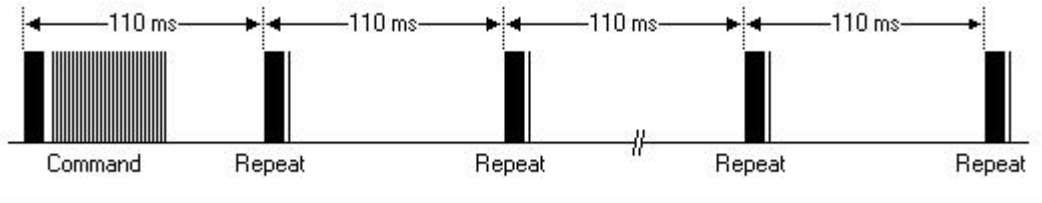
Protocol



The picture above shows a typical pulse train of the NEC protocol. With this protocol the LSB is transmitted first. In this case Address $59 and Command $16 is transmitted. A message is started by a 9ms AGC burst, which was used to set the gain of the earlier IR receivers. This AGC burst is then followed by a 4.5ms space, which is then followed by the Address and Command. Address and Command are transmitted twice. The second time all bits are inverted and can be used for verification of the received message. The total transmission time is constant because every bit is repeated with its inverted length. If you're not interested in this reliability you can ignore the inverted values, or you can expand the Address and Command to 16 bits each!



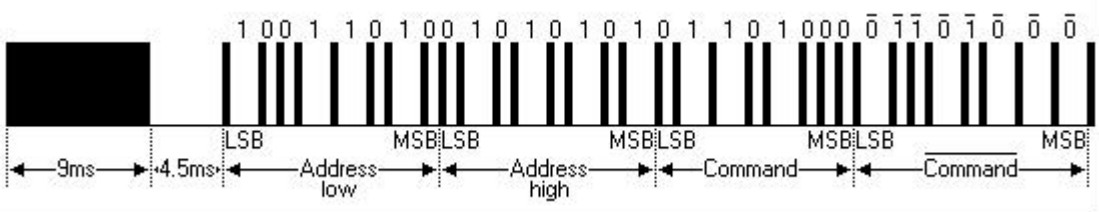
A command is transmitted only once, even when the key on the remote control remains pressed. Every 110ms a repeat code is transmitted for as long as the key remains down. This repeat code is simply a 9ms AGC pulse followed by a 2.25ms space and a 560µs burst.



Extended NEC protocol

The NEC protocol is so widely used that soon all possible addresses were used up. By sacrificing the address redundancy the address range was extended from 256 possible values to approximately 65000 different values. This way the address range was extended from 8 bits to 16 bits without changing any other property of the protocol.

The command redundancy is still preserved. Therefore each address can still handle 256 different commands.



Keep in mind that 256 address values of the extended protocol are invalid because they are in fact normal NEC protocol addresses. Whenever the low byte is the exact inverse of the high byte it is not a valid extended address.

http://jimlu.spaces.live.com/blog/cns!9B1C2AEA8D078F9A!609.entry
NEC Protocol http://www.sbprojects.com/knowledge/ir/nec.htm
Philips RC-5 Protocol http://www.sbprojects.com/knowledge/ir/rc5.htm