Linux IR_RX

开发指南

版本号: 1.0
发布日期: 2020.06.24

版本历史

版本号	日期	制/修订人	内容描述
1.0	2020.06.24	AW1637	version 1.0

目　录

1 前言 **1**
1.1 文档简介 . . . 1
1.2 目标读者 . . . 1
1.3 适用范围 . . . 1

2 模块介绍 **2**
2.1 模块功能介绍 . . . 2
2.2 结构框图 . . . 3
2.3 相关术语介绍 . . . 3
2.4 模块配置介绍 . . . 4
2.4.1 sys_config.fex 配置说明 . . . 4
2.4.1.1 linux-4.9 的设备树配置 . . . 4
2.4.1.2 linux-5.4 的设备树配置 . . . 5
2.4.1.3 board.dts 的配置 . . . 5
2.4.2 menuconfig 配置说明 . . . 5
2.5 源码结构介绍 . . . 10

3 接口设计 **11**
3.1 内部接口 . . . 11
3.1.1 evdev_open() . . . 11
3.1.2 evdev_read() . . . 11
3.1.3 evdev_write() . . . 12
3.1.4 evdev_ioctl() . . . 12
3.2 外部接口接口 . . . 12

4 模块使用范例 **14**

5 FAQ **16**

插　图

2-1 IR-RX 模块 2
2-2 IR-RX 模块结构框图 3
2-3 Device Drivers 6
2-4 Multimedia support 7
2-5 Remote Controller support 8
2-6 Sunxi IR remote control 9
2-7 NEC protocol and RC-5 protocol 10

1 前言

1.1 文档简介

介绍 IR-RX 模块的使用方法，方便开发人员使用。

1.2 目标读者

IR-RX 模块的驱动开发/维护人员。

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
T509	Linux-4.9	drivers/media/rc/sunxi-ir-dev.c
MR813	Linux-4.9	drivers/media/rc/sunxi-ir-dev.c
R818	Linux-4.9	drivers/media/rc/sunxi-ir-dev.c
A133	Linux-5.4	drivers/media/rc/sunxi-ir-dev.c

2 模块介绍

2.1 模块功能介绍

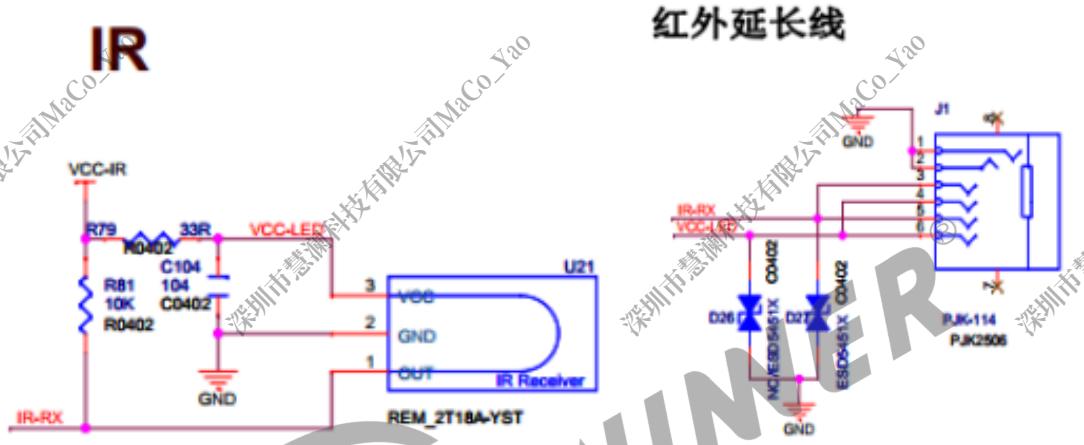

图 2-1: IR-RX 模块

VCC-PL 为 1.8V 供电，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

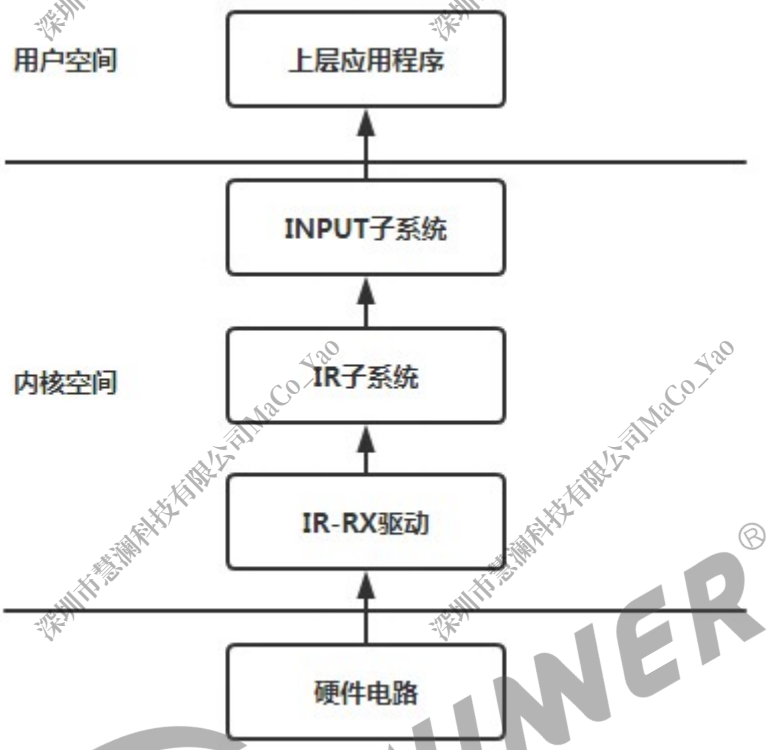

图 2-2: IR-RX 模块结构框图

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 相关术语介绍

表 2-1: 术语介绍

术语	解释说明
sunxi	指 Allwinner 的一系列 soc 硬件平台
IR	红外模块
RX	接收
NEC 协议	红外协议由脉冲调制和脉宽调制两种，NEC 协议是由 NEC 公司开发的，属于脉冲调制

2.4 模块配置介绍

2.4.1 sys_config.fex 配置说明

IR-RX 模块的设备树配置位于 longan 的内核目下的 arch/arm64(32 位地址为 arm)/boot/dts/sunxi/XXX.dtsi

2.4.1.1 linux-4.9 的设备树配置

linux-4.9 中 IR_RX 的配置如下所示：

```
    s_cir0:s_cir@07040000{
        compatible = "allwinner,s_cir";      //具体设备，用于和设备树绑定
        reg = <0x0 0x07040000 0x0 0x400>;    //设备使用的地址
        interrupts = <GIC_SPI 106 IRQ_TYPE_LEVEL_HIGH>; //设备使用的中断
        pinctrl-names = "default";
        pinctrl-0 = <&s_cir0_pins_a>;        //设备使用的引脚
        clocks = <&clk_hosc>,<&clk_cpurcir>;     //设备使用的时钟
        supply = "vcc-pl";           //设备使用的电源
        supply_vol = "3300000";
        status = "disabled";
        wakeup-source;       //作为唤醒源使用
    };
```

其中 s_cir0_pins_a 的配置位于内核目录下的 arch/arm64(32 位地址为 arm)/boot/dts/sunxi/XXX-pinctrl.dtsi。具体配置如下所示：

```
s_cir0_pins_a: s_cir0@0{
        allwinner,pins = "PH10";
        allwinner,function = "ir";
        allwinner,muxsel = <3>;
        allwinner,drive = <2>;
        allwinner,pull = <1>;
};
```

clk_cpurcir，clk_hosc 的配置位于内核目录下的 arch/arm64(32 位地址为 arm)/boot/dts/sunxi/XXX-clk.dtsi。具体配置如下所示：

```
clk_cpurcir: cpurcir {
     #clock-cells = <0>;
     compatible = "allwinner,periph-cpus-clock";
     clock-output-names = "cpurcir";
};

clk_hosc: hosc {
    #clock-cells = <0>;
    compatible = "allwinner,fixed-clock";
    clock-frequency = <24000000>;
    clock-output-names = "hosc";
};
```

2.4.1.2 linux-5.4 的设备树配置

linux-4.9 中 IR_RX 的配置如下所示：

```
s_cir0: s_cir@7040000 {
    compatible = "allwinner,s_cir";
    reg = <0x0 0x07040000 0x0 0x400>;
    interrupts = <GIC_SPI 151 IRQ_TYPE_LEVEL_HIGH>;
    clocks = <&r_ccu CLK_R_APB0_BUS_IRRX>, <&dcxo24M>, <&r_ccu CLK_R_APB0_IRRX>;
    clock-names = "bus", "pclk", "mclk";
    resets = <&r_ccu RST_R_APB0_BUS_IRRX>;
    supply = "";
    supply_vol = "";
    status = "disabled";
};
```

2.4.1.3 board.dts 的配置

若要使用 IR-RX 功能，需要在 longan/device/config/chips/{IC}/config/{BOARD}/board.dts 里配置相关参数：

```
    s_cir0:s_cir@07040000{
        status = "okay";
    };
```

也可以使用 sys_config.fex, 如果采用该文件作为设备树，

在 longan/device/config/chips/{IC}/config/{BOARD}/sys_config.fex 里面配置相关参数：

```
[s_cir0]
s_cir0_used         = 1 //使能红外接收
.....
```

以上三个配置文件优先级依次为 sys_config.fex 最高, board.dts 次之, 最后为内核下面的 dtsi 配置。

IR-RX 驱动模块中的模块参数 debug_mask 变量用来控制打印信息的开关

2.4.2 menuconfig 配置说明

linux-4.9 在命令行中进入内核根目录，执行 make ARCH=arm menuconfig （64 位平台执行 make ARCH=arm64 menuconfig）进入配置主界面，并按以下步骤操作：

linux-5.4 在 longan 根目录中执行./build.sh menuconfig

- 首先，选择 Device Drivers 选项进入下一级配置，如下图所示：

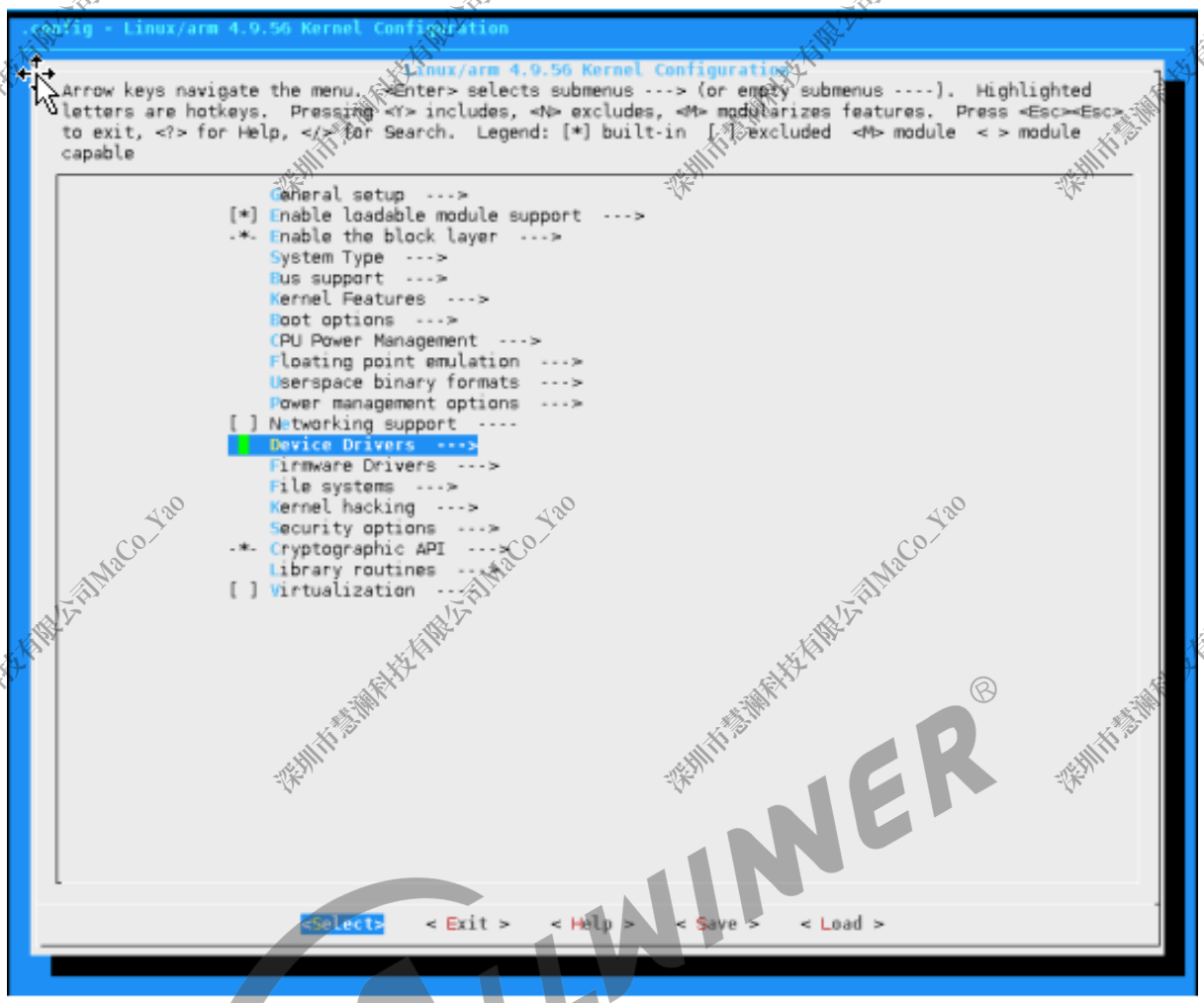

图 2-3: Device Drivers

- 在 Device Drivers 配置项下选择 Multimedia support, 如下图所示：

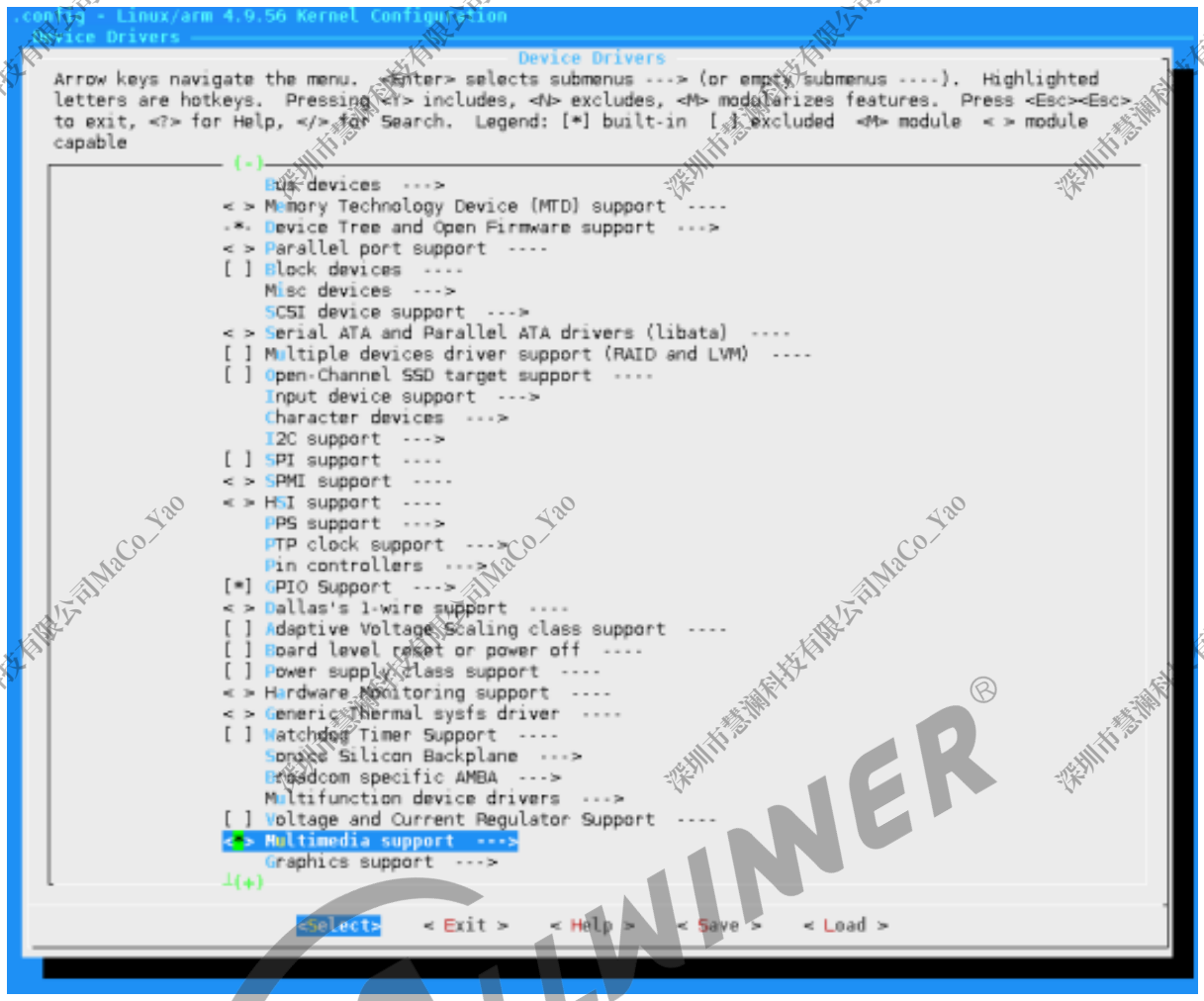

图 2-4: Multimedia support

- 在 Multimedia support 配置项下选择 Remote Controller support, 如下图所示：

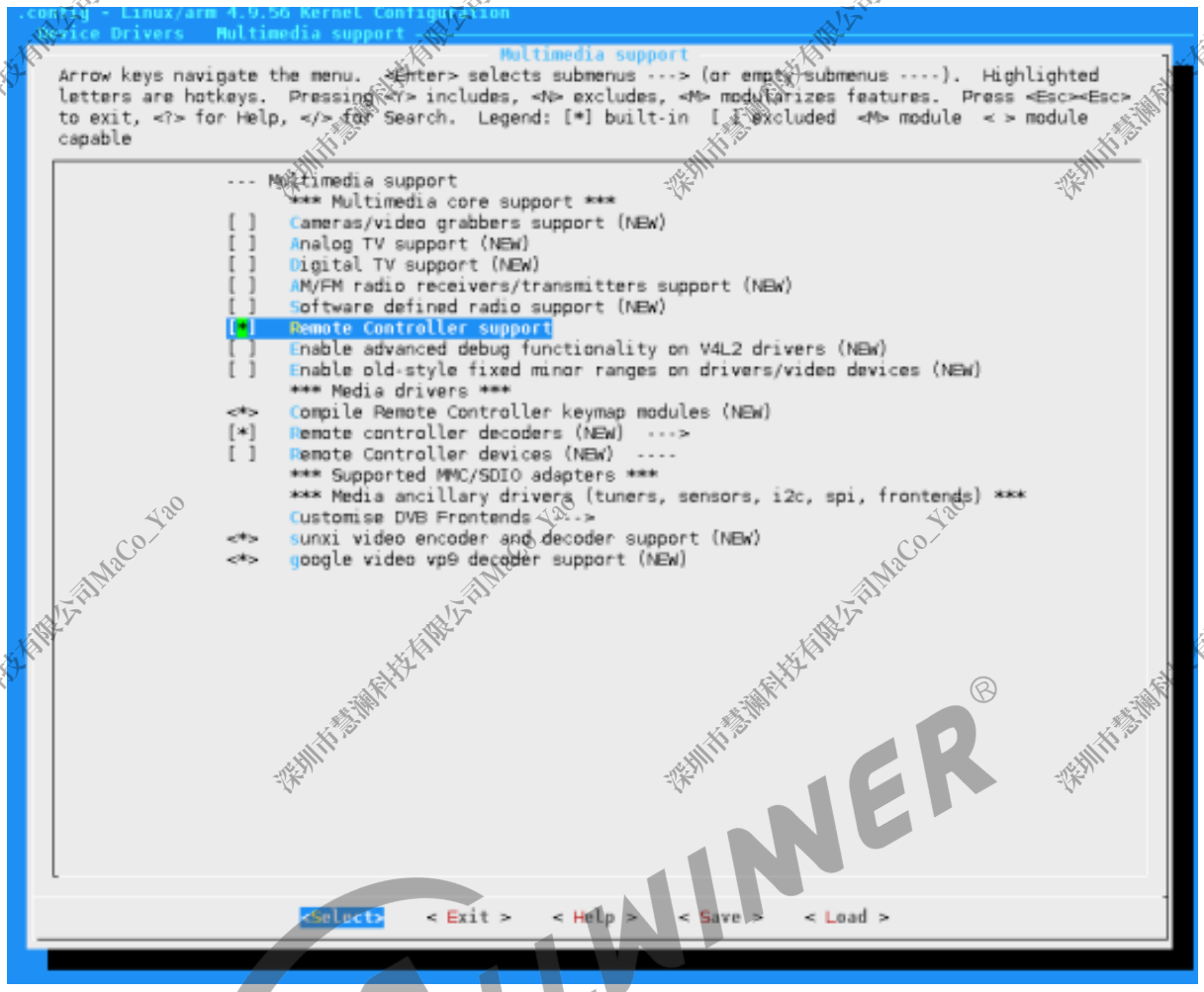

图 2-5: Remote Controller support

- 在 Remote Controller devices 中配置项下选择 Sunxi IR remote control, 如下图所示：

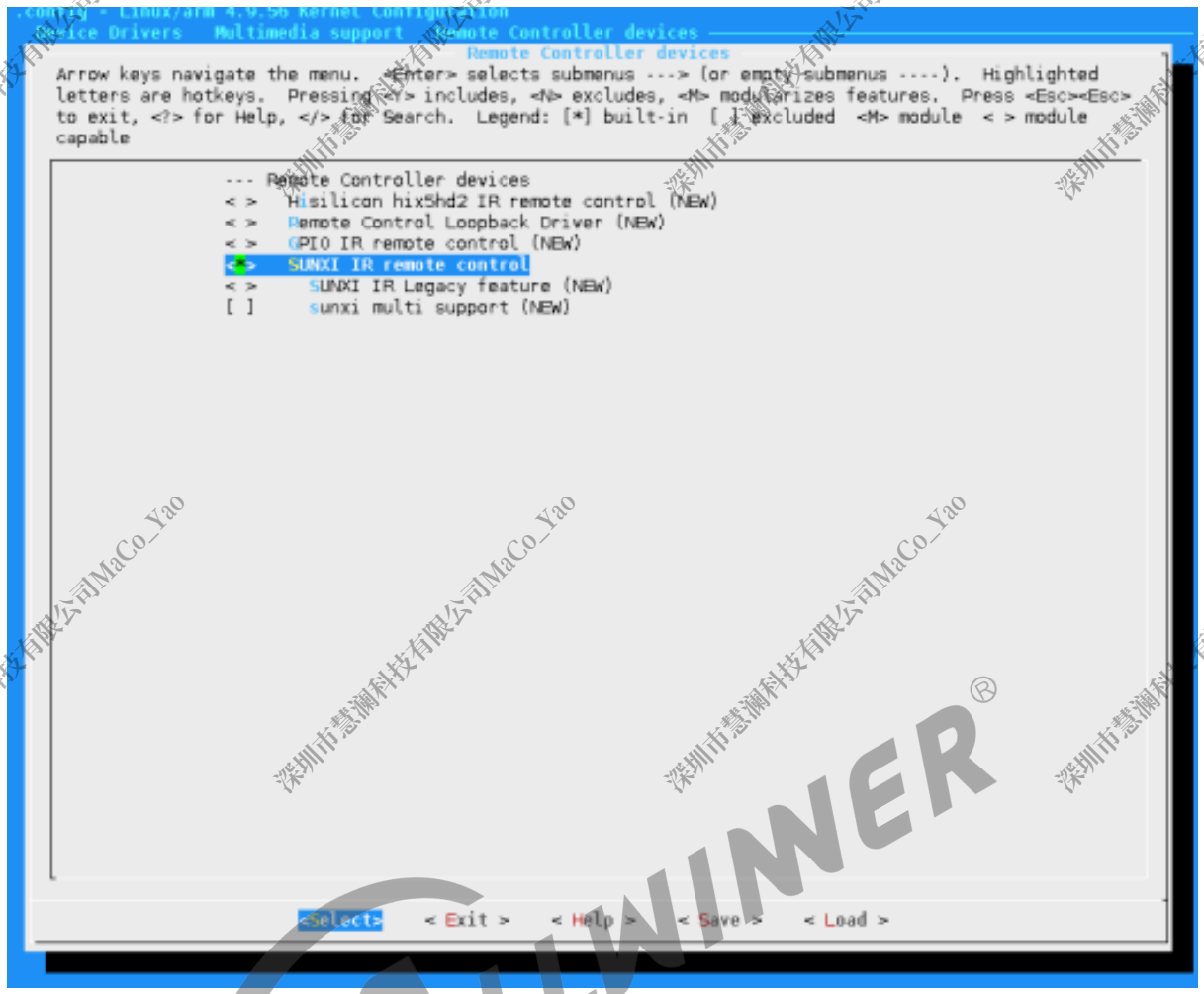

图 2-6: Sunxi IR remote control

在 Remote Controller decoders 中配置项下选择 Enable IR raw devoder for the NEC protocol (NEW) 与 Enable IR raw devoder for the RC-5 protocol (NEW). 如下图所示

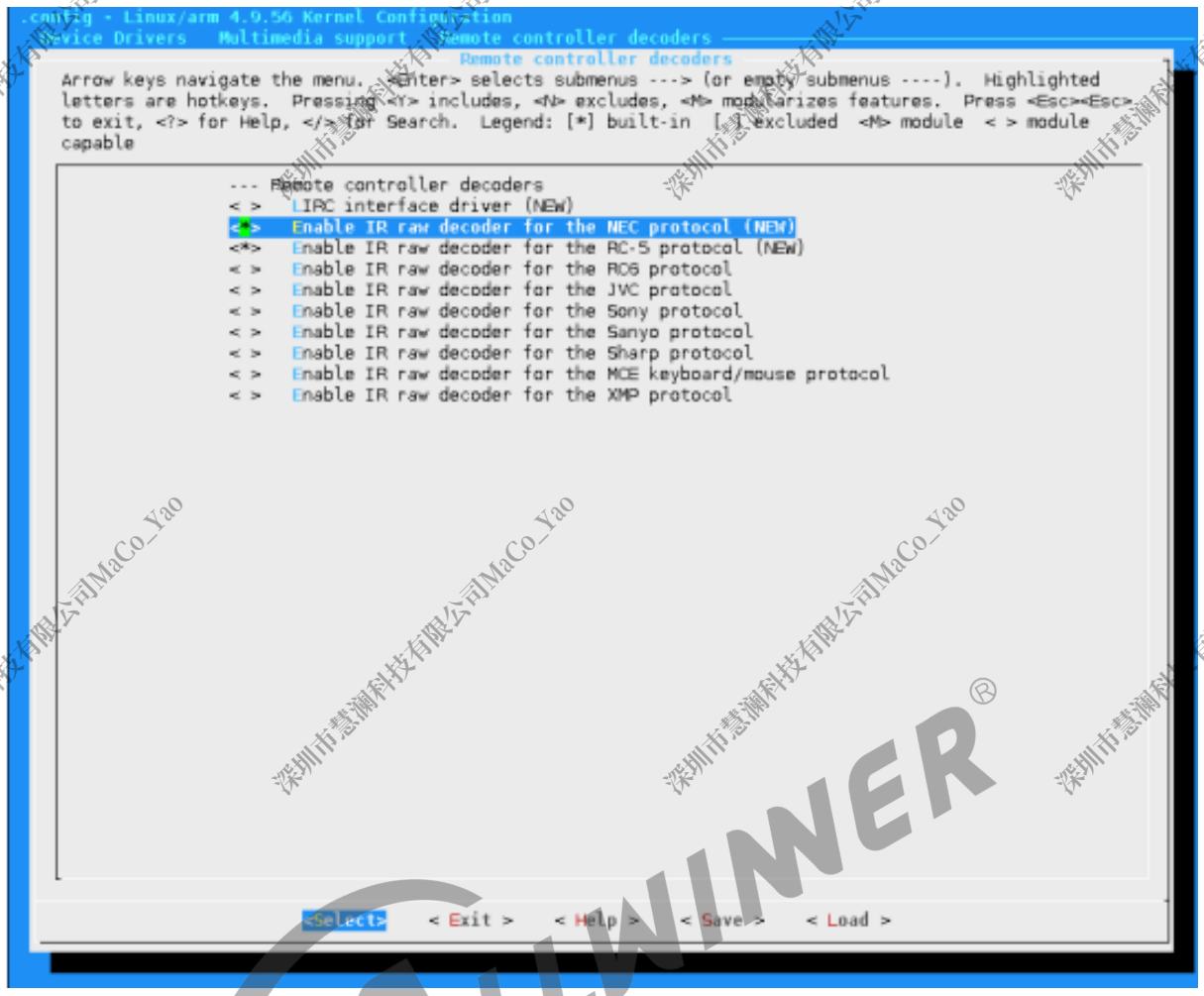

图 2-7: NEC protocol and RC-5 protocol

2.5 源码结构介绍

```
drivers/media/ rc/
├── sunxi-ir-dev.c      //获取设备节点，设置并注册rc_dev结构，申请中断并在中断处理函数进行事件的上报。
├── sunxi-ir-rx.h       //IR-RX模块头文件
├── rc-sunxi-keymaps.c      //SUNXI平台按键匹配函数，linux3.10的平台后匹配的过程都放在安卓。
├── ir-core.h           //定义了rc_dev等结构体
├── ir-main.c           //定义了rc_register_device，rc_keydown等核心函数
├── ir_nec_decoder.c        //NEC协议解码文件
├── ir_raw.c            //定义了ir_raw_init等函数来加载解码模块
```

3 接口设计

IR-RX 模块采用通用的 IR 框架进行读写，所以可以使用通用的接口。

3.1 内部接口

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：

3.1.1 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。

3.1.2 evdev_read()

- 函数原型：static ssize_t evdev_read(struct file *file, char __user *buf, 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。

3.1.3 evdev_write()

- 函数原型：static ssize_t evdev_read(struct file *file, const char __user *buf,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。

3.1.4 evdev_ioctl()

- 函数原型：static long evdev_read(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。

3.2 外部接口接口

在内核中，查看 /proc/bus/input/devices，确认 IR-RX 模块的数据上报节点，看下面的 Handlers 属性。

```
I: Bus=0019 Vendor=0001 Product=0001 Version=0100
N: Name="sunxi-ir"
P: Phys=sunxi-ir/input0
S: Sysfs=/devices/platform/soc/7040000.s_cir/rc/rc0/input2
U: Uniq=
H: Handlers=kbd event2
B: PROP=0
B: EV=100013
B: KEY=2
B: MSC=10
```

然后直接在内核中 hexdump 相应的 event 节点，当 IR-RX 模块采集到数据的时候，可以看到 IR-RX 模块上报的数据。

```
/ # hexdump /dev/input/event2
0000000 bcc6 0000 3dbd 000f 0004 0004 081f 00f7
0000010 bcc6 0000 3dbd 000f 0000 0000 0000 0000
0000020 bcc6 0000 0e1d 0009 0004 0004 0801 00f7
0000030 bcc6 0000 0e1d 0009 0000 0000 0000 0000
```

android 提供了 getevent 来获取输入设备的信息，作用跟上面 hexdump /dev/input/event2 类似。具体用法如下：

```
    Usage: getevent [-t] [-n] [-s switchmask] [-S] [-v [mask]] [-d] [-p] [-i] [-l] [-q] [-c
     count] [-r] [device]
        -t: show time stamps    //前部打印时间
        -n: don't print newlines
        -s: print switch states for given bits
        -S: print all switch states
        -v: verbosity mask //打印设备的简易信息，同时也获取上报信息 ！！！！
        -d: show HID descriptor, if available
    -p: show possible events (errs, dev, name, pos. events)
        -i: show all device info and possible events    //打印出各个设备的具体信息 ！！！
        -l: label event types and names in plain text   //打印出上报事件类型       ！！！
        -q: quiet (clear verbosity mask)    //不打印设备信息，只捕获事件
        -c: print given number of events then exit
    -r: print rate events are received
```

可以通过 cmd 进入 adb shell 直接输入 getevent -l /dev/input/event2 查看

4 模块使用范例

下面是一个用来读取 IR-RX 模块的按键输入的一个 Demo。代码如下：

```
#include <stdio.h>
#include <linux/input.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <sys/time.h>
#include <limits.h>
#include <unistd.h>
#include <signal.h>

#define DEV_PATH "/dev/input/event0"   //difference is possible
const int key_exit = 102;
static int keys_fd = 0;

unsigned int test_keyboard(const char * event_file)
{
    int code = 0,i;

    struct input_event data;

    keys_fd=open(DEV_PATH, O_RDONLY);

    if(keys_fd <= 0)
    {
        printf("open %s error!\n",DEV_PATH);
        return -1;
    }

    for(i = 0;i < 10;i++)
    {
        read(keys_fd,&data,sizeof(data));

        if(data.type == EV_KEY && data.value == 1)
        {
            printf("key %d pressed\n",data.code);
        }
        else if(data.type == EV_KEY && data.value == 0)
        {
            printf("key %d releaseed\n",data.code);
        }
    }

    close(keys_fd);
    return 0;
}

int main(int argc,const char *argv[])
{
```

```
50
51          int rang_low = 0,rang_high = 0;
52          return test_keyboard(DEV_PATH);
53      }
```

该 Demo 用来读取 IR-RX 模块用于 KEY 的按键上报事件（其他类似）。其循环 10 次读取按键上报事件输入，并且显示出相应按键的值。

5 FAQ

无

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


、 
、全志科技、  （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。