



D1-H Linux LRADC 开发指南

版本号: 1.0
发布日期: 2021.4.08

版本历史

版本号	日期	制/修订人	内容描述
1.0	2021.4.08	XAA0192	1. 创建该文档



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	2
2.2 结构框图	3
2.3 模块配置	4
2.3.1 设备树配置	4
2.3.2 menuconfig 配置	4
2.4 模块源码结构	8
3 接口设计	9
3.1 外部接口	9
3.1.1 确认 LRADC 模块的 event 节点	9
3.1.2 读取 LRADC 模块的上报数据	9
3.1.3 查看 LRADC 模块的中断次数	9
4 模块使用范例	11
5 FAQ	13

1 前言

1.1 文档简介

介绍 LRADC 模块的使用方法，方便开发人员使用。

1.2 目标读者

LRADC 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
D1-H	Linux-5.4	drivers/input/keyboard/sunxi-keyboard.c

2 模块介绍

2.1 模块功能

传统的按键采用矩阵式电路，这种按键电路复杂，同时需要多个 IO 口，比较消耗硬件资源。这种按键软件需要不断轮询去查询是否有按键按下，对系统软件造成了一定的负担。

而我们的 LRADC 按键只需要一个 adc 采样接口即可，电路也比较简单。软件上采用中断的方式，按键按下，会有中断产生。中断服务程序完成对电路采样，并进行按键解码。此种方式软件负担少，工作效率高。

LRADC 模块属于 INPUT 输入设备，这里提供了一个测试按键供使用。

regulator 为 1.8V 的供电，LRADC 口按键的电压不同，CPU 通过对这个电压的采样来确定是否为按键按下。

2.2 结构框图

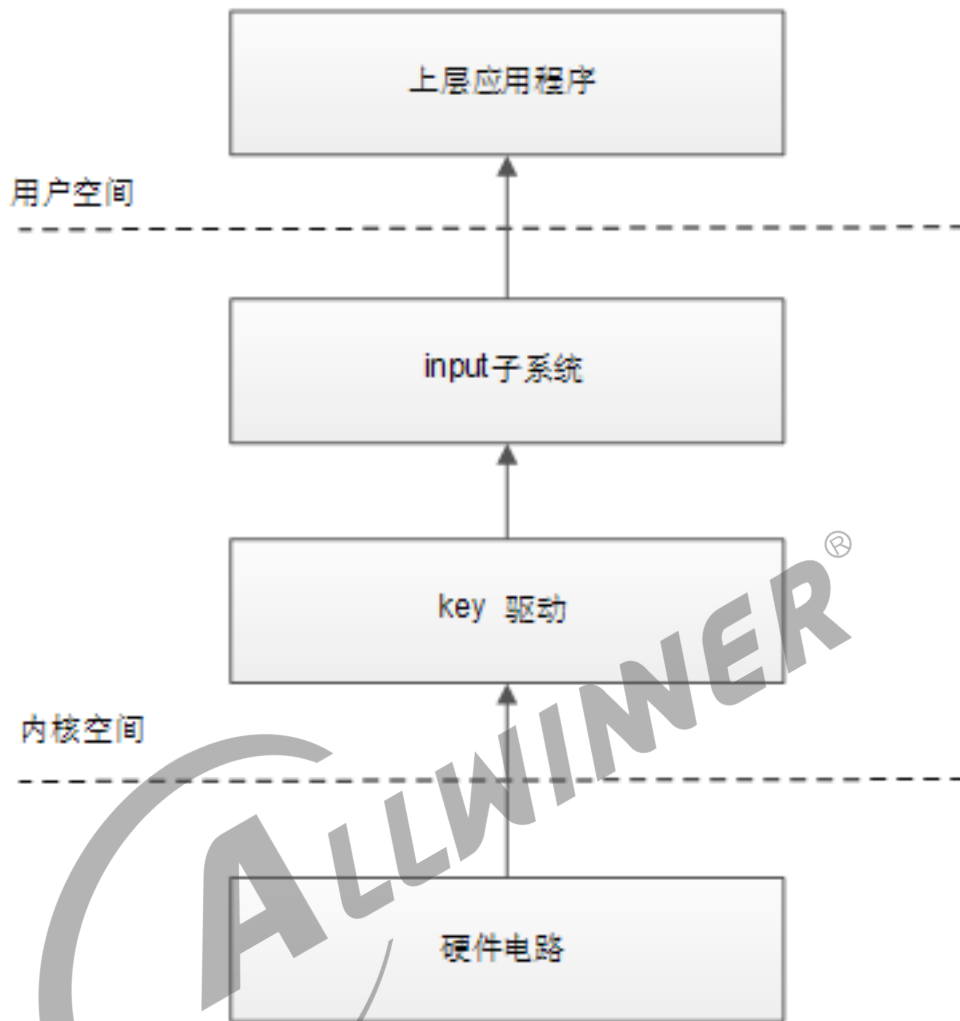


图 2-1: KEY 模块结构框图

整个系统框架流程如下所示：当用户按下按键的时候，会触发一个中断。这里的中断通过读取中断状态寄存器，判断是按键按下中断，还是数据中断，还是按键释放中断。KEY 按键驱动会进入中断，然后读取整个按键电路的电压值，然后对该电压值进行解码，然后将该事件上报给 INPUT 子系统。INPUT 子系统找到相应的事件处理程序之后，会将该按键事件上报给用户空间，等待用户程序对该按键信息的读取与处理。

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
LRADC	全志平台使用的按键模块

2.3 模块配置

2.3.1 设备树配置

LRADC 模块的设备树配置位于 Tina 的内核目录, 位于

linux-5.4/arch/riscv/boot/dts/sunxi/sun20iw1p1.dtsi, 下面为配置:

```
keyboard0: keyboard@2009800 {
    compatible = "allwinner,keyboard_1350mv"; // 与驱动进行匹配
    reg = <0x0 0x02009800 0x0 0x400>; // 基地址
    interrupts-extended = <&plic0 77 IRQ_TYPE_EDGE_RISING>; // 中断号和中断类型
    clocks = <&ccu CLK_BUS_LRADC>; // 时钟
    resets = <&ccu RST_BUS_LRADC>; // 时钟
    key_cnt = <5>; // 默认按键数
    key0 = <210 115>; // 按键电压及键值
    key1 = <410 114>;
    key2 = <590 139>;
    key3 = <750 28>;
    key4 = <880 172>;
    status = "okay"; // 是否使能,若需要开启,则将状态设为"okay"
};
```

若要使用 LRADC 功能, 需要在 Tina/device/config/chips/d1-h/config/nezha/linux-5.4/board.dts 里配置相关参数:

```
&keyboard0 {
    key_cnt = <1>; // 实际按键数
    key0 = <210 115>; // 按键电压及键值
    status = "okay"; // 是否使能,若需要开启,则将状态设为"okay"
};
```

2.3.2 menuconfig 配置

```
source build/envsetup.sh ----配置tina环境变量
lunch ----选择d1-h_nezha
make kernel_menuconfig ----进入内核配置主界面
```

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

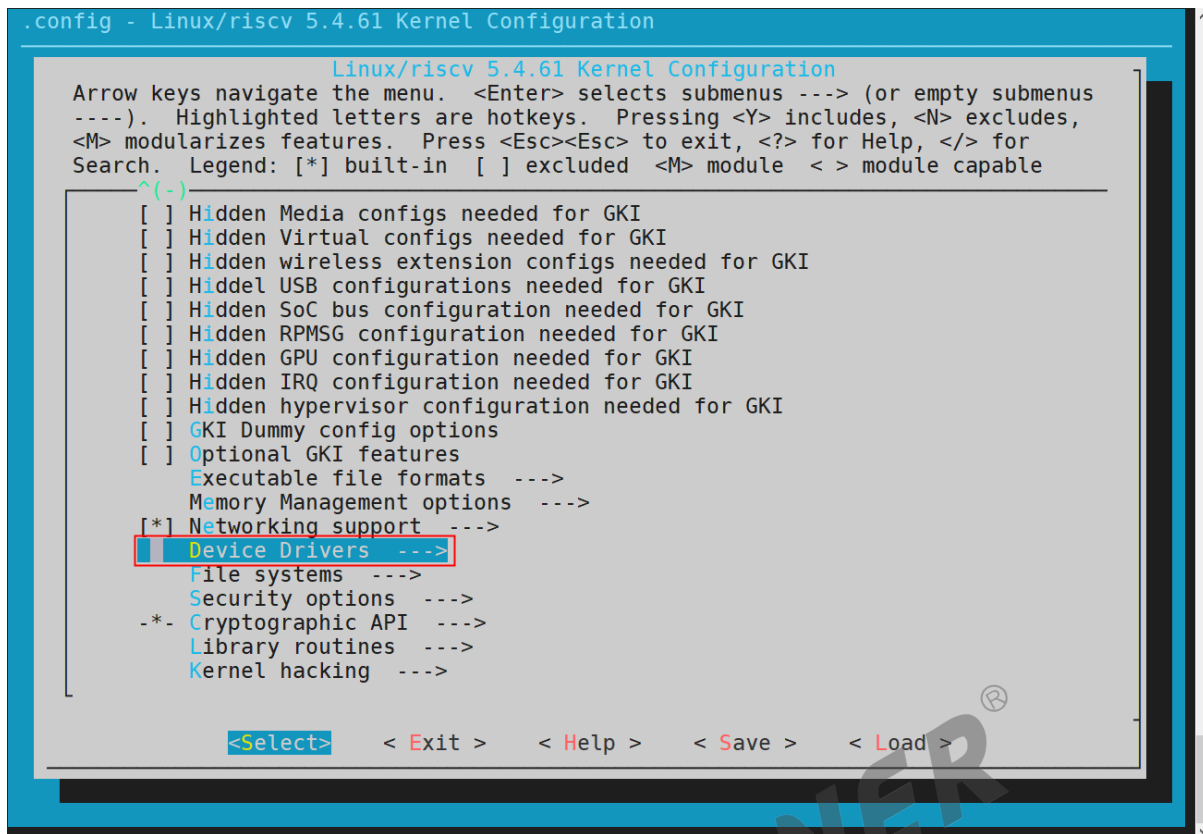


图 2-2: Device Drivers

- 选择 Input device support 选项进入下一级配置，如下图所示：

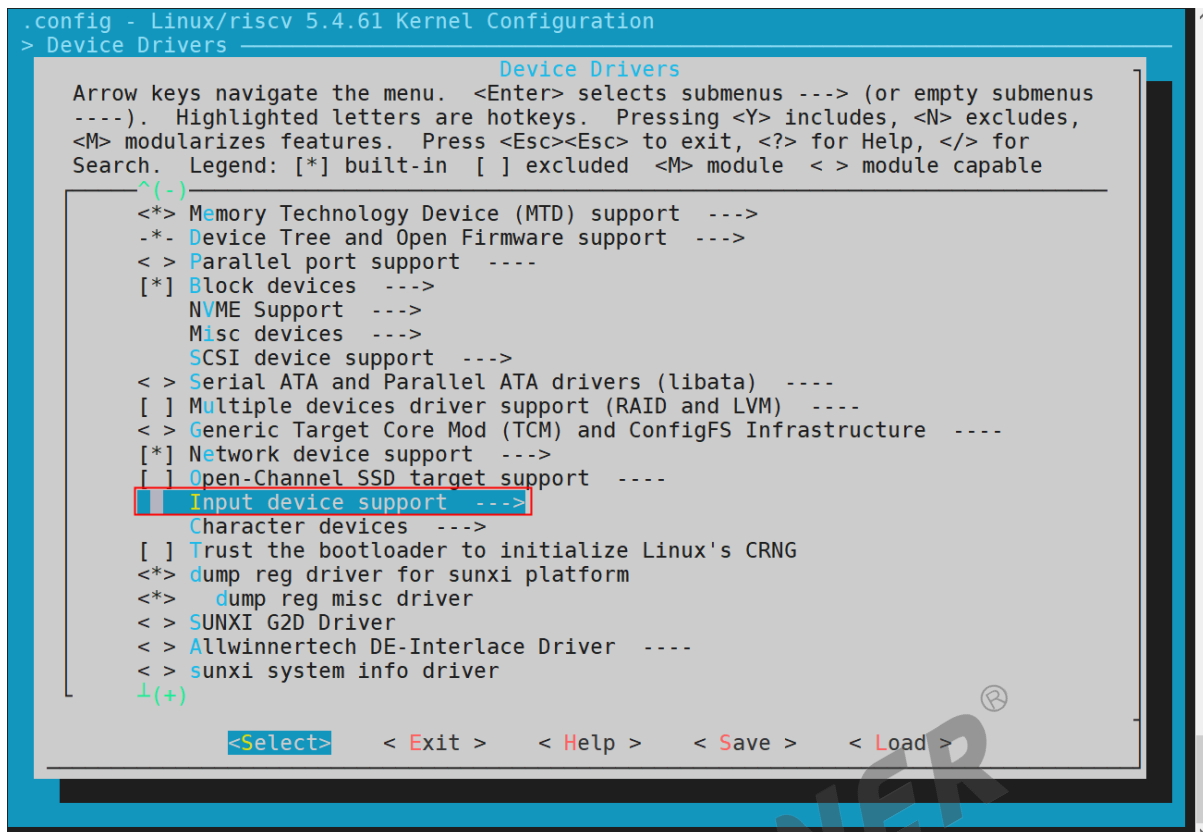


图 2-3: Input device support

- 选择 softwinner KEY BOARD support 加载按键支持，如下图所示：

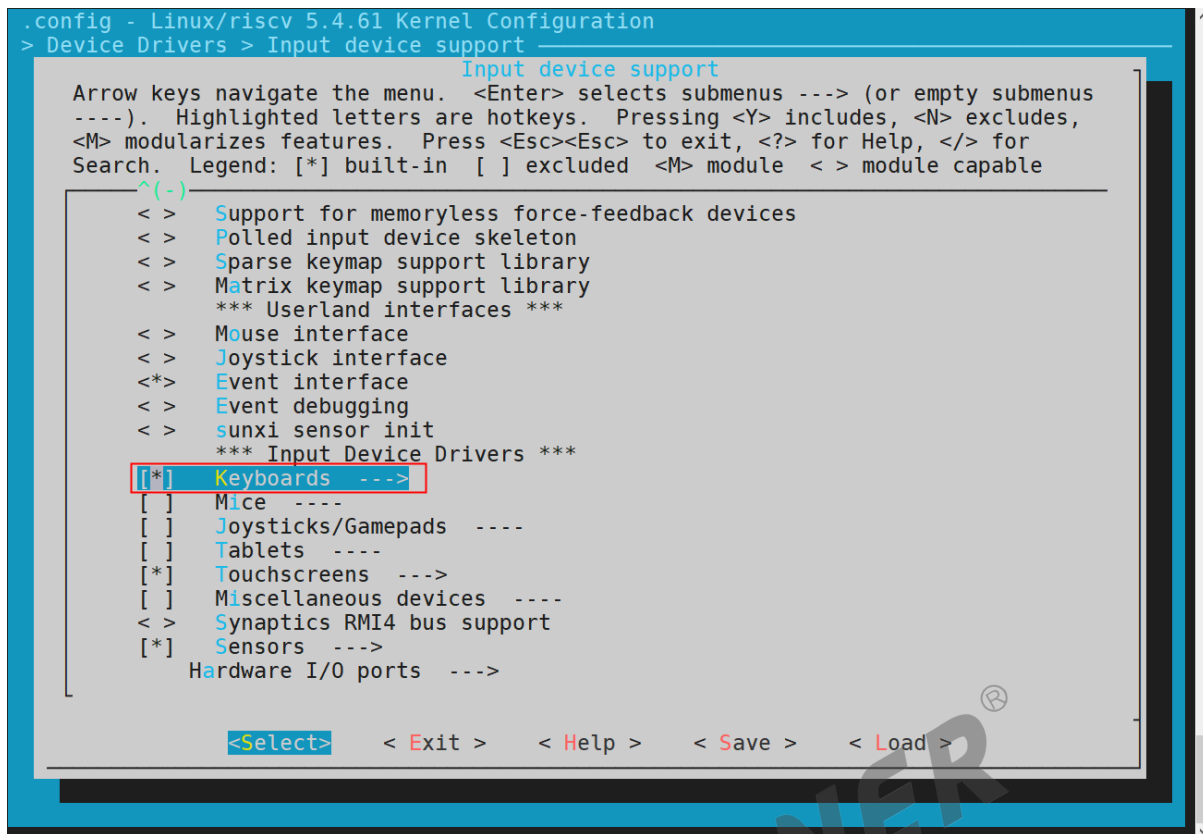


图 2-4: softwinner KEY BOARD support

- 返回 Input device support 选项，选择 Event Interface 选项，如下图所示：

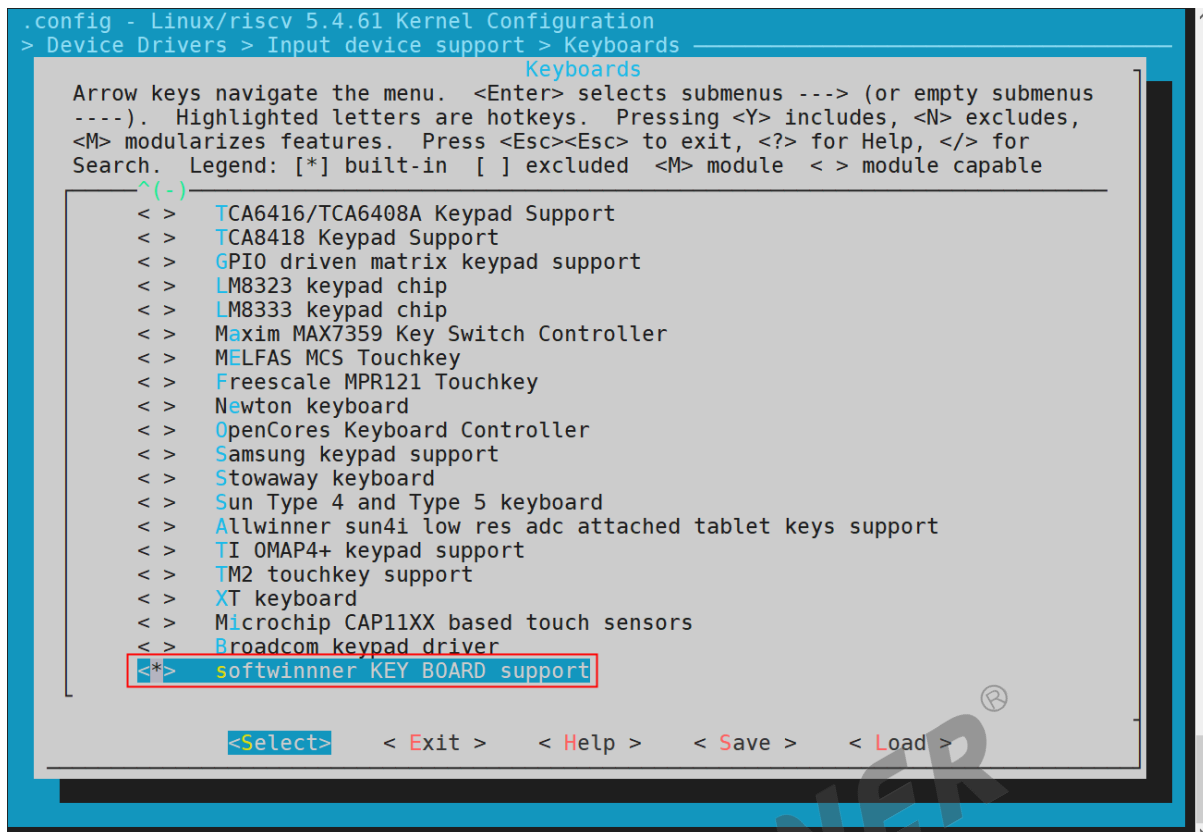


图 2-5: Event Interface

注：如需使用此模块功能，则需要同时使能设备树并开启 menuconfig 中的配置项。

2.4 模块源码结构

KEY 模块的源码结构如下所示：

```
drivers/input/keyboard/
├─ sunxi-keyboard.c
└─ sunxi-keyboard.h
```

3 接口设计

3.1 外部接口

3.1.1 确认 LRADC 模块的 event 节点

在内核中，查看 `cat /proc/bus/input/devices`，确认 LRADC 的数据上报节点。

```
/ # 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
```

3.1.2 读取 LRADC 模块的上报数据

直接在内核中 `hexdump` 相应的 event 节点，当按下按键后 LRADC 模块采集到数据的时候，可以看到 event 节点上报的数据。

```
/ # hexdump /dev/input/event0
00000000 bcc6 0000 3dbd 0009 0001 008b 0001 0000
00000010 bcc6 0000 3dbd 0009 0000 0000 0000 0000
00000020 bcc6 0000 0e1d 000b 0001 008b 0000 0000
00000030 bcc6 0000 0e1d 000b 0000 0000 0000 0000
```

其中，在读取到 event 节点的数据后，我们可以进行分析这些数据：每行的开头 4 个字节是 `hexdump` 打印的长度信息，后面跟着的是 16 字节的数据，`struct timeval` 占了 8 个字节，后面是 2 个字节的 `type`，2 个字节的 `code`，4 个字节的 `value`。具体实现也可以在内核代码中查看 `input_event` 结构体查看。

3.1.3 查看 LRADC 模块的中断次数

查看 `/proc/interrupts` 可以查看相关的 LRADC 模块中断次数。

```
/ # cat /proc/interrupts
          CPU0      CPU1      CPU2      CPU3
67:         280         0         0         0  wakeupgen  34 Level  sunxi-mmc2
68:          0         0         0         0  wakeupgen  32 Level  sunxi-mmc0
69:         12         0         0         0  wakeupgen  33 Level  sunxi-mmc1
85:          0         0         0         0  wakeupgen  31 Edge  sunxikbd  //
sunxi keyboard
```



4 模块使用范例

为了演示 LRADC 模块的使用，下面将演示用 C 语言对 LRADC 模块上报的数据进行读写：

```
#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 released\n", data.code);
        }
    }

    close(keys_fd);
    return 0;
}
```

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

```
int rang_low = 0, rang_high = 0;  
return test_keyboard(DEV_PATH);  
}
```

该 Demo 用来读取 LRADC 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

版权所有 © 2022 珠海全志科技股份有限公司。保留一切权利。

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。