全 志 科 技
Allwinner Technology

A20 平台 I2C 设备调试使用文档

V1.0

2013-06-17

Revision History

Version	Date	Changes compared to previous issue
V1.0	2013-06-17	Initial version

目录

1. 前言

1.1. 编写目的

模块主要针对 i2c 设备的相关调试，获取或者设置设备的相关信息，快速的获取设备的相关信息，加快定位问题的时间。

（由于文档不断补充，代码也不断更新，有些地方可能和实际代码中有细微差别，请注意）

1.2. 适用范围

适用于 A20 对应平台。

1.3. 相关人员

项目中 i2c 设备驱动的开发，维护以及使用人员应认真阅读该文档。

1.4. 文档介绍

本文主要针对模块进行相关的介绍，对其中的使用方法做详细的介绍。

2.模块介绍

2.1 模块功能介绍

模块的基本功能：

（1）对 I2C 设备写入数据

（2）读取 I2C 设备数据

（3）根据 sysconfig.fex 的主键值，打印主键值下的各子健值信息

（4）打印 input 设备（ctp，sensors，IR）的 xxx_config_info 的关键信息

（5）设置 ctp 的中断引脚的时钟源与分频比

（6）获取 ctp 的中断引脚以及时钟源与分频比信息。

2.2 模块源码位置

模块的源码文件为：i2c_device_debug.c,信息位于 input 目录下，如下所示：

Linux3.3 目录：\lichee\linux-3.3\drivers\input\i2c_device_debug.c

Linux3.4 目录：\lichee\linux-3.4\drivers\input\i2c_device_debug.c

2.2 模块配置介绍

对于模块的内核配置，可到 linux-3.x 目录下通过命令 make ARCH=arm menuconfig 进入配置主界面，以 linux3.4 为例子进行说明。并按以下步骤操作：

首先，选择 Device Drivers 选项进入下一级配置，如图 1 所示：

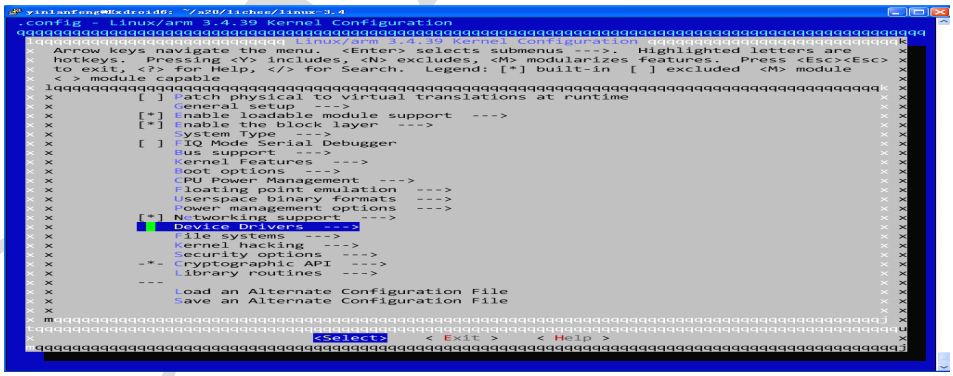

图 1 Device Drivers 选项配置

进入 Device Drivers 配置后，选择 Input device support 选项，如图 2 所示

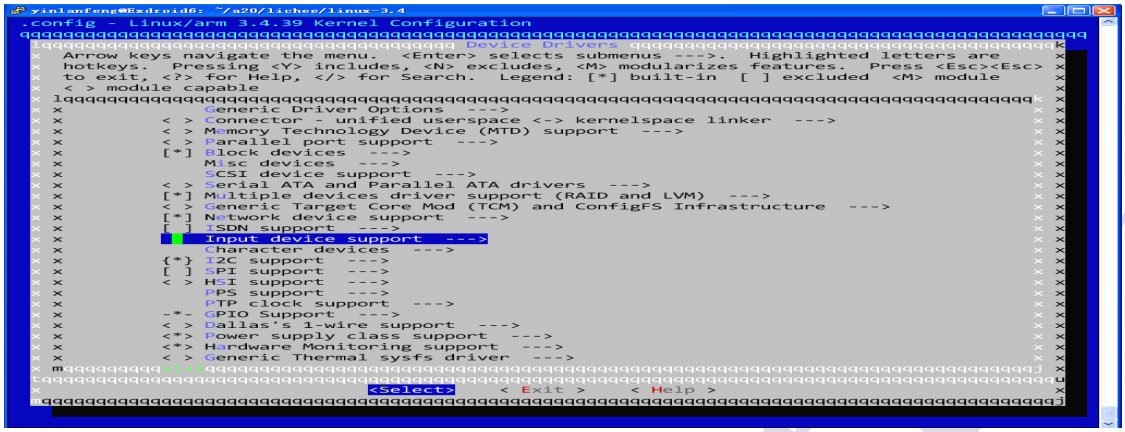

图 2 Input device support 选项配置

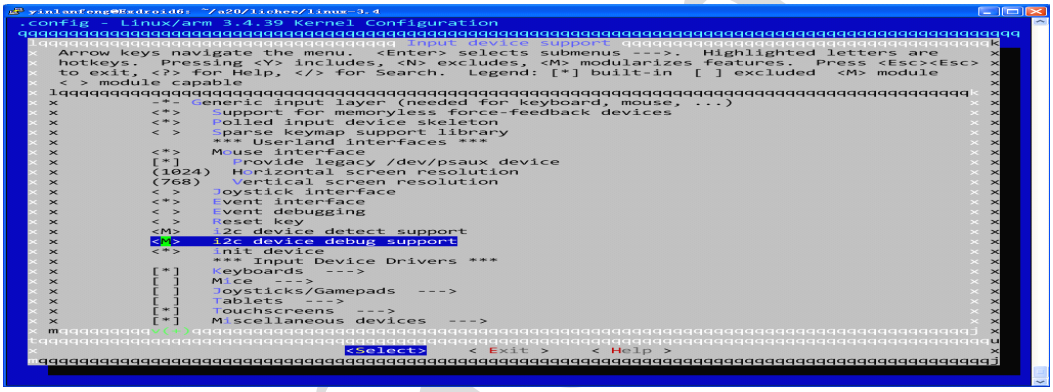

图 3 模块编译为模块

由图 3 可以看出，I2C DEBUG 编译为模块的形式，当需要使用调试时，加载该驱动即可。

3.使用方法

本模块作为调试 i2c 设备的一种辅助手段，编译为模块的形式，在需要时，在串口或者使用 adb shell 进行操作，且需要先加载模块的驱动。以 adb shell 环境下为例子进行说明如下：

（1）启动 shell，加载模块驱动 i2c_device_debug.ko

实验的环境下面由于添加 adb 的环境变量，因此在任何目录下都可以直接使用。如下所示：

```
C:\Documents and Settings\A>adb shell
root@android:/ # insmod system/vendor/modules/i2c_device_debug.ko
insmod system/vendor/modules/i2c_device_debug.ko
root@android:/ #
```

（2）进入模块节点，cd sys/class/device_of_i2c

模块加载成功之后，会生成 device_of_i2c 的 class 类，到该目录下进行相关节点的操作。如下：

```
root@android:/ # cd sys/class/device_of_i2c
cd sys/class/device_of_i2c
root@android:/sys/class/device_of_i2c #
```

（3）查看模块类下面存在的节点

模块类下面，存在 device，read，write，para，config，irq 等节点，如下所示：

```
root@android:/sys/class/device_of_i2c # ls
ls
config
device
irq
para
read
write
root@android:/sys/class/device_of_i2c #
```

各节点的作用如下表所示：

节点名称	节点作用
device	该节点为设备信息的描述，用于存放 I2C 设备的 i2c 组别号以及设备通讯地址信息
read	I2C 设备读取数据节点
write	I2C 设备写数据节点
para	用于打印 sysconfig.fex 下某主键下的所有子健的信息
config	用于显示 ctp，sensors，ir 等设备的 xxx_config_info 结构体的信息，xxx_config_info 结构体定义在 init-input.h 中
irq	1.设置 ctp 中断引脚的时钟源以及分频比 2.获取 ctp 中断引脚以及时钟源、分频比等信息

各节点的使用方法如下所示。

3.1 写 I2C 数据

通过 write 写数据时，需要先设置设备的信息，即希望向什么地方写数据。步骤如下：

（1）写 device 节点

通过 device 节点，写设备的 i2c 组别号以及设备 i2c 地址。如需要向位于第二组 i2c 总线下的 gt82x 设备写数据，该设备地址为 0x5d，如下：

```
root@android:/sys/class/device_of_i2c # echo 2,0x5d > device
echo 2,0x5d > device
root@android:/sys/class/device_of_i2c # _
```

注意：

echo 后面需要一个空格；

写入的格式：先写 i2c 的组别号再写设备地址，组别号与设备地址用逗号隔开且中间不能存在空格；

在“>”的前后各有一个空格。

写好后，可用 cat 命令查看是否已经写入成功。如果写入不成功，请检查是否写入时格式存在错误。查看 device 节点如下所示：

```
root@android:/sys/class/device_of_i2c # cat device
cat device
twi_id:2
dev_addr:0x005d
root@android:/sys/class/device_of_i2c #
```

（2）写 i2c 设备数据

写 i2c 数据，是通过 write 节点向设备写数据，如下：

```
root@android:/sys/class/device_of_i2c # echo 0xf80,0x00,0x11,0x03 > write
echo 0xf80,0x00,0x11,0x03 > write
```

注意：

echo 后面需要一个空格；

写入的格式：写入的数据均用逗号隔开且中间不能存在空格。写入的第一个数据必须为写入数据的起始寄存器地址，寄存器地址 8 位或者是 16 位寄存器都可以，在寄存器后面在为需要写入的数据；

在“>”的前后各有一个空格。

（3）用 cat 命令查看写入的数据

可以使用 cat write 命令，查看写入的数据，如下所示：

```
root@android:/sys/class/device_of_i2c # cat write
cat write
first write reg addr is:0x0f80
write reg         write val
0x0f80            0x0000
0x0f81            0x0011
0x0f82            0x0003
```

显示的第一排为寄存器地址，第二排为需要写入到该寄存器的数据。如果需要读取寄存器的数据，查看是否有正确写入，请看 3.2 节，如何读取寄存器地址。

3.2 读 I2C 数据

通过 read 读取数据时，需要先设置设备的信息，即希望从什么地方，什么设备读取数据。步骤如下：

（1）写 device 节点

通过 device 节点，写设备的 i2c 组别号以及设备 i2c 地址。如需要向位于第二组 i2c 总线下的 gt82x 设备读取数据，该设备地址为 0x5d，如下：

```
root@android:/sys/class/device_of_i2c # echo 2,0x5d > device
echo 2,0x5d > device
root@android:/sys/class/device_of_i2c #
```

注意：

echo 后面需要一个空格；

写入的格式：先写 i2c 的组别号再写设备地址，组别号与设备地址用逗号隔开且中间不能存在空格；

在“>”的前后各有一个空格。

写好后，可用 cat 命令查看是否已经写入成功。如果写入不成功，请检查是否写入时格式存在错误。查看 device 节点如下所示：

```
root@android:/sys/class/device_of_i2c # cat device
cat device
twi_id:2
dev_addr:0x005d
root@android:/sys/class/device_of_i2c #
```

（2）“echo 3,0xf80 > read”

向 read 节点写入需要读取的字节数，以及起始寄存器地址。

```
root@android:/sys/class/device_of_i2c # echo 3,0xf80 > read
echo 3,0xf80 > read
```

“echo 3,0xf80 > read”为需要读取从 0xf80 开始的 3 个数据，即读取 0xf80，0xf81，0xf82 这三个寄存器的数据。

注意：

echo 后面需要一个空格；

写入的格式：先写需要读取的数据个数再写读取数据的起始地址（寄存器地址可以为 8 位或者是 16 位，直接写即可，如十六位地址：0xf80，八位地址：0x80 等），且两者用逗号隔开不能有空格；

在“>”的前后各有一个空格。

（3）“cat read”

通过“cat read”命令查看希望读取的寄存器数据。如下：

```
root@android:/sys/class/device_of_i2c # cat read
cat read
number:3
reg_addr:0x0f80
reg_addr            val
0x0f80              0x0002
0x0f81              0x0011
0x0f82              0x0003
root@android:/sys/class/device_of_i2c #
```

打印的第一组为寄存器地址，第二组为相对应的寄存器中的数据。

3.3 打印 sysconfig.fex 中某主键下的子健信息

当希望打印某配置的选项时，又不希望通过驱动去增加相关的语句，那么 para 节点来帮您。使用步骤如下:

（1）“echo ctp_para > para”，向 para 节点写入需要打印的主键名称。

需要向 para 节点先写入 sysconfig.fex 下的主键名称。如希望打印 ctp_para 下的所有子健信息，如下：

```
root@android:/sys/class/device_of_i2c # echo ctp_para > para
echo ctp_para > para
```

注意：

echo 后面需要一个空格；

在“>”的前后各有一个空格。

（2）“cat para”，查看打印结果

由于打印的信息不是直接在 shell 中显示，因此需要另外启动一个 cmd，且启动 adb shell。使用命令：cat /proc/kmsg 查看打印信息。如下：

```
C:\Documents and Settings\A>adb shell
root@android:/ # cat /proc/kmsg
```

在之前的 shell 中，输入命令“cat para”即可在后启动的 shell 中看到打印的信息。

第一个 shell 中，输入命令“cat para”

```
root@android:/sys/class/device_of_i2c # cat para
cat para
key name:ctp_para
print the sysconfig para success!
root@android:/sys/class/device_of_i2c #
```

第二个 shell 中看到的打印信息：

```
<5>[ 1462.458026] init: process 'magdservice' killing any children in process group
<4>[ 1465.407830] script_dump_mainkey: dump ctp_para
<4>[ 1465.412862] ------------------------------------------------------
<4>[ 1465.421504]     name:      ctp_para
<4>[ 1465.429023]     sub_key:   name           type      value
<4>[ 1465.435156]                ctp_int_port   gpio      (gpio: 188, mul: 6, pull -1, drv -1, data
<4>[ 1465.451482]                ctp_wakeup     gpio      (gpio:  37, mul: 1, pull -1, drv -1, data
<4>[ 1465.462583]                ctp_name       string    "gt813_evb"
<4>[ 1465.470554]                ctp_screen_max_xint       1280
<4>[ 1465.477789]                ctp_screen_max_yint       800
<4>[ 1465.484332]                ctp_revert_x_flagint      1
<4>[ 1465.491176]                ctp_revert_y_flagint      1
<4>[ 1465.498236]                ctp_exchange_x_y_flagint      1
<4>[ 1465.505063]                ctp_used       int       1
<4>[ 1465.511708]                ctp_twi_id     int       2
<4>[ 1465.518600] ------------------------------------------------------
```

3.4 查看 input 设备结构体信息

通过 config 节点可以查看 ctp,sensors,ir 设备的 xxx_config_info 结构体中的信息，如希望查看 ctp 设备的相关信息。步骤如下：

（1）向 config 节点写入希望查看的设备编号。

设备编号如下所示：

设备名称	设备编号值
ctp	0
gsensor	1
gyr	2
ir	3
lightsensor	4
compass	5

查看 ctp 设备需要写入 0，如下：

```
root@android:/sys/class/device_of_i2c # echo 0 > config
echo 0 > config
root@android:/sys/class/device_of_i2c #
```

注意：

echo 后面需要一个空格；

在“>”的前后各有一个空格。

（2）“cat config”，将需要的信息打印出来

使用 cat 命令，将写入的设备编号的信息打印出来。如下：

```
root@android:/sys/class/device_of_i2c # cat config
cat config
ctp_used : 1
ctp_twi_id : 2
ctp_screen_max_x: 1280
ctp_screen_max_y: 800
ctp_exchange_x_y_flag: 1
int number: 188
wakeup number: 37
root@android:/sys/class/device_of_i2c #
```

3.5 ctp 中断引脚的操作

3.5.1 设置中断时钟源以及分频比

通过 irq 节点的 store 函数，设置 ctp 的中断时钟源以及分频比。该 store 函数中，首先获取 ctp_para 下的配置信息即获取中断引脚，使用时，请注意 ctp 的中断引脚的配置参数是否给有所改变。

ctp 的中断时钟有有 24Mhz（写 1），与 32Khz（写 0）。分频比由 0 到 7，将分频比设置为 n，则 ctp 的时钟 = 时钟源/2^n。

通过“echo 0,0 > irq”，设置 ctp 的中断时钟源为 32Khz，分频比为 0。

如下所示，将时钟源设置为 24Mhz，分频比为 7。

```
root@android:/sys/class/device_of_i2c # echo 1,7 > irq
echo 1,7 > irq
```

注意：

echo 后面需要一个空格；

写入的格式：第一个数据为时钟源，第二个数据为分频比，两者需要一个逗号隔开，且中间不能有空格。

在“>”的前后各有一个空格。

3.5.2 获取中断号以及时钟源及分频比信息

使用 cat 命令通过 irq 节点，可以获取当前 ctp 的中断引脚以及时钟源，分频比等信息。

如下所示：

```
root@android:/sys/class/device_of_i2c # cat irq
cat irq
int number : 188
clk : 1 (0 is 32Khz, 1 is 24Mhz)
clk_pre_scl : 7 (Frequency division coefficient)
root@android:/sys/class/device_of_i2c #
```

4. Declaration

The information furnished by Allwinner is believed to be accurate and reliable. Allwinner reserves the right to make changes in circuit design and/or specifications at any time without notice. Allwinner does not assume any responsibility and liability for its use. Nor for any infringements of patents or other rights of the third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of Allwinner. This datasheet neither states nor implies warranty of any kind, including fitness for any particular application.