**InnCable 调试记录文档**

|  |  |  |
| --- | --- | --- |
| **文件状态： [√] 草稿 [ ] 正在修改**  **[ ] 正式发布** | **部门** | **系统软件组** |
| **版本** | 0.3 |
| **作者** | **朱坤华** |
| **完成时间** |  |
| **审核** |  |
| **审核时间** |  |
| **密级状态：绝密( ) 秘密( ) 内部资料(√) 公开( )** | | |

**修改记录:**

|  |  |  |  |
| --- | --- | --- | --- |
| 版本 | 修订者 | 时间 | 说明 |
| 0.1 | 朱坤华 | 2018 / 3 / 8 | InnCable 调试记录文档 |
| 0.2 | 朱坤华 | 2018 / 4 / 19 | InnCable chromecast 和usb网卡接入后同时开机问题 |
| 0.3 | 朱坤华 | 2018 / 4 / 28 | InnCable AM1805 RTC 校准问题 |
|  |  |  |  |

**一. 背景及问题:**

Inncable 项目的调试记录，记录项目调试要点，方便以后时间久了查以前的调试记录，或者找相关的文件资料。

**二. 调试记录:**

**1. InnCalbe简介**

InnCable的定位大概是广告机。比如公共场所的电梯里面的广告机，广告机的工作模式就是不停的自动播放广告，或根据需要在指定的时间段播放广告。

这个项目调试过的驱动：

1. AM1805 RTC实时时钟支持
2. USB Vbus power控制提供出接口
3. Uart0 ttlen pin脚控制提供出接口
4. 客户chromecast导致usb以太网不能上网问题

S905X\_B 项目相关调试资料

\\192.168.1.8\work\home\zkh\Amlogic\项目资料\S905X

版本信息：

\\192.168.1.5\firmware\m9\_release\version\s905x\_dongle\version.ver

dts：gxl\_stvs9\_dongle\_2g.dts

烧录方法：

全烧：

烧录工具调试资料目录下：USB\_Burning\_Tool

单烧：

1. 开机串口敲进入uboot模式：

烧录硬件版本信息：usb start;fatload usb 0 12000000 version.ver;hw\_write 12000000

boot.mg烧录：usb start;fatload usb 0 $loadaddr boot.img;store write boot $loadaddr 0 2000000

dtb.img：usb start;fatload usb 0 1080000 dtb.img;store dtb write 1080000;

**2. AM1805调试**

AM1805是集RTC、Watchdog、Alarm、Countdown四大功能于一身，功能齐全的实时时钟芯片，包含片上振荡器以提供最低的功耗，完整的RTC功能，包括电池备份，可编程计数器和定时器和看门狗功能的报警，以及用于与主机控制器通信的I2C或SPI串行接口。 集成电源开关和具有计数器，定时器，闹钟和中断功能的复杂系统睡眠管理器使AM18X5可用作基于主机微控制器系统的监控组件。

SDK中默认没有该IC的驱动支持，所以里面的驱动是根据规格书编写的。

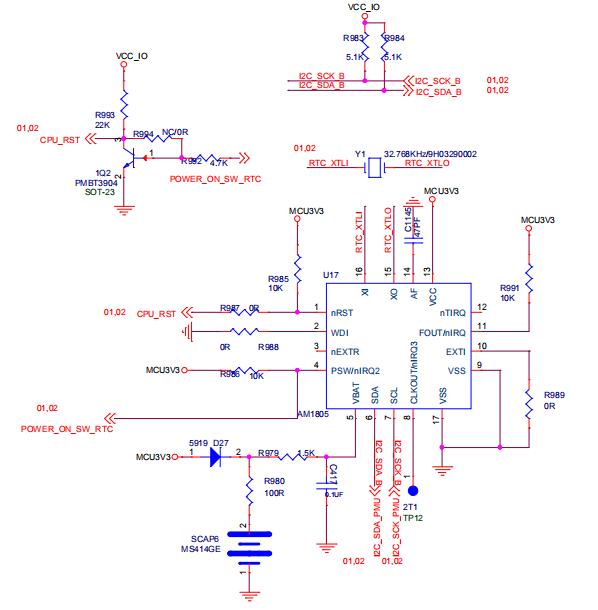
AM1805相关资料：

\\192.168.1.8\work\home\zkh\Amlogic\项目资料\S905X\AM1805

驱动编写的调试步骤：

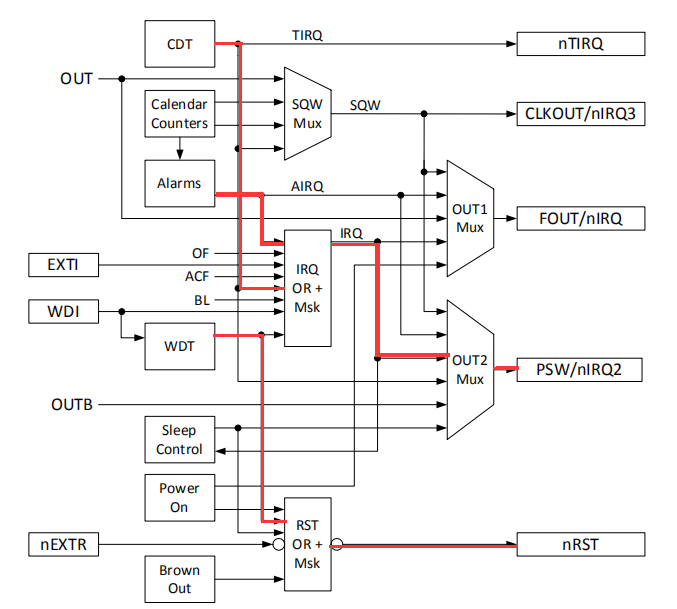
1. 调通I2C，能通过I2C读写寄存器。
2. 做调试工具，驱动里面写一个sys控制寄存器，实现读写单个寄存器和打印全部寄存器。然后通过串口命令行用cat和echo直接去读写寄存器，并在调试RTC/ALARM/COUNTDOWN/WATCHDOG之前，先用读写寄存器命令验证功能。
3. 根据规格书提供的寄存器资料，开始通过直接读写寄存器，去先检验RTC、Alarm、Watchdog、Countdown功能，确认用寄存器能调试没问题后，在驱动里面把相关代码写好。
4. 调完后，写一个用户空间的程序，通过IOCTL来测试驱动。
5. 校正RTC外部晶振

硬件原理图接线方式：



硬件上的特性：

硬件原理图上的中断输入WDI没有接，中断输出接的有nRST、PSW/nIRQ2，其他中断引脚nTIRQ、CLKOUT/nIRQ3、FOUT/nIRQ。目前软件Watchdog中断输出走的是nRST，其他Alarm、Countdown走的是nIRQ2中断脚。功能总述为下图中红色的走线，CDT和Alarm到或门控制输出到nIRQ2，WDT到或门输出到nRST：

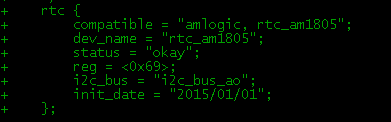


前期调试：

由于硬件上VCC直接上电，所以不用去控制IO上电。

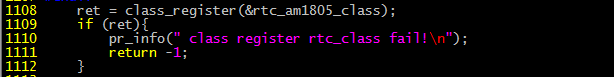
AM1805规格书I2C地址是0x8c/0x8d，这个地址包含读写位，bit0是读写位，所以bit1~7是实际的I2C地址，0x8c/0x8d>>1可以算出I2C地址是0x69。I2C实际挂载为i2c ao，不是原理图的i2c b。

在dts中加上RTC的配置编译烧录了dtb后，在驱动的probe函数中加读取chip id并打印，看到I2C通了。

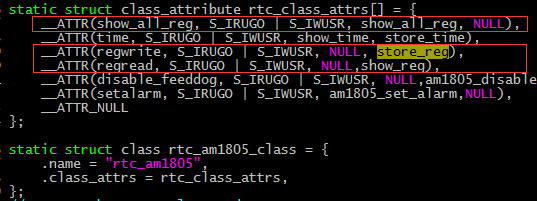


由于验证功能需要频繁读写寄存器，所以注册了一个class直接操作寄存器。目前调试或者查问题都可以通过命令行直接操作class读写寄存器，方便快速定位问题。

驱动里面通过class\_create 来创建一个设备的逻辑类：



然后创建三个功能的属性：读单个寄存器，写单个寄存器，读所有寄存器。这三个属性目前基本能满足这个芯片的调试。



**使用示例：**

这里举例在命令行直接设置一个闹钟Alarm

关闭AIE使能，设置IM为电平模式

echo "0x12 0xe0">/sys/class/rtc\_am1805/write\_reg

清除ALM Status

echo "0x0f 0x00">/sys/class/rtc\_am1805/write\_reg

设置为IRQ2中断

echo "0x11 0x0c">/sys/class/rtc\_am1805/write\_reg

设置时间

echo "0x08 0x01">/sys/class/rtc\_am1805/write\_reg

echo "0x09 0x01">/sys/class/rtc\_am1805/write\_reg

echo "0x0a 0x50">/sys/class/rtc\_am1805/write\_reg

echo "0x0b 0x00">/sys/class/rtc\_am1805/write\_reg

echo "0x0c 0x02">/sys/class/rtc\_am1805/write\_reg

echo "0x0e 0x01">/sys/class/rtc\_am1805/write\_reg

一年一次

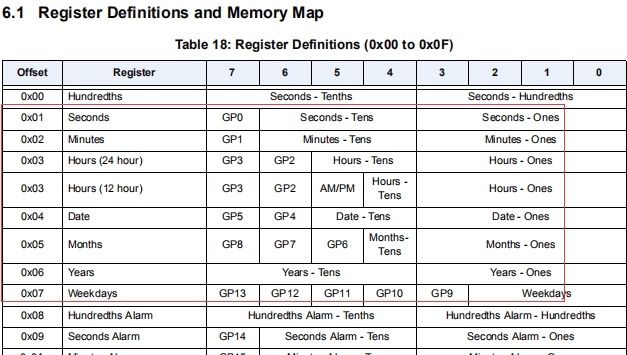
echo "0x18 0xc6">/sys/class/rtc\_am1805/write\_reg

启动

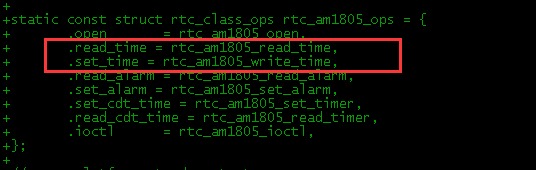
echo "0x12 0xe4">/sys/class/rtc\_am1805/write\_reg

**RTC时间调试**

RTC可给系统提供时间，当系统关机后，RTC芯片由于有电池供电，时间就不会停止，这样下次设备开机后就知道时间。RTC时间由于只需要读写时间相关的寄存器，直接在命令行用echo设置一下相关寄存器，可直接设置时间，然后在驱动里面实现RTC驱动提供的接口。寄存器：



实现的接口：



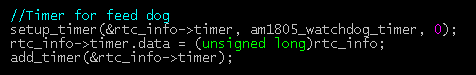
**watchdog调试**

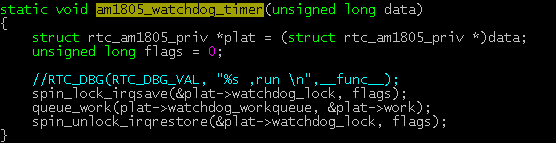
看门狗是用来定期监控系统的运行情况，一旦系统死机，看门狗就发出重启电路的信号。用途上比如广告机，我们需要它一直播放广告，如果突然死机了怎么办，所以我们就通过看门狗来解决，死机后看门狗自动重启设备，然后继续播放广告。

由于原理图上没有WDI中断输入，所以需要驱动模拟实现。模拟喂狗驱动上用一个计时器和一个工作队列来实现。计时器每隔一定时间发出一个需要喂狗的工作，工作队列对看门狗倒计时的寄存器重新填值。

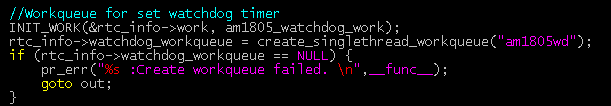
目前我在驱动上设计喂狗的时间为2秒，超时的时间4秒。也就是4秒内没有喂狗，看门狗就产生一个中断到nRST引脚复位设备。

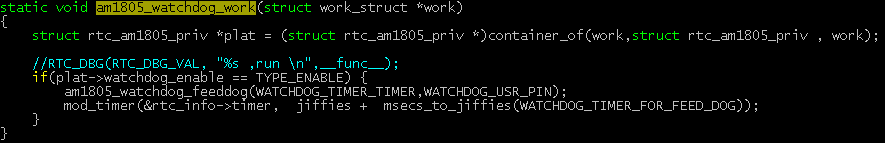
目前驱动里面计时器所做的内容就是添加一个工作到队列里：





工作的内容就是喂狗：

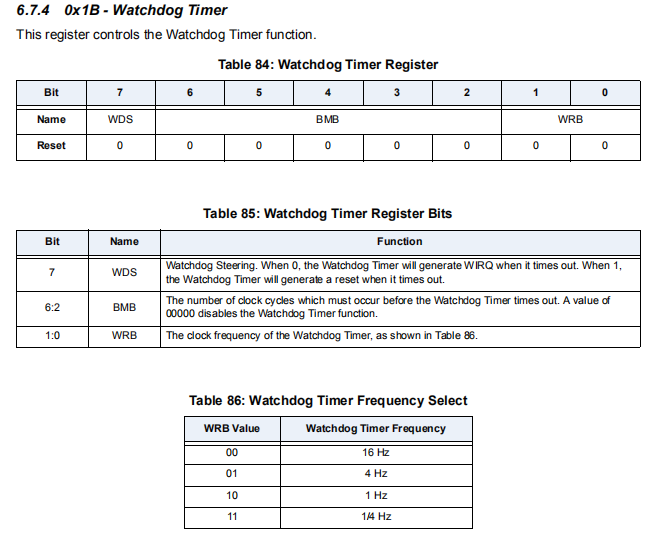




工作流程：第一次add\_timer的时候会启动一次计时器，计时器会添加一个工作到工作队列，工作队列填完值后，mod\_timer会改动计时器的启动时间，也就是2秒，2秒后会再次进入计时器函数，然后计时器又添加一个工作到工作队列，如此循环。也就是不停的喂狗。

假如kernel跑飞了，这时驱动不会去喂狗了，这时看门狗的计时器会自动减到0，进而出发电平中断到nRST引脚，设备就会复位。

寄存器：



第七个bit是使能watchdog的，第6到2bit是设置计时器超时时间，第0到1bit是时钟频率选择的。

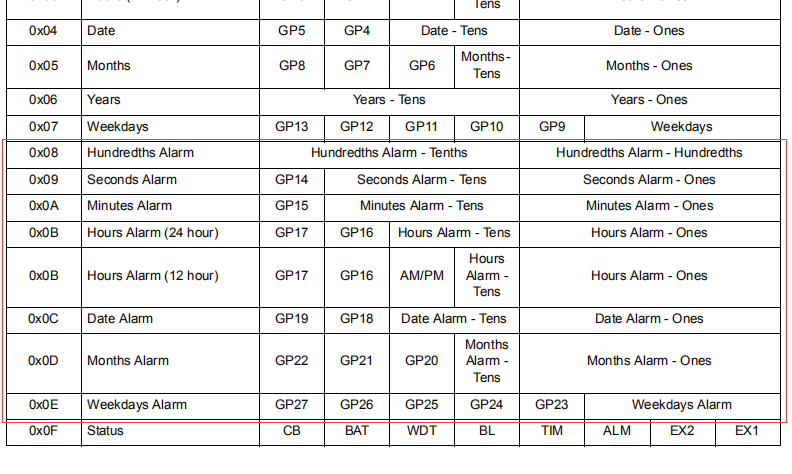
Watchdog测试  
    串口命令行中输入命令su;然后输入echo 1>/sys/class/rtc\_am1805/disable\_feeddog,设备能够自动reset重启

**Alarm/cdt 调试**

cdt/alarm都是为了在nIRQ2上输出想要的电平或脉冲中断。现在调试的Alarm功能是为了实现定时开机的功能。当然，这个功能也可以用CDT（计时器来做），但ALARM和CDT不同的是设置时间的方式，Alarm可直接设置为我们熟悉的时分秒的时间格式，但CDT只能设置多久后触发中断，不能直接指定时间。

硬件上我们需要从nIRQ2引脚输出alarm的中断，中断的类型可以分为电平和脉冲。

alarm时间的寄存器，当设置对应的时间并启动alarm功能后，IC会自动对比RTC时间和alarm时间，如果对应的寄存器值都匹配，就会发出中断。



设置的相关的寄存器，然后用示波器量中断脚，看看时间到了后是否有中断输出。



**AM1805校正外部晶振**

由于实测RTC时间1天会慢30秒左右，所以要做外部晶振校准，把晶振频率调到16.384KHz。

校准参考资料目录的《T0001\_RTC外部晶振校准.pdf》文档。计算公式的参考值是16.384KHz。做校准先根据例程am\_config\_sqw输出方波，方波输出到CLKOUT或者FOUT引脚上，板子上现在有一个测试点，是连接到CLKOUT上，也可以输出到这个引脚。

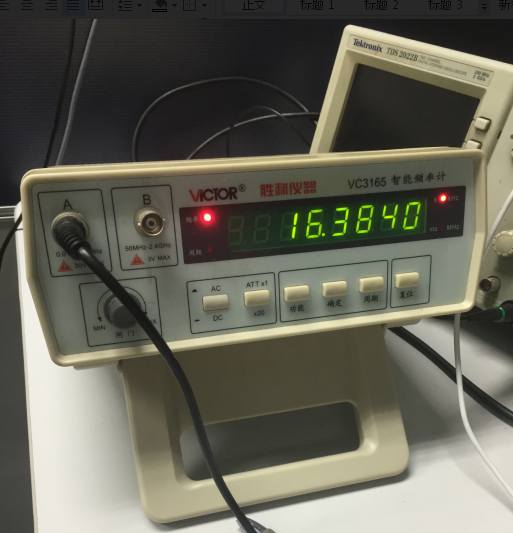
方波输出后，然后用频率计测量频率，注意频率用示波器测不准，因为示波器的精度只能到小数点后两位，所以要用专门的频率计，频率计需要向供应商FAE借。

**校准值计算方法**：

假如用频率计量到的频率为16.388Khz，则可用下面计算公式计算出值来，基准值是16.384

(（16.384-16.388）/16.384)\*1000000=-244.14

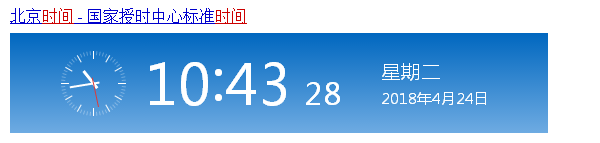
计算出的校准值小数点可以不记。将-244填入到驱动H文件的宏 CALIBRATION\_VALUE，编译烧录到板子后再量频率，正常的话应该是16.384KHz。

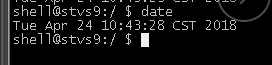


H头文件里面校准控制还有另外一个宏 CALIBRATION\_POSITIVELY\_BIASED\_VALUE，这个宏是微调值。添加这个微调值是因为当初用-244的时候，一直调不到16.384KHz，查代码的二进制计算过程，发现-244在计算校准值时，中间判断会通过switch case，>-129会进入一个case，否则进入另外一个case，而我们计算的值刚好是128.4左右，这时代码是进入>-129部分，但调出来的频率，没有达到效果，所以根据实际情况，加了一个这个宏，将这个宏微调加3到5左右试试，这样switch就会进入另一个case。目前这个值设置的是5，测试频率看到的是16.384。

另外，矫正后时间是否精准，需要把板子上的时间先网络同步，时区设置里面调一下+8时区，时间同步完后要把网络关掉，不然下次开机后会自动同步，这样可能会导致看不到是否校准了。参考时间尽量用网络时间，尽量不要用电脑上的时间，因为电脑上的时间可能也是不精准的，但网络时间一般是原子钟的，时间比较精确，RTC校准后把板子放几天，然后看板子上RTC跑的时间和网络时间，秒数对得上就可以了。

如图，RTC跑了72小时左右后的时间和网络时间对比：





由于晶振用料的好与不好的差异会直接导致时间偏差，所以不同项目的板子都要做个时间测试，就是放个几天然后看时间偏差，如果有偏差，需要到供应商FAE借频率计调一下。

**AM1805已知问题：**

1.安卓时钟应用设置闹钟，时间到了后APK会提示，但用示波器量不到中断出来。

这个问题是，驱动已经实现了alarm的syspend和resume函数，但系统没suspend的时候，设置闹钟并不会往rtc芯片的寄存器上写数据，因为不需要唤醒系统。上层直接把闹钟设置写到/dev/alarm，AlarmThread 会不停的去轮寻下一个时间有没有闹钟。系统要是进入susupend的话，alarm 的alarm\_suspend就把设置的闹钟写到驱动 rtc芯片的寄存器上去，然后即使系统suspend之后，闹钟通过rtc也能唤醒系统。

**3.USB VBUS Power/UARTA0 TTL\_EN控制**

这个功能的控制主要是pin脚提供接口出来，现在是做一个pin脚的class控制，通过echo可验证功能。

Usb vbus控制方法：  
    串口进入命令行下，输入下面命令控制

    1.拉低：su;echo 0x00 > /sys/class/usb\_vbus\_pin/vbuspin

    2.拉高：su;echo 0x01 > /sys/class/usb\_vbus\_pin/vbuspin

Uart0 ttlen控制方法：  
    串口进入命令行下，输入下面命令控制

    1.拉低：su;echo 0x00 > /sys/class/uart\_ttl\_pin/ttlenpin 进入RS232模式

    2.拉高：su;echo 0x01 > /sys/class/uart\_ttl\_pin/ttlenpin 进入TTL 模式

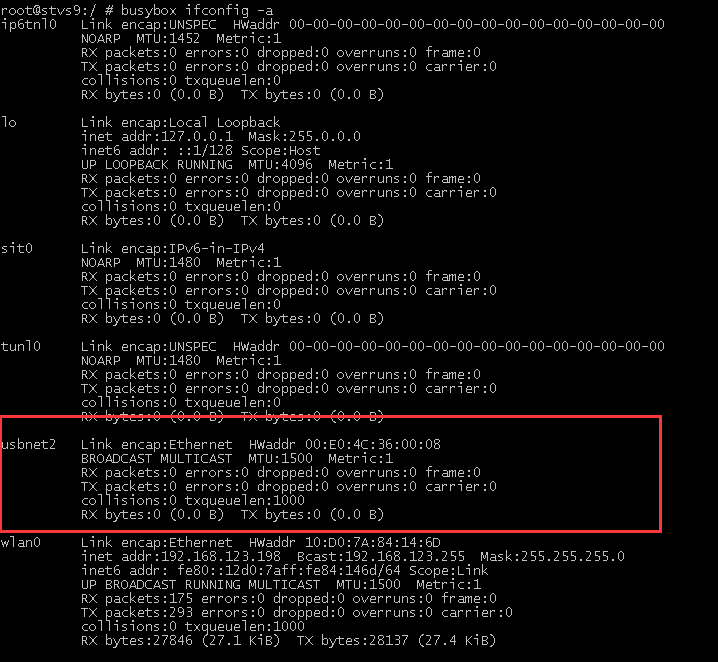
发送数据 echo “字符串” > /dev/ttyS2

**4.客户chromecast和usb以太网同时接入后开机以太网连接不上问题**

同时接入后，开机后，busybox ifconfig -a 可以看到usb 以太网的网络节点是usbnet0，chromecast的网络节点是usbnet1。Usbnet0的ip地址可以获取到，但是不能上网，设置里面以太网选项，点击查看ip是没有获取到的。Logcat看到是usbnet0先加载，usbnet1后加载，usbnet1的相关配置会冲掉usbnet0的配置，导致以太网不能上网。

解决方法是底层将usb网卡口默认为usbnet0，另外两个usb口接的chromecast固定为usbnet1和usbnet2。

Framework层修改代码保证usbnet0的相关配置不会被usbnet1或usbnet2的值覆盖掉。单独接chromecast的 时候，也会将将网络节点固定为usbnet1或者usbnet2，例如：



**三. 技术总结**

主要描述此问题相关技术总结, 可以有哪些扩展的应用, 可以用来解决哪些其它的问题.

文档整体描述简洁明了, 可以插入代码及图片以助理解.