I2C部分

裸机

在realtouch\drivers\stm32\_i2c.c 找到一个貌似可以用的i2c驱动。但是在我移植24C02程序过来之后发现不能使用。发现从机没有应答。

于是找原板程序测试，还是不行。明天再测试。找找程序哪里有问题。

经过测试原板程序没有问题，那么问题就在rtt的驱动上面了，手里面没有逻辑分析仪，这个就麻烦了。。。。。买一个？还是自己用STM32做一个简单的东西出来玩玩？？哈哈。。。。。玩玩吧，反正也是在玩。

在网上找到一个开源的逻辑分析仪PC端，只需要移植STM32端就可以了。也在rtt上面做吧，顺便可以熟悉熟悉rtt。做一个东西真的不容易啊。。。。好多东西要学。。不过，呵呵。

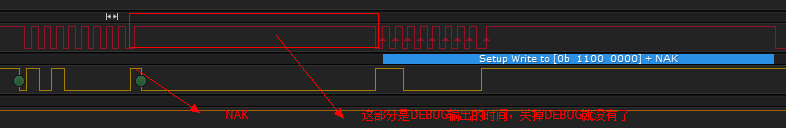
决定开始移植逻辑分析仪。。。。。。。。可以参考网上开源的代码。但是不是我说，那份代码写得太恶心了。。简直没法看了。

把代码封装好，把前几天看的一些思想用进去。。。加油。

I2C BUS (模拟)

20150815 这个部分完成：

下图是写好GPIO驱动之后用逻辑分析仪捕获到的图



从图中可以看出第一次传输数据时没有接收到ACK，但是用开发板自带的程序测试是可以通过的，所以这里是程序没有处理好。

1. GPIO没有进入输入模式。
2. rtt 驱动代码处理有问题。

最后在反复定位到i2c-bit-ops.c中的

rt\_inline rt\_bool\_t i2c\_waitack(struct rt\_i2c\_bit\_ops \*ops) 函数，发现这里直接读SDA的值，然后怀疑这里没有让GPIO进入输入模式，于是在函数开始的地方加入如下红色代码，就能成功收到ACK信号。

rt\_inline rt\_bool\_t i2c\_waitack(struct rt\_i2c\_bit\_ops \*ops)

{

rt\_bool\_t ack;

SDA\_H(ops);

i2c\_delay(ops);

ack = !GET\_SDA(ops); /\* let stm32 gpio into input mode \*/

if (SCL\_H(ops) < 0)

{

bit\_dbg("wait ack timeout\n");

return -RT\_ETIMEOUT;

}

ack = !GET\_SDA(ops); /\* ACK : SDA pin is pulled low \*/

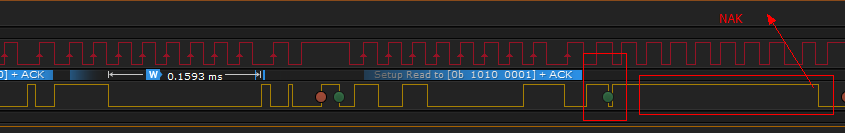
bit\_dbg("%s\n", ack ? "ACK" : "NACK");

SCL\_L(ops);

return ack;

}

下载代码后发现还是没有成功，逻辑分析仪找到的图如下：



发现分析仪根本没有分析这部分（时钟没有箭头），而且可以看到这里的读到的数据全都是1，说明这里可能和上面一样，没有让GPIO进入输入模式就直接读数据了。于是想上面一样在代码中加入红色代码，成功。

static rt\_int32\_t i2c\_readb(struct rt\_i2c\_bus\_device \*bus)

{

rt\_uint8\_t i;

rt\_uint8\_t data = 0;

struct rt\_i2c\_bit\_ops \*ops = bus->priv;

SDA\_H(ops);

GET\_SDA(ops);/\* let stm32 gpio into input mode \*/

i2c\_delay(ops);

for (i = 0; i < 8; i++)

{

data <<= 1;

if (SCL\_H(ops) < 0)

{

bit\_dbg("i2c\_readb: wait scl pin high "

"timeout at bit %d\n", 7 - i);

return -RT\_ETIMEOUT;

}

if (GET\_SDA(ops))

data |= 1;

SCL\_L(ops);

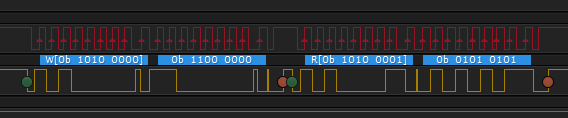
i2c\_delay2(ops);

}

return data;

}

逻辑分析仪图像如下：



修正一个我的错误，i2c-bit-ops.c没有问题，而是我不熟悉寄存器写法而造成把GPIO切换到输入模式的时候出了问题，换成库函数写法就没有问题了。代码如下，完整参考stm32\_i2c.c

static void SDA\_IN(void)

{

GPIO\_InitTypeDef GPIO\_InitStructure;

GPIO\_InitStructure.GPIO\_Pin = PIN\_I2C\_SDA ;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_IN\_FLOATING; // 浮空输入

GPIO\_Init(GPIO\_PORT\_I2C\_SCL, &GPIO\_InitStructure); // 选择C端口

}

static void SDA\_OUT(void)

{

GPIO\_InitTypeDef GPIO\_InitStructure;

GPIO\_InitStructure.GPIO\_Pin = PIN\_I2C\_SDA ;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_Out\_OD; // 开漏输出

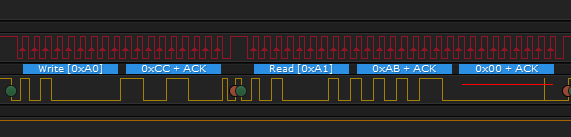
GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz; // 最高输出速率50MHz

GPIO\_Init(GPIO\_PORT\_I2C\_SCL, &GPIO\_InitStructure); // 选择C端口

}

I2C BUS (硬件)

因为miniSTM32战舰 上面的24C02是GPIO，要做硬件实验只能外接一个24C02，然后发现用上面的程序数据写不进去，参考24C02时序，做了简单的修改就ok了。捕获到的逻辑图如下：



但是发现每次都会读两个byte，不知道为什么。测试可以对24C02读写。说明I2C协议没有问题。可以进行下面的实验了。

MPU6050

测试的时候发现硬件有问题 SCL 和SDA 两根线画反。改正过来之后网上找了一个程序，硬件，软件I2C都能够读取到数据。相关程序参考本目录下的相关工程，看名字就知道。

接下来等I2C BUS 搞定之后就能够集成到RTT上面去了。

I2C BUS 搞定，顺便去github上面找到了mpu6050的驱动代码，在代码里面mpu6050被封装成了一个device注册到了rtt中，我在想，是不是nrf2401也应该做成一个device呢??也许在github上面也能找到源码。

这里源码用模拟I2C可以用，中途还遇到I2C不能用的问题，郁闷，调了好久之后发现是在把GPIO切换成输入模式的时候有问题。。。。。。修正后修OK了。但是用硬件I2C就不行，网上说是stm32f10x的ack时间太短的原因。那暂时不用吧。

mpu6050的原始数据也能读出来了。接下来就是通信和控制了。