**20120806:**

bypass : 是否绕过当前设备，即一个设备如果被设定为bypass则这个设备的将被忽略为一根"导线",如分频器

gating : 开关

**20120807：**

1、EEPROM学习

管脚定义：

A2-A0 ：器件地址选择引脚

sda：i2c数据线

scl：i2c时钟线

wp： 硬件写保护控制线

gnd：地

vcc：电源

通过I2C读写:

2、\_\_WFI

#define \_\_WFI \_\_wfi

#ifdef TEST\_EN

// Wait For Interrupt

\_\_WFI(); // Enter sleep mode

// Wakeup when I2C interrupt is triggered

#endif

WFI suspends execution until one of the following events occurs:

a、an IRQ interrupt, regardless of the CPSR I-bit

b、an FIQ interrupt, regardless of the CPSR F-bit

c、an Imprecise Data abort, unless masked by the CPSR A-bit

d、a Debug Entry request, regardless of whether Debug is enabled.

**20120810：**

KEIL同时挂多个jlink:

在option->debug->settings->jlink cmd 输入命令行：USBADDR = <address>

**20120811：**

1、i2c slave产生ack后 sda会产生一个脉冲的原因

原因是slave产生ack后会主动释放sda数据线，因为sda是会有一个上拉电阻，所以会产生一个高脉冲

2、rtc 小结

a、写rtc寄存器之前之后需要同步（waitting）的原因：

//while (rtc\_rtc\_GetCR(QN\_RTC) & RTC\_MASK\_CR\_SYNC\_BUSY);

//rtc\_rtc\_SetCRWithMask(QN\_RTC, RTC\_MASK\_MIN\_IE|RTC\_MASK\_SEC\_IE, 0);

//while (rtc\_rtc\_GetCR(QN\_RTC) & RTC\_MASK\_CR\_SYNC\_BUSY);

原因为：rtc本身的寄存器的时钟为低速时钟（32768Hz），但是对外开放给用户的rtc寄存器“镜像”所使用的是高速时钟，这样一来就存在了高速时钟域寄存器到低速时钟域的转换了，也就产生了同步（waitting）。QN9020 rtc的同步一章很清楚的介绍了原理。

b、calibration 校准原理

待解决

**20120815：**

1、rvct指令内在函数

void \_\_breakpoint(int val); //设置手工断点

unsigned int \_\_current\_pc(void); //返回当前位置pc值

unsigned int \_\_current\_sp(void); //返回当前sp值

void \_\_nop(void); //空操作

unsigned int \_\_return\_address(void); //此内在函数返回当前函数的返回地址。

void \_\_wfi(void); //wait for interrupt

void \_\_wfe(void); //wait for event

int \_\_disable\_fiq(void);

int \_\_disable\_irq(void);

void \_\_enable\_fiq(void);

void \_\_enable\_irq(void);

2、rvct内联汇编、嵌入式汇编

A、内联汇编器只支持 ARM 汇编语言，而不支持以下语言和指令：

a、Thumb 汇编语言

b、Thumb-2 汇编语言

c、ARMv7 指令

d、VFP 指令

e、NEON 指令。

B、可以使用嵌入式汇编器来支持 Thumb 和 Thumb-2。

C、内联汇编器：

int f(int x)

{

int r0;

\_\_asm ;单行或多行书写

{

ADD r0, x, 1

EOR x, r0, x

}

return x;

}

D、嵌入式汇编器：

\_\_asm void my\_strcpy(const char \*src, char \*dst)

{

loop

LDRB r2, [r0], #1 ;注释：src == r0

STRB r2, [r1], #1 ;注释：dst == r1

CMP r2, #0

BNE loop

BX lr ;注释：如果需要返回值 则把相应值放入到r0中

}

3、再谈\_\_wfi \_\_wfe

WFI/WFE指令进入Idle模式。一旦进入Idle模式，CPU将立刻进入Sleep模式而停止运作，因此在WFI /WFE指令之后的程序都必须等到CPU被唤醒后才会被执行。此时NUC1xx的耗电量将降到约16mA@3.3V-48MHz。

　　进入Idle省电模式之后，如果要唤醒CPU，回到一般的工作模式，就必须先产生中断信号。该中断信号可以是NUC1xx中的任何模块所产生的中断，也可以通过外部中断的方式来唤醒CPU。需要注意的是，当决定CPU要由某个中断来唤醒时，就必须在进入Idle省电模式前，将该中断设定完成，才能使其产生中断事件来唤醒CPU。

WFI/WFE，低功耗睡眠模式的进入。后者功耗更低。

**20120820:**

1、关于使用for()做延时的一点心得:

a、循环时间基本固定（编译器优化一定的情况下）：

for(time\_out = 0; time\_out < TIME\_OUT\_1SEC\*5; ++time\_out)、

{

}

b、循环时间不能确定：

for(time\_out = 0; time\_out < TIME\_OUT\_1SEC\*5; ++time\_out)

{

if(timer0\_top\_overflow\_int\_occur &&

timer1\_top\_overflow\_int\_occur &&

timer2\_top\_overflow\_int\_occur &&

timer3\_top\_overflow\_int\_occur)

break;

}

为什么不能固定？

答：因为"&&"的原因，如果"&&"前面的内容为FALSE，则后面的判断不会进行。

**20120823:**

1、i2c general call

host产生的地址: 0x00；

所有的slave都返回ack；

host通过general call只能判别总线上是否有设备，而不能判别有多少设备。 因为：低电平为ack，从i2c总线是线与连接的。

**20120827:**

1、i2c 仲裁丢失

多个host同时去访问一个slave设备，其中只能有一个host得到对slave的控制，其他的host就会产生仲裁丢失的现象。

**20120828:**

1、EEPROM写读之间要有一些延时

操作EEPROM时，不能写完之后马上读取，要有一些延时之后再进行读取

使用i2c来操作eeprom时候，记得写之后要有一段时间延时，再进行其他操作，否则eeprom直接返回nack

2、I2c出现的怪异波形的原因

