CNASIC

第六讲 中断与设备驱动（Interrupt & Device Driver）

凌 明

trio@seu.edu.cn

东南大学国家专用集成电路系统工程技术研究中心

中断的分类

- 硬件中断 (Hardware Interrupt):一般是由外部（相对CPU内核而言）的硬件引起的事件，比如串口来数据，键盘击键等；
- 软件中断（Soft Interrupt）：通过在程序中执行的中断指令引起的中断，又叫软陷；
  - 80X86：int 指令
  - 68000：trap 指令
  - ARM： SWI 指令
  - 软中断指令一般用于操作系统的系统调用入口；
- 异常（Exception）：由于CPU内部在运行过程中引起的事件，比如指令预取错，数据中止，未定义指令等等，异常事件一般由操作系统接管。

中断的处理过程

- 虽然中断产生的原因不同，但是中断响应的硬件过程基本上是相同的。
  - 拷贝CPSR到SPSR_<mode>
  - 设置正确的CPSR位
    - 切换到ARM状态
    - 切换到异常模式
    - 禁止中断
  - 保存返回地址在LR_<mode>
  - 设置PC到异常向量地址

硬件完成

  - 中断服务程序可能保存需要使用的寄存器（堆栈中）
  - 用户服务程序可以打开中断，以接受中断嵌套
  - 恢复保存的寄存器
  - 通过调用Reti（或其他相应指令）指令将PSR和PC出栈，从而恢复原来的执行流程。

中断服务程序完成

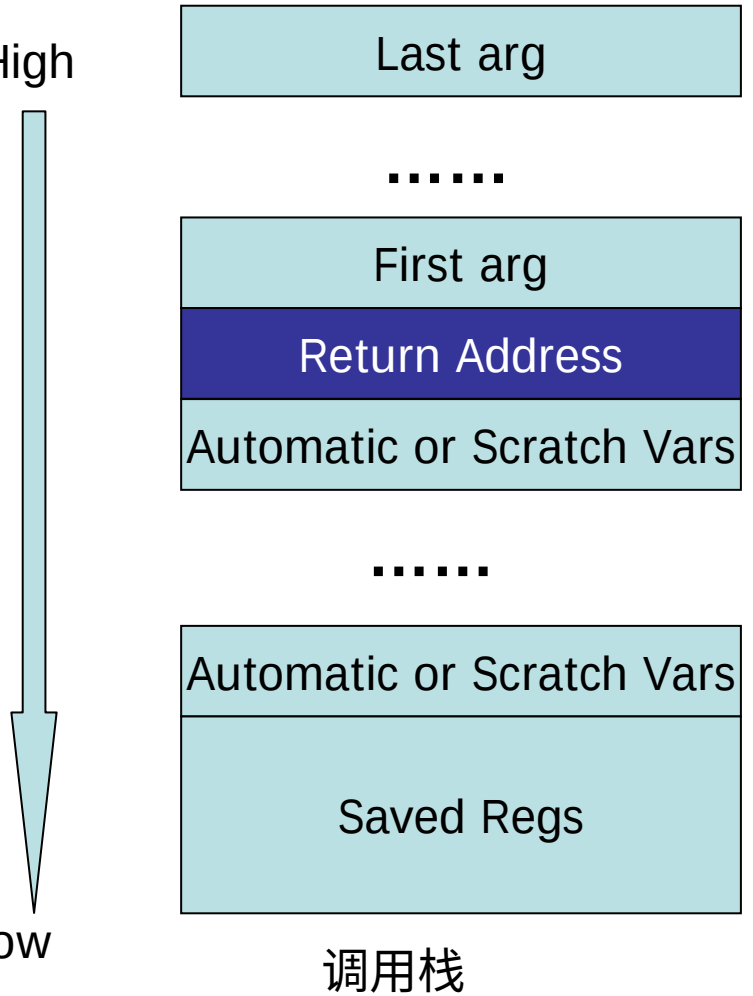
中断的重要性！！

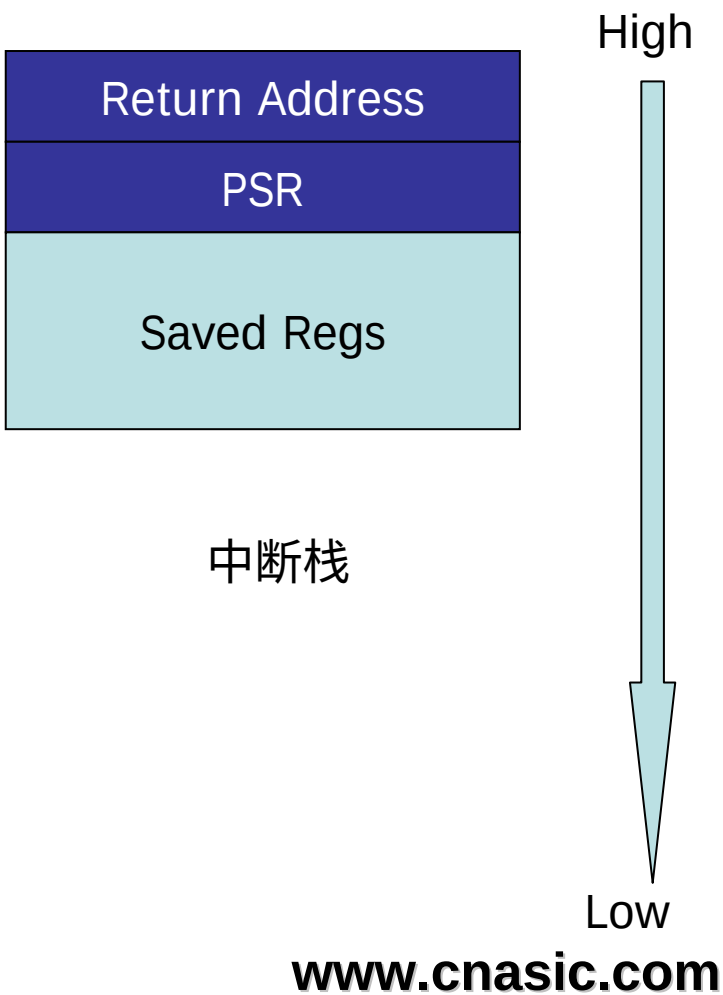
如何强调中断的重要性都不过份，我们将在内核的实现中详细介绍！

- 理解处理器对中断的管理以及这其中的堆栈管理对于理解操作系统是至关重要的！
- 中断是操作系统的入口，用户访问操作系统提供的服务的唯一途径是依靠中断来实现的。
- 实时系统对异步事件的处理，依靠的是中断！
- 任务的调度靠的是中断
- 系统调用的实现靠的是中断
- 在有MMU的系统中，虚存的管理也是依靠中断！
- 中断是理解操作系统的入口！！

CNASIC

调用栈和中断栈的不同





C语言中的中断处理

- 在标准C中不包含中断。许多编译开发商在标准C上增加了对中断的支持，提供新的关键字用于标示中断服务程序 (ISR)，类似于__interrupt、#program interrupt
- 当一个函数被定义为ISR的时候，编译器会自动为该函数增加中断服务程序所需要的中断现场入栈和出栈代码。

用C编写中断服务程序应该注意的

- 不能返回值；
- 不能向ISR传递参数；
- ISR应该尽可能的短小精悍；
- printf(char * lpFormatString,…)函数会带来重入和性能问题，不能在ISR中采用。（在ARM平台上由于半主机机制，该函数的速度更慢！）
  - 其实还包括所有的不可重入的函数都不应该在中断中使用。程序员应该仔细地评估ANSI C库函数和OS的系统调用
  - 浮点运算以及其他的耗时操作都不应该在中断程序中使用

加快中断处理程序的方法

- 在中断处理程序中只进行最基本的硬件操作，比如读出硬件寄存器的数据，或者改变状态寄存器的值
- 然后通过一定的方法将中断的事件做一个标志，在离开中断处理程序后，由其他代码根据中断标志进行后续的处理
  - 这样做的好处是大大加快了中断的处理时间
  - 常见的方法：
  - 在没有OS的情况下可以使用自定义的队列，在中断处理程序之外的主循环中对中断的事件进行处理。
  - Linux 下的Bottom half & Top half
  - ASIX OS中的系统任务

CNASIC

没有OS的中断服务队列

```
/* 存放中断的队列 */
typedef struct tagIntQueue
{
   int intType; /* 中断类型 */
   struct tagIntQueue *next;
}IntQueue;

IntQueue lpIntQueueHead;

__interrupt ISRexample ()
{
   int intType;
   intType = GetSystemType();
   QueueAddTail(lpIntQueueHea
新的中断 */
}
```

```
While(1)// 在主循环中检查中断并处理之
{
   If( !IsIntQueueEmpty() )
   {
      intType = GetFirstInt();
      switch(intType) /* 是不是很象WIN32程序的消息解析函数
*/
      {
         /* 对，我们的中断类型解析很类似于消息驱动 */
         case xxx: /* 我们称其为"中断驱动"吧？ */
            …
            break;
         case xxx:
            …
            break;
         …
      }
   }
}
```

有OS的情况下中断将变得更复杂

- 一般而言OS将接管中断向量表，中断发生时，首先由OS接管中断
- OS将检查真正的中断源是什么，然后才调用真正的中断处理程序

CNASIC

ASIX OS下的中断初始化

```
;/*****************************************************
;               file name :             boot.s
;               descrition:             boot the arm processor
;               history:                2003-1-7 15:59 lc create
;*****************************************************/

                include hardware_gfd.h

                extern      main
                AREA BOOT, CODE, READONLY
                    ENTRY                                   ; Mark first instruction to execute
;vector table
                bal                     RST_DO
                bal                     EXTENT_INSTRU
                bal                     SWI_DO
                bal                     ABORT_PREFETCH_DO
                bal                     ABORT_DATA_DO
                mov                     R1,         R1          ;reserved exception
                bal                     Irq_Do
                mov                     r0, r0
                bal                     Fiq_Do
                ;the code for the fiq
```

```
;*****************************************************************************
Irq_Do
stmfd       sp!, {r0,r1}
ldr         r0, =IRQ_R1
str         r1, [r0]
ldmfd       sp!, {r0}
ldr         r1, =IRQ_R0
str         r0, [r1]
add         r13, r13, #4;           //restore the sp_irq top to original irq top
sub         r14, r14, #4
mov         r0, r14
mrs         r1, spsr
orr         r1, r1, #0x80
msr         cpsr_cxsf, r1;          //change irq mode into svc
;-------------------------------------------
bic         r1, r1, #0x80;          //clear the irq mask
stmfd       sp!, {r0}
stmfd       sp!, {r14}
stmfd       sp!, {r1}
ldr         r0, =IRQ_R1
ldr         r1, [r0]
stmfd       sp!, {r1}
ldr         r1, =IRQ_R0
ldr         r0, [r1]
stmfd       sp!, {r0}
ldmfd       sp!, {r0,r1}
stmfd       sp!, {r0-r12};          //save the registers r0--r12
IMPORT      int_vector_handler
bl          int_vector_handler
```

```
void (*IntHandler[32])(void)={
  /*interrupt number and description,  handler     */
  /* 00  INT_NULL,        */                ENT_INT_EMPTY     ,
  /* 01  INT_EXT0,  (PE0)  */                ENT_INT_RING1     ,
  /* 02  INT_EXT1,  (PE1)  */                NULL              ,
  /* 03  INT_EXT2,  (PE2)  */                NULL              ,
  /* 04  INT_EXT3,  (PE3)  */                ENT_INT_RING2     ,
  /* 05  INT_EXT4,  (PE4)  */                NULL              ,
  /* 06  INT_EXT5,  (PE5)  */                NULL              ,
  /* 07  INT_EXT6,  (PE6)  */                ENT_INT_RING3     ,
  /* 08  INT_EXT7,  (PE7)  */                NULL              ,
  /* 09  INT_EXT8
  /* 10  INT_EXT9
  /* 11  INT_EXT1
  /* 12  INT_EXT1
  /* 13  INT_EXT1
  /* 14  INT_EXT1
  /* 15  INT_EXT1
  /* 16  INT_NON
  /* 17  INT_EXT1
  /* 18  INT_EXT1
  /* 19  INT_EXT1
  /* 20  INT_LCD
  /* 21  INT_AC97
  /* 22  INT_PWM
  /* 23  INT_UAR
  /* 24  INT_UAR
  /* 25  INT_MMC
  /* 26  INT_SPI,
  /* 27  INT_USB,
  /* 28  INT_GPT,
  /* 29  INT_EMI,
  /* 30  INT_DMA
  /* 31  INT_RTC,
```

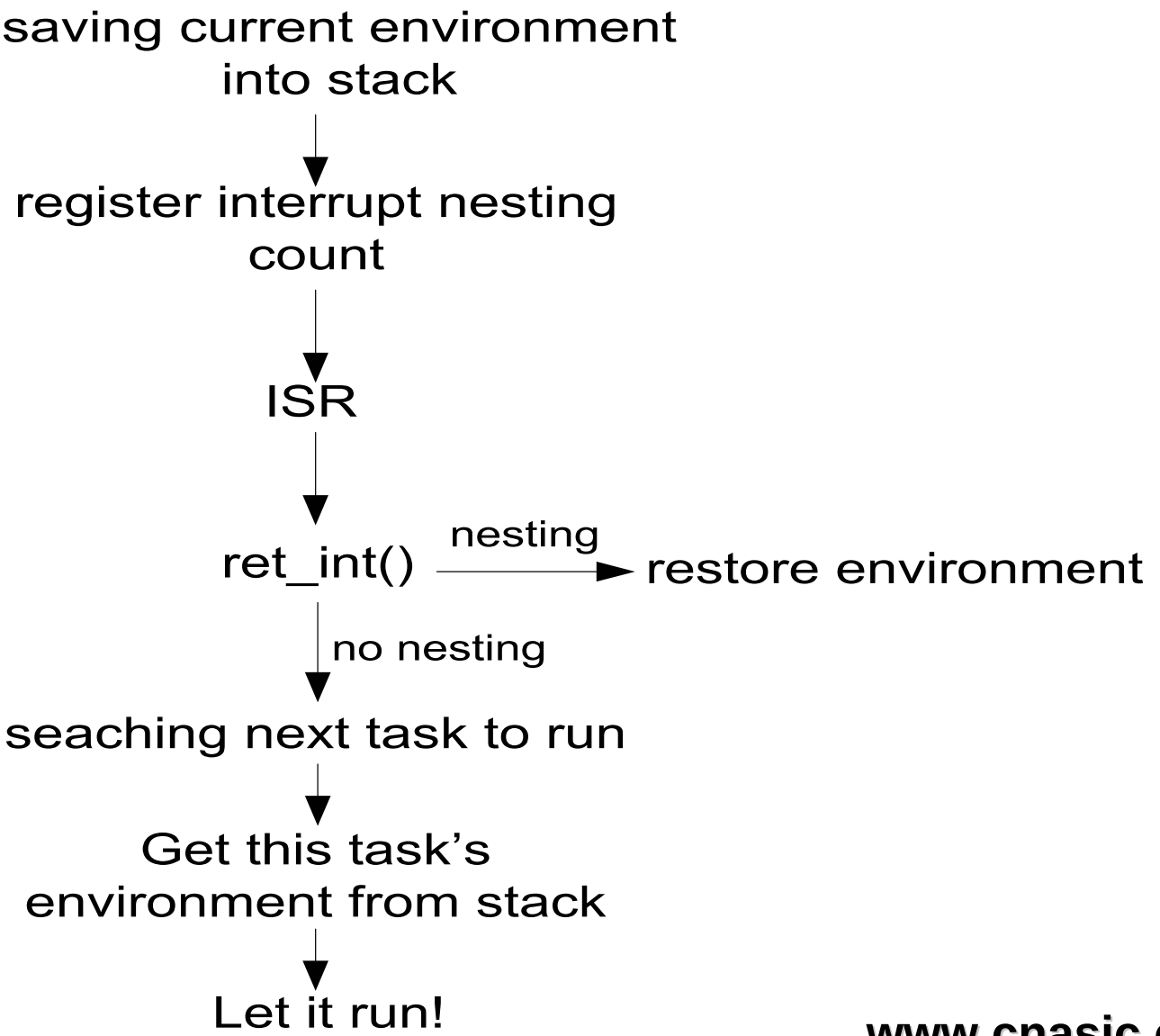
```
void int_vector_handler(void)
{ int i;
  unsigned long IFSTAT=*(RP)(INTC_IFSTAT);
  if(IFSTAT<1)IFSTAT=*(RP)(INTC_ISTAT) & (~*(RP)(INTC_IMSK)) & *(RP)(INTC_IEN);
  if(IFSTAT>1)
  { i=-1;
    while(IFSTAT)
     { IFSTAT>>=1;
       i++;
     }
  }else i=0;

  if(IntHandler[i])(*IntHandler[i])();
  else
  { ent_int();
    printf("No interrupt entry for INT NO.%d\n",i);
    ret_int();
  }

}
```

通知内核！

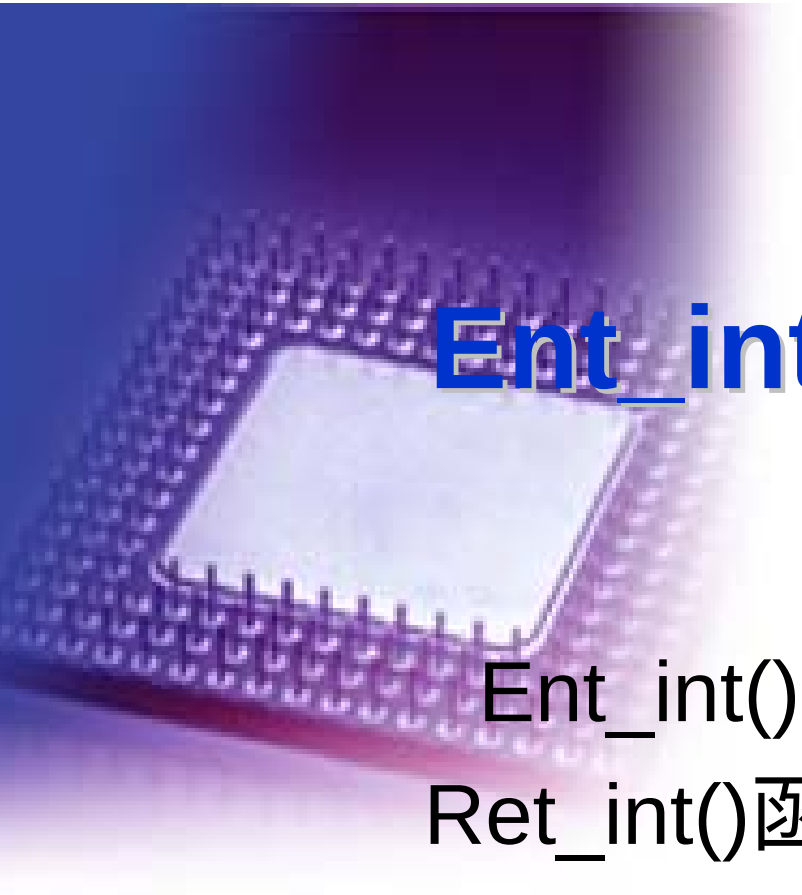
- 为了通知内核中断的发生，往往需要在用户的中断服务程序中显示地调用OS提供的系统调用
  - Ent_int() 通知内核我们现在中断中
  - Ret_int() 通知内河我们离开中断了，如果系统允许则进行调度，否则按照中断的方式离开－返回

CNASIC

Sample



CNASIC



Ent_int()与Ret_int()函数代码分析

Ent_int()函数在ros33.c

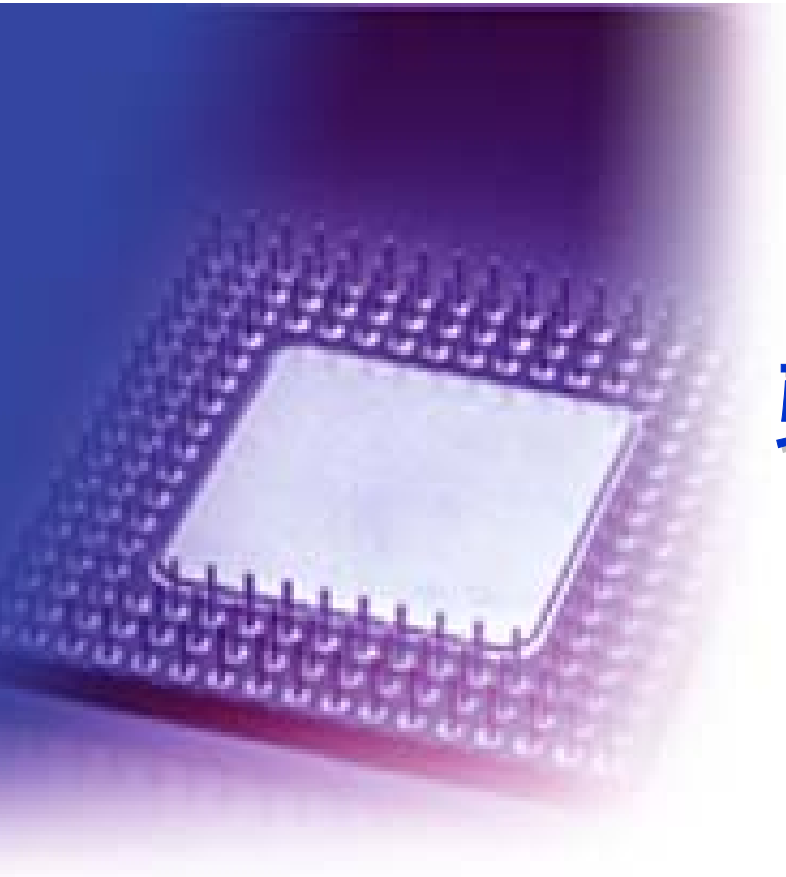
Ret_int()函数在dispatch.c

CNASIC

ASIX OS 中用户ISR的一般格式

```
void ENT_INT_RTC( void ) //int_vector_handler()函数调用
{
   ent_int();//告诉内核，中断发生了

   rtc_isr();//用户真正的中断服务程序

   ret_int();//我们要返回了，或者我们要切换任务了
}
```

CNASIC



驱 动

Timer Manager

- GPT ISR
- sys_clk()
  - 维护系统时间；
  - 维护延迟任务队列，包括恢复到期任务为就绪态，并添加至就绪队列；
  - 维护定时器队列，包括恢复到期定时器至就绪定时器队列；
- ret_int()

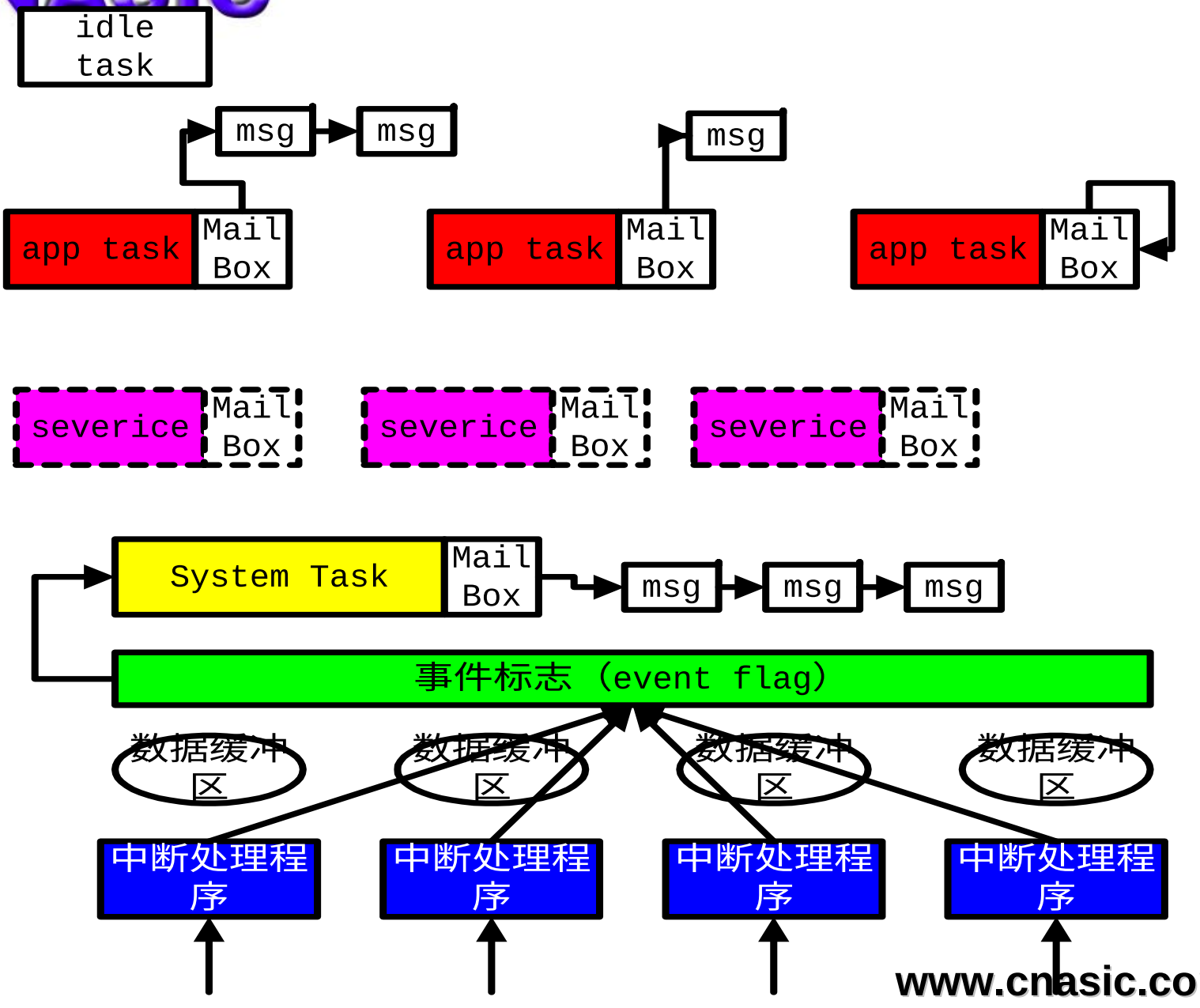
CNASIC

GPT驱动代码阅读

CNASIC

ASIX OS的驱动架构

SYSTASK－系统任务！



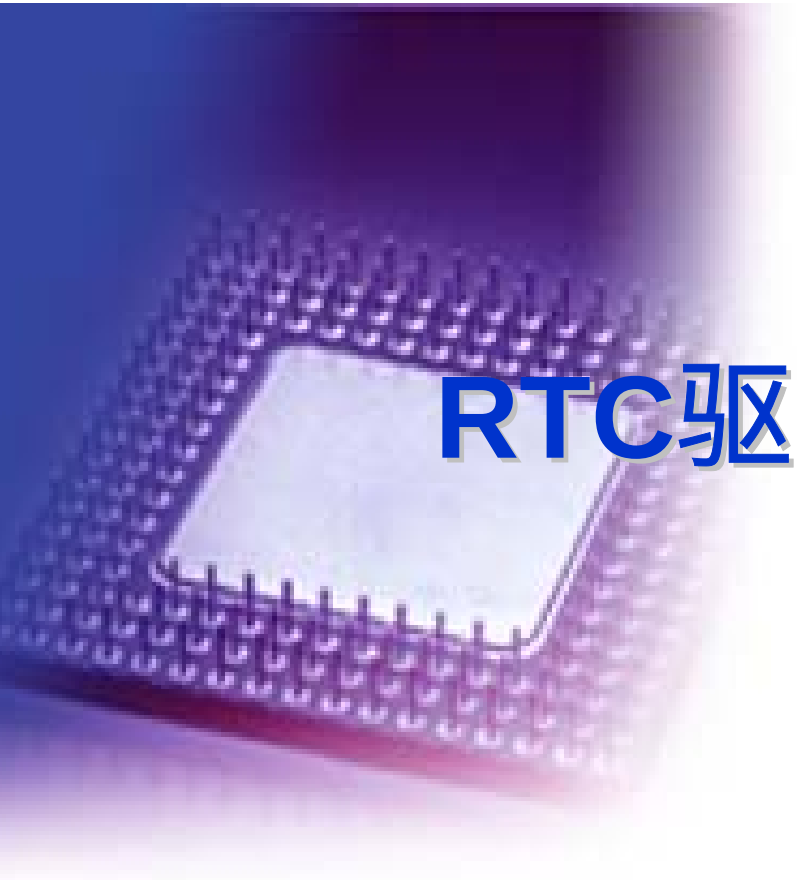
System Task

- Hardware Event
  - Key
  - Pen
    - Active Area
- Task Message
  - Start
  - Switch
  - End

CNASIC

SYSTASK代码阅读

CNASIC



RTC驱动代码阅读

CNASIC

Timer Task

- RTC Event