Tornado中VxWorks内核时钟频率的修改

VxWorks系统必须激活一个时钟中断，作为操作系统内核的时基。时钟中断频率由`SYS_CLK_RATE`指定，这个参数可以重新指定，缺省值为60，VxWorks使用的时间单位为`tick`，一个时钟中断一个`tick`，具体的时间值由前面设置的频率确定。

系统时钟的频率不能太高，高频率会使内核调度时比率偏高，而实际任务可用的CPU时间下降，使整个系统的运行效率下降或不可用。若想使用高分辨率时钟，可以使用时间戳、辅助时钟。

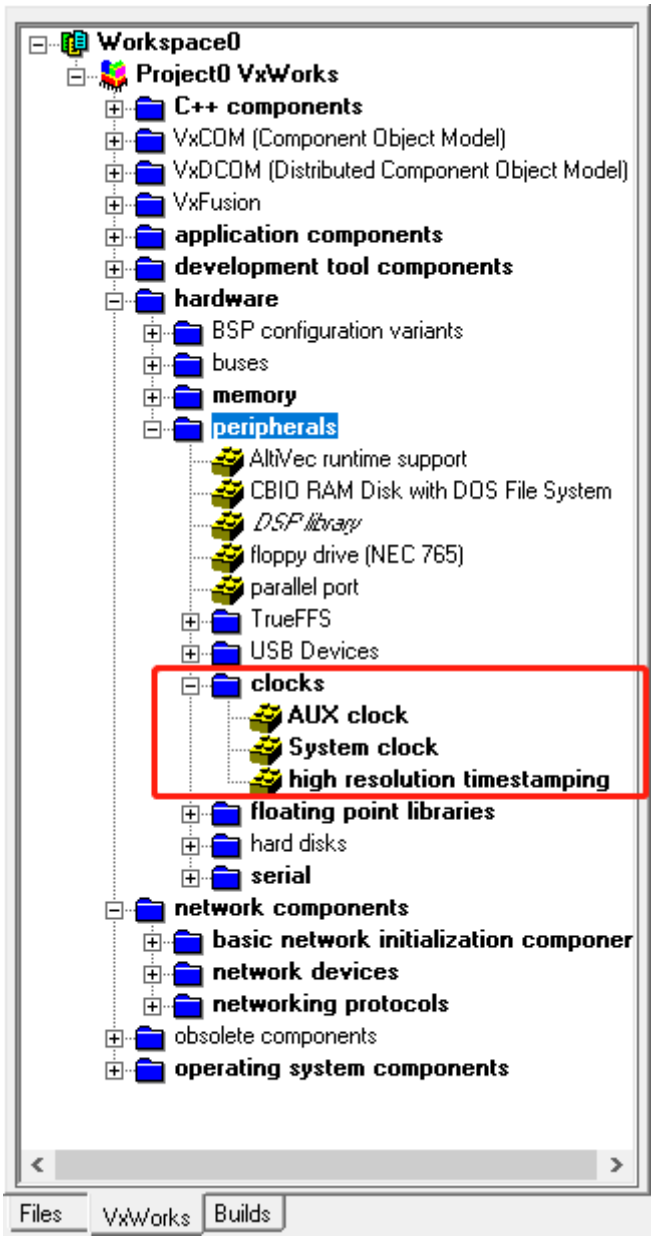
辅助时钟是另外一个获得高分辨率时钟的机制。他启用的是与系统时钟不同的硬件定时器，挂接用户自己的中断处理函数，去掉用户自己的中断处理函数，去掉内核驱动的负荷，时钟分辨率的高低取决于硬件定时器的精度和用户中断函数的长度。

辅助时钟通过`AUX_CLK`,结构和系统时钟类似。这也是使用者定时中断主要使用的时钟。提供与系统时钟类似的接口函数来使用。

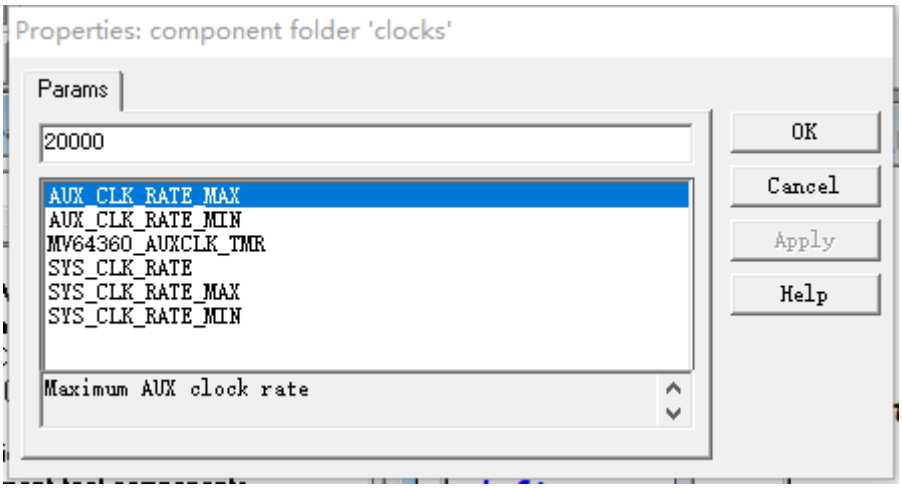
函数	说明
sysAuxClkConnect	将用户处理函数挂接在辅助时钟中断上
sysAuxClkRateSet	设置辅助时钟中断频率
sysAuxClkEnable	启动辅助时钟
sysAuxClkDisable	关闭辅助时钟

在使用辅助时钟的时候，我们需要在VxWorks组件中包含相关的组件，并修改相关的配置。

需要包含的组件如下图所示：



加粗的字体代表已经包含进镜像中，然后双击clocks文件夹，会弹出属性设置框。



在这里可以设置系统时钟和辅助时钟最大最小限制和缺省值。

更改完这些就可以编译出相应的镜像，然后在板卡启动的时候下载进板卡即可。

使用者可以随意设置中断的频率。例如：

```
sysAuxClkDisable();
sysAuxClkRateSet(20000);
sysAuxClkConnect((FUNCPTR)auxInt,0);
sysAuxClkEnable();
```

时间戳[Timestamp]依附于系统定时器。Timestamp机制用查询方式取得当前定时器的硬件计数值，去除中断处理负荷，可以获取比系统时钟高几十倍的分辨率。

常用函数接口如下表所示：

函数	说明
sysTimestamp	取得定时器当前的硬件计数值
sysTimestampFreq	取得硬件计数的频率

可以使用时间戳来评估一段代码的执行时间。

```
oldCount=sysTimestamp();
    /*code*/
newCount=sysTimestamp();
usedTime=(newCount-oldCount)/sysTimestampFreq();
```

由于取得是硬件原始计数，累加溢出后又会从零开始计数，所以需要根据实际情况来处理计数值回绕的问题。

中断处理函数在代码编写方面有些限制。由于中断处理函数不是在通常的任务上下文中运行，没有任务控制块，所有中断处理共享同一栈空间，所以中断处理函数不能调用会引起阻塞的系统函数，如取信号量，内存操作，IO操作，硬浮点处理等。

为了系统的实时性，要求中断处理函数尽量短小，中断事件的进一步处理可通过任务通信机制延迟到任务上下文中执行。

只要遵循上面描述的中断代码编写规范，任何C函数都可以作为用户中断处理函数，而不用添加编译器相关的预处理指令。