Abgabe zur 2. Praktikumsaufgabe im Fach „Embedded Computing“

erstellt von

Markus Schmidt  
Maximilian Gaul

Nachweis der korrekten Takterzeugung von 1ms:

Die Systemfunktion *clock\_nanosleep* erlaubt über das setzen des Attributes *tv\_nsec* auf *1.000.000* der Struktur *timespec* ein nanosekundengenaues Schlafen:

**Code:** *struct timespec time\_to\_wait;*  
*time\_to\_wait.tv\_nsec = 1000 \* 1000; /\* Wait 1ms \*/* *clock\_nanosleep(CLOCK\_REALTIME, 0, &time\_to\_wait, NULL);*

Damit kann ein Takt von einer Millisekunde realisiert werden. Um den korrekten Takt von 1ms zu überprüfen, messen wir die Zeit vor *clock\_nanosleep* und danach:

**Code:** *clock\_gettime(CLOCK\_REALTIME, &time\_before\_sleep);  
 …  
 clock\_gettime(CLOCK\_REALTIME, &time\_after\_sleep)*

Die Zeitdifferenz zwischen *time\_after\_sleep* und *time\_before\_sleep* sollte also möglichst genau **1ms** ergeben. Unseren Messungen zufolge ergibt sich mit der unveränderten Systemuhr eine Wartezeit von **2ms**. Wir führen diese ungenaue Wartezeit auf eine Systemuhr mit zu geringer Auflösung zurück.

Einstellen der Genauigkeit der Systemuhr

Die Genauigkeit der Systemuhr lässt sich über das Setzen des Attributes *nsec* der Struktur *\_clockperiod* und anschließendes Übergeben an *ClockPeriod* einstellen:

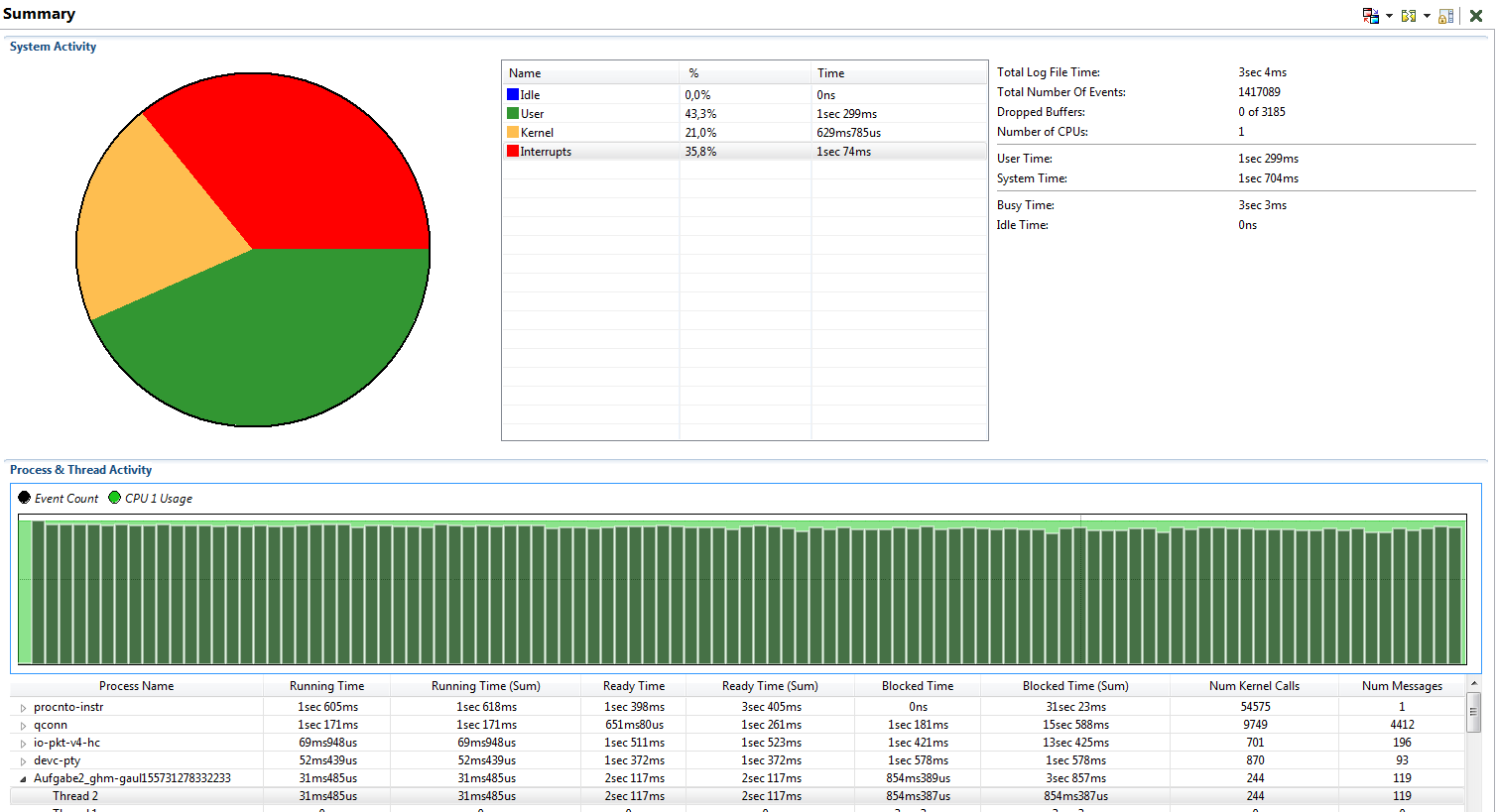
**Code:** *struct \_clockperiod set\_period;  
set\_period.nsec = microsecs \* 1000; /\* Setzen auf übergebenen Parameter der Funktion \*/  
set\_period.fract = 0;  
ClockPeriod(CLOCK\_REALTIME, &set\_period, NULL, 0);*

Auslesen der Genauigkeit lässt sich wie folgt realisieren:

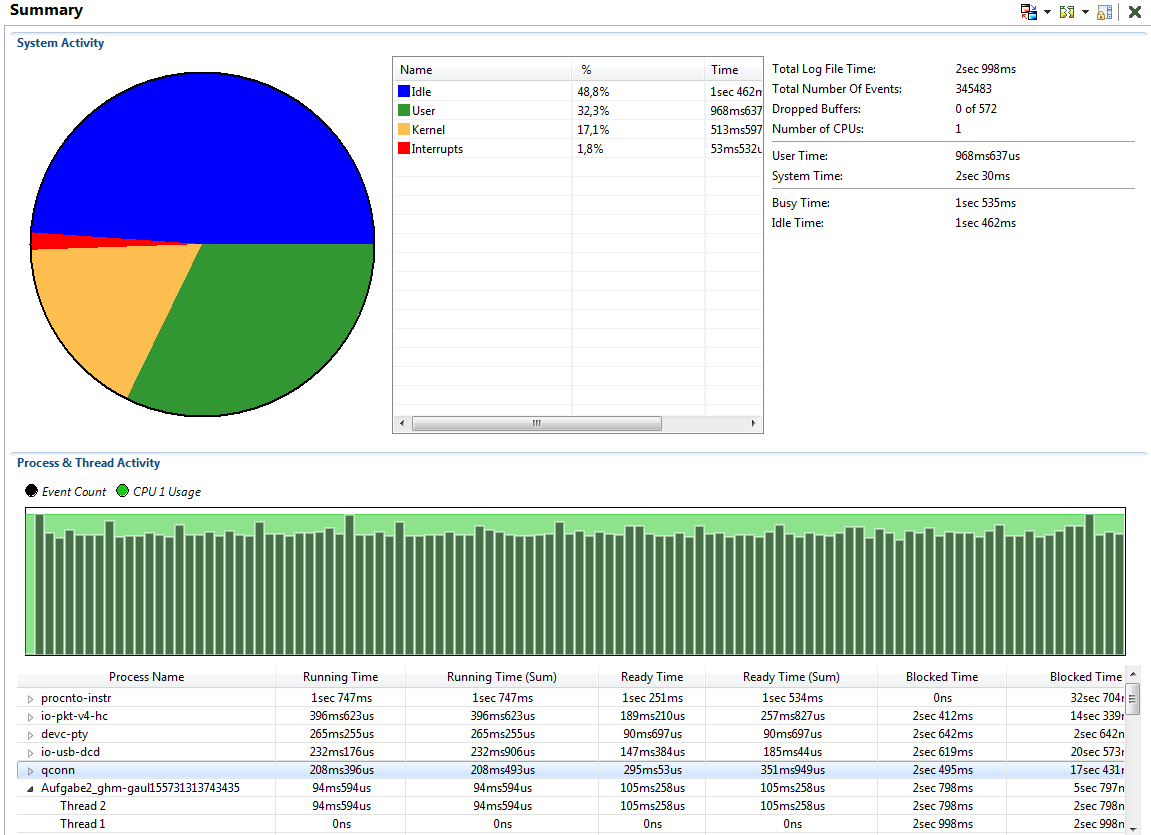
**Code:** *struct \_clockperiod get\_period;*  
*ClockPeriod\_r(CLOCK\_REALTIME, NULL, &get\_period, 0);*

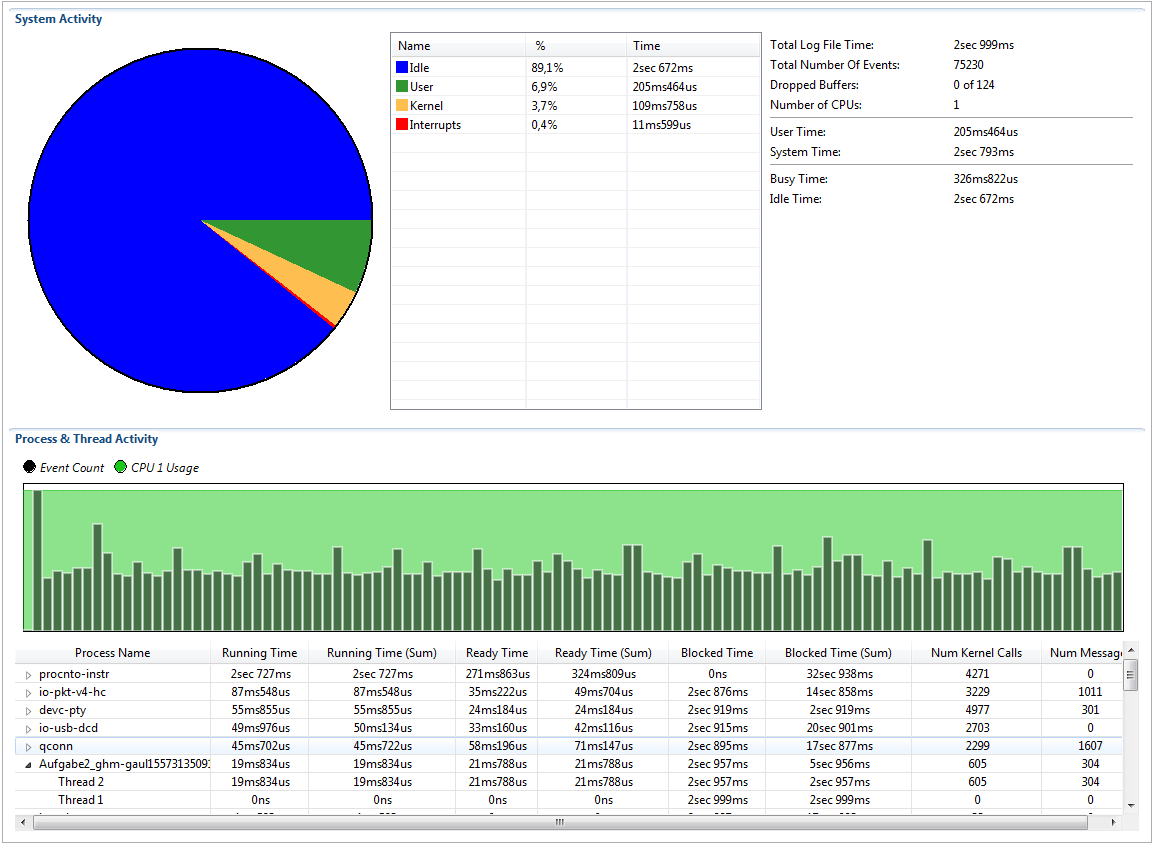
Unseren Messungen zufolge ist die Systemuhr beim Starten auf *1000us* eingestellt. Mit dieser Information lässt sich auch die falsche Wartezeit aus *a)* erklären (es ist schwer, *1ms* genau zu messen, wenn die Auflösung der Uhr nur *1ms* beträgt und beim Scheduling noch Zeit für Kontextwechsel anfallen außerdem kostet der Funktionsaufruf noch zusätzlich Zeit).

Ein auf *10us* eingestellter Systemtakt erzeugt folgende Messungen des *Kernel Events Tracer*



Wie zu erkennen ist, führt ein Systemtakt von *10us* zur kompletten Auslastung des Systems (*Idle =* *0.0%*) da dadurch sehr häufig Interrupts erzeugt werden, die natürlich alle behandelt werden müssen (ggf. Änderung auf einen Prozess mit höherer Priorität + Kontextwechsel) = *35.8%*.

Wie auf den folgenden Abbildungen zu sehen reduziert sich die Systemauslastung mit Verringerung des Systemtakts (1. Bild = 1000us, 2. Bild = 5000us) systematisch: 



Je niedriger der Systemtakt desto ungenauer ist die Wartezeit aus *a)*. Mit *10us* Systemtakt wird die Wartezeit von *1ms* wie von *a)* gefordert erreicht.