# Code:

#include <stdlib.h>,

#include <stdio.h>

#include <time.h>

#include <string.h>

#include <sys/neutrino.h>

struct \_clockperiod clockperiodNew, clockperiodOld;

void changeSystemTick(unsigned int microsecs){

struct timespec res;

// Berechnen des neuen Taktes

clockperiodNew.nsec = microsecs \* 1000000;

clockperiodNew.fract = 0;

// Ausgeben des alten Taktes

int retgetRes = clock\_getres(CLOCK\_REALTIME, &res);

if (retgetRes == -1) {

perror("clock Res old");

exit(-1);

}

printf("Old system period: %ld \n", res.tv\_nsec);

// Änderen des System Taktes

int retPer = ClockPeriod(CLOCK\_REALTIME, &clockperiodNew, &clockperiodOld, 0);

if (retPer == -1) {

perror("clock ret Per");

exit(-1);

}

// Ausgeben des neuen Taktes

retgetRes = clock\_getres(CLOCK\_REALTIME, &res);

if (retgetRes == -1) {

perror("clock Res new");

exit(-1);

}

printf("New system period: %ld \n", res.tv\_nsec);

}

int main(int argc, char \*argv[]) {

struct timespec start, stop;

//changeSystemTick(1);

// Stoppen des Start Zeitpunktes

int retStart = clock\_gettime(CLOCK\_REALTIME, &start);

if (retStart == -1) {

perror("clock gettime Start");

exit(-1);

}

int i;

for (i = 0; i < 1000; ++i) {

start.tv\_nsec += 1000000L;

start.tv\_nsec = start.tv\_nsec % 1000000000;

if ( start.tv\_nsec == 0) {

++start.tv\_sec;

}

int retNano = clock\_nanosleep(CLOCK\_REALTIME, TIMER\_ABSTIME, &start, NULL);

if (retNano != 0) {

printf("Error Nano: %s \n", strerror(retNano));

exit(-1);

}

}

// Stoppen des End Zeitpunktes

int retStop = clock\_gettime(CLOCK\_REALTIME, &stop);

if (retStop == -1) {

perror("clock gettime Stop");

exit(-1);

}

if ((int)(stop.tv\_nsec - start.tv\_nsec) != 0) {

printf("Something wrong at: %d with start: %ld, and stop: %ld \n", i, start.tv\_nsec, stop.tv\_nsec);

exit(-1);

}

printf("Success!\n");

return EXIT\_SUCCESS;

}

# Kommentare und Anmerkungen zum Code:

1. Es werden zwei Timestamps start und stop erzeugt. Auf start wird die zu arbeitende Zeit addiert und dann eine gewisse Zeit gewartet mittels clock\_nanosleep() --> Umsetzung der Aufgabe
2. Nach besagter Zeit, wird nun start und stop verglichen. Sind diese beiden gleich, also war der Aufruf erfolgreich, so wird dies auf der Kommandozeile ausgegeben. Falls nicht, wird dies ebenfalls auf der Kommandozeile erkennbar --> Validierung der Aufgabe
3. Die Ermittlung der aktuellen Systemuhr kann mit clock\_getres(CLOCK\_REALTIME, &res) ermitteln und ausgeben. Er ist 1ms.
4. Die Systemzeit wird mit dem Aufruf der Fuktion ClockPeriod(CLOCK\_REALTIME, &clockperiodNew, &clockperiodOld, 0) auf clockperiodNew geändert.
5. Die kleinste sinnvolle Zykuluszeit ist \*\*\*, weil \*\*\*
6. Wenn die Systemtakt nun kein Teiler der Wartezeit aus a) ist, wird eine falsche Zeit lang gewartet, es tritt ein Fehler auf.   
   Kein Fehler: Wartezeit 1ms und Systemtakt 0.5ms  
   Fehler: Wartezeit 1ms und Systemtakt 0.3ms --> es wird 4\*0.3ms = 1.2ms gewartet