Praktikum 5

Lernziele: Timer, Watchdog und Scheduler in GNU C auf PC (online-Praktikum)

***First try:***

Install Linux Ubuntu 64Bit VM via Oracle Virtualmachine  
  
-> Install prerequisites:  
- sudo apt-get install upgrade  
- sudo apt-get install update  
- sudo apt install g++ codeblocks git  
- sudo apt-get install libncurses5-dev libncursesw5-dev  
  
-> Download codeblocks template from moodle and open  
-> Project->Build options->Linker settings->Other linker options->   
 -pthread  
 -lcurses  
-> Exclude conio.h from keyb\_ITR.h (only windows)

* Project not runnable (xcore dump error)

-> Include files from Samuel Koob - Forum Beitrag

* **No acceptable results - all points printed at once,  
  no matter the \_SUS\_ and sleep vars**

***Second try:***

Use Windowsx64 environment and try changing \_SUS\_ arguments

* **pthread.h** not found
* Try fixing pthread on windows via <https://stackoverflow.com/questions/19467455/how-to-set-up-pthreads-on-windows>   
  -> no results

***Successful try:***

Use Windowsx64 environment and install compiler components with minGW

-> Download minGW and install into C:\  
-> Run C:\MinGW\libexec\mingw-get\guimain.exe and install pthread extensions for Windows



-> In Codeblocks use minGW Compiler under Settings->Compiler-> Selected compiler = ”GNU GCC Compiler”  
-> In Codeblocks Compiler Settings under Settings->Compiler-> Selected compiler -> Linker Settings include “-pthread”

Credit to forum under <https://lernen.h-da.de/mod/forum/discuss.php?d=146124> from 26.01. 11:24

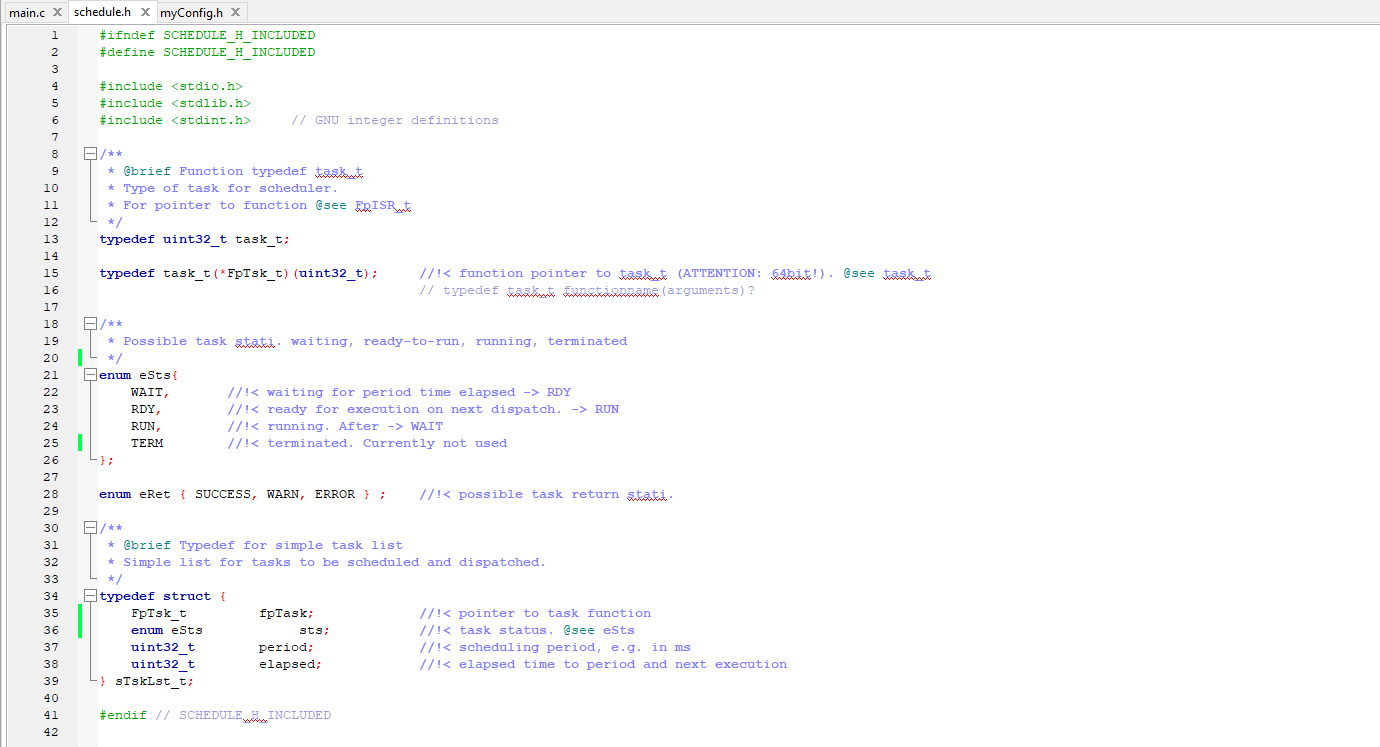
* Program now runnable

1. Testen Sie das Template. Beachten Sie die mit doxygen erzeugte Dokumentation [1]!
2. Der Watchdog verfügt über eine eigene Instanz eines Timers (wd\_Thread). Wenn dieser abgelaufen ist wird aufgerufen:  
   Funktionspointer pfpISR\_WD (struct des watchdogs), zeigt wiederum auf  
   Funktion FpISR\_t des keyb\_ITR.c, zeigt wiederum auf  
   Funktion simISR.h: typedef ISR\_t(\*FpISR\_t)();

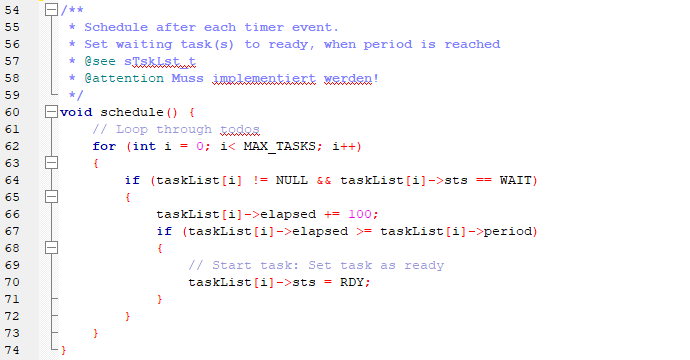
Also wird eine interrupt service Routine abgefeuert, die das Programm beendet.

1. Kommentieren Sie dann erst einmal den Codebereich für den Watchdog aus.  
   Folge: Das Programm läuft in einer Endlosschleife und wird nicht nach der voreingestellten Zeit abgebrochen!
2. Die Timer-ISR sollte alle 1/10 s feuern. Testen Sie!
3. Mehrere kleine Tasks (Funktionen) mit unterschiedlichen Perioden (zeitlichen Häufigkeiten) ausgeführt werden. "schedule.h" gibt Vorgaben zu den Zuständen der Tasks: WAIT warten auf Ausführung, RDY ausführungsbereit, RUN wird ausgeführt.
4. Die (zu schreibende) Funktion schedule() (dt. „Planen“) überprüft den Zustand jedes Tasks, setzt die bereits gewartete Zeit hoch und falls period erreicht, den Zustand auf RDY

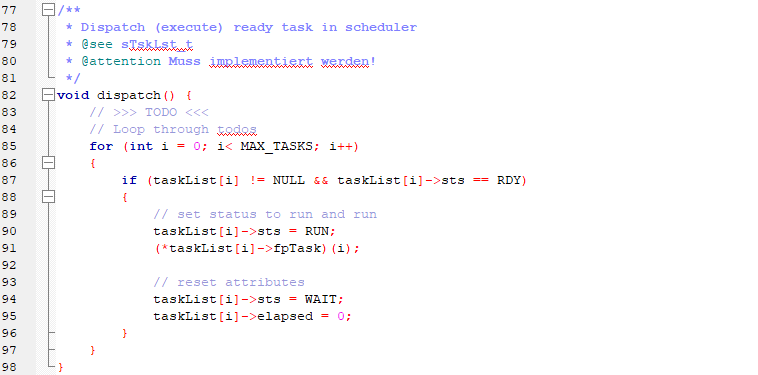
schedule.h



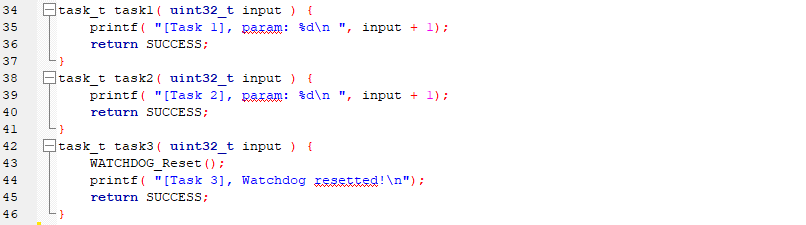
main.c



1. Die (zu schreibende) Funktion dispatch() (dt. „Ausführen“), in der Endlosschleife aufzurufen, führt alle RDY-Tasks aus und setzt danach den Status wieder auf WAIT und elapsed natürlich wieder auf 0.

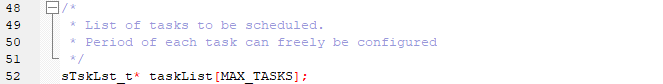
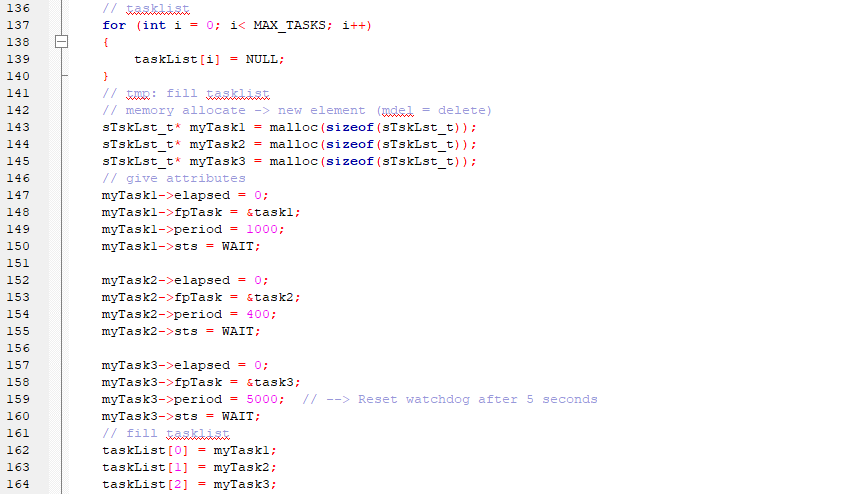
main.c  


1. Ergänzen Sie weitere Tasks vom Typ task\_t zum Testen.

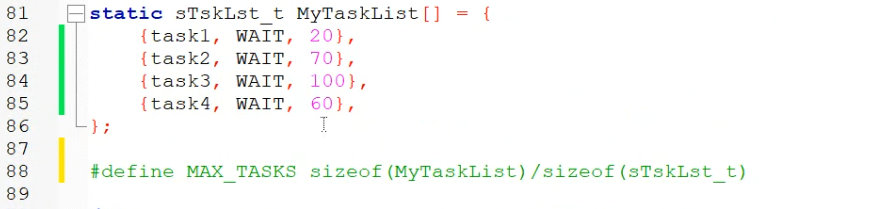
main.c  


1. Verwalten Sie diese Tasks unter Benutzung von **sTskLst\_t**.

main.c

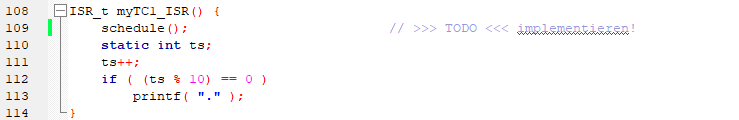
  
  


Alternativ:

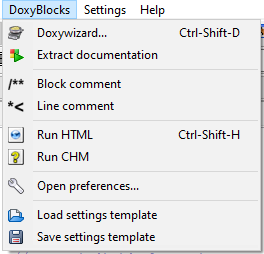


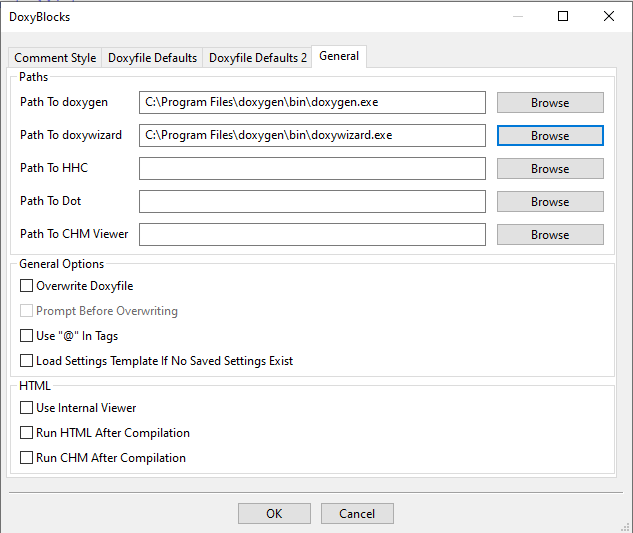
1. Nutzen Sie einen Timer (10 – 100 ms) um den Scheduler schedule() regelmäßig aufzurufen.

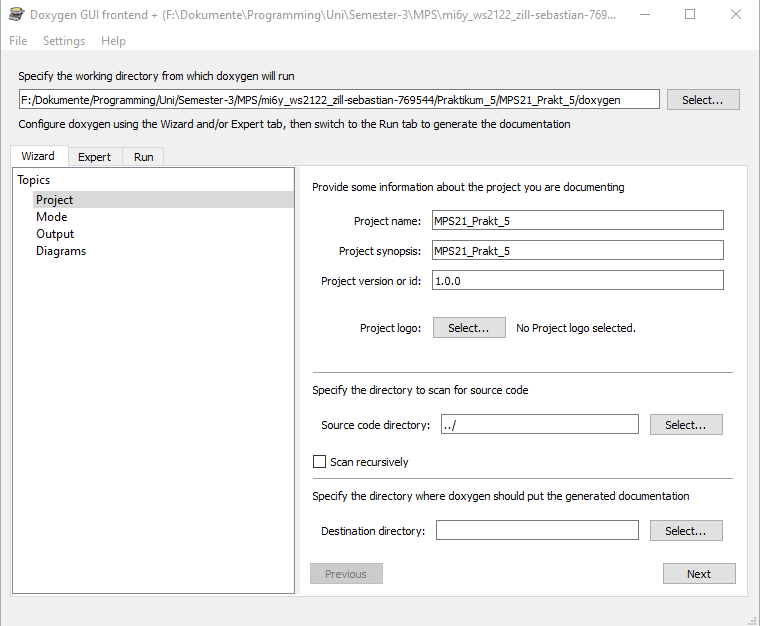
main.c (in main)  

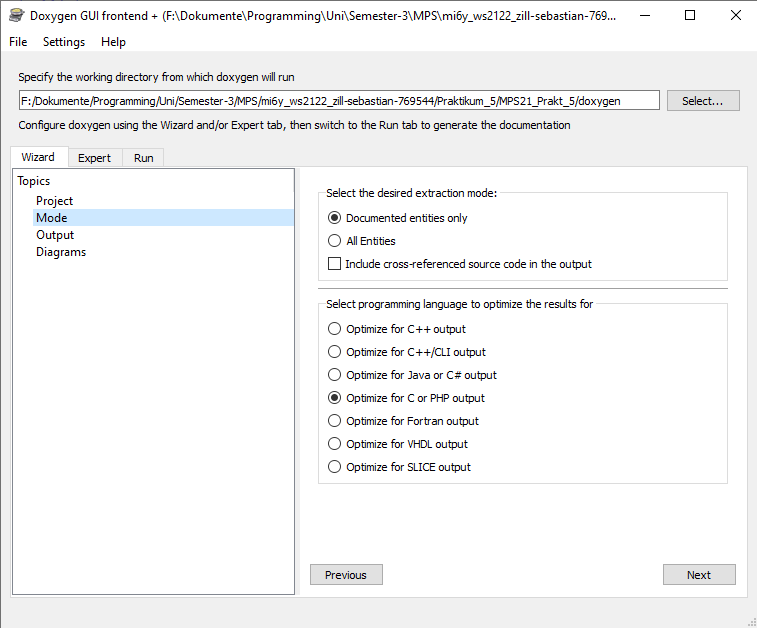

main.c (register ISR before main)  


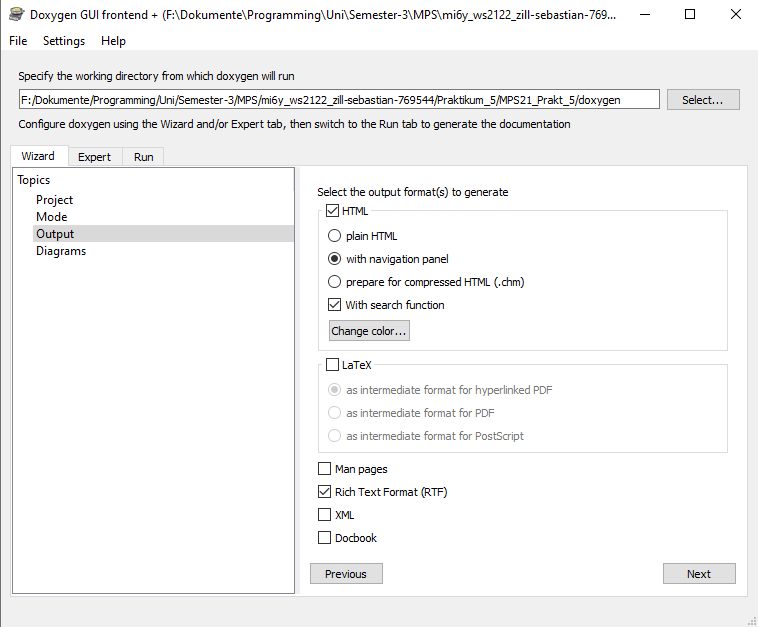
1. Download Windows installer (exe) via <https://www.doxygen.nl/download.html>  
   Run doxywizard
2. + b)  
   Set up Doxywizard and run documentation

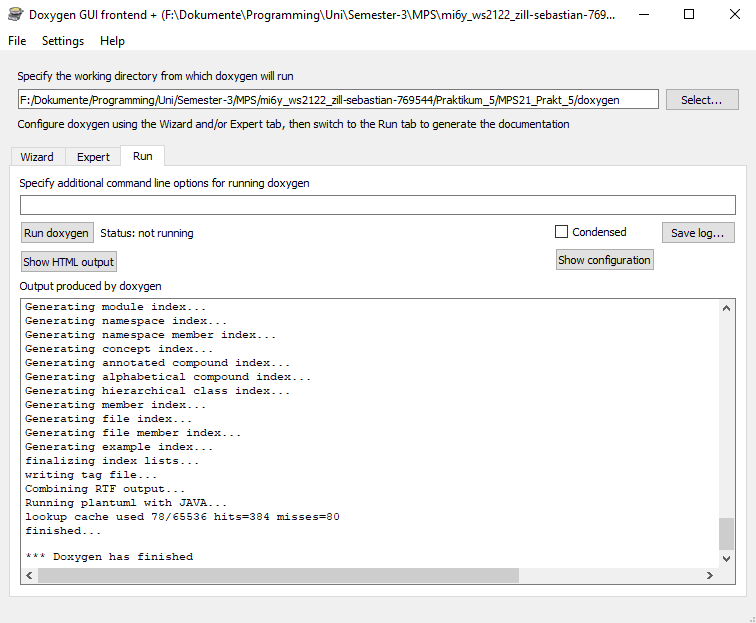


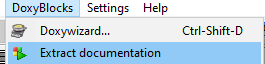












c) + d) + e) + f)  
