

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

	mingw32-pthreads-w32	dev	2.10-pre-201608...	2.10-pre-201608...	POSIX threading library for Win32
	mingw32-pthreads-w32	doc	2.10-pre-201608...	2.10-pre-201608...	POSIX threading library for Win32
	mingw32-pthreads-w32	lic	2.10-pre-201608...	2.10-pre-201608...	POSIX threading library for Win32

-> 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]!

- a) 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.

- b) 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!

- c) Die Timer-ISR sollte alle 1/10 s feuern. Testen Sie!

2. 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.

- a) 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 X schedule.h X myConfig.h X
1  #ifndef SCHEDULE_H_INCLUDED
2  #define SCHEDULE_H_INCLUDED
3
4  #include <stdio.h>
5  #include <stdlib.h>
6  #include <stdint.h> // GNU integer definitions
7
8  /**
9   * @brief Function typedef task_t
10  * Type of task for scheduler.
11  * For pointer to function @see FpISR_t
12  */
13  typedef uint32_t task_t;
14
15  typedef task_t(*FpTask_t)(uint32_t); //!< function pointer to task_t (ATTENTION: 64bit!). @see task_t
16  // typedef task_t functionname(arguments)?
17
18  /**
19   * Possible task stati. waiting, ready-to-run, running, terminated
20   */
21  enum eSts{
22      WAIT, //!< waiting for period time elapsed -> RDY
23      RDY, //!< ready for execution on next dispatch. -> RUN
24      RUN, //!< running. After -> WAIT
25      TERM //!< terminated. Currently not used
26  };
27
28  enum eRet { SUCCESS, WARN, ERROR }; //!< possible task return stati.
29
30  /**
31   * @brief Typedef for simple task list
32   * Simple list for tasks to be scheduled and dispatched.
33   */
34  typedef struct {
35      FpTask_t fpTask; //!< pointer to task function
36      enum eSts sts; //!< task status. @see eSts
37      uint32_t period; //!< scheduling period, e.g. in ms
38      uint32_t elapsed; //!< elapsed time to period and next execution
39  } sTaskList_t;
40
41  #endif // SCHEDULE_H_INCLUDED
42
```

main.c

```

54  /**
55   * Schedule after each timer event.
56   * Set waiting task(s) to ready, when period is reached
57   * @see sTaskList_t
58   * @attention Muss implementiert werden!
59   */
60  void schedule() {
61      // Loop through todos
62      for (int i = 0; i < MAX_TASKS; i++)
63      {
64          if (taskList[i] != NULL && taskList[i]->sts == WAIT)
65          {
66              taskList[i]->elapsed += 100;
67              if (taskList[i]->elapsed >= taskList[i]->period)
68              {
69                  // Start task: Set task as ready
70                  taskList[i]->sts = RDY;
71              }
72          }
73      }
74  }

```

- b) 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

```

77  /**
78   * Dispatch (execute) ready task in scheduler
79   * @see sTaskList_t
80   * @attention Muss implementiert werden!
81   */
82  void dispatch() {
83      // >>> TODO <<<
84      // Loop through todos
85      for (int i = 0; i < MAX_TASKS; i++)
86      {
87          if (taskList[i] != NULL && taskList[i]->sts == RDY)
88          {
89              // set status to run and run
90              taskList[i]->sts = RUN;
91              (*taskList[i]->fpTask)(i);
92
93              // reset attributes
94              taskList[i]->sts = WAIT;
95              taskList[i]->elapsed = 0;
96          }
97      }
98  }

```

c) Ergänzen Sie weitere Tasks vom Typ `task_t` zum Testen.

main.c

```

34 task_t task1( uint32_t input ) {
35     printf( "[Task 1], param: %d\n ", input + 1);
36     return SUCCESS;
37 }
38 task_t task2( uint32_t input ) {
39     printf( "[Task 2], param: %d\n ", input + 1);
40     return SUCCESS;
41 }
42 task_t task3( uint32_t input ) {
43     WATCHDOG_Reset();
44     printf( "[Task 3], Watchdog resetted!\n");
45     return SUCCESS;
46 }

```

d) Verwalten Sie diese Tasks unter Benutzung von `sTskLst_t`.

main.c

```

17 #define MAX_TASKS 3          //!< Maximum of tasks to be handled by scheduler in taskList
18 /*
19  * List of tasks to be scheduled.
20  * Period of each task can freely be configured
21  */
22 sTskLst_t* taskList[MAX_TASKS];

136 // tasklist
137 for (int i = 0; i < MAX_TASKS; i++)
138 {
139     taskList[i] = NULL;
140 }
141 // tmp: fill tasklist
142 // memory allocate -> new element (mdel = delete)
143 sTskLst_t* myTask1 = malloc(sizeof(sTskLst_t));
144 sTskLst_t* myTask2 = malloc(sizeof(sTskLst_t));
145 sTskLst_t* myTask3 = malloc(sizeof(sTskLst_t));
146 // give attributes
147 myTask1->elapsed = 0;
148 myTask1->fpTask = &task1;
149 myTask1->period = 1000;
150 myTask1->sts = WAIT;
151
152 myTask2->elapsed = 0;
153 myTask2->fpTask = &task2;
154 myTask2->period = 400;
155 myTask2->sts = WAIT;
156
157 myTask3->elapsed = 0;
158 myTask3->fpTask = &task3;
159 myTask3->period = 5000; // --> Reset watchdog after 5 seconds
160 myTask3->sts = WAIT;
161 // fill tasklist
162 taskList[0] = myTask1;
163 taskList[1] = myTask2;
164 taskList[2] = myTask3;

```

Alternativ:

```

81 static sTskLst_t MyTaskList[] = {
82     {task1, WAIT, 20},
83     {task2, WAIT, 70},
84     {task3, WAIT, 100},
85     {task4, WAIT, 60},
86 };
87
88 #define MAX_TASKS sizeof(MyTaskList)/sizeof(sTskLst_t)
89

```

- e) Nutzen Sie einen Timer (10 - 100 ms) um den Scheduler `schedule()` regelmäßig aufzurufen.

main.c (in main)

```
133 // Timer for Scheduler
134 TC1config( myTC1_ISR, 100 ); // config timer 1, ISR, 100 ms
135 TCengine_start(); // start simulated timer
```

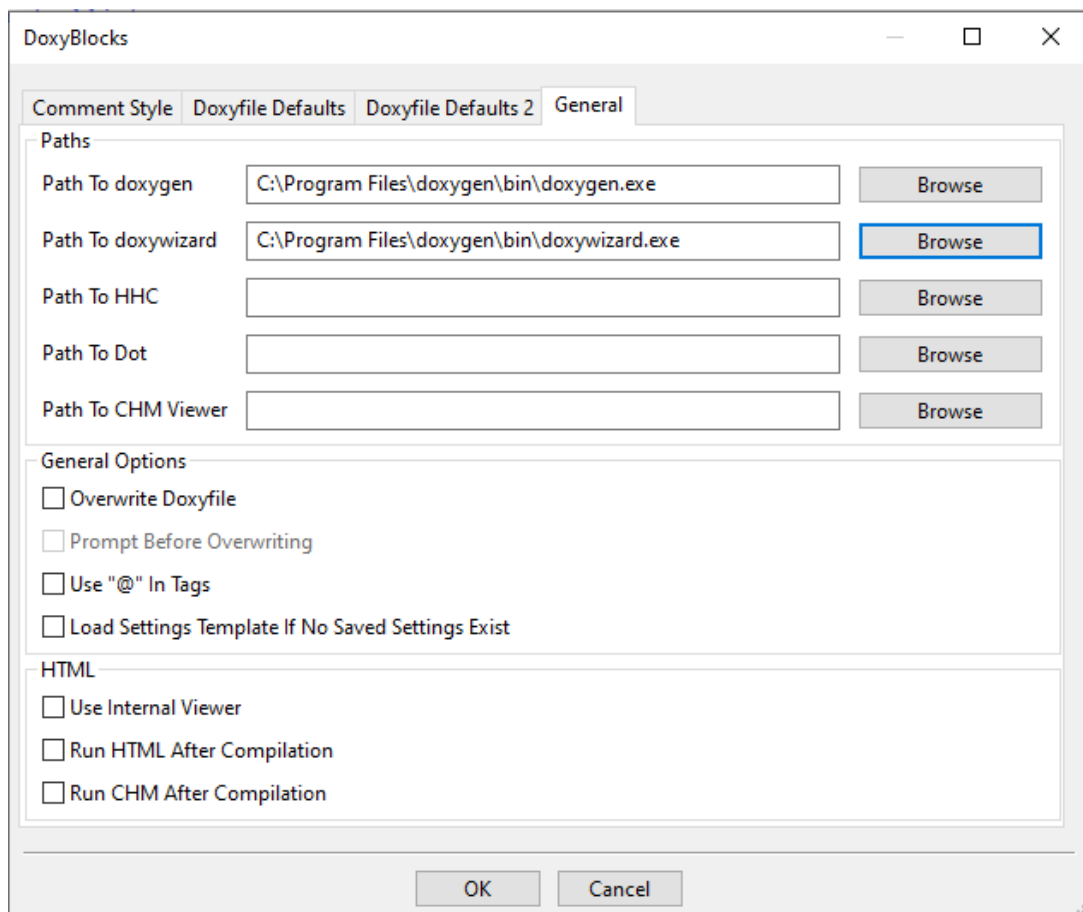
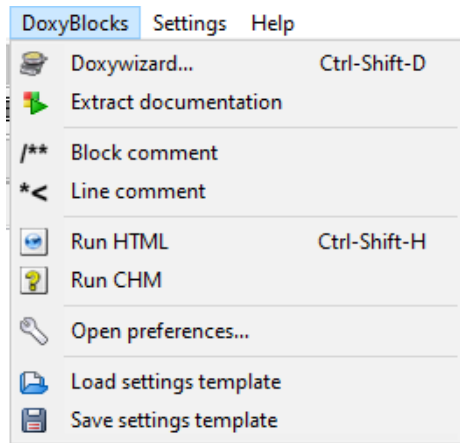
main.c (register ISR before main)

```
108 ISR_t myTC1_ISR() {
109     schedule(); // >>> TODO <<< implementieren!
110     static int ts;
111     ts++;
112     if ( (ts % 10) == 0 )
113         printf( "." );
114 }
```

3. Download Windows installer (exe) via <https://www.doxygen.nl/download.html>
Run doxywizard

a) + b)

Set up Doxywizard and run documentation



Doxygen GUI frontend + (F:\Dokumente\Programming\Uni\Semester-3\MPS\mi6y_ws2122_zill-sebastian-769...

File Settings Help

Specify the working directory from which doxygen will run

F:\Dokumente\Programming\Uni\Semester-3\MPS\mi6y_ws2122_zill-sebastian-769544\Praktikum_5\MPS21_Prakt_5\doxygen Select...

Configure doxygen using the Wizard and/or Expert tab, then switch to the Run tab to generate the documentation

Wizard Expert Run

Topics

- Project
- Mode
- Output
- Diagrams

Provide some information about the project you are documenting

Project name: MPS21_Prakt_5

Project synopsis: MPS21_Prakt_5

Project version or id: 1.0.0

Project logo: Select... No Project logo selected.

Specify the directory to scan for source code

Source code directory: ../ Select...

☐ Scan recursively

Specify the directory where doxygen should put the generated documentation

Destination directory: Select...

Previous Next

Doxygen GUI frontend + (F:\Dokumente\Programming\Uni\Semester-3\MPS\mi6y_ws2122_zill-sebastian-769...

File Settings Help

Specify the working directory from which doxygen will run

F:\Dokumente\Programming\Uni\Semester-3\MPS\mi6y_ws2122_zill-sebastian-769544\Praktikum_5\MPS21_Prakt_5\doxygen Select...

Configure doxygen using the Wizard and/or Expert tab, then switch to the Run tab to generate the documentation

Wizard Expert Run

Topics

- Project
- Mode
- Output
- Diagrams

Select the desired extraction mode:

☒ Documented entities only

☐ All Entities

☐ Include cross-referenced source code in the output

Select programming language to optimize the results for

☐ Optimize for C++ output

☐ Optimize for C++/CLI output

☐ Optimize for Java or C# output

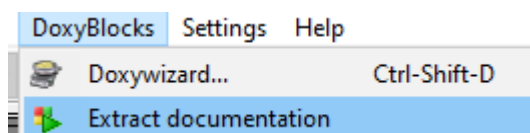
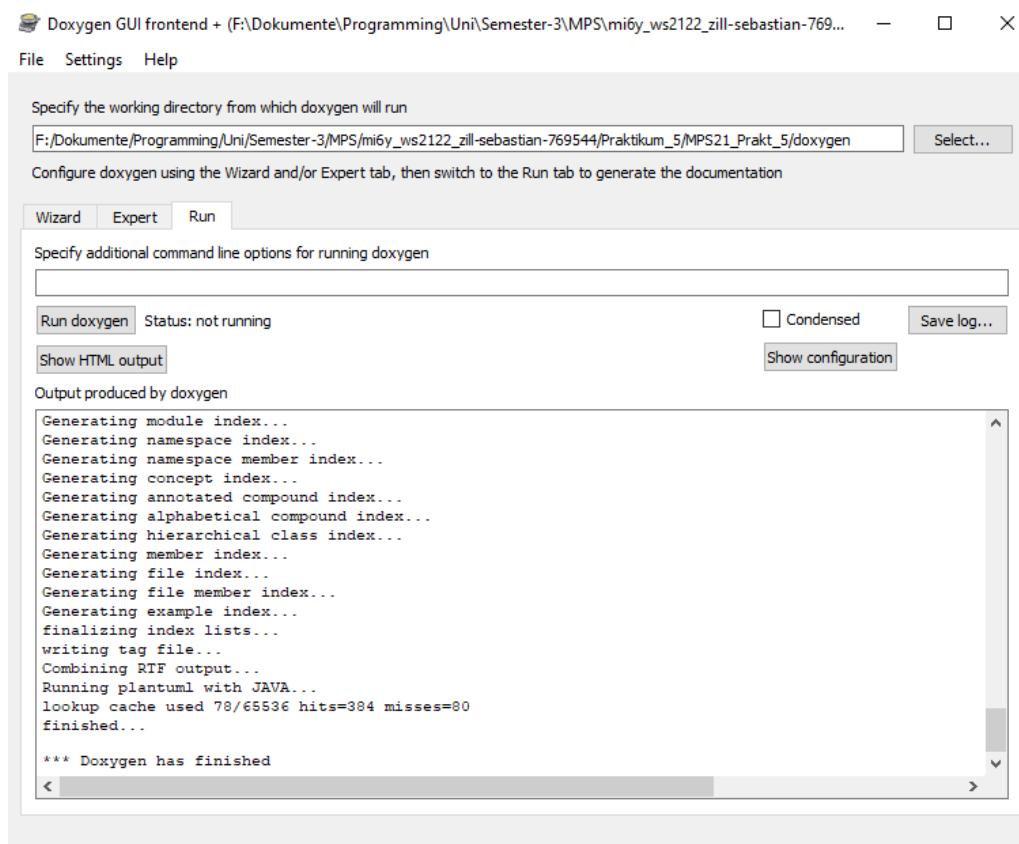
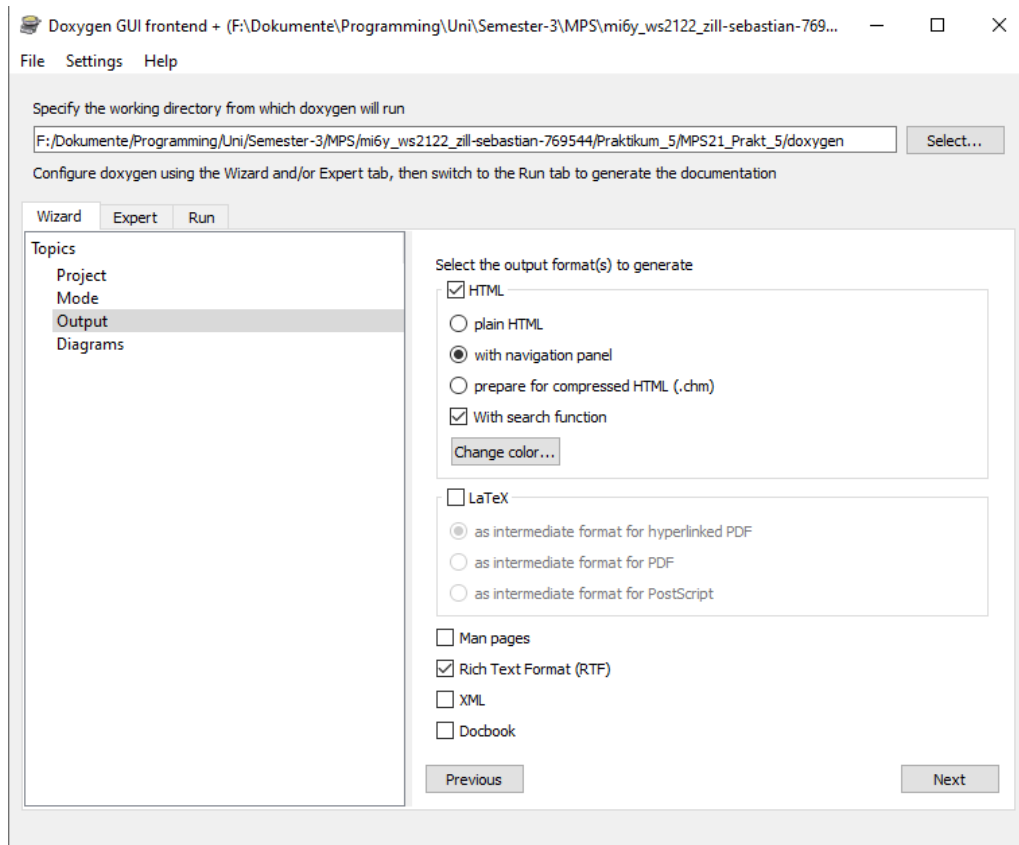
☒ Optimize for C or PHP output

☐ Optimize for Fortran output

☐ Optimize for VHDL output

☐ Optimize for SLICE output

Previous Next



MPS21_Prakt_5

sTcconf_t Struct Reference

struct for timer counter internal configuration individual TC configuration block ...

Public Attributes

FpsISR_J	pfpISR_TC
address of watchdog ISR Timer function, int to NULL. More ...	
write_J TC_hw	
TC compare register	
write_J TC_cnt	
count TC register	
write_J TC_capt(J)	
capture registers for later use	
write_J TC_sta	
status: 0: not started, 1: TC active	

Detailed Description

struct for timer counter internal configuration individual TC configuration

Date
2022-01-11

Member Data Documentation

- pfpISR_TC**

FpsISR_J sTcconf_t pfpISR_TC
address of watchdog ISR Timer function, int to NULL.

See also
[FpsISR_J](#)

The documentation for this struct was generated from the following file:

- TCam.c

AUTHOR
Version 1.0.0
Wed Jan 26 2022