

Angewandte Informatik

Semester 4

Dokumentation

Simulator eines Mikrocontrollers PIC16F84

Name: Aaron Moser

Matrikelnummer: 187648

E-Mail: aaronmoser@t-online.de

Prüfungsdatum: 05.05.2022

Betreuer: Dipl.-Ing. Stefan Lehmann

Inhaltsverzeichnis

[Abbildungsverzeichnis 4](#_Toc102909072)

[Einleitung 5](#_Toc102909073)

[Mikrocontroller PIC16F84 5](#_Toc102909074)

[Graphische Benutzeroberfläche des Simulators 6](#_Toc102909075)

[Hauptmenü 6](#_Toc102909076)

[LST Anzeigefeld 7](#_Toc102909077)

[SFR und W Registertabelle 7](#_Toc102909078)

[STATUS, OPTION und INTCON Registertabelle 8](#_Toc102909079)

[RAM (Datenspeicher) Tabelle 8](#_Toc102909080)

[PORT A und PORT B Feld 8](#_Toc102909081)

[STACK Tabelle 9](#_Toc102909082)

[Laufzeitzähler Feld und Quarzfrequenz Drop-Down 9](#_Toc102909083)

[Simulatorsteuerungsfeld 10](#_Toc102909084)

[Implementierung 11](#_Toc102909085)

[Klasse MyControlModel 12](#_Toc102909086)

[Klasse MyControlView 13](#_Toc102909087)

[Klasse Alu 14](#_Toc102909088)

[Klasse Eeprom 14](#_Toc102909089)

[Klasse EepromThread 15](#_Toc102909090)

[Klasse InstructionDecoder 15](#_Toc102909091)

[Klasse Pic 15](#_Toc102909092)

[Klasse ProgramMemory 15](#_Toc102909093)

[Klasse Ram 15](#_Toc102909094)

[Klasse Stack 15](#_Toc102909095)

[Klasse Time 15](#_Toc102909096)

[Klasse ReadProgramFile 15](#_Toc102909097)

[Klasse MyModel 15](#_Toc102909098)

[Klasse MyModelData 15](#_Toc102909099)

[Klasse ProgramStepInformation 15](#_Toc102909100)

[Klasse Main 15](#_Toc102909101)

[Klasse GUIAbout 15](#_Toc102909102)

[Klasse GUIHelp 15](#_Toc102909103)

[Klasse GUIMainFrame 15](#_Toc102909104)

[Klasse GUIMCMenu 15](#_Toc102909105)

[Klasse GUIMenuBar 15](#_Toc102909106)

[Klasse GUIPorts 15](#_Toc102909107)

[Klasse GUIProgramMemory 15](#_Toc102909108)

[Klasse GUIRamTable 16](#_Toc102909109)

[Klasse GUIRegister 16](#_Toc102909110)

[Klasse GUIRegistersDetailed 16](#_Toc102909111)

[Klasse GUIStack 16](#_Toc102909112)

[Klasse GUITestFileTable 16](#_Toc102909113)

[Klasse GUITime 16](#_Toc102909114)

[Klasse IMyView 16](#_Toc102909115)

[Klasse MyColors 16](#_Toc102909116)

[Klasse MyView 16](#_Toc102909117)

# Abbildungsverzeichnis

[Abbildung 1 (Q1) 5](file:///C:\Users\Windows%2010\Desktop\Studium\Repos_Computerarchitecture\Rechnerarchitektur_Simulator_PIC\documentation\WORD\Dokumentation_Final.docx#_Toc102907752)

[Abbildung 2 6](file:///C:\Users\Windows%2010\Desktop\Studium\Repos_Computerarchitecture\Rechnerarchitektur_Simulator_PIC\documentation\WORD\Dokumentation_Final.docx#_Toc102907753)

[Abbildung 3 7](file:///C:\Users\Windows%2010\Desktop\Studium\Repos_Computerarchitecture\Rechnerarchitektur_Simulator_PIC\documentation\WORD\Dokumentation_Final.docx#_Toc102907754)

[Abbildung 4 7](file:///C:\Users\Windows%2010\Desktop\Studium\Repos_Computerarchitecture\Rechnerarchitektur_Simulator_PIC\documentation\WORD\Dokumentation_Final.docx#_Toc102907755)

[Abbildung 5 8](file:///C:\Users\Windows%2010\Desktop\Studium\Repos_Computerarchitecture\Rechnerarchitektur_Simulator_PIC\documentation\WORD\Dokumentation_Final.docx#_Toc102907756)

[Abbildung 6 8](file:///C:\Users\Windows%2010\Desktop\Studium\Repos_Computerarchitecture\Rechnerarchitektur_Simulator_PIC\documentation\WORD\Dokumentation_Final.docx#_Toc102907757)

[Abbildung 7 8](file:///C:\Users\Windows%2010\Desktop\Studium\Repos_Computerarchitecture\Rechnerarchitektur_Simulator_PIC\documentation\WORD\Dokumentation_Final.docx#_Toc102907758)

[Abbildung 8 9](file:///C:\Users\Windows%2010\Desktop\Studium\Repos_Computerarchitecture\Rechnerarchitektur_Simulator_PIC\documentation\WORD\Dokumentation_Final.docx#_Toc102907759)

[Abbildung 9 9](file:///C:\Users\Windows%2010\Desktop\Studium\Repos_Computerarchitecture\Rechnerarchitektur_Simulator_PIC\documentation\WORD\Dokumentation_Final.docx#_Toc102907760)

# Einleitung

Die Aufgabe der Projektarbeit im Fach Rechnerarchitektur ist es, den Simulator eines Mikrocontrollers zu entwickeln, der dessen Funktionen so gut wie möglich nachbildet.

Ein Simulator ist ein Modell eines real existierenden Objekts, welches nachgebildet werden soll. Er ist nicht als exaktes Abbild zu verstehen, sondern kann je nach Anforderung und Aufgabe vereinfacht werden.

Dementsprechend bildet der hier dokumentierte Simulator des Mikrocontrollers PIC16F84 diesen nur zu Teilen ab. Manche Teile wurden zur Vereinfachung nicht implementiert. Reale Eigenschaften wie verändertes Verhalten bei erhöhten Bauteiltemperaturen oder andere physikalische Eigenschaften werden nicht nachgebildet.

Der Simulator besteht aus einer graphischen Benutzeroberfläche (GUI), welche die internen und externen Zustände des Mikrocontrollers wie z.B. Ein- und Ausgangspins oder den Datenspeicher darstellt und einem im Hintergrund laufenden Programm, welches die interne logische Funktion des Mikrocontrollers nachbildet.

# Mikrocontroller PIC16F84

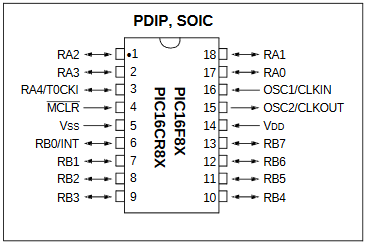
Der Mikrocontroller PIC16F84 wurde im Jahr 1998 von *Microchip Technology Inc.* entwickelt. Er gehört zur Produktfamilie der PIC16CXX. Die interne Architektur entspricht der Harvard-Architektur, bei der Daten- und Programmspeicher voneinander getrennt sind, dadurch entfällt das in der von-Neumann-Architektur typische Nadelöhr. Grundsätzlich umfässt die Größe des Befehlssatzes 35 Befehle, wobei beim Simulator wenige dieser Befehle nicht implementiert sind (CLRWDT, SLEEP, TRIS, OPTION). Der Programmspeicher kann mit einem Programm geladen werden, welches eine Reihe an Befehlen beinhaltet.

Abbildung 1 (Q1)

# Graphische Benutzeroberfläche des Simulators

Abbildung 2

Die graphische Benutzeroberfläche des Simulators besteht aus einem Hauptfenster mit einem Hauptmenü, mehreren Tabellen, 4 Schaltflächen und einem Drop-Down-Menü.

## Hauptmenü

Das Hauptmenü besteht aus 4 Punkten (Datei, Ansicht, Microcontroller und Hilfe), die wiederum in kleinere Punkte unterteilt sind.

Der Menüpunkt „Datei“ wird in die Punkte „Testdatei laden“, „Programmzustand laden“, „Programmzustand speichern“ und „Simulation beenden“ aufgeteilt. Wobei nur die Punkte „Testdatei laden“ und „Simulation beenden“ zurzeit implementiert sind, da die anderen Punkte für die Bewertung nicht relevant sind.

„Ansicht“ besteht aus dem Punkt „Thema“ über den zwischen den Anzeigemodi „Dunkler Modus“ und „Heller Modus“ gewählt werden kann.

Zusätzlich zu den vier Schaltflächen im Hauptfenster sind unter dem Hauptmenüpunkt „Microcontroller“ vier Punkte zum Steuern der Simulation (Programm starten, Programm pausieren, Programm zuruecksetzen, Schritt fuer Schritt). Außerdem gibt es einen Unterpunkt „Bearbeitungsintervall“, über den die Geschwindigkeit der Programmausführung geändert werden kann um die Möglichkeit zu bieten, die Programmausführung besser nachzuvollziehen. Zur Auswahl stehen: „Sofort“, „1 Sekunde“ und „2 Sekunden“.

Der Punkt „Hilfe“ enthält die Unterpunkte „Sprache“, „Anleitung“ und „Ueber“. Wobei der Punkt „Anleitung“ nicht komplett implementiert ist. Unter „Sprache“ gibt es die Möglichkeit zwischen „Deutsch“ und „Englisch“ zu wählen.

## LST Anzeigefeld

Um den Simulator ein Programm simulieren zu lassen, muss eine nach bestimmten Regeln formatierte Datei mit .lst-Endung geladen werden. Um dies zu tun, klickt man im Hauptfenster des Simulators auf den Menüpunkt „Datei“ und anschließend auf „Testdatei laden“. Nun öffnet sich ein Dateiauswahlfenster, in dem man dann eine geeignete Datei auswählt.

Abbildung 3

Die Textdatei wird Zeile für Zeile in einer Tabellenform in das entsprechende Teilfenster(s. rote Markierung) geladen. Die Tabelle besteht aus drei Spalten, zum einen die Nummerierung, welche ab 1 aufsteigend erfolgt. In der zweiten Spalte befinden sich die entsprechenden Breakpoints, welche als Kästchen zum Anhaken implementiert sind. Falls sich eine Zeile um einen Kommentar handelt, so wird der zugehörige Breakpoint deaktiviert. Die dritte Spalte entspricht eins zu eins den Inhalten der geladenen .lst-Datei. Der als nächstes auszuführende Befehl wird durch eine türkise Färbung markiert.

## SFR und W Registertabelle

SFR (Special Function Register) ist ein Register mit Variablen, welche von der CPU (Central Processing Unit) und Peripheriegeräten genutzt werden um die Operationen des Mikroprozessors zu steuern.

Abbildung 4

Das W-Register wird genutzt um Zwischenergebnisse zu speichern und ist flüchtig. Alle Werte werden in der Tabelle als Dezimalwerte angezeigt.

## STATUS, OPTION und INTCON Registertabelle

Die drei Teilregister des SF-Registers STATUS, OPTION und INTCON werden anhand ihrer einzelnen Bits beurteilt, weshalb sie in der Benutzeroberfläche explizit dargestellt werden, um die einzelnen Bits besser ablesen zu können. Eine 0 entspricht einem logischen False (clear) und eine 1 dem logischen True (set).

Abbildung 5

## RAM (Datenspeicher) Tabelle

Das RAM ist der Datenspeicher für Werte, die eine längere Zeit gespeichert werden sollen und später verwendet werden. Er besteht aus zwei Bänken, die allerdings in der GUI visuell nicht leicht voneinander unterschieden werden können. Jede Bank umfässt eine Speicherkapazität von 128 Byte. Wobei bei einem Byte der Wertebereich auf 0 – 255 begrenzt ist.

Abbildung 6

## PORT A und PORT B Feld

Port A und B bestehen aus einem Register für die Ein- und Ausgänge und einem Register, welches bestimmt, ob die Ein-/Ausgänge als Ein- oder Ausgang definiert sind. Pin 4 von Port A wird auch „externer Zählereingang“ genannt, denn er kann dazu genutzt werden den Timer0 zu inkrementieren, falls das T0CS-Bit Pin 4 zugewiesen ist. Außerdem ist Pin 0 von Port B der „externe Interrupteingang“, der abhängig vom INTEDG-Bit bei einer steigenden oder einer fallenden Flanke einen Interrupt auslöst. Zusätzlich wird ein Interrupt ausgelöst, wenn an einem der vier Pins 4 bis 7 von Port B eine Zustandsänderung stattfindet. Allerdings müssen noch die Bits GIE und INTE oder RBIE gesetzt sein.

Abbildung 7

## STACK Tabelle

Abbildung 8

Der Stack besteht aus acht Einträgen, die hier initial -1 sind, da bei dieser Implementierung bei einer Operation, bei der ein Eintrag vom Stack geholt wird geprüft wird, ob der Stack leer ist und wenn ja, wird eine Exception geworfen. Ansonsten wird der Eintrag, auf den der Stackpointer zeigt, aus dem Stack geholt und in den Programmzähler geschrieben.

## Laufzeitzähler Feld und Quarzfrequenz Drop-Down

Abbildung 9

Der Laufzeitzähler ist ein Aufwärtszähler im Millisekundenbereich, der ab 0 beginnt zu zählen. Der Wert, der hinzuaddiert wird, ist abhängig von der eingestellten Quarzfrequenz und der Zykluszeit eines Befehls. Die Möglichkeiten sind: 32kHz => 0.125ms, 100kHz => 0.04ms, 500kHz => 0.008ms, 1MHz => 0.004ms, 2MHz => 0.002ms, 4MHz => 0.001ms, 8MHz => 0.0005ms, 12MHz => 0.0003ms, 16MHz => 0.00025ms und 20MHz => 0.0002ms.

## Simulatorsteuerungsfeld

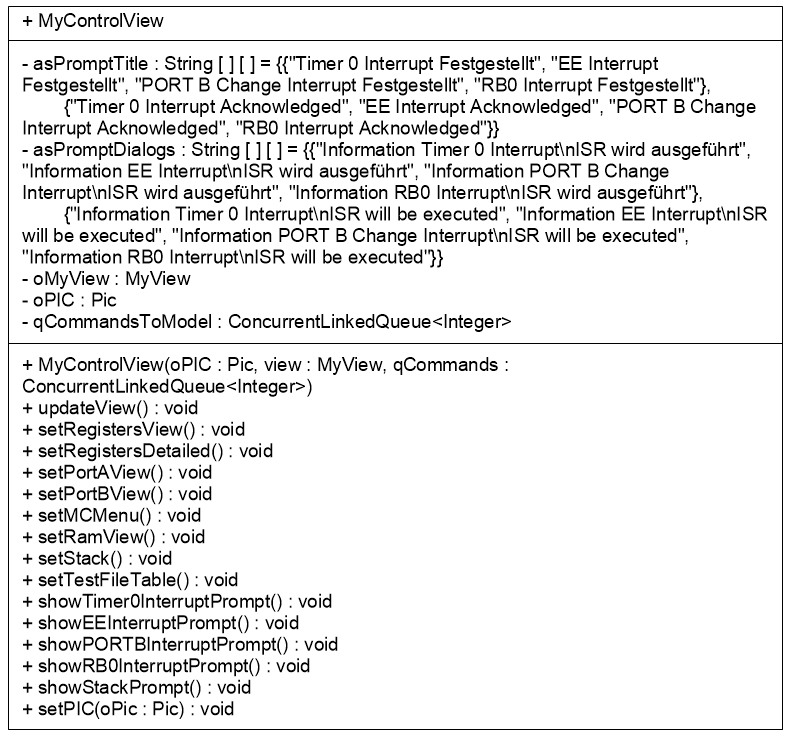
Das Simulatorsteuerungsfeld besteht aus 4 Schaltflächen. „Start“ startet die Programmausführung, wobei der Simulator Schritt für Schritt durch den Programmspeicher läuft, der die Befehle der vorher geladenen Testdatei enthält. Falls kein Breakpoint gesetzt ist, läuft das Programm weiter, bis es terminiert oder in einer Endlosschleife.

# Implementierung

Bei der Implementierung des Projekts wurde versucht die Programmstruktur nach der MVC (Model, View, Controller) – Architektur zu realisieren Dies ist zu einem gewissen Grad gelungen, wobei jedoch die Schnittstellen drei ConcurrentLinkedQueues sind. Über diese werden Änderungen an der GUI an das Backend gesendet und vom Backend Informationen an das Frontend.

## Klasse MyControlModel

## Klasse MyControlView



## Klasse Alu

## Klasse Eeprom

## Klasse EepromThread

## Klasse InstructionDecoder

## Klasse Pic

## Klasse ProgramMemory

## Klasse Ram

## Klasse Stack

## Klasse Time

## Klasse ReadProgramFile

## Klasse MyModel

## Klasse MyModelData

## Klasse ProgramStepInformation

## Klasse Main

## Klasse GUIAbout

## Klasse GUIHelp

## Klasse GUIMainFrame

## Klasse GUIMCMenu

## Klasse GUIMenuBar

## Klasse GUIPorts

## Klasse GUIProgramMemory

## Klasse GUIRamTable

## Klasse GUIRegister

## Klasse GUIRegistersDetailed

## Klasse GUIStack

## Klasse GUITestFileTable

## Klasse GUITime

## Klasse IMyView

## Klasse MyColors

## Klasse MyView

