PIC Simulator

Inhalt

[Vorwort 2](#_Toc483847001)

[Was ist ein Simulator ? 2](#_Toc483847002)

[Die Benutzeroberfläche 2](#_Toc483847003)

Vorwort

Im dritten Semester des Studiengangs Informationstechnik an der DHBW Karlsruhe wurde im Modul Rechnertechnik der Microcontroller PIC16F84 vorgestellt. Im vierten Semester galt es einen Simulator des in der Vorlesung behandelten Microcontrollers zu programmieren. Es sollen die wichtigsten Funktionen des PICs nachimplementiert werden. Während des Programmablaufs werden alle wichtigen Speicherbänke, Register und Ports auf einer grafischen Benutzeroberfläche dargestellt. Die Arbeit erfolgte in zweier Teams.

# Was ist ein Simulator ?

Ein Simulator wird beispielsweise eingesetzt, wenn sich ein reales System nicht direkt

beobachten lässt. Durch die Simulation kann das interne Verhalten dieses Systems

dargestellt und nachvollzogen werden.

Der PIC-Simulator hat vielerlei Vorteile, jedoch auch Nachteile.

Zu den Vorteilen zählt vor allem, dass der Simulator ein Entwicklungswerkzeug darstellt.

Assembler-Programme für den PIC können somit auf Korrektheit geprüft werden. Für

Anfänger kann der Simulator zum Verstehen eines Microcontrollers beitragen, außerdem

werden Kosten eingespart, da kein realer Microcontroller vorliegen muss.

Andererseits ist ein Nachteil von Simulatoren, dass diese Funktionen nur einfach nachbilden.

Sie entsprechen nicht in allen Fällen der realen Welt.

Für den zu implementierenden PIC-Simulator steht fest, dass dieser die Hauptfunktionen

unterstützen und korrekt arbeiten soll. Es ist aber auch unverkennbar, dass er nicht zu

hundert Prozent den realen Microcontroller darstellen wird.

## Vorteile einer Simulation

Durch Simulationen lassen sich auch gefährliche reale Gegebenheiten sicher nachstellen. So werden Simulationen zur Ausbildung von Piloten genutzt oder es werden mit Crash-Test-Dummies Verkehrsunfälle simuliert. Darüber hinaus lassen sich Abläufe verlangsamt darstellen und sind somit leichter nachvollziehbar.

Im Falle das PIC16 Mikrocontroller können Programme für diesen Controller vorab getestet und debuggt werden und somit Fehlerquellen vor dem aktiven Einsatz beseitigt werden.

## Nachteile einer Simulation

Eine Simulation ist immer der Begrenztheit der Mittel unterworfen. Sei es nun Rechenleistung, Zeit oder Geld. Somit kann die Realität oftmals nur in einem vereinfachten Modell nachgebildet werden. Durch die Vereinfachung des Modells sind auch die Messergebnisse von einer gewissen Ungenauigkeit. Die Modelle für eine Simulation werden für einen gewissen Parameterbereich entwickelt. Die Anwendung eines Simulators für Modelle außerhalb seines Parameterbereiches kann zu von der Realität verschiedenen Ergebnissen führen.

Für den PIC16-Simulator gilt, dass er fehlerfrei und möglichst genau arbeiten muss, denn Fehler innerhalb des Simulators lassen falsche Rückschlüsse auf das Programm zu. Ebenso arbeitet der Simulator verlangsamt und kann nicht die realen 2

Zeiten abbilden. Das bedeutet, dass die Laufzeit eines Programms in der Realität nicht der des Simulators entspricht.

## Mikrocontroller

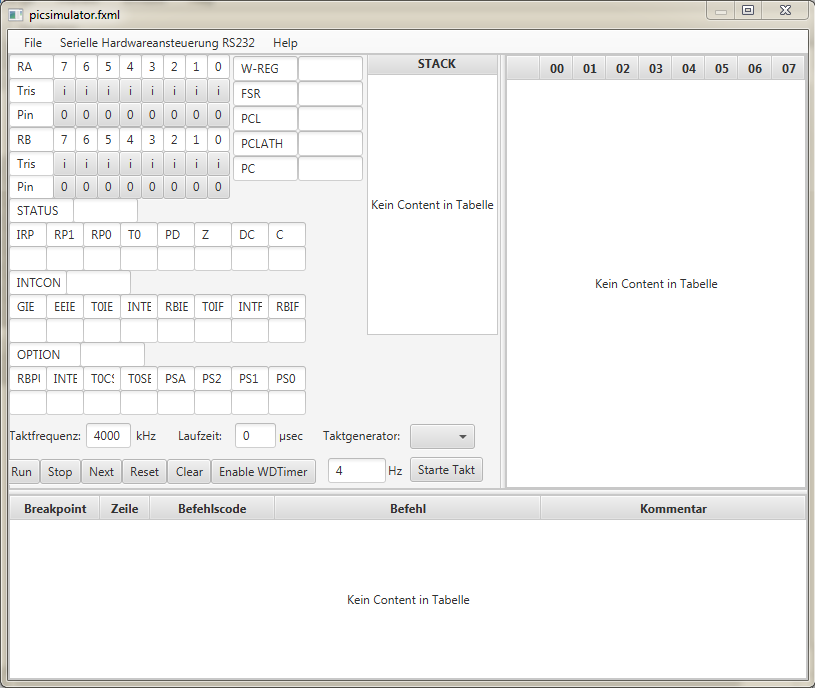
Ein Mikrocontroller ist eine Art Mikrorechnersystem, bei welchem neben ROM und RAM auch Peripherieeinheiten wie Schnittstellen, Timer und Bussysteme auf einem einzigen Chip integriert sind.

Die Hauptanwendungsgebiete sind die Steuerungs-, Mess- und Regelungstechnik, sowie die Kommunikationstechnik und die Bildverarbeitung. Mikrocontroller sind in der Regel in Embedded Systems, in die Anwendung eingebettete Systeme, und somit in der Regel von außen nicht sichtbar. Ebenso verfügen sie, im Gegensatz zum PC, nicht über eine direkte Bedien- und Prorgrammierschnittstelle zum Benutzer. Sie werden in der Regel einmal programmiert und installiert.

# Die Benutzeroberfläche

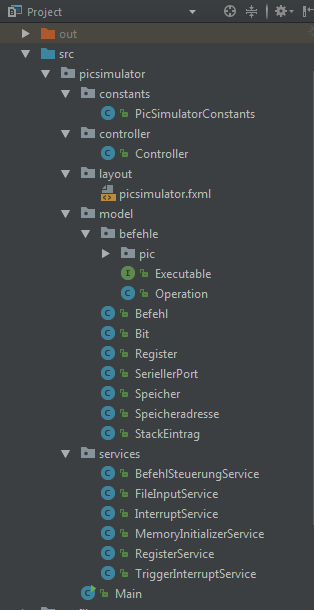
Die Benutzeroberfläche lässt sich in drei Bereiche unterteilen. Im erste Bereich, unteres Drittelwird der Programmcode angezeigt. Darüber sind die Steuerbuttons des Simulators implementiert. Auf der rechten Seite des Fensters, wird RAM-Speicher in Form einer Tabelle abgebildet.

Der Großteil der Benutzeroberfläche nehmen die SFR (Special Function Register)in Anspruch. Darin werden die Register Bänke, sowie besondere Register wie (Status, Intcon, Option)angezeigt. Auch der Stack wird durch eine Tabelle symbolisiert.



# Realisierung

### Projektstruktur



## picsimulator.xml

In der picsimulator.xml wird das Layout unserer Oberfläche beschrieben. Buttons sowie alles was der Benutzer auf der Oberfläche sieht, wird dort erstellt.

## Controller

Controller, welcher direkt von der grafischen Oberfläche angesteuert wird, mit

dieser interagiert und die entsprechende Logik in den Services ansteuert.

## Services

Die Service-Klassen sind jeweils für die Logischen Funktionen zur Verarbeitung der Bits.

Aufgaben:

* Auswertung und Ansteuerung der Befehle
* Laden und Auswerten eines LST-Files
* Überprüfung von Interrupts
* Initialisierung des Speichers
* Änderungen die entsprechenden Bits für die Triggerung eines Interrupts

## Operation (Befehle Vaterklasse):

Vaterklasse der einzelnen Befehle, welche alle von dieser erben. Hier werden allgemeingültige Variablen sowie Methoden definiert, die von allen Befehlen benötigt werden.

## Befehle

Im Ordner befehle ist für jeden Befehl eine eigene Klasse, die die Logik des Befehls nachstellt.

# Gesamtübersicht

Hier wird die gesamt Struktur der Klasse, sowie die Verbindung untereinander dargestellt.

# Funktionalitäten

## Code einlesen

## Befehlsabarbeitung

## Interrupts

Bei einem Interrupt verlässt der PIC16F seine normale Routine und springt in eine Interruptroutine, die er abarbeitet um dann wieder an die Stelle des normalen Ablaufs zurückzukehren.

Bei einem Interrupt müssen von Seiten des Simulators immer die gleichen Dinge ausgeführt werden. Der aktuelle Programm-Counter muss auf den Stack geschrieben werden und in den Programm-Counter muss die Adresse eingetragen werden, in der die Interruptroutinen liegen.

Bevor aber ein Interrupt wirklich ausgelöst wird, wird erst geprüft ob das Global-Interrupt-Enable (GIE) Bit gesetzt ist.

Es wird im Folgenden beschrieben wie die Interrupt Funktionalität für die einzelnen Interrupttypen implementiert wurde.

Da laufend geprüft werden muss, ob ein Interrupt vorliegt, wurde eine Methode checkInterrupt() geschrieben.

Die Methode checkInterrupt() prüft alle möglichen Interruptquellen.

Bevor auf einen RB0 Interrupt oder einen RB Port Change Interrupt überprüft wird, wird geprüft, ob im INTCON-Register das Interrupt Enable Bit (INTE) für einen RB0 Interrupt oder das RBIE für einen RB Port Change Interrupt gesetzt sind.

# Befehlsbeschreibung

## BTFSC (f,b)

## GOTO (k)

## MOVF (f,d)

## RLF(f,d)

## SUBWF (f,d)

## DECFSZ(f,d)

## IORLW (k)

## ZeroBit, Digitcarry und Carry Funktionen

# Zusammenfassung

Umsetzung

Zu Beginn des Projektes wurde ein Pflichtenheft erstellt in dem die Muss,

Kann und Abgrenzungskriterien festgelegt wurden. Diese wurden soweit alle

eingehalten. Zu den Muss Kriterien gehrten:

\_ Quellcode sichtbar anzeigen, einlesen und ausführen

\_ Einzelschritte, Start, Stopp

\_ Register

\_ Ports

\_ Flags anzeigen

\_ Interrupt

\_ Hilfe anzeigen

\_ externer Takt

Die Abgrenzungen müssen soweit weiterhin eingehalten werden. Der Simulator

funktioniert nur mit einem korrekt funktionierenden Programmcode,

und er übernimmt keinerlei Aufgaben eines Compilers.

## Fazit

Durch verschiedene Erfahrungslevel im Bereich Software Entwicklung teilten

wir die Aufgaben von Beginn an untereinander auf wie z.B. GUI Programmierung,

Prozessor Programmierung und Dokumentation. Zeitaufwändigere

Abschnitte wurden jedoch auch zusammen realisiert (wie z.B. Befehle ausprogrammieren).

Durch Projekte in anderen Studienfächern konnten wir einige

Erfahrungen zwischen den Projekten im Bereich Entwicklungsumgebung

oder Versionierung (Git) austauschen. Zeitlich lag das Projekt parallel zu

einem anderen im Fach Software Engineering wodurch es oftmals durch unerwarteten

Problemen die zunächst aufwendiger erschienen als sie wirklich

waren die Zeit knapp. Jedoch konnten Zum Schluss alle zuvor im Pflichtenheft bestimmten Muss-Kriterien erfüllt werden.