Dokumentation PIC16F84 Simulator

Version: 0.1

Autor: Noah Disch

Autor: Eduard Wayz

1 Inhaltsverzeichnis

2 Einleitung

In dem vorliegenden Dokument wird die Entwicklung und Implementierung eines Simulators in Java für den Mikrocontroller PIC16F84 dargelegt. Der Fokus liegt hierbei auf den wesentlichen Funktionen und Eigenschaften des Simulators sowie den gewonnenen Erkenntnissen aus der Programmierung. Zudem werden die Herausforderungen aufgezeigt, denen wir bei der Entwicklung des Simulators begegnet sind.

3 Allgemeines

3.1 Funktionsweise eines Simulators

3.2 Vor- und Nachteile der Simulation

3.3 Beschreibung der Programmoberfläche

4 Realisierung

4.1 Grundkonzept des Simulators

4.2 Struktur und Gliederung

4.3 Programmstruktur und Ablaufdiagramme

4.4 Beschreibung der verwendeten Variablen

4.5 (Auswahl und Begründung der Programmiersprache)

4.6 Detailbeschreibung ausgewählter Funktionen BTFSx, CALL, MOVF, RRF, SUBWF, DECFSZ, XORLW

4.7 Flags und deren Implementierung

4.8 Interrupts (Auszu aus dem Listing)

Beschreibung und Diagramme

Eine wesentliche Herausforderung bei der Erstellung der grafischen Benutzeroberfläche (GUI) war die notwendige kontinuierliche Aktualisierung der Oberfläche nach jedem einzelnen Schritt. Um den gewünschten Ablauf zu gewährleisten, war es erforderlich, eine öffentlich zugängliche, statische Instanz innerhalb der Klasse zu initialisieren, welche auf sich selbst referenziert (z. B. im PortController). Diese Instanz musste einmalig initialisiert werden, um anschließend im ButtonsController die Methode `buildUI()` des PortControllers aufzurufen. Ohne diese spezifische Struktur wäre eine fortlaufende Aktualisierung des Views nach jedem Schritt nicht möglich gewesen. Die Identifikation und Implementierung dieser Logik war daher für alle betroffenen Controller zwingend erforderlich, was jedoch ein tiefgehendes Wissen in der Strukturierung und im Zusammenspiel von JavaFX-Komponenten erforderte.

Ene weitere signifikante Herausforderung stellte das korrekte Größenmanagement der GUI-Elemente dar. Wie im nachfolgenden Kapitel detailliert beschrieben wird, entschieden wir uns bewusst gegen die Verwendung eines UI-Builders und gestalteten das Projekt eigenständig. Dies erforderte eine gut abgestimmte Kombination von Eigenschaften innerhalb der `.fxml`-Dateien, der verknüpften CSS-Dateien sowie der Controller-Klassen. Dabei war es insbesondere herausfordernd, die geeigneten Stellen für bestimmte Attribute zu identifizieren. Oftmals funktionierten Attribute wie `vgrow="ALWAYS"` oder `fillHeight="true"` zunächst nicht wie erwartet, da sich später herausstellte, dass sie an insgesamt drei verschiedenen Stellen definiert werden musste. Die systematische Ermittlung dieser Zusammenhänge erwies sich als zeitaufwendig und erforderte mehrere Stunden intensiver Fehlersuche und Experimente. Zusätzlich bestand die Anforderung von uns, ein responsives Layout umzusetzen, um sicherzustellen, dass die Benutzeroberfläche auf unterschiedlichen Bildschirmauflösungen und variablen Fenstergrößen richtig dargestellt wird. Dies erforderte den gezielten Einsatz von JavaFX-Komponenten wie z. B. GridPane, wobei wir

Layout-Attribute wie GridPane.rowSpan="4" und percentWidth="10" nutzen mussten, um eine flexible und anpassungsfähige Darstellung der GUI zu erreichen

5.3 Persönliche Erfahrung und Reflexion

6 Zeitmanagement

6.1 Beschreibung des zeitlichen Projektverlaufs

7 Versionsverwaltung

7.1 Beschreibung der verwendeten Versionsverwaltung

8 Anhang

8.1 Programmlisting

8.2 Weitere Anlagen

Auszüge aus dem Datenblatt