PIC16F84-Simulator

Author: André Schmitt, Dominik Vogel

DHBW Karlsruhe

TINF15B3

Inhaltsverzeichnis

[Arbeitsweiße 1](#_Toc483831737)

[Vor- und Nachteile 2](#_Toc483831738)

[Programmoberfläche und deren Handhabung 3](#_Toc483831739)

[Realisierung 4](#_Toc483831740)

[Grundkonzept 4](#_Toc483831741)

[Gliederung 4](#_Toc483831742)

[Programmstruktur 4](#_Toc483831743)

[Programmiersprache 4](#_Toc483831744)

[Beschreibung der Funktionen 4](#_Toc483831745)

[Flags 4](#_Toc483831746)

[Interrupts 4](#_Toc483831747)

[TRIS-Register 4](#_Toc483831748)

[Breakpoints 4](#_Toc483831749)

[Hardware Ansteuerung 4](#_Toc483831750)

[State-Machine / EEPROM 4](#_Toc483831751)

[Fazit 4](#_Toc483831752)

# Arbeitsweiße

# Vor- und Nachteile

# Programmoberfläche und deren Handhabung

TODO: 3

# Realisierung

## Grundkonzept

## Gliederung

## Programmstruktur

Das Programm untergliedert sich grob in drei Teile. Dem Hauptfenster, dem emulierten Prozessor und der Verknüpfung des Prozessors mit externer Peripherie.

### Hauptfenster

Das Hauptfenster wird von der Klasse **TfrmMain** repräsentiert. Sie beinhaltet neben der Benutzeroberfläche auch eine Instanz der **TProcessor** Klasse. Des Weiteren, können über das Hauptfenster diverse Peripherie-Emulationen geöffnet werden.

### Prozessor

Der Prozessor wird durch die **TProcessor** Klasse dargestellt. In dieser Klasse sind alle, für die vollständige Emulation eines PIC16F84-Prozessors, relevanten Felder und Funktionen implementiert.

Darunter fallen unter anderem:

* Programmspeicher inklusive Programmzähler und Stack
* Working-Register
* File Map inklusive Special-Function-Register
* Externer EEPROM inklusive State-Machine
* Pins (mithilfe der **TPin**/**TPinArray** Klassen)
* WatchDog-Timer
* Breakpoints
* Interrupts

### Peripherie

Verschiedene Peripherien können einfach hinzugefügt werden, indem man von der **TPeripheralForm** Klasse erbt und die virtuellen Funktionen für Benutzereingaben, Visualisierung und Logik überschreibt.

In der aktuellen Version der Anwendung sind zwei Peripherien verfügbar. Ein Button-Panel, für externe Eingaben am Prozessor und ein LED-Array, um eine vom Prozessor generierte Ausgabe anzuzeigen.

Die Verbindung zwischen den Pins am Prozessor und den Pins eines Peripherie-Objekts können einfach per „drag’n‘drop“ hergestellt werden. Es ist sowohl möglich, Pins einzeln zu verbinden, aber es können auch direkt alle Pins auf einmal verbunden werden.

## Programmiersprache

## Beschreibung der Funktionen

### Rechen Operationen

Es werden Addition (**addwf**/**addlw**) und Subtraktion (**subwf**/**sublw**) unterstützt. Beide nutzen eine weitere Funktion, welche eine Addition durchführt, das Ergebnis zurückliefert und dabei auch die Status-Flags setzt.

### Logische Operationen

Für logische Operationen werden die nativ von Lazarus unterstützen Operatoren benutzt:

* **and 🡪**  **andwf**, **andlw**
* **or 🡪 iorwf**, **iorlw**
* **xor 🡪 xorwf**, **xorlw**
* **not 🡪 comf**
* **shl/shr 🡪 rlf**, **rrf** (rotate passiert nicht automatisch)

### Bit Operationen

Für Bitorientiere Operationen wurde eine Property angelegt, welche es erlaubt auf die einzelnen Bits der innerhalb der Filemap direkt zuzugreifen.

* Set 🡪 **bfs**
* Clear 🡪 **bfc**

### Andere zuweisende Operationen

Folgende Befehle nutzen lediglich die Lazarus eigene Zuweisung, um ihren Zweck zu erfüllen:

* Nullsetzend 🡪 **clrf**, **clrw**
* Kopierend 🡪 **movwf**, **movlw**, **movf**
* Inc/Decrement 🡪 **incf**, **decf**
* Swap 🡪 **swapf**

### Sprung Operationen

* Direkter Sprung 🡪 **goto**, **call**, **return**, **retlw**, **retfie**
* Bedingter Sprung 🡪 **incfsz**, **decfsz**¸ **btfsc**, **btfss**

### Sleep

Der Sleep-Befehl ist mittels eines Flags implementiert, welches während einem Sleep-Befehl gesetzt wird. Dieses Flag verhindert jegliche Ausführung von Befehlen. Wird jedoch ein Interrupt erkannt, so wird dieses Flag wieder zurückgesetzt, wodurch Befehle wieder ausgeführt werden.

Auch der Watchdog-Timer kann den Sleep-Modus beenden. Dieser Timer kann mittels des Befehls **clrwdt** zurückgesetzt werden.

## Flags

## Interrupts

## TRIS-Register

## Breakpoints

## Hardware Ansteuerung

n/a

## State-Machine / EEPROM

# Fazit