|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| DHBW-Logo  Entwicklung eines Simulators für den PIC16F4 Microcontroller  Für die Prüfung zum  Bachelor of Engineering  des Studiengangs Informatik Studienrichtung Informationstechnik  an der  Dualen Hochschule Baden-Württemberg Karlsruhe  von  Paul Giesa & Chris Steven Todt  Bearbeitungszeitraum:  03.04.2017 - 29.05.2017   |  |  | | --- | --- | | Kurs | TINF15B3 | | Gutachter/In der Studienakademie | Hr. Lehmann | |  |  | |  |  | |

Inhaltsverzeichnis

Inhaltsverzeichnis 2

Abbildungsverzeichnis 3

Es konnten keine Einträge für ein Abbildungsverzeichnis gefunden werden. 3

Tabellenverzeichnis 3

Abkürzungsverzeichnis 4

1. Einleitung 5

Literaturverzeichnis 7

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

BTFSC Bit Test File, Skip if Clear

BTFSS Bit Test File, Skip if Set

CALL Call Subroutine

DECFSZ Decrement File, Skip if Zero

PAP Programmablaufplan

RRF Rotate Right f through Carry

SUBWF Subtract W from f

XORLW Exclusive OR Literal with W

# Vorwort

Im Fach Rechnertechnik II soll der Aufbau und die Funktionsweise eines Microcontroller gelernt werden. Über den „Umweg“ ein Simulator-Programm zu schreiben, das die Funktionen eines realen oder imaginären Controllers nachbildet, müssen die Studenten neben dem Studium des Datenblattes auch die bereits erlernten Fertigkeiten aus der Vorlesung Software-Engineering, Digitaltechnik und Rechnertechnik I anwenden. Eine einfache Hardwarebeschaltung an der seriellen oder parallelen Schnittstelle bildet die Brücke zwischen virtueller und realer Welt.

# Einleitung

Im Rahmen der Vorlesung ……. Sollen die Studenten einen Simulator für den Microcontroller …. Der Firma …. Entwickeln.

# Allgemeines

Allgemeines:

Grundsätzliche Arbeitsweise eines Simulators.

Vor- und Nachteile einer Simulation.

Programmoberfläche und deren Handhabung.

(Was ist ein Microcontroller?)

# Simulator

## Aufbau

GUI

# Struktur

# Implementierung der Befehle

Die Realisierung der Maschienenbefehle des Microcontrollers wird beispielhaft an den folgenden Befehlen tiefgehender erläutert:

* BTFSC (Bit Test, Skip if Clear)
* BTFSS (Bit Test, Skip if Set)
* CALL (Call Subroutine)
* MOVF (Move f)
* RRF (Rotate Right f through Carry)
* SUBWF (Subtract W from f)
* DECFSZ (Decrement f, Skip if 0)
* XORLW (Exclusive OR Literal with W)

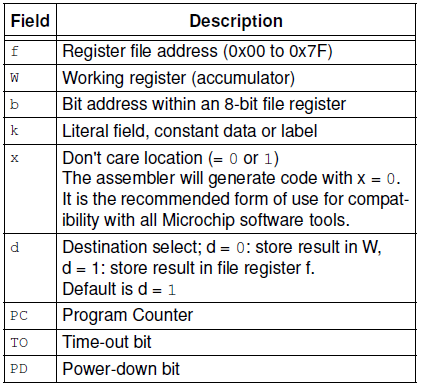


Abbildung : Abkürzungen – Auszug PIC

## C:\Users\Chris Todt\AppData\Local\Microsoft\Windows\INetCache\Content.Word\BTFSC.PNGBTFSC (Bit Test, Skip if Clear)

Abbildung : BTFSC – Auszug PIC

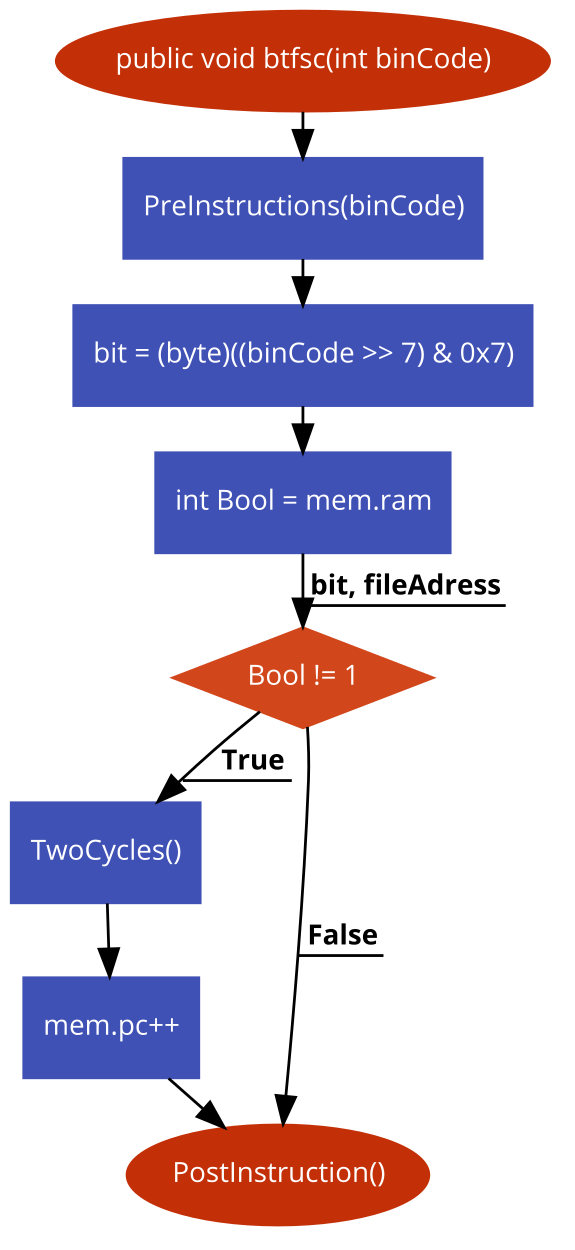


Abbildung : PAP BTFSC

**Implementation**:

public void btfsc(int binCode)

{

PreInstructions(binCode);

bit = (byte)((binCode >> 7) & 0x7);

int Bool = mem.ram[bit, fileAdress];

if (Bool != 1)

{

TwoCycles();

mem.pc++;

}

PostInstruction();

}

## C:\Users\Chris Todt\AppData\Local\Microsoft\Windows\INetCache\Content.Word\BTFSS.PNGBTFSS (Bit Test, Skip if Set)

Abbildung : BTFSS – Auszug PIC

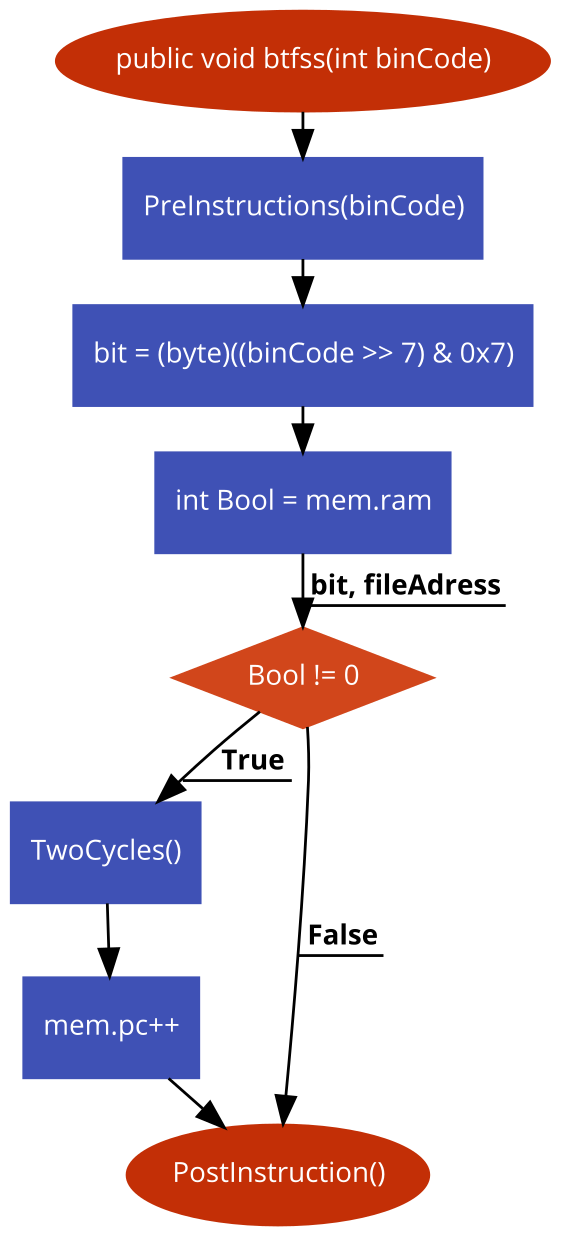


Abbildung : PAP BTFSS

**Implementation**:

public void btfss(int binCode)

{

PreInstructions(binCode);

bit = (byte)((binCode >> 7) & 0x7);

int Bool = mem.ram[bit, fileAdress];

if (Bool != 0)

{

TwoCycles();

mem.pc++;

}

PostInstruction();

}

## C:\Users\Chris Todt\AppData\Local\Microsoft\Windows\INetCache\Content.Word\CALL.PNGCALL (Call Subroutine)

Abbildung : CALL – Auszug PIC

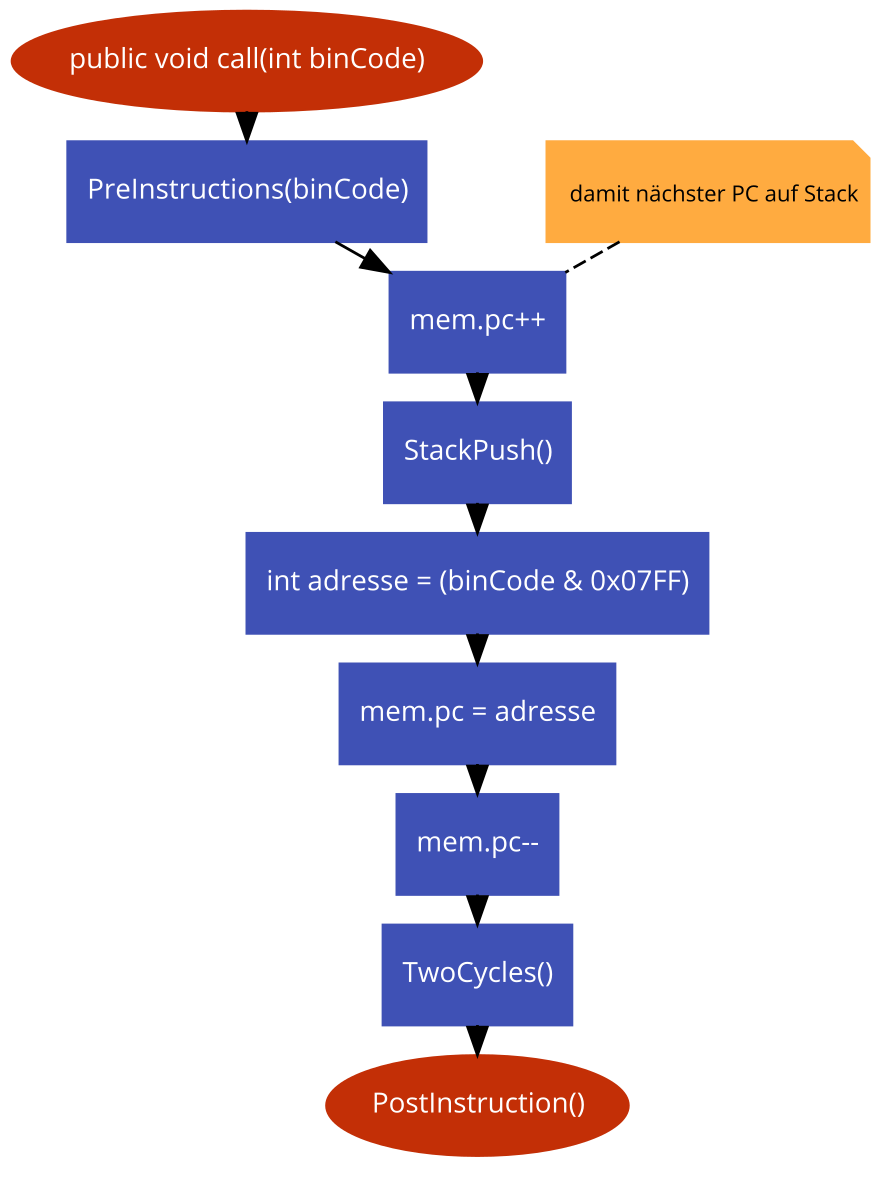


Abbildung : PAP CALL

**Implementation**:

L

public void call(int binCode)

{

PreInstructions(binCode);

mem.pc++; //damit nächster PC auf Stack

StackPush();

int adresse = (binCode & 0x07FF);

mem.pc = adresse;

mem.pc--;

TwoCycles();

PostInstruction();

}

## C:\Users\Chris Todt\AppData\Local\Microsoft\Windows\INetCache\Content.Word\MOVF.PNGMOVF (Move f)

Abbildung : MOVF – Auszug PIC

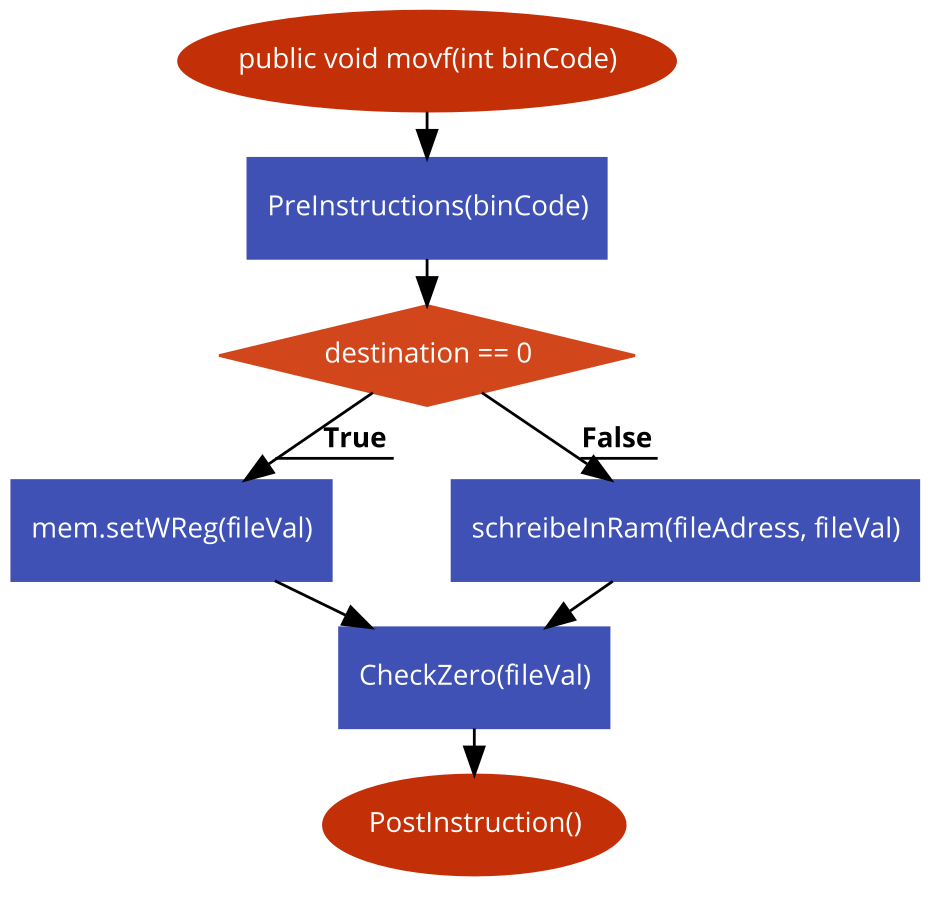


Abbildung : PAP MOVF

**Implementation**:

public void movf(int binCode)

{

PreInstructions(binCode);

if (destination == 0)

{

mem.setWReg(fileVal);

}

else

{

schreibeInRam(fileAdress, fileVal);

}

CheckZero(fileVal);

PostInstruction();

}

## C:\Users\Chris Todt\AppData\Local\Microsoft\Windows\INetCache\Content.Word\RRF.PNGRRF (Rotate Right f through Carry)

Abbildung : RRF - Auszug PIC

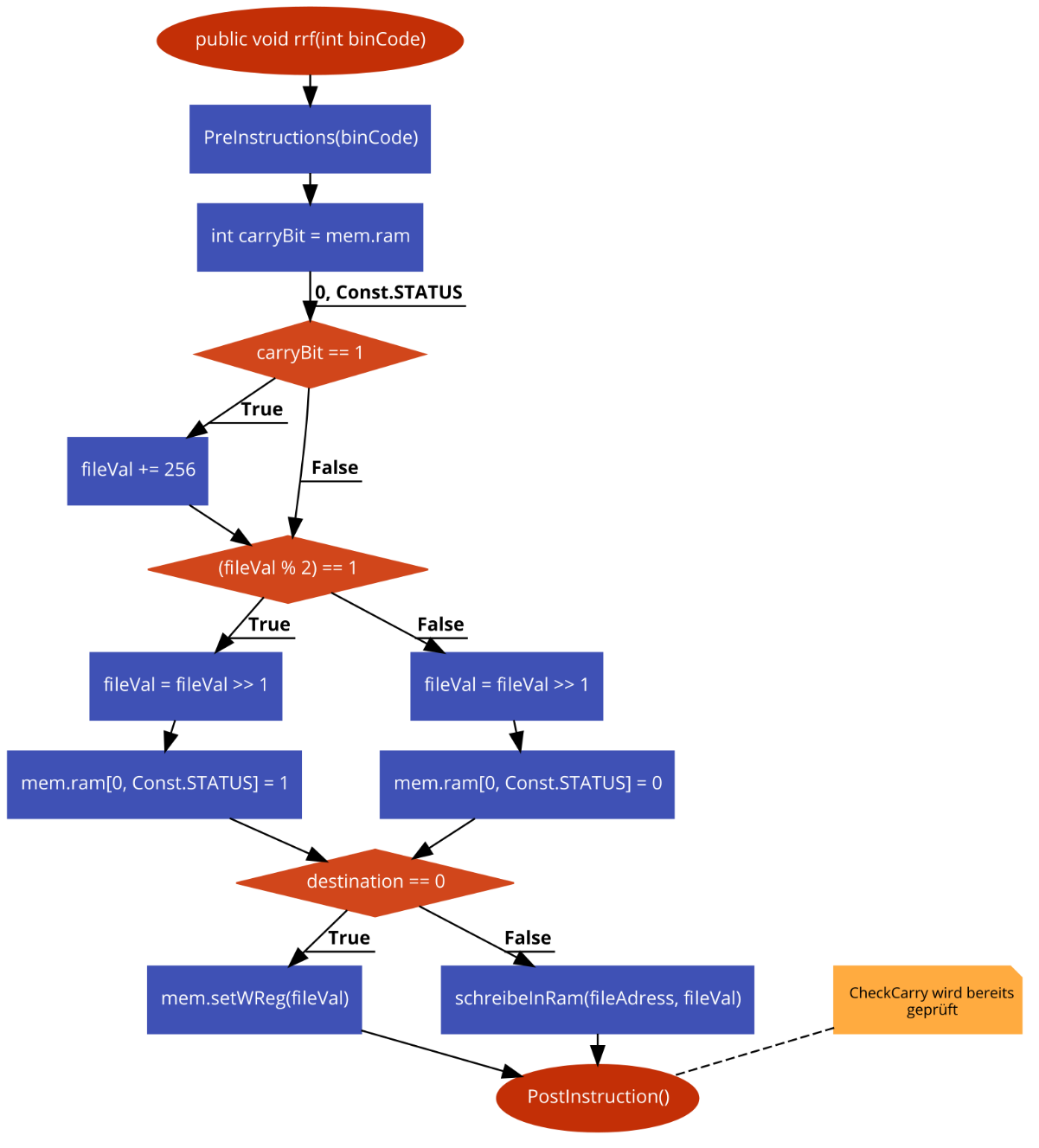


Abbildung : PAP RRF

**Implementation**:

public void rrf(int binCode)

{

PreInstructions(binCode);

int carryBit = mem.ram[0, Const.STATUS];

if (carryBit == 1)

{

fileVal += 256;

}

if ((fileVal % 2) == 1)

{

fileVal = fileVal >> 1;

mem.ram[0, Const.STATUS] = 1;

}

else

{

fileVal = fileVal >> 1;

mem.ram[0, Const.STATUS] = 0;

}

if (destination == 0)

{

mem.setWReg(fileVal);

}

else

{

schreibeInRam(fileAdress, fileVal);

}

//CheckCarry wird bereits geprüft

PostInstruction();

}

## 

## C:\Users\Chris Todt\AppData\Local\Microsoft\Windows\INetCache\Content.Word\SUBWF.PNGSUBWF (Subtract W from f)

Abbildung : SUBWF - Auszug PIC

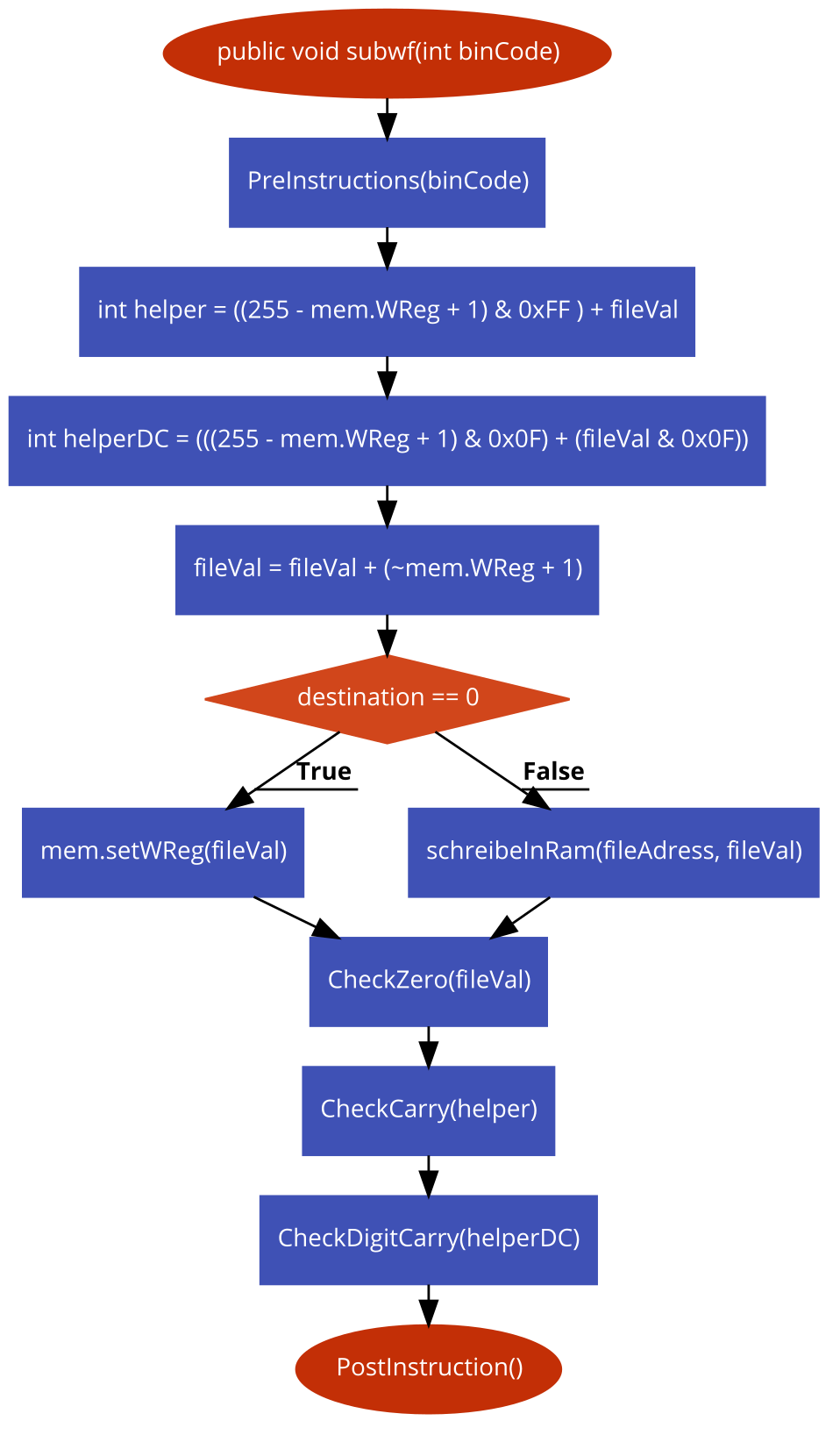


Abbildung : PAP SUBWF

**Implementation**:

public void subwf(int binCode)

{

PreInstructions(binCode);

int helper = ((255 - mem.WReg + 1) & 0xFF ) + fileVal;

int helperDC = (((255 - mem.WReg + 1) & 0x0F) + (fileVal & 0x0F));

fileVal = fileVal + (~mem.WReg + 1);

if (destination == 0)

{

mem.setWReg(fileVal);

}

else

{

schreibeInRam(fileAdress, fileVal);

}

CheckZero(fileVal);

CheckCarry(helper);

CheckDigitCarry(helperDC);

PostInstruction();

}

## C:\Users\Chris Todt\AppData\Local\Microsoft\Windows\INetCache\Content.Word\DECFSZ.PNGDECFSZ (Decrement f, Skip if 0)

Abbildung : PAP DECFSZ

Abbildung : DECFSZ - Auszug PIC

**Implementation**:

public void decfsz(int binCode)

{

PreInstructions(binCode);

if (destination == 0)

{

mem.setWReg(fileVal - 1);

}

else

{

schreibeInRam(fileAdress, fileVal - 1);

}

//neu zuweisen, da oben geändert

fileVal = getFileVal(fileAdress);

if (fileVal == 0)

{

mem.pc++;

TwoCycles();

}

PostInstruction();

}

## C:\Users\Chris Todt\AppData\Local\Microsoft\Windows\INetCache\Content.Word\XORLW.PNGXORLW (Exclusive OR Literal with W)

Abbildung : XORLW - Auszug PIC

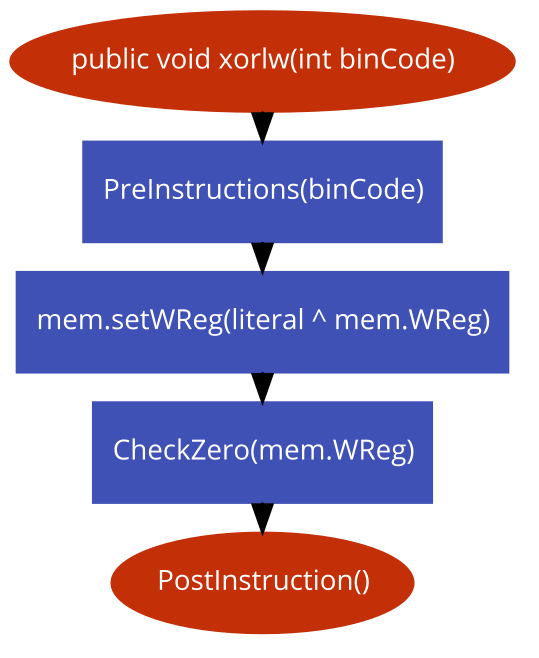


Abbildung : PAP XORLW

**Implementation**:

public void xorlw(int binCode)

{

PreInstructions(binCode);

mem.setWReg(literal ^ mem.WReg);

CheckZero(mem.WReg);

PostInstruction();

}

Syntax: [label] BTFSC f,b

Operanden: 0 ≤ f ≤ 127

0 ≤ b ≤ 7

Befehl : Wenn bit b in f = 0 🡪 überspringe nächsten Befehl

Beeinflusste Flags: ~

Beschreibung

# Programmiersprache

Als Programmiersprache bot sich C# an, da für diese bereits eine Schulung seitens des Südwestrundfunks geboten wurde und durch eine Windows-Forms-Anwendung auf einfache Art und Weise eine GUI entwickelt werden kann.

***MVC?***

# Fazit

Wie weit konnten die Funktionen des Bausteins per Software nachgebildet

werden?

Fazit, persönliche Erfahrung und Erkenntnis. Was passierte während der

Entwicklung des Projektes? Welche Probleme tauchten auf und wie wurden Sie

gelöst. Vermeiden Sie dabei negative Formulierungen. Was würde ich anderst

machen, wenn ich das Projekt nochmals realisieren müsste? (Umfang des Fazits

ca. ¾ bis 1 Seite oder 10 % des Gesamtumfangs)

**TODO: DIAGRAMME UND BESCHREIBUNG INTERRUPTFUNKTION**tiefergehende Beschreibung der Funktionen an Hand ausgewählter Beispiele

( BTFSx, CALL, MOVF, RRF, SUBWF, DECFSZ, XORLW ). Diese und ggf.

weitere Befehle anhand von kurzen Programmsequenzen und Ablaufdiagrammen

erläutern.

# Literaturverzeichnis