

Inhaltsverzeichnis

Einführung16

Kapitel 1 - Überblick über ATtiny-Mikrocontroller26

1.1 ATtiny-Mikrocontroller 26

1.1.1 8-Pin-Bausteine 27

1.1.1.1 ATtiny13 und ATtiny13A 27

1.1.1.2 ATtiny25, ATtiny45 und ATtiny85 27

1.1.2 14-Pin ATtiny24, ATtiny44 und ATtiny84 28

1.1.3 20-Pin-Bausteine 28

1.1.3.1 ATtiny26 28

1.1.3.2 ATtiny261, ATtiny461 und ATtiny861 29

1.1.3.3 ATtiny2313 und ATtiny4313. 29

1.1.4 28-Pin-Bausteine 30

1.1.4.1 ATtiny48 und ATtiny88 30

1.1.4.2 ATtiny28L und ATtiny28V 30

1.2 Welcher ATtiny soll verwendet werden? 30

1.3 Vergleich der ATtiny-Bausteine 30

1.4 Zusammenfassung. 31

Kapitel 2 - Einrichtung von Hardware und Software32

2.1 Microchip Studio installieren 32

2.1.1 Download-Seite und Software-Version. 32

2.1.2 Neueste Software-Version 32

2.1.3 Starten Sie die Installation. 33

2.1.4 Installation 33

2.2 8-pol-ATtiny-Mikrocontroller-LED-Schaltung 33

2.2.1 Stromversorgung 36

2.2.2 Entkopplungskondensator 37

2.3 Ein erstes Assemblerprogramm 37

2.3.1 Starten Sie ein neues AVR Assembler Projekt in Microchip Studio. 37

2.3.2 LED-Blinklicht Assemblercode. 39

2.3.3 Das Projekt erstellen 40

2.3.4 Behebung von Erstellungsfehlern	40
2.3.5 Laden des Programms in den AVR-Mikrocontroller	41
2.3.6 Fehlersuche	41
2.4 Ein erstes C-Programm	42
2.4.1 Starten eines neuen GCC-C-Projekts in Microchip Studio	42
2.4.2 LED-Blinken C-Code	43
2.4.3 Das Projekt erstellen	44
2.4.4 Behebung von Erstellungsfehlern	44
2.4.5 Laden des Programms in den AVR-Mikrocontroller	44
2.4.6 Fehlersuche	45
2.5 Zusammenfassung.	45
Kapitel 3 • Grundlagen der Assemblersprache	46
3.1 Mnemonics	46
3.2 AVR-Befehlssatz	47
3.3 Opcodes und Operanden	48
3.4 Kommentare im Assemblerprogramm	48
3.5 Funktionsweise des LED-Blinkprogramms	49
3.6 Verwendung des Debuggers und Simulators	51
3.6.1 Verwendung des AVR Simulators und Debuggers	52
3.6.2 Verwendung eines realen AVR-Mikrocontrollers mit Debugger	54
3.6.3 Einen AVR wieder in den ISP-Modus versetzen	57
3.7 Zusammenfassung.	57
Kapitel 4 – Binäre Zahlen und Speicher	59
4.1 Bits und Bytes.	59
4.1.1 Zählen im Binärformat.	60
4.1.2 Bytes und Nibbles.	62
4.2 Stellenwertsysteme	62
4.2.1 Stellenwertsystem mit Dezimalzahlen	62
4.2.2 Stellenwertsystem mit Binärzahlen	64
4.3 Hexadezimale Zahlen.	64
4.3.1 Zählen im Hexadezimalformat	65

4.3.2 Hexadezimale Darstellung von Binärzahlen	65
4.4 Berechnung der Wertebereiche	67
4.4.1 Berechnung des Maximalwerts einer Binärzahl	67
4.4.2 Berechnung der Speichergröße.	68
4.4.3 Umrechnung zwischen Bytes und Kilobytes	69
4.5 Zahlen in Assemblerprogrammen	70
4.6 Zahlen in C-Programmen	71
4.7 Zusammenfassung.	71
Kapitel 5 • AVR-Register und Speicher	73
5.1 AVR-Register.	73
5.1.1 Allzweck-Arbeitsregister.	73
5.1.2 I/O-Register.	73
5.2 Verwendung von Arbeitsregistern	73
5.2.1 Addition zweier Register	74
5.2.2 Inkrementieren eines Registers	77
5.3 AVR-Memory Map	78
5.3.1 Programmspeicher	79
5.3.2 Datenspeicher	80
5.3.3 EEPROM.	81
5.4 Zugriff auf SRAM in Assemblerprogrammen	81
5.4.1 SRAM - Beispiel zum Speichern und Laden	82
5.4.2 Beispiel für Speichern und Laden im Simulator.	83
5.4.3 SRAM-Startadressen	84
5.5 Zusammenfassung.	85
Kapitel 6 • Interne Architektur des AVR	86
6.1 Taktimpulse und Programmzähler	86
6.1.1 Taktimpulse	86
6.1.2 Taktperiode und Frequenz	87
6.1.2.1 ATtiny13(A) Standard-Taktfrequenz	87
6.1.2.2 ATtiny25/45/85 Standard-Taktfrequenz	88
6.1.2.3 RC-Oszillator-Genauigkeit	88

6.1.2.4 Taktquellen und Verbesserung der Genauigkeit	88
6.1.2.5 Berechnung der Taktperiode aus der Frequenz.	88
6.1.2.6 Berechnung der Taktfrequenz aus der Periode	89
6.1.3 Der Programmzähler	89
6.2 Mikrocontroller-Busse.	91
6.3 Harvard- und von-Neumann-Architekturen	92
6.5 Statusregister	95
6.5.1 Zero-Flag.	95
6.5.2 Negative Flag	97
6.6 Stapel und Stapelzeiger	98
6.6.1 Wie der Stapel funktioniert	98
6.6.2 Zugriff auf den Stapel mit PUSH und POP	100
6.6.3 Aufruf einer Unteroutine.	103
6.7 LED-Blinkprogramm vollständig erklärt	106
6.8 AVR-Befehlskodierung	108
6.9 Adressierungsmodi	114
6.9.1 Direkte Adressierung der Register.	114
6.9.2 I/O-Direktadressierung	115
6.9.3 Andere Adressierungsmodi.	116
6.10 Zusammenfassung.	116
Kapitel 7 • Arithmetische und logische Befehle.	118
7.1 Positive und negative Zahlen.	118
7.1.1 1er-Komplement-Zahlen	118
7.1.2 2er-Komplement-Zahlen	119
7.2 Addition	120
7.2.1 ADD - Addieren ohne Übertrag	121
7.2.2 ADC - Addieren mit Übertrag	122
7.3 Subtraktion.	123
7.4 Logische Befehle	124
7.4.1 Logisches UND	125
7.4.2 Logisches ODER	126

7.4.3 Logisches Exklusiv-ODER	128
7.5 Sonstige arithmetische und logische Befehle	129
7.6 Zusammenfassung	129
Kapitel 8 • Programmierung der AVR I/O Ports	130
8.1 Befehle für den Zugriff auf I/O-Register	130
8.1.1 Lesen und Schreiben von I/O-Registern mit IN und OUT	130
8.1.1.1 Der IN-Befehl	130
8.1.1.2 Der OUT-Befehl	131
8.1.2 I/O-Register-Bit-Manipulation und -Test.	132
8.1.3 Zugriff auf I/O-Register als Datenspeicher	132
8.2 I/O-Anschlüsse	133
8.2.1 Konfigurieren von I/O-Pins als Ausgänge in Assembler	134
8.2.1.1 8-polige ATtiny-Schaltung mit fünf LEDs	134
8.2.1.2 Programmieren eines ATtiny über debugWIRE oder ISP/SPI	134
8.2.1.3 Aufbau der 5 LED ATtiny-Schaltung auf dem Breadboard	136
8.2.1.4 Assemblercode für den 5-LED-Zähler	136
8.2.1.5 Auswahl des Debuggers oder Simulators.	137
8.2.1.6 Was der LED Count Assembler Code macht.	137
8.2.1.7 Erstellen und Ausführen des Programms.	137
8.2.1.8 Funktionsweise des LED Count Assembler Programms	138
8.2.1.9 Einen Break-Point im Debugger verwenden	140
8.2.2 Konfigurieren von I/O-Pins als Ausgänge in C.	141
8.2.2.1 LED-Zählung C-Code	141
8.2.2.2 Funktionsweise des LED Count C-Programms	142
8.2.2.3 Ausführen des LED Count C-Codes.	142
8.2.2.4 LED Count C Disassembler Code	143
8.2.3 Begrenzung eines Zählwertes.	150
8.2.4 Drei Möglichkeiten, eine LED mit einem AVR umzuschalten	152
8.2.4.1 Umschalten einer LED nach Wert	152
8.2.4.2 Umschalten einer LED durch Exklusiv-ODER	152
8.2.4.3 Umschalten einer LED mit dem PINB-Register	154

8.2.5 Konfigurieren von I/O-Pins als Eingänge	156
8.3 Zusammenfassung.	161
Kapitel 9 • Assembler Features	162
9.1 Befehle und Labels.	162
9.2 Der Präprozessor und die Include-Dateien	163
9.3 Assembler-Direktiven.	165
9.3.1 Festlegen von Code- und Datenpositionen	166
9.3.1.1 ORG-Assembler-Anweisung	166
9.3.1.2 CSEG-Code-Segment-Anweisung	167
9.3.1.3 DSEG-Datensegment und BYTE-Anweisungen	168
9.3.1.4 ESEG EEPROM Segment Anweisung	169
9.3.2 Speicher reservieren	169
9.3.2.1 BYTE	169
9.3.2.2 DB - Konstantes Byte definieren	169
9.3.2.3 DW - Konstantes Wort definieren	176
9.3.2.4 DD - Konstante Doppelwort definieren	176
9.3.2.5 DQ - Konstante Quad-Wort definieren.	176
9.3.3 Definition von Namen für Register mit DEF	176
9.3.4 Ausdrücke mittels EQU und SET mit Namen versehen	176
9.3.5 Bedingte Assemblierung	177
9.4 Weitere Assembler-Elemente	179
9.5 Weiterführende Literatur	179
9.6 Zusammenfassung.	180
Kapitel 10 • AVR Timing, Timer und Interrupts	181
10.1 Zeitlicher Ablauf von Befehlen	181
10.1.1 AVR CPU-Versionen	181
10.1.2 Beispiele für das Zeitverhalten von Befehlen	181
10.1.2.1 Der NOP-Befehl	182
10.1.2.2 16-Bit- und 32-Bit-Befehls-Timing	183
10.1.2.3 Timing von Verzweigungsbefehlen	183
10.2 Assembler Zeitverzögerung.	188

10.2.1 Berechnung der Dauer eines Verzögerungs-Unterprogramms	188
10.2.1.1 Messung des Zykluszählers des Unterprogramms Delay	189
10.2.1.2 Mathematische Formel für das Unterprogramm Verzögerung	190
10.2.2 Ein besseres Software-Verzögerungs-Unterprogramm	200
10.2.2.2 32-Bit-Subtraktion mit 8-Bit-Registern	201
10.2.2.3 Zeitlicher Ablauf des Codes	203
10.2.2.4 Testen des Codes in Microchip Studio	204
10.2.3 Ein variables Software-Zeitverzögerungs-Unterprogramm	204
10.2.3.1 Aufteilung eines Assemblerprojekts in Dateien	208
10.2.3.2 Funktionsweise der Unterroutine wait_ms	208
10.2.3.3 Grenzen der Unterroutine wait_ms	211
10.2.4 Übergabe eines Wertes an eine Unterroutine	224
10.2.4.1 Der einfachste Weg, einen Wert zu übergeben	224
10.2.4.2 Verbesserte Art der Wertübergabe	226
10.3 Aufruf einer Assembler-Unterroutine aus C-Code	233
10.3.1 Wert-Übergabe an ein Assembler-Unterprogramm aus C	234
10.3.1.2 Die Funktionsweise des wait_ms_c-Projektcodes	236
10.3.2 Wert-Rückgabe aus einem Assembler-Unterprogramm in C	238
10.4 Timer-Verzögerung aufrufen	240
10.4.1 Timer/Counter0-Register	240
10.4.1.1 Timer/Counter0 Register Adressen	241
10.4.1.2 Counter/Timer0 als Zeitgeber verwenden	242
10.4.2 Timer-Assemblerprogramm mit Polling	242
10.4.2.2 Links-Verschiebungs-Operator (Left Shift)	243
10.4.2.3 In Register schreiben oder lesen-ändern-schreiben (read-modify-write)	244
10.4.2.4 Initialisierung von Timer 0	245
10.4.2.5 Polling (Abfrage) des Timer0	247
10.4.2.6 Ausführen des Codes im Simulator	248
10.4.3 Polling des Timer0 in einem C-Programm	248
10.5 Timer-Verzögerung mit Interrupt	249
10.5.1 Wie Unterbrechungen funktionieren	249

- 10.5.2 Die Interrupt-Vektor-Tabelle 250
- 10.5.3 Assemblerprogramm Timer-Interrupt 250
 - 10.5.3.1 ATtiny13(A) Timer-Interrupt-Projekt. 250
 - 10.5.3.2 ATtiny25/45/85 Timer-Interrupt-Projekt 255
 - 10.5.3.3 Universal Timer Interrupt Projekt. 256
 - 10.5.3.4 C-Programm Timer Interrupt. 257
- 10.6 Zusammenfassung. 259
- Kapitel 11 • Der AVR-Befehlssatz. 260**
 - 11.1 AVR-Befehlssatz Übersicht und Kategorien 260
 - 11.2 Ein geführter Rundgang durch den ATtiny AVR-Befehlssatz 260
 - 11.2.1 Arithmetische und logische Befehle 260
 - 11.2.1.1 Befehle zum Addieren und Subtrahieren 261
 - 11.2.1.2 Logische Befehle 264
 - 11.2.1.3 Inkrement- und Dekrementbefehle. 264
 - 11.2.1.4 Befehle zur Vorzeichenänderung 264
 - 11.2.1.5 Bits setzen und löschen 265
 - 11.2.1.6 Register setzen, löschen und Testbefehle 266
 - 11.2.2 Verzweigungsbefehle. 267
 - 11.2.2.1 Sprungbefehle 267
 - 11.2.2.2 Befehle zum Aufruf von Unterprogrammen 269
 - 11.2.2.3 Rückkehr aus Unterprogramm und Unterbrechungsbefehl 269
 - 11.2.2.4 Vergleichsbefehle 269
 - 11.2.2.5 Befehle überspringen 270
 - 11.2.2.6 Verzweigungsbefehle 271
 - 11.2.3 Bit- und Bit-Test-Befehle 274
 - 11.2.3.1 Bits setzen und löschen 274
 - 11.2.3.2 Befehle zum Verschieben, Rotieren und Tauschen 274
 - 11.2.3.3 Befehle zum Setzen und Löschen der Statusregister-Bits 279
 - 11.2.4 Befehle zur Datenübertragung 281
 - 11.2.4.1 Befehle zum Verschieben/Kopieren. 281
 - 11.2.4.2 Unmittelbarer Ladebefehl 281

11.2.4.3 Indirekte Ladebefehle	281
11.2.4.4 Indirekte Speicherbefehle	282
11.2.4.5 Indirektes Laden mit Adress-Verschiebung	283
11.2.4.6 Indirektes Speichern mit Verschiebung	283
11.2.4.7 Direktes Laden aus dem SRAM.	283
11.2.4.8 Direktes Speichern im SRAM	284
11.2.4.9 Befehle zum Laden des Programmspeichers	284
11.2.4.10 In den Programmspeicher schreiben	284
11.2.4.11 Port IN und OUT Befehle	284
11.2.4.12 PUSH- und POP-Befehle	284
11.2.5 MCU-Steuerbefehle	284
11.3 Der komplette AVR-Befehlssatz	285
11.3.1 Anzahl der AVR-Befehle	285
11.3.1.1 Gesamtzahl der AVR-Befehle	285
11.3.1.2 Anzahl der ATtiny AVR-Befehle.	286
11.3.1.3 Unstimmigkeiten in der Dokumentation	286
11.3.2 Andere AVR-Befehle	286
11.3.2.1 Arithmetische und logische Befehle	286
11.3.2.2 Verzweigungsbefehle	287
11.3.2.3 Befehle zur Datenübertragung	287
11.3.3 Befehle näher betrachtet	288
11.3.3.1 Der vollständige AVR-Befehlssatz	288
11.3.3.2 Die einfache AVR-CPU.	288
11.3.3.3 Reduzierter AVRrc-Kern	288
11.3.3.4 AVRe und AVRxt Kerne	289
11.3.3.5 AVRe+ CPU	289
11.3.3.6 AVRxm CPU.	289
Kapitel 12 • Software-Tools und Einstellungen	290
12.1 AVR-Assembler	290
12.1.1 Der AVRASM2-Assembler.	290
12.1.1.1 Erstellen eines Projekts mit AVRASM2.	290

12.1.1.2 AVRASM2 auf der Kommandozeile	292
12.1.1.3 AVRASM2 Optionen in Microchip Studio.	293
12.1.1.4 Dateioptionen auflisten	294
12.1.2 Der AVR-AS Assembler	295
12.1.3 Der AVRA-Assembler.	295
12.2 Die GNU C Toolchain	295
12.3 Wohin von hier aus?	296
12.3.1 Erreichte Ziele	296
12.3.2 Was nicht abgedeckt wurde	297
12.3.3 Andere AVR-Mikrocontroller	298
12.3.4 Assembler-Ressourcen.	298
Anhang A - Einrichtung des externen Programmiergeräts	299
A.1 Features des Hobby-USB-Programmiergeräts	299
A.1.1 Unterschiede zwischen Hobby-Programmiergeräten und dem Atmel-ICE	300
A.1.2 Programmierschnittstellen	300
A.1.3 Verwendung von Hobby-USB-Programmiergeräten mit diesem Buch.	301
A.2 Überblick über die Einrichtung des externen Programmiergeräts.	301
A.3 Externes Programmiergerät einrichten	302
A.3.1 Installieren eines Treibers	302
A.3.1.1 USBasp-Treiber	302
A.3.1.2 USBtinyISP-Treiber.	304
A.3.1.3 Arduino Uno Sketch	306
A.3.2 Herunterladen und Installieren von avrdude.	307
A.3.3 Eine AVR-Schaltung aufbauen und einen Programmer anschließen	308
A.3.3.1 Anschließen einer USBasp.	308
A.3.3.2 Anschließen eines USBtinyISP	310
A.3.3.3 Anschließen eines Arduino Uno ArduinoISP	310
A.3.4 Programmierparameter	311
A.3.4.1 Dokumentation für avrdude	311
A.3.4.2 Erläuterung der Parameter für avrdude	311
A.3.4.3 Parameter für einen USBasp	314

A.3.4.4 Parameter für einen USBtinyISP	314
A.3.4.5 Parameter für einen ArduinoISP.	314
A.3.5 Microchip Studio Externes Tool einrichten	314
A.3.5.1 Öffnen eines Assembler- oder C-Projekts	315
A.3.5.2 Hinzufügen eines externen Tools in Microchip Studio	315
A.3.5.3 Hinzufügen einer Symbolleistenschaltfläche für das externe Werkzeug.	317
A.3.6 Testen des Programmers	319
A.3.7 Programmierprobleme und Lösungen	319
Anhang B • Alternative Schaltungen und Programme.	321
B.2 14-Pin PDIP ATtiny24/44/84	322
B.3 20-Pin-PDIP ATtiny26/261/461/861 und ATtiny2313/4313	323
B.4 28-Pin PDIP ATtiny48/88	324
B.5 Alternative Programme	325
B.5.1 Alternatives LED-Blink-Assemblerprogramm	325
B.5.2 Alternatives LED-Blink-C-Programm	326
Anhang C • Die ASCII-Tabelle	327
C.1 Druckbare Zeichen	327
C.2 ASCII-Tabelle	327
Index	328