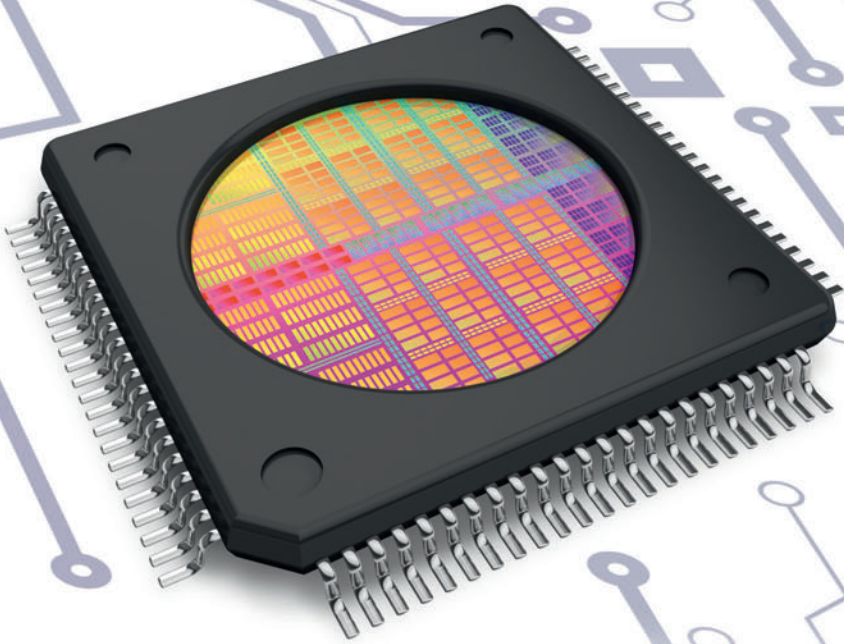


Irmtraut Meister / Lukas Salzburger



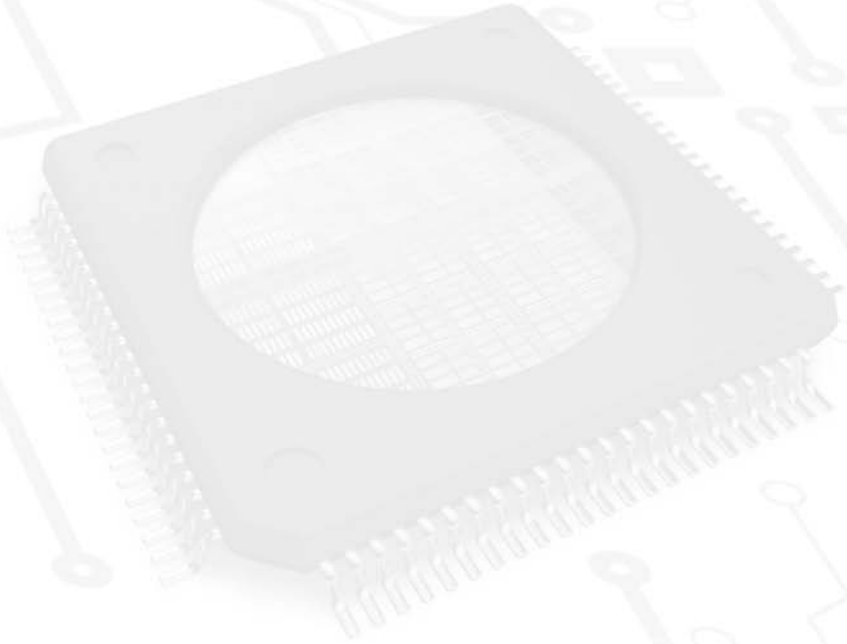
AVR-

Mikrocontroller-Kochbuch

Entwurf und Programmierung praktischer Anwendungen

Irmtraut Meister / Lukas Salzburger
AVR-Mikrocontroller-Kochbuch

Irmtraut Meister / Lukas Salzburger



AVR-

Mikrocontroller-Kochbuch

Entwurf und Programmierung praktischer Anwendungen

Bibliografische Information der Deutschen Bibliothek

Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte Daten sind im Internet über <http://dnb.ddb.de> abrufbar.

Hinweis: Alle Angaben in diesem Buch wurden von den Autoren mit größter Sorgfalt erarbeitet bzw. zusammengestellt und unter Einschaltung wirksamer Kontrollmaßnahmen reproduziert. Trotzdem sind Fehler nicht ganz auszuschließen. Der Verlag und die Autoren sehen sich deshalb gezwungen, darauf hinzuweisen, dass sie weder eine Garantie noch die juristische Verantwortung oder irgendeine Haftung für Folgen, die auf fehlerhafte Angaben zurückgehen, übernehmen können. Für die Mitteilung etwaiger Fehler sind Verlag und Autoren jederzeit dankbar. Internetadressen oder Versionsnummern stellen den bei Redaktionsschluss verfügbaren Informationsstand dar. Verlag und Autoren übernehmen keinerlei Verantwortung oder Haftung für Veränderungen, die sich aus nicht von ihnen zu vertretenden Umständen ergeben. Evtl. beigefügte oder zum Download angebotene Dateien und Informationen dienen ausschließlich der nicht gewerblichen Nutzung. Eine gewerbliche Nutzung ist nur mit Zustimmung des Lizenzinhabers möglich.

© 2013 Franzis Verlag GmbH, 85540 Haar bei München

Alle Rechte vorbehalten, auch die der fotomechanischen Wiedergabe und der Speicherung in elektronischen Medien. Das Erstellen und Verbreiten von Kopien auf Papier, auf Datenträgern oder im Internet, insbesondere als PDF, ist nur mit ausdrücklicher Genehmigung des Verlags gestattet und wird widrigenfalls strafrechtlich verfolgt.

Die meisten Produktbezeichnungen von Hard- und Software sowie Firmennamen und Firmenlogos, die in diesem Werk genannt werden, sind in der Regel gleichzeitig auch eingetragene Warenzeichen und sollten als solche betrachtet werden. Der Verlag folgt bei den Produktbezeichnungen im Wesentlichen den Schreibweisen der Hersteller.

Satz: DTP-Satz A. Kugge, München

art & design: www.ideehoch2.de

Druck: C.H. Beck, Nördlingen

Printed in Germany

ISBN 978-3-645-65126-4

Vorwort und »Gebrauchsanweisung«

Ein gutes Kochbuch erkennt man an zwei Dingen: Zum einen verfügt es über einen allgemeinen Teil, der nützliche grundlegende Tipps und Anweisungen rund ums Kochen enthält, ohne deren Kenntnis das tollste Rezept nicht gelingt. Darunter fallen beispielsweise die korrekte Zubereitung der verschiedenen Fleischsorten, die Lagerung von Gemüse oder einige einfache Saucen. Zum anderen sollen die Rezepte selbst von einer Qualität sein, die es erlaubt, aus möglichst einfachen Zutaten ohne großen Aufwand ein möglichst schmackhaftes Ergebnis zu zaubern. Idealerweise gefolgt von möglichen Variationen, falls eine Zutat gerade nicht zur Hand ist.

Die Grundlagen sind zwar vielleicht für erfahrene Hobbyköche eher uninteressant. Aber wenn sie nicht vorhanden sind – und der eine oder andere Hinweis ist mit an Sicherheit grenzender Wahrscheinlichkeit für jemanden neu – so können neue Versuche an Kleinigkeiten scheitern und rasch macht sich Frustration breit.

Nach dieser Logik haben wir den ersten Teil des Buches gestaltet. Die Grundlagen sollen eine gemeinsame Basis schaffen und Einsteigern Konzepte vermitteln, auf denen alles Weitere aufbaut. Auf diese folgen einfache Anwendungen, in welchen die Umsetzung der theoretischen Konzepte an einfachen praktischen Beispielen demonstriert wird.

Den Einstieg bilden also *Mikrocontroller-Grundlegenden (Kapitel 1)*, gefolgt von Allgemeinem zur *Programmierung und Implementierung (Kapitel 2)*, von Bitoperationen bis zu einfachen Codegerüsten, die den Brückenschlag zwischen den theoretischen Grundlagen und der realen Umsetzung bilden sollen. Letzteres geht auf *Erste Schritte (Kapitel 2.5)* mit einem Mikrocontroller ein, den *Allgemeinen Programmaufbau (Kapitel 2.6)* sowie detailliert auf die allgemeine Implementierung der einzelnen Peripherieeinheiten und *Grundbausteine (Kapitel 2.7)*.

Schließlich kommen wir zu den eigentlichen »Rezepten« (Kapitel 3 bis 12), anhand derer wir unsere Mikrocontroller-»Suppen« nach Lust und Laune kochen können. Hier finden sich Beispiele zum Schalten von LEDs und anderen Lasten, zur Messung verschiedener Größen (Strom, Spannung, Kapazität, Temperatur, Frequenz, ...), zum Erzeugen von Signalen oder zur Kommunikation über diverse Schnittstellen – um einige zu nennen.

Wir haben versucht, eventuell auftretende Unklarheiten an Ort und Stelle etwa durch angeführte Beispiele im Text zu beseitigen. Für den Fall, dass trotzdem welche auftreten, bildet ein Anhang mit verschiedenen Hinweisen und Tabellen den Abschluss (*Kapitel 13*).

Die Autoren möchten sich bei all jenen bedanken, die bei der Entstehung dieses Buches mitgewirkt und es dadurch erst ermöglicht haben.

Abschließend bleibt uns nur noch, dem Leser viel Freude und Erfolg auf dem spannenden Gebiet der Mikrocontrollerprogrammierung zu wünschen, und wir hoffen, mit diesem Buch dazu beizutragen.

Sicherheitshinweis:

Unfälle mit Strom können schmerzhaft oder sogar tödlich sein, daher ist besondere Vorsicht geboten. Bestehende Normen und Sicherheitsrichtlinien zum Umgang mit dem elektrischen Strom sind unbedingt zu konsultieren und zu beachten, folgende Hinweise gelten nur als Richtlinie.

Niedrige Spannungen unter 25 V Wechselspannung beziehungsweise 60 V Gleichspannung gelten als ungefährlich. Unter außerordentlichen Bedingungen kann aber auch eine geringe Wechselspannung unter Anderem den Herzrhythmus stören. Vor allem Mischspannungen (Gleichstrom mit Wechselstromanteil) sind gefährlich.

Zudem gelten viele in der Elektronik eingesetzte Chemikalien als gesundheitsschädigend, insbesondere Blei und Kadmium in Lötzinn. Daher sind diesbezüglich ebenfalls Vorsichtsmaßnahmen zu treffen.

Atmel[®], Atmel logo and combinations thereof, AVR[®], PicoPower[®] and others are registered trademarks or trademarks of Atmel Corporation or its subsidiaries.

Einführung

Mikrocontroller können als kleine Computer verstanden werden, die im Gegensatz zu ihren großen Verwandten gezielt für spezielle, mehr oder weniger komplexe Aufgaben eingesetzt werden. Sie bilden den Kern von vielen *Eingebetteten Systemen (Embedded Systems)*, welche für die Funktion der meisten modernen elektronischen und mechanischen Systeme verantwortlich sind, die inzwischen so selbstverständlich zum alltäglichen Leben gehören.

Die möglichen Einsatzszenarien sind so vielseitig, dass es womöglich einfacher wäre, jene modernen technischen Geräte herauszupicken, in denen ausnahmsweise *kein* Mikrocontroller steckt.¹ Von MP3-Playern und Handys über Haushaltsgeräte und Kraftfahrzeugkomponenten bis hin zu Kraftwerken und Industrie-Großanlagen – Mikrocontroller übernehmen eine kaum überschaubare Vielzahl unterschiedlichster Aufgaben. Allein in einem modernen Personenkraftwagen können über 100 Mikrocontroller verbaut sein.

Ein Mikrocontroller ist insofern mit einem *Personal Computer* vergleichbar, als er aus ähnlichen Komponenten aufgebaut ist: Ein Prozessor bildet sein Herzstück, es gibt einen Arbeits- und einen dauerhaften Programmspeicher sowie Zusatzperipherie wie beispielsweise Ein- und Ausgabeeinheiten. Man spricht allerdings erst von einem Mikrocontroller, wenn all diese Bausteine auf einem einzelnen Chip integriert sind.² Im Laufe des folgenden Abschnitts *Grundlegende Konzepte* wird genauer auf die Bestandteile und Charakteristiken eingegangen, die für das Verständnis der Funktionsweise, der Möglichkeiten und Grenzen eines Mikrocontrollers essenziell sind.

Zur Erläuterung wurden *AVR[®]-Mikrocontroller* von *Atmel[®]* gewählt, da sie weit verbreitet und leicht erhältlich sind. Auch ist der Umgang mit ihnen einfach zu erlernen, da sie einen eleganten und vergleichsweise einfachen Aufbau haben. Die Codebeispiele wurden in der Programmiersprache C geschrieben, um dem Industriestandard gerecht zu werden. Die Grundlagen und Beispiele gelten aber auch für Mikrocontroller anderer Hersteller beziehungsweise sind, mit geringen Abweichungen, auf jene übertragbar. Gerade dieses erste Kapitel ist so ausgelegt, dass es den Leser befähigen soll, sich rasch

¹ Neben Mikrocontrollern gibt es beispielsweise noch ASICs, FPGAs und DSPs, welche eine vergleichbare beziehungsweise spezialisiere Rolle übernehmen können und je nach Anwendung vorzuziehen sind.

² Hier gibt es auch Ausnahmen, etwa ältere oder besonders leistungsstarke Mikrocontroller, bei welchen der Speicher auch extern angeschlossen werden kann. Die genannte Definition ist auch nur eine der möglichen.

auch in »fremde« Mikrocontroller einzuarbeiten sowie deren Besonderheiten zu erkennen.

Falls der Leser bereits einen Blick in einschlägige Internetforen geworfen hat, werden ihm womöglich die beinahe religiös anmutenden Diskussionen rund um die Frage auffallen: »Welche Mikrocontroller sind die besten?« Generell kann man diesen Punkt – wie bei den meisten ähnlich lautenden Fragen – nur beantworten mit: »Es kommt darauf an«.

Mikrocontroller werden je nach Anwendung bezüglich Architektur, Speicher und Zusatzfunktionen sowie Kosten, Leistungsverbrauch und Verfügbarkeit ausgewählt (nach Abschluss des Grundlagenkapitels wird das verständlicher). Üblicherweise hat jeder große Hersteller eine Vielzahl vergleichbarer Lösungen parat, wodurch es letztendlich nicht selten auf persönliche Präferenzen hinausläuft, ob man nun einen Mikrocontroller von *Atmel*[®], *Freescale*[™], *Microchip*, *NXP*, *Renesas*, *Texas Instruments* oder einem anderen »Global Player« wählt.

Dieses Buch setzt Basiskenntnisse in der Programmiersprache C voraus. Zum Erlernen sei auf entsprechende Fachbücher und Online-Tutorials verwiesen. Die Grundlagen werden jedoch auch in einem eigenen Abschnitt kurz wiederholt und um mikrocontrollerspezifische Details erweitert.

Hinweis:

Wir haben viele generell für Mikrocontrolleranwendungen wichtige Maßnahmen an der Stelle beschrieben, an der sie das erste Mal auftauchen beziehungsweise von elementarer Bedeutung sind. Das bedeutet nicht, dass sie nicht auch für andere Anwendungen zu beachten sind – wir benutzen daher viele Querverweise und haben uns bemüht, ein möglichst umfangreiches Stichwortverzeichnis zu pflegen, damit jeder Hinweis und jede Erklärung auffindbar sind.

Inhaltsverzeichnis

1	Mikrocontroller-Grundlagen	15
1.1	Was ist ein Mikrocontroller?	15
1.2	Grundlegende Konzepte.....	18
1.2.1	Die Prozessorarchitektur.....	18
1.2.2	Gehäuse (Package).....	20
1.2.3	Datenblätter, Manuals und Errata	23
1.2.4	Versorgungsspannung und Signalpegel	26
1.2.5	Speicher.....	29
1.2.6	Takt und Taktgenerierung.....	33
1.2.7	Interrupts.....	38
1.2.8	Timer (Zähler).....	40
1.2.9	Register	42
2	Programmierung und Implementierung	45
2.1	Allgemeines zur Programmierung.....	45
2.1.1	Eigenheiten der Mikrocontrollerprogrammierung	47
2.1.2	Schlüsselwörter.....	47
2.1.3	Portierbarkeit von Code.....	49
2.1.4	Codeoptimierung.....	51
2.1.5	Compilereinstellungen.....	52
2.2	Programmierung des Mikrocontrollers.....	53
2.2.1	Programmierungsumgebung.....	54
2.2.2	Programmieradapter.....	54
2.2.3	ISP	55
2.2.4	Fuses.....	57
2.2.5	Bootloader.....	59
2.3	Debugging	60
2.3.1	Printf-Debugging	60
2.3.2	Software-Emulator	61
2.3.3	JTAG und DebugWIRE	62
2.3.4	Hardware-Debugging	64
2.4	Bitoperationen	65
2.4.1	Bitoperatoren.....	65
2.4.2	Bits setzen, löschen und toggeln	68

2.4.3	Bits prüfen	70
2.5	Erste Schritte – ein einführendes Programm	72
2.5.1	Schaltungsaufbau	72
2.5.2	Programmcode.....	76
2.5.3	Programmierung des Controllers.....	78
2.6	Allgemeiner Programmaufbau	79
2.6.1	Außerhalb der Hauptroutine.....	80
2.6.2	Hauptroutine.....	83
2.7	Grundbausteine – Funktionsweise und Implementierung	84
2.7.1	Interrupts.....	84
2.7.2	Timer.....	93
2.7.3	Delay.....	98
2.7.4	IO-Pins (GPIOs).....	99
2.7.5	AD-Wandler.....	100
2.7.6	DA-Wandler.....	113
2.7.7	Komparator.....	114
2.7.8	PWM.....	117
2.7.9	UART/USART	121
2.7.10	SPI / Microwire.....	135
2.7.11	I ² C / TWI / 2-Wire	141
2.7.12	CAN	149
2.7.13	USB.....	149
2.7.14	Zustandsautomat (State Machine)	150
2.7.15	Watchdog	154
2.7.16	Brownout-Detektor	158
2.7.17	Low Power und Schlafzustände	159
2.7.18	General Purpose I/O Register und Fehlerbehandlung.....	165
3	Digitale Ein- und Ausgänge	167
3.1	Pegelwandler	167
3.2	Pinerweiterung mit I/O-Bausteinen	168
3.2.1	SPI-Schieberegister: die 74xx595-Familie.....	169
3.3	Schalten großer Lasten	172
3.3.1	Schalten mit MOSFETs	174
3.3.2	Schalten mit Bipolartransistoren	178
3.3.3	Ausgangstreiber	180
3.3.4	Relais	181
3.3.5	Wechselspannungen schalten.....	182
3.4	Schutzschaltungen	190

4	Spannungsmessung.....	195
4.1	Anpassung des Eingangsspannungsbereichs	195
4.2	AD-Wandlungsergebnis berechnen	197
4.3	Referenzspannung.....	198
4.3.1	Interne Referenzspannungsquelle.....	198
4.3.2	Externe Referenzspannungsquelle	199
4.4	Interner AD-Wandler	200
4.4.1	Konfiguration	200
4.5	Externer AD-Wandler.....	203
4.5.1	ADC mit I ² C-Schnittstelle	204
4.5.2	ADC mit SPI-Schnittstelle	204
4.5.3	Parallel angeschlossene ADCs.....	206
4.6	Verifizieren der Messung.....	207
4.6.1	Referenz- und Versorgungsspannung.....	207
4.6.2	Das Histogramm	208
4.6.3	Für Fortgeschrittene: Zufall und Korrelation	209
4.7	Messen von Wechselspannungen.....	210
4.7.1	Parameter einer Wechselspannung.....	210
4.7.2	Effektivwertmessung (RMS).....	212
4.7.3	Spitzenwertmessung	212
5	Spannungen ausgeben.....	215
5.1	Interner DA-Wandler	216
5.2	Externer DA-Wandler.....	216
5.3	Analogspannung mit PWM generieren.....	217
5.3.1	Analogspannung mit Fast-PWM	218
5.3.2	Für Fortgeschrittene: Filterauslegung.....	219
5.4	Software-PWM.....	222
5.4.1	Software-PWM mit Compare Match	222
5.4.2	Software-PWM mit Timer Overflow	223
6	Widerstandsmessung.....	227
6.1	Spannungsteiler.....	227
6.2	Messung mit Konstantstrom.....	228
6.3	Ratiometrische Messung.....	230
7	Strommessung	233
7.1	Messung mit Shuntwiderstand	233
7.1.1	Current Shunt Monitor	234
7.2	Transimpedanzverstärker.....	235

7.3	Hallsensor	237
7.4	Stromwandler.....	237
7.5	Strommessung mit einem Kondensator.....	237
8	Zeit- und Frequenzmessung.....	239
8.1	Periodendauermessung.....	239
8.1.1	Beispiel Messung der Periodendauer mit Timer und analogem Komparator.....	240
8.2	Zählen der Nulldurchgänge	243
9	Kapazitäts- und Induktivitätsmessung	247
9.1	Ladekurve.....	247
9.2	Schwingkreis.....	249
9.3	RC-Oszillator	250
10	Temperaturmessung	253
10.1	Widerstandstemperatursensoren.....	253
10.1.1	Ptxxx (Pt100, Pt1000, ...).	254
10.1.2	KTY-Serie	256
10.1.3	PTC.....	257
10.1.4	NTC	257
10.2	Halbleitertemperatursensoren.....	257
10.2.1	AVR®-interner Temperatursensor.....	258
10.2.2	Externe Temperatursensoren mit Spannungs-/Stromausgang	260
10.3	Thermoelement	262
10.4	Digitale Temperatursensoren.....	264
10.4.1	Beispiel LM75-kompatibler I ² C-Temperatursensor.....	265
11	Kommunikation mit Menschen.....	271
11.1	Eigenes »printf()«	271
11.2	LEDs und 7-Segment-Anzeigen	272
11.2.1	LEDs	272
11.2.2	7-Segment-Anzeige	273
11.2.3	RGB-LED mit PWM.....	278
11.3	Taster und Keypads	281
11.3.1	Matrixtastatur	281
12	Daten speichern.....	287
12.1	Internes EEPROM.....	287
12.2	Interner Flash-Speicher	289
12.2.1	Lookup-Tabelle im Flash	290

12.3	Externe Speicher	291
12.3.1	SPI-Flash.....	291
13	Anhang	293
13.1	Elektrotechnische Grundgleichungen	293
13.1.1	Das Ohm'sche Gesetz	293
13.1.2	Serien- und Parallelschaltung R, C, L	294
13.1.3	Spannungsteiler	294
13.1.4	Grenz- und Resonanzfrequenz	295
13.1.5	Bandbreite eines Rechtecksignals	296
13.2	Darstellung von Bauteilwerten	298
13.3	E-Reihe	299
13.4	Temperaturbereiche.....	299
13.5	LED-Vorwiderstand	300
13.5.1	Berechnung	301
13.6	Dezibel (dB)	303
13.6.1	Signal-Rausch-Verhältnis SNR	304
13.7	Kalibrierung	305
13.7.1	Kalibrieren, Justieren und Eichen	305
13.7.2	Grundprinzip der Kalibrierung.....	306
13.7.3	Ein-Punkt-Kalibrierung	308
13.7.4	Zwei-Punkt-Kalibrierung.....	310
13.8	Linearisierung	311
13.8.1	Vorgehensweise	311
13.9	Lookup-Tabellen.....	315
13.10	Steckbrett, Loch- und Streifenrasterplatten.....	315
13.11	Dualsystem	321
13.12	Zweierkomplement.....	322
	Stichwortverzeichnis.....	325

1 Mikrocontroller-Grundlagen

1.1 Was ist ein Mikrocontroller?

Man spricht von einem Mikrocontroller (μC , MicroController Unit MCU), wenn außer dem *Mikroprozessor* selbst noch Peripherieeinheiten in einen Chip integriert sind. Dieses erste Kapitel dient dem genaueren Verständnis der grundlegenden Funktionen, Möglichkeiten und Grenzen eines Mikrocontrollers. An dieser Stelle wird vieles noch bewusst vereinfacht dargestellt, auf das in späteren Kapiteln noch genauer eingegangen werden soll.

Das Herzstück einer jeden MCU ist der beinhaltete *Prozessor* oder *Kern (Core)*, also die für die Ausführung der Programme zuständige Recheneinheit. Je nach Anzahl der auf einmal verarbeitbaren Bits spricht man von 4-, 8-, 16- oder 32-Bit-Mikrocontrollern. Ein *Bit (Binary Digit)* ist die kleinste Informationseinheit in digitalen Systemen, mit der ein Prozessor auf unterster Ebene rechnet. Es kann entweder den Wert *null* oder *eins* haben³. Jeder Befehl eines Programms, jede Form von gespeicherten Daten – also alles, was eine digitale Recheneinheit verarbeiten soll – wird letztendlich in mehr oder weniger komplexe Bitmuster aus Einsen und Nullen umgewandelt, die der Prozessor dann der Reihe nach bewältigen kann.

Zum Betrieb benötigt ein Prozessor außerdem einen *Takt*, also ein Signal bestehend aus periodischen Taktimpulsen (abwechselnd 1 und 0) in einer gewissen Geschwindigkeit. Der Takt kann entweder intern generiert oder extern zugeführt werden. Angegeben wird er in *Hertz* ($1 \text{ Hz} = 1/\text{s}$), also in Taktimpulsen pro Sekunde.

Da wir aber nur selten mit so niedrigen Frequenzen zu tun haben, findet man eher Angaben in kHz (Kilo-Hertz, also 1.000 Hertz) oder MHz (Mega-Hertz, also 1.000.000 Hertz). Signalen im GHz-Bereich (Giga-Hertz, 1.000.000.000 Hertz) wiederum begegnet man nur in Ausnahmefällen.

³ Wie »null« oder »eins« physikalisch aussehen, dazu gibt es eine Vielzahl möglicher Umsetzungen. Auf einer Signalleitung wird bei »eins« ein festgelegter Spannungspegel über- bzw. bei »null« unterschritten. Sollen Daten gespeichert werden, so hängt es von der Technologie des Speichers ab, auf welche Art jede seiner Speicherzellen den Wert »null« oder »eins« annimmt.

Hinweis:

Im Zusammenhang mit Datenübertragung werden Taktfrequenzen eher in *Baud* (1 Bd = 1/s) angegeben. Ein Baud entspricht also einem Hertz und wird auch gleichermaßen skaliert (1 kBd = 1.000 Bd etc.).

Bei der Angabe von Übertragungsgeschwindigkeiten kann man auch von einer *Baudrate* anstatt einer Taktrate sprechen.

Mehr Hertz bedeuten also mehr Taktimpulse pro Sekunde und daher einen schnelleren Takt und eine größere *Ausführungsgeschwindigkeit*, also mehr Rechenoperationen pro Sekunde.

Die zuvor in Verbindung mit dem Prozessor erwähnte Bit-Zahl sagt nun aus, wieviele Bits ein Prozessor mit einem Taktschritt verarbeiten kann. Arbeitet beispielsweise ein 8-Bit-Mikrocontroller mit 20 MHz (Mega-Hertz), so kann er 20 Millionen Mal pro Sekunde jeweils 8 Bits verarbeiten.⁴ Der AVR[®] ist ein Vertreter der Gattung der 8-Bit-Prozessoren.

Prozessoren, die mehr Bits gleichzeitig verarbeiten können (etwa 32 statt 8), sind also leistungsfähiger. Die Ausführungsgeschwindigkeit des Programms steigt außerdem mit dem Takt, mit dem der Prozessor betrieben wird. Ein 32-Bit-Prozessor ist aber bei gleichem Takt nicht automatisch viermal so schnell wie ein 8-Bit-Prozessor, da viele Faktoren eine Rolle spielen.

Bei der Angabe von Geschwindigkeiten und Stromverbrauch gilt bei vielen Mikrocontrollern die alte Steigerungsregel: Notlüge, Lüge, Benchmark. Nicht dass wir den Herstellern unterstellen würden, gefälschte Werte zu veröffentlichen, es wird aber so stark geschönt und für das eigene Produkt vorteilhaft gemessen, dass ein konkreter Vergleich zwischen den unterschiedlichen Architekturen und Herstellern nur sehr schwer möglich ist.

Aus Kosten- und Komplexitätsgründen wird tendenziell für eine gewisse Anwendung jener Mikrocontroller ausgewählt werden, der die Mindestanforderungen an Prozessorleistung, Speicher und Peripherie gerade noch erfüllt. Warum also einen 32-Bit-Prozessor benutzen, wenn auch ein 8-Bit-Prozessor ausreicht? 8- und 32-Bit-Prozessoren sind inzwischen gebräuchlicher als jene mit 4 oder 16 Bit. Für die meisten einfacheren Anwendungen genügen die tendenziell stromsparenderen 8-Bit-Prozessoren, und falls

⁴ Dies ist eigentlich eine grobe Vereinfachung, mit der wir uns aber für den Anfang zufrieden geben. Der extern zugeführte Takt muss zudem nicht mit dem internen Takt übereinstimmen, mit dem der Prozessor tatsächlich arbeitet. Darüber hinaus wird nicht jeder Befehl in genau einem Taktzyklus ausgeführt, manche können etwa bis zu vier Taktzyklen benötigen.

nicht, springt man meistens gleich zu 32-Bit-Prozessoren, weil der Fertigungsaufwand und damit auch der Preis für 16-Bit-Prozessoren kaum geringer ist.

Abgesehen von wenigen Ausnahmen gibt es außerdem keine 8-Bit-Mikrocontroller, die mit mehr als 50 MHz getaktet werden. Der Grund liegt darin, dass es ab einer gewissen Taktrate einfach effizienter ist, einen 32-Bit-Mikrocontroller einzusetzen. Es sei aber auch erwähnt, dass es Mischformen gibt, also z. B. 8-Bit-Controller, die teilweise 16 Bit parallel verarbeiten können.

Ergänzend zum Prozessor enthält ein moderner Mikrocontroller auch zwei verschiedene Speicher: den *Arbeitsspeicher* (*Random Access Memory, RAM*) und den *Programmspeicher*. Sie unterscheiden sich dadurch, dass der Arbeitsspeicher die darin abgelegte Information bei Unterbrechung der Stromversorgung verliert (man sagt: »er ist *flüchtig*«), während der Programmspeicher sie behält (»er ist *nicht-flüchtig*« oder »*statisch*«). An dieser Stelle ist die Frage berechtigt, wieso nicht ausschließlich nicht-flüchtiger Speicher benützt wird – wozu soll ein Speicher gut sein, der beim »Ausschalten« des Mikrocontrollers seine Daten verliert?

Zur Beantwortung müssen wir etwas ausholen: Man spricht vom *Programmieren* des Mikrocontrollers, wenn ein zuvor erstelltes *Programm* (der Code dafür, was der Mikrocontroller »machen soll«) in den nicht-flüchtigen Programmspeicher geschrieben wird. Dieser kann allerdings technologiebedingt im Laufe seiner Lebenszeit lediglich rund 10.000 bis 100.000 Mal beschrieben werden. Deshalb werden sich häufig ändernde Daten im flüchtigen Arbeitsspeicher zwischengelagert, der (nahezu) unbegrenzt viele Schreibzyklen erlaubt. Hinzu kommt bei schnelleren, komplexeren Mikrocontrollern, dass die höhere Zugriffsgeschwindigkeit auf den Arbeitsspeicher zur Leistungsfähigkeit des Gesamtsystems beiträgt.

Außer Prozessor, Arbeits- und Programmspeichern beinhaltet ein Mikrocontroller eine oder mehrere Zusatzeinheiten und damit -funktionalitäten. Dazu zählen beispielsweise *digitale* und/oder *analoge Ein- und Ausgänge*, *Timer* und diverse Schnittstellen – dies wird später noch im Detail behandelt.

Vorerst genügt es, zu wissen, dass es diverse Zusatzfunktionen gibt, welche es ermöglichen, mit einem einzelnen Mikrocontroller bereits viele spezialisierte Aufgaben zu lösen, ohne dass eine aufwendige Zusatzbeschaltung oder zusätzliche Chips nötig sind.

Wir fassen zusammen:

Ein Mikrocontroller (μC , MCU) vereint einen Mikroprozessor und einige Peripherie auf einem Chip. Mikrocontroller unterscheiden sich durch:

- die Anzahl der Bits, die ein Controller in einem Taktschritt bearbeiten kann. Es gibt 4-, 8-, 16- und 32-Bit- μC .
- die Größe des internen Arbeits- und Programmspeichers
- die integrierten Zusatzbausteine und -funktionen, welche für die Anforderung benötigt werden; z. B. digitale und analoge Ein- und Ausgänge, Timer, Schnittstellen etc.

Bei der konkreten Auswahl eines Mikrocontrollers müssen diese Eigenschaften und zusätzliche wichtige Punkte wie Bauform, Betriebsspannungsbereich, Lieferbarkeit und natürlich der Preis berücksichtigt werden.

Nachdem wir nun wissen, was einen Mikrocontroller ausmacht, können wir im nächsten Abschnitt genauer ins Detail gehen.

1.2 Grundlegende Konzepte

1.2.1 Die Prozessorarchitektur

Generell unterscheidet man bei Prozessoren zwei verschiedene Architekturen. Die Prozessoren der meisten aktuellen Computer (z. B. der x86-Familie) basieren auf der *Von-Neumann-Architektur*. Sie besteht aus einer *CPU (Central Processing Unit)*, welche den Prozessor und das Steuerwerk umfasst, der *I/O Unit* (für *Input/Output*, also für die Kommunikation mit der Außenwelt) und dem Speicher. CPU und I/O Unit bzw. Speicher kommunizieren über ein gemeinsames Bussystem, über das sowohl Daten als auch Befehle ausgetauscht werden. Programme können problemlos auch im Datenspeicher ausgeführt werden.

Die *Harvard-Architektur* hingegen, welche für eine Vielzahl von Mikrocontrollern einschließlich der AVR[®] typisch ist, besitzt zwei getrennte Bussysteme sowie getrennten Speicher für Daten und Programme. Das hat den Vorteil, dass simultan Befehle ausgeführt und Daten ausgelesen oder geschrieben werden können. Der Datenspeicher kann allerdings auch nicht einfach als Programmspeicher genutzt werden.⁵

⁵ Zwischen diesen beiden Architekturen kann in der Praxis oft nicht so einfach unterschieden werden, da es inzwischen eine Vielzahl von Mischformen gibt.

Unter dem Familiennamen AVR[®] fasst Hersteller Atmel[®] mehrere Mikrocontroller-typen mit 8-, 8/16- oder 32-Bit-Harvard-Architektur zusammen. Wir werden uns im Rahmen dieses Buchs auf die »klassischen« 8-Bit-AVRs beschränken⁶. Dabei sind vor allem zwei Serien für unsere Zwecke interessant:

Die ATtiny-Serie umfasst die günstigeren und teilweise sehr stromsparenden oder mit Sonderfunktionen ausgestatteten Modelle. Sie tragen die Bezeichnung *ATtiny[x][y]* mit der Zahl [x] und eventuell einem Buchstaben [y], welche den konkreten Mikrocontroller genauer spezifizieren (Variante, Leistungsaufnahme, Spezialfunktionen). Beispiele sind der winzige *ATtiny10* oder der *ATtiny261A*.

Die Mikrocontroller der ATmega-Serie verfügen über deutlich mehr Peripherieeinheiten und Speicher, ihr Kern kennt einige Befehle mehr und besitzt eine Hardwaremultiplikationseinheit. Die Bezeichnung folgt demselben Schema, also *ATmega[x][y]* mit der Zahl [x] und dem/den Buchstabe/n [y] gemäß seiner Spezifikation. Ein Vertreter ist der *ATmega48*, der auch im Kapitel 2.5 *Erste Schritte – ein einführendes Programm* verwendet wird.

Die Bezeichnung AT90 trugen die ersten Controller mit dem AVR-Kern. Heute wird dieser Name jedoch für »Spezialtypen« mit z. B. USB, CAN oder speziellen PWM-Einheiten verwendet.

Das Namensschema ist ziemlich kompliziert und auch nicht wirklich durchgängig. Wenn man einen geeigneten Mikrocontroller für den jeweiligen Einsatzzweck sucht, verwendet man am besten die – leider nicht besonders durchdachte – parametrische Suche auf der Atmel[®]-Webseite.

Im Hobbybereich, aber durchaus auch im professionellen Umfeld, sind der ATmega8 und seine »Nachfolger« ATmega48, -88, -168 und -328 in allen Varianten beliebt, vor allem da sie im 28Pin-DIP-Gehäuse erhältlich und recht preiswert sind. Ein weiterer Vorteil ist die Kompatibilität der Baureihe, deren Modelle sich nur im Speicherausbau (und einigen kleinen Details) voneinander unterscheiden. Man kann daher die Entwicklung mit der größten Variante beginnen und das Endprodukt dann nach Möglichkeit mit einer Version mit weniger Speicher (also billiger) ausliefern, ohne die Soft- oder andere Hardware zu ändern.

⁶ Außerdem gibt es noch die 32-Bit-AVR-UC3-Familie, welche mit den hier behandelten, abgesehen vom Namen, nicht viel gemeinsam hat, und die 8/16-Bit-AVR-XMEGA-Reihe.

Hinweis

Es gibt laufend neue Typen und Modelle in den AVR[®]-Serien. So ist etwa der *ATmega48A* ein Nachfolger des *ATmega48*, der mit einem neuen Herstellungsprozess gefertigt wurde und zusätzlich zu einigen Detailverbesserungen ab 1,8 V statt 2,7 V betrieben werden kann. Die Varianten *ATmega48P* beziehungsweise *ATmega48PA* sind die zugehörigen stromsparenden Varianten mit einigen Zusatzfunktionen für diesen Einsatzbereich.

Controller wie der *ATmega328* allerdings haben keinen unmittelbaren Vorgänger. Obwohl kein »A« angehängt ist, wird er nach dem gleichen, neuen Herstellungsprozess gefertigt und läuft ab 1,8 V. Man sieht, die Mikrocontrollerbezeichnungen sind eine Wissenschaft für sich, daher sollten immer Hersteller-Homepages und Datenblätter konsultiert werden.

1.2.2 Gehäuse (Package)

Wie die meisten massengefertigten ICs⁷ sind AVR[®]-Mikrocontroller in verschiedenen *Packages*, den Gehäusetypen, erhältlich. Der eigentliche Controller nimmt als *Die* (engl.: Chip, Halbleiterplättchen) nur sehr wenig Platz im finalen Gehäuse ein. Dieses übernimmt gewissermaßen die Aufgabe, den Halbleiterchip gegen äußere Einflüsse zu schützen und den Einbau in eine Schaltung zu ermöglichen.

Um die Funktionen des Chips zugänglich zu machen, muss er *gebondet* werden (eingedeutschter Begriff, von engl. *Bonding*: Binden, Verbindung), also mittels dünner Drähte eine Verbindung zwischen den Kontaktflächen des Halbleiterchips und den Außenkontakten des Gehäuses – den *Pins* – hergestellt werden.⁸

Der Anfänger wird wahrscheinlich als erstes mit AVRs im DIP-Package zu tun haben (*Abb. 1.1*). DIP steht für *Dual In-line Package* und bezeichnet, wie der Name sagt, einen länglichen Gehäusotyp mit paarweisen Pins an jeder Seite.

⁷ IC steht für *Integrated Circuit*, bezeichnet also eine in einen einzelnen Baustein integrierte Schaltung.

⁸ Mit »Pin« ist also meistens der »physikalische«, äußere Kontakt eines Controllers gemeint. Es muss aber nicht immer jeder Pin am Gehäuse eine elektrische Funktion haben. Es gibt auch nicht verbundene (not connected; NC) Pins, die z. B. nur eine mechanische Funktion haben.

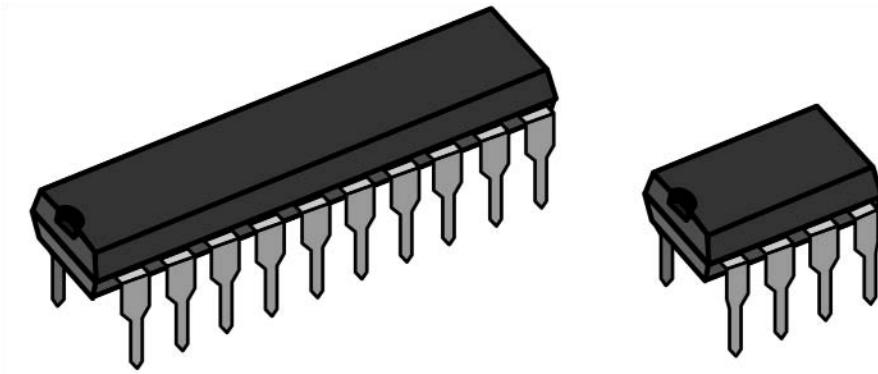


Abb. 1.1: 20- und 8-Pin-DIP-Gehäuse

Das ist der »größte« und »bastlerfreundlichste« Gehäusertyp, die Pins sind verhältnismäßig große, nach unten abgeknickte Kontaktstifte, mit denen der Controller etwa in Steckbretter oder Lochrasterplatinen passt. Bauelemente, die zum Schaltungsaufbau »in Löcher gesteckt« werden können, werden auch als »Through-Hole«-Komponenten bezeichnet. Daher (und weil sie ihrer Größe wegen einfacher zu handhaben und schwerer zu verlieren sind) eignen sich DIP-Gehäuse gut für erste Schaltungsexperimente und Prototypen.

Bausteine im DIP-Gehäuse sind zudem sockelbar, das heißt anstatt des Bausteins selbst kann auch ein Sockel eingelötet werden (*Abb. 1.2*). Der Baustein, beispielsweise der Mikrocontroller, kann daraufhin in den Sockel eingesetzt und wie gewohnt genutzt werden. Wird er beschädigt oder soll er aus einem anderen Grund ausgetauscht werden, so kann er einfach wieder aus dem Sockel gelöst werden, ohne dass aufwendiges und materialbeanspruchendes Lötens notwendig ist.

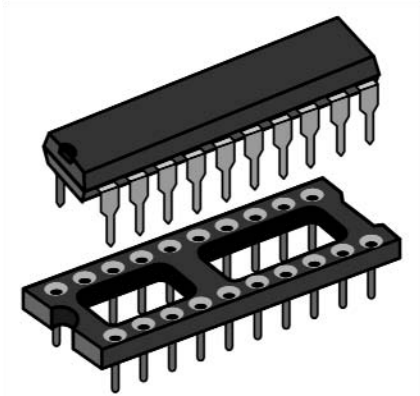


Abb. 1.2: Sockel für DIP-Bauteile

Professionelle Schaltungen beinhalten heutzutage kaum mehr Bauelemente in DIP-Gehäusen. Anders als bei Through-Hole-Komponenten sind bei der Oberflächenmontage (SMT, *Surface Mount Technology*) keine Bohrungen nötig und die Schaltung kann wesentlich kompakter ausgeführt werden. Die hierbei verwendeten Bauteile werden als SMD (*Surface Mounted Device*) bezeichnet. Ihre Anschlusspins sind so gefertigt, dass sie plan auf dafür vorgesehenen *Pads* auf den Platinen zu liegen kommen und an diesen angelötet werden können. Es gibt mehrere Bauformen von SMD-Bauteilen in unterschiedlichen Größen. SO-Gehäuse (*Small Outline*) sind die »größten« SMD-Gehäuse und noch sehr gut per Hand lötbar (Abb. 1.3). Ihr Pinabstand ist halb so groß wie bei den DIP-Gehäusen.

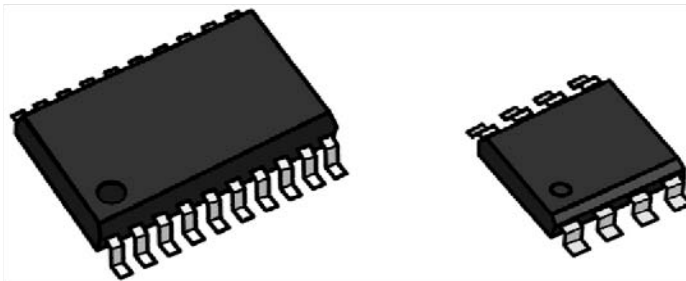


Abb. 1.3: 20- und 8-Pin-SO-Gehäuse

Neben dem geringeren Platzbedarf haben SMD-Komponenten auch elektrisch bessere Eigenschaften als ihre großen Verwandten. So sind beispielsweise Hochfrequenzschaltungen in der Regel mit DIP-Bauteilen undenkbar. Wenn es die Anforderungen hingegen erlauben, spricht nichts dagegen, Entwicklung und Programmierung zunächst mit einem

Controller in DIP-Bauform vorzunehmen und später, falls die Schaltung in Massenfertigung hergestellt werden soll, denselben Controller in einem SO-Package oder einem noch kleineren SMD-Gehäuse vorzusehen,⁹ etwa in einem QFP-Gehäuse (*Quad Flat Package*, Abb. 1.4a). Bei den AVR[®]-Mikrocontrollern verbreitet ist die dünnere Bauform TQFP (*Thin Quad Flat Package*) mit 0,8 mm bis 0,5 mm Pinabstand. Benötigt man noch kleinere Gehäuse, greift man zum »beinchenlosen« QFN (*Quad-Flat No-leads package*) oder den fast nur mit professioneller Ausrüstung zu verarbeitenden BGA-Gehäusen (*Ball Grid Array*), wo die Anschlüsse auf der Unterseite positioniert sind (Abb. 1.4b).

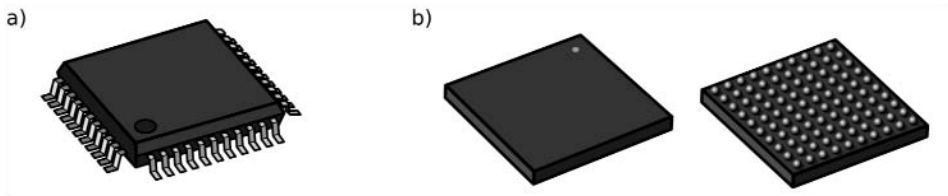


Abb. 1.4: QFP und BGA

Wir fassen zusammen:

Ein und derselbe Mikrocontroller kann in verschiedene Gehäuse eingebaut sein. Für einfache Prototypen sowie zu Experimentierzwecken eignet sich insbesondere das verhältnismäßig große DIP-Gehäuse, welches sich in Steckbretter und Lochrasterplatten einsetzen lässt.

Im professionellen Bereich hingegen kommen hauptsächlich SMD-Gehäuse zum Einsatz, welche kleiner sind und bessere elektrische Eigenschaften aufweisen. Sie werden plan auf der Platine aufgelötet, Bohrungen wie beim Einsatz von Through-Hole-Bauteilen wie DIP-Gehäusen sind nicht nötig.

1.2.3 Datenblätter, Manuals und Errata

Jeder Controller hat ein Datenblatt, in dem (idealerweise) alle seine relevanten technischen Daten aufgelistet werden. Wie von nahezu jedem elektronischen Bauteil findet man sie zum kostenlosen Download auf der Webseite des Herstellers, in diesem Fall also bei Atmel[®] (www.atmel.com/avr).

⁹ Natürlich darf deswegen nicht auf entsprechende Tests verzichtet werden – eine Schaltung kann sich auf dem Steckbrett signifikant anders verhalten als auf einer professionell gefertigten Platine!

Der Leser mag sich vielleicht wundern: Datenblätter, so »früh« im Buch? Tatsächlich ist ein gut geschriebenes und strukturiertes Datenblatt im Idealfall ein Crashkurs für Betrieb und Programmierung des jeweiligen Mikrocontrollers, bei dem kaum Fragen offen bleiben. Viele Informationen und Erklärungen aus diesem Buch ließen sich auch aus einem guten Datenblatt extrahieren. Die Voraussetzung dafür – und das kann nicht oft genug betont werden – ist natürlich, dass das Datenblatt auch gelesen wird. Wenn dem Entwickler zum ersten Mal ein Mitglied einer neuen Mikrocontrollerfamilie unterkommt, ist es sehr hilfreich, das Datenblatt zumindest einmal grob zu überblicken. Viele Fragen könnten geklärt, Fehler von vornherein vermieden und damit so einiger frustrierender Erfahrung vorgebeugt werden, wenn Datenblätter gewissenhafter konsultiert würden.

Sehen wir uns ein solches Datenblatt also genauer an: Auf der ersten Seite steht üblicherweise eine kurze Zusammenfassung aller wichtigen Eigenschaften wie Speichergrößen, Geschwindigkeit, Betriebsspannung, Leistungsaufnahme und eine Liste der Peripherieeinheiten.

Daneben sind alle Mikrocontroller-Typen aufgelistet, für die das Datenblatt gültig ist. Sehr viele Mikrocontroller unterscheiden sich nämlich nur im Speicherausbau und höchstens noch einigen Details und werden daher gemeinsam behandelt. Eine Software, die für einen Controller dieser Familie geschrieben wurde, kann meist ohne Änderungen am Code auf ein anderes »Familienmitglied« portiert werden (natürlich nur, wenn der Speicher ausreicht).

Von einigen Controllern gibt es auch besonders stromsparende Varianten, welche mit einer niedrigeren Versorgungsspannung auskommen. Diese werden im gleichen Datenblatt behandelt, wobei abweichende Werte für den jeweiligen Untertyp einzeln angegeben werden. Als Nächstes im Datenblatt folgt die Pinbelegung für alle erhältlichen Gehäusearten und eine Beschreibung aller Anschlusspins. Anschließend wird ein Überblick über die AVR-Architektur gegeben.

Praxistipp:

Meistens gibt es (abgesehen von Masse- und Versorgungsspannungsanschlüssen) pro Pin mehrere Möglichkeiten, wie dieser intern belegt werden kann. So könnte ein und derselbe »physikalische« Pin beispielsweise als allgemeiner Ein-/Ausgang (GPIO, General Purpose Input/Output), als ADC-Eingang oder als Datenleitung für eine Schnittstelle gewählt werden.¹⁰

¹⁰ Es liegt am Entwickler, die Belegung sinnvoll zu wählen, damit nicht plötzlich ein ADC benötigt wird, nachdem sämtliche in Frage kommenden Pins als GPIO eingesetzt wurden.

Stichwortverzeichnis

Symbole

#define 48, 50, 81

#include 80

16-Bit-Register 43, 87, 98

2-wire 87, 141, 168, 204, 266, 291

7-Segment-Anzeige 273

A

Abschlusswiderstand 130

Absolute Genauigkeit 105

Absolute Maximum Ratings 25, 26, 29,
167, 191

Abtastfrequenz 107

Abtastrate 107, 215

Abtasttheorem 107

Abtastung 100

ACK 145

Active HIGH 272

Active LOW 272

ADC 74, 100, 195, 197, 198, 200, 203,
230, 255, 260, 278, 305, 314

ADC Left Adjust 203, 279

ADC Noise Reduction 110, 162, 163, 202,
203

Aktives Filter 221

Aliasing 107, 212, 215

Amplitude 36, 106, 119, 208, 211, 213,
240, 296

Analog-Digital-Wandler 100, 195

AND 66

Anfangsgenauigkeit 198, 199, 258

Anode 77, 183, 278

Ansteuerspannung 181

Anstiegszeit 213, 297

antiparallel 173

Application Note 46

Arbeitsspeicher 17

Architektur 18

AREF 74, 198, 199, 208

Assembler 53

asynchron 38, 90, 97, 121, 128, 162, 163

AT90 19

ATmega 19

Atmel® Studio 45, 54, 61, 72, 76

atomar 43

ATtiny 19

Audio-ADC 203

Audio-DAC 216

Auflösung 93, 97, 99, 101, 102, 112, 120,
186, 195, 203, 216, 218, 223, 265, 305,
314

Ausgang 100

Ausgangstreiber 180

Ausgleichsstrom 173

Autotrigger 201, 202

AVCC 74, 111, 200, 208, 231

AVR Dragon 55, 72

AVR Libc 45, 86

AVR® Framework 45

AVR-Studio 54, 61

B

Bandabstand 113

Bandbreite 220, 296

Bandgap 113

Bandlücke 113

- Basis 178
- Batterie 164, 207
- Baud 16, 122
- Baudrate 16, 81, 122, 123, 124, 128, 160
- BAUDRATE 48
- Baudratenquarz 36, 124
- Benchmark 16
- Beschaltung 72
- BGA 23
- bidirektional 63, 122, 132, 135, 142, 149, 234
- Big Endian 136
- Binärsystem 321
- Bipolartransistor 168, 173, 176, 178, 181
- Bitmaske 69
- Bitoperation 65
- Blockierendes Warten 98
- Bonding 20
- Bootloader 59
- Brownout-Detektor 158
- Brückengleichrichter 192
- Bug 60
- Bussystem 18, 128, 149

- C**
- C standard library 47
- CAN 149
- Cast 288
- CKDIV8 34
- CKOUT 37
- CKSEL 34
- Clear Timer on Compare 120
- Clear Timer on Compare Match 97
- cli() 86
- CLKPR 34, 159
- CMOS 27, 161, 168, 191
- Codeoptimierung 51, 62, 89, 99
- Codierung 322
- Cold-Junktion 262
- Common Anode 277
- Common Cathode 277
- Common Mode 263
- Common Mode Range 129
- Common Mode Voltage 234
- Compare Match 94, 222
- Compiler 40, 44, 45, 47, 51, 52, 62, 79
- const 49
- Count 51, 102, 103, 195, 197, 208, 230, 314
- CPU 18
- crystal oscillator 35
- Current Shunt Monitor 234

- D**
- DAC 215
- Darlington-Transistor 180
- Datenblatt 23, 42, 46
- Datentyp 49, 50
- DC Characteristics 27
- Debugging 55, 57, 60, 161
- DebugWIRE 61, 62
- Dekrementieren 41
- Delay 98
- Delta-Sigma-Wandler 102, 264
- Device Under Test 230
- Dezibel 109, 219, 220, 303, 304
- Dezimalsystem 321
- Die 20
- differenziell 128
- Digital Input Disable 116, 200
- Digital-Analog-Wandler 113, 215, 216
- DIP 19, 20, 54, 63, 72, 201, 316, 319
- diskret 100, 195
- Division 51, 71
- Dotierung 256
- Drain 174
- DRAM 32
- Drossel 200

Dualsystem 321
 Durchlassbereich 109
 Durchlassspannung 301
 Durchsteppen 52, 61, 62
 DUT 230
 duty cycle 118
 Dynamische Speicherverwaltung 47

E

EEMEM 287
 EEPROM 30, 287
 Effektivwert 211, 212
 Eichen 305
 Eigenerwärmung 253
 einadrig 317
 Eingang 100
 Eingangsspannungsbereich 102, 195, 203,
 210, 260, 305
 Einschwingen 35, 58, 90, 159, 162, 163
 Einweggleichrichter 213
 Electrical Characteristics 25
 Elektrolytkondensator 72
 ELKO 74
 Embedded Systems 7
 Emitter 178
 Emulator 51, 61
 EMV 108, 174, 182
 Endlosschleife 77, 79
 Entprellen 90, 281, 285
 enum 153
 EPROM 30
 E-Reihe 299
 Errata 25
 Erwartungswert 209
 ESD 27, 55, 167, 191, 193, 281
 Externer Interrupt 88
 Externer Takt 36
 Externer Takteingang 243

F

factory calibration 37
 Fast-PWM 119, 218, 222
 Fehlerbehandlung 43, 165
 Feldeffekttransistor 173
 FET 173
 Filter 108, 122, 135, 200, 219, 261, 281,
 295
 Flag 42, 70
 Flankensteilheit 193
 Flash 31, 289, 315
 Flashen 53
 Flash-Wandler 103
 Fließkomma 49, 51, 61, 314
 Fließkommaeinheit 51
 Fließkommazahlen 45
 Flüchtige Speicher 31
 Folienkondensator 238
 FRAM 31
 Free Running 202
 Frequenz 96, 239, 240
 Frequenzgang 221
 Full Swing Crystal Oscillator 58
 Full-Duplex 130
 Fuse 34, 54, 57

G

GAIN 196
 Galvanische Trennung 129, 136, 148, 183,
 237
 Gate 174
 Gauß 208
 GCC 45, 86
 Gehäuse 20
 Genauigkeit 203, 233, 253, 260, 298
 General Purpose I/O Register 43, 165
 Geradengleichung 308
 Gleichspannung 211
 Global Interrupt Enable 42

Glockenkurve 209
GND 73
GPIO 77, 99
GPOR 165
Grace Hopper 60
Grenzfrequenz 108, 295, 304

H

Halbleiter 254
Halbleitertemperatursensor 257
Half-Duplex 130
Half-Scale 217
Hallsensor 237
Handshake 127
Harvard-Architektur 18
Hauptroutine 79
Headerfile 48
HIGH 27, 33, 99, 123, 128, 167
High Voltage Serial Programming 57
Highside 172
Highside-Messung 234
Histogramm 208
Hochpass 296
Hysterese 247

I

I2C 141, 204, 264
IGBT 178
In System Programming 55
Induktivität 173, 181, 247, 294
Initial Accuracy 198, 258
Inkrementieren 41, 93
Input Bias Current 236
Instrumentenverstärker 262
Integrale Nichtlinearität 105
Interner Oszillator 35
Interrupt 38, 84
Interrupt-Flag 87
Interruptquelle 84

Interruptroutine 39, 47, 51, 61
Interruptserviceroutine 79, 82, 84
Interruptvektor 84
invertierend 114
Isolierung 317
ISP 55, 73, 78, 136
ISP-Header 56
ISR 39, 84

J

Jitter 106, 202, 241
JTAG 57, 62
JTAG-Fuse 63
Justieren 305

K

Kalibrierung 37, 159, 254, 258, 305
Kapazität 237, 247, 294
Kathode 183, 278
Kausalität 210
Kelvin Sensing 228, 233
Kennlinie 300
Keramik 254
Keramikkondensator 72, 238
Keramikresonator 36
KERKO 73, 199, 238, 281
Kern 15
Keypad 281
Kleinsignaldiode 213, 281
Kollektor 178
Komparator 114, 238, 243, 251
Kompiliervorgang 53
Konstantstromquelle 229
Kontaktwiderstand 182, 229, 233, 260
kontinuierlich 101, 195
Korrelation 209
KTY 256, 311
Kurzschluss 172, 177, 191, 192, 263

L

Ladekurve 247
 Langzeitstabilität 198, 199, 207, 228, 233
 Latch 169
 Latenz 89
 LC-Filter 200, 220
 Least Significant Bit 102
 Leckstrom 263
 LED 41, 64, 72, 77, 162, 168, 173, 272, 300
 Leitfähigkeit 253
 Leitungskapazität 229
 Lichtbogen 182
 LIN-Bus 38
 Linearbetrieb 177
 Linearisierung 311
 Linearität 203
 Little Endian 136
 LM75 264
 Lochrasterplatine 319
 Lock-Bit 59
 Logarithmus 248
 Logic Level 176
 Lookup-Tabelle 290, 315
 LOW 27, 33, 99, 123, 128, 167
 Low Power 159
 Low Power Crystal Oscillator 58
 Lowside 172
 Lowside-Messung 234
 LSB 102, 195

M

Magnetfeld 237
 Manual 26
 Maschinensprache 53, 62
 Massepotenzial 173
 Masseschluss 172
 Matrixtastatur 281
 MCU 15

Metalloxid 254
 Microwire 122, 135, 167, 169, 205, 206
 Mischspannung 239
 MISO 135
 Mittelwert 52, 104, 115, 208, 209, 210, 261, 314
 Modem 127
 Modulo 276
 MOSFET 173, 174, 181
 MOSFET-Treiber 177
 MOSI 135
 Most Significant Bit 102
 MRAM 31
 MSB 102
 Multimaster 130
 Multiplex 110, 115, 201
 Multi-Processor Communication Modus 132

N

NACK 145
 Nichtflüchtige Speicher 30
 nichtinvertierend 114
 NMOS 174
 Noise Canceler 241
 normalverteilt 208
 Normreihe 221, 299
 NOT 67
 NPN 176, 178
 NTC 254, 257
 Nulldurchgang 183, 188, 239
 Nulldurchgangsdetektor 184, 188

O

Offset 203, 234, 255, 311
 OPCode 292
 Operating Voltage 26
 Operationsverstärker 196, 213, 221, 229, 235

Optimierungslevel 52
Optokoppler 130
OPV 196, 212
OR 66, 281
OSCCAL 37
Oszillator 34, 154, 159
Oszilloskop 64, 297
OTP 30
Overflow 38, 40, 93
Oversampling 106, 112

P

Package 20
Parallel Programming 57
PCB 64
Pegelwandler 167
Periodendauer 211, 223, 239
Peripheriemodul 110
Personal Computer 7
Phase and Frequency Correct PWM 121
Phase Correct PWM 121, 278
Phasenabschnittsteuerung 185
Phasenanschnittsteuerung 183
Pin 99
Pinchange-Interrupt 91
Pinerweiterung 168
Pipeline-Wandler 103
Platine 64
PLL 35, 93
PMOS 174
PNP 178
Polarität 210
Port 99
Portierbarkeit 49, 51
Potentiometer 162, 260, 278, 306
ppm 35, 106, 298
Präfix 298
Präprozessor 81
Preliminary 26
Prescaler 41, 93, 201, 244, 278
printf 60, 61, 271
Priorität 84
PROGMEM 289
Programmieradapter 53, 54
Programmieren 17
Programmiersprache C 45, 47
Programmierungsumgebung 54
Programmspeicher 17
Promille 298
Prozent 298
Prozessor 15
Pt100 254
Pt1000 254, 311
PTC 193, 253, 257
Pufferkondensator 74
Pulldown 161, 167, 169, 175, 217
Pullup 73, 88, 100, 141, 161, 167, 169,
175, 217, 281
Pulspaketsteuerung 188
Pulsweitenmodulation 117
Punktstreifenrasterplatine 319
Punkt-zu-Punkt 128, 135
PWM 109, 117, 118, 121, 170, 217, 278

Q

QFN 23
QFP 23
Quarz 36

R

Rail-to-Rail 196
RAM 17
ratiometrisch 219, 230, 247, 313
Rauschen 104
RC-Filter 217
RC-Oszillator 35, 37, 250
RC-Snubber 173
Receive 121, 127

- Receiver 130
- Reedrelais 182
- Referenzspannung 102, 103, 105, 197, 198, 200, 207, 215, 227, 229, 230, 258
- Referenzwiderstand 227, 229, 230, 256, 313
- Refresh 32
- Register 40, 42, 46
- Relais 181, 182
- Relative Genauigkeit 105
- Relaxationsoszillator 250
- Reset 154
- RESET 55, 58, 63, 72, 136, 167, 175
- Resonanzfrequenz 249, 261, 295, 296
- Revision 25
- RGB-LED 278
- RMS 212
- RS-232 122, 127
- RS-422 123, 128
- RS-485 123, 128
- RSTDISBL 57
- RTD 254
- Rundungsfehler 52, 124, 239, 314
- Rx 121, 127

- S**
- Samples-Per-Second 101
- Sättigung 179, 213
- Sättigungsgrenze 247
- Schaltswelle 176
- Schiebeoperator 68
- Schieberegister 169, 273
- Schlafzustände 81, 159, 162
- Schreibzugriff 287
- Schutzdiode 191, 195, 263
- Schutzschaltung 190
- Schwingkreis 249, 296
- Schwingquarz 35
- Schwingungsformel 249
- SCK 135
- SCL 141
- SDA 141
- SDRAM 32
- sei() 86, 164
- sequentiell 39
- Serielle Schnittstelle 122, 127
- Settling Time 113, 215
- Shift 68, 71
- Shunt 233, 235
- Sicherung 192
- Signalpegel 27
- Signal-Rauschverhältnis 304
- Signed 49, 65, 83
- Single Conversion 202
- Sleep 111
- Slew Rate 213
- SMBus 148
- SMD 22
- SNR 304
- Socket 21
- Software-PWM 222
- SO-Gehäuse 22
- Source 174
- Spannungsfolger 196, 213
- Spannungsreferenz 113
- Spannungsteiler 114, 195, 227, 240, 255, 256, 294
- Speed Grades 33
- Sperrbereich 109
- Sperrschicht 178
- SPI 56, 122, 135, 204, 291
- SPI-Flash 291
- Spitzenwert 211, 212
- Spitzenwertgleichrichter 212
- SRAM 31
- SREG 42
- Stabilität 198, 233
- Standard C Library 61

Startbit 122
State Machine 150
static 83
Statusmeldungen 61
Steckbrett 72, 315
Steuerleitung 127
Stopbit 122
Störung 104, 110, 115, 129, 155, 203, 207,
215, 239, 241
Streifenrasterplatine 319
String 271
Stringoperation 290
Strom-Spannungswandler 235
Stromverstärkung 179
Stromwandler 237
Sukzessive Approximation 102, 204
Summary 26
Supressordiode 191

T

Takt 15, 33, 40, 46, 58, 163
Taktrate 34, 99, 159
Taster 281
Tastgrad 118
Tastkopf 64
Tastverhältnis 118
Temperatur 253
Temperaturabhängigkeit 35, 199
Temperaturbereich 299
Temperaturdrift 207
Temperaturerhöhung 229
Temperaturkoeffizient 253, 256
Temperatursensor 227
Terminalprogramm 271
Terminierungswiderstand 130
Testpunkt 64
Thermistor 254
Thermocouple 262
Thermoelement 262

Thermospannung 262
Threshold 158
Through-Hole 21
Thyristor 183
Tiefpass 295, 304
Tiefpassfilter 108, 115, 118, 217, 220, 241
Timer 40, 93
Timer Overflow 40, 222
Toggeln 41, 70
Torzeit 243
Transformator 130, 237
Transimpedanzverstärker 235
Transmit 121, 127
Transmitter 130
Treiber 29, 176, 180, 181, 192, 217
TRIAC 183, 188
Trigger 64, 84
TTL 28, 168
TWI 87, 141, 168, 204, 266, 291
Twisted-Pair 130
Tx 121, 127
typedef 153

U

UART 60, 121
Überabtastung 112
Übergangsbereich 109
Überspannung 191
Uhrenquarz 36, 38, 97, 162
Ungenauigkeit 52, 155, 230, 248, 256
unidirektional 234
Unit Load 132
Unsigned 49, 65, 83, 125
USART 121, 123, 135, 271
USB 149
USI 141

V

Varianz 209

Varistor 191
VCC 26, 73, 195
verdrillt 129, 130
Verlustleistung 233
Verpolung 192
Versorgungsspannung 26
Verstärker 178, 196, 215, 234, 235, 255,
256, 262, 304, 314
Vierleitermessung 228, 233, 256
volatile 47, 97, 112, 186, 189, 224, 242,
244
Von-Neumann-Architektur 18
Vorwiderstand 72, 77, 177, 191, 273, 277,
278, 300
vorzeichenbehaftet 83
VREF 258

W

Watchdog 154

Wechselspannung 182, 210
Wellenwiderstand 130
Widerstand 227
Window Watchdog 157
Worst Case 155

X

XOR 67

Z

Zähler 40, 93
Z-Diode 176, 192
Zeiger 47
Zeitkonstante 241, 248
Zufall 209
Zustandsautomat 150
Zweierkomplement 65, 265, 322

Irmtraut Meister / Lukas Salzburger

AVR-

Mikrocontroller-Kochbuch

Programmieren ist wie Kochen

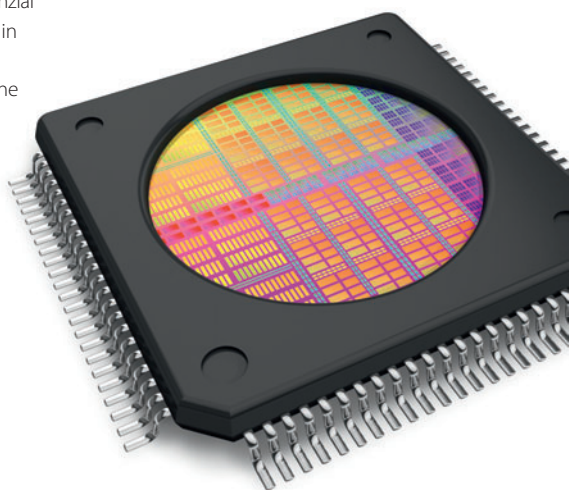
Rezept auswählen, Zutaten zusammenstellen – und genießen. Nach genau diesem Konzept finden Sie in diesem Buch alles, um Ihr „Mikrocontroller-Süppchen“ zu kochen: Von den ersten Programmierschritten über Messungen unterschiedlichster Größen bis zum Erzeugen von Signalen und zur Kommunikation über diverse Schnittstellen.

Entdecken Sie die schier endlosen Möglichkeiten der Mikrocontroller! Mit nur wenig Programmieraufwand verwirklichen Sie im Handumdrehen Ihre Ideen. Schritt für Schritt begleitet dieses Buch Sie von den allgemeinen Grundlagen zur praktischen Umsetzung und erleichtert so auch komplexe Programmierungen.

Am Beispiel des AVR®-Mikrocontrollers von Atmel® lernen Sie das Potenzial von Mikrocontrollern kennen und können sich dadurch auch leicht in „fremde“ Mikrocontroller einarbeiten. Für Einsteiger bietet das Buch auch Hinweise zur Programmierung von Bitoperationen und einfache Codegerüste – so bleiben keine Fragen offen.

Die Rezepte aus diesem Buch:

- Mikrocontroller-Grundlagen
- Programmierung und Implementierung
- Digitale Ein- und Ausgänge
- Spannungsmessung
- Spannungen ausgeben
- Widerstandsmessung
- Strommessung
- Zeit- und Frequenzmessung
- Kapazitäts- und Induktivitätsmessung
- Temperaturmessung
- Kommunikation mit Menschen
- Daten speichern



39,95 EUR [D]
ISBN 978-3-645-65126-4

Besuchen Sie unsere Website
www.franzis.de

FRANZIS