1 Einführung

Seit der Einführung des 8080 von INTEL hat sich die Entwicklung in 2 Hauptrichtungen gegabelt.

a) Hochleistungsprozessoren wie der PENTIUM von INTEL für PC's und Workstations in CISC-Architektur (CISC = Complex Instruction Set Computer)
b) Einfache Controller mit kompaktem Aufbau; sogenannte single chip Mikrocontroller. Der erste Vertreter war 1976 der 8048: 1k ROM, 64 Byte RAM, 8 Bit Timer, 27 I/O's,...
   Einsatzbereiche: Messtechnik, Konsumelektronik, Auto, Computerperipherie, Waschmaschine,...
   ⇨ sehr große Stückzahlen => Preis von 1 ... 25 US$

In den 80er-Jahren haben sich der 8051 von INTEL und seine zahlreichen Derivate deutlich durchgesetzt. Seit einiger Zeit gibt es aber von 2 Seiten eine Bedrängung:

a) 16 Bit uC
b) 8 Bit RISC Controller (RISC = Reduced Instruction Set Computer) wie

+ die PIC – Familie von Fa. MICROCHIP
+ die AVR – Familie von Fa. ATMEL

Vorteile der AVR – Familie: Kombination von leistungssparendem CMOS Herstellungsprozess und RISC – Architektur und im System programmierbarem Flach – EPROM
=> sehr effiziente Lösung für On-Board-Controllingaufgaben.

1.1 Die RISC – Architektur

Früher, als die Speichertechnologie der Technologie zur Herstellung von CPU's hinterherhinkte, wurden komplexe Befehle entwickelt, um die Zeit zwischen zwei Befehlsdecodierungen zu nutzen. Es zeigte sich jedoch, dass von den manchmal bis zu 300 ! Befehlen im Wesentlichen ca. 20 % in immer wiederkehrender Folge den Großteil des Programmablaufes bestreiten.
Da inzwischen auch die Speicherzugriffe drastisch verbessert wurden, kehrte man zur RISC – Architektur zurück !

Merkmale :
+ Beschränkung des Befehlsvorrates auf eine übersichtliche Anzahl effektiver Befehle

+ Nicht mehr **1** Akkumulator, sondern eine größere Anzahl gleichberechtigter Arbeitsregister

+ Speicherorganisation nach Havard (getrennte Speicherbereiche für Programme und Daten)

+ Abarbeitung (fast) aller Befehle innerhalb eines einzigen Maschinentaktes !

+ Optimierung von Hardware und Befehlsvorrat für Hochsprachenprogrammierung !

AVR – Kenndaten:
+ 118 Befehle (89)
+ 32 Arbeitsregister (statt 1 Akku)
+ Havard – Architektur (Befehl ausführen und nächsten Befehl holen in einem einzigen Taktzyklus
+ verschiedene Adressierungsmodi
+ max. 12 MIPS Durchsatz (bei 12 MHz Taktfrequenz)
+ für C optimiert

1.2 Die Mitglieder der AVR – Basic – Line

Aus Tab. 1.2.1 ist ersichtlich, welche Leistungsmerkmale die 4 Vertreter der AVR – Basic Line haben.
Zu den Features des AT90S8535 bzw. des AT90LS8535 siehe Datenblatt Seite1 .

Funktion	*AT90S8515*	*AT90S4414*	*AT90S2313*	*AT90S1200*
Anzahl der Befehle	118	118	118	89
Flash-EPROM	8 K Byte	4 K Byte	2 K Byte	1 K Byte
Programmierung über SPI-Interface	✓	✓	✓	✓
EEPROM	512 Byte	256 Byte	128 Byte	64 Byte
SRAM	512 Byte	256 Byte	128 Byte	(nur Stack)
Working-Registers	32	32	32	32
Ein- / Ausgabe- Pins	32	32	15	15
Serieller UART	✓	✓	✓	–
SPI-Interface	✓	✓	–	–
Watchdog-Timer	✓	✓	✓	✓
8-Bit Timer/Counter mit Vorteiler	✓	✓	✓	✓
16-Bit Timer/Counter mit Vorteiler	✓	✓	✓	–
Pulsweitenmodulator	2	2	1	–
Interruptquellen	12	12	10	3
Analog-Comparator	✓	✓	✓	✓
Idle Mode	✓	✓	✓	✓
Power Down Mode	✓	✓	✓	✓
Software-Sicherung	✓	✓	✓	✓
Betriebsspannung	2,7 ... 6,0 V	2,7 ... 6,0 V	2,7 ... 6,0 V	2,7 ... 6,0 V
Taktfrequenz bei				
V_{CC} = +5V	0 ... 8 MHz	0 ... 8 MHz	0 ... 10 MHz	0 ... 12 MHz
V_{CC} = +3V	0 ... 4 MHz	0 ... 4 MHz	0 ... 4 MHz	0 ... 4 MHz
Gehäuse	PDIP 40	PDIP 40	PDIP 20	PDIP 20
	PLCC 44	PLCC 44	SOIC 20	SOIC 20
				SSOP 20

Tab. 1.2.1: Leistungsmerkmale der vier Vertreter der *AVR*-Basic Line

2 Überblick

2.1 Grundlegende Leistungsmerkmale

Der Grundaufbau aller Mikrocontroller ist praktisch immer derselbe und wird als bekannt vorausgesetzt. (CPU = Ablaufsteuerung aus Programmzähler u. Dekodierlogik + Rechenwerk; Programmspeicher, Datenspeicher, Ein-/Ausgabe – Interface zur Peripherie, Systembus).

ALU und 32 Arbeitsregister sind direkt miteinander verbunden => Nadelöhr Akkumulator beseitigt.

Befehlsstruktur: 118 (89) Befehle mit einer Wort-Breite von16 Bit = 2 Byte (2 Ausnahmen)
Bis auf wenige Ausnahmen werden alle Befehle des AVR innerhalb eines Systemtaktes ausgeführt!

Befehlsausführungszeiten: Die Abb. 2.1.2 zeigt in einer Gegenüberstellung die Anzahl der benötigten Oszillator-Taktimpulse für die Ausführung zweier aufeinanderfolgender Instruktionen für die Prozessoren 68HC05, 80C51 und Vertreter der PIC- und der AVR-Familie.

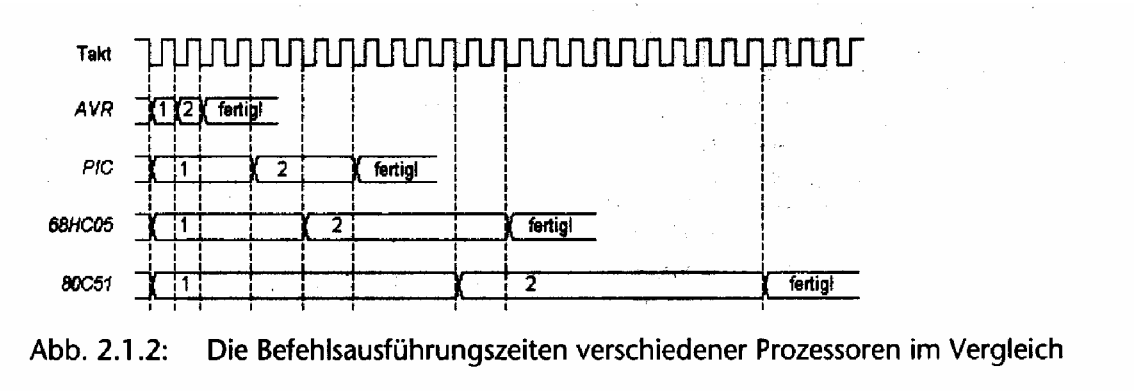


Abb. 2.1.2: Die Befehlsausführungszeiten verschiedener Prozessoren im Vergleich

Speicherarchitektur: Nicht das klassische von Neumann Konzept mit Daten und Befehle in einem Speicher ! Bei AVR – Controllern sind diese beiden Bereiche auch physikalisch verschieden aufgebaut !

Programmspeicher**: Flash – EPROM** on-Chip, 2 Byte breit, Erweiterung nicht möglich !
Interner Datenspeicher für flüchtige Daten: **SRAM** => bis 0 Hz möglich !, extern erweiterbar
Datenspeicher für nichtflüchtige Daten: **EEPROM**, kein eigenes Programmiergerät notwendig !

Der I/O – Bereich: 64 Byte im SRAM – Block für PORT A bis D und auch für die Status und Steuer – Register der on-Chip Peripheriefunktionen (Timer/Counter, UART, EEPROM-Schreib-/Lesezugriff, SPI-Interface,...)

Interrupts und Unterprogramme:
Unterprogramm und Interrupttechniken lassen sich in gewohnter Weise einsetzen !
Unterprogramme lassen sich auch verschachteln.

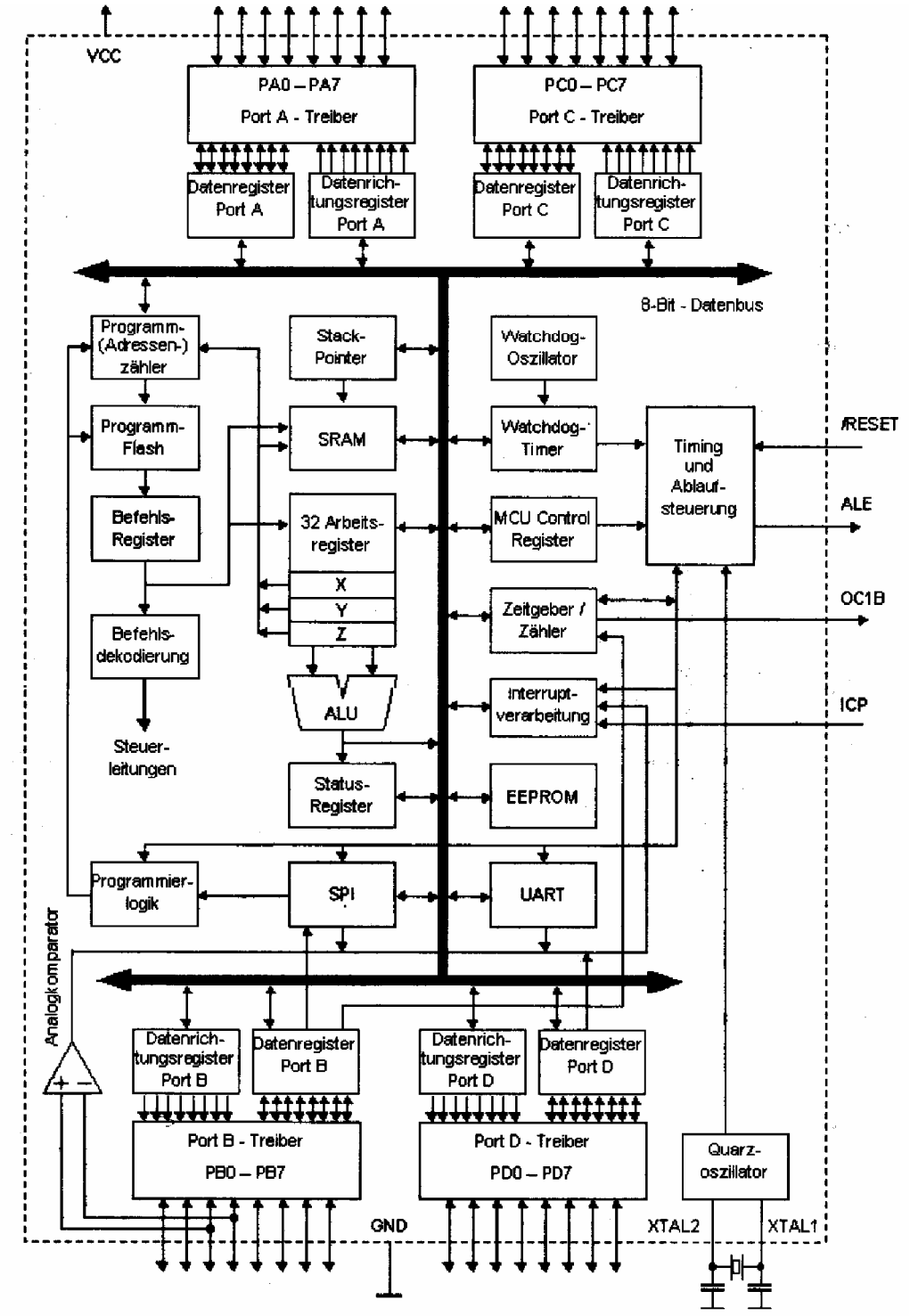
Eine Sonderform stellen die Interruptroutinen dar: Bei ihnen erfolgt der Aufruf nicht gezielt aus dem Programm heraus, sondern asynchron zu diesem durch Eintritt eines bestimmten Ereignisses.
z.B.: Ablauf eines Timers, externe Interruptanforderung, Empfang eines Zeichens über den UART, ...

Je nach Controller stehen bis zu 14 Einsprungadressen (Interruptvektoren) zur Verfügung !
Achtung: Derselbe Unterbrechungsgrund liegt bei einem anderen Controller meist auf einer anderen Adresse !

Peripheriefunktionen: z.B.: beim AT90S8535:

+ 4 PORT's a 8 Ein-/Ausgabeleitungen
+ 1 programmierbarer UART
+ 1 synchrone SPI-Schnittstelle
+ 1 8-Bit T/C0
+ 1 8-Bit T/C2 mit 1 PWM Ausgang
+ 1 16-Bit T/C1 mit Compare/Capture - Funktionen und 2 PWM Ausgängen
+ 1 Watchdog - Timer
+ 1 Analogkomparator

2.2 Blockschaltbild des AT90S8515



2.3 Gehäusebauformen und Pinbelegung

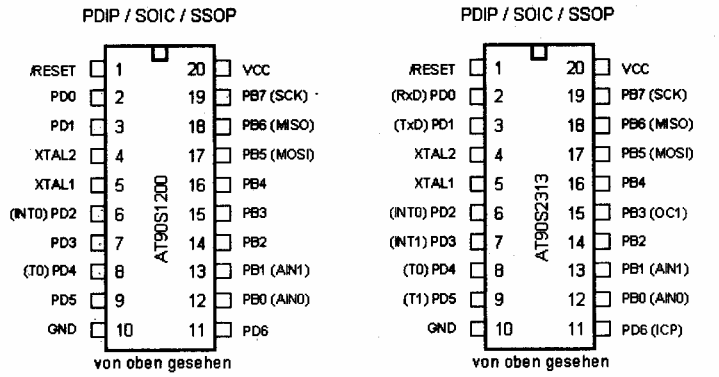


Abb. 2.3.1: Gehäusebelegung des AT90S1200 und des AT90S2313

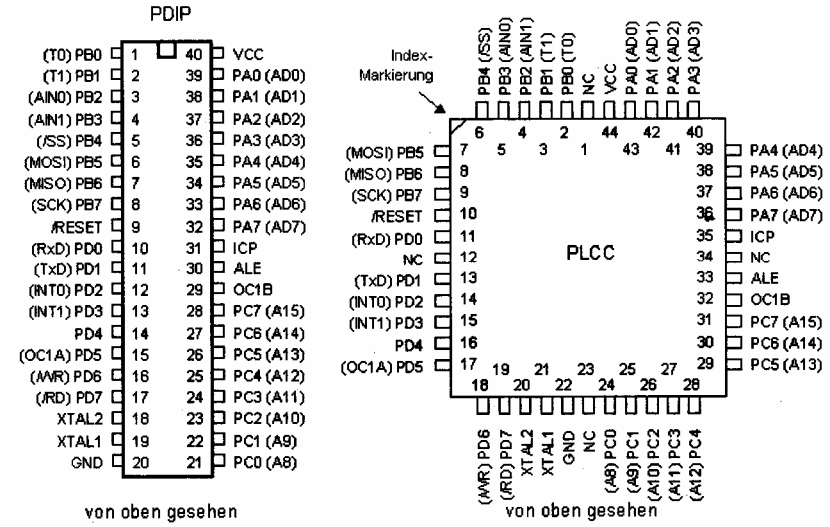


Abb. 2.3.2: Gehäusebauformen des AT90S4414 und des AT90S8515. Die Anschlußbelegung ist für beide Bausteine gleich.

Die Pinbelegung des AT90S8535 finden Sie im Datenblatt !

2.4 Systemtakt Φ

Ein integrierter Oszillator sorgt mit einem externen Quarz für die Erzeugung des Systemtaktes Φ (XTAL1 u. XTAL2) !
Weiters kann ein externes Taktsignal an XTAL1 eingespeist werden !
Beim AT90S1200 kann auch der on-Chip verfügbare RC-Oszillator zur Systemtakterzeugung verwendet werden.

2.5Programmierung in C

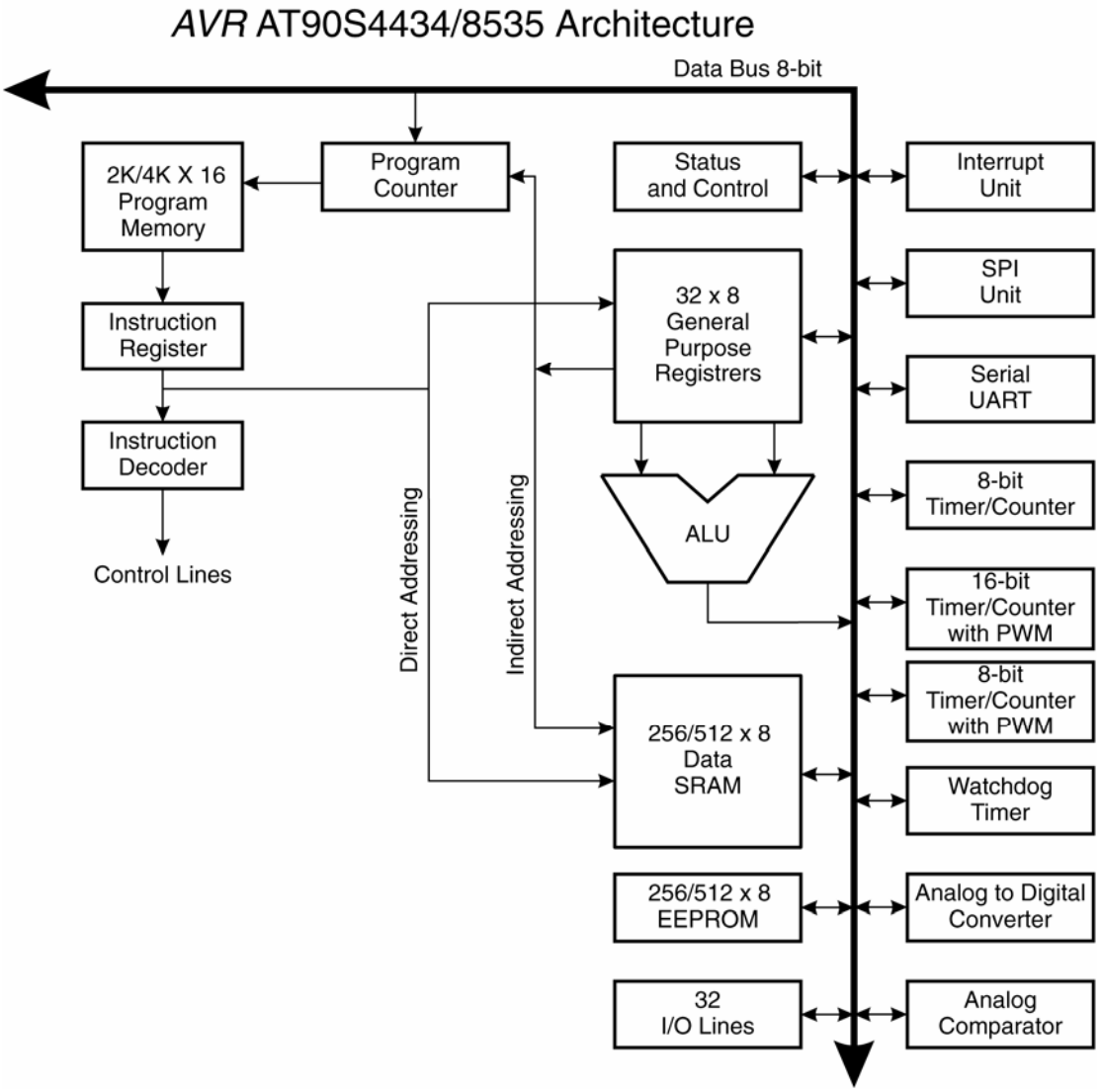
Da die AVR Mikrocontroller Neuentwicklungen sind, wurde auf die Optimierung bei der Nutzung von C hoher Wert gelegt ! Bei Benchmarks benötigte z.B.: der 80C51 ca. 1,7 mal länger als der AVR !
C-Compiler gibt es bereits ab US$ 50,- !

3 CPU und Speicher

Allen Vertretern der AVR – Basic – Line gemeinsam sind die CPU und mehrere interne Speicherblöcke; dies ist Inhalt dieses Kapitels !

3.1 Systemsteuerung und ALU

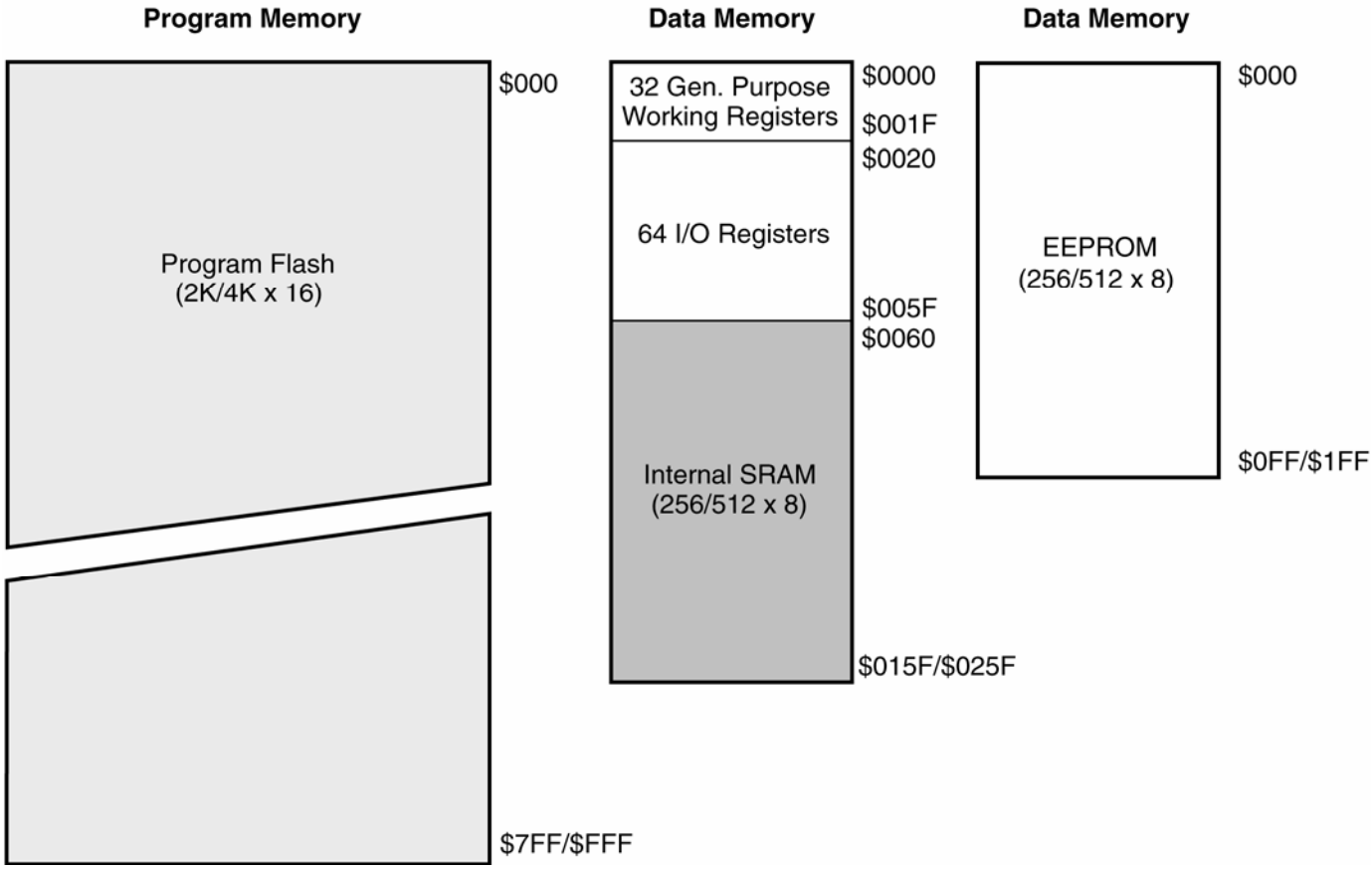
Einen Überblick über die AVR RISC µC – Architektur zeigt uns nachfolgendes Bild:



Die Systemsteuerung regelt den Programmablauf und die Zusammenarbeit der einzelnen chip-internen Module. Alle für die Verarbeitung eines Befehls notwendigen Operationen, wie Holen des Befehls vom Befehlsspeicher, Decodierung und Ausführung werden von der Systemsteuerung koordiniert.
Die ALU und die 32 Arbeitsregister sind direkt miteinander verbunden, sie führen Additionen, Subtraktionen, Schiebeoperationen und die logischen Verknüpfungen AND, OR und EXOR aus. Die ALU der AVR – Controller ist so leistungsfähig, dass während eines einzigen Systemtaktes zwei Operanden aus dem Registerbereich in die ALU geholt werden können, die Operation durchgeführt und das Ergebnis im Zielregister gespeichert wird.

3.2 Die Speicheraufteilung

Nachfolgendes Bild zeigt die Speicheraufteilung beim 4434/8535:
Die Speicherbereiche gliedern sich in Programmspeicher (Flash EPROM), Datenspeicher (SRAM) und Datenspeicher für nichtflüchtige Daten (EEPROM).



Der Programmspeicher (Flash-EPROM):

Alle AVR RISC µC's beinhalten ein (jeweils verschieden großes) Flash-EEPROM, welches vom Anwender frei programmiert und elektrisch wieder gelöscht werden kann.
Dies spart mehrfach Platz, insbesondere sind alle I/O-Pins für Anwendungen zur Verfügung. Das EEPROM kann auch über die serielle SPI-Schnittstelle direkt in der Schaltung programmiert werden. Größe siehe Tab. 1.2.1 (Seite 2)

Der Datenspeicher (SRAM):

Das statische RAM ist aus RS-FF's aufgebaut, benötigt daher keinen Refresh, damit kann die Taktrate auch gegen 0 gehen ! Es besteht aus 3 Teilbereichen, die einen durchgehend adressierten Block darstellen. Der unterste Teilbereich ist der Registerbereich, dann folgt der I/O - Bereich, gefolgt vom internen RAM und schlussendlich das optionale externe RAM !

Der Arbeits - Registerbereich: (0x000 bis 0x01F = 00d bis 31d = 32 Working - Register)
Der Ein-/Ausgabe- (I/O-) Bereich: (0x20F bis 0x05F = 32d bis 95d = 64 I/O - Register)
In diesem Teil des SRAMs sind die gesamten Register zur Programmierung, Steuerung und Signalisierung sämtlicher peripherer Funktionen der AVR Controller untergebracht.

Adresse	*Name*	*Funktion*	*Initial*	*1200*	*2313*	*4414*	*8515*
$3F ($5F)	SREG	StatusREGister	$00	✓	✓	✓	✓
$3E ($5E)	SPH	Stack Pointer, High Byte	$00	–	–	✓	✓
$3D ($5D)	SPL	Stack Pointer, Low Byte	$00	–	✓	✓	✓
$3B ($5B)	GIMSK	General Interrupt MaSKen Register	$00	✓	✓	✓	✓
$3A ($5A)	GIFR	General Interrupt Flag Register	$00	–	✓	✓	✓
$39 ($59)	TIMSK	Timer/Counter Interrupt MaSKen Register	$00	✓	✓	✓	✓
$38 ($58)	TIFR	Timer/Counter Interrupt Flag Register	$00	✓	✓	✓	✓
$35 ($55)	MCUCR	MCU general Control Register	$00	✓	✓	✓	✓
$33 ($53)	TCCR0	Timer/Counter0 Control Register	$00	✓	✓	✓	✓
$32 ($52)	TCNT0	Timer/Counter0 (8 Bit)	$00	✓	✓	✓	✓
$2F ($4F)	TCCR1A	Timer/Counter1 Control Register A	$00	–	✓	✓	✓
$2E ($4E)	TCCR1B	Timer/Counter1 Control Register B	$00	–	✓	✓	✓
$2D ($4D)	TCNT1H	Timer/Counter1 High Byte	$00	–	✓	✓	✓
$2C ($4C)	TCNT1L	Timer/Counter1 Low Byte	$00	–	✓	✓	✓
$2B ($4B)	OCR1AH	T/C1 Output Compare Reg. A High Byte	$00	–	✓	✓	✓
$2A ($4A)	OCR1AL	T/C1 Output Compare Reg. A Low Byte	$00	–	✓	✓	✓
$29 ($49)	OCR1BH	T/C1 Output Compare Reg. B High Byte	$00	–	–	✓	✓
$28 ($48)	OCR1BL	T/C1 Output Compare Reg. B Low Byte	$00	–	–	✓	✓
$25 ($45)	ICR1H	T/C1 Input Capture Register High Byte	$00	–	✓	✓	✓
$24 ($44)	ICR1L	T/C1 Input Capture Register Low Byte	$00	–	✓	✓	✓
$21 ($41)	WDTCR	Watchdog Timer Control Register	$00	✓	✓	✓	✓
$1F ($3F)	EEARH	EEPROM Address Register High Byte	$00	–	–	–	✓
$1E ($3E)	EEAR(L)	EEPROM Address Register (Low Byte)	$00	✓	✓	✓	✓
$1D ($3D)	EEDR	EEPROM Daten Register	$00	✓	✓	✓	✓
$1C ($3C)	EECR	EEPROM Control Register	$00	✓	✓	✓	✓
$1B ($3B)	PORTA	Datenregister Port A	$00	–	–	✓	✓
$1A ($3A)	DDRA	Datenrichtungsregister Port A	$00	–	–	✓	✓
$19 ($39)	PINA	Eingangs-Pins Port A	Hi-Z	–	–	✓	✓
$18 ($38)	PORTB	Datenregister Port B	$00	✓	✓	✓	✓
$17 ($37)	DDRB	Datenrichtungsregister Port B	$00	✓	✓	✓	✓
$16 ($36)	PINB	Eingangs-Pins Port B	Hi-Z	✓	✓	✓	✓
$15 ($35)	PORTC	Datenregister Port C	$00	–	–	✓	✓
$14 ($34)	DDRC	Datenrichtungsregister Port C	$00	–	–	✓	✓
$13 ($33)	PINC	Eingangs-Pins Port C	Hi-Z	–	–	✓	✓
$12 ($32)	PORTD	Datenregister Port D	$00	✓	✓	✓	✓
$11 ($31)	DDRD	Datenrichtungsregister Port D	$00	✓	✓	✓	✓
$10 ($30)	PIND	Eingangs-Pins Port D	Hi-Z	✓	✓	✓	✓
$0F ($2F)	SPDR	SPI I/O Daten-Register	$00	–	–	✓	✓
$0E ($2E)	SPSR	SPI Status-Register	$00	–	–	✓	✓
$0D ($2D)	SPCR	SPI Control-Register	$04	–	–	✓	✓
$0C ($2C)	UDR	UART Daten-Register	$00	–	✓	✓	✓
$0B ($2B)	USR	UART Status-Register	$20	–	✓	✓	✓
$0A ($2A)	UCR	UART Control-Register	$00	–	✓	✓	✓
$09 ($29)	UBRR	UART Baudraten-Register	$00	–	✓	✓	✓
$08 ($28)	ACSR	Analogkomparator Control und Status Register	$00	✓	✓	✓	✓

Tab. 3.2.1: Der I/O-Bereich der AVR Basic Line (Reservierte oder unbenutzte Adressen sind nicht enthalten)

Die auf der vorigen Seite dargestellte Tabelle liefert Informationen zu allen I/O Registern der AVR - Basic Line. Die entsprechenden Register für den 4434/8535 sehen Sie in Table 2 des Datenblattes auf den Seiten 15 und 16.

Die Bedeutung der einzelnen Bits der Register wird bei der Behandlung der verschiedenen Peripheriefunktionen detailliert erklärt.

Hier nur zwei spezifische Register:

Status Register **SREG**: enthält die Bedingungsbits (Flags) der AVR – Familie.
MCU Control Register **MCUCR**: enthält die Steuerbits für generelle MCU – Funktionen.

Das interne RAM: (0x60 bis 0xXY = 0 bis 512 Byte Speicher)

Darin werden in der Regel vom Programm benutzte Daten und Variablen abgespeichert; dessen Größe ist je nach Baustein unterschiedlich (siehe Tab. 1.2.1)

Beim 4414, 8515 sowie beim 4434/8535 besteht die Möglichkeit den SRAM – Bereich durch Anschluss externer RAM - Bausteine auf bis zu 64 kB zu vergrößern !
Dazu muss allerdings auf die beiden Ausgabeports PA und PC verzichtet werden, die in diesem Fall als Daten- und Adressbus verwendet werden.

Der Datenspeicher für nichtflüchtige Daten (EEPROM):

Die AVR – Mikrocontroller haben zwischen 64 Byte und 512 Byte an EEPROM – Speicher zur Speicherung nichtflüchtiger Daten und Konstanten. Verwendet wird dieser Bereich z. B. zur Speicherung von Korrekturparametern bei der Kalibrierung von uC-gesteuerten Messgeräten.

Lebensdauer: > 100.000 Schreib/Lesezyklen !

Verwendete Register:
+ EEPROM Adress Register EEAR
+ EEPROM Data Register EEDR
+ EEPROM Control Register EECR

Der Stapelzeiger (Stackpointer)

Der Stack dient zur Abspeicherung von Rücksprungadressen bei Unterprogrammaufrufen und der Parameterübergabe an Unterprogramme. Häufig werden auch Variablen und Zwischenergebnisse temporär auf dem Stack zwischengespeichert.
Der Stack ist je nach Controller unterschiedlich realisiert. Bei den größeren Vertretern kann er wahlweise im internen oder im externen RAM liegen. Dort ist der Stapelzeiger 16 Bit breit und besteht aus zwei 8bit – Registern: SPH und SPL .

3.3 Adressierungsmöglichkeiten von Programm- und Datenbereichen

Siehe Datenblatt Seiten 10 bis 13.

3.4 RESET und Interrupt

Der RESET – Vorgang sorgt bei allen AVR – Mikrocontrollern für einen definierten Start des Programms ab der Adresse 0 und für die Belegung der internen Steuerregister mit den vordefinierten Werten nach Tabelle 3.2.1.

Auch die verschiedenen Interrupts haben genauso wie der RESET fest definierte Einsprungadressen im Programmspeicher, sogenannte Interruptvektoren.

Verarbeitung von Interrupts: Dazu sind eine Reihe von Registern zuständig:

	Details siehe Datenblatt Seite
General Interrupt MaSK Register GIMSK	23
General Interrupt Flag Register GIFR	24
Timer/Counter Interrupt MaSK Register TIMSK	24
Timer/Counter Interrupt Flag Register TIFR	25

3.5 Die Ruhezustände der CPU (Sleep – Modes)

uCs werden oft auch in batteriebetriebenen Geräten eingesetzt. => möglichst geringer Stromverbrauch erforderlich !
Zielerreichung durch : CMOS – Technologie
Geringe Versorgungsspannung
Geringstmögliche Taktfrequenz

Zusätzlich kann die CPU in Arbeitspausen in einen von zwei verschiedenen Schlafzustände versetzt werden: „Idle Mode“ und „Power Down Mode“. Details siehe Datenblatt Seiten 27 u. 28.

4. Die Ein/Ausgabe – Ports (I/O-Ports)

	AT90S1200	AT90S2313	AT90S4414	AT90S8515	AT90S8535
I/O-Port A	--	--	ja	ja	ja
I/O-Port B	ja	ja	ja	ja	ja
I/O-Port C	--	--	ja	ja	ja
I/O-Port D	ja (7 Bit)	ja (7 Bit)	ja	ja	ja

Alle Ports der AVR Basic Line sind 8 Bit breite, bidirektionale I/0- (Ein- / Ausgabe-) Ports. Eine Ausnahme stellt nur der 7 Bit breite Port D der Controller AT90S1200 und AT90S2313 dar. Hier musste auf das Bit PD7 verzichtet werden, da insgesamt nur 20 Gehäuseanschlüsse zur Verfügung stehen.

Jeder Pin eines Ports kann individuell als Eingang oder als Ausgang konfiguriert und in der Funktion als Eingang kann ihm wahlweise ein Pull-Up-Widerstand zugeschaltet werden.

Abb. 10.1 zeigt exemplarisch für einen Pin das Schema, nach dem die AVR-Ports aufgebaut sind. Das »X« in der Bezeichnung des Pins, des Datenrichtungs- und des Port-Registers steht als Platzhalter für die Bezeichnung des Ports, also »A«, »B«, »C« oder »D«. Die Nummer 0 ... 7 des Bits innerhalb des Registers wird durch »n« repräsentiert.

Die Schalterstellungen von S1 ... S3 in Abb. 10. 1 sind für Low-Pegel an den Registerausgängen DDXn und Portn gezeichnet.

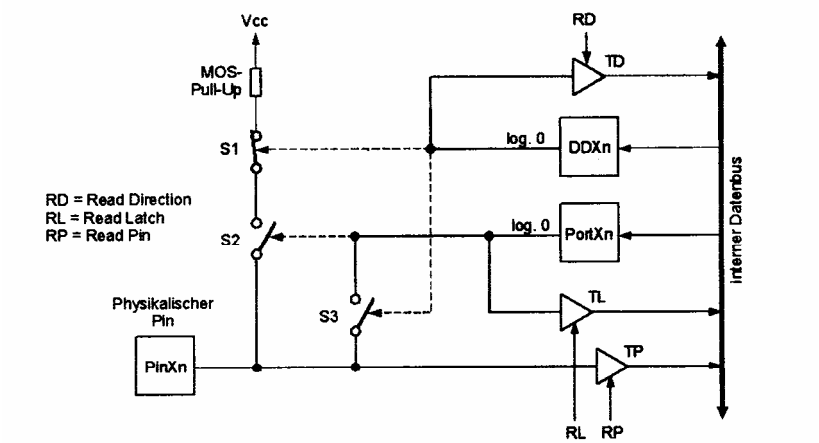


Abb. 10.1: Prinzipieller Aufbau eines I/O-Pins

Jeder Port wird über drei verschiedene Adressen im I/0-Registerbereich angesprochen:

1.Im Datenrichtungsregister (DDX in Abb. 10.1) wird für den Port festgelegt, ob es sich um einen Eingangs- oder einen Ausgangs-Pin handeln soll.

+ Hat das Bit n im Datenrichtungsregister Low-Pegel, so ist der entsprechende Pin als Eingang konfiguriert. Der Schalter S3 (Schließer) ist in diesem Fall offen und trennt PortXn von PinXn. Schalter S1 (Öffner) ist geschlossen und schaltet den Pull-Up-Widerstand an PinXn, falls S2 durch eine log. 1 am Ausgang von PortXn geschlossen ist.

+ Hat das Bit n im Datenrichtungsregister High-Pegel, so ist der entsprechende Pin als Ausgang konfiguriert. Der Schalter S3 (Schließer) ist in diesem Fall geschlossen und verbindet PortXn mit PinXn. Schalter S1 (Öffner) ist offen und trennt den Pull-Up-Widerstand von PinXn, unabhängig von der Stellung von S2 bzw. dem Wert am Ausgang von PortXn.

2. In das eigentliche Port-Register (PortX in Abb. 10.1) wird im Falle eines Ausgangs der auszugebende Wert eingeschrieben. Im Falle eines Eingangs kann über das Port-Register der Schalter S2 geschlossen und somit ein Pull-Up-Widerstand an den Eingangs-Pin geschaltet werden.

3. Der logische Pegel am Gehäuseanschluss PinXn des Ports kann über die dritte Adresse eingelesen werden. Die Steuerlogik erzeugt in diesem Fall das Lesesignal RP (Read Pin), das über den Treiber TP den Pin direkt auf den internen Datenbus schaltet.
Die nachfolgende Tabelle listet die möglichen Konfigurationen für Datenrichtung und Pull-Up-Widerstand eines Ports auf.

DDXn	PortXn	I/O	Pull-Up	Beschreibung
0	0	Input	Nein	Hochohmiger Eingang (Tri-State, Hi-Z)
0	1	Input	Ja	Der Pull-Up belastet die Eingangsquelle
1	0	Output	Nein	Gegentakt-Ausgang: log. 0
1	1	Output	Nein	Gegentakt-Ausgang: log. 1

Tab. 10.1: Konfigurationen für Datenrichtung und Pull-Up-Widerstand eines Ports

Die Datenrichtungsregister und die Port- Register können gelesen und geschrieben, die Hardware-Pins können nur gelesen werden. Um die Inhalte auszulesen, aktiviert die Steuerlogik die Signale RD (Read Direction), RL (Read Latch) bzw. RP (Read Pin), die den jeweiligen Wert über den entsprechenden Treiber TD, TL bzw. TP auf den internen Datenbus schalten.

Der **Ausgangstreiber** eines AVR-Ports ist bei Low-Pegel an seinem Ausgang in der Lage, in den Ausgang hineinfließende Ströme bis zu **20 mA aufzunehmen** (»Sink«) und dadurch z.B. gegen die Versorgungsspannung geschaltete LEDs direkt zu treiben.

Die meisten Pins werden alternativ auch für Sonderfunktionen verwendet, z.B. für Timer-, Interrupt-, Analogkomparator- Eingänge sowie diverse Ausgänge für Timer, SPI, UART u.a.m.

Diese Funktionen werden in der Beschreibung der jeweiligen Module erklärt.

Je PORT sind 3 Register im I/O-Bereich vorgesehen:

	Read/Write	nach Reset
Datenregister PORTX	R/W	0x00
Datenrichtungsregister DDRX	R/W	0x00
Adresse des Port-Anschlusses PinX	R	Hi-Z

Das heißt nach dem Reset des µC sind alle Portleitungen auf den Ausgangszustand „Hochohmiger Eingang“ zurückgesetzt !

Quellenangabe:
Dieses Skriptum entstand durch eine komprimierte Darstellung der Inhalte des Buches AVR-RISC Mikrocontroller von Wolfgang Trampert aus dem Francis’ Verlag.