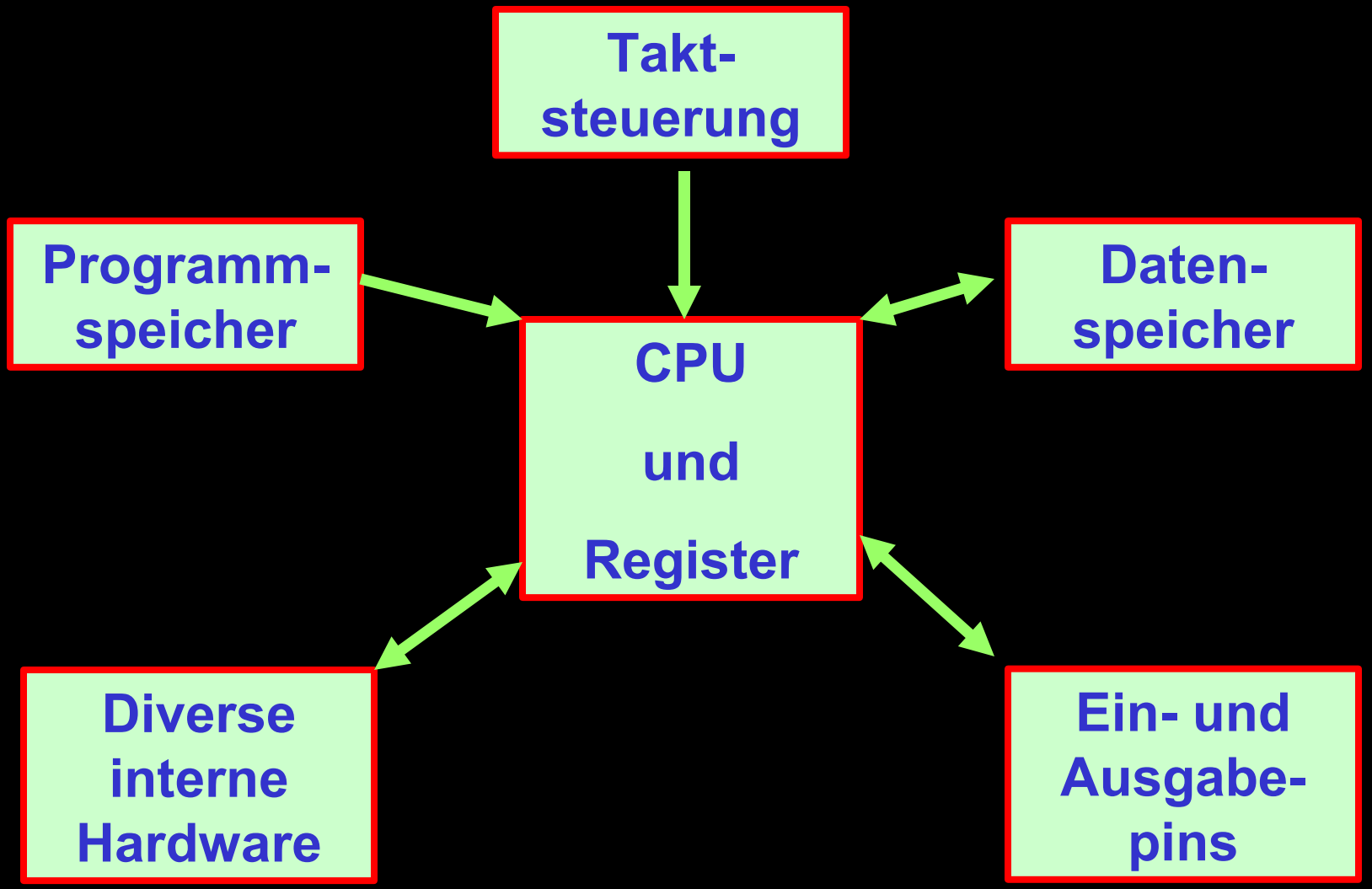
Programmierung von ATMEL AVR Mikroprozessoren am Beispiel des ATtiny13

Eine Einführung in Aufbau, Funktionsweise, Programmierung und Nutzen von Mikroprozessoren

Teil III: Wat macht ene Mikrokontroller?

Interner Aufbau eines Mikrokontrollers

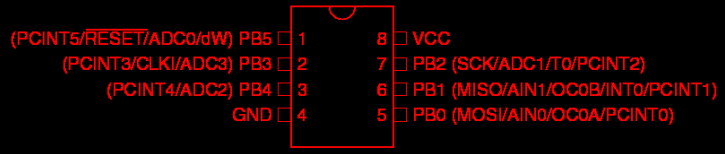


Typen und Eigenschaften von ATMEL AVR

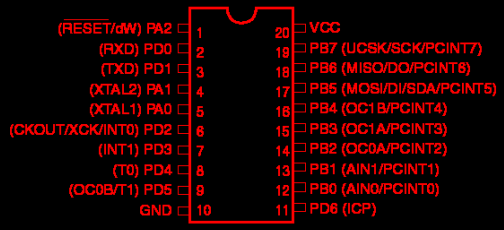
Device	Flash kB	EEPROM Byte	SRAM Byte	I/O Pins	F.max MHz	Ub V	16-bit Timer	8-bit Timer	PWM	UART	A/D 10-Bit	Ana Comp	Interrupts rupts	Pin Dip
ATtiny12	1	64	--	6	8	1.8-5.5	--	1	--	--	--	Yes	5	8
ATtiny13	1	64	64	6	20	1.8-5.5	--	1	2	--	4	Yes	9	8
ATtiny15L	1	64	--	6	1,6	2.7-5.5	--	2	1	--	4	Yes	8	8
ATtiny25	2	128	128	6	20	1.8-5.5	--	2	4	--	4	Yes	15	20
ATtiny26	2	128	128	16	16	2.7-5.5	--	2	2	--	11	Yes	11	32
ATtiny28L	2	--	32	11	4	1.8-5.5	--	1	--	--	--	Yes	5	28
ATtiny45	4	256	256	6	20	1.8-5.5	--	2	4	--	4	Yes	15	20
ATtiny85	8	512	512	6	20	1.8-5.5	--	2	4	--	4	Yes	15	20
ATtiny2313	2	128	128	18	20	1.8-5.5	1	1	4	1	--	Yes	8	20
ATmega8	8	512	1024	23	16	2.7-5.5	1	2	3	1	8	Yes	18	28
ATmega128	128	4096	4096	53	16	2.7-5.5	2	2	8	2	8	Yes	34	64
ATmega16	16	512	1024	32	16	2.7-5.5	1	2	4	1	8	Yes	20	40
ATmega162	16	512	1024	35	16	1.8-5.5	2	2	6	2	--	Yes	28	40
ATmega164	16	512	1024	32	20	1.8-5.5	1	2	6	2	8	Yes	31	40
ATmega165	16	512	1024	54	16	1.8-5.5	1	2	4	1	8	Yes	23	64
ATmega168	16	512	1024	23	20	1.8-5.5	1	2	6	1	8	Yes	26	28
ATmega169	16	512	1024	53	16	1.8-5.5	1	2	4	1	8	Yes	23	64
ATmega2560	256	4096	8192	86	16	1.8-5.5	4	2	16	4	16	Yes	57	100
ATmega32	32	1024	2048	32	16	2.7-5.5	1	2	4	1	8	Yes	19	40
ATmega406	40	512	2048	18	1	1.4-5,5	1	1	1	--	--	Yes	23	48
ATmega48	4	256	512	23	20	1.8-5.5	1	2	6	1	8	Yes	26	32
ATmega64	64	2048	4096	53	16	2.7-5.5	2	2	8	2	8	Yes	34	64
ATmega649	64	2048	4096	53	16	1.8-5.5	1	2	4	1	8	Yes	25	64
ATmega88	8	512	1024	23	20	1.8-5.5	1	2	6	1	8	Yes	26	32
ATmega8515	8	512	512	35	16	2.7-5.5	1	1	3	1	--	--	16	40
ATmega8535	8	512	512	32	16	2.7-5.5	1	2	4	1	8	Yes	20	40

Liste ist im Original doppelt so lang! Da ist für jeden was dabei!

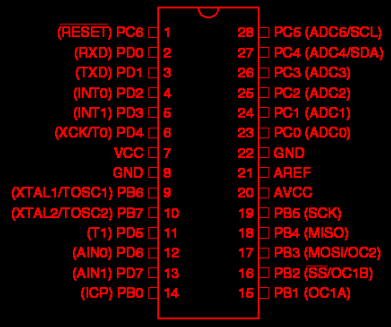
Typenvergleich



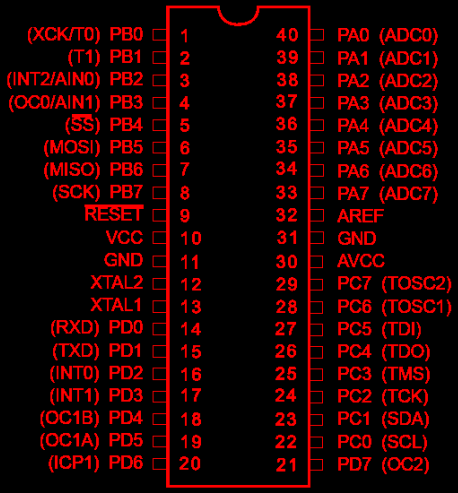
ATtiny13



ATtiny2313



ATmega8

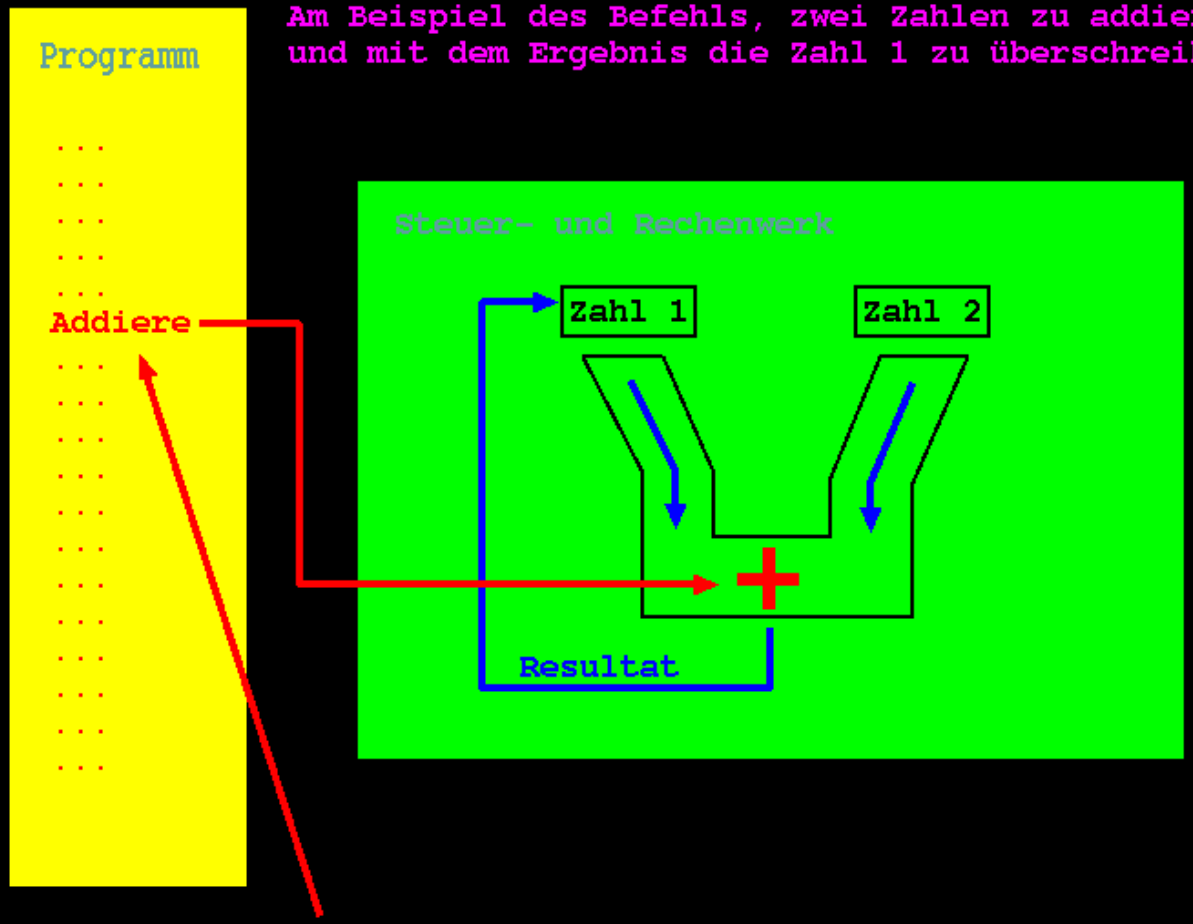


ATmega16

Befehlsablauf

Was ist ein Rechner (engl.: Computer)?

Am Beispiel des Befehls, zwei Zahlen zu addieren
und mit dem Ergebnis die Zahl 1 zu überschreiben



In AVR-Assemblersprache: ADD R1,R2
In AVR-Maschinensprache: 0000.1100.0001.0010

Befehlsablauf im Einzelnen

- 
- Taktsignal:

- Befehl bearbeiten: | Programmzähler an Programmspeicher ausgeben
  | Befehlswort aus Programmspeicher auslesen
  | Befehlswort dekodieren
  | Befehl ADD ausführen
  | Ergebnis speichern
  Fertig |

- Befehl braucht zwei Takte zur Bearbeitung: Takt 1 liest aus Programmspeicher, Takt 2 führt den Befehl aus, @4 MHz: 0,5 µs pro Befehl
- Pre-Fetch bei ATMEL-AVRs:
  Während der Ausführung des vorausgehenden Befehls wird bereits der nächste Befehl geholt; falls kein Sprung erfolgt, kann der nächste Befehl direkt bearbeitet werden; Konsequenzen: doppelte Ausführungs-geschwindigkeit, @4 MHz: 0,25 µs pro Befehl, Befehle ohne Sprung brauchen nur einen Takt, mit Sprung zwei Takte
- Der Zeitbedarf für jeden Befehl ist exakt vorhersehbar
- Alle Abläufe lassen sich exakt zeitlich planen und ausführen
- Im PC nahezu unmöglich, da Multitasking-Betriebssysteme darüber entscheiden, welches Programm wann wieviel Zeit bekommt!

Befehle, Instruktionen (Assembler *Mnemomonic*)

- **Die wichtigsten Befehle/Instruktionen beim ATtiny13**
  - **Rechnen: Addieren (*Add*, *Adc*) und Subtrahieren (*Sub*, *Sbc*) ohne und mit Übertrag, Vergleichen mit Konstante (*Cpi*) / Register (*Cp*) / Register und Übertrag (*Cpc*), Um eins erhöhen (*Inc*) oder vermindern (*Dec*), Logisches Und (*And*) / Oder (*Or*) / Exklusiv-Oder (*Eor*), Bits auf 0 (*Cbr*) oder 1 (*Sbr*) setzen, Register auf Festwert setzen (*Ldi*) / auf Null setzen (*Clr*) / auf 255 (*Ser*) setzen**
  - **Bit und Bit-Test: Bit im Port auf Eins (*Sbi*) oder Null (*Cbi)* setzen, Logisches Links-Schieben (*Lsl*) oder Rechtsschieben (*Lsr*), Links- (*Rol*) oder Rechts- (*Ror)* Schieben über Carry, Arithmetisches Rechtsschieben (*Asr*), Unteres und oberes Nibble tauschen (*Swap*), Flaggbit setzen (*Set*) / rücksetzen (*Clt*) / auf Registerbit setzen (*Bst*) / in Register laden (*Bld*), Flags im Statusregister setzen (*Sex*) oder löschen (*Clx*) mit x={Z,C,N,V,S,H,T,I}, Einer- (*Com*) und Zweier-(*Neg*) Komplement**
  - **Sprünge: Relativer (*Rjmp*) / Indirekter (*Ijmp*) Sprung, Relativer (*Rcall*) / Indirekter (*Icall*) Unterprogrammaufruf, Relativsprung bei gesetztem (*Brxs*) / gelöschtem (*Brxc*) Statusbit mit x={Z,C,N,V,S,H,T,I}, Überspringen bei gesetztem (Sbxs) oder gelöschtem (*Sbxc*) Bit mit {x=r Register, x=i I/O-Port)**
  - **Datentransfer: Register zu Register (*Mov*), Registerwort zu Registerwort (*Movw*), Speicher zu Register direkt (*Ld*) / indirekt (*Ldd*) / Festadresse (*Lds*), Register in Speicher direkt (*St*) / indirekt (*Std*) / Festadresse (*Sts*), Programmspeicher in Register (*Lpm*), Register in I/O-Port (*Out*), I/O-Port in Register (*In*), Register auf Stapel (*Push*) und vom Stapel (*Pop*)**
  - **CPU-Kontrolle: Nichtstun (*Nop*), Schlafen (*Sleep*), Wachhund rücksetzen (*Wdr*), Debug-Unterbrechung (*Break*)**
- **Jeder Befehl belegt ein Wort (=2 Byte) im Programmspeicher.**
- **Die meisten Instruktionen benötigen einen einzigen Takt für die Ausführung, Sprungbefehle zwei Takte.**

Zusammenfassung

- Beim Abarbeiten von Befehlen/Instruktionen im Mikrokontroller geht es streng geordnet und überaus berechenbar zu.
- Scheffe ist CPU; Hoflieferant der Programmspeicher; Höflinge mit Vitamin B und direktem Zugang zum Chef sind die 32 Register; Indianer sind die Speicher und diverse interne Gerätschaften. Für die Außenpolitik sorgen externe Pins, die geruhen, etwas ein- oder auszugeben.
- Alle Befehle/Instruktionen werden nacheinander bearbeitet (konsekutiv), ein Nebeneinander verschiedener Aufgaben kann und darf es auch nicht geben.
- Den Takt der Verarbeitungsmusik gibt der Taktgeber vor, die aktuelle Bearbeitungsadresse steht immer im Programmzähler (program counter).
- Nach außen hin herrscht absolute Ruhe! Kein Pin gibt interne Adressen oder Daten nach außen, es sei denn, es wird so angewiesen! Unterschied zu PC und MC!
- AVRs holen schon mal den nächsten Befehl, während sie den letzten noch gar nicht richtig bearbeitet haben (Pre-Fetch).
- Wildes Umherspringen im Programmablauf wird deshalb mit Strafzeiten von mindestens einer Taktlänge geahndet.
- Es gibt mehr als 100 verschiedene Befehle, die alle was anderes bewirken. Kein Mensch kann die auswendig lernen und sich merken, was die anstellen, wie lange sie brauchen, welche Flaggen sie wann setzen oder in Ruhe lassen. Für vergessliche Menschen gibt es das vollständige Instruction Set Summary am Ende jedes Datenblattes, für umme bei http://www.atmel.com, Microprozessors, 8-Bit-RISC. Da steht alles drin.