Einführung in die Technische Informatik

Prof. Dr.-Ing. Stefan Kowalewski

WS 22/23

Kapitel 13: Mikrocontroller

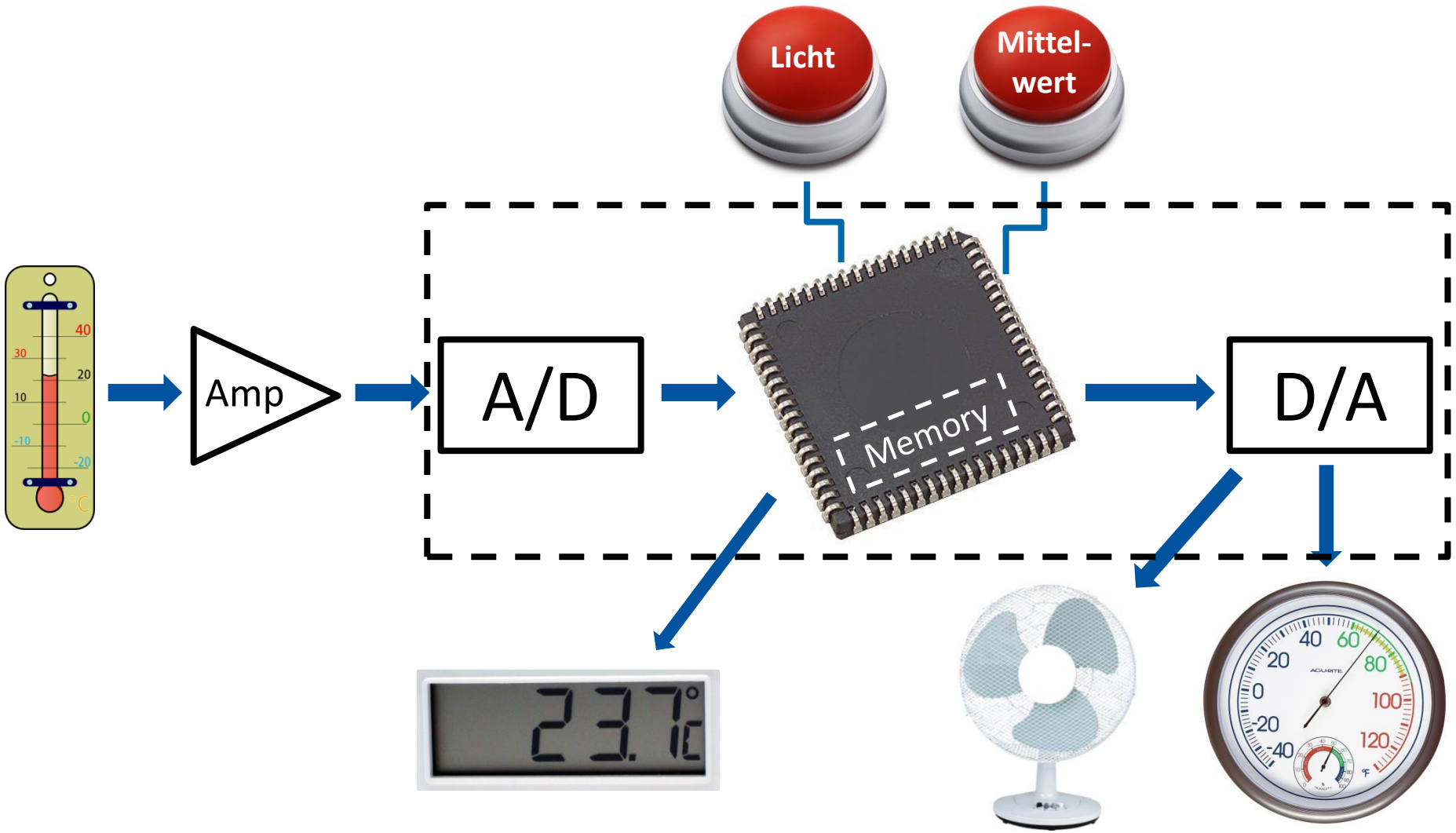
Informatik 11
Embedded Software

RWTH AACHEN UNIVERSITY

Abschnitt 13.1

Aufbau

Anwendungsbeispiel



Was ist der Unterschied zwischen Mikroprozessoren und Mikrocontrollern?

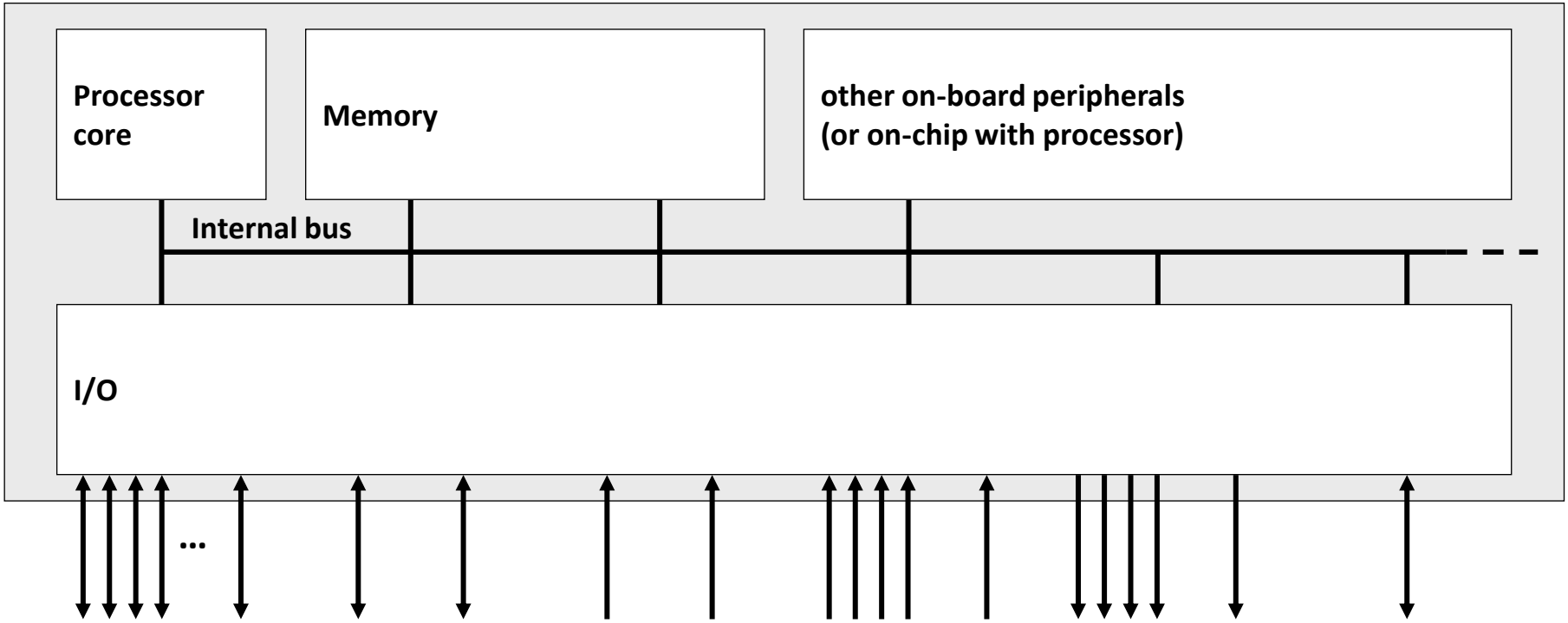
- Mikrocontroller:
  - Stand-alone Device für Anwendungen in eingebetteten Systemen
  - Low-end Mikroprozessor + Speicher + I/O + zusätzliche Komponenten
  - Kein universelles Gerät
  - Kostenoptimierte Steuerungseinheit für bestimmte Anwendungsfelder
- Verglichen mit High-end-Mikroprozessoren:
  - Häufig kein Pipelining, Cache, etc.
  - Vorhersagbares Zeitverhalten

Erinnerung: Rechnerarchitektur

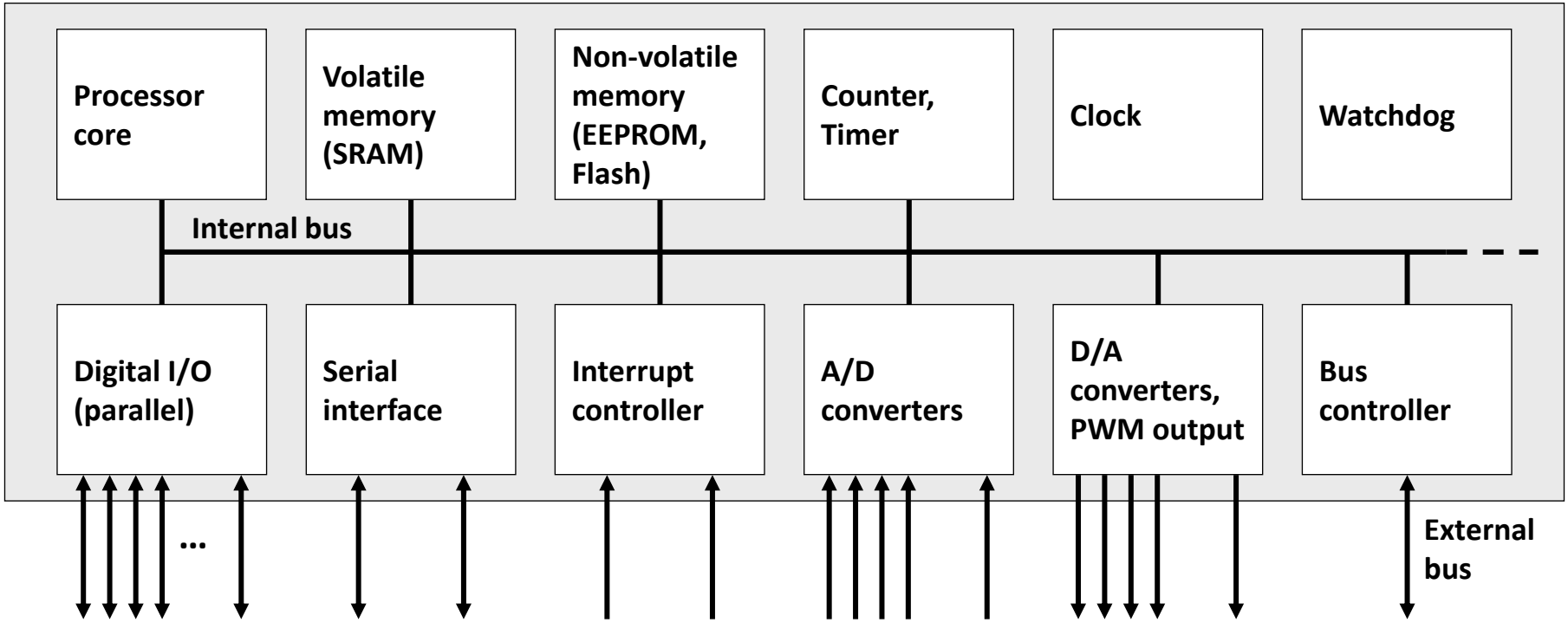
- RISC: Reduzierter Befehlssatz
- RISC-Architekturen sind Load/Store-Architekturen, d.h. Daten können nur über Register in den Speicher geschrieben oder aus dem Speicher gelesen werden.
- Harvard-Architektur: Datenspeicher und Programmspeicher sind voneinander getrennt

- Ab hier: Betrachtung des Atmel ATmega 16
  - ein RISC-Mikrocontroller
  - mit Harvard-Architektur
  - 16 KB Flash Speicher
  - 8-Bit-Wortbreite

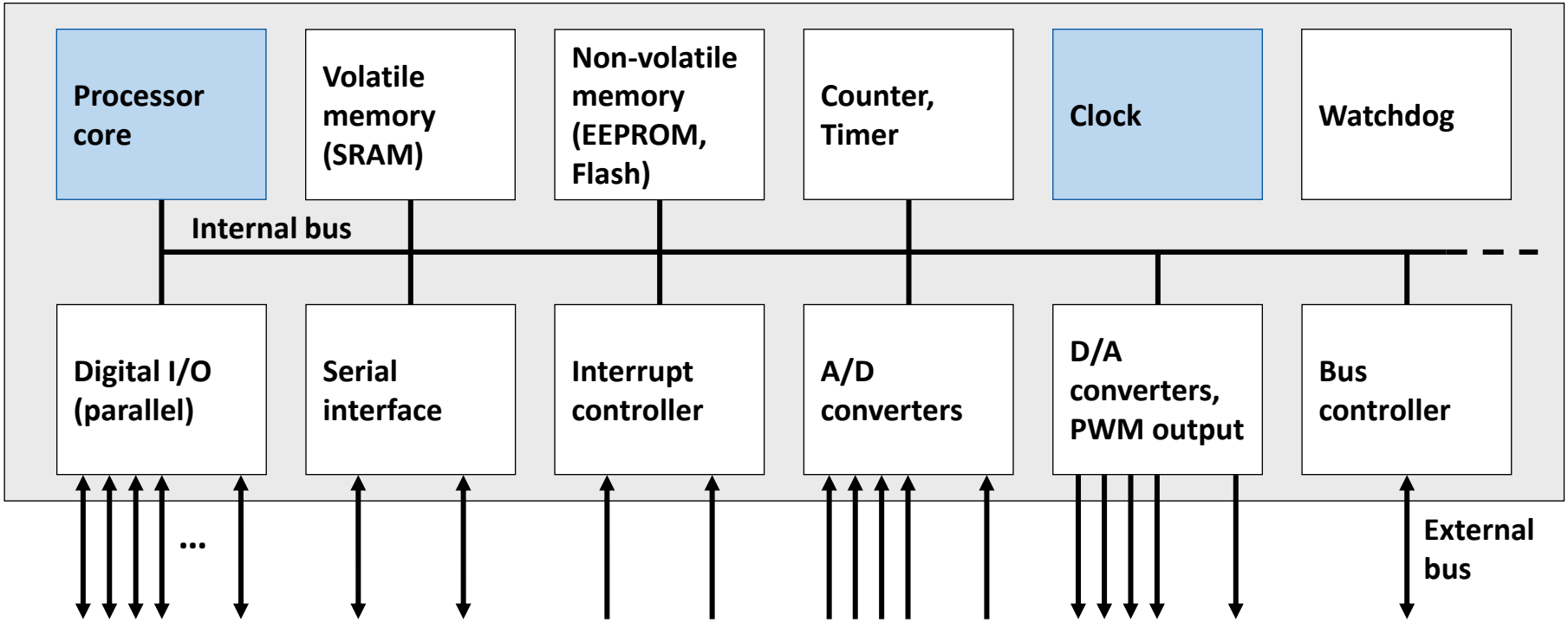
Basis-Struktur eines Mikrocontrollers



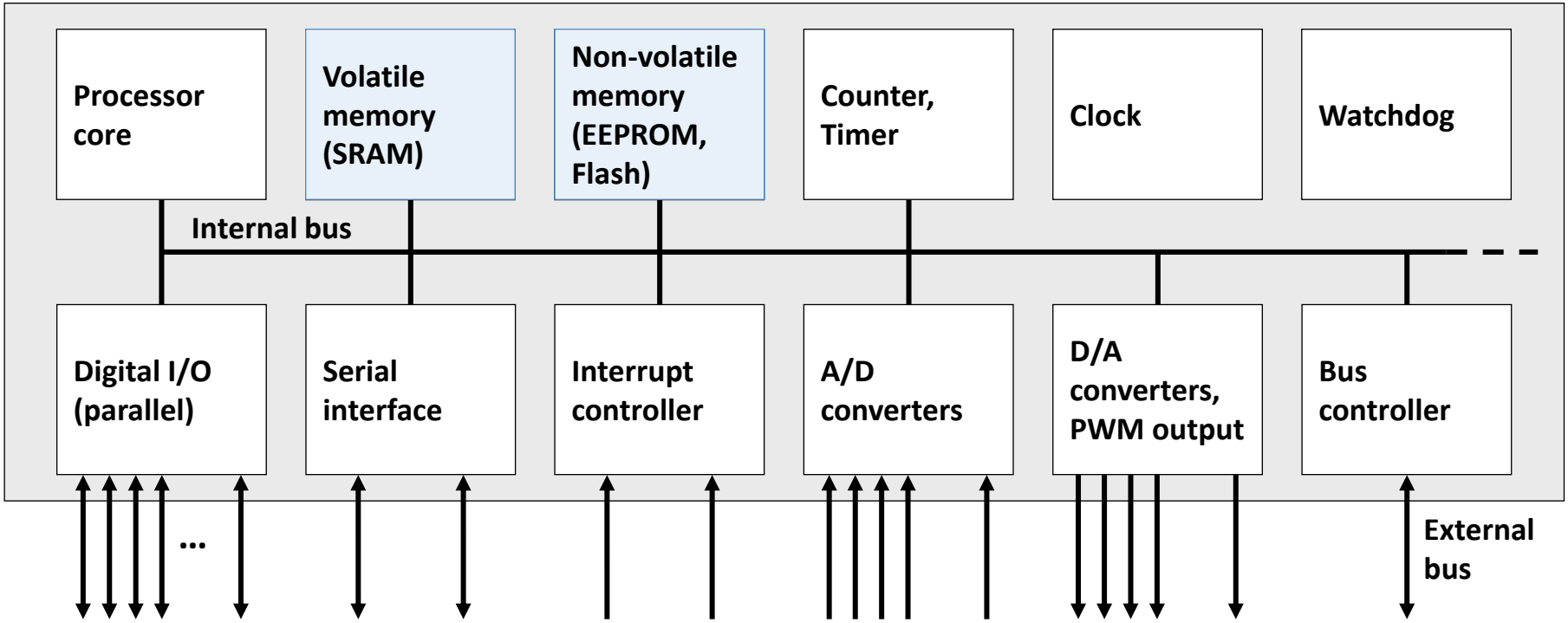
Basis-Struktur eines Mikrocontrollers - verfeinert -



Basis-Struktur eines Mikrocontrollers
- Kern -



Basis-Struktur eines Mikrocontrollers
- Speicher -

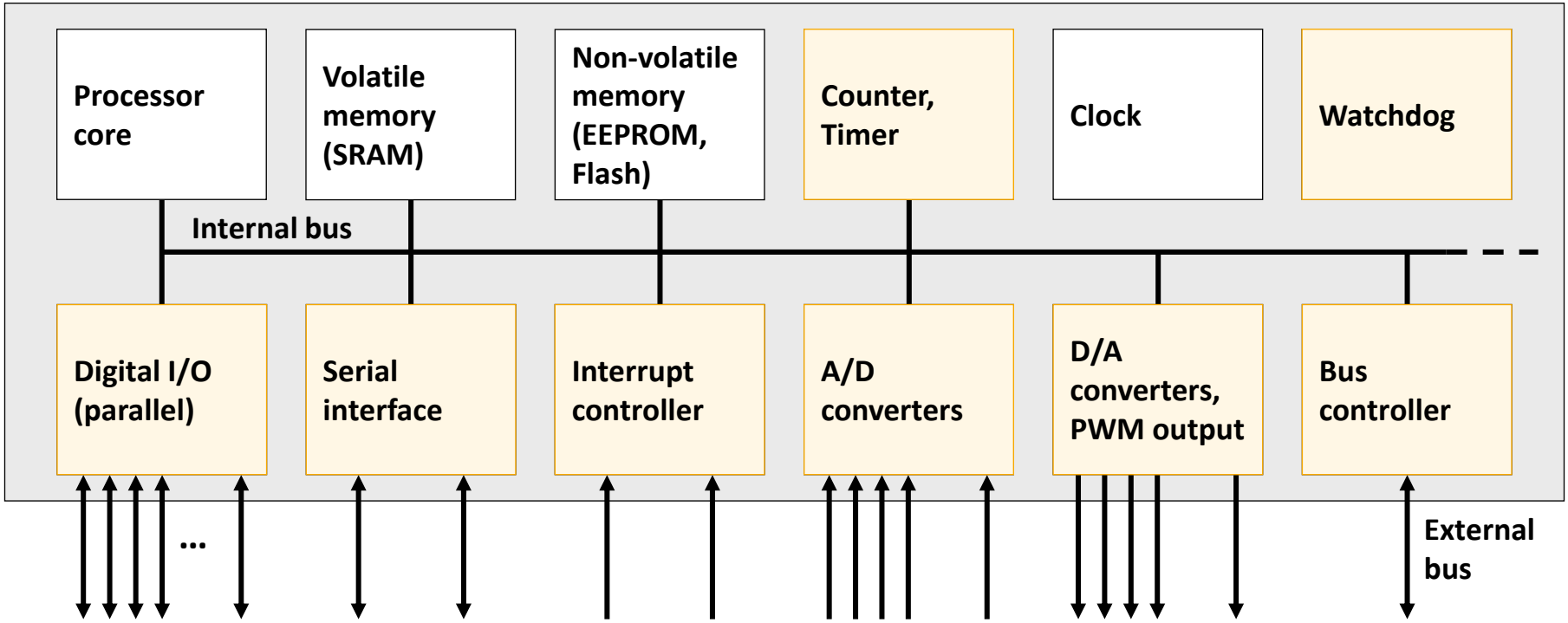


Speicher: - haben eigenen Adressbereich (Flash, EEPROM)

- teilen sich einen Adressbereich (Register, I/O-Register, SRAM)

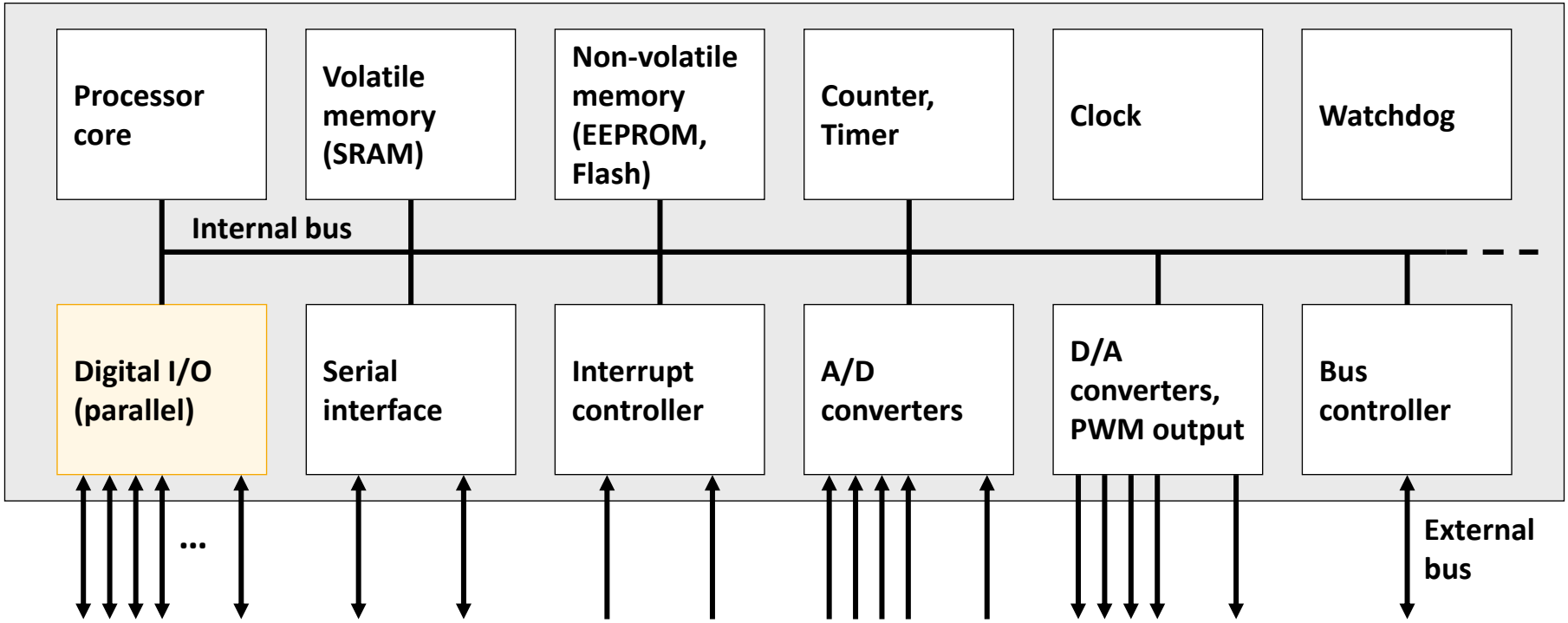
(Wenn Sie C benutzen, wird vieles vom Compiler übernommen.)

Basis-Struktur eines Mikrocontrollers
- Digital I/O und On-chip Peripherie -



Digital I/O und On-Chip Peripherie werden durch zweckgebundene Register (I/O-Register) angesteuert.

Basis-Struktur eines Mikrocontrollers
- Digital I/O -



Abschnitt 13.2

Digital I/O

Digitale I/O-Anschlüsse

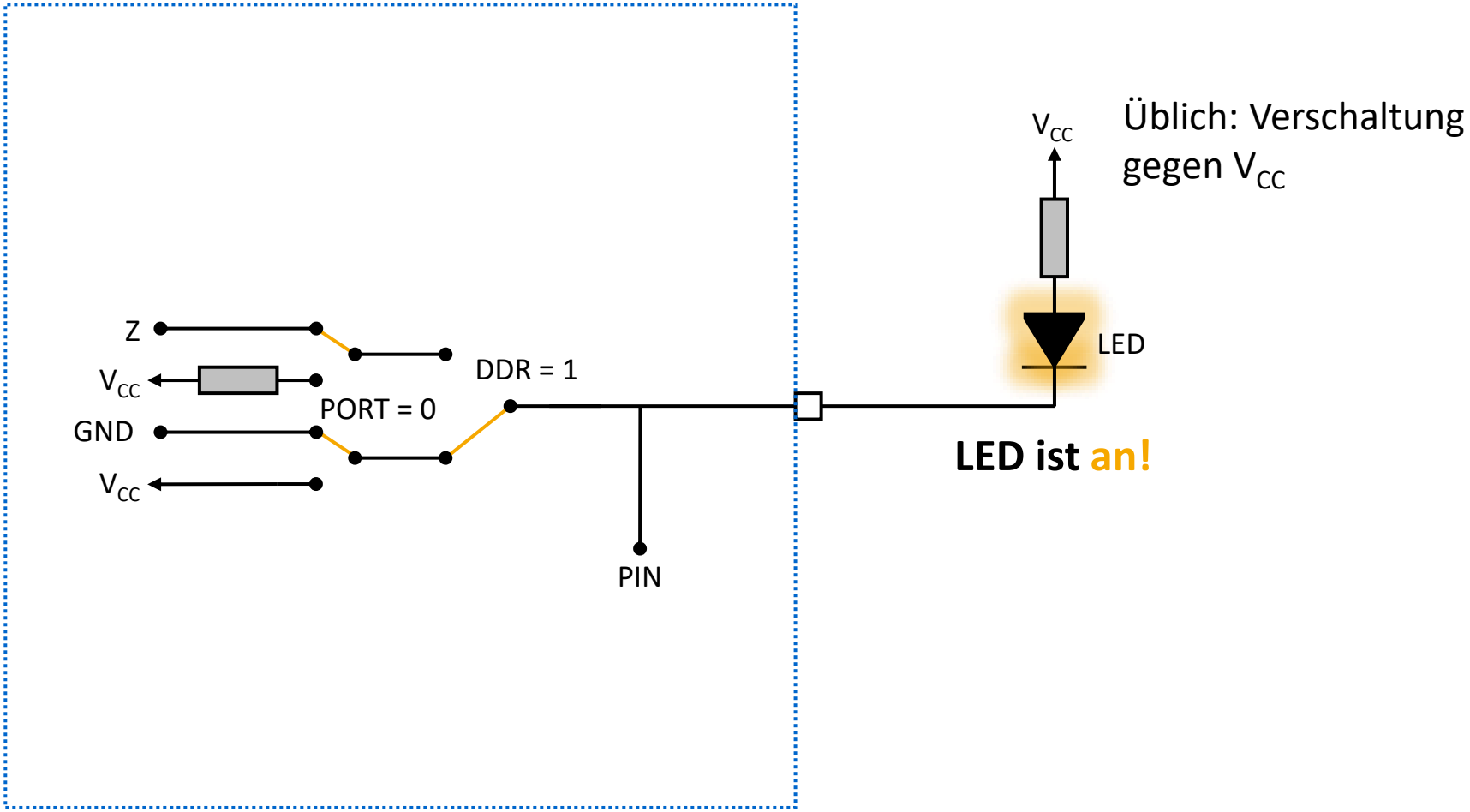
- Digitale I/O Anschlüsse
  - Werden zu **Ports** von 8 Anschlüssen zusammengefasst (Byte orientierter Zugriff).
  - Sind **bidirektional** (d.h. sie können als Ein- und Ausgang genutzt werden).
  - Können wechselnde Funktionen haben
    - Direkte Verwendung
    - Übernahme durch On-Chip-Peripherie: „port capturing“

- Zugriff, Steuerung und Überwachung von digitalen I/O Anschlüssen wird durch drei zweckgebundene Register für jeden Port einzeln realisiert:
  - **Data Direction Register** (DDR) → Eingang (0) oder Ausgang (1)
  - **Port Register** (PORT) → Ausgabewert
  - **Port Input Register** (PIN) → Eingabewert

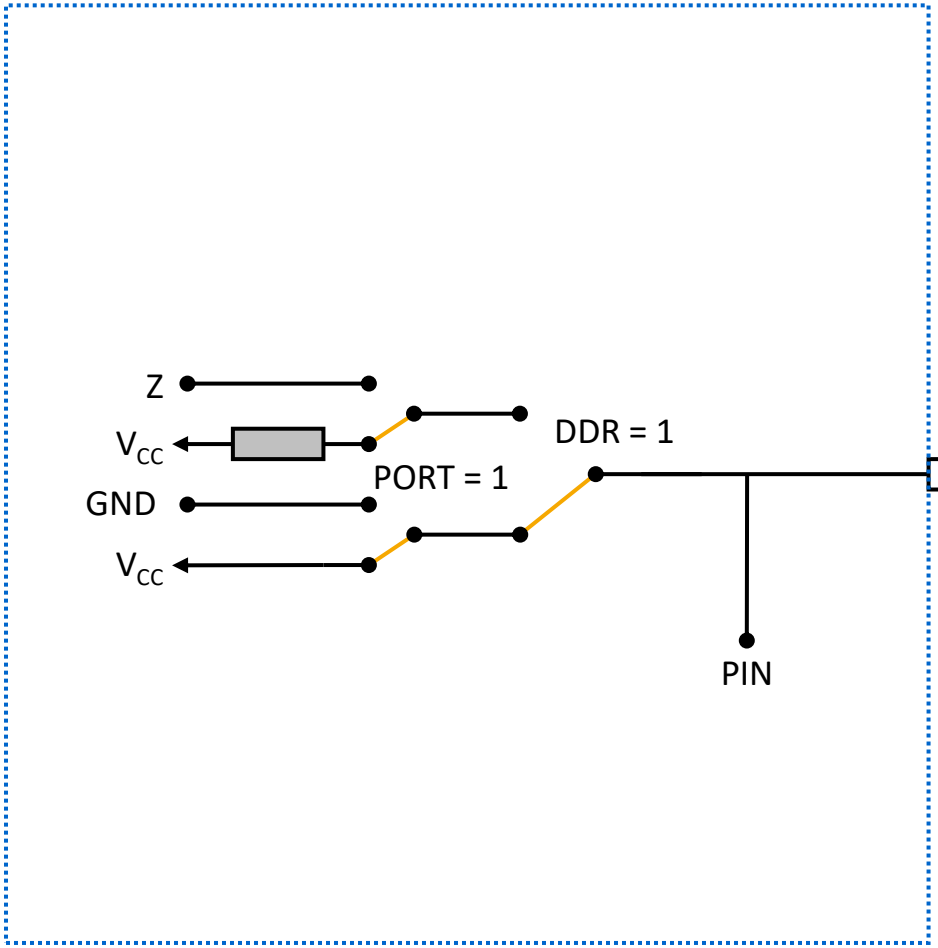
Steuerung der I/O Anschlüsse durch Register

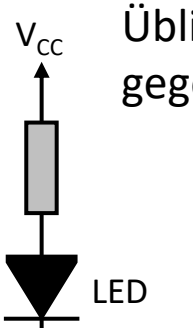
- Data Direction Register (DDR):
  - Lesen/Schreiben
  - Legt für jedes Bit des betrachteten Ports fest, ob es ein **Ein- oder Ausgangsbit** ist
- Port Register (PORT):
  - Lesen/Schreiben
  - Legt für **Ausgangspins** fest, ob der Ausgangswert **high (1)** oder **low (0)** ist
  - Legt für **Eingangspins** fest, ob Pull-up Widerstände eingeschaltet sind
- Port Input Register (PIN):
  - **Nur Lesen**
  - Enthält die aktuellen Werte (high oder low) aller Pins (input und output)
  - Übliche Verwendung: Einlesen der Werte der **Eingangspins**
  - **Achtung:** manche Chips erlauben Schreibzugriffe, aber mit unintuitiver Semantik!

Beispiel: Einschalten einer LED



Beispiel: Ausschalten einer LED





Üblich: Verschaltung gegen V_{CC}

LED ist aus!

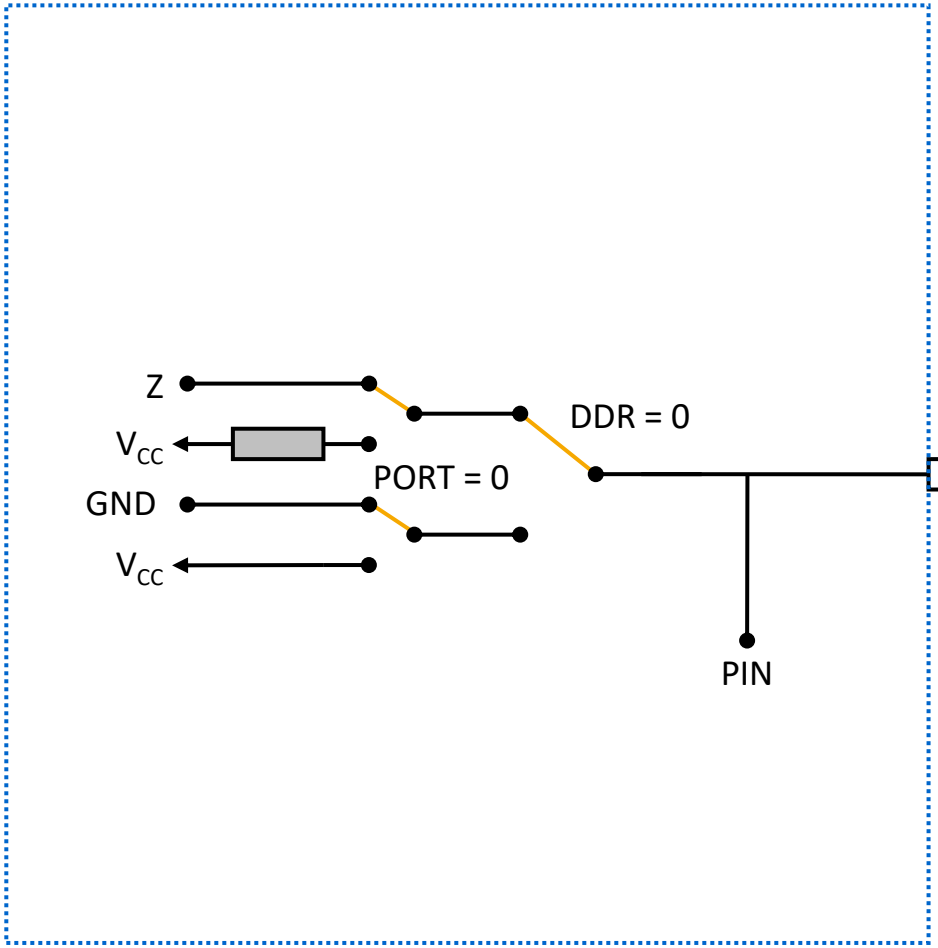
Digitaler Ausgang

- Wenn das DDR-Bit eines Pins als Ausgang gesetzt wird, dann betreibt der Controller den Pin unter Berücksichtigung des korrespondierenden PORT-Bit.
- Der Controller verwendet die korrespondierende Spannung.

- Der max. fließende Strom ist von dem externen Stromkreis abhängig (kann kurzgeschlossen sein).

$\Rightarrow$ Externe Strombegrenzung ist notwendig.

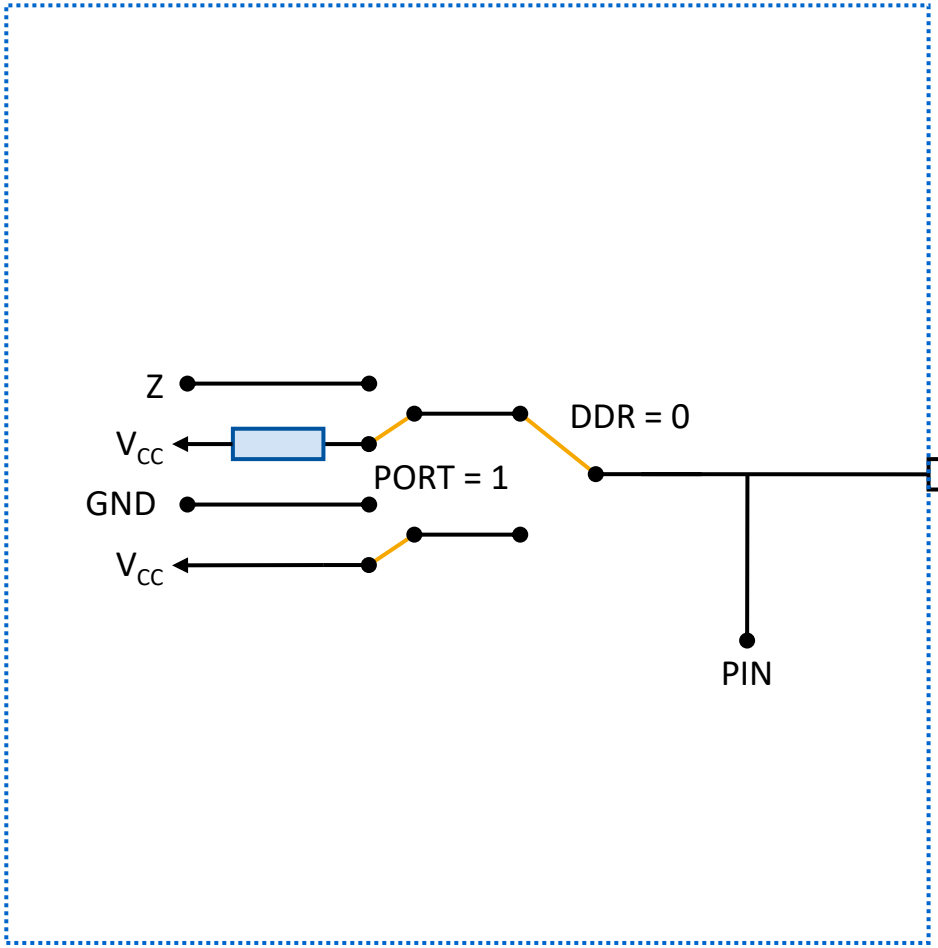
Beispiel: Lesen des Eingangs



Üblich: Verschaltung gegen GND

PIN ist undefiniert, wenn S1 offen ist!

Beispiel: Lesen des Eingangs



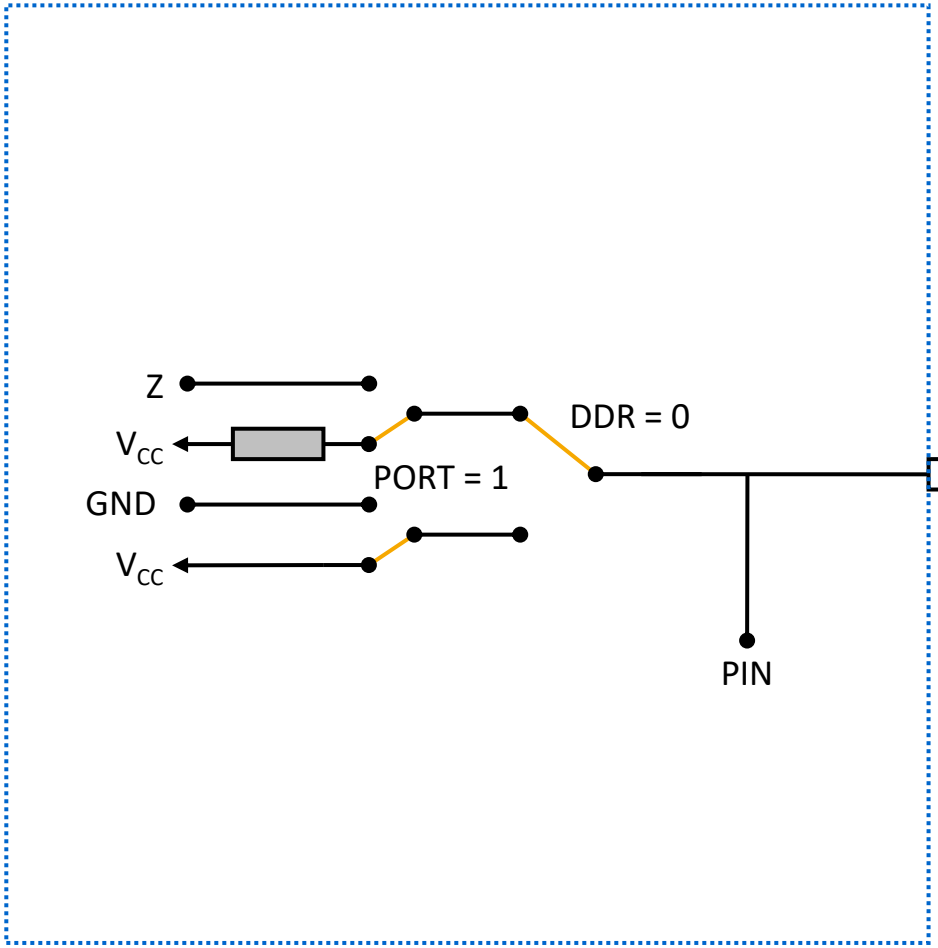
Üblich: Verschaltung gegen GND

PIN ist jetzt definiert, wenn S1 offen ist!

Pull-up Widerstand

Weiter verbreitet als Pull-down Widerstände

Beispiel: Lesen des Eingangs

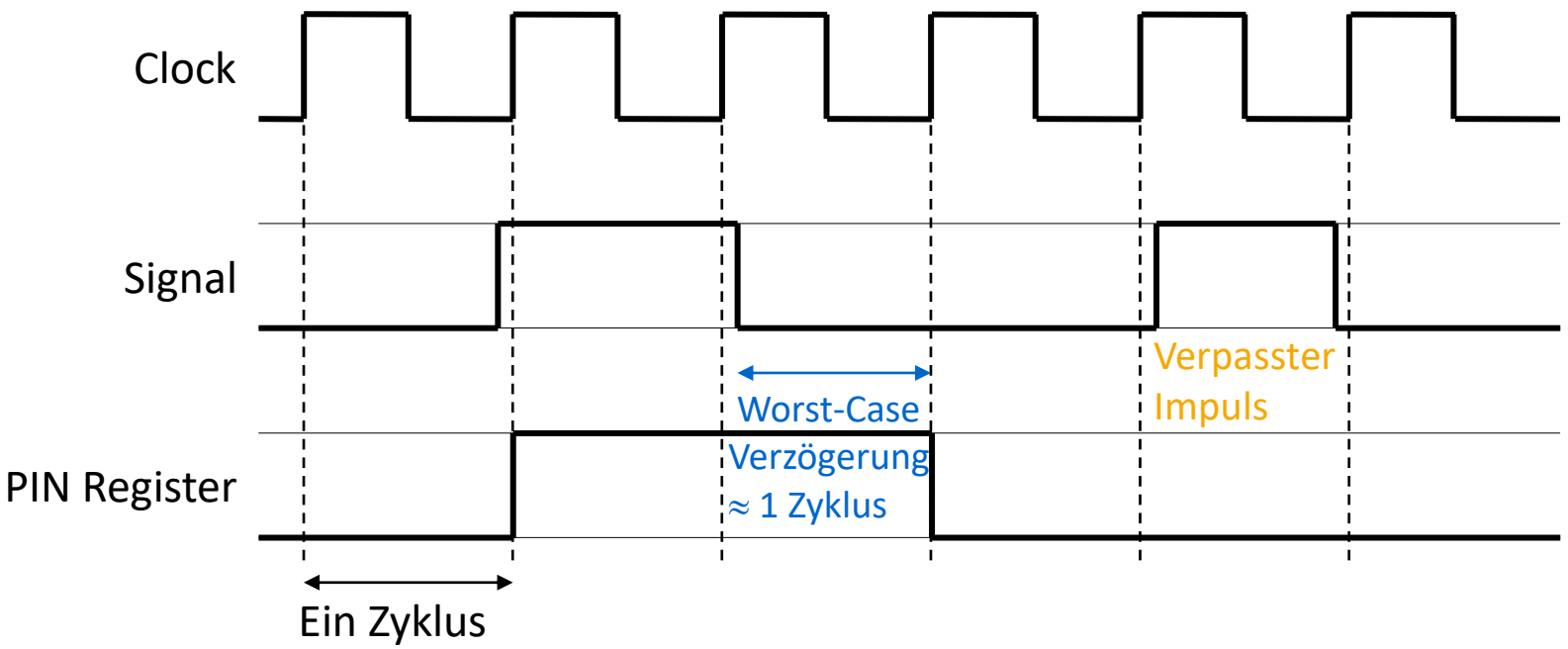


Üblich: Verschaltung gegen GND

An PIN liegen 0 Volt an

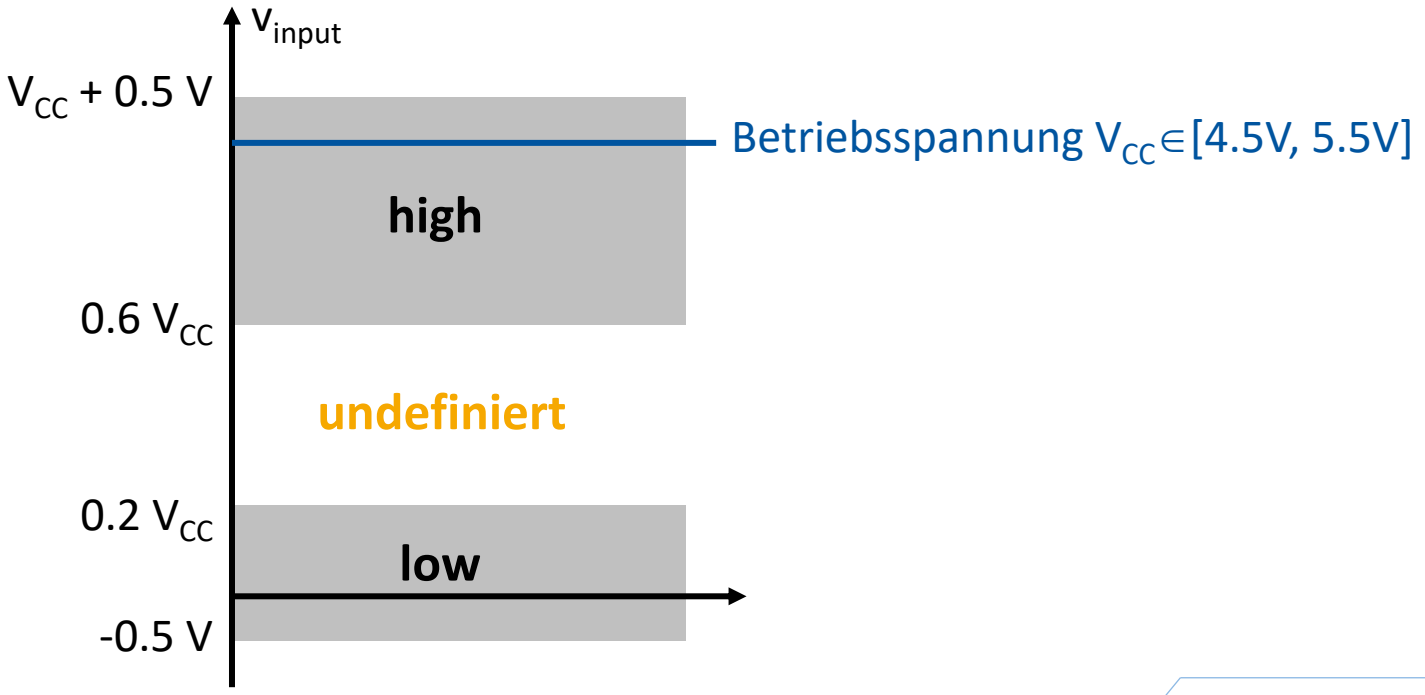
Digitaler Eingang: Sampling

- Sampling (engl. für Abtastung) innerhalb eines Clock-Zyklus erzeugt eine Verzögerung von ~ 1 Clock-Zyklus im Worst-Case.
- Impulse, die kürzer als ein Clock-Zyklus sind, bleiben eventuell unentdeckt.

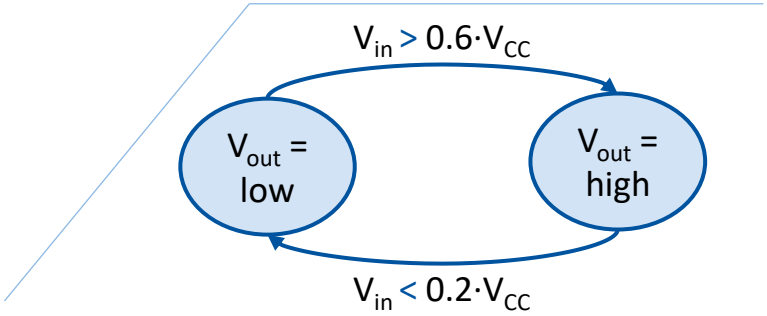


Digitaler Eingang

- Problem: Signale haben nicht immer ein genau definiertes Niveau.

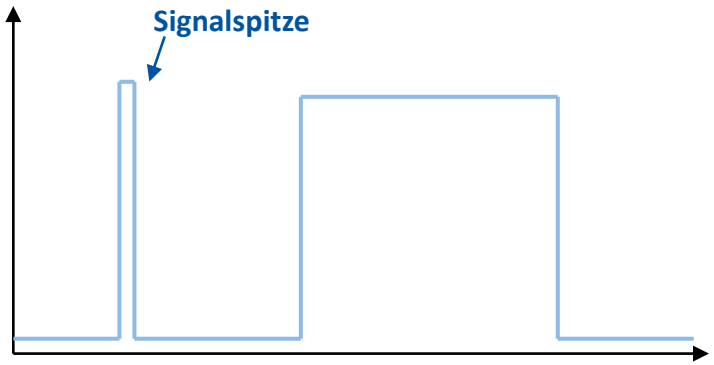


- Lösung: Schmitt-Trigger:

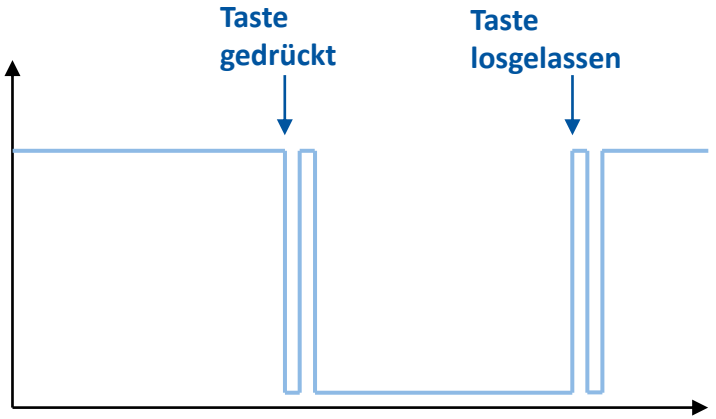


Digitaler Eingang

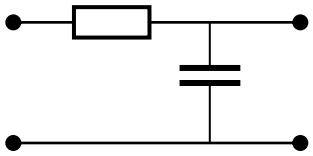
Störung



Prellen



- Lösungen:
  - durch Hardware: a) Tiefpass Filter (Kap. 5.3)
    b) Eingebaute Rauschunterdrückung



  - durch Software: Mehrmaliges Einlesen der Signale (Kap. 12.7)

Abschnitt 13.3

Interrupts

Warum Interrupts?

- Mikrocontroller müssen auf interne bzw. externe Ereignisse reagieren.
- Wie kann jedoch eine zeitlich passende Reaktion gesichert werden?

1. Polling
   - Periodische Prüfung auf Ereignisse
   - Nachteile:
     - Verschwendete CPU-Kapazitäten bei seltenen Ereignissen
     - Pollingsequenz muss in den restlichen Programmcode eingefügt werden (schwer zu modifizieren oder zu erweitern)
     - Verlängerte Reaktionszeit
     - Unvereinbar mit Sleep-Mode
   - Aber: es gibt auch Szenarien, in denen Polling die bessere Wahl ist

Warum Interrupts?

2. Interrupts
   - Bei Auftritt eines Signals wird die Programmausführung unterbrochen.
   - Die MCU ruft eine Interrupt Service Routine (ISR) zur Behandlung des Ereignisses auf.
   - Ähnlichkeiten zu Multitasking, aber keine parallele Ausführung.
   - Benötigt spezielle Hardware
   - Kann Polling verbessern

Interrupt-Steuerung

- Zur Nutzung von Interrupts müssen diese durch Modifizierung der betreffenden Steuerregister aktiviert werden.

- Üblicherweise gibt es
  - eine globale Aktivierungsmöglichkeit für alle Interrupts (Global IE) und
  - eine individuelle Aktivierungsmöglichkeit.

- Mit der Interrupt-Vektor-Tabelle wird die Zuordnung zwischen Interrupts und korrespondierender ISR festgelegt.

Interrupt-Steuerung

Vektor Nr.	Quelle	Programmadresse
1	Reset	$0000
2	External Interrupt 0	$0004
3	External Interrupt 1	$0008
…	…	…

- Ein Sprungbefehl zur passenden ISR muss an der jeweiligen Programmadresse eingefügt werden.
- Leere Vektoren sollten Sprungbefehl in Endlosschleife enthalten
- Vektortabelle kann an unterschiedlichen Stellen stehen

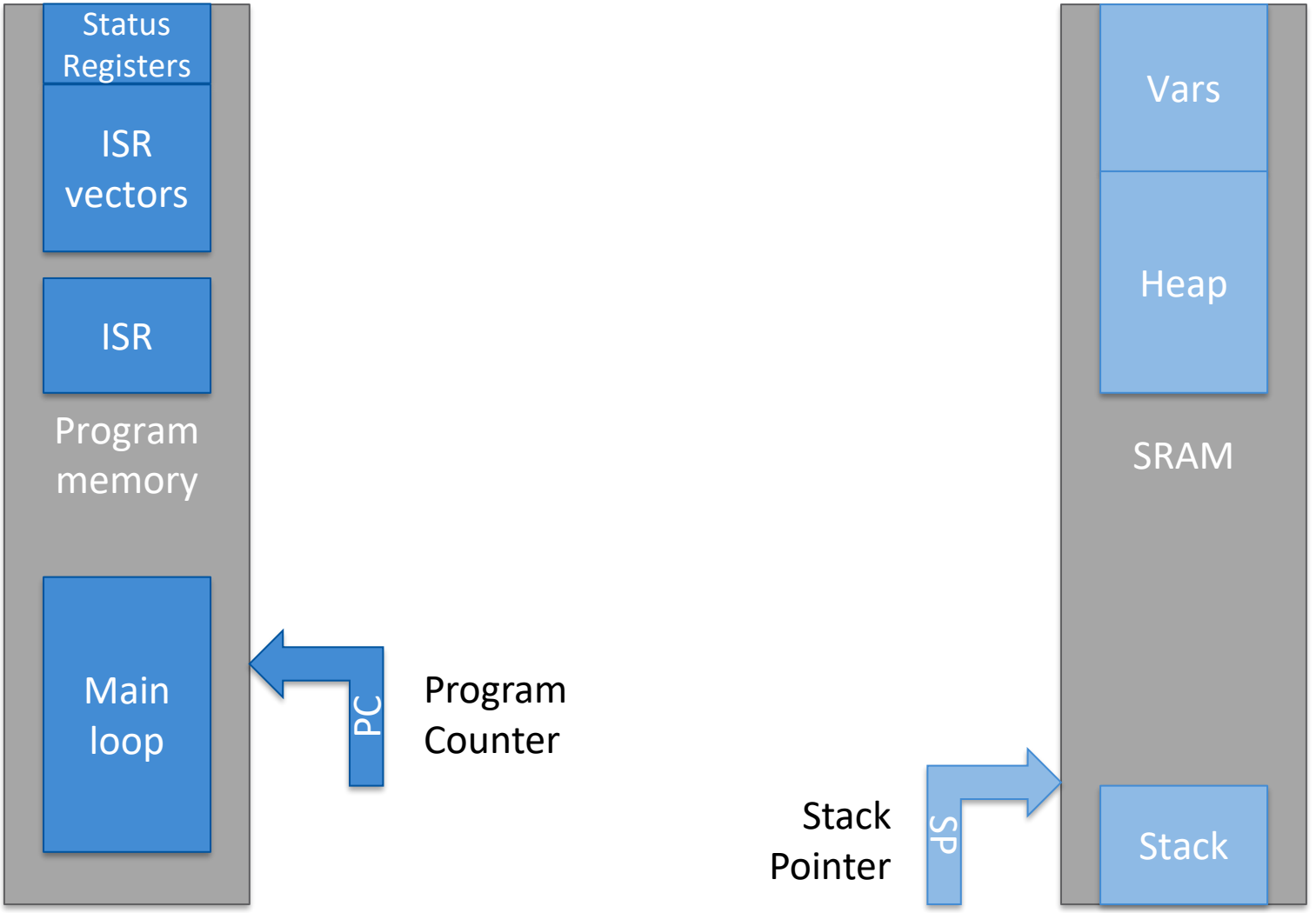
Interrupt Handling

- MCU überwacht bestimmte Ereignisse
- Tritt ein Ereignis ein, wird ein Interrupt-Flag gesetzt

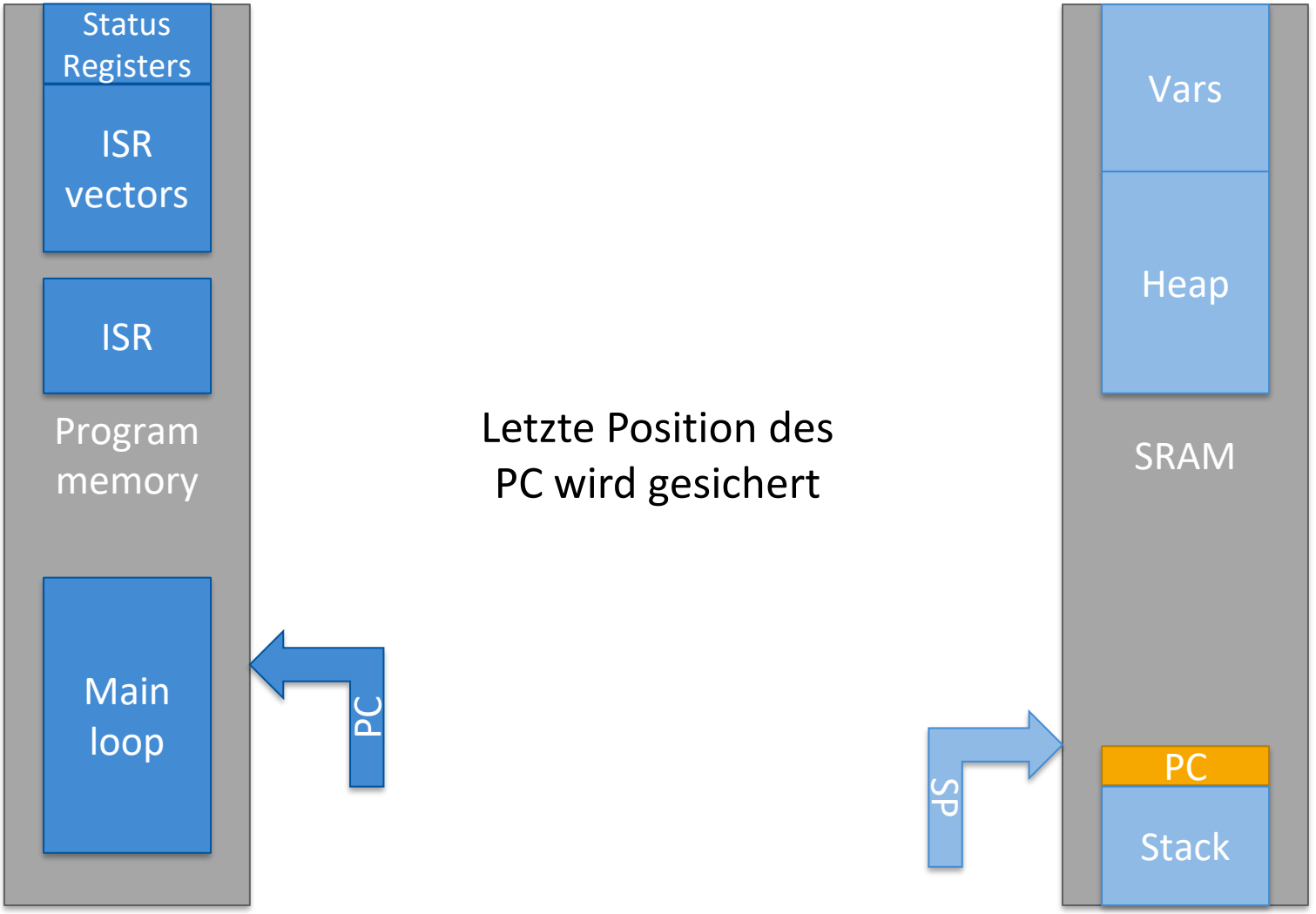
- MCU ruft ISR auf, wenn drei Bits gesetzt sind:
  - Global Interrupt Enable Bit (I-Bit)
  - Interrupt Enable Bit (für Timer0 Overflow: TOIE0 in TIMSK)
  - Interrupt Flag (für Timer0 Overflow: TOV0 in TIFR)

- Prioritäten lösen Konflikte auf
  - Statische Prioritäten (z.B. Atmel ATmega)
  - Dynamische Prioritäten (z.B. Renesas R8C)

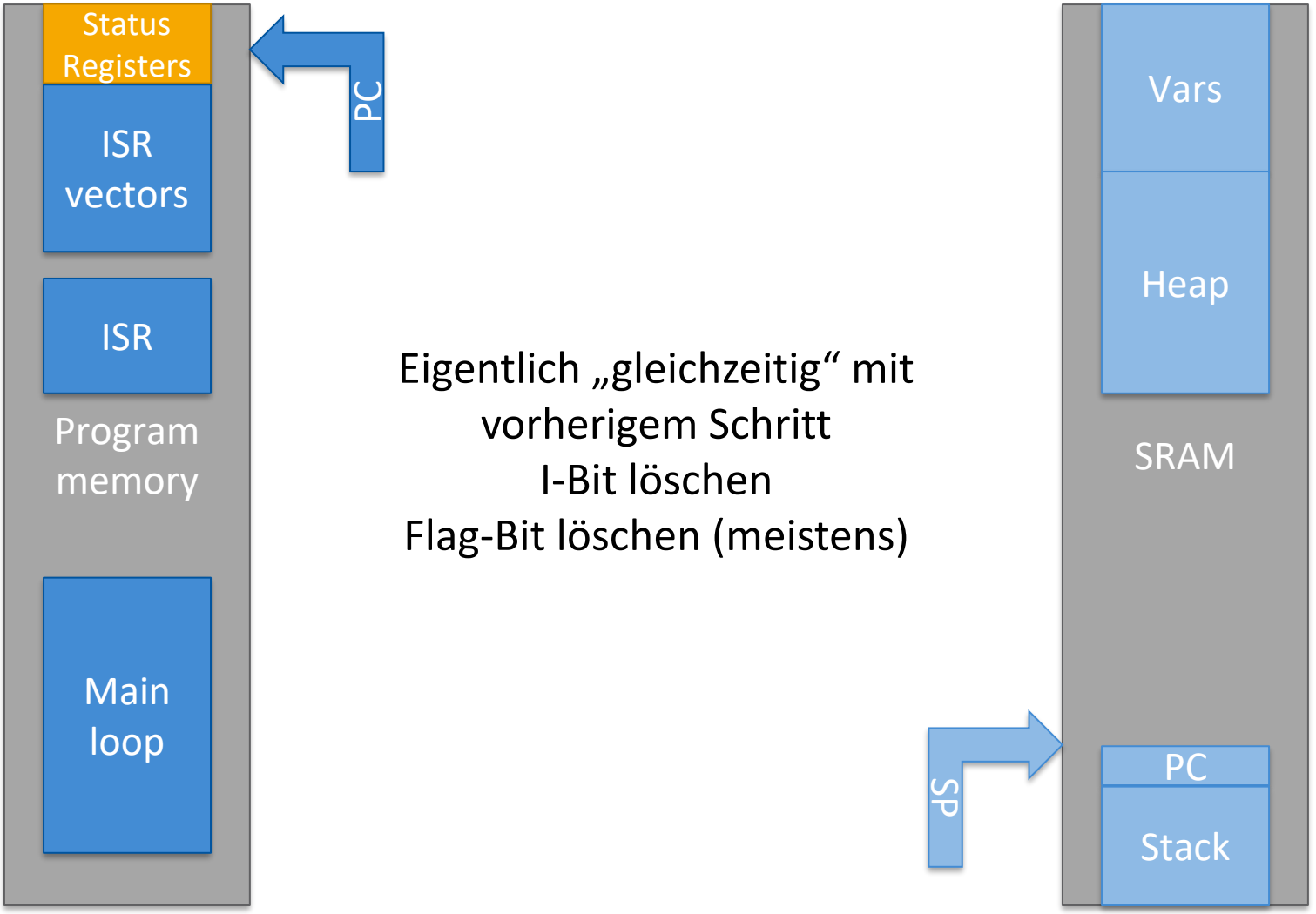
Vor ISR Aufruf



Rücksprungadresse Sichern

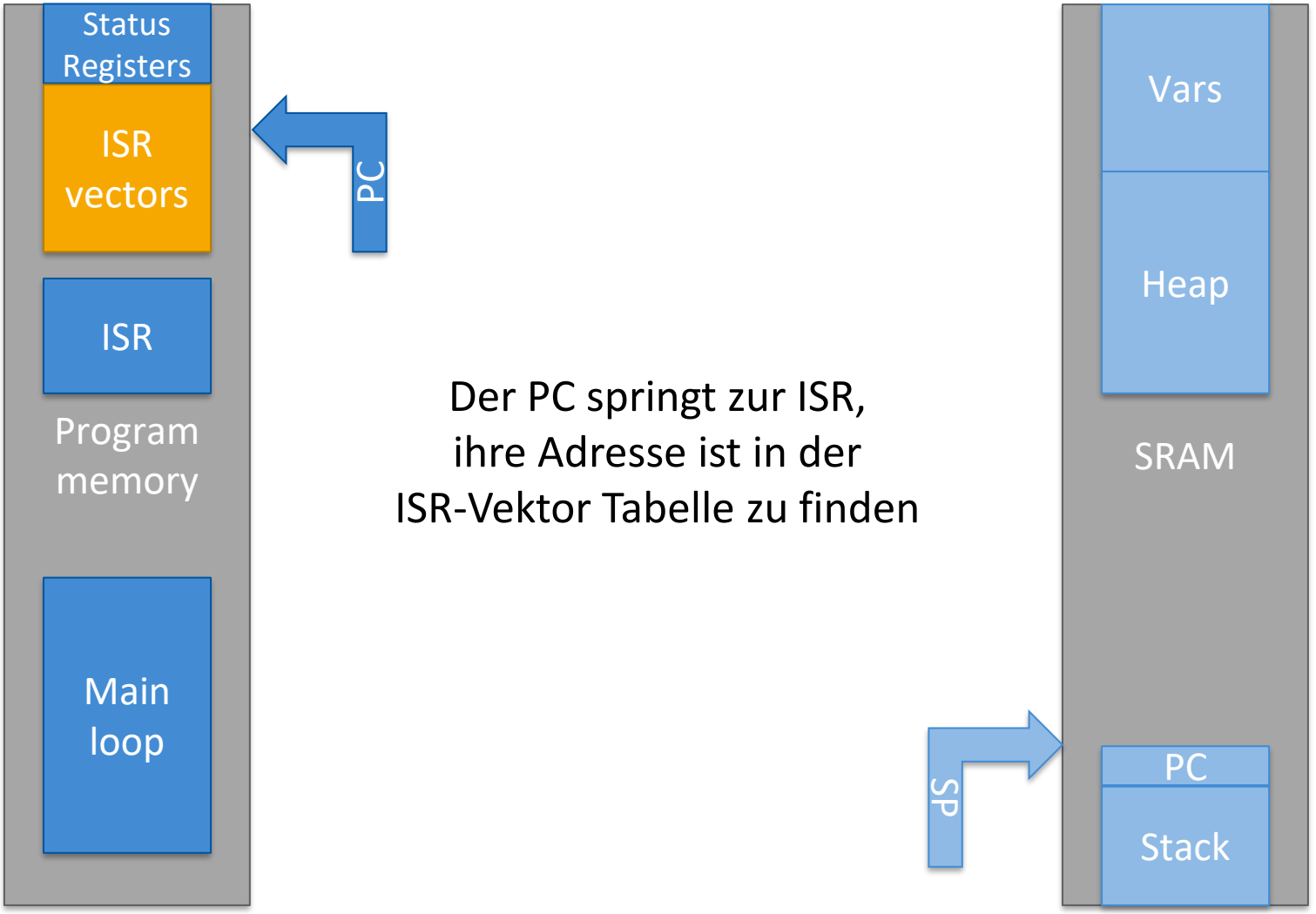


Sprung zum Interruptvektor



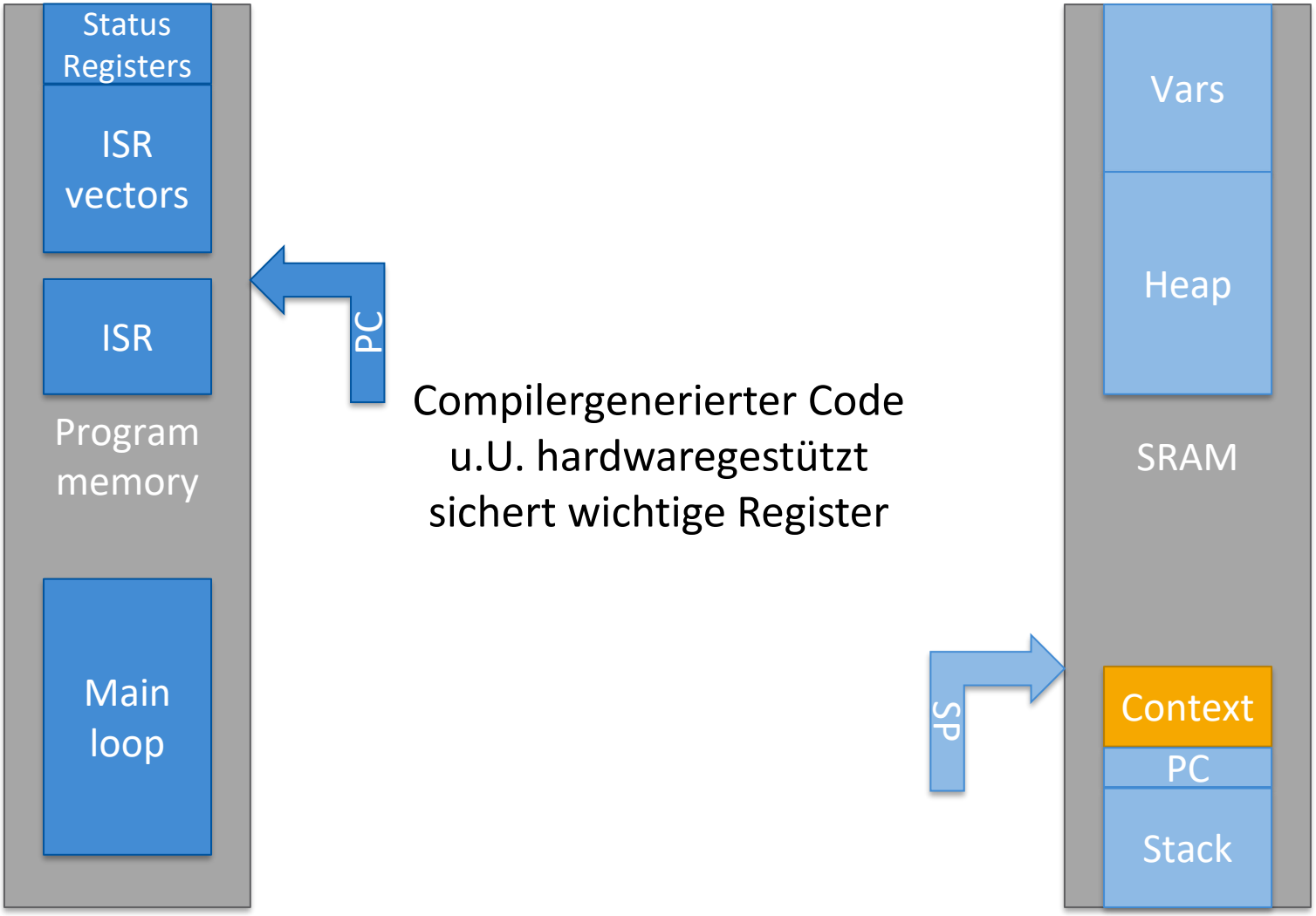
Eigentlich „gleichzeitig“ mit vorherigem Schritt
I-Bit löschen
Flag-Bit löschen (meistens)

Einsprung in ISR

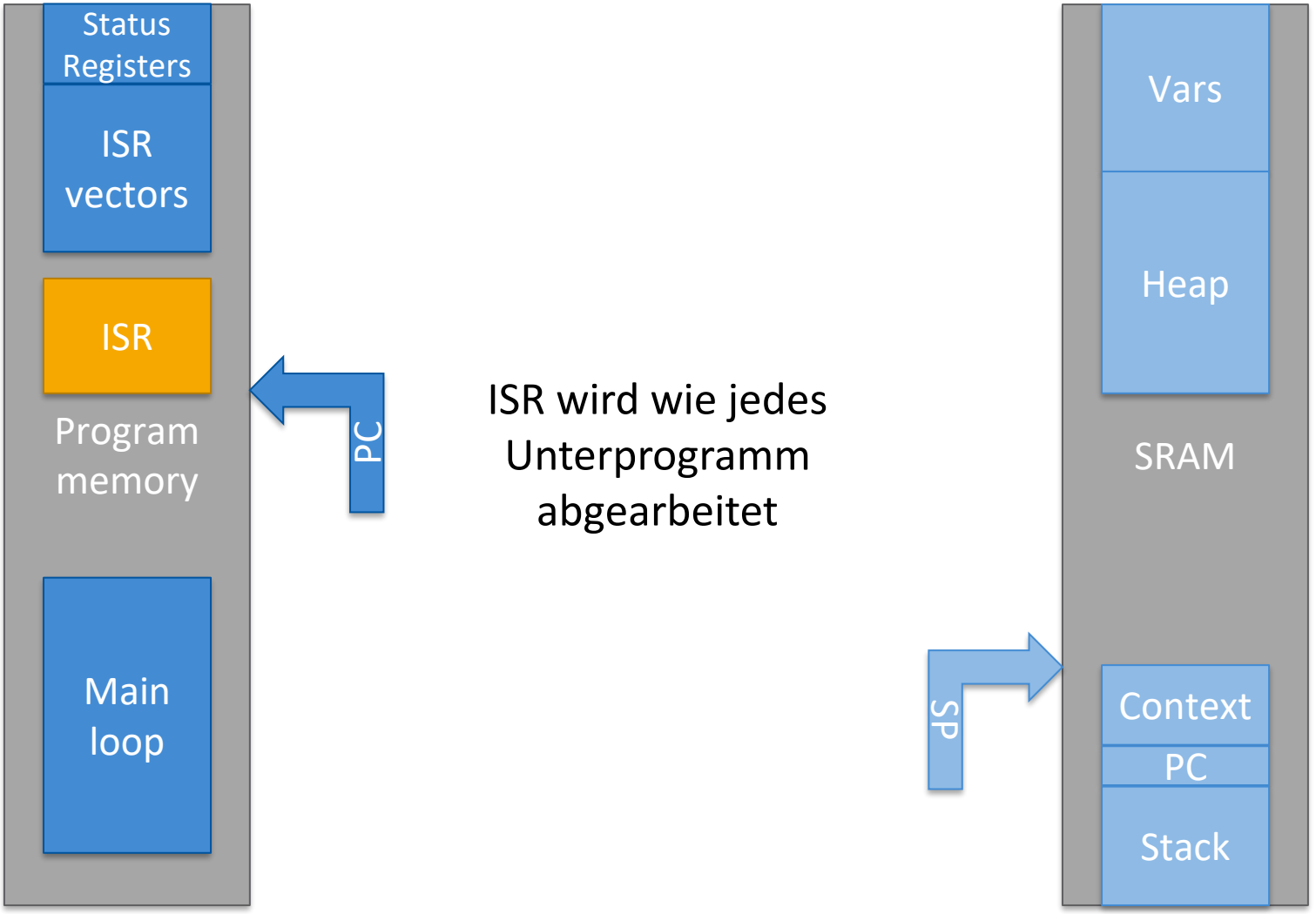


Der PC springt zur ISR,
ihre Adresse ist in der
ISR-Vektor Tabelle zu finden

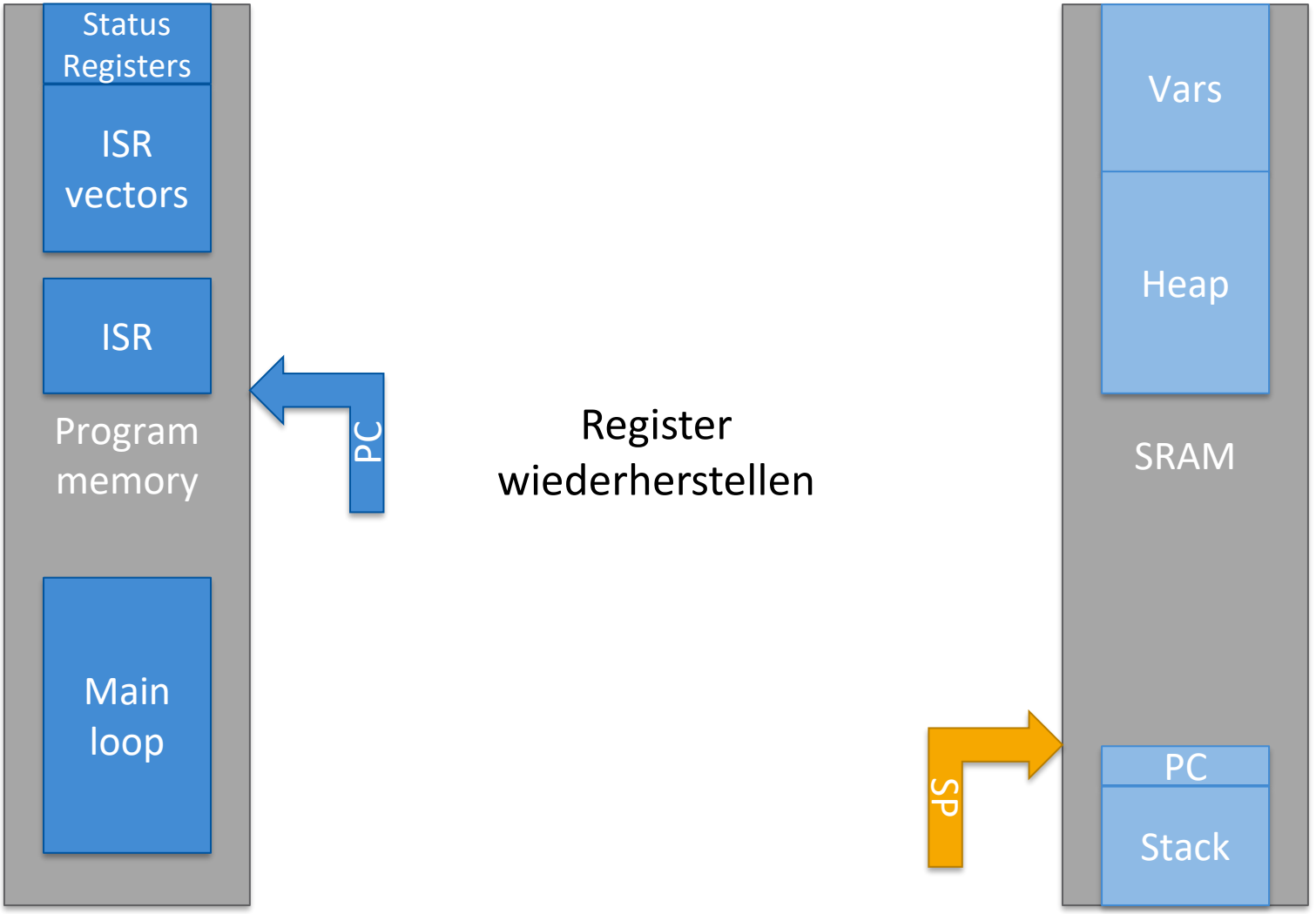
Kontextsicherung



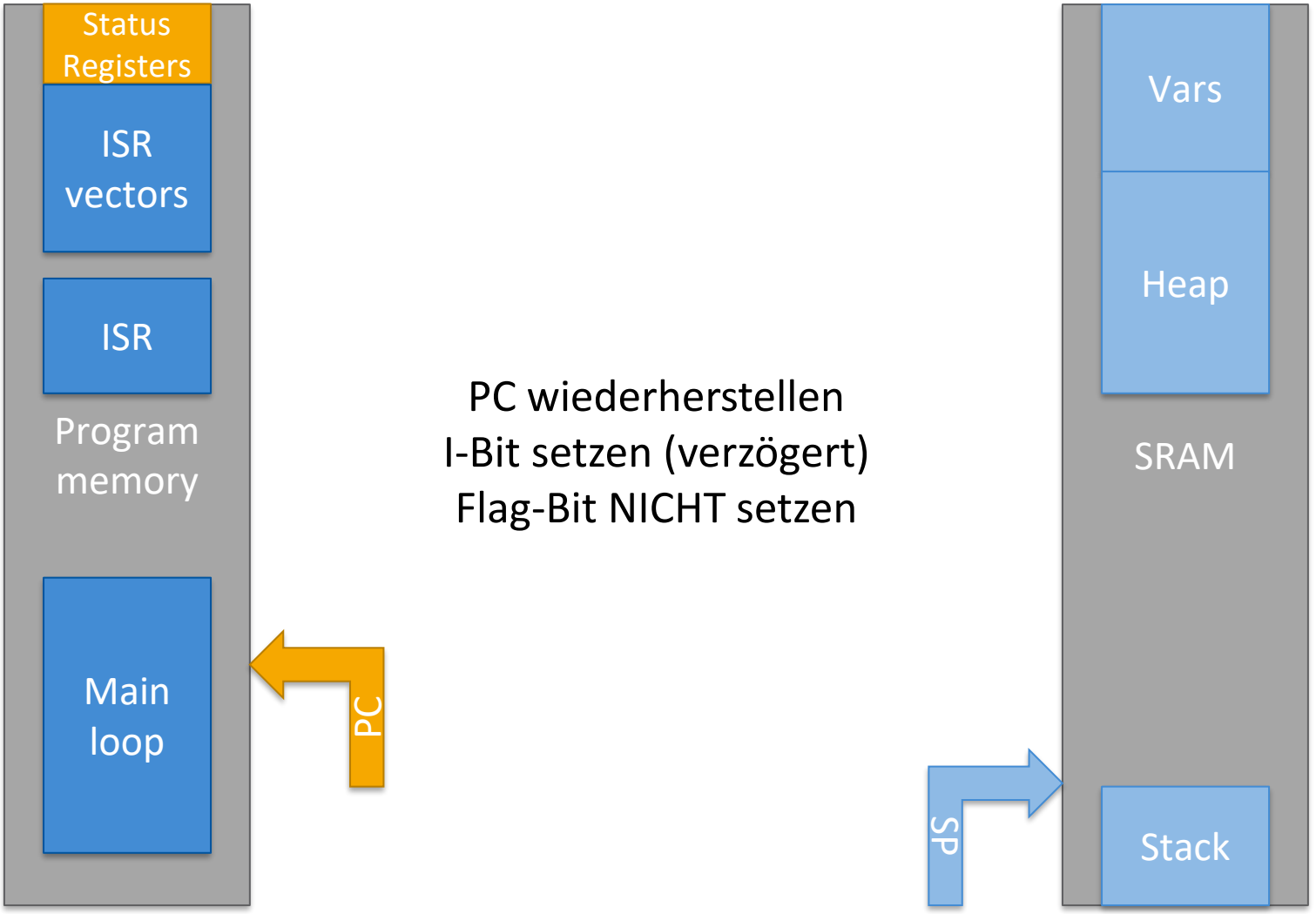
ISR Abarbeitung



Kontextwiederherstellung



Rücksprung ins Hauptprogramm



ISR Ablauf - Zusammenfassung

1. ISR wird durch Ereignis ausgelöst
   - Rücksprungadresse (aktuellen PC) auf Stack sichern
   - Global Interrupt Enable Bit löschen
   - Interrupt Flag Bit löschen (in der Regel)
   - Entsprechenden Eintrag der Interrupt Vektor Tabelle anspringen
2. Dort steht Sprunganweisung zur eigentlichen ISR
3. Weitere Kontextsicherung
4. ISR Ausführung
5. Kontextwiederherstellung
6. ISR wird durch Maschinenbefehl RETI verlassen
   - Rücksprungadresse vom Stack lesen und anspringen
   - Global Interrupt Enable Bit setzen (verzögert)

Abschnitt 13.4

Timer / Counter

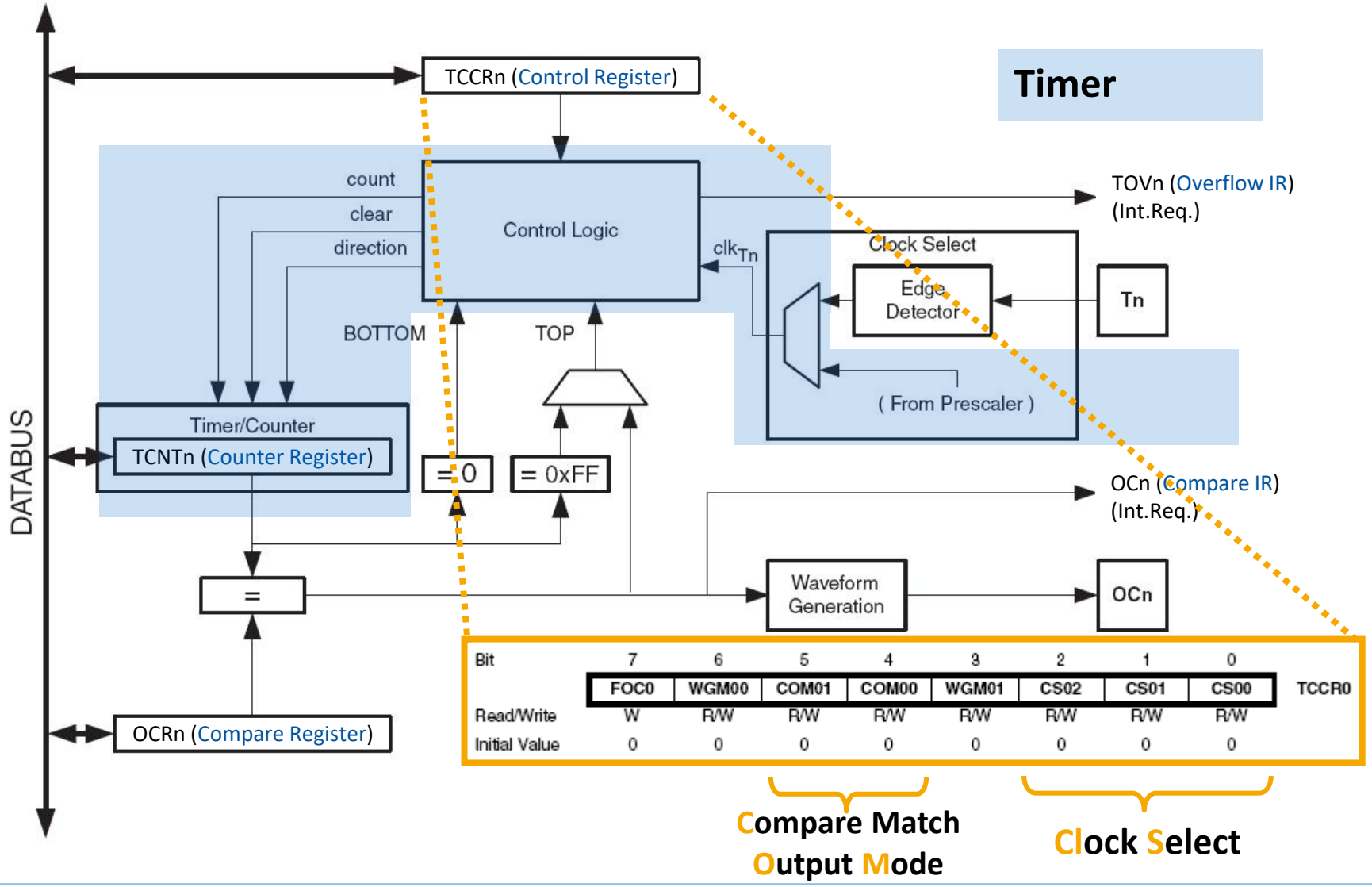
Timer/Counter

- On-Chip-Peripherie
- Counter
  - **zählt externe Ereignisse**,
  - z.B. Anzahl der steigenden Flanken an PINB2.
- Timer
  - **zählt Clock-Zyklen** (mit oder ohne Prescaler).
- Jeder Timer ist ein Counter.
- Die meisten Mikrocontroller haben einen oder mehrere Timer/Counter mit 8 und/oder 16 Bit Auflösung.

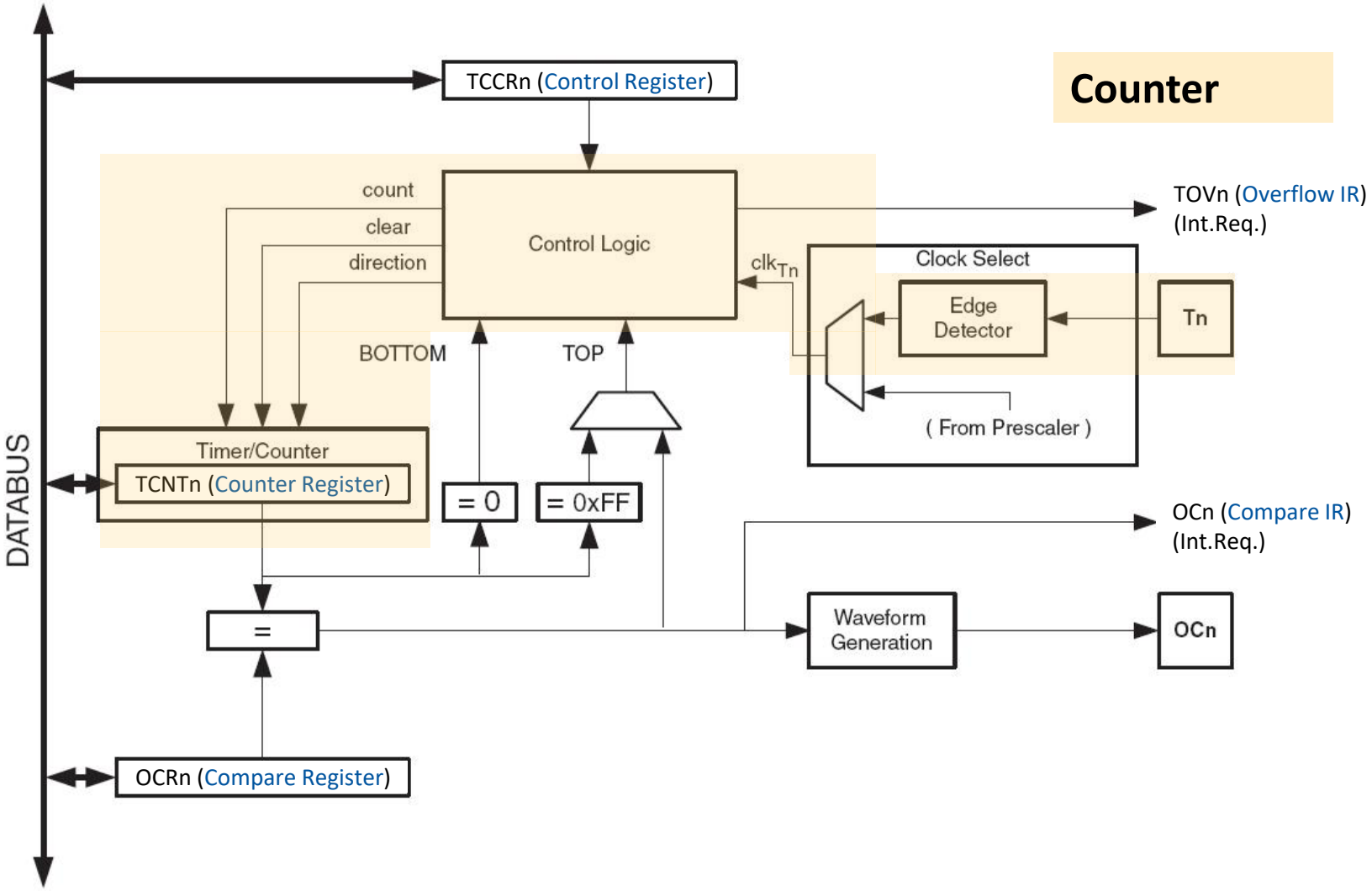
Timer/Counter

- Jede Timer/Counter-Einheit besteht aus einem Counter-Register, welches inkrementiert oder dekrementiert wird.
- Wichtig für das Verhalten der Einheit sind die dazugehörigen Control Register:
  - Ein- / Ausschalten
  - Operationsmodus
  - Vorteiler (engl. Prescaler)
- Häufig werden Compare Register angeboten, welche Interrupt-Flags setzen, wenn der Inhalt mit dem Counter Register übereinstimmt.
- Externe Signale können außerdem das Speichern eines Zeitstempels (engl. Input Capture) verursachen und einen Interrupt auslösen.

Timer/Counter



Timer/Counter



Anwendungsgebiete

- (Counter) Zählen von Ereignissen
- (Timer) Zeitmessung zwischen zwei Ereignissen
- (Timer-Interrupt) Regelmäßige Unterprogrammausführung
- (Input Capture) Zeitstempel für Ereignisberichte
- (Output Compare) Digitale Signalerzeugung

Anwendungsgebiete

- Digital/Analog-Wandlung
  - Pulsweiten Modulation (Pulse Width Modulation, PWM)
  - Timer erzeugt schnelles, digitales, periodisches Signal
  - High-Time und Periode einstellbar
  - Externe Glättung führt zu analoger Spannung zwischen V_{CC} und GND
  - Sehr platzsparend und kostengünstig
  - Schlechte Qualität
  - Näheres in Kapitel 15

Watchdog Timer (WD)

- Spezieller Timer
- Zählt von einstellbarem Wert runter auf 0
- Erreicht der WD 0, wird der Mikrocontroller resettet
- Während der Programmausführung muss der WD zurückgesetzt werden, um einen Neustart zu verhindern
- Deadlock detection
- Achtung: falsche Annahmen über Ausführungszeiten können ungewünschte Resets herbeiführen!

Weitere Peripherie

- Kommunikationsschnittstellen
- Analogmessung und –wandlung
- Externer Speicher
- Werden über Spezialregister gesteuert
  - Kontrollregister
  - Statusregister
  - Datenregister
- Keine eigenen Befehle, sondern Nebenwirkungen
- Genaue Semantik stark hardwareabhängig