Interrupt

Interrupts

Hochschule Offenburg
University of Applied Sciences

- Bei Interrupts handelt es sich um einen „Benachrichtigungs-mechanismus“, welcher es einer ALU erlaubt, mit minimaler Verzögerung aufgetretene Ereignisse zu erkennen und zu verarbeiten
- Interrupts werden in drei Kategorien unterteilt
  - Externen Interrupts (z.B.: Reset)
  - Interne Interrupts (z.B.: Timer)
  - Software Interrupts (irrelevant bei Mikrocontrollern)

Interrupt-Verarbeitung Intel x86-Architektur

Hochschule Offenburg
University of Applied Sciences

- Interrupts sperren
- Program Counter (PC) und Statusregister sichern
- Privilege Level prüfen
- PC und Statusregister der Interrupt-Routine laden
- Interrupt freigeben
- Ausführen der Interruptbehandlungsroutine
- Interrupts sperren
- PC und Statusregister wiederherstellen
- Interrupts freigeben

Interrupt-Verarbeitung Atmel AVR

Hochschule Offenburg
University of Applied Sciences

- Interrupts sperren
- Program Counter (PC) sichern
- PC der Interrupt-Routine laden
- Ausführen der Interruptbehandlungsroutine
- PC wiederherstellen
- Interrupts freigeben

Timing der Interruptroutine

Hochschule Offenburg
University of Applied Sciences

- Start der Interruptroutine
  - Beim eintreffen eines Interrupts wird der aktuelle Befehl noch vollständig Bearbeitet (drei Taktzyklen bei einem Sprungbefehl).
  - Vier weitere Takte werden für die Interrupt-Sperrung, das Löschen des jeweiligen Interrupt Flags, das Sichern des PC sowie der Sprung an die Startadresse der Interrupt-Routine benötigt (in einem Sleep-Modus werden acht Takte benötigt)
- Ende der Interruptroutine
  - Es werden wiederum vier Takte benötigt, in denen der urspüngliche PC vom Stack wiederhergestellt wird, der Sprung an die ursprüngliche Adresse erfolgt und die Interrupts freigegeben werden

Intialisierung Interrupt

Hochschule Offenburg
University of Applied Sciences

- Ein Programm mit Interruptsteuerung hat folgende Merkmale
  - Das Global Interrupt Enable Bit muss gesetzt sein (SREG |= (1<<7))
    - Bei AVR GCC Compiler wird dies durch das Makro sei() erreicht.
  - Eine Interrupt-Service-Routine muss vorhanden sein

```
# include <avr/interrupt.h>

ISR(INT0_vect) //Vector name
{
   PORTC=0x55;
}
```

  - Interrupt muss initialisiert werden

```
...
MCUCR|=(1<<ISC01)|(1<<ISC00); //Steigende Flanke löst aus
GICR|=(1<<INT0);              //INT0 enable
...
```

AD - Wandler

AD-Wandler

Hochschule Offenburg
University of Applied Sciences

- **10-bit Resolution**
- **0.5 LSB Integral Non-linearity**
- **±2 LSB Absolute Accuracy**
- **13 - 260 µs Conversion Time**
- **Up to 15 kSPS at Maximum Resolution**
- **8 Multiplexed Single Ended Input Channels**
- **7 Differential Input Channels**
- **2 Differential Input Channels with Optional Gain of 10x and 200x**
- **Optional Left adjustment for ADC Result Readout**
- **0 - VCC ADC Input Voltage Range**
- **Selectable 2.56V ADC Reference Voltage**
- **Free Running or Single Conversion Mode**
- **ADC Start Conversion by Auto Triggering on Interrupt Sources**
- **Interrupt on ADC Conversion Complete**
- **Sleep Mode Noise Canceler**

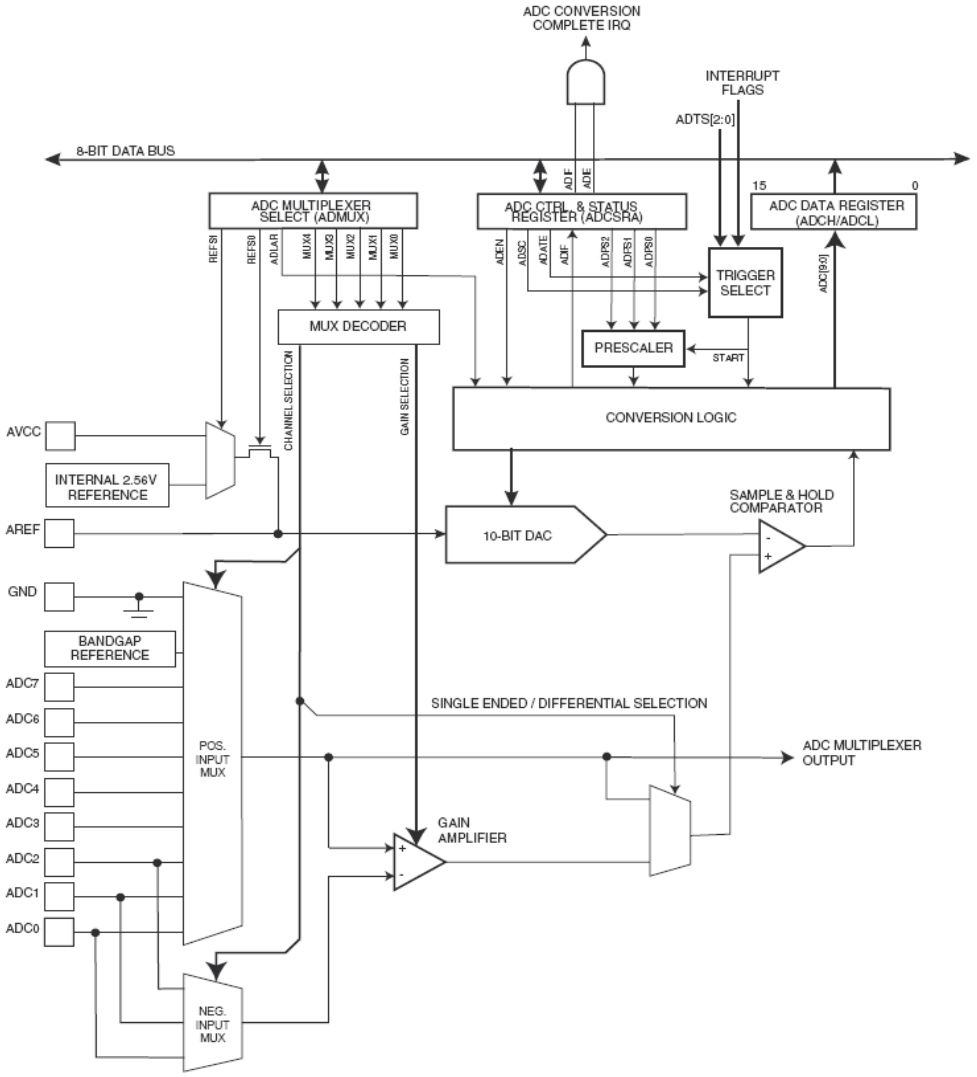
AD-Wandler

- Single endet Conversion

$$ADC = \frac{V_{IN} \cdot 1024}{V_{REF}}$$

- Differential channels

$$ADC = \frac{(V_{POS} - V_{NEG}) \cdot GAIN \cdot 512}{V_{REF}}$$



ADMUX Register

Bit	7	6	5	4	3	2	1	0	
	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- REFS[1:0]:

  Reference Selection Bits

- ADLAR:

  ADC Left Adjust Result

- MUX[4:0]:

  Analog Channel and Gain Selection Bits

ADCSRA Register

Bit	7	6	5	4	3	2	1	0	
	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- ADEN: ADC Enable
- ADSC: ADC Start Conversion
- ADATE: ADC Auto Trigger Enable
- ADIF: ADC Interrutpt Flag
- ADIE: ADC Interrupt Enable
- ADPS[2:0]: ADC Prescaler Selection Bits

SFIOR Register

Bit	7	6	5	4	3	2	1	0	
	ADTS2	ADTS1	ADTS0	–	ACME	PUD	PSR2	PSR10	SFIOR
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- ADTS[2:0]:

  ADC Auto Trigger Source

Beispiel 1

Hochschule Offenburg
University of Applied Sciences

```
#include <stdlib.h>
#include <avr/io.h>

int main (void)
{   ...
    uint16_t result;

    ADCSRA = (1<<ADEN) | (1<<ADPS1) | (1<<ADPS0);    // Frequenzvorteiler
                                                     // und ADC aktivieren
    ADMUX = 0;                                       // Kanal waehlen (Kanal 0)

    ADMUX |= (1<<REFS1) | (1<<REFS0);                // interne Referenzspannung

    ADCSRA |= (1<<ADSC);                             // eine Wandlung
    while(!(ADCSRA & (1<<ADIF)));                    // auf Abschluss warten
    ADCSRA |= (1<<ADIF);                             // ADIF Bit loeschen
    result = ADC;                                    // Ergebnis speichern
    ...
```