Atmel AVR für Dummies

fd0@koeln.ccc.de

29.12.2005

Übersicht

- Hardware
  - Kurzvorstellung Atmega8
  - Programmierkabel (Eigenbau vs. Kommerzlösung)
  - Alternative: Bootloader (Programmieren via rs232)
- Software
  - Speicher (Flash, RAM, EEPROM)
  - Instruktionen (Syntax, Dokumentation)
  - (viele) Beispiele

Atmega8

- Günstig
- Einfach zu Programmieren
- Viele Ein-/Ausgänge
- Kann direkt LEDs treiben und Schalter abfragen
- Freie Compiler und Programmer verfügbar

Einfaches Programmierkabel

Einfaches Programmierkabel am Parallelport:

Pin am AVR	Schutzwiderstand	Pin am Parallelport
GND	–	GND (18)
SCK (19)	470 Ohm	Strobe (1)
MISO (18)	220 Ohm	Busy (11)
MOSI (17)	470 Ohm	D0 (2)
Reset (1)	–	Init (16)

- **Vorteile:** Einfach, schnell gebaut und günstig.
- **Nachteil:** Elektrisch sehr fragwürdig. . .

Alternative: Bootloader

Mikrocontroller programmiert sich selbst, Bootloaderprogramm installiert in speziellem Speicherbereich.

- **Vorteile:** Programmieren über verschiedene Schnittstellen (rs232, I2C, CAN, . . .) möglich.
- **Nachteile:** Belegt Platz im Flashrom, Mikrocontroller muss initialisiert werden (Henne-Ei-Problem.)

Atmel AVR Interna - Speicher im Mikrocontroller

Verschiedene Speicherbereiche im Atmega8:

- Flash: 8KB
- RAM: 1KB
- EEPROM: 512 Byte

Atmel AVR Interna - Register in der CPU

- Arbeitsregister (R0-R31)
- Spezialregister (zb. SREG, SP, PC, . . .)
- Konfigurationsregister (zb. DDRB, . . .)

Arbeitsregister

Ein Register enthält genau ein Byte (also 8 Bit.)

Es gibt:

- 16 niedrige Register: R0-R15
- 16 hohe Register: R16-R31
- davon 3 mal 2 Stück als 16Bit-Register verwendbar

Spezialregister

Die wichtigsten Spezialregister:

- Statusregister: SREG (8bit)
- Program Counter: PC (16bit)
- Stack Pointer: SP (16bit)

Atmel AVR Interna - Übersicht

Speicher:

- Flash
- RAM
- EEPROM

Register: R0-R31

Spezialregister: SREG, PC, SP

Befehlssyntax - Beispiel

```
ldi r16, 255            ; lade 255 in das Register r16
out DDRB, r16           ; konfiguriere PB0-PB7 als Output-Pins
ldi r16, 0              ; lade 0 in das Register r16
out PORTB, r16          ; setze PB0-PB7 auf low (0)
```

Befehlssyntax - Beispiel Prolog (Programmanfang)

Prolog:

```
.include "m8def.inc"
.ORG 0                          ; das folgende wird an Adresse 0
                                ; in den Flash-Speicher geschrieben
rjmp reset                      ; springe zu "reset", wenn der
                                ; Prozessor gestartet wird
.ORG INT_VECTORS_SIZE           ; springe ein Stueck vorwaerts, ab
                                ; hier kommt das eigentliche Programm
```

Befehlssyntax - Beispiel Reset-Prozedur

```
.ORG INT_VECTORS_SIZE
reset:
    ldi r16, 255            ; lade 255 in das Register r16
    out DDRB, r16           ; konfiguriere PB0-PB7 als Output-Pins
    ldi r16, 0              ; lade 0 in das Register r16
    out PORTB, r16          ; setze PB0-PB7 auf low (0)
    ldi r16, 255            ; lade 255 in das Register r16
loop:
    in r17, PORTB           ; lese aktuellen Portstatus
                            ; in das Register r17
    eor r17, r16            ; drehe jedes Bit in r17
    out PORTB, r17          ; setze Portstatus neu
    rjmp loop               ; Hauptschleife
```

komplettes Beispiel

```
.include "m8def.inc"
.ORG 0                      ; das folgende wird an Adresse 0
                            ; in den Flash-Speicher geschrieben
rjmp reset                  ; springe zu "reset", wenn der
                            ; Prozessor gestartet wird
.ORG INT_VECTORS_SIZE       ; springe ein Stueck vorwaerts, ab
                            ; hier kommt das eigentliche Programm
reset:
    ldi r16, 255            ; lade 255 in das Register r16
    out DDRB                ; konfiguriere PB0-PB7 als Output-Pins
    ldi r16, 0              ; lade 0 in das Register r16
    out PORTB, r16          ; setze PB0-PB7 auf low (0)
    ldi r16, 255            ; lade 255 in das Register r16
loop:
    in r17, PORTB           ; lese aktuellen Portstatus
                            ; in das Register r17
    eor r17, r16            ; drehe jedes Bit in r17
    out PORTB, r17          ; setze Portstatus neu
    rjmp loop               ; Hauptschleife
```

Stack

„Kellerspeicher" oder „LIFO" (Last-In-First-Out)

Initialisierung:

```
ldi r16, low(RAMEND)          ; Stack liegt am Ende des RAM
out SPL, r16
ldi r16, high(RAMEND)
out SPH, r16
```

Stack - rcall

Für Unterprozeduraufrufe muss die alte Programmposition ,,gemerkt" werden, sie wird auf den Stack geschrieben und bei der Rückkehr in das Programm wieder vom Stack gelesen.

rcall Beispiel

```
reset: [...]
    ldi r16, low(RAMEND)        ; Stack liegt am Ende des RAM
    out SPL, r16
    ldi r16, high(RAMEND)
    out SPH, r16
    [...]
    rcall test                  ; Speicher diese Position
                                ; und springe zu "test"
test:                           ; Beginn der Unterprozedur
    [...]
    ret                         ; Rueckkehr an die vorher
                                ; gemerkte Position (vom Stack)
```

Preisfrage: Was ist ein Interrupt?

Der Atmega8 kann in der CPU auf Ereignisse reagieren und abweichend vom Hauptprogramm ,,automatisch" einen Prozeduraufruf ausführen.

Beispiel: Timer

CPU ,,zählt" mit und löst bei Erreichen eines voreingestellten Wertes einen Interrupt aus.

Weiterführende URLs

- http://www.mikrocontroller.net/tutorial
- http://www.avrfreaks.net
- http://www.avrbeginners.net
- http://wiki.koeln.ccc.de/index.php/Atmel_Mikrocontroller_Kickstart
- http://del.icio.us/fd0/atmel