Assemblerdirektiven Atmel AVR (AVRASM)

24. 4. 2013

Aufbau einer Programmzeile:

Allgemein:

Marke (Label)**:**	Operationscode Operanden	**;** Kommentar

	Operationscode
	Operationscode Operand
	Operationscode Zieloperand**,** Quelloperand
Marke (Label)**:**	*(Marke in eigener Zeile. Es darf ein Kommentar nachfolgen.)*
; Kommentar	*(Kommentarzeile)*

Konstanten in Befehlen:

Format	Angabe	Beispiele
Dezimal	Zahlen 0...9	ldi r16, 231 cpi r17, –15
Hexadezimal	0x, 0X oder $	subi r16,0x2F ldi r16, $21
Binär	0b, 0B	subi r16, 0b11010011
Zeichen	' '	ldi r16, 'G'

Der Konstantenangabe darf ein Minuszeichen vorangestellt werden.
Überschreitungen des Wertebereichs werden vom Assembler angezeigt.

Assemblerdirektiven:

Bezeichnung	Wirkung	Beispiele
Definition mnemonischer Bezeichnungen		
DEF	Benennung von Registern	.def time_count = r17
EQU	Benennung von Konstanten	.equ limit = 0x34
SET	Benennung von Konstanten	.set limit = 0x34
UNDEF	Benennung von Registern aufheben (zwecks erneuter Benennung)	.undef time_count

Bezeichnung	Wirkung	Beispiele
Segmentauswahl		
CSEG	Die nachfolgenden Angaben betreffen das Programmsegment (Codesegment), also den Flash-ROM	.cseg
DSEG	Die nachfolgenden Angaben betreffen das Datensegment, also den SRAM	.dseg
ESEG	Die nachfolgenden Angaben betreffen den EEPROM	.eseg
Festlegen der Startadresse		
ORG	Im betreffenden Segment wird die Startadrese zum Assemblieren der nachfolgenden Angaben eingestellt	.org 0
Platz für Variabe reservieren (SRAM, EEPROM)		
BYTE	Die angegebene Byteanzahl wird im betreffenden Segment (SRAM oder EEPROM) freigehalten. Der zugehörigen Marke wird die Anfangsadresse zugewiesen	string: .byte 30 count: .byte 2
Konstanten definieren (Programmspeicher, EEPROM)		
DB, DW, DD, DQ	Die angegebenen Konstanten werden in das jeweilige Format umgesetzt (dabei werden sie ggf. gerundet) und im betreffenden Segment (Programmspeicher oder EEPROM) abgelegt	.db 0x23, 15, 0b01101110, "Textbeispiel", 0 .dw 0x23af .dd 0x1234567 .dq 0x123456789abcdef
Organisation und Programmentwicklung		
INCLUDE	Einbinden der angegebenen Quelldatei	.include "m16def.inc"
DEVICE	Zuweisung des Schaltkreistyps, für den der Code erzeugt werden soll	.device atmega16
MACRO	Beginn einer Makrodefinition	.macro outi ldi r16, @0 out @1, r16 .endm
ENDMACRO, ENDM	Ende einer Makrodefinition	.endmacro
LIST	Die Übersetzung des nachfolgenden Quelltexts wird ins Listfile übernommen	.list
NOLIST	Die Übersetzung des nachfolgenden Quelltexts wird nicht ins Listfile übernommen	.nolist

Bezeichnung	Wirkung	Beispiele
LISTMAC	Die Übersetzung der Makros wird ins Listfile übernommen (ansonsten erscheint nur der Makroaufruf)	.listmac
EXIT	Der nachfolgende Quelltext wird beim Assemblieren übergangen	.exit

Makros:

Es ist möglich, Befehlsfolgen, die immer wiederkehren, unter einer einzigen frei wählbaren Bezeichnung zusammenzufassen. Eine solche Befehlsfolge wird als Makrobefehl oder kurz als Makro bezeichnet. Sie ist nur einmal zu schreiben und wird vom Assembler immer dann ins Programm eingefügt, wenn sie aufgerufen wird.

Makros und Unterprogramme

Makro	Unterprogramm
Wird nur einmal geschrieben	Wird nur einmal geschrieben
Wird im Maschinencode bei jedem Aufruf erneut gespeichert	Wird im Maschinencode nur einmal gespeichert
Bei jedem Aufruf wird die gesamte Befehlsfolge erneut eingefügt	Bei jedem Aufruf wird ein Aufrufbefehl eingefügt
Parameterübergabe zur Assemblierzeit. Die aktuellen Parameter werden einfach hingeschrieben: macrobeispiel zeichenadresse, pixeladresse	Parameterübergabe zur Laufzeit. Muß im Assembler ausprogrammiert werden: load zeichenadresse* load pixeladresse* call unterprogrammbeispiel
Maschinenprogramm länger	Maschinenprogramm kürzer
Läuft schneller, weil kein Overhead	Läuft nicht so schnell, weil Parameterübergabe, Aufruf und Rückkehr Zeit kosten (Overhead)

Aufbau eines Makros:

.macro Name

– Der Programmtext. –

Die Parameter heißen @0, @1 usw. Sie werden nicht deklariert.

.endmacro (oder .endm)

Wieviele Parameter?

- AVRASM: maximal 10 (@0...@9).
- AVRASM2: unbegrenzt.

Makros im Makro?
AVRASM2 erlaubt das Schachteln von Makros. Atmel empfiehlt aber, es damit nicht zu übertreiben. Erfahrungswert: Schachtelungstiefe 3 wird noch ohne Probleme unterstützt.

Makroaufruf:

Name 1. aktueller Parameter, 2. aktueller Parameter usw.

Der Assembler verwendet die erste Zeichenkette nach dem Makronamen als 1. Parameter, die durch Komma getrennte zweite Zeichenkette als 2. Parameter usw. Wenn es nicht zusammenpaßt, erscheint eine Fehlermeldung.

Makrobeispiele:

```
.macro lity             ; 16 bit literal (Direktwert)
ldi yl,low(@0)
ldi yh,high(@0)
.endmacro

.macro showtext         ; Textanzeige mit Direktwert (Zeichenkette)
lity @0                 ; FCB-Adresse
ldi b,@1                ; Adreßauswahl (0=Cursoradresse, 1=Hilfsadresse)
ldi bh,@2               ; Update (0=nein, 1=ja)
ldi zl,low(e1*2)        ; Zeiger auf Zeichenkette
ldi zh,high(e1*2)
rcall flashtolcd
rjmp e2
e1:
.db @3,0
e2:
.endm
```

Aufrufbeispiel:

showtext lcd_buffer, 0, 1, “HELLO, WORLD”