# AVR-ChipBasic8: BASIC-Referenz

## (c) 2006-2008 Jörg Wolfram


# 1 Lizenz

Das Programm unterliegt der GPL (GNU General Public Licence) Version 3 oder höher, jede Nutzung der Software/Informationen nonkonform zur GPL oder ausserhalb des Geltungsbereiches der GPL ist untersagt!

Die Veröffentlichung dieses Programms erfolgt in der Hoffnung, daß es Ihnen von Nutzen sein wird, aber OHNE IRGENDEINE GARANTIE, auch ohne die implizite Garantie der MARKTREIFE oder der VERWENDBARKEIT FÜR EINEN BESTIMMTEN ZWECK.

Alle im Text genannten Marken sind Eigentum des entsprechenden Inhabers.

# 2 Allgemeines

Jedes Programm besteht aus maximal 20 Programmzeilen (1-20). Nach jedem Schlüsselwort sollte ein Leerzeichen stehen. Viele Befehle blenden nicht benutzte Bits bei den Parametern aus oder begrenzen auf den gültigen Wertebereich (z.B. PL für PLOT).

# 3 Zahlen, Variablen und Funktionen

AVR-ChipBASIC8 kennt nur einen Datentyp, und das sind 8 Bit vozreichenlose Integerzahlen. Dazu gibt es 26 Variablen (A-Z). Konstanten können sowohl in Dezimalform als auch in Hexadezimalform (beginnend mit **$**) eingegeben werden. Folgende Operationen sind erlaubt:

- Addition **+**
- Subtraktion **-**
- Multiplikation *****
- Division **/**
- Modulo **%**
- Und-Verknpfung **&**

- Oder-Verknpfung **#**
- Vergleiche (=,<,>) liefern 0 für falsch oder 1 für wahr

Dazu kommen noch öffnende und schließende Klammern sowie Funktionen:

| | |
|---|---|
| NO(n) | invertiert den eingeklammerten Ausdruck bitweise |
| RV(n) | erzeugt eine Zufallszahl zwischen 0 und dem eingeklammerten Ausdruck |
| IN(n) | liefert einen digitalen Pegelwert, siehe Abschnitt Ein-/Ausgabe |
| AD(n) | liefert einen analogen Pegelwert, siehe Abschnitt Ein-/Ausgabe |
| EP(n) | liest ein Datenbyte aus dem internen EEPROM, Adresse 0...255 |
| KY(n) | liefert verschiedene Tastaturabfragen, siehe Abschnitt Tastatur |

# 4 Wertzuweisung

Wertzuweisungen beginnen nicht mit einem Schlüsselwort, sondern mit einem Variablennamen gefolgt von einem Gleichheitszeichen und einem Ausdruck.

```
01 A=-4
02 B=8*2
```

## 4.1 LIM v,min,max

Der Wert der Variable v wird auf den Wertebereich min...max begrenzt.

```
01 LIM A,1,6
02 LIM B,C,C+4
```

im ersten Beispiel wird die Variable A auf den Bereich 1...6 begrenzt, im zweiten Beispiel die Variable B auf den Wertebereich, der durch den Inhalt der Variable C und die 4 darauffolgenden Zahlen begrenzt ist.

## 4.2 INV v

Wandelt die Variable V in ihr Zweierkomplement um.

```
01 A=1
02 INV A
```

A ist jetzt 255 (-1).

# 5 Programmsteuerung

## 5.1 END

Einen END-Befehl gibt es nicht, anstelle muß ein GO zu einer nicht existierenden Zeile (0 oder >20) benutzt werden.

## 5.2 GO n

Mit dem GOTO-Befehl kann die Programmabarbeitung mit einer anderen Zeile fortgesetzt werden. Argument ist ein beliebiger Ausdruck.

```
01 B=0:GO 2
02 GO B+1
```

Das ist eine Endlosschleife, die nur mit CTRL+C abgebrochen werden kann.

### 5.3 IF

Die bedingte Anweisung besteht aus **IF** gefolgt von einem Ausdruck. Ist das Ergebnis des Ausdrucks Null, wird zum Anfang der nächsten Zeile gesprungen, andernfalls wird die Zeile weiter abgearbeitet.

```
10 IF A>5:A=5
```

### 5.4 FOR - NXT

Bei der Schleifenabarbeitung gibt es nur die Grundform **FOR A=1 TO C** ohne die Angabe der Schrittweite, die konstant 1 ist. Da der Stack auf 16 Einträge begrenzt ist, lassen sich nur 16 Schleifen bzw. Unterprogrammaufrufe schachteln.

```
01 CLS
02 FOR A=0 TO 9
03 FOR B=0 TO 9
04 PL A,B
05 NXT :NXT
```

Dieses Programm zeichnet ein Rechteck in die obere linke Ecke des Bildschirmes.

### 5.5 SUB - RET

**SUB Expr** ruft das Unterprogramm in der durch den Ausdruck definierten Zeile auf, mit **RET** wird wieder zurückgesprungen. Da der Stack auf 16 Einträge begrenzt ist, lassen sich nur insgesamt 16 Schleifen bzw. Unterprogrammaufrufe schachteln.

### 5.6 VID n

**VID 0** schaltet die Videoausgabe ab (auf schwarz). Programme laufen dadurch schneller ab. Mit **VID 1** wird die Bildschirmausgabe wieder eingeschaltet.

## 6 Bildschirm-Ausgabe

### 6.1 CO n

Mit dem COlor-Befehl wird die Vorder- sowie die Hintergrundfarbe festgelegt. Akzeptiert werden die Werte 0 und 1, wird 8 dazuaddiert werden die Zeichen in Großschrift ausgegeben. Dabei bedeutet:

| Value | Textfarbe | Textgröße | Pseudografik |
|---|---|---|---|
| 0 | schwarz auf weiß | normal | löschen |
| 1 | weiß auf schwarz | normal | zeichnen |
| 8 | schwarz auf weiß | groß | löschen |
| 9 | weiß auf schwarz | groß | zeichnen |

Nach dem Priogrammstart ist der Voreinstellwert 1.

### 6.2 CLS

Mit dem CLS-Befehl wird der Bildschirm gelöcht. Beim Programmstart geschieht das automatisch. Es wird die eingestellte Hintergrundfarbe (beim Programmstart schwarz) verwendet.

### 6.3 PRINT [?]

Der PRINT-Befehl dient zur Ausgabe auf den Bildschirm Zusätzlich kann die Ausgabe noch formatiert werden. Anstelle des PRINT Befehls muß ein Fragezeichen verwendet werden.

| | |
|---|---|
| ”TEXT” | der Text TEXT wird ausgegeben |
| @Expr1,Expr2 | Cursorpositionierung (Y=Expr1, X=Expr2) |
| %Expr | Direkte Ausgabe eines Zeichens mit Zeichencode=Expr |
| Expr | gibt das Ergebnis des Ausdrucks mit dem eingestellten Format aus |
| ; | Trenner zwischen Ausdrücken |
| , | Trenner zwischen Ausdrücken, Leerzeichen bis zur nächsten durch 8 teilbaren Position |

Steht am Ende des PRINT-Befehls einer der beiden Trenner, wird kein Zeilenvorschub ausgeführt.

### 6.4 SCR n

Es gibt ein Scrollfenster, in dem in alle 4 Richtungen gescrollt werden kann. Das Fenster beginnt bei Zeichen 2 in Zeile 2 und geht bis Zeichen 27 in Zeile 20. Somit gibt es um das Scrollfenster einen 2 Zeichen breiten Rand in dem z.B. Informationen stehen, die nicht mitgescrollt werden sollen.
Die freiwerdende Zeile wird mit Leerzeichen in der aktuellen Farbeinstellung gefüllt.

| n | Richtung |
|---|---|
| 0 | nach oben |
| 1 | nach rechts |
| 2 | nach unten |
| 3 | nach links |

```
10 SCR 0
```

verschiebt das Scrollfenster um 1 Zeichen nach unten.

## 7 Tastatur

### 7.1 INP V

die Variable V wird an der aktuellen Stelle eingelesen.

```
01 ? "Zahl: ";
02 INP Z:?
03 ? "Die Zahl war ";M
```

Enthält die Eingabe ungültige Zeichen, ist das Resultat 0.

### 7.2 Die Keycodes

Neben den „normalen” ASCII-Werten für Ziffern, Zahlen und Satzzeichen liefern RKEY und WKEY auch Codes für Funktionstasten etc.

| Code Hexadezimal | Code Dezimal | Taste |
|---|---|---|
| $E0 | 224 | Pos1 |
| $E1 | 225 | End |
| $E2 | 226 | Pfeiltaste nach links |
| $E3 | 227 | Pfeiltaste nach rechts |
| $E4 | 228 | Pfeiltaste nach oben |
| $E5 | 229 | Pfeiltaste nach unten |
| $E6 | 230 | Bild hoch |
| $E7 | 231 | Bild runter |
| $E8 | 232 | Einfg (Ins) |
| $E9 | 233 | Entf (Del) |
| $EA | 234 | Enter |
| $EB | 235 | Tabulator |
| $EC | 236 | Backspace |
| $ED | 237 | Esc |
| $F1...$FC | 241...252 | F1...F12 |

### 7.3 Die Funktion KY

Diese Funktion liefert verschiedene Tastaturabfragen als -1,0,1 Wert. Als Parameter wird die Art der Abfrage eingetragen. Bei den Abfragen 2...5 gilt: Ist keine der beiden Tasten betätigt, wird 0 zurückgegeben.

| | |
|---|---|
| KEY(0) | liefert den Keycode der gerade gedrückten Taste oder 0 zurück |
| KEY(1) | wartet auf einen Tastendruck und liefert den Keycode zurück |
| KEY(2) | linke Shift-Taste liefert 1, linke Control-Taste liefert -1, beide 0 |
| KEY(3) | rechte Shift-Taste liefert 1, rechte Control-Taste liefert -1, beide 0 |
| KEY(4) | linke Control-Taste liefert 1, rechte Control-Taste liefert -1, beide 0 |
| KEY(5) | Taste Cursor links liefert -1, Taste Cursor rechts liefert 1 |
| KEY(6...) | Wartet auf die Enter-taste und liefert 0 zurück |

## 8 Pseudografik

Für die Pseudografik wird jedes Zeichen in 4 „Pixel" aufgeteilt. Bei 23 Zeilen a 30 Zeichen ergibt sich so eine Arbeitsfläche von 60x46 Punkten.
Ist die Zeichenfarbe gleich 0 so werden die Pixel nur gelöscht, bei 1 entsprechend fesetzt.

### 8.1 PL Y,X

Mit dem PLOT-Befehl wird ein „Pixel" gesetzt.

```
10 PL 3,7+A
```

Zeichnet ein „Pixel" an der Position Y=3 und X=7+A mit der aktuellen Vordergrundfarbe.

### 8.2 DR Y1,X1,Y2,X2

Zeichnet eine Linie von Y1,X1 nach Y2,X2.

```
10 DR 0,0,29,47
```

Hier wird eine Linie von der linken oberen in die rechte untere Ecke gezeichnet.

### 8.3 BX Y1,X1,Y2,X2

Mit dem BX-Befehl wird ein Rechteck gezeichnet.

```
10 BX 1,1,10,10
```

### 8.4 FBX Y1,X1,Y2,X2

Mit dem FBX-Befehl wird ein gefülltes Rechteck gezeichnet. Ist Y1=Y2 oder X1=X2 werden horizontale oder vertikale Linien gezeichnet.

```
10 FBX 0,0,0,100,4
```

zeichnet eine waagerechte Linie am oberen Bildschirmrand, der Wert 100 wird zur Laufzeit auf den Bildschirmbereich begrenzt. Auch wenn die Ausgabe auf den sichtbaren bereich begrenzt ist, verbrauchen auch nicht gesetzte Pixel Rechenzeit.

## 9 Audio

### 9.1 NO n

Ein Ton mit der Tonhöhe n (Halbtonschritte ab 220 Hz aufwärts) wird ausgegeben. Bei n=0 bis 63 werden Noten ausgegeben, bei n=63 wird keine Note sondern Rauschen ausgegeben.

```
01 FOR N=0 to 63:NO N
02 SY 10:NXT
```

gibt nacheinander alle spielbaren Noten aus.

## 10 Zeit

### 10.1 SY n

Mit dem SYNC-Befehl wird auf N Bildsynchronimpuse gewartet. N kann wieder ein beliebiger Ausdruck sein.

```
10 SY 100
```

wartet ca. 2 Sekunden, bis das Programm fortgesetzt wird.

## 11 Ein- und Ausgabe

### 11.1 DIR n

Setzt die I/O-Richtung der 8 Portpins an der parallelen Schnittstelle. Eine 0 bedeutet Eingang, eine 1 Ausgang.

```
01 DIR $7
```

Pin I/O 0-2 werden als Ausgang, I/O 3 als Eingang konfiguriert.

### 11.2 OUT n

Der Wert N wird an I/O 0-3 ausegegeben. Ist der I/O als Ausgang konfiguriert, wird er entsprechend gesetzt. Ist der I/O als Eingang konfiguriert, wird der interne Pull-Up Widerstand aus- bzw. eingeschaltet. Beispiele:

```
02 OUT 7
```

setzt I/O 0-2 auf 1-Pegel und I/O 3 auf 0.

### 11.3 IN(n)

Gibt den aktuellen Status des I/O Pins n zurück. Je nach anliegendem Pegel ist das Resultat 0 oder 1.

```
03 ? IN(0)
```

gibt den Status von I/O Pin 0 aus.

### 11.4 AV(n)

Gibt den aktuellen Spannungswert des I/O Pins n zurück. Der Wertebereich liegt zwischen 0 und 255 für 0...5V..

```
04 ? AD(3)
```

gibt den Spannungswert von I/O Pin 3 aus.

## 12 Kommunikation

### 12.1 PUT n

das Byten wird an die serielle Schnittstelle ausgegeben.

```
01 PUT 10
```

gibt einen Zeilenvorschub an die serielle Schnittstelle aus.

### 12.2 GET V

Ein Zeichen von der seriellen Schnittstelle wird eingelesen und in die Varaiable V gespeichert.

```
01 GET C
```

wartet auf ein Zeichen von der seriellen Schnittstelle und speichert die in die Variable C

## 13 Speicher

### 13.1 POK adr,dat

Speichert ein Datenbyte in den internen EEPROM. Als Adressen sind 0...255 möglich.

```
02 POK V,$12
```

speichert den Wert 18 an die Adresse V im internen EEPROM.

### 13.2 EP(adr)

Die Funktion liest ein Byte aus dem internen EEPROM. Als Adressen sind 0...255 möglich.

```
02 W=EP(V)
```

liest den Wert von EEPROM-Adresse V (internes EEPROM) und speichert diesen in der Variable W.

## 14 Fehlermeldungen

### 14.1 Fehlermeldungen

Insgesamt gibt es 5 verschiedene (Fehler) Meldungen. Im Editor werden diese oben in der zweiten Zeile mit Zeilen- und Statementnummer angezeigt. Im Allgemeinen lässt sich damit der Fehler recht schnell finden, manchmal kann er sich auch am Ende der vorherigen Zeile befinden.

| | |
|---|---|
| OK | Das Programm wurde mit Sprung in eine nichtexistente Zeile beendet. |
| BREAK | Das Programm wurde mit der Tastenkombination Control (Strg) + C abgebrochen. |
| SYN ERR | Fehlende Parameter, ungültige Zeichen |
| EXPR ERR | In einer Formel befinden sich syntaktische Fehler |
| STACK ERR | zuviele oder zuwenig FOR/NXT/SUB/RET |