AVR-ChipBasic2: BASIC-Referenz

Grafik und Sound

1 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. Die 4 Pixel eines Zeichens haben immer die gleiche Hinter- und Vordergrundfarbe.
Ist die Zeichenfarbe gleich der Hintergrundfarbe des zu ändernden Zeichens, so werden die Pixel nur gelöscht, Farbinformationen werden nicht verändert. Ist die Zeichenfarbe gleich der Vordergrundfarbe, werden nur Pixel gesetzt, Farbinformationen werden auch hier nicht verändert. Gibt es keine Übereinstimmung zwischen der Zeichenfarbe und der Vorder- und Hintergrundfarbe des betreffenden Zeichens, wird die Vordergrundfarbe neu gesetzt wobei bereits gesetzte Pixel zwangsläufig mit umgefärbt werden.

1.1 PLOT Y,X(,C) [PL]

Mit dem PLOT-Befehl wird ein „Pixel" gesetzt. Dabei kann es passieren, dass benachbarte Punkte auch ihre Farbe wechseln. Zusätzlich zu den Koordinaten kann auch die Zeichenfarbe temporär gesetzt werden. Temporär deshalb, weil die Farbe nur für den aktuellen Zeichenvorgang verwendet wird.

```
10 PLOT 3,7+A
```

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

```
10 PLOT 3,7+A,6
```

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

1.2 DRAW Y1,X1,Y2,X2(,C) [DR]

Zeichnet eine Linie von Y1,X1 nach Y2,X2. Zusätzlich zu den Koordinaten kann auch die Zeichenfarbe temporär gesetzt werden.

```
10 DRAW 0,0,45,59
```

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

```
10 DRAW 0,0,45,59,4
```

Zeichnet eine grüne Linie von der linken oberen in die rechte untere Ecke.

1.3 DRAWTO Y2,X2(,C) [DTO]

Zeichnet eine Linie vom letzten Endpunkt nach Y2,X2. Zu Beginn sind die „letzten Koordinatenäuf 0,0 gesetzt, danach entsprechen sie immer den zuletzt angegebenen. Zusätzlich zu den Koordinaten kann auch die Zeichenfarbe temporär gesetzt werden.

```
10 PLOT 4,4
11 DRAWTO 10,8
```

Zeichnet eine Linie von Y=4,X=4 nach Y=10,X=8 in der aktuell eingestellten Farbe.

1.4 BOX Y1,X1,Y2,X2(,C)

Mit dem BOX-Befehl wird ein Rechteck gezeichnet. Zusätzlich zu den Koordinaten kann auch die Zeichenfarbe temporär gesetzt werden.

```
10 BOX 1,1,10,10
```

1.5 FBOX Y1,X1,Y2,X2(,C)

Mit dem FBOX-Befehl wird ein gefülltes Rechteck gezeichnet. Ist Y1=Y2 oder X1=X2 werden horizontale oder vertikale Linien gezeichnet. Zusätzlich zu den Koordinaten kann auch die Zeichenfarbe temporär gesetzt werden.

```
10 COLOR 2,0:FBOX 0,0,0,100,4
```

zeichnet eine waagerechte grüne 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.

1.6 CIRCLE Y1,X1,RY,RX(,C) [CI]

Mit dem CIRCLE-Befehl können Kreise und Ellipsen gezeichnet werden. Zusätzlich zu den Koordinaten kann auch die Zeichenfarbe temporär gesetzt werden.

```
10 CIRCLE 20,20,10,10
```

Zeichnet einen Kreis mit dem Mittelpunkt 20,20 und dem Radius 10

```
10 CIRCLE 20,20,10,6,4
```

Zeichnet eine grüne Ellipse mit dem Mittelpunkt 20,20 und den Radien 10 in Y-Richtung sowie 6 in X-Richtung.

1.7 FCIRCLE Y1,X1,RY,RX(,C) [FCI]

Mit dem CIRCLE-Befehl können ausgefüllte Kreise und Ellipsen gezeichnet werden. Zusätzlich zu den Koordinaten kann auch die Zeichenfarbe temporär gesetzt werden.

```
10 CIRCLE 20,20,10,10
```

Zeichnet einen ausgefüllten Kreis mit dem Mittelpunkt 20,20 und dem Radius 10

```
10 CIRCLE 20,20,10,6,2
```

Zeichnet eine ausgefüllte rote Ellipse mit dem Mittelpunkt 20,20 und den Radien 10 in Y-Richtung sowie 6 in X-Richtung.

1.8 LPIX n

Es werden (am oberen Bildrand beginnend) n Zeichenzeilen um ein Pixel nach links verschoben. Am rechten Rand rücken nicht gesetzte Pixel nach. Es wird nur die Pixelinformation verschoben, nicht aber die Vordergrund- und Hintergrundfarbe.

```
05 LPIX 10
```

verschiebt die obersten 20 Zeichenzeilen um 1 Pixel nach links.

2 Vollgrafik

Zusätzlich zur Pseudografik im Textmodus gibt es noch 3 Grafikmodi, die sich in Auflösung und Anzahl der darstellbaren Farben unterscheiden. Allerdings lässt der verfügbare Speicher des Controllers keine sehr hohen Auflösungen zu. Ebenfalls aus Speicherplatzgründen ist es nicht möglich, den Monitor in den Grafikmodi aufzurufen.
Werden Vorder- und Hintergrundfarbe gesetzt, zeigen diese auf den Palettenindex und geben nicht die Farbe direkt an.

2.1 VMODE n [VM]

Schaltet den Videomodus um. Dabei werden der Bildschirm gelöscht und die Textkoordinaten auf die linke obere Ecke gesetzt.

Mode	Auflösung X	Auflösung Y	Darstellbare Farben
0	Text 30	Text 23	8 Vordergrund und 8 Hintergrund
1	168	116	2 aus 8
2	120	76	4 aus 8
3	84	58	8 aus 8

Zusätzlich werden die im jeweiligen Modus genutzten Paletteneinträge vorbelegt.

2.2 PALETTE start,anzahl,c1(,c2,c3,c4) [PAL]

Mit diesem befehl wird die Farbpalette für die Videomodi 1 bis 3 eingestellt. Für Modus 1 werden nur die Einträge 0 und 1 verwendet, für Modus 2 die Einträge 0 bis 3. Die Anzahl der zu setzenden Paletteneinträge muss gleich der angegebenen Anzahl sein, ist jedoch auf 4 begrenzt, anderenfalls kommt es zu einer Fehlermeldung. Als Farbwerte müssen im Bereich (0. . . 7) angegeben werden. Nach dem Umschalten des Videomodus sind die Paletteneinträge wie folgt vorbelegt:

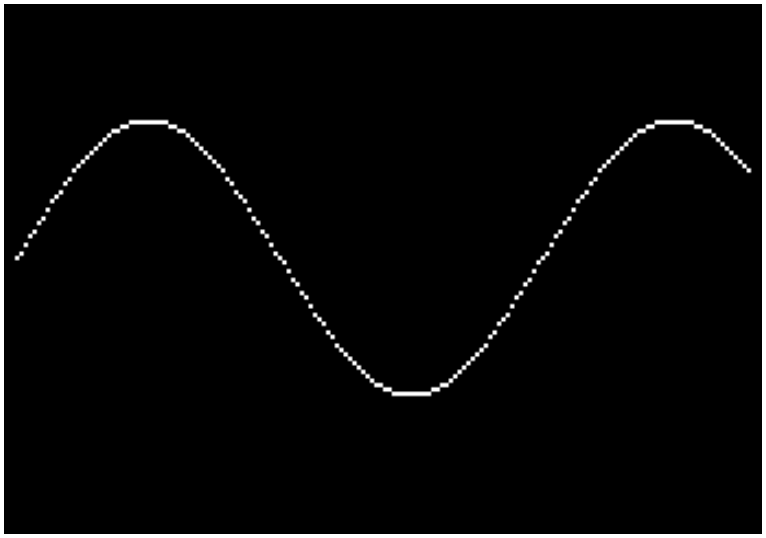
Index	Modus 1	Modus 2	Modus 3
0	schwarz (0)	schwarz (0)	schwarz (0)
1	weiss (7)	rot (2)	blau (1)
2	—	cyan (5)	rot (2)
3	—	weiss (7)	magenta (3)
4	—	—	grün (4)
5	—	—	cyan (5)
6	—	—	gelb (6)
7	—	—	weiss (7)

2.3 PLOT Y,X(,C) [PL]

Mit dem PLOT-Befehl wir ein Punkt an der Stelle Y,X gesetzt. Bei Bedarf kann zusätzlich die Zeichenfarbe mit angegeben werden.

```
01 CLS:VMODE 1
02 FOR X=1 TO 167
03 V=SIN(3*X)/6
04 PLOT 55-V,X
05 NEXT
#
```

Zeichnet eine Sinuskurve (Videomode 1).

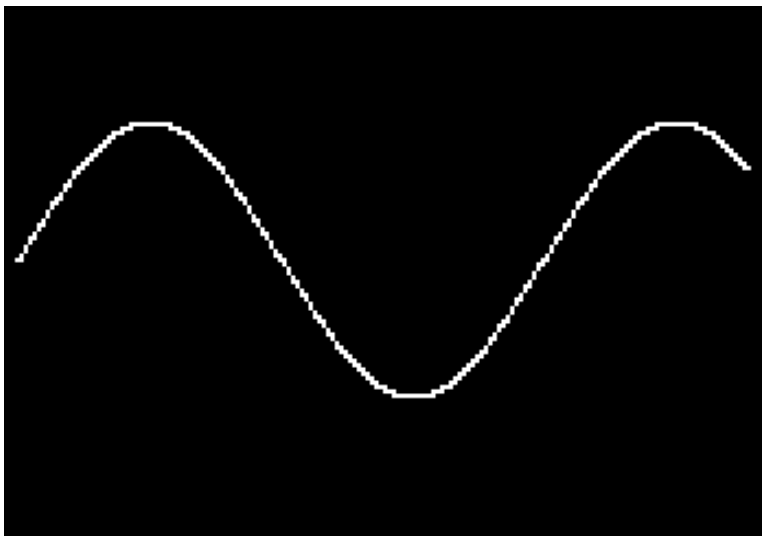


Wie man sieht, ist der Kurvenzug teilweise unterbrochen, da halt nur einzelne Punkte gesetzt werden. Um solche Unterbrechungen zu vermeiden gibt es den folgenden Befehl.

2.4 DRAWTO Y2,X2(,C) [DTO]

Zeichnet eine Linie vom zuletzt gezeichneten Pixel nach Y2,X2 in der aktuellen Vordergrundfarbe. Bei Bedarf kann zusätzlich die Zeichenfarbe mit angegeben werden.

```
01 CLS:VMODE 1,PLOT 55,0
02 FOR X=1 TO 167
03 V=SIN(3*X)/6
04 DRAWTO 55-V,X
05 NEXT
#
```



Diesmal gibt es keine Unterbrechungen. Der Plot-befehl in der ersten Zeile setzt die Anfangs-Koordinate für das erste Linienstück, fehlt sie, wird mit dem Zeichnen bei 0,0 begonnen.

2.5 DRAW Y1,X1,Y2,X2(,C) [DR]

Zeichnet eine Linie vo Y1,X1 nach Y2,X2 in der aktuellen Vordergrundfarbe. Bei Bedarf kann zusätzlich die Zeichenfarbe mit angegeben werden.

```
01 CLS:VMODE 1,PLOT 55,0
02 FOR X=1 TO 167
03 V=SIN(3*X)/6
04 DRAWTO 55-V,X
05 NEXT
06 DRAW 55,0,55,167
#
```

Hier wird die Null-Linie durch die gerade gezeichnete Sinuskurve gezogen.

2.6 BOX Y1,X1,Y2,X2(,C)

Mit dem BOX-Befehl wird ein Rechteck in der aktuellen Vordergrundfarbe gezeichnet. Bei Bedarf kann zusätzlich die Zeichenfarbe mit angegeben werden.

```
11 BOX 0,0,115,167
```

2.7 FBOX Y1,X1,Y2,X2(,C)

Mit dem FBOX-Befehl wird ein gefülltes Rechteck gezeichnet. Ist Y1=Y2 oder X1=X2 werden horizontale oder vertikale Linien gezeichnet. Bei Bedarf kann zusätzlich die Zeichenfarbe mit angegeben werden.

```
10 FBOX 0,0,0,100
```

zeichnet eine waagerechte Linie am oberen Bildschirmrand, der Wert 100 wird zur Laufzeit auf den Bildschirmbereich begrenzt, falls er auuserhalb des Bildschirms liegt.

2.8 CIRCLE Y,X,RY,RX(,C) [CI]

Zeichnet einen Kreis (eine Ellipse) mit dem Mittelpunkt Y,X und den Radien RY und RX in der aktuellen Vordergrundfarbe. Für Kreise muss RX=RY sein. Bei Bedarf kann zusätzlich die Zeichenfarbe mit angegeben werden.

```
10 CIRCLE 50,50,20,20
```

Zeichnet einen Kreis mit den Mittelpunktkoordinaten 50,50 und dem Radius 20.

2.9 FCIRCLE Y,X,RY,RX(,C) [FCI]

Zeichnet einen gefüllten Kreis (eine gefüllte Ellipse) mit dem Mittelpunkt Y,X und den Radien RY und RX in der aktuellen Vordergrundfarbe. Für Kreise muss RX=RY sein. Bei Bedarf kann zusätzlich die Zeichenfarbe mit angegeben werden.

```
10 FC 50,50,20,10
```

Zeichnet eine gefüllte Ellipse mit den Mittelpunktskoordinaten 50,50, dem Y-Radius 10 und dem X-Radius 20.

2.10 GETPIX variable,Y,X [GPX]

Ermittelt die Farbe des Pixels an der Stelle Y,X und speichert den Palettenindex (nicht die Farbe!) in der angegebenen Variablen.

```
10 GPIX P,0,0
```

Bestimmt den Palettenindex des Pixels in der oberen linken Ecke.

2.11 BCOPY [BC]

Der BCOPY Befehl ermöglicht es, rechteckige Blöcke innerhalb des Bildschirms oder ins/vom Array zu kopieren. Damit lassen sich z.B. Sprites im Grafikmodus oder Scrolling realisieren. Blöcke können von und an (fast) beliebigen Koordinaten kopiert werden, lediglich die Größe ist Beschränkungen unterworfen. Es können nur ganze Teilblöcke kopiet werden.

Videomode	Pixel je Teilblock
1	8x8
2	4x4
3	2x2

Im Videomodus 1 ist ein Teilblock 8x8 Pixel groß, um einen 16(V)x24(H) Pixel großen Block zu kopieren, müssen DX=2 und DY=3 sein, im Videomodus 2 DX=4 und DY=6 und im Videomodus 3 DX=8 und DY=12.
Der erste Wert gibt den Kopiermodus an, von diesem ist auch die Anzahl der restlichen Parameter abhängig.

Modus	Quelle	Ziel	zus.Parameter
1	Bildspeicher	Bildspeicher	Y1,X1,Y2,X2,DY,DY
2	Bildspeicher	Array	Y,X,DY,DX,ARRAYPOS
3	ARRAY	Bildspeicher	ARRAYPOS,Y,X

Im Modus 2 wird geprüft, ob die Grenzen des Arrays überschritten werden, in diesem Fall wird dann sofort abgebrochen. Mit den folgenden Formeln lässt sich der erforderliche Speicherbedarf berechnen:

Videomode 1 (8 * DX * DY) +2
Videomode 2 (4 * DX * DY) +2
Videomode 3 (2 * DX * DY) +2

DX und DY ist hierbei die Anzahl der Teilblöcke. Kopiermodus 3 sollte nur auf Arraypositionen angewendet werden, die auch sinnvolle Informationen enthalten. Diese können entweder durch Kopieren im Modus 2 oder auch durch „DATA" entstanden sein.
Die Pixelinformationen sind derart organisiert, dass in einem Byte je nach Videomode 8, 4 oder 2 Pixel codiert sind. dabei ist das niederwertigste Bit ganz links.

```
01 DATA 0,1,1,$FF,$81,$81,$81
02 DATA 6,$81,$81,$81,$FF
03 VMODE 1
04 BCOPY 3,0,0,0
05 WAIT 10
```

Das Programm zeichnet ein kleines Quadrat in die obere linke Bildschirmecke. Zuerst wird das Array gefüllt. In den beiden ersten Bytes stehen die Anzahl der Teilblöcke in Y und X-Richtung, danach folgen die Pixeldaten.

2.12 PRINT [?]

Der PRINT Befehl entspricht weitestgehend seinem Pendant im Textmodus. Unterschiedlich sind die Koordinatenangaben die hier anstelle von Zeichen in Pixeln angegeben werden. Und das Format-Attribut „Grossschrift" entspricht doppelter Zeichenhöhe und -breite. Ausserdem funktioniert der Zeilenumbruch nicht wie man es erwartet.

3 Sprites

Sprites sind einzelne Zeichen oder einfache Zeichengruppen, die sich einfach an beliebiger Stelle auf dem Bildschirm anzeigen lassen. Dabei wird der ursprüngliche Bildschirminhalt an dieser Stelle „gemerkt" und beim Verschieben oder Löschen des Sprites wieder restauriert. In der aktuellen Version können (theoretisch) bis zu 85 Sprites definiert werden, die tatsächliche Anzahl hängt von der Größe und dem verfügbaren Platz im Array ab. Sprites funktionieren nur im Textmodus und nicht in den Grafikmodi.

3.1 Sprite definieren

Dafür gibt es keinen eigenständigen Befehl (mehr), da die Daten einfach im Array liegen (Byte-Bereich). Jedes Sprite besteht aus einem Header von 5 Bytes, den eigentlichen Daten und dem Backup-Bereich. Die Belegung der Bytes ist wie folgt:

Header-Byte	Adresse	Funktion
1	N	das Kollisionsflag (0/2)
2	N+1	Y-Position (am Anfang $ff oder -1)
3	N+2	X-Position (am Anfang $ff oder -1)
4	N+3	Y-Ausdehnung in Zeichen (dy)
5	N+4	X-Ausdehnung in Zeichen (dx)

Ist das Kollisionsflag mit 0 vorbelegt, wird das Sprite bei einer Kollision nicht gezeichnet. Ist es mit 2 vorbelegt, wird das Sprite bei einer Kollision gezeichnet. Danach folgen dx*dy Bytes für die darzustellenden Zeichen, gefolgt von dx*dy Bytes für die Attribute. Die Attribut-Bytes sind Vordergrundfarbe + 16 * Hintergrundfarbe, bei Addition von 128 bleibt beim Zeichnen die ursprüngliche Hintergrundfarbe erhalten. Danach folgen noch 2*dx*dy Bytes in denen der ursprüngliche Bildschirminhalt beim Zeichnen automatisch gespeichert wird.
Ein minimales Sprite (1 Zeichen) braucht also 5+4=9 Bytes, eins mit 2x2 Zeichen schon 5+16=21 Bytes. Die Werte für die X und Y Koordinaten sollten ausserhalb des Bildschirmbereiches liegen, da ansonsten beim erstmaligen Aufrufen des Sprites der Backup-Bereich des Sprites dorthin kopiert wird.

3.2 SPRITE n,y,x [SPR]

Zeichnet das Sprite von Array-Adresse n an die Position Y,X. War es bereits an einer anderen Stelle gezeichnet, wird es vorher gelöscht. Befindet sich an der Stelle wohin das Sprite gezeichnet wird bereits ein anderes Zeichen, dessen Position in der Zeichentabelle nicht ein Vielfaches von 16 ist, wird das Kollisionsflag gesetzt. Ist Bit 1 des Kollisionsflags nicht gesetzt, wird das Sprite nicht gezeichnet. Das Kollisionsflag wird nicht automatisch zurückgesetzt. Sprites dürfen nicht über den Bildschirmrand hinaus gezeichnet werden, sonst wird mit einer Fehlermeldung abgebrochen. Werden für X und oder Y Koordinaten ausserhalb des Bildschirms (X>29, Y>22) angegeben, wird das Sprite vom Bildschirm verschwinden.

```
01 DA 0,0,-1,-1,1,1,"*",$86
02 ?@10,10;"#"
03 X=RND(29): Y=RND(22)
04 SPRITE 0,Y,X
05 IF (AR(0)&1)=1 THEN END
05 WAIT 1
06 GOTO 2
```

Zuerst werden die Sprite-Daten in das Array geschrieben (in diesem fall ein gelbes Sternchen), dann wird noch ein Hindernis an Position (10,10) gezeichnet.
Im Anschluß hüpft das Sternchen hin und her, bis es auf das Hindernis trifft. Dieses bleibt stehen, das Sternchen ist "weg". Wird die erste Zeile abgeändert in:

```
01 DA 0,2,-1,-1,1,1,"*",$86
```

Bleibt das Stecnchen stehen und das Hinedrnis ist nicht mehr zu sehen.

4 Audio

4.1 NOTE n [NO]

Ein Ton mit der Tonhöhe n (Halbtonschritte ab 220 Hz aufwärts) wird ausgegeben. Bei n=0 bis 63 werden Noten in einer „dunklen" Klangfarbe ausgegeben, bei n=64 bis 127 Noten mit einer „hellen" Klangfarbe. Bei n=255 wird keine Note sondern Rauschen ausgegeben.

```
01 FOR N=0 to 63:NOTE N
02 WAIT 5:NEXT
```

gibt nacheinander alle spielbaren Noten aus.

4.2 PLAY start,stop,speed

Ab Version 0.69 hat AVR-Chipbasic2 einen einfachen Sequenzer eingebaut. Dieser benutzt das Array als Notenspeicher und läuft im Hintergrund. Als Parameter werden die Nummer der Arrayzelle (Byte-bereich) mit der ersten zu spielenden Note und die mit der letzten zu spielenden Note sowie die die Anzahl der Halbbilder (1/50s bei PAL, 1/60s bei NTSC) zwischen zwei Noten benötigt. Die Noten zwischen Start- und End Array-Zelle werden zyklisch abgespielt, ein Aufruf mit weniger als 3 Parametern schaltet den Sequenzer (sofort) ab.

Wert	Bedeutung
0...63	Noten in einer „dunklen" Klangfarbe
64...127	Noten in einer „hellen" Klangfarbe
128...253	nicht genutzt entspricht den Werten 0...125
254	Stille, es wird keine Note gespielt
255	Rauschen

Das folgende Beispiel spielt fortlaufend 3 Töne auf und ab...

```
01 DA 0,39,44,49,44
02 PLAY 0,3,20
03 GOTO 3
```

Wird die letzte Zeile weggelassen hört man nichts oder maximal einen einzigen Ton, da beim Programmende der Sequenzer sofort wieder gestoppt wird.

4.3 PSTAT(mode)

Diese Funktion liefert Aufschluss darüber, was der Sequenzer gerade „macht". Ist der Parameter mode=1, dann wird die laufende Sequenz noch bis zum Ende gespielt und dann gestoppt. Ist der Parameter mode=2, dann wird die laufende Sequenz sofort gestoppt. Alle anderen Parameterwerte liefern nur den aktuellen Status, der wie folgt codiert ist:

Wert	Bedeutung
0	Sequenzer läuft, Noten werden gespielt
1	Sequenzer läuft noch bis zum Ende der Sequenz und wird dann stoppen
2	Sequenzer ist gestoppt