AVR-ChipBasic2: Interna

V1.00 (c) 2009 Jörg Wolfram

1 Organisation des Videospeichers

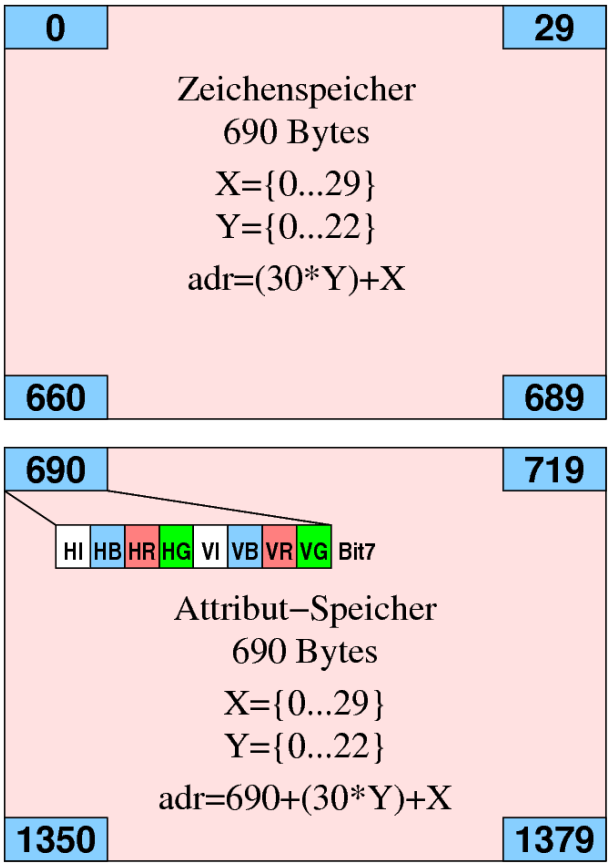
1.1 Unterschiede bei 8 und 16 Farben

Die I (Intensitäts-) Bits sind nur von Relevanz, wenn die 16-Farben-Erweiterung angeschlossen ist. Die etwas ungewöhnliche Verteilung der Bits (Intensität als LSB) ist durch die Kompatibilität zum 8-Farben Modus der ”normalen” Hardware bedingt.

1.2 Videomode 0 (Standard-Mode)

Dieser Mode ist ein Textmodus mit festem Zeichensatz von 256 Zeichen. Jedem Zeichen kann getrennt Vorder- und Hintergrundfarbe zugewiesen werden. Es gibt 23 Zeilen a 30 Zeichen, dies resultiert einfach daraus daß das Projekt zum Teil mit einem 37 cm TV-Gerät mit recht großem Rand entwickelt wurde. Im Bildspeicher finden sich nacheinander:

Anfangsadresse	Endadresse	Bytes	Funktion
0	689	690	Zeichen
690	1379	690	Attribute
1380	2759	1380	Backup für Monitor

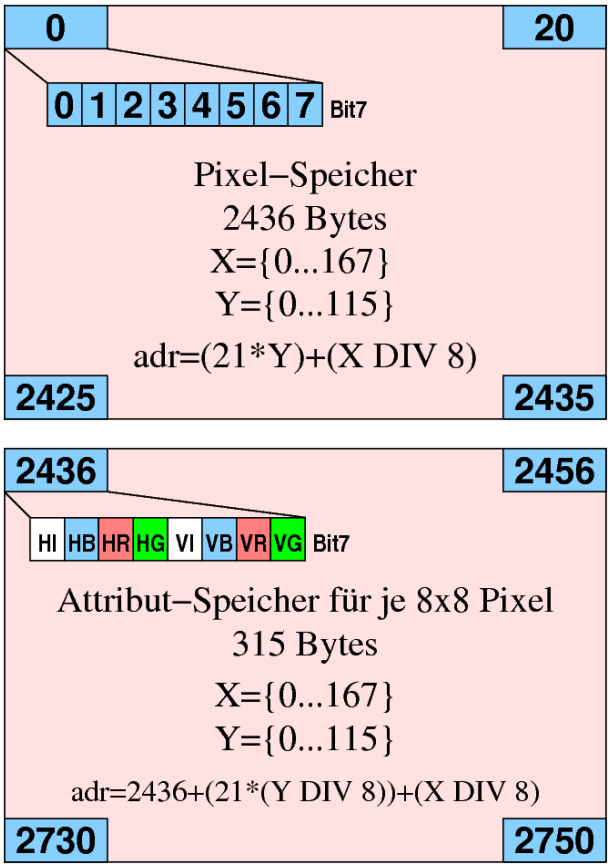


Für die Pseudografik sind die Zeichen 0...15 in 4 „Pixel” aufgeteilt. Bei 23 Zeilen a 30 Zeichen ergibt sich so eine Arbeitsfläche von 60x46 Punkten.

1.3 Videomode 1

Die Pixelauflösung beträgt 168x116 Pixel, wobei für jeweils 8x8 (am unteren Rand nur 4x8) Pixel Vorder- und Hintergrundfarbe eingestellt werden können.

Anfangsadresse	Endadresse	Bytes	Funktion
0	2435	2436	Pixelinformation
2436	2750	315	Attribute
2751	2759	9	ungenutzt

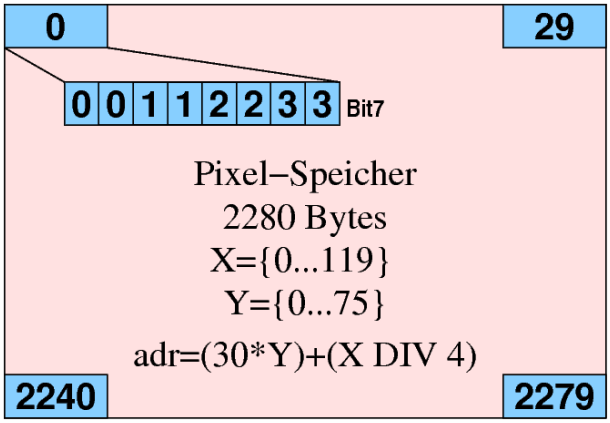


Die Pixelbytes sind von links nach rechts sowie von oben nach unten organisiert. Innerhalb des Bytes entspricht Bit 0 dem Pixel ganz links.

1.4 Videomode 2

Die Pixelauflösung beträgt 120x76 Pixel, jedes Pixel kann eine aus 4 über die Palette einstellbaren Farben annehmen.

Anfangsadresse	Endadresse	Bytes	Funktion
0	2279	2280	Pixelinformation
2280	2759	480	ungenutzt

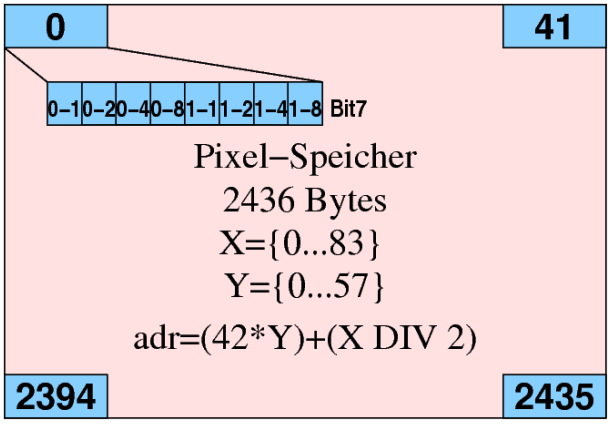


Die Pixelbytes sind von links nach rechts sowie von oben nach unten organisiert. Innerhalb des Bytes entsprechen Bit 0 und 1 dem Pixel ganz links. Jedes Pixel wird durch zwei Bits repräsentiert, die auf einen der ersten 4 Paletteneinträge zeigen. Der Paletteneintrag bestimmt dann die Farbe.

1.5 Videomode 3

Die Pixelauflösung beträgt 84x58 Pixel, jedes Pixel kann eine aus 16 über die Palette einstellbaren Farben annehmen.

Anfangsadresse	Endadresse	Bytes	Funktion
0	2435	2436	Pixelinformation
2436	2759	324	ungenutzt

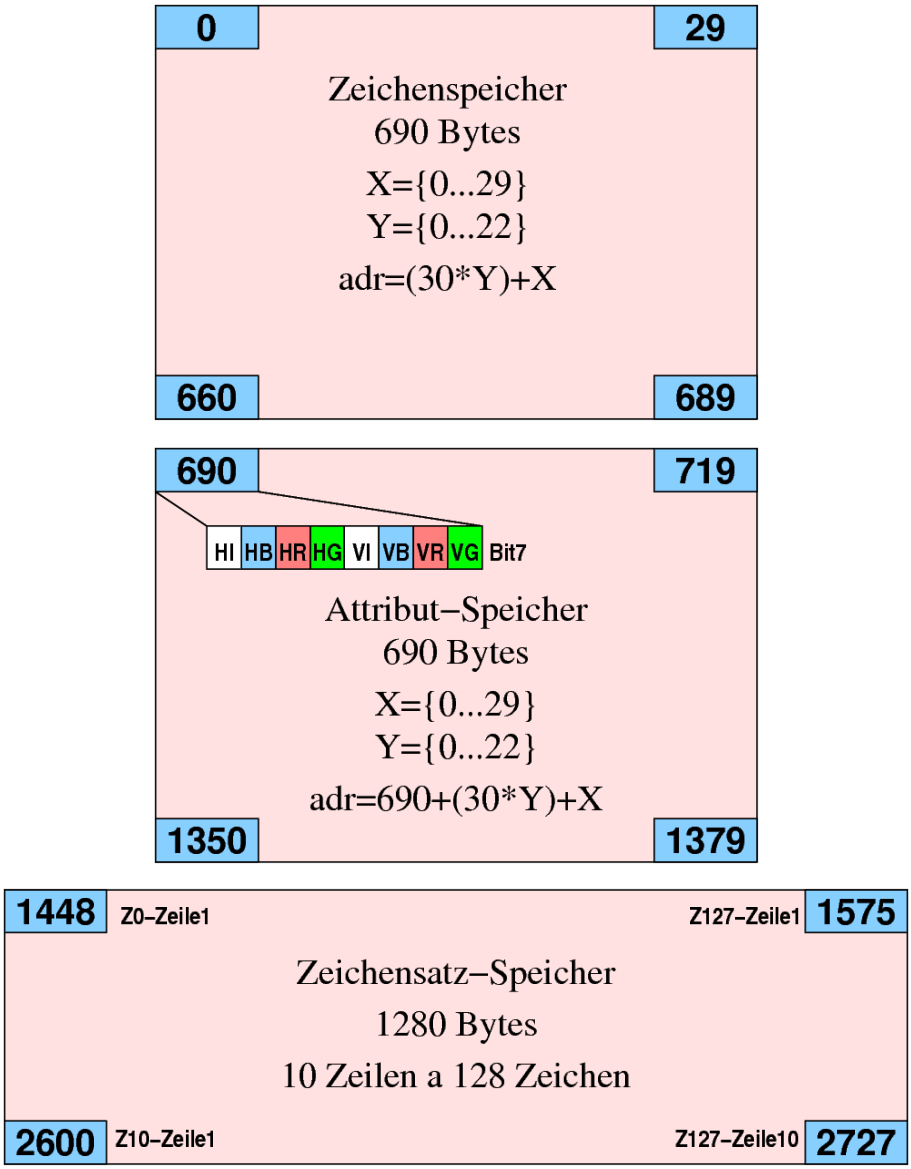


Die Pixelbytes sind von links nach rechts sowie von oben nach unten organisiert. Jedes Byte ist in 2 Nibbles zu 4 Bit unterteilt, die auf einen der 16 Paletteneinträge zeigen. Der Paletteneintrag bestimmt dann die Farbe.

1.6 Videomode 4

Dieser Mode ist ein Textmodus mit variablen Zeichensatz von 128 Zeichen. Jedem Zeichen kann getrennt Vorder- und Hintergrundfarbe zugewiesen werden. Beim Wechsel in den Videomodus werden die unteren 128 Zeichen aus dem festen Zeichensatz in den variablen Zeichensatz kopiert. Im Bildspeicher finden sich nacheinander:

Anfangsadresse	Endadresse	Bytes	Funktion
0	689	690	Zeichen
690	1379	690	Attribute
1448	2727	1280	Zeichensatz
2728	2759	32	ungenutzt

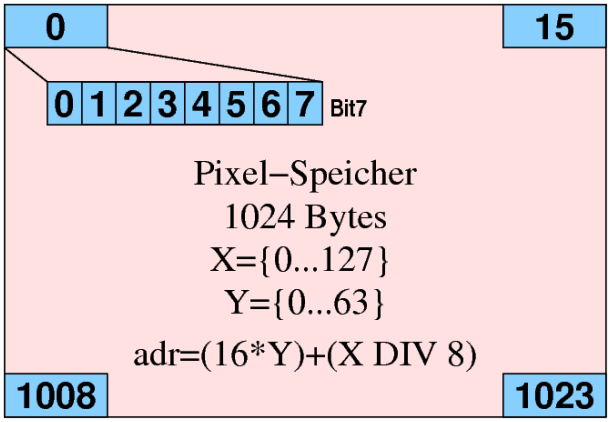


Im Zeichensatz sind die Bytes nach Zeichenzeilen geordnet, also 128 Bytes für die erste Zeichenzeile, dann 128 Bytes für die zweite Zeichenzeile...
Dabei sind die Pixel innerhalb einer Zeichenzeile noch differentiell gespeichert, Bit 7 ist das Pixel ganz links, ist Bit 6 ”0”, hat das nächste Pixeldie gleiche Farbe, bei einer ”1” ändert sie sich. Ein Byte ergibt eine Zeichenzeile von 6 Pixeln wobei Bit 0 und 1 nicht berücksichtigt werden, ein Zeichen besteht aus 10 dieser Zeichenzeilen.

1.7 Videomode 5

Die Pixelauflösung beträgt 128x64 Pixel, jedes Pixel kann eine aus 2 über die Palette einstellbaren Farben annehmen. Dieser Modus dient hauptsächlich zur Emulation von Grafik-LCD mit der entsprechenden Auflösung.

Anfangsadresse	Endadresse	Bytes	Funktion
0	1023	1024	Pixelinformation
1024	2759	1736	Backup für Monitor

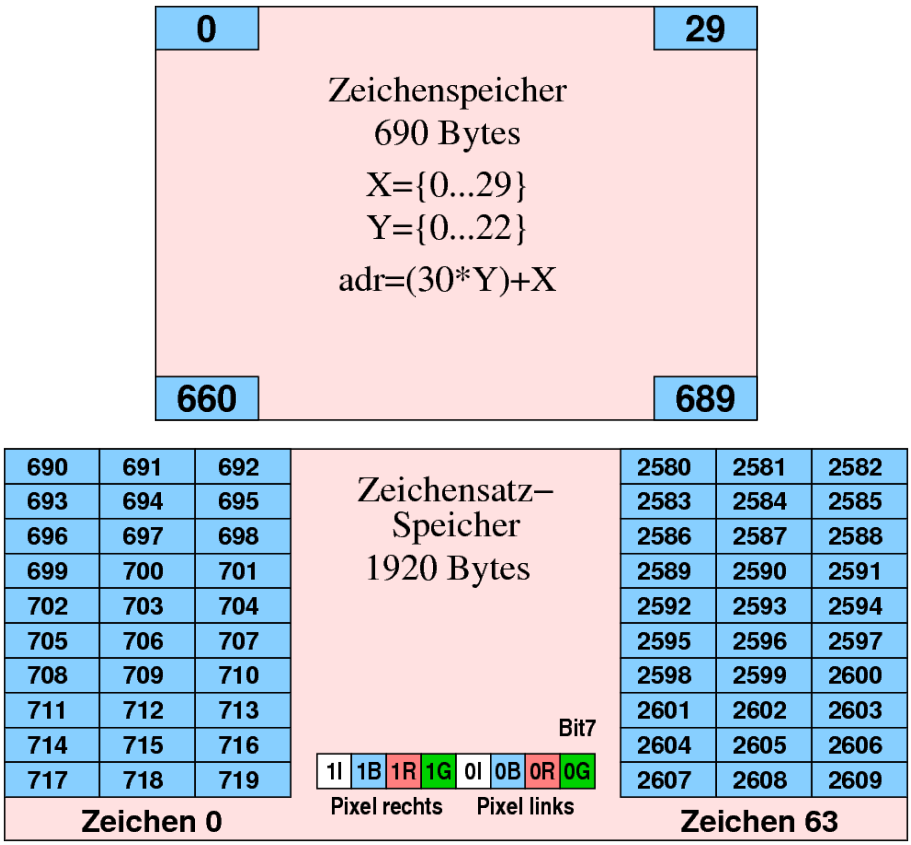


Die Pixelbytes sind von links nach rechts sowie von oben nach unten organisiert. Innerhalb des Bytes entspricht Bit 0 dem Pixel ganz links. Für jedes Pixel gibt es ganau ein Bit, welches auf einen der ersten beiden Paletteneinträge zeigt. Der Paletteneintrag bestimmt dann die Farbe.

1.8 Videomode 6

Dieser Modus ist ein Textmodus mit variablen Zeichensatz von 64 Zeichen. Jedem Pixel in jedem Zeichen kann eine der 8 Farben zugewiesen werden. Beim Wechsel in den Videomodus werden die unteren Zeichen aus dem festen Zeichensatz in den variablen Zeichensatz kopiert. Im Bildspeicher finden sich nacheinander:

Anfangsadresse	Endadresse	Bytes	Funktion
0	689	690	Zeichen
690	2609	1920	Zeichensatz
2610	2759	150	ungenutzt

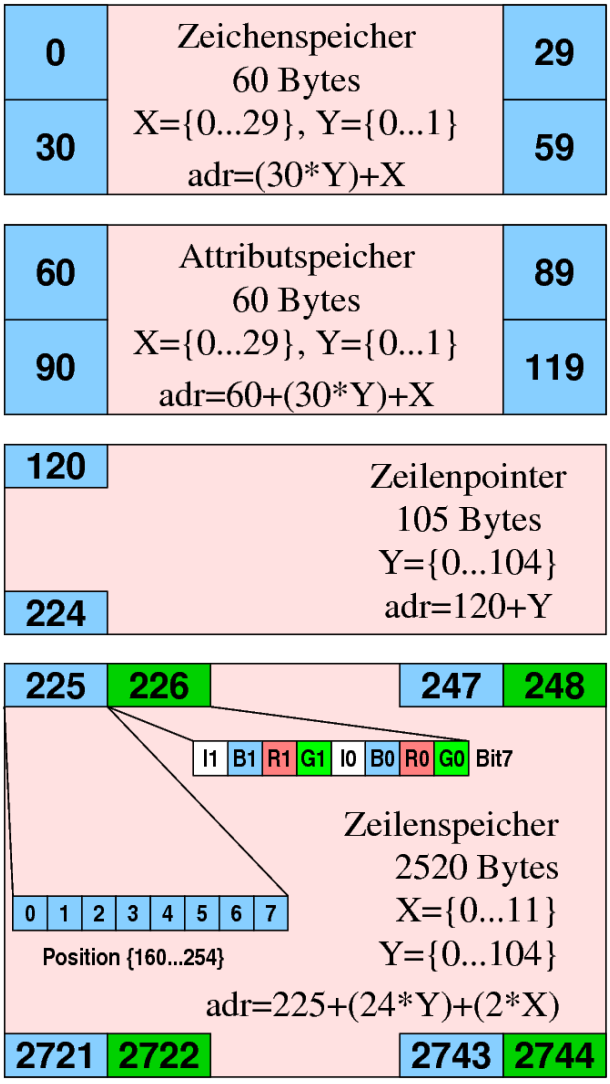


Im Zeichensatz liegen die 30 Bytes jedes Zeichens direkt hintereinander. Jedes Pixel wird durch drei Bits repräsentiert, innerhalb des Bytes entsprechen Bit 1…3 dem rechten und Bit 5…7 dem linken Pixel. Drei Bytes ergeben eine Zeichenzeile von 6 Pixeln, ein Zeichen besteht aus 10 dieser Zeichenzeilen.

1.9 Videomode 7

Diesen Modus könnte man als eine Art ”Vektormodus” bezeichnen. Am oberen und unteren Bildschirmrand gibt es jeweils eine Textzeile, die sich wie die ersten beiden Zeilen im Modus 0 verhalten. Dann hört aber die Ähnlichkeit schon auf. Danach kommen 105 Byte-Zeiger, gefolgt von 105 Zeilendefinitionen a 24 Bytes. Jeder der Zeiger gibt an, welche der 105 Zeilendefinitionen für die gerade aktuelle Zeile relevant ist. Somit können alle Zeilen ihre eigene Definition oder auch alle Zeilen die gleiche Definition haben. Werte größer als 104 sollten nicht eingestellt werden, da dann das Verhalten nicht definiert ist.
Der Bildspeicher ist folgendermaßen aufgeteilt:

Anfangsadresse	Endadresse	Bytes	Funktion
0	59	60	2 Zeilen Zeichen
60	119	60	2 Zeilen Attribute
120	224	105	105 Zeilenpointer
225	2744	2520	105 Zeilendefinitionen
2745	2759	15	ungenutzt



Die Zeilendefinitionen bestehen aus 12 Einträgen zu jeweils 2 Bytes. Das erste Byte bestimmt den Abstand zum linken Bildschirmrand wozu noch 160 addiert werden muß. Höchster Wert ist 254, so dass die horizontale Auflösung 95 Pixel beträgt. Das zweite Byte gibt die anzuzeigenden Farben an. Dabei werden im Wechsel Die Farbwerte aus dem oberen und dem unteren Nibble angezeigt. Zu beachten ist noch, dass der kleinste Abstand 160 sein muß und jeder folgende Abstandswert mindestens um 1 größer als der vorherige.

2 Das API für Binärprogramme

2.1 Allgemeines

Hmmm..., API - was ist das? API ist die Abkürzung für "Apllication Programming Interface" und dient in diesem fall dazu, auch native (mit dem Assembler erzeugte) Binärprogramme auf dem ChipBasic2 Computer laufen zu lassen.

2.2 Möglichkeiten und Einschränkungen

Da wäre zuerst die Programmgröße. Diese ist auf 3072 Bytes begrenzt, wobei noch am Anfang des Speicherbereiches 12 Bytes für den Programmnamen, 1 Byte für die Unterscheidung zu BASIC Programmen (ist bei Binärprogrammen immer "N") und 3 weitere Bytes reserviert sind.
Damit die Programme auch nach einem Update des Systems noch funktionieren, sollten direkte Unterprogrammaufrufe (ohne Umweg über die API Funktionen) vermieden werden. Ebenso sollten die Auskunfts-Funktionen genutzt werden, um die Adressen von bestimmten Speicherbereichen abzufragen, anstelle mit den gerade aktuellen Adressen zu arbeiten. Außerdem ist es nicht sinnvoll, innerhal des Programmes mit **JMP** oder **CALL** zu arbeiten, da die Programm-Adresse je nach Programmplatz variiert. Damit ist leider auch nicht so einfach, auf z.B. im Quelltext eingebunde Tabellen zuzugreifen. Allerdings ist das über das API möglich:

```
.org        0x4000
start:      .db "TestprogrammN",0xff,0xff,0xff

lese:       api_getvram                 ;setzt Y auf den Anfang des Bildspeichers
            api_getbase                 ;setzt Z auf Anfang des Programmes
            ldi   XL,LOW(daten-start)   ;Bestimmung des Offsets
            ldi   XH,HIGH(daten-start)
            call  api_dataptr           ;setzt Z auf die Adresse von "daten:"
lese_1:     lpm   r16,Z+
            cpi   r16,0
loop:       breq  loop
            st    Y+,r16
            rjmp  lese_1

daten:      .db "Mein Text",0
```

Dieses Programm schreibt "Mein Test" in die linke obere Bildschirmecke, allerdings geht so etwas auch viel einfacher:

```
.org        0x4000
start:      .db "TestprogrammN",0xff,0xff,0xff

lese:       api_thistext
daten:      .db 0,0,"Mein Text",0
loop:       rjmp  loop
```

Am Anfang müssen 12 bytes Programmname und ein "N" (0x4E) stehen, danach folgen 3 reservierte Bytes (0xFF). Das eigentliche Programm beginnt mit einem Offset von 16 Bytes (8 Words).

Es ist nicht möglich, eigene Interrupt-Routinen zu einzubinden und ein Sperren der Interrupts hat die Abschaltung der Bild- und Tonausgabe zur Folge. Weiterhin sollten die Register r8 (aktuelle Bildschirmzeile LOW) und r9 (aktuelle Bildschirmzeile HIGH) nach Möglichkeit nicht beschrieben werden.

2.3 Build-Prozess zum Erstellen einer eigenen Applikation

Das Erstellen eines Binärprogrammes unterscheidet sich etwas von der ”normalen ” Vorgehensweise. Die nachfolgende Aufzählung beschreibt das Vorgehen unter Linux und wird unter anderen Betriebssystemen nur eingeschränkt oder nicht funktionieren. Folgende Schritte sind dabei notwendig:

1. Erstellung des Programmes
2. Assemblierung mit **AVRA**
3. Umwandlung in das Binärformat mit **hex2bin**
4. Kontrolle, dass die Binärdatei maximal 3072 Bytes groß ist
5. Erzeugung einer beliebigen Datei, welche zusammen mit der Binärdatei aus dem vorigen Schritt genau 3072 Bytes lang ist
6. Die erzeugte Datei an die Binärdatei anhängen
7. Die im Schritt 5 erzeugte Datei kann nun wieder gelöscht werden

Um das nicht jedesmal von Hand machen zu müssen, gibt es im **API**-Verzeichnis ein kleines Build-Script, welches einen Großteil der Arbeit abnimmt. Aufgerufen wird es mit **./build.pl name**, wobei **name** für die ASM-Datei (ohne .asm) steht. Nach Abschluss sollte im Verzeichnis eine Datei **name.bin** stehen, die dann zum AVR-Computer übertragen werden kann.

3 Verzeichnis der API-Funktionen

3.1 Aufbau der Funktionstabelle

Funktionsname	Name der API-Funktion, entspricht auch dem Makro-Aufruf
Aufruf ohne Makro	Aufruf der Funktion, wenn ohne Makros gearbeitet wird
Funktionsbeschreibung	Kurzbeschreibung der Funktion
Parameter	Registerzuordnung zu den Parametern
Rückgabewerte	Registerzuordnung zu den Rückgabewerten
Register	Register, deren Inhalt verändert wird (ausser Rückgabewerte)
Videomodi	Videomodi, in denen die Funktion sinnvoll einsetzbar ist

Zum derzeitigen Zeitpunkt sind noch nicht alle API-Funktionen dokumentiert, die Vervollständigung wird im Laufe der Zeit erfolgen.

3.2 Sichern und Restaurieren von Registern

Funktionsname	**api_pushxyz**
Aufruf ohne Makro	call api_pushxyz
Funktionsbeschreibung	sichert die Register X,Y und Z auf den Stack
Parameter	—
Rückgabewerte	—
Register	R0, R1

Funktionsname	**api_popxyz**
Aufruf ohne Makro	jmp api_popxyz
Funktionsbeschreibung	holt die Register X,Y und Z vom Stack und führt ein RET aus
Parameter	—
Rückgabewerte	—
Register	X, Y, Z

Funktionsname	**api_pushregs**
Aufruf ohne Makro	call api_pushregs
Funktionsbeschreibung	sichert die Register X,Y und Z sowie R20-R23 auf den Stack
Parameter	—
Rückgabewerte	—
Register	R0, R1

Funktionsname	**api_popxyz**
Aufruf ohne Makro	jmp api_popregs
Funktionsbeschreibung	holt die Register X,Y und Z sowie R20-R23 vom Stack und führt ein RET aus
Parameter	—
Rückgabewerte	—
Register	X, Y, Z, R20, R21, R22, R23

3.3 Auskunftsfunktionen

Funktionsname	**api_getvram**
Aufruf ohne Makro	call api_getvram
Funktionsbeschreibung	bestimmt die Anfangsadresse des Bildspeichers
Parameter	—
Rückgabewerte	Y zeigt auf den Anfang des Bildspeichers
Register	—

Funktionsname	**api_getsysram**
Aufruf ohne Makro	call api_getsysram
Funktionsbeschreibung	bestimmt die Anfangsadresse des System-Speichers
Parameter	—
Rückgabewerte	Y zeigt auf den Anfang des System-Speichers
Register	—

Funktionsname	**api_getpal**
Aufruf ohne Makro	call api_getpal
Funktionsbeschreibung	bestimmt die Anfangsadresse des Paletten-Speichers
Parameter	—
Rückgabewerte	Y zeigt auf den Anfang des Paletten-Speichers
Register	—

Funktionsname	**api_getarray**
Aufruf ohne Makro	call api_getarray
Funktionsbeschreibung	bestimmt die Anfangsadresse des Arrays
Parameter	—
Rückgabewerte	Y zeigt auf den Anfang des Arrays
Register	—

Funktionsname	**api_getchart0**
Aufruf ohne Makro	call api_getchart0
Funktionsbeschreibung	bestimmt die Anfangsadresse der Zeichentabelle für Font 0 im Flash
Parameter	—
Rückgabewerte	Z zeigt auf den Anfang der Zeichentabelle für Font 0
Register	—

Funktionsname	**api_getchart1**
Aufruf ohne Makro	call api_getchart1
Funktionsbeschreibung	bestimmt die Anfangsadresse der Zeichentabelle für Font 1 im Flash
Parameter	—
Rückgabewerte	Z zeigt auf den Anfang der Zeichentabelle für Font 1
Register	—

Funktionsname	**api_getbase**
Aufruf ohne Makro	call api_getbase
Funktionsbeschreibung	bestimmt die Anfangsadresse des aufrufenden Programmes im Flash
Parameter	—
Rückgabewerte	Z zeigt auf den Anfang des aufrufenden Programmes
Register	—

Funktionsname	**api_dataptr**
Aufruf ohne Makro	call api_dataptr
Funktionsbeschreibung	bestimmt die Adresse eines Datenblocks im aufrufenden Programm
Parameter	X = Offset
Rückgabewerte	Z zeigt auf die absolute Adresse eines Datenblocks im aufrufenden Programm
Register	—

3.4 Einstellungen

Funktionsname	**api_setcolor**
Aufruf ohne Makro	call api_setcolor
Funktionsbeschreibung	stellt Vorder- und Hintergrundfarbe ein
Parameter	XL (LOW-Nibble=FG, HIGH-Nibble=hg [igrbIGRB])
Rückgabewerte	—
Register	—

Funktionsname	**api_setfcolor**
Aufruf ohne Makro	call api_setfcolor
Funktionsbeschreibung	stellt Vordergrundfarbe ein, Hintergrundfarbe bleibt erhalten
Parameter	XL (LOW-Nibble=FG [xxxxIGRB])
Rückgabewerte	—
Register	—

Funktionsname	**api_setborder**
Aufruf ohne Makro	call api_setborder
Funktionsbeschreibung	stellt die Randfarbe ein
Parameter	XL (LOW-Nibble=Farbe [xxxxIGRB])
Rückgabewerte	—
Register	—

Funktionsname	**api_setvmode**
Aufruf ohne Makro	call api_setvmode
Funktionsbeschreibung	stellt den Videomode ein
Parameter	XL
Rückgabewerte	—
Register	X

Funktionsname	**api_setpalette**
Aufruf ohne Makro	call api_setpalette
Funktionsbeschreibung	setzt einen Paletteneintrag
Parameter	XL (LOW-Nibble=Farbe [xxxxIGRB]), XH Palettenindex (0...15)
Rückgabewerte	—
Register	—

Funktionsname	**api_setchannel**
Aufruf ohne Makro	call api_setchannel
Funktionsbeschreibung	ändert den Ausgabekanal (siehe PRINT in der BASIC-Referenz)
Parameter	XL = Ausgabekanal
Rückgabewerte	—
Register	—

Funktionsname	**api_setfont**
Aufruf ohne Makro	call api_setfont
Funktionsbeschreibung	ändert den Textfont in den Grafikmodi
Parameter	XL = Font (0/1, es wird nur Bit 0 ausgewertet)
Rückgabewerte	—
Register	—

3.5 Programmsteuerung

Funktionsname	**api_sync**
Aufruf ohne Makro	call api_sync
Funktionsbeschreibung	wartet auf das Ende des gerade dargestellten Halbbildes (unterer Rand des genutzen Bereiches)
Parameter	—
Rückgabewerte	—
Register	—

Funktionsname	**api_fast**
Aufruf ohne Makro	call api_fast
Funktionsbeschreibung	schaltet die Bildausgabe ab
Parameter	—
Rückgabewerte	—
Register	—

Funktionsname	**api_slow**
Aufruf ohne Makro	call api_slow
Funktionsbeschreibung	schaltet die Bildausgabe ein
Parameter	—
Rückgabewerte	—
Register	—

3.6 Textausgabe

Funktionsname	**api_clrscr**
Aufruf ohne Makro	call api_clrscr
Funktionsbeschreibung	löscht den Bildschirm mit der aktuellen Farbeinstellung
Parameter	—
Rückgabewerte	—
Register	R20
Videomodi	0. . . 6

Funktionsname	**api_clearvec**
Aufruf ohne Makro	call api_clearvec
Funktionsbeschreibung	löscht den Vektorbereich im Grafikmode 7
Parameter	—
Rückgabewerte	—
Register	—
Videomodi	7

Funktionsname	**api_cleartext**
Aufruf ohne Makro	call api_cleartext
Funktionsbeschreibung	löscht den Textbereich im Grafikmode 7 mit der aktuellen Farbeinstellung
Parameter	—
Rückgabewerte	—
Register	—
Videomodi	7

Funktionsname	api_gotoxy
Aufruf ohne Makro	call api_gotoxy
Funktionsbeschreibung	setzt die Position des Text-Cursors (auch in den Grafikmodi)
Parameter	XL = X-Position, XH = Y-Position, In den Grafikmodi als Pixelposition
Rückgabewerte	—
Register	Im Textmodus werden XL und XH auf die maximalen Bildschirmkoordinaten begrenzt
Videomodi	alle

Bei den Ausgabefunktionen bestimmt R25 den Modus, die Werte sind beim PRINT Befehl näher erläutert. Ebenso bestimmt der eingestellte Channel, wohin ausgegeben wird.

Funktionsname	api_outchar
Aufruf ohne Makro	call api_outchar
Funktionsbeschreibung	gibt ein Zeichen aus
Parameter	R20 = Zeichen, R25 = Mode
Rückgabewerte	—
Register	R20, R21
Videomodi	alle

Funktionsname	api_newline
Aufruf ohne Makro	call api_newline
Funktionsbeschreibung	gibt einen Zeilenvorschub aus
Parameter	—
Rückgabewerte	—
Register	R20, R21
Videomodi	0, 4, 6

Funktionsname	api_outdez
Aufruf ohne Makro	call api_outdez
Funktionsbeschreibung	gibt den Inhalt von X dezimal aus (-32768...32767)
Parameter	X= 16-Bit Wert, R25 = Mode/Format
Rückgabewerte	—
Register	R20, R21
Videomodi	alle

Funktionsname	api_outhex
Aufruf ohne Makro	call api_outhex
Funktionsbeschreibung	gibt den Inhalt von X hexadezimal aus (00...FF oder 0000...FFFF)
Parameter	X= 16-Bit Wert, R25 = Mode/Format
Rückgabewerte	—
Register	R20, R21
Videomodi	alle

Funktionsname	api_romtext
Aufruf ohne Makro	call api_romtext
Funktionsbeschreibung	gibt einen nullterminierten String aus dem Flash aus
Parameter	Z = Zeiger auf den String im Flash
Rückgabewerte	—
Register	Z, R20, R21
Videomodi	alle

Funktionsname	**api_thistext**
Aufruf ohne Makro	call api_thistext
Funktionsbeschreibung	gibt einen nullterminierten String aus dem Flash aus, der direkt auf den Aufruf folgt.
Parameter	Wenn das erste Byte 0xFF ist, folgt direkt der auszugebende String, andernfalls wird das erste Byte als Y-Koordinate und das zweite als X-Koordinate gewertet, bevor die eigentliche Zeichenkette folgt.
Rückgabewerte	—
Register	Z, R20, R21
Videomodi	alle

Funktionsname	**api_cbox**
Aufruf ohne Makro	call api_cbox
Funktionsbeschreibung	Löscht eine Rechteckfläche mit der eingestellten Farbe
Parameter	YL = X1, YH = Y1, ZL = X2, ZH = Y2
Rückgabewerte	—
Register	—
Videomodi	0, 1, 4

Funktionsname	**api_ibox**
Aufruf ohne Makro	call api_ibox
Funktionsbeschreibung	Invertiert eine Rechteckfläche (Tauscht Vorder- mit Hintergrundfarbe)
Parameter	YL = X1, YH = Y1, ZL = X2, ZH = Y2
Rückgabewerte	—
Register	—
Videomodi	0, 1, 4

Funktionsname	**api_scroll**
Aufruf ohne Makro	call api_scroll
Funktionsbeschreibung	Scrollt den inneren Bildschirmbereich, siehe SCROLL Befehl
Parameter	R20 = Richtung (0=nach oben, 1=nach rechts, 2=nach unten, 3=nach links)
Rückgabewerte	—
Register	—
Videomodi	0, 4