AVR-ChipBasic2 - Treiber

V1.48 (c) 2006-2014 Jörg Wolfram

1 Videotreiber ohne externe Hardware

1.1 24x40 Zeichen Treiber (Bibliothekscode 0x80)

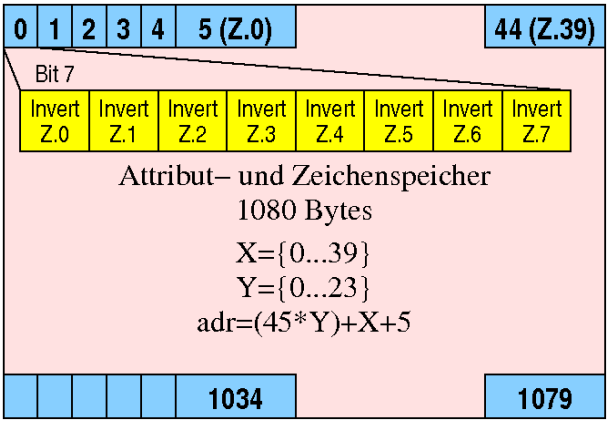
1.1.1 Allgemeines

Diesen Modus erlaubt ein Darstellung von Text und Pseudografik mit eine Auflösung von 24 Zeilen zu je 40 Zeichen. Allerdings ist die Darstellung nur monochrom (grün auf schwarz) möglich, die Ausgabepins für den roten und blauen Farbkanal werden hochohmig geschaltet.

Eine Zeile im Bildschirmspeicher besteht aus 45 Bytes. Zuerst kommen 5 Bytes bei denen jeweils 1 Bit für die Invertierung eines der nachfolgenden Zeichen verantwortlich ist. Danach folgen 40 Zeichen-Bytes, Bit 7 der Zeichen wird ignoriert. Die Zeichen sind mit 8 Pixeln horizontal und 10 Pixeln Vertikal definiert, der Zeichensatz mit insgesamt 128 Zeichen befindet sich im RAM und kann auch geändert werden. Da der Zeichensatz vom Speicherbedarf her nicht mehr in den Treiber gepasst hat, muß dieser auf den Programmplatz 7 geladen werden. Ein passender Zeichensatz ist **charset10.bin**.

Der Bildspeicher ist folgendermaßen aufgeteilt:

Anfangsadresse	Endadresse	Bytes	Funktion
0	1079	1080	24 Zeilen Zeichen
1080	1319	240	z.T.intern benutzt
1320	2599	1280	Zeichensatz
2600	2759	160	ungenutzt



Der Zeichensatz im RAM ist zeilenweise aufgebaut, zuerst 128 Bytes für die erste Zeile, dann 128 Bytes für die zweite Zeile und so weiter bis zu 128 Bytes für die zehnte und damit letzte Zeile. Der Zeichensatz auf Programmplatz 7 der als Quelle dient, hat eine ähnliche Struktur, nur dass hier eine Zeile 256 Bytes hat. Das heisst auch dass der Treiber nur die unteren 128 Zeichen des Zeichensatzes nutzt.

1.1.2 Funktionen

Der aktuelle Treiber hat keine zusätzlichen Funktionen eingebaut, die Ansteuerung erfolgt über die ”normalen” BASIC Ausgabebefehle. Um den Treiber zu aktivieren, muß nur mittels **VMODE 7** in den USER-Videomode gewechselt werden. Dabei wird dann der Zeichensatz von Programmplatz 7 in den RAM kopiert. Mit **COLOR 0** werden die Zeichen invertiert ausgegeben, **COLOR 1** schaltet in den nichtinvertierten Modus zurück.

Ebenfalls ist es möglich, 16 (0x00...0x0f) benutzerdefinierte Zeichen zu nutzen. Dazu ist der I/O Bereich von 0xa00 bis 0xaff reserviert. Dazu sind jeweils 16 aufeianderfolgende Bytes für ein Zeichen verantwortlich, genutzt werden nur die ersten 10 Bytes. Innerhalb der Bytes werden die Bits 7...0 benutzt, wobei Bit 7 dem am weitesten links liegenden Pixel entspricht. Alternativ kann auch mittels **VPOKE** direkt in den Bildschirmspeicher geschrieben werden, hier gibt es aber keinen linearen Zusammenhang zwischen Zeichenbytes und Adresse wie beim **OUT** Befehl, der bereits die Umrechnung der Adressen vornimmt.

1.2 24x50 Zeichen Treiber (Bibliothekscode 0x81)

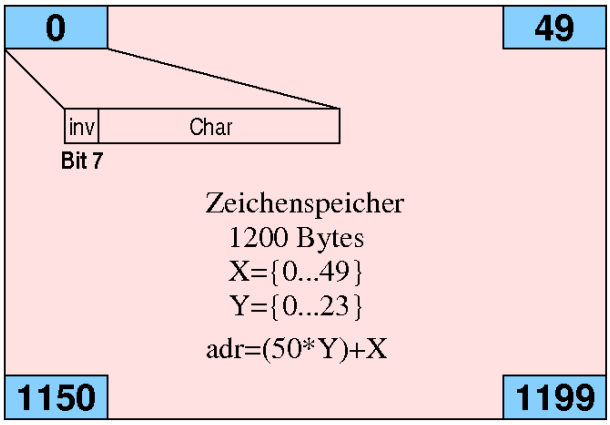
1.2.1 Allgemeines

Diesen Modus erlaubt ein Darstellung von Text und Pseudografik mit eine Auflösung von 24 Zeilen zu je 50 Zeichen. Allerdings ist die Darstellung nur monochrom (grün auf schwarz) möglich, die Ausgabepins für den roten und blauen Farbkanal werden hochohmig geschaltet.

Eine Zeile im Bildschirmspeicher besteht aus 50 Bytes. Der dazugehörige Zeichensatz mit 6 Pixeln horizontal und 10 Pixeln vertikal muss auf Programmplatz 7 geladen werden. Er enthält 256 Zeichen wobei die Zeichen ab 128 denen von 0...127 entsprechen aber invertiert dargestellt werden. Ein passender Zeichensatz ist **charset_50z.bin**.

Der Bildspeicher ist folgendermaßen aufgeteilt:

Anfangsadresse	Endadresse	Bytes	Funktion
0	1199	1200	24 Zeilen Zeichen
1200	2759	1560	ungenutzt



Der Zeichensatz im Flash ist zeilenweise aufgebaut, zuerst 256 Bytes für die erste Zeile, dann 256 Bytes für die zweite Zeile und so weiter bis zu 256 Bytes für die zehnte und damit letzte Zeile. Der Zeichensatz muss auf Programmplatz 7 stehen, die Anfangsadresse ist im Treiber fest codiert.

1.2.2 Funktionen

Der aktuelle Treiber hat keine zusätzlichen Funktionen eingebaut, die Ansteuerung erfolgt über die ”normalen” BASIC Ausgabebefehle. Um den Treiber zu aktivieren, muß nur mittels **VMODE 7** in den USER-Videomode gewechselt werden. Mit **COLOR 0** werden die Zeichen invertiert ausgegeben, **COLOR 1** schaltet in den nichtinvertierten Modus zurück.

1.3 24x60 Zeichen Treiber (Bibliothekscode 0x82)

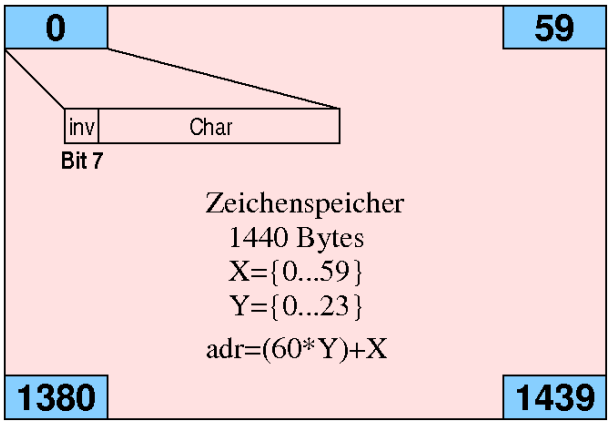
1.3.1 Allgemeines

Diesen Modus erlaubt ein Darstellung von Text und Pseudografik mit eine Auflösung von 24 Zeilen zu je 60 Zeichen. Allerdings ist die Darstellung nur monochrom (grün auf schwarz) möglich, die Ausgabepins für den roten und blauen Farbkanal werden hochohmig geschaltet.

Eine Zeile im Bildschirmspeicher besteht aus 50 Bytes. Der dazugehörige Zeichensatz mit 6 Pixeln horizontal und 10 Pixeln vertikal muss auf Programmplatz 7 geladen werden. Er enthält 256 Zeichen wobei die Zeichen ab 128 denen von 0...127 entsprechen aber invertiert dargestellt werden. Ein passender Zeichensatz ist **charset_60z.bin**.

Der Bildspeicher ist folgendermaßen aufgeteilt:

Anfangsadresse	Endadresse	Bytes	Funktion
0	1439	1440	24 Zeilen Zeichen
1440	2759	1320	ungenutzt



Der Zeichensatz im Flash ist zeilenweise aufgebaut, zuerst 256 Bytes für die erste Zeile, dann 256 Bytes für die zweite Zeile und so weiter bis zu 256 Bytes für die zehnte und damit letzte Zeile. Der Zeichensatz muss auf Programmplatz 7 stehen, die Anfangsadresse ist im Treiber fest codiert. Das letzte Pixel jedes Zeichens wird 2 mal wiederholt, dadurch sind z.B. die Pseudografiksymbole unsymmetrisch angelegt.

1.3.2 Funktionen

Der aktuelle Treiber hat keine zusätzlichen Funktionen eingebaut, die Ansteuerung erfolgt über die ”normalen” BASIC Ausgabebefehle. Um den Treiber zu aktivieren, muß nur mittels **VMODE 7** in den USER-Videomode gewechselt werden. Mit **COLOR 0** werden die Zeichen invertiert ausgegeben, **COLOR 1** schaltet in den nichtinvertierten Modus zurück.

1.4 Tile/Sprite Mode Treiber (Bibliothekscode 0x90)

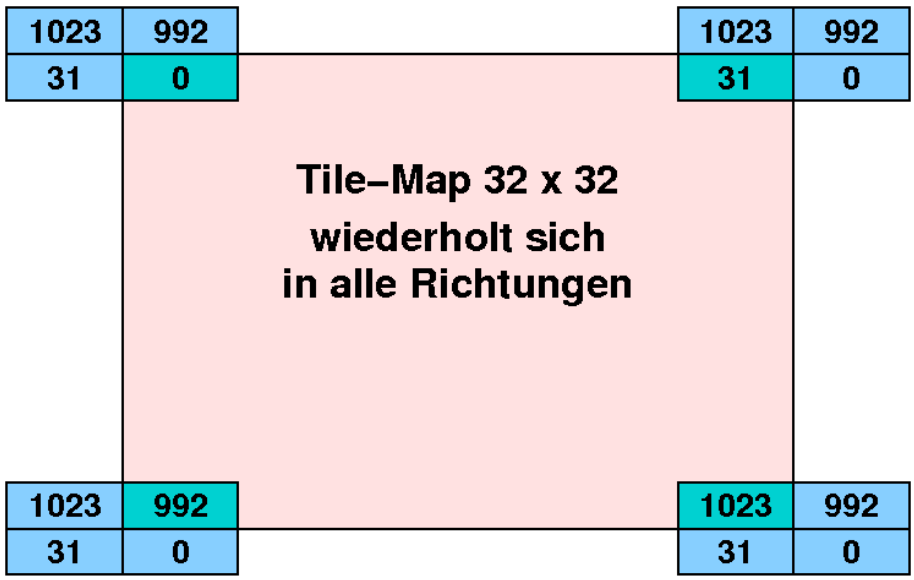
1.4.1 Allgemeines

Der Tile-und-Sprite Treiber ist in erster Linie zum Programmieren von Spielen gedacht. Es gibt einen Tilemap der sich sowohl in X als auch in Y-Richtung wiederholt. Von diesem Map ist ein Fenster von 16 Tiles horizontal und 13 Tiles Vertikal sichtbar, die Position dieses Fensters lässt sich pixelweise über den gesamten Bereich verschieben. Jede der 1024 Tile-Positionen enthält einen Index auf eine der 49 Tiles. Jede dieser Tiles (”Fliesen”) besteht aus 8 x 8 Pixeln, von denen jedes eine der 16 darstellbaren Farben haben kann. Neben umfangreichen Tile-Manipulationsmöglichkeiten gibt es auch noch Software-Sprites die selbst aus einer Tile bestehen und sich im gesamten Tilemap pixelweise bewegen können. Am oberen und unteren Bildschirmrand gibt es jeweils noch eine Textzeile, die sich wie die ersten beiden Zeilen im Modus 0 verhalten.

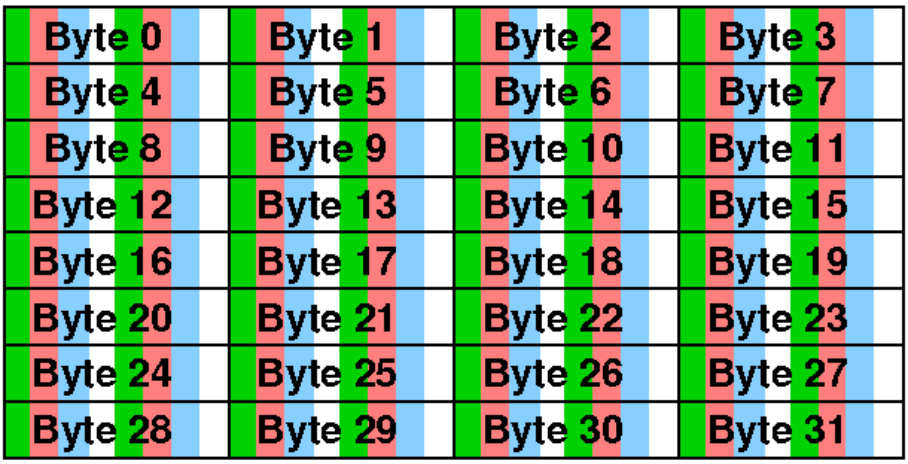
Der Bildspeicher ist folgendermaßen aufgeteilt:

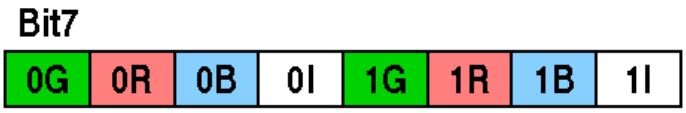
Anfangsadresse	Endadresse	Bytes	Funktion
0	39	40	System
40	1063	1024	Tilemap
1064	2599	1568	49 Tiledefinitionen
2632	2691	60	2 Zeilen Zeichen
2692	2751	60	2 Zeilen Attribute
2752	2759	8	nicht benutzt

Der Tilemap hat eine Größe von 32 mal 32 Tiles und wiederholt sich sowohl in X als auch in Y-Richtung.



Jede der 49 Tiles von je 8 x 8 Pixeln wird durch 32 Bytes repräsentiert, wobei jeweils 4 Bit für ein Pixel stehen.





1.4.2 Systemvariablen

Die Systemvariablen liegen alle in den ersten 40 Bytes der Bildspeichers und lassen sich mit **VPOKE** und **VPEEK()** ansprechen.

Adresse	Größe	Funktion
0	Byte	Y-Koordinate (Punkt links oben) des sichtbaren Ausschnitts
1	Byte	X-Koordinate (Punkt links oben) des sichtbaren Ausschnitts
4	Word	Anfangsadresse des Sprite-Bereiches im Speicher, wird durch Funktion 24 Initialisiert
6	Byte	Anzahl der initialisierten Sprites, wird durch Funktion 24 Initialisiert

1.4.3 Die Sprite-Struktur im Array

Jedes der Sprites belegt 16 Bytes im Array

Offset	Bedeutung
0	Neue Y-Koordinate
1	Neue X-Koordinate
2	Alte Y-Koordinate
3	Alte X-Koordinate
4	Sprite-Tile
5	Beginn der 4 Backup-Tiles
6	Transparenz-Farbe
7	sichtbar (0/1)
8	Sprite aktiv (0/1)
9	Kollisionsflag
10	horizontale Schrittweite
11	vertikale Schrittweite
12	Tile-Index 1 für Restore
13	Tile-Index 2 für Restore
14	Tile-Index 3 für Restore
15	Tile-Index 4 für Restore

1.4.4 Funktionen

Um den Treiber zu aktivieren, muß nur mittels **VMODE 7** in den USER-Videomode gewechselt werden. Die Ansteuerung der beiden Textzeilen am oberen und unteren Bildschirmrand erfolgt über die ”normalen” BASIC Ausgabebefehle. Die Zeile am oberen Rand entspricht dabei der Textzeile 0 und die am unteren Rand der Textzeile 1.
Weiterhin stehen noch folgende Funktionen über **CALL** zur Verfügung:

Funktionsnummer	**Funktion**
9	Tile-Map mit konstantem Tile-Index füllen
10	Tile-Map mit einer Tile nach Bitmap setzen
11	Tile-Map Bereich mit konstantem Tile Index füllen
12	Einen Tile-Index im Tile-Map setzen
13	Einen Tile-Index im Tile-Map abfragen
14	Tile mit Farbe füllen
15	Vordefiniertes Muster in Tile zeichnen
16	Array-definiertes Muster in Tile zeichnen
17	Muster aus Zeichensatz in Tile zeichnen
18	Tile horizontal spiegeln
19	Tile vertikal spiegeln
20	Tile 90 Grad im Uhrzeigersinn drehen
21	Tile 90 Grad entgegen dem Uhrzeigersinn drehen
22	Tile kopieren
23	Tiles zusammenfügen
24	2 Tiles rotieren
25	Sprites initialisieren
26	Sprite definieren
27	Sprite positionieren
28	Sprite aktivieren
29	Sprite deaktivieren
30	Ein einzelnes Sprite zeichnen
31	Ein gezeichnetes Sprite löschen
32	Alle Sprites sequentiell zeichnen
33	Alle Sprites sequentiell löschen
34	Alle Sprites-Koordinaten neu berechnen
35	Alle Sprites bewegen
36	—
37	—

1.4.5 Funktion 9 : Tile-Map mit konstantem Tile-Index füllen

Diese Funktion setzt den gesamten Tile-Bereich auf die angegebene Tile.

Parameter	**Bedeutung**
Parameter 1	Tile-Nummer (0...48)
Parameter 2	—
Parameter 3	—
Rückgabewert	0

1.4.6 Funktion 10 : Tile-Map mit einer Tile nach Bitmap setzen

Diese Funktion setzt im gesamten Tile-Bereich auf die angegebene Tile, wenn das korrespondierende Bit im Array auf "1" gesetzt ist. Der Bitmap-Bereich im Array umfasst 128 Bytes und ist wie folgt organisiert:

- Die 32 Zeilen sind von oben nach unten organisiert
- Jede Zeile besteht aus 4 Bytes, von links nach rechts
- Innerhalb jedes Bytes steht Bit 7 für die am weitesten links stehende Tile

Parameter	**Bedeutung**
Parameter 1	Bitmap-Position im Array (0...640)
Parameter 2	Tile-Nummer (0...48)
Parameter 3	—
Rückgabewert	0

1.4.7 Funktion 11 : Tile-Map Bereich mit konstantem Tile Index füllen

Diese Funktion setzt den angegebenen Tile-Bereich auf die angegebene Tile. Start- und Endposition ergeben sich aus **32 * Zeile + Spalte**. Dabei gibt es weder in X noch in Y Richtungen Einschränkungen, da sich der Tile-Map quasi unendlich wiederholt.

Parameter	**Bedeutung**
Parameter 1	Startposition
Parameter 2	Endposition
Parameter 3	Tile-Nummer (0...48)
Rückgabewert	0

1.4.8 Funktion 12 : Einen Tile-Index im Tilemap setzen

Setzt einen Tile-Index im Tilemap auf den angegebenen Wert.

Parameter	**Bedeutung**
Parameter 1	Y-Koordinate (0...31)
Parameter 2	X-Koordinate (0...31)
Parameter 3	Tile-Nummer (0...48)
Rückgabewert	0

1.4.9 Funktion 13 : Einen Tile-Index im Tilemap abfragen

Liefert den an der angegebenen Stelle befindlichen Tile-Index zurück.

Parameter	**Bedeutung**
Parameter 1	Y-Koordinate (0...31)
Parameter 2	Y-Koordinate (0...31)
Parameter 3	—
Rückgabewert	Tile-Index

1.4.10 Funktion 14 : Tile mit Farbe füllen

Füllt die Tile **Parameter 1** mit der Farbe **Parameter 2**.

Parameter	**Bedeutung**
Parameter 1	Tile-Nummer (0...48)
Parameter 2	Farbe (0...15)
Parameter 3	—
Rückgabewert	0

1.4.11 Funktion 15 : Vordefiniertes Muster in Tile zeichnen

Im Treiber sind bereits 32 verschiedene (zweifarbige) Muster definiert, die zum Erstellen von Tiles genutzt werden können. Parameter sind die Tile-Nummer, die Musternummer und ein Byte für die Festlegung der Zeichenfarbe. Dabei werden die Pixel in der Tile mit der eingestellten Farbe gesetzt, die eine ”1” im Muster haben, alle anderen Pixel bleiben unverändert.

Parameter	**Bedeutung**
Parameter 1	Tile-Nummer (0...48)
Parameter 2	Musternummer (0...31)
Parameter 3	Zeichenfarbe (0...15)
Rückgabewert	0

1.4.12 Funktion 16 : Array-definiertes Muster in Tile zeichnen

Es können auch Muster im Array definiert werden, um Tiles zu erstellen. Erster Parameter ist die Tile, die gefüllt werden soll. Der zweite Parameter gibt die Arrayzelle an, ab der das Muster abgelegt ist. Dieses besteht aus 8 aufeinanderfolgenden (Byte-) Zellen, bei denen Bit 7 das Pixel ganz links und Bit 0 das Pixel ganz rechts bestimmen. Der dritte Parameter ist wieder ein Byte für die Festlegung der Zeichenfarbe. Dabei werden die Pixel in der Tile mit der eingestellten Farbe gesetzt, die eine ”1” im Muster haben, alle anderen Pixel bleiben unverändert.

Parameter	**Bedeutung**
Parameter 1	Tile-Nummer (0...48)
Parameter 2	Array-Startzelle (0...760)
Parameter 3	Zeichenfarbe (0...15)
Rückgabewert	0

1.4.13 Funktion 17 : Muster aus Zeichensatz in Tile zeichnen

Mit dieser Funktion können Zeichen aus dem Zeichensatz als Muster für Tiles benutzt werden. Da die Tiles eine Größe von 8x8 Pixeln haben, der Zeichensatz aber eine von 6x10 Pixeln, werden oben und unten je eine Pixelzeile abgeschnitten und rechts sowie links eine leere Pixelzeile hinzugefügt. Erster Parameter ist die Tile, die gefüllt werden soll. Der zweite Parameter gibt das Zeichen aus dem Systemzeichensatz an, welches das Muster bestimmt. Der dritte Parameter ist wieder ein Byte für die Festlegung der Zeichenfarbe, wobei nur die Pixel in der Tile mit der eingestellten Farbe gesetzt werden, die eine ”1” im Muster haben. Alle anderen Pixel bleiben unverändert.

Parameter	**Bedeutung**
Parameter 1	Tile-Nummer (0...48)
Parameter 2	Zeichen (0...255)
Parameter 3	Zeichenfarbe (0...15)
Rückgabewert	0

1.4.14 Funktion 18 : Tile horizontal spiegeln

Die angegebene Tile wird horizontal gespiegelt. Was vorher links war ist danach rechts und umgekehrt.

Parameter	**Bedeutung**
Parameter 1	Tile-Nummer (0...48)
Parameter 2	—
Parameter 3	—
Rückgabewert	0

1.4.15 Funktion 19 : Tile vertikal spiegeln

Die angegebene Tile wird vertikal gespiegelt. Was vorher oben war ist danach unten und umgekehrt.

Parameter	**Bedeutung**
Parameter 1	Tile-Nummer (0...48)
Parameter 2	—
Parameter 3	—
Rückgabewert	0

1.4.16 Funktion 20 : Tile um 90 Grad im Uhrzeigersinn drehen

Die angegebene Tile wird um 90 Grad im Uhrzeigersinn gedreht.

Parameter	**Bedeutung**
Parameter 1	Tile-Nummer (0...48)
Parameter 2	—
Parameter 3	—
Rückgabewert	0

1.4.17 Funktion 21 : Tile um 90 Grad entgegen dem Uhrzeigersinn drehen

Die angegebene Tile wird um 90 Grad entgegen dem Uhrzeigersinn gedreht.

Parameter	**Bedeutung**
Parameter 1	Tile-Nummer (0...48)
Parameter 2	—
Parameter 3	—
Rückgabewert	0

1.4.18 Funktion 22 : Tile kopieren

Diese Funktion kopiert den Inhalt einer Tile in eine andere.

Parameter	**Bedeutung**
Parameter 1	Quell-Tile (0...48)
Parameter 2	Ziel-Tile (0...48)
Parameter 3	—
Rückgabewert	0

1.4.19 Funktion 23 : Tiles zusammenfügen

Diese Funktion zeichnet den Inhalt der zweiten Tile in die erste. Pixel, die in der zweiten Tile die Transparenz-Farbe haben, bleiben unverändert.

Parameter	**Bedeutung**
Parameter 1	Quell- und Ziel-Tile (0...48)
Parameter 2	Zeichen-Tile (0...48)
Parameter 3	Transparenzfarbe (0...15)
Rückgabewert	0

1.4.20 Funktion 24 : Tiles rotieren

Diese Funktion rotiert den Inhalt zweier Tiles. Die beiden Tiles liegen gedacht nebeneinander oder übereinander, was am einen "Ende" hinausgeschoben wird, erscheint wieder am anderen Ende. Der erste Parameter gibt die Tile an, die im gedachten Verbund oben bzw. links steht, der zweite Parameter gibt die Tile an, die im gedachten Verbund unten bzw. rechts steht.

Parameter	**Bedeutung**
Parameter 1	Tile 1 (0...48)
Parameter 2	Tile 2 (0...48)
Parameter 3	Verschiebe-Richtung (0...3)
Rückgabewert	0

Der dritte Parameter gibt die Richtung an:

Parameter 3	**Richtung**
0	Verschiebung/Rotation nach oben
1	Verschiebung/Rotation nach rechts
2	Verschiebung/Rotation nach unten
3	Verschiebung/Rotation nach links

1.4.21 Funktion 25 : Sprites initialisieren

Mit dieser Funktion lassen sich bis zu 8 Sprites initialisieren. Die Sprite-Daten liegen danach im Array ab der angegebenen Adresse, für jedes Sprite werden 16 Bytes benötigt.

Parameter	**Bedeutung**
Parameter 1	Array Position (0...511)
Parameter 2	Anzahl der Sprites (1...8)
Parameter 3	—
Rückgabewert	Maximale Sprite-Nummer

1.4.22 Funktion 26 : Sprite definieren

Mit dieser Funktion lassen sich die in Funktion 24 Initialisierten Sprites hinsichtlich ihres Aussehens definieren. Der erste Parameter ist die Sprite-Nummer, der zweite die Tile welche zur Darstellung benutzt werden soll. Der dritte Parameter gibt schließlich die Farbe an, welche bei Darstellen des Sprites transparent sein soll. Gleichzeitig wird das betreffende Sprite aktiviert.

Parameter	**Bedeutung**
Parameter 1	Sprite Nummer (0...7)
Parameter 2	Tile Nummer (0...48)
Parameter 3	Transparenz-Farbe (0...15)
Rückgabewert	0

1.4.23 Funktion 27 : Sprite positionieren

Mit dieser Funktion werden die in Funktion 24 Initialisierten Sprites positioniert. Der erste Parameter ist die Sprite-Nummer, der zweite und dritte Parameter geben die Position des Sprites im Map an. Der 4. und fünfte Parameter geben die Schrittweite an, um die das Sprite bei den Bewegungsfunktionen verschoben wird. Der Wert 1 bedeutet dabei eine Verschiebung nach rechts oder unten um 1 Pixel, der Wert 255 eine Verschiebung nach links bzw. oben um 1 Pixel.

Parameter	**Bedeutung**
Parameter 1	Sprite Nummer (0...7)
Parameter 2	Y Position (0...255)
Parameter 3	X Position (0...255)
Parameter 4	Y Bewegung (0...255)
Parameter 5	X Bewegung (0...255)
Rückgabewert	0

1.4.24 Funktion 28 : Sprite aktivieren

Aktiviert das Sprite innerhalb der Sequenz, so dass es nächstes mal mitgezeichnet wird.

Parameter	**Bedeutung**
Parameter 1	Sprite Nummer (0...7)
Parameter 2	—
Parameter 3	—
Rückgabewert	0

1.4.25 Funktion 29 : Sprite deaktivieren

Deaktiviert das Sprite innerhalb der Sequenz, so dass es nächstes mal nicht mitgezeichnet wird.

Parameter	**Bedeutung**
Parameter 1	Sprite Nummer (0...7)
Parameter 2	—
Parameter 3	—
Rückgabewert	0

1.4.26 Funktion 30 : Ein einzelnes Sprite zeichnen

Zeichnet ein einzelnes Sprite, Daten befinden sich an der angegebenen Arrayadresse. Dabei werden die original Tiles in die 4 Backup-Tiles kopiert, das Sprite dorthinein gezeichnet und die Tile-Indizies für die 4 Tiles auf die Backup-Tiles ”umgebogen”. Zusätzlich wird das Visible-Flag gesetzt.

Parameter	**Bedeutung**
Parameter 1	Array-Adresse (0...752)
Parameter 2	Y Position
Parameter 3	X Position
Rückgabewert	0

1.4.27 Funktion 31 : Ein gezeichnetes Sprite löschen

Löscht ein einzelnes Sprite, Daten befinden sich an der angegebenen Arrayadresse. Es werden lediglich die 4 Tile-Indizies auf die ursprünglichen Werte zurückgestellt und das Visible-Flag zurückgesetzt.

Parameter	**Bedeutung**
Parameter 1	Array-Adresse (0...752)
Parameter 2	—
Parameter 3	—
Rückgabewert	0

1.4.28 Funktion 32 : Alle Sprites sequentiell zeichnen

Zeichnet alle aktivierten Sprites in aufsteigender Reihenfolge.

Parameter	**Bedeutung**
Parameter 1	—
Parameter 2	—
Parameter 3	—
Rückgabewert	0

1.4.29 Funktion 33 : Alle Sprites sequentiell löschen

Löscht alle aktivierten Sprites in absteigender Reihenfolge.

Parameter	**Bedeutung**
Parameter 1	—
Parameter 2	—
Parameter 3	—
Rückgabewert	0

1.4.30 Funktion 34 : Alle Sprites-Koordinaten neu berechnen

Berechnet die Koordinaten aller aktivierten Sprites neu.

Parameter	**Bedeutung**
Parameter 1	—
Parameter 2	—
Parameter 3	—
Rückgabewert	0

1.4.31 Funktion 35 : Alle Sprites bewegen

Bewegt alle aktivierten Sprites. Dazu werden alle Sprites gelöscht, die Koordinaten neu berechnet und danach wieder gezeichnet. Dazu erfolgt das Löschen in absteigender Reihenfolge und das Zeichnen in aufsteigender Reihenfolge.

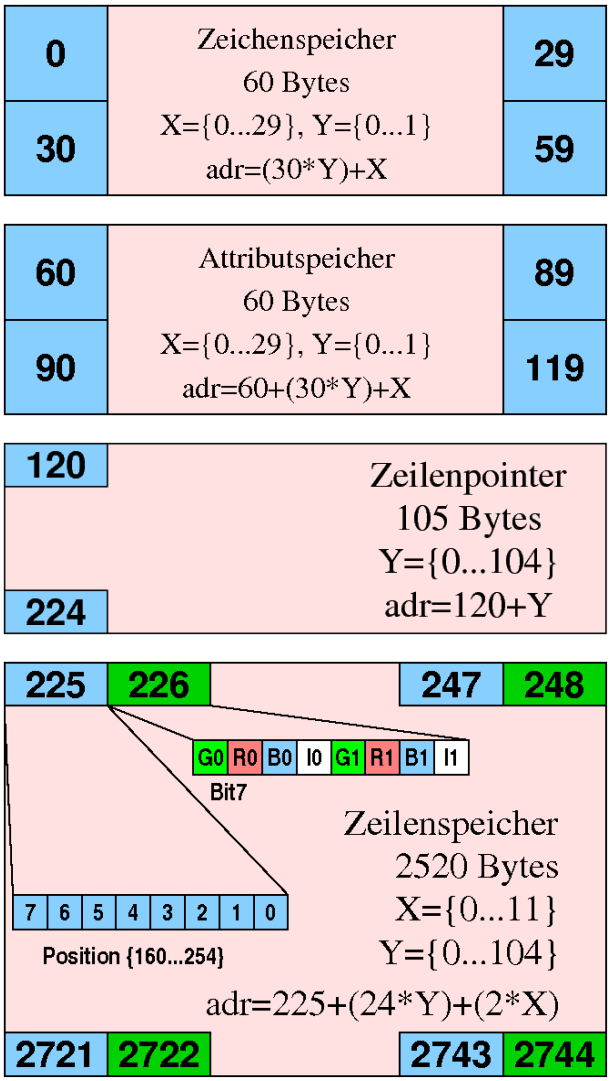
Parameter	**Bedeutung**
Parameter 1	—
Parameter 2	—
Parameter 3	—
Rückgabewert	0

1.5 Vector Mode Treiber (Bibliothekscode 0x91)

1.5.1 Allgemeines

Diesen Modus könnte man als eine Art ”Vektormodus” oder ”Differentialmodus” bezeichnen. Er entspricht dem Videomode 7 in früheren Versionen. 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.

1.5.2 Funktionen

Um den Treiber zu aktivieren, muß nur mittels **VMODE 7** in den USER-Videomode gewechselt werden. Die Ansteuerung der beiden Textzeilen am oberen und unteren Bildschirmrand erfolgt über die ”normalen” BASIC Ausgabebefehle. Die Zeile am oberen Rand entspricht dabei der Textzeile 0 und die am unteren Rand der Textzeile 1.
Weiterhin stehen noch weitere Funktionen über **CALL** zur Verfügung:

Funktionsnummer	**Funktion**
16	löscht den Vektor-Bereich
17	Zeilenzeiger konstant füllen
18	Zeilenzeiger aufsteigend füllen
19	Zeilenzeiger absteigend füllen
20	Zeilenzeiger mit Array-Werten füllen
21	Zeilenzeigerbereiche kopieren
22	Zeilen mit konstanten Array-Werten füllen
23	Zeilen mit aufeianderfolgenden Array-Werten füllen
24	—
25	—
26	—
27	—

1.5.3 Funktion 16 : Vektor-Bereich löschen

Diese Funktion löscht nur den gesamten Vektor-Bereich, wohingegen ein **CLS** nur den Textbereich löscht.

Parameter	**Bedeutung**
Parameter 1	—
Parameter 2	—
Parameter 3	—
Rückgabewert	0

1.5.4 Funktion 17 : Zeilenzeiger konstant füllen

Setzt die Zeilenzeiger von **Parameter 1** bis **Parameter 2** auf den Wert von **Parameter 3**. Die Werte werden dabei automatisch auf den gültigen Bereich begrenzt, ebenso findet ein Überlauf von 104 nach 0 statt.

Parameter	**Bedeutung**
Parameter 1	Startzeile
Parameter 2	Zeilenanzahl
Parameter 3	Wert
Rückgabewert	0

1.5.5 Funktion 18 : Zeilenzeiger aufsteigend füllen

Setzt die Zeilenzeiger von **Parameter 1** bis **Parameter 2** auf den Wert von **Parameter 3**, welcher von Zeile zu Zeile um 1 erhöht wird. Die Werte werden dabei automatisch auf den gültigen Bereich begrenzt, ebenso findet ein Überlauf von 104 nach 0 statt.

Parameter	**Bedeutung**
Parameter 1	Startzeile
Parameter 2	Zeilenanzahl
Parameter 3	Startwert
Rückgabewert	0

1.5.6 Funktion 19 : Zeilenzeiger absteigend füllen

Setzt die Zeilenzeiger von **Parameter 1** bis **Parameter 2** auf den Wert von **Parameter 3**, welcher von Zeile zu Zeile um 1 erniedrigt wird. Die Werte werden dabei automatisch auf den gültigen Bereich begrenzt, ebenso findet ein Überlauf von 104 nach 0 bei den Adressen sowie von 0 nach 104 beim Wert statt.

Parameter	**Bedeutung**
Parameter 1	Startzeile
Parameter 2	Zeilenanzahl
Parameter 3	Startwert
Rückgabewert	0

1.5.7 Funktion 20 : Zeilenzeiger mit Array-Werten füllen

Kopiert die Werte der Arrayzellen ab **Parameter 3** in die Zeilenzeiger ab **Parameter 1**, die Anzahl steht in Parameter 2. Die Werte werden dabei automatisch auf den gültigen Bereich begrenzt, ebenso findet ein Überlauf von 104 nach 0 bei den Adressen und einer von 767 nach 0 bei den Arrayzellen statt.

Parameter	**Bedeutung**
Parameter 1	Start-Zeilenzeiger
Parameter 2	Zeilenanzahl
Parameter 3	Array-Startposition
Rückgabewert	0

1.5.8 Funktion 21 : Zeilenzeigerbereiche intern kopieren

Kopiert Zeilenzeiger innerhalb des Zeilenzeigerbereiches. Parameter 1 gibt den ersten Zeilenzeiger des Quellbereiches und Parameter 2 den ersten Zeilenzeiger des Zielbereiches an. Der dritte Parameter gibt die Anzahl der zu kopierenden Zeilenzeiger an, wobei bei positiven Werten der Bereich beginnend beim ersten Zeilenzeiger kopiert wird, bei negativen Werten beginnen beim letzten Zeilenzeiger des zu kopierenden Bereiches.
Die Werte werden dabei automatisch auf den gültigen Bereich begrenzt, ebenso findet ein Überlauf von 104 nach 0 (sowie auch umgekehrt) bei den Adressen statt.

Parameter	**Bedeutung**
Parameter 1	Startzeile Quellbereich
Parameter 2	Startzeile Zielbareich
Parameter 3	Zeilenanzahl
Rückgabewert	0

1.5.9 Funktion 22 : Zeilen mit konstanten Array-Werten füllen

Kopiert die Werte der Arrayzellen ab **Parameter 3** in die Zeilen ab **Parameter 1**, die Anzahl steht in Parameter 2. Je Zeile werden die selben 24 Bytes kopiert. Es findet ein Überlauf von 104 nach 0 bei den Zeilen und einer von 767 nach 0 bei den Arrayzellen statt.

Parameter	**Bedeutung**
Parameter 1	Startzeile
Parameter 2	Zeilenanzahl
Parameter 3	Array-Startposition
Rückgabewert	0

1.5.10 Funktion 23 : Zeilen mit aufeinanderfolgenden Array-Werten füllen

Kopiert die Werte der Arrayzellen ab **Parameter 3** in die Zeilen ab **Parameter 1**, die Anzahl steht in Parameter 2. Je Zeile werden 24 Bytes kopiert. Es findet ein Überlauf von 104 nach 0 bei den Zeilen und einer von 767 nach 0 bei den Arrayzellen statt.

Parameter	**Bedeutung**
Parameter 1	Startzeile
Parameter 2	Zeilenanzahl
Parameter 3	Array-Startposition
Rückgabewert	0

2 Videotreiber für externe Displays

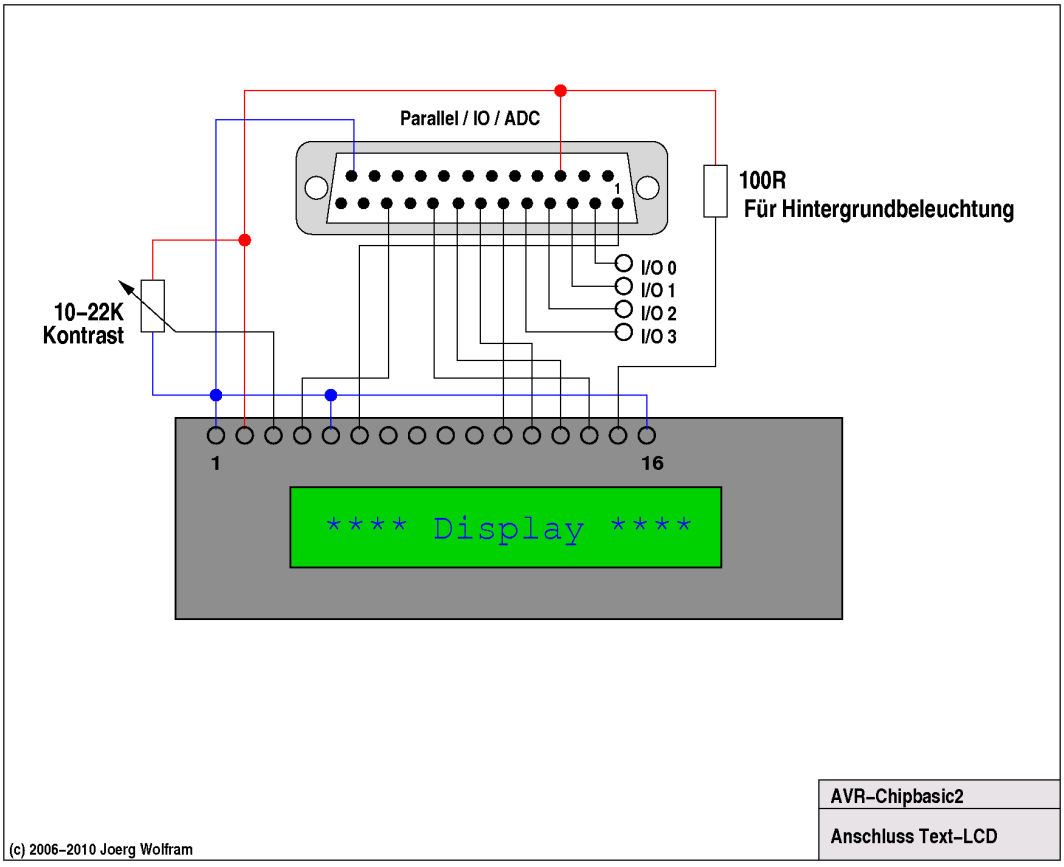
2.1 DUAL Text-LCD Treiber (Bibliothekscode 0xA0...0xA3)

2.1.1 Allgemeines

Mit diesen Treibern lassen sich verschiedene einzeilige und mehrzeilige Text-LCD ansteuern. Es werden die Datenleitungen D4...D7 benutzt, die Datenleitungen D0...D3 lassen sich über den I/O Adressbereich ab 0x800 bzw. die **IN** und **ADC**-Funktionen frei verwenden.
Die eigentliche Ausgabe erfolgt mit **PRINT** oder direktes Schreiben in das Video-RAM im Adressbereich ab 0x0000. Eine zyklische Update-Funktion sorgt dafür, dass alle Zeichen Zeichen zum LCD übertragen werden.
Ebenfalls ist es möglich, die benutzerdefinierten Zeichen zu nutzen. Dazu ist der I/O Bereich von 0xa00 bis 0xa40 reserviert. Dazu sind jeweils 8 aufeianderfolgende Bytes für ein Zeichen verantwortlich. Innerhalb der Bytes werden die Bits 4...0 benutzt, wobei Bit 4 dem am weitesten links liegenden Pixel entspricht. Benutzt werden diese Zeichen sinnvollerweise im Bereich von 8...15, der Bereich von 0...7 ist auch möglich, in einigen Fällen führt aber ein 0x00 im String zum Abbruch der Ausgabe.
Parallel zum LCD wird auch der Displayinhalt am TV-Ausgang ausgegeben, die Anzeige sollte mit der des LCD übereinstimmen, wobei für Zeichen größer 128 die Zeichensätze voneinander abweichen können. Vorder- und Hintergrundfarbe lassen sich ganz einfach über den **COLOR** Befehl einstellen, wobei die Einstellung sofort für die ganze (TV-) Anzeige übernommen wird.

2.1.2 Hardware

Angeschlossen wird das Display an den Parallelport, dazu müssen allerdings auch die +5V mit herausgeführt sein. Es werden die Leitungen Strobe, Busy und die Datenleitungen D4...D7 benutzt.



An die Datenleitungen D4...D7 können noch Taster etc. angeschlossen werden. Diese müssen die Signale über ca. 1KOhm nach Masse schalten. Der Status der vier Datenleitungen lässt sich mittels **IN($804)**...**IN($807)** abfragen.

2.1.3 unterstützte Displays

Da nur eine (relativ langsame) Initialisierung mit 4 Bit Datenbus sowie das direkte Schreiben in das DD- und CG-RAM genutzt werden, sollte der Treiber mit den meisten auf dem Markt erhältlichen Displays funktionieren. Wenn Sie in Erfahrungen mit einem in der nachstehenden Tabelle nicht aufgeführten Display gemacht haben, würde ich um eine kurze Meldung bitten, damit ich es in die nachstehende Tabelle einfügen kann. Zur Zeit werden folgende Display-Typen unterstützt:

Treiber	**Treibercode**	**Display**	**Display funktioniert**	**Display funktioniert nicht**
lcd116.bin	0xA0	1 x 16 Zeichen	Powertip PC1601A	.
lcd216.bin	0xA1	2 x 16 Zeichen	.	.
lcd220.bin	0xA2	2 x 20 Zeichen	Powertip PC2002-M	.

2.1.4 Funktionen

Der aktuelle Treiber hat keine zusätzlichen Funktionen eingebaut, die Ansteuerung erfolgt über die ”normalen” BASIC Ausgabebefehle. Um den Treiber vzu aktivieren, muß nur mittels **VMODE 7** in den USER-Videomode gewechselt werden. Beim Wechsel in einen anderen Videomode bleibt die Anzeige auf dem LCD erhalten. Allerdings sollte zwischen dem letzten Schreibvorgang und dem Wechsel des Videomode eine kleine Pause eingelegt werden (0,1s), damit die Daten auch sicher im LCD aktualisiert werden können. Um den Treiber zu testen, muss kein LCD angeschlossen sein.

2.2 DUAL Grafik-LCD Treiber 128x64 PIXEL (Bibliothekscode 0xA8)

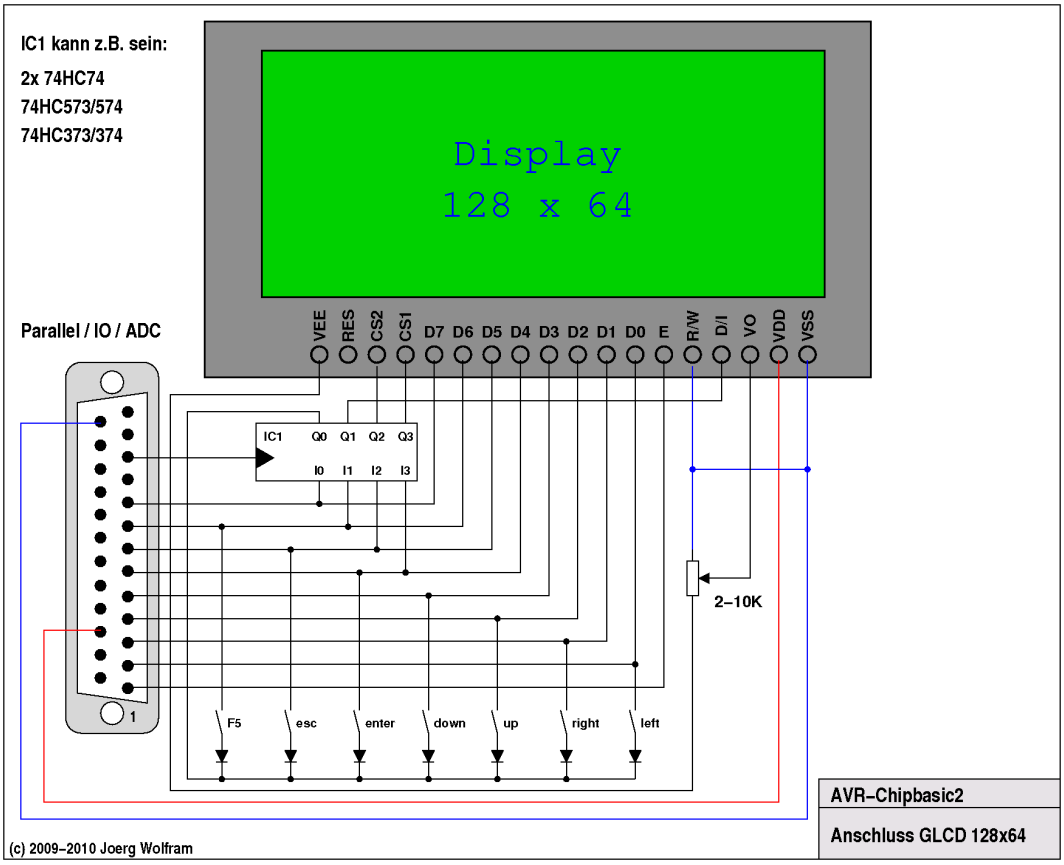
2.2.1 Allgemeines

Mit diesem Treiber lassen sich verschiedene KS0107/0108 basierte bzw. kompatible Grafikdisplays mit 128x64 Pixeln ansteuern. Die Ausgabe erfolgt dabei parallel auch am TV-Ausgang, funktionell ist der Videomode gleich dem eingebauten Mode 5. Für die Ansteuerung wird der komplette Parallelport benutzt, zusätztlich ist ein 4 Bit breites Latch (bei H transparent) oder Register (beide Flanken möglich) notwendig. Weiterhin können 7 Tasten angeschlossen werden. Von dem Treiber gibt es zwei Versionen, die sich in der Polarität der Chip-Select Signale unterscheiden. Die Vsrsion mit der Endung **_ncs** eignet sich für Displays mit LOW-aktiven Chip-Select Signalen, die mit der Endung **_pcs** dementsprechend für Displays mit HIGH-aktiven Chip-Select Signalen.

Eine zyklische Update-Funktion sorgt dafür, dass der Bildschirminhalt ca. 12 mal pro Sekunde zum LCD übertragen wird. Vorder- und Hintergrundfarbe sind dabei mit blau und cyan fest zugeordnet und lassen sich nicht ändern.

2.2.2 Hardware

Angeschlossen wird das Display an den Parallelport, dazu müssen allerdings auch die +5V mit herausgeführt sein. Die Datenleitungen werden direkt zum Display geführt, die Steuerleitungen I/D, CS1 und CS2 sowie das Treibersignal für die Zusatztasten werden über ein 4 Bit Latch/Register zwischengespeichert. Dazu dient das BUSY Signal, welches als Ausgang benutzt wird. Das E-Signal wird vom STROBE-Ausgang geliefert, die R/W Leitung liegt fest an Masse, es wird nur zum Display hin geschrieben.



2.2.3 Ausgabe-Funktionen

Der aktuelle Treiber hat keine zusätzlichen Funktionen eingebaut, die Ansteuerung erfolgt über die ”normalen” BASIC Ausgabebefehle. Um den Treiber vzu aktivieren, muß nur mittels **VMODE 7** in den USER-Videomode gewechselt werden. Beim Wechsel in einen anderen Videomode bleibt die Anzeige auf dem LCD erhalten. Allerdings sollte zwischen dem letzten Schreibvorgang und dem Wechsel des Videomode eine kleine Pause eingelegt werden (0,1s), damit die Daten auch sicher im LCD aktualisiert werden können. Um den Treiber zu testen, muss kein LCD angeschlossen sein.

2.2.4 Tasten-Funktionen

Es können bis zu 7 Tasten angeschlossen werden. Über die Funktion **KEY(8)** oder die API-Funktion **api_scancode** kann der Status direkt gelesen werden, wobei eine ”1” für eine gedrückte Taste steht. Zusäztzlich werden verschiedene Tastendrücke emuliert, im Schaltbild ist dies neben den Tastern angegeben.

Taste	Bit im Scancode	Keycode
left	0	0xE2
right	1	0xE3
up	2	0xE4
down	3	0xE5
enter	4	0xEA
esc	5	0xED
F5	6	0xF5

ACHTUNG! Sobald eine der Tasten gedrückt wurde, funktioniert eine angeschlossene PS2-Tastatur nicht mehr richtig.

2.3 DUAL Grafik-LCD Treiber 24x40 Zeichen (Bibliothekscode 0xA9)

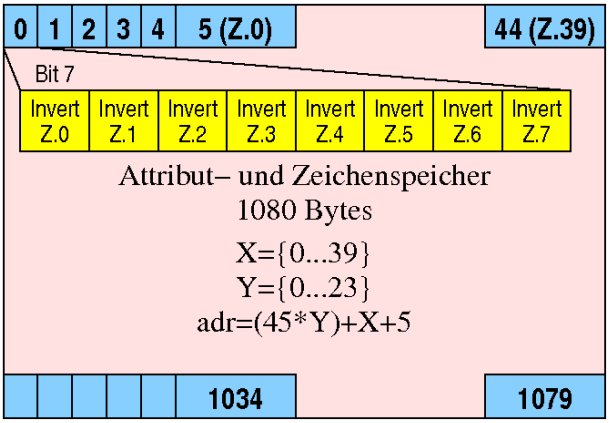
2.3.1 Allgemeines

Diesen Modus erlaubt ein Darstellung von Text und Pseudografik mit eine Auflösung von 24 Zeilen zu je 40 Zeichen. Allerdings ist die Darstellung nur monochrom (grün auf schwarz) möglich, die Ausgabepins für den roten und blauen Farbkanal werden hochohmig geschaltet. Wahlweise kann der Displayinhalt auf ein am Parallelport angeschlossenes Grafikdisplay ausgeben. Unterstützt werden monochrome Displays ohne eigenen Controller mit 320x240 Pixeln. Die Anzeige kann nur auf einem Weg erfolgen, TV-Ausgang oder Display, eine simultane Anzeige ist nicht möglich. Die jeweils nicht aktive Anzeige ist dann abgeschaltet, beim LCD gibt es einen Steuer-Pin mit dem die LCD-Spannung abgeschalten werden kann und auch sollte.

Eine Zeile im Bildschirmspeicher besteht aus 45 Bytes. Zuerst kommen 5 Bytes bei denen jeweils 1 Bit für die Invertierung eines der nachfolgenden Zeichen verantwortlich ist. Danach folgen 40 Zeichen-Bytes, Bit 7 der Zeichen wird ignoriert. Die Zeichen sind mit 8 Pixeln horizontal und 10 Pixeln Vertikal definiert, der Zeichensatz mit insgesamt 128 Zeichen befindet sich im RAM und kann auch geändert werden. Da der Zeichensatz vom Speicherbedarf her nicht mehr in den Treiber gepasst hat, muß dieser auf den Programmplatz 7 geladen werden. Ein passender Zeichensatz ist **charset10.bin**.

Der Bildspeicher ist folgendermaßen aufgeteilt:

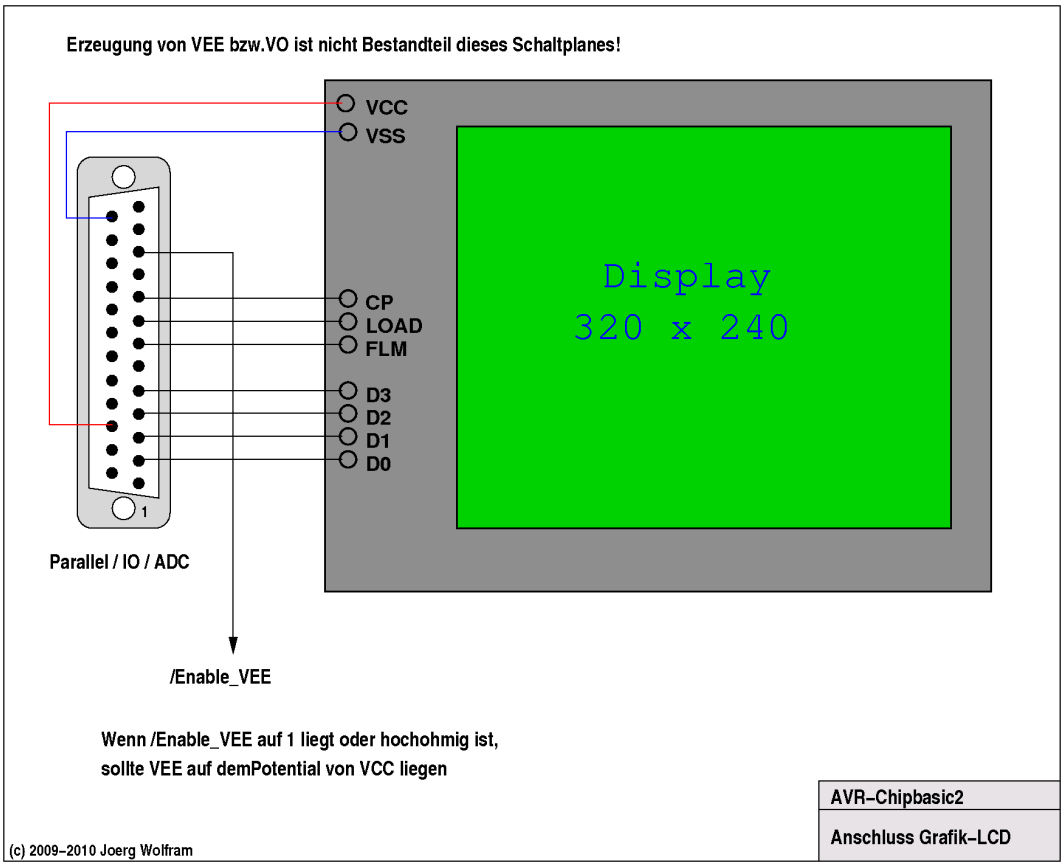
Anfangsadresse	Endadresse	Bytes	Funktion
0	1079	1080	24 Zeilen Zeichen
1080	1319	240	z.T.intern benutzt
1320	2599	1280	Zeichensatz
2600	2759	160	ungenutzt



Der Zeichensatz im RAM ist Zeilenweise aufgebaut, zuerst 128 Bytes für die erste Zeile, dann 128 Bytes für die zweite Zeile und so weiter bis zu 128 Bytes für die zehnte und damit letzte Zeile. Der Zeichensatz auf Programmplatz 7 der als Quelle dient, hat eine ähnliche Struktur, nur dass hier eine Zeile 256 Bytes hat. Das heisst auch dass der Treiber nur die unteren 128 Zeichen des Zeichensatzes nutzt.

2.3.2 Hardware

Zum Anschluss des Displays wird der Parallelport benutzt. Vorhanden sind die Datenignale **D0...D3** (an den gleichnamigen Port-Pins), **CP** an Pin D7, **LOAD** an Pin D6 und **FLM/FRAME** an Pin D5. An BUSY-Pin steht zusätzlich ein Schaltsignal (LOW-aktiv) zur Verfügung, mit dem die Display-Spannungen VEE und VO geschaltet werden sollten. Andernfalls kann es zu Schäden am Display kommen, sowohlbeimEinschalten bis zumWechsel in den Videomode 7 als auch wenn in einen anderen Videomode gewechselt wird.



2.3.3 Funktionen

Der aktuelle Treiber hat keine zusätzlichen Funktionen eingebaut, die Ansteuerung erfolgt über die ”normalen” BASIC Ausgabebefehle. Um den Treiber zu aktivieren, muß nur mittels **VMODE 7** in den USER-Videomode gewechselt werden. Dabei wird dann der Zeichensatz von Programmplatz 7 in den RAM kopiert. Mit **COLOR 0** werden die Zeichen invertiert ausgegeben, **COLOR 1** schaltet in den nichtinvertierten Modus zurück.

Ebenfalls ist es möglich, 16 (0x00...0x0f) benutzerdefinierte Zeichen zu nutzen. Dazu ist der I/O Bereich von 0xa00 bis 0xaff reserviert. Dazu sind jeweils 16 aufeianderfolgende Bytes für ein Zeichen verantwortlich, genutzt werden nur die ersten 10 Bytes. Innerhalb der Bytes werden die Bits 7...0 benutzt, wobei Bit 7 dem am weitesten links liegenden Pixel entspricht. Alternativ kann auch mittels **VPOKE** direkt in den Bildschirmspeicher geschrieben werden, hier gibt es aber keinen linearen Zusammenhang zwischen Zeichenbytes und Adresse wie beim **OUT** Befehl, der bereits die Umrechnung der Adressen vornimmt. Mit dem Befehl **OUT $900,1** kann die Anzeige auf das angeschlossene LCD umgestellt werden, mit **OUT $900,0** wird wieder auf TV-Ausgabe zurückgeschaltet. Standardmäßig wird der Videomode mit TV-Anzeige gestartet.

3 Videotreiber mit PWM-Funktionen

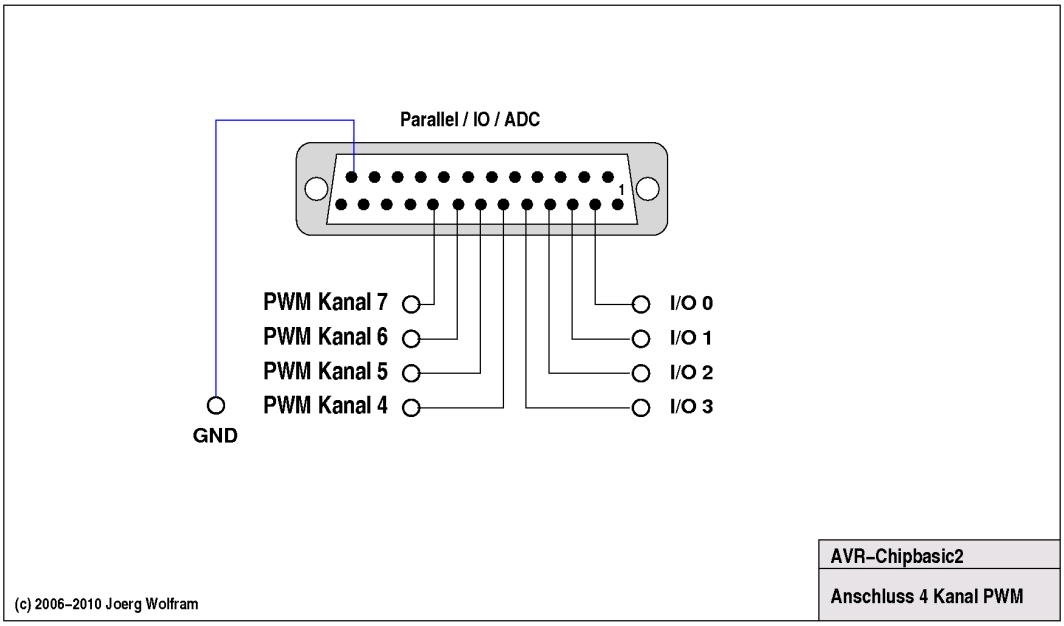
3.1 PWM 4 x 8 Bit Treiber (Bibliothekscode 0xB0)

3.1.1 Allgemeines

Dieser Treiber stellt 4 pulsweitenmodulierte Ausgänge zur Verfügung. Diese haben eine Auflösung von 8 Bit und eine Frequenz von etwa 61 Hz. Daneben gibt es eine zum Videomode 0 weitestgehend kompatible Anzeige, allerdings sind von jeder Zeile nur die erten 24 Zeichen sichtbar.

3.1.2 Hardware

Die 4 PWM-Kanäle liegen an den Datenbits D4...D7 an, die Datenbits D0...D3 lassen sich über die Adressbereiche 0x800...0x803 (Daten) und 0x810...0x813 (Datenrichtung) sowie über die IN() und ADC() Funktionen frei benutzen.



3.1.3 Funktionen

Der aktuelle Treiber hat keine zusätzlichen Funktionen eingebaut, die Ansteuerung erfolgt über die ”normalen” BASIC Ausgabebefehle. Um den Treiber vzu aktivieren, muß nur mittels **VMODE 7** in den USER-Videomode gewechselt werden. Beim Aktivieren und beim Verlassen des Videomodes werden alle PWM-Kanäle auf 0 gesetzt.
Die einzelnen Kanäle werden mittels OUT Befehlen auf die Adressen 0x900...0x903 gesteuert, ei Zurücklesen ist über die Funktion IN() möglich. Die folgende Tabelle zeigt eine Übersicht über die nutzbaren I/O Adressen:

I/O Adresse	Zugriff	Funktion
0x800	R/W	Datensignal D0
0x801	R/W	Datensignal D1
0x802	R/W	Datensignal D2
0x803	R/W	Datensignal D3
0x808	R/W	Datensignal STROBE
0x809	R/W	Datensignal BUSY
0x810	W	Datenrichtung D0 (0=IN, 1=OUT)
0x811	W	Datenrichtung D1 (0=IN, 1=OUT)
0x812	W	Datenrichtung D2 (0=IN, 1=OUT)
0x813	W	Datenrichtung D3 (0=IN, 1=OUT)
0x818	R/W	Datenrichtung STROBE
0x819	R/W	Datenrichtung BUSY
0x900	R/W	Pulsweite an D4 (0...255)
0x901	R/W	Pulsweite an D5 (0...255)
0x902	R/W	Pulsweite an D6 (0...255)
0x903	R/W	Pulsweite an D7 (0...255)

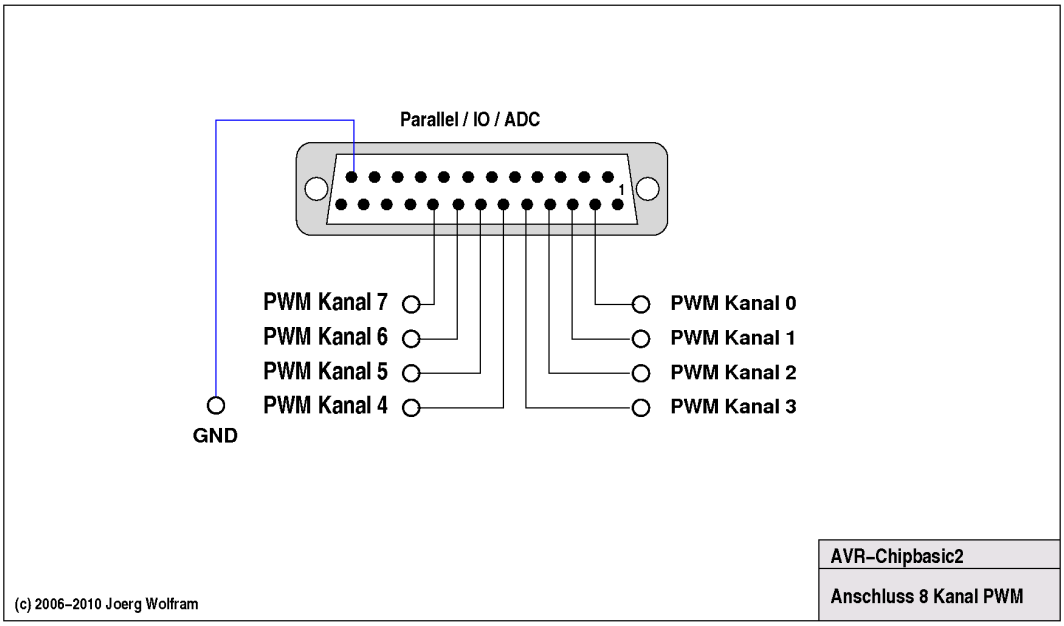
3.2 PWM 8 x 8 Bit Treiber (Bibliothekscode 0xB1)

3.2.1 Allgemeines

Dieser Treiber stellt 8 pulsweitenmodulierte Ausgänge zur Verfügung. Diese haben eine Auflösung von 8 Bit und eine Frequenz von etwa 61 Hz. Daneben gibt es eine zum Videomode 0 weitestgehend kompatible Anzeige, allerdings sind von jeder Zeile nur die erten 24 Zeichen sichtbar.

3.2.2 Hardware

Die 8 PWM-Kanäle liegen an den Datenbits D0...D7 an.



3.2.3 Funktionen

Der aktuelle Treiber hat keine zusätzlichen Funktionen eingebaut, die Ansteuerung erfolgt über die "normalen" BASIC Ausgabebefehle. Um den Treiber vzu aktivieren, muß nur mittels **VMODE 7** in den USER-Videomode gewechselt werden. Beim Aktivieren und beim Verlassen des Videomodes werden alle PWM-Kanäle auf 0 gesetzt.
Die einzelnen Kanäle werden mittels OUT Befehlen auf die Adressen 0x900...0x907 gesteuert, ein Zurücklesen der Daten ist über IN() möglich. Die folgende Tabelle zeigt eine Übersicht über die nutzbaren I/O Adressen:

I/O Adresse	Zugriff	Funktion
0x808	R/W	Datensignal STROBE
0x809	R/W	Datensignal BUSY
0x818	R/W	Datenrichtung STROBE
0x819	R/W	Datenrichtung BUSY
0x900	R/W	Pulsweite an D0 (0...255)
0x901	R/W	Pulsweite an D1 (0...255)
0x902	R/W	Pulsweite an D2 (0...255)
0x903	R/W	Pulsweite an D3 (0...255)
0x904	R/W	Pulsweite an D4 (0...255)
0x905	R/W	Pulsweite an D5 (0...255)
0x906	R/W	Pulsweite an D6 (0...255)
0x907	R/W	Pulsweite an D7 (0...255)

3.3 PWM 2 x 8 Bit + 2 x Puls Treiber (Bibliothekscode 0xB2)

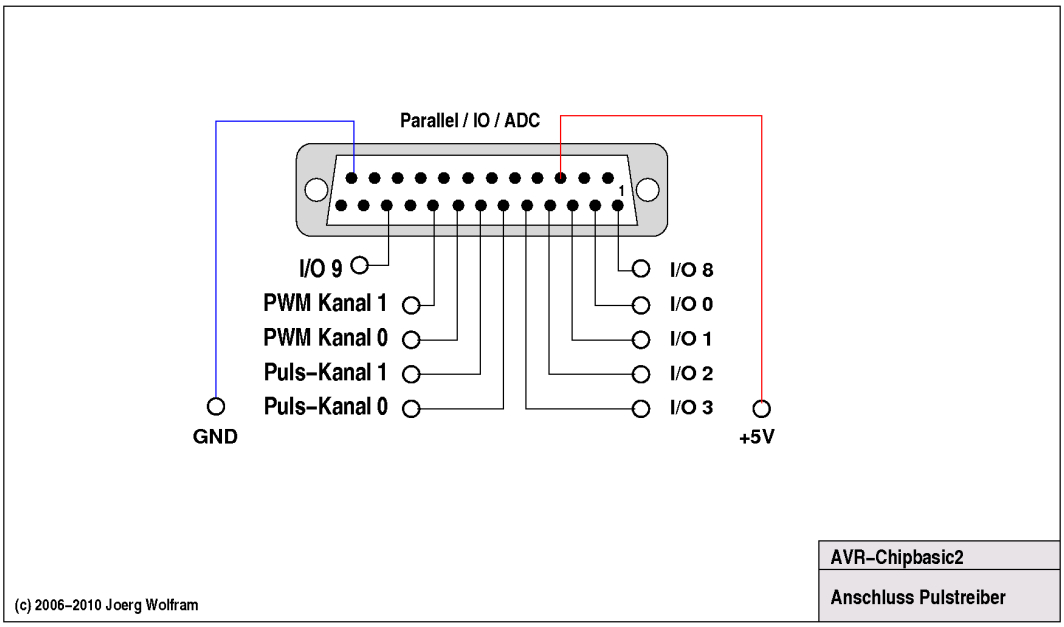
3.3.1 Allgemeines

Dieser Treiber stellt 2 pulsweitenmodulierte Ausgänge zur Verfügung. Diese haben eine Auflösung von 8 Bit und eine Frequenz von etwa 61 Hz. Zusätzlich gibt es noch zwei Pulsausgänge mit denen sich periodische Pulse im Bereich von 0...2,24 Milliskeunden erzeugen lassen. Die Wiederholfrequenz entspricht dabei der Vertikal-Synchronfrequenz, also ca. 50Hz für PAL und 60Hz für NTSC.

Daneben gibt es eine zum Videomode 0 weitestgehend kompatible Anzeige, allerdings werden nur die ersten 20 Zeilen angezeigt von denen auch nur 28 Zeichen sichtbar sind.

3.3.2 Hardware

Die 2 PWM-Kanäle liegen an den Leitungen D6 und D7 an, die beiden Pulskanäle an den Datenleitungen D4 und D5. Die Datenleitungen D0...D3 sowie **BUSY** und **STROBE** lassen sich über die Adressbereiche 0x800...0x809 (Daten) und 0x810...0x819 (Datenrichtung) sowie über die IN() und ADC() Funktionen (letztere nur D0...D3) frei nutzen.



3.3.3 Funktionen

Der aktuelle Treiber hat keine zusätzlichen Funktionen eingebaut, die Ansteuerung erfolgt über die "normalen" BASIC Ausgabebefehle. Um den Treiber zu aktivieren, muß nur mittels **VMODE 7** in den USER-Videomode gewechselt werden. Beim Aktivieren und beim Verlassen des Videomodes werden alle PWM-Kanäle auf 0 gesetzt.
Die einzelnen Kanäle werden mittels OUT Befehlen auf die Adressen 0x900...0x903 gesteuert, die eingestellten Werte können auch über IN() zurückgelesen werden. Die folgende Tabelle zeigt eine Übersicht über die nutzbaren I/O Adressen:

I/O Adresse	Zugriff	Funktion
0x800	R/W	Datensignal D0
0x801	R/W	Datensignal D1
0x802	R/W	Datensignal D2
0x803	R/W	Datensignal D3
0x808	R/W	Datensignal STROBE
0x809	R/W	Datensignal BUSY
0x810	W	Datenrichtung D0 (0=IN, 1=OUT)
0x811	W	Datenrichtung D1 (0=IN, 1=OUT)
0x812	W	Datenrichtung D2 (0=IN, 1=OUT)
0x813	W	Datenrichtung D3 (0=IN, 1=OUT)
0x818	R/W	Datenrichtung STROBE
0x819	R/W	Datenrichtung BUSY
0x900	R/W	Pulsbreite an D4 (0...559)
0x901	R/W	Pulsbreite an D5 (0...559)
0x902	R/W	Pulsweite an D6 (0...255)
0x903	R/W	Pulsweite an D7 (0...255)

4 Treiber für Zusatzspeicher

4.1 Memory-Treiber für 2K internes RAM (Bibliothekscode 0xF0)

Dieser Treiber nutzt 2K des internen Speichers, wozu ein Teil des Bilschirmspeichers sowie ein Teil des Arrays benutzt werden. Dieser Treiber dient hauptsächlich dazu, Programme die XMEM benötigen zu testen. Es sind nur die Videomodi 0 und 5 nutzbar, bei den Videotreibern sollten zumindest die LCD Treiber funktionieren.

4.2 Memory-Treiber für 64K externes RAM (Bibliothekscode 0xF1)

Dieser Treiber erlaubt die Nutzung des XRAM64 Speichermodules wie es bei den Erweiterungen beschrieben wird. Es können die Pages 0...255 benutzt werden, bei nur einem 32K RAM die Pages 0...127.

4.3 Memory-Treiber für 12K internes RAM (Bibliothekscode 0xF2)

Dieser Treiber nutzt die freien 12K des internen Speichers, wenn ein ATMega1284P eingesetzt wird. Es sind die Pages 0 bis 48 verfügbar.