AVR-ChipBasic2: BASIC-Referenz (3)

Ein-/Ausgabe

1 Ein- und Ausgabe über den Parallelport

1.1 DIR n

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

```
10 DIR $F0
```

Pin D0-D3 werden als Eingang, D4-D7 als Ausgang konfiguriert.

1.2 OUT n,b

Hier gibt es zwei Funktionen, n=0..7 setzt einzelne Bits, n=$100 bis $1ff erlaubt es, mehrere Bits gleichzeitig zu setzen bzw. zu löschen. dabei dienen die unteren 8 Bits von n als Maske.
Beispiele:

```
10 OUT 7,0
```

Setzt D0 auf 0-Pegel.

```
10 OUT $1F0,$F0
```

Setzt D3-D7 auf 1-Pegel.

2 Kommunikation

2.1 SPUT n

das Byten wird an die serielle Schnittstelle ausgegeben.

```
01 SPUT 10
```

gibt einen Zeilenvorschub an die serielle Schnittstelle aus.

2.2 SGET V

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

```
01 SGET C
```

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

2.3 PUMP n

Schaltet die Ladungspumpe für die serielle Schnittstelle aus (n=0) oder ein (n=1).

```
01 PU 1
```

schaltet die Ladungspumpe ein. Ist der AutoRun-Jumper gesetzt, ist die Ladungspumpe per default nach dem Start ausgeschaltet.

2.4 ICOMM adr,start,num [IC]

Generische I2C Routine. Der Erste Parameter **adr** gibt die Slave-Adresse an. Gleichzeitig wird mit Bit 0 festgelgt, ob geschrieben (0) oder gelesen (1) werden soll. Die beiden anderen Parameter geben die Startadresse im Array und die Anzahl der zuübertragenden Bytes an.

```
01 BDATA HI(B),LO(B),LO(A),HI(A):ICOMM \$a0,0,4
```

Schreibt den Wert der Variable A an die Adresse B des Daten-EEPROMS. Dabei ist auf die Reihenfolge der Bytes zu achten, da Beim I2C EEPROM zuerst das High-Byte der Adresse und danach das Low-Byte der Adresseübertragen werden muss. Um eventuelle Wartezeiten nach der Aktion muss man sich selbst kümmern, Ein Schreiben in den EEPROM mit darauffolgendem Lesen gibt mit hoher Sicherheit einen I2C-Fehler,**SYNC 2** schafft eine Pause von mindestens 20ms. Wird die Anzahl der zu schreibenden Bytes auf 0 gesetzt, werden solange Bytes aus dem Array ausgegeben, bis 0x00 erkannt wird. Im Lesemodus bedeutet eine 0 als Anzahl, dass als erstes die Anzahl der zu lesenden Bytes eingelesen wird. Ein I2C-Slave, der nichts senden möchte, muss nach 0x00 noch ein Dummy-Byte an den Master (BASIC-Controller) senden, ist mindestens 1 Byte zu senden braucht kein Dummy-Byte zu folgen.

3 Speicher

3.1 EPOKE adr,dat [EP]

Speichert ein Datenbyte in den internen EEPROM. Als Adressen sind 0...1999 möglich.

```
02 EPOKE V,$12
```

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

3.2 EPEEK(adr)

Die Funktion liest ein Byte aus dem internen EEPROM. Als Adressen sind 0...1999 möglich.

```
02 W=EPEEK(V)
```

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

An die I2C-Schnittstelle kann ein zusätzlichs EEPROM (derzeit nur 24C64...24C512 und Adresse 1) angeschlossen werden. Bei anderen Typen lässt sich die gleiche Funktionalität über ICOMM realisieren.

3.3 XPOKE adr,dat [XP]

Speichert ein Byte im externen EEPROM (Adresse=1).

```
01 XPOKE 100,C
```

speichert den Wert der Variablen C an die Adresse 100.

3.4 XPEEK(adr)

Die Funktion liest ein Byte aus dem externen EEPROM. Als Adressen sind 0...$FFFF möglich.

```
02 W=XPEEK(100)
```

liest den Wert von EEPROM-Adresse 100 /externes EEPROM) und speichert diesen in der Variable W.

4 Dateisystem-Funktionen

Wenn Sie bereits mit Personal Computern vertraut sind, werden Sie jetzt ein klein wenig umdenken müssen, da das Dateisystem von AVR-ChipBasic2 ein bisschen anders aufgebaut ist. Voraussetzung um mit Dateien arbeiten zu können ist ein Dataflash-Modul, welches in den Programmier-Port gesteckt wird. Je nach Typ können bis zu 0,5 oder 1 Megabyte Daten gespeichert werden. Natürlich ist es immer etwas weniger, denn die Daten wollen auch organisiert werden, was wiederum Speicherplatz auf dem Medium beansprucht. Um mit dem Dataflash-Modul arbeiten zu können, muss dieses „formatiert" werden. Das geschieht im DFLASH Menü, welches über das Hauptmenü erreichbar ist. Im Gegensatz zu flüchtigen Speichern (RAM) oder Festplatten sind Schreibzugriffe auf Flash oder EEPROM Speicher nur in begrenzter Anzahl möglich. Meistens liegt diese Zahl zwischen 10000 und 1 Million, je nach Speichertyp und Hersteller. Um die Speicherblöcke möglichst gleichmäßig „abzunutzen" enthalten sie einen Zähler, der bei jedem Schreibvorgang um 1 erhöht wird. Wenn eine Datei erzeugt wird, werden nun Blöcke mit möglichst niedrigem Zählerstand verwendet und auch beim häufigen Schreiben auf den gleichen Block kann es passieren, dass ein neuer Block mit niedrigerem Zählerstand gesucht wird. Darum muss man sich aber nicht kümmern, das Ganze passiert völlig transparent, kann aber zu kurzzeitigen Verzögerungen im Programmablauf führen. Die Dataflash-Module sollten nicht allzu häufig formatiert werden, da dabei die Informationen über den „Abnutzungsgrad" der Blöcke verloren gehen.

Der freie Speicherplatz ist in Blöcke (Pages) von 256 Bytes aufgeteilt, die wiederum maximal 128/256 Dateien (Files) zugeordnet werden können. Jede Datei kann minimal 1 und maximal 128 Datenblöcke enthalten was Dateigrößen von 256 Bytes bis 32 Kilobytes erlaubt. Dateinamen, wie man sie vom PC her kennt, gibt es hier aber nicht. Damit Dateien in der Fileselect-Box unterscheidbar sind, haben sie einen fest eincodierten Typ und die ersten 12 Bytes werden als Dateiname angezeigt. Wenn USR Dateien angelegt werden, sollte auch in die ersten 12 Bytes eine mehr oder weniger eindeutige Identifizierung geschrieben werden. Der Einfachkeit halber kann man auch den ersten Block der Datei nur für den Namen reservíeren, auch wenn dadurch Speicherplatz verschenkt wird. Ansonsten wird einfach mit der Dateinummer gearbeitet.

4.1 FTYPE(n)

Die Funktion FTYPE liefert entweder den Flashtyp (n=-1) oder der Dateityp der Datei n. Für n=-1 ist das Resultat

Wert	**Bedeutung**
0	kein Dataflash angeschlossen
2048	4MBit Dataflash angeschlossen
4096	8MBit Dataflash angeschlossen

für 0<=n<=255 ist das Resultat (derzeit)

Wert	**Bedeutung**
252	Datei ist nicht belegt
16	BASIC-Programm
18	Backup-Datei
20	USR-Datei
-1	Dateinummer existiert nicht

4.2 FSIZE(n)

Die Funktion FSIZE(n) liefert entweder die Zahl der verfügbaren Files/Pages oder die Größe einer datei in 256Bytes-Pages.

n	**Rückgabewert**
-1	Anzahl der freien Files
-2	Anzahl der freien Pages
0...255	Anzahl der Pages in der Datei oder 0 (nicht belegt)

4.3 FCREATE n,p [FCR]

Erzeugt das USR-File n mit p Pages. Die Anzahl der Pages kann dabei zwischen 1 und 128 liegen was einer Dateigröße von 256 Bytes...32 Kbytes entspricht. Da FCREATE im Fehlerfall abbricht, sollte vorher mit FTYPE und FSIZE sichergestellt werden, dass die Datei auch erzeugt werden kann. Es werden immer Files vom Typ USR erzeugt.

```
40 'create file F with P pages
41 'R=F or -1 if failed
42 R=-1:IF P<1 THEN RETURN
43 IF P>128 THEN RETURN
44 IF FTYPE(-1)=0 THEN RETURN
45 IF FTYPE(F)<>0 THEN RETURN
46 IF FSIZE(-2)<P THEN RETURN
47 FCREATE F,P: R=F: RETURN
```

Die Subroutine erzeugt das File F mit P Pages, wenn dies möglich ist. In der Variable R steht dann das Resultat: entweder die Filenummer F oder -1, falls das Anlegen des Files nicht möglich ist.

4.4 FWRITE n,p,a [FWR]

Schreibt 256 Bytes aus dem Array in die Page P von File F. Der dritte Parameter bestimmt die Arrayposition:

a	Arraybereich
0	0...255
1	256...511
2	512...767

Da FWRITE im Fehlerfall abbricht, sollte vorher mit FTYPE und FSIZE sichergestellt werden, dass die Page auch geschrieben werden kann. Es kann nur in Files vom Typ USR geschrieben werden.

```
50 'write array 0...255 to
51 'file F in page P
52 R=-1:IF P<1 THEN RETURN
53 IF FTYPE(F)<>20 THEN RETURN
54 IF P>FSIZE(F) THEN RETURN
55 FWRITE F,N,0: R=F: RETURN
```

Die Subroutine schreibt den Inhalt der Arrayzellen 0...255 in die Page P von File F. In der Variable R steht das Resultat: entweder die Filenummer F oder -1, falls das Schreiben der Page nicht möglich ist.

4.5 FREAD n,p,a [FRD]

Liest 256 Bytes aus Page P von File F und schreibt sie in das Array. Der dritte Parameter bestimmt wieder die Arrayposition:

a	Arraybereich
0	0...255
1	256...511
2	512...767

Da FREAD im Fehlerfall abbricht, sollte vorher mit FTYPE und FSIZE sichergestellt werden, dass die Page auch geschrieben werden kann. Es kann nur von Files vom Typ USR gelesen werden.

```
50 'read page P of file F to
51 'array 256...511
52 R=-1:IF P<1 THEN RETURN
53 IF FTYPE(F)<>20 THEN RETURN
54 IF P>FSIZE(F) THEN RETURN
55 FREAD F,N,1: R=F: RETURN
```

Die Subroutine liest den Inhalt der Page P von File F in die Arrayzellen 256...511. In der Variable R steht das Resultat: entweder die Filenummer F oder -1, falls das Lesen der Page nicht möglich ist.

4.6 FDELETE n [FDEL]

Löscht das File N, wenn es vorhanden und vom Typ USR ist. Da FDELETE im Fehlerfall abbricht, sollte vorher mit FTYPE und FSIZE sichergestellt werden, dass das File vorhanden und von TYP USR ist.

```
56 'delete file F
57 R=-1
58 IF FTYPE(F)<>20 THEN RETURN
59 FDELETE F: R=F: RETURN
```

Die Subroutine löscht das File F, falls es vorhanden und vom Typ USR ist.

4.7 FSELECT v,a [FSEL]

Stellt die Fileselect-Box dar, falls ein formatierter Dataflash-Baustein angeschlossen ist. Die Nummer des gewählten Files liegt im Anschluss in der Variable v. Ist der Dataflash nicht ansprechbar oder wird die Auswahl mit der ESC-Taste abgebrochen, wird die Variable v auf -1 gesetzt. Der wert a ist die Array-Position, ab der die nullterminierte Boxüberschrift steht. Der Rahmen der Fileselct-Box wird mit der aktuell eingestellten Vordergrundfarbe gezeichnet, der Hintergrund ist immer schwarz. Der ursprüngliche Bildinhalt wird nach Verlassen der Fileselect-Box wiederhergestellt.

```
60 'draw fileselct box
61 DATA 0,"Select file:",0
62 FSELECT F,0: RETURN
```

Die Subroutine zeigt eine Filelesect-Box mit dem Titel „Select file:”.

5 Meldungen und Fragen in der Box

5.1 ALERT n

Zeigt eine Alert-Box an. Die Daten dazu befinden sich im Array ab Position n. Zuerst stehen dort Vordergrund- und Hintergrundfarbe, gefolgt von Nullterminiertem Text. Fehlt dder wert 0x00, wird die Ausgabe nach 20 Zeichen abgebrochen.

```
01 DATA 0,1,6,"Hallo Welt",0
02 ALERT 0
```

Sieht dann so aus:

5.2 ASK V,n

Zeigt eine Frage-Box an. Die Daten dazu befinden sich im Array ab Position n. Zuerst stehen dort Vordergrund- und Hintergrundfarbe, gefolgt von Nullterminiertem Text. Fehlt dder wert 0x00, wird die Ausgabe nach 20 Zeichen abgebrochen.

```
01 DATA 0,1,6,"Weitermachen?",0
02 ASK P,0
03 ? P
04 WAIT 20
```

Sieht dann so aus:

Wenn Y(y) gedrückt wurde, bekommt P den Wert 1 zugewiesen, wurde N(n) gedrückt, erhält P den Wert 0