SEPIA
Advanced BASIC

Praktische Beispiele

Inhalt

* Funktion zur Zeit noch in Erprobung

Einführung

In der Grundausstattung eines SEPIA-Systems werden neben BIOS, Debugger und Monitor, ein kleines OS-Betriebssystem, sowie ein BASIC-Interpreter im EPROM mitgeliefert. Damit kann der Anwender SEPIA sofort starten und für seine Applikationen programmieren.

Folgende Punkte waren uns bei der Entwicklung ganz besonders wichtig:

- Großer Anwendernutzen durch hohe Sicherheit und Vielseitigkeit
- Unabhängige Plattform mit eigenem BIOS und Betriebssystem, ohne weitere Kosten
- Schneller Projekterfolg durch leichte Programmierung mittels einfacher Hochsprache
- Möglichkeit zum Debuggen und Einbringung von Binärcode in Maschinensprache
- Skalierbarkeit der Hardware durch vielfältige Schnittstellen (kein plug-and-pray)
- Nutzen von vorhandenen Ressourcen (bsp. VGA-Monitor, PS/2 Tastatur, CF-Card...)
- Möglichkeit zur Vernetzung über Ethernet (virtueller COM-Port für Windows-Systeme)
- Programmierung als Host-Rechner, oder fernbedienbar als Remote-I/O-System

SEPIA kann somit fernab vom üblichen Marktgeschehen punkten, da Änderungen in der Programmierung, quasi vor Ort, jederzeit und ohne zusätzlichen PC durchführbar sind.

Die Wahl der Programmiersprache viel gleich zu Anfang auf BASIC. Mit dieser einfachen Hochsprache können alle notwendigen Bedingungen für die Automation erfüllt werden. Somit schließt SEPIA die Lücke zwischen einfachen, kontaktplangesteuerten SPS-Steuerungen und hochkomplizierten, wie kostenaufwendigen, Automationssystemen. In der Praxis werden diese Systeme immer unübersichtlicher und dessen Handhabung bleibt oft nur noch wenigen Softwarespezialisten vorbehalten, welche regelmäßig Fortbildungskurse bei den jeweiligen Herstellern belegen und teure Up-Dates nachbestellen und einpflegen müssen.

Für SEPIA benötigen Sie weder ein Seminar noch teure Up-Dates. Diese Dokumentation reicht vollkommen aus, da während der Entstehung alle Beispiele an praxisorientieren Aufbauten ausprobiert wurden. Diese lassen sich bei entsprechender Nachbildung sehr einfach wiederholen und nach eigenen Vorstellungen im Rahmen der angeboteten Möglichkeiten erweitern. Alle Beispiele wurden dazu an der Vollausstattung ausgetestet.

Die in dieser kleinen Dokumentation gezeigten Beispiele können, je nach Revisionsstand, in folgenden Angaben zu Ihrem Modell abweichen: BIOS-Version, Datum, Debugger/Monitor- und IDE-Version, Advanced-BASIC Version, RTC-Zeit und Datum, sowie Variablen wie RND und CPU-Temperatur. Die hier gezeigten Beispiele können nur dann genutzt werden, wenn die entsprechenden Erweiterungskarten im System physisch vorhanden sind. Die einzelne Komponentenadressierung orientiert sich dazu nach unserer MAPALL.ASM Empfehlung.

Symbole:

Alle wichtigen Hinweise sind mit diesem Kennzeichen versehen: 

Zahlengenauigkeit, vorbelegte Variablen, Strings und ASCII-Zeichen

Bei der Programmierung von BASIC dürfen, ähnlich, wie bei einem Taschenrechner, bestimmte Grenzen und Bereiche nicht überschritten werden, da für alle Variablen nur ein bestimmter Speicherplatz zu Verfügung steht. Zudem werden bestimmte Variablen, wie die Kreiszahl „Pi“, mit festen Zahlen intern vorbelegt.

Hinweis: Das „**E**“ bedeutet x $^{10 \text{ hoch}}$

Bit	0/1 = Null oder Eins	
Character	0...127 (bsp. ASCII-Zeichen)	
Byte	00..FF, 0...255 Dez. oder 00000000...11111111 Bin.	
Ganze Zahlen, ohne Dezimalpunkt	-32768 bis +32767 einschl.	
Genauigkeit	-1.701411**E**+38 bis +1.701411**E**+38 einschl.	
PRINT EXP(88) <CR>	Anzeige: 1.651636255E+38	
Print EE <CR>	= Eulersche Zahl	Anzeige: 2.7182818285
Print Pi <CR>	= Kreiszahl	Anzeige: 3.1415926536
Zahlengenauigkeit	max. 12 Stellen	
Beispiel für Exponent 234.988E-7	= 0.0000235988	
Beispiel für Exponent 2359E6	= 2359000000	

int. Speicherplatzbedarf	
BASIC-Zeile	min. 5 Byte
Zeilennummer	min. 2 Byte
Pointer	min. 2 Byte
CR Zeilenvorschub	1 Byte
Stringvariable	min. 6 Byte
Aktive FOR-NEXT Schleife	16 Byte
GOSUB, ohne Return	6 Byte
String Bereich	max. 255 Zeichen pro String-Variable
BASIC-Zeilennummern	0 bis 65529
Länge einer Programmzeile	bis zu 255 Zeichen
Speicher	64 kB gesamt (ohne Erweiterung MEM-Karte)

Wichtig für Portzugriff, bsp. OUT, INP:		
Hexadezimale Zahlen, 8 bit	&00...&FF	entspricht 0...255
Hexadezimale Zahlen, 16 bit	&0000...&FFFF	entspricht 0...65535

Sprachen	
Ausgabemeldungen	deutsch
BASIC Programmierung	englisch
Unterstützte ASCII-Zeichen	US Zeichensatz ohne Umlaute (US/DE Tastatur)

Sonstiges	
VGA ASCII-Zeichenzatz	7/8 Bit, 0...127, ohne Umlaute
VGA Zeichen p.Sekunde	5000 bei 115.200 kBaud
Schleifenbefehl p. Sekunde	~ 4500 (0...100000 For-Next-Schleife bei 20 MHz)

In SEPIA verwendete Standard BASIC-Befehle

```
A: ABS, AND, ASC, AUTO, ATN
B: BYE
C: CHR, CLEAR, COS, CALL, CLOSE, CONT
D: DATA, DIM
E: EE, EDIT, ELSE, END, ERL, ERR, ERR1, ERROR, EQV, EXP, EXIT
F: FOR, FRE, FIX, FIX$, FIND
G: GOTO, GOSUB
H: HEX
I: INP, INPUT, IF, IN, INP, INSTR, INT, IMP
K: KILL
L: LET, LVAR, LIST, LOG, LOG10, LEN, LEFT$, LOCATE, LINE
M: MID$, MOD
N: NEW, NEXT, NOT
O: OUT, OUTBYTE, ON, OR
P: PI, POKE, POS, PEEK, PRINT, PRECISION, PRIVACY
R: READ, RESTORE, RETURN, RANDOMIZE, REM, RESUME, RENUMBER,
   RUN, RND, RND1, RIGHT$, REPLACE
S: STOP, SIN, SGN, SQR, STR$, STRING$, STEP, SPC
T: TAN, TRACE, TO, THEN, TAB
U: USING
V: VAL, VARPTR
W: WAIT
X: XOR
```

Eine ausführliche Erläuterung zu den hier verwendeten BASIC-Befehlen finden Sie im HEBAS Benutzerhandbuch. Da HEBAS modular strukturiert wurde, sind in dieser Unterlage möglicherweise weitere Befehle erläutert, die in SEPIA jedoch aus Platzgründen nicht implementiert wurden. Dafür bietet SEPIA jedoch spezielle Advanced-Befehle für die Automation, auf welche wir in dieser Anleitung näher eingehen werden.

Im Anschluss finden Sie eine Übersicht zum erweiterten BASIC-Befehlssatz, welcher fortan Advanced-BASIC genannt wird. Manche Advanced-BASIC Beispiele können jedoch nur bei Verwendung entsprechender I/O-Hardware genutzt werden. So sind beispielsweise alle Laufwerks-Befehle wie DRIVE, DIR, LOAD, LOADGO und SAVE nur nutzbar, wenn SEPIA über ein IDE-Interface mit Datenträger (bsp. CF-Card) verfügt.

Wichtiger Hinweis:
SEPIA prüft während des bootens nach, ob IDE-Interface und Datenträger vorhanden sind. Fällt die Prüfung negativ aus, wird eine entsprechende Mitteilung ausgegeben. So auch bei der RTC-Karte, da diese neben dem WatchDog eine Echtzeituhr enthält und bei fehlender Hardware keine Angaben zu Uhrzeit und Datum erzeugen kann. Die Nutzung der restlichen Hardware ist davon nicht betroffen, da im Remote I/O-Betrieb die IDE und RTC mit WDC möglicherweise nicht benötigt werden. Dies gilt gleichfalls für den VGA- und Ethernet-Port, da eine Remote-Verbindung direkt über den RS232-Port der CPU-Karte erfolgen kann.

SEPIA Advanced BASIC

VGA

```
XTCOLOR V,H              ; Textfarbe 0..9 setzen, V=Vordergrund, H=Hintergrund
XRSTCOL                  ; Textfarbe rücksetzen, org. Ausgabe Weiß auf Blau
XINVERT                  ; Textfarbe invertieren (rücksetzen mit XRSTCOL)
XSTSON                   ; Statusmeldung VGA-Terninal EIN
XSTSOFF                  ; Statusmeldung VGA-Terninal AUS
XGR.......               ; siehe Sonderbefehle VGA Grafik
```

A/D Karte

```
XSETAD1(Port),Kanal      ; AD1-Karte Messkanal 1...16 auf MUX einstellen
x=XGETAD1(Port)          ; ließt von AD1 16 Bit Wandlung nach Variable x, oder
x=XRADC(Kanal*256+Port) ; Kanal mit Port übergeben und Ergebins nach x lesen
```

D/A Karte

```
XDAC(16Bit_Wert),Port    ; DA1-Karte dig. 12 Bit-Wert auf DAC-Port einschreiben
XDASET,BasisPort         ; DAC Latch und Freigabe für Output-Switch
```

COM Port

```
XCOUT(x),byte            ; ein Byte auf COMx senden (x = 1,2,3,4,5,6,7)
byte = XCIN(x)           ; ein Byte von COMx empfangen
A$ = XSTRING(x)          ; empfängt eine Zeile (bis 80 Zeichen) von COMx
XSTROUT(COMx),A$         ; sendet einen String auf COMx
```

I/O diverse

```
XIPEAK(byte),time,mode   ; erzeugt auf I/O-Adresse(byte) einen zeitlich def. Impuls
XPULSE(byte),word,mode   ; erzeugt auf I/O-Adresse(byte) bis 65535 Burst/Pattern
XGETEMP                  ; Text Ausgabe gibt CPU-Temperatur auf Konsole aus
XOUTWD(word),byte        ; auf 16Bit I/O-Adresse ein byte schreiben
byte = XINPWD(word)      ; auf 16Bit I/O-Adresse ein byte lesen
XWAITKEY                 ; Programm wartet endlos auf einen Tastendruck
XDELAY(word)             ; Delay 0...65535 (50000 = ca. 10 Sek.)
XPBELL byte,byte         ; var. Ton erzeugen: Tonhöhe, Tonlänge
XGETIME                  ; Text Ausgabe gibt Uhrzeit auf Konsole aus
XGEDATE                  ; Text Ausgabe gibt Datum auf Konsole aus
byte = XGETSEC(0)        ; von RTC0 aktuelle Sekunde nach Variable lesen
byte = XGETMIN(0)        ; von RTC0 aktuelle Minute nach Variable lesen
byte = XGETSTD(0)        ; von RTC0 aktuelle Stunde nach Variable lesen
XIRQON / XIRQOFF         ; Schaltet Tick Interrupt-Generator von RTC-Karte Ein/Aus
```

MEM - Befehle

```
XPOKE(Adresse),byte      ; Byte auf ext. MEM-Karte (hohe Adresse) schreiben
byte = XPEEK(Adresse)    ; Byte von ext. MEM-Karte (hohe Adresse) lesen
XSEC(x)                  ; 64k Segmentadresse vorwählen (x = 1...15)
```

Disk-Direktbefehle auf IDE Interface mit CF/SD Card

```
XROOT x                  ; x = Verzeichnisnummer (0 ... max.255)  0 = Root
RESET                    ; IDE-Medien neu initialisieren (IDE RESET-Command)
DIR                      ; Directory - Verzeichnis ausgeben
LOAD                     ; Programmdatei laden, gefolgt von Dateinummer
SAVE                     ; Programmdatei abspeichern, gefolgt von Dateinummer
RENAME                   ; Programmdatei umbenennen, gefolgt von Dateinummer
ERASE                    ; Programmdatei löschen, gefolgt von Dateinummer
LOADGO                   ; läd Programmdatei über Nummer und startet Anwendung
DRIVE 0                  ; auf Laufwerk 0 umschalten (IDE-Port0: Master)
DRIVE 1                  ; auf Laufwerk 1 umschalten (IDE-Port1: Slave)
DRIVE S                  ; Anschluss auf vorhandene Medien (Master & Slave) prüfen
DRIVE P                  ; Festplatte parken & in sleep-mode versetzen
DRIVE T                  ; Ausgabe Drive Status & Error Register, für Testzwecke
```

SEPIA Advanced BASIC

```
Farbtabelle zu Befehl: XTCOLOR V,H
je 10 VGA Textfarben für Vorder- und Hintergrundfarbe

          ; Farbe Vordergrund                ; Farbe Hintergrund
          ; 0 = Schwarz                      ; 0 = Schwarz
          ; 1 = Rot                          ; 1 = Rot
          ; 2 = Grün                         ; 2 = Dunkel-Grün
          ; 3 = Gelb                         ; 3 = Oliv/Khaki-Braun
          ; 4 = Blau                         ; 4 = Dunkel-Blau
          ; 5 = Rosa                         ; 5 = Mangenta
          ; 6 = Cyan                         ; 6 = Dunkel-Cyan
          ; 7 = Grau                         ; 7 = Hell-Grau
          ; 8 = Amber                        ; 8 = Hell-Blau
          ; 9 = Hell-Lila                    ; 9 = Orange
```

Grafik auf VGA-Monitor

Die Ausgabe von einfachen Diagrammen und Kurven erfolgt über einen gesonderten VGA-Mode mit 640 x 240 Pixel in zwei von 64 Farben. Damit die BASIC-Programmierung möglichst übersichtlich bleibt, wurden dazu folgende Advanced-Befehle erweitert:

```
XGRON                 ; schaltet VGA-Monitor in den Grafik-Mode
XGROFF                ; schaltet VGA-Monitor wieder in den Text-Mode zurück
XGRESCOL              ; setzt die Farbpalette zurück (original)
XGRCLS                ; löscht die gesamte Grafik
XGRSTCOL V,H          ; legt Vordergrund-Farbe und Hintergrund-Farbe fest
XGRTEST               ; erzeugt ein statisches Testbild mit Farbbalken
XGRPSET X,Y           ; zeichnet einen Pixel an der Koordinate X,Y
XGRLINE X1,Y1,X2,Y2   ; zeichnet eine Linie von X1,Y1 nach X2,Y2
```

Weitere Befehle und Beispiele:

```
A$ = INKEY$           ; ließt ein Zeichen von Tastatur (COM1) nach A$ ein
CLS                   ; Löscht Bildschirm + Cursor Pos. 1,1 (ANSI-BBS)
LOCATE x,y            ; Text-Cursor positionieren (ANSI-BBS)
BEEP                  ; kurzer Quittungston auf CPU-Karte ausgeben
RINGBELL              ; wie BEEP, jedoch mit ASCII-Code 07h auf COM1
var = INP(Port)       ; direkte Eingabe von I/O-Port (bsp. Optokoppler)
OUT(Port),byte        ; direkte Ausgabe auf I/O-Port (bsp. Relais)
var = INP(Port)       ; direkte Eingabe von I/O-Port lesen
OPTION                ; Text Ausgaben formatieren
CTRL <E>              ; Tastenkombination für Programmabbruch
NEW                   ; BASIC-Programm im Speicher (RAM) löschen
BYE                   ; BASIC verlassen, zum Debugger-Menue wechseln
LIST                  ; BASIC-Listing anzeigen (Pause mit Leertaste)
RENUMBER <100,20>     ; Listing Zeilennummer neu formatieren
CONT                  ; nach Abbruch, Programm weiter ausführen
RUN                   ; ab Programmanfang starten
STOP                  ; Programm abbrechen
AUTO                  ; Automatische Zeilennummer bei Eingabe
ERROR <1..48>         ; Fehlermeldung von Nr. ausgeben
REPLACE               ; Programmtext ersetzen (zeilennummerorientiert)
TRACE                 ; Programmzeilen im Einzelschritt mitverfolgen
WAIT                  ; Ließt I/O-Port so lange, bis Bedingung erfüllt ist
```

EPROM-Inhalt, X-Sonderbefehle

SEPIA Advanced-BASIC besitzt ein ca. 24 kB großes Standard-BASIC, plus 8 kB erweiteres (Advanced) BASIC mit Sonderfunktionen. Sonderbefehle kennzeichnen sich dadurch, dass diese mit einem „X“ wie XPOKE beginnen und mit keinem „normalen“ BASIC vergleichbar, aber dennoch leicht verständlich sind.

Machen Sie dazu gleich den ersten Test und schalten Sie dazu SEPIA ein:

- SEPIA booten (ohne eingeschobene CF-Card)

```
***** S E P I A  -  B O O T *****
---------------------------------
BIOS 3.30 (C) H.Kolter & Dr. Hehl
24.02.2012

Init SIO ..... fertig
Init PIO ..... fertig
Kein HDD/CF/SD Zugriff

22:03:42  26.02.2012  Dienstag

CPU +022 Grad C.

<ESC>    = Neustart
    D    = Debugger
    L    = RAM loeschen
    K    = BASIC Kaltstart
    W    = BASIC Warmstart
    R    = REMOTE I/O
>_
```

Der Nutzer muss jetzt eine Eingabe tätigen: Taste D, L, K, W oder R.

Im ersten Menue (s.o.) bitte zunächst die Taste „K“ für Kaltstart wählen/eingeben.

Ausführliche Erläuterungen finden Sie dazu auf der nächsten Seite.

Booten von SEPIA nach einem RESET

Das System lässt sich grundsätzlich in zwei unterschiedlichen Betriebszuständen anwenden: Host- oder Remote-I/O-Mode. Im Host-Mode können zusätzliche Erweiterungen wie IDE, Ethernet, VGA, PS/2, MEM, sowie RTC mit WatchDog, oder weitere COM-Schnittstellen ergänzt werden. Im Remote-I/O-Betrieb sind diese Erweiterungen nicht zwingend notwendig, da alle Steuerbefehle von einem externen Rechner (i.d.R. PC) über eine COM-Schnittstelle erfolgen. Die Kommunikation der Steuerbefehle erfolgt in Klartext mittels Text-String$, so dass andere Laboranwendungen, oder aus Hochsprachen heraus, leichten Zugriff auf SEPIA erhalten. Alle Befehle sind dazu als Direkt-Befehl anwendbar. Für eigenständige Applikationen verwendet das System im Host-Betrieb, neben dem BIOS, ein kleines OS-Betriebssystem mit Debugger + Monitor für Maschinensprache, sowie ein textbasierter Advanced-BASIC-Interpreter für die Automation. Die Unterscheidung, ob Host- oder Remote-I/O-Betrieb, trifft der Anwender in der Startmenue-Abfrage. Im Host-Mode erkennt ein „Such-Modus“ automatisch, wenn eine CF-Karte (am IDE-Interface) eingeschoben ist, welche eine AUTOBAS Start-Datei enthält. So kann gleich nach dem Booten eine Initialisierung stattfinden, oder weitere Programme nachgeladen werden.

Nach einem Hardware-RESET startet das SEPIA-BIOS das Startmenue mit verschiedenen Eingabemöglichkeiten, die der Nutzer wählen kann. Während des bootens prüft das BIOS mehrere Schaltungen intern ab, um festzustellen, ob beipielsweise eine RTC oder IDE Karte verbaut wurde. Bei Fehlen dieser Hardware wird automatisch eine entsprechende Meldung auf die Konsole (COM1) generiert. Der Nutzer hat nun die Möglichkeit zwischen:

```
<ESC>    = Neustart
    D    = Debugger
    L    = RAM loeschen
    K    = BASIC Kaltstart
    W    = BASIC Warmstart
    R    = REMOTE I/O
```

auszuwählen, damit der weitere Verlauf seiner Programmierung stattfinden kann. Bei betätigen der ESC-Taste wird ein erneuter RESET ausgeführt (Warmstart), bei „D“ gelangt man direkt in den Debugger-Modus mit speziellen Befehlen, die eine maschinennahe Programmierung und Analyse zulässt. Die Taste „L“ löscht den gesamten SRAM-Speicher, so dass hier keine zufälligen Datenfragmente entstehen. „K“ startet den BASIC-Interpreter im Kalt-Modus, so dass alle internen BASIC-Variablen gelöscht und weitere Grundinitialisierungen vorgenommen werden. Mit der Taste „W“ werden diese Initialisierungen übergangen, so dass ein noch vorhandenes BASIC-Programm im SRAM-Speicher nutzbar bleibt. Mit dem „LIST“-Befehl kann man im BASIC-Modus dieses Progamm dann einsehen und bearbeiten. Somit besteht die u.a. Möglichkeit, auch nach einem Hardware-RESET (bei nicht unterbrochener Spannungsversorgung) ein BASIC-Prgramm im Speicher weiter nutzen zu können.

Die Taste „R“ startet den BASIC-Interpreter ohne Zeilennummer und Rückmeldungen, so dass hierbei eine einfache Remote-Kommunikation auf Basis der BASIC-Befehle über die COM Konsolenschnittstelle stattfinden kann. Da in diesem Modus ebenfalls alle BASIC-Befehle verwendet werden können, steht dem Nutzer eine Vielfalt von Einsatzmöglichkeiten, zur Ansteuerung der Hardware, zur Verfügung. Alle I/O-Leitungen können in dieser Betriebsart über „INP“ und „OUT“ oder den X... Advanced-Befehlen direkt angesteuert werden.

Mein erstes Programm...

Das BASIC meldet sich zunächst mit einer Abfrage zur Speichergrenze, da man beipielsweise später noch Maschinenprogramme in den oberen RAM-Speicher hinterlegen kann.

Drücken Sie zunächst nach „Speichergrenze“ die Return/Enter-Taste, bis folgende Meldung auf dem Monitor erscheint:

```
BASIC gestartet

Speichergrenze ? _

31821 Bytes frei

ADVANCED BASIC 6.02 (C) Dr.Hehl & H.Kolter

Fertig

_
```

Geben Sie nun folgenden Advanced-BASIC-Befehl mit abschließender Return/Enter-Taste ein:

```
XGETEMP
```

SEPIA meldet nun die aktuelle CPU-Temperatur (in diesem Fall +32 Grad Celsius)

```
+032 Grad C.
Fertig

_
```

Im nächsten Beispiel wollen wir nun die CPU-Temperatur wiederholt auf dem Bildschirm fromatiert ausgeben lassen, bis wir die Endlosschleife mit der Tastenkombination „CTRL+E“ manuell unterbrechen.

Weiter auf der nächsten Seite >>>

XGETEMP - Advanced Befehl

Da der BASIC-Programmspeicher noch leer ist, können wir gleich die ersten Programmzeilen eingeben. Löschen Sie aber zunächst den Bildschirm mit der folgenden Eingabe:

CLS gefolgt von der Return/Enter-Taste...

und geben Sie nach der „Fertig-Meldung“ folgende Programmzeilen ein:

```
Fertig

100 CLS
110 LOCATE 10,10
120 XGETEMP : PRINT
130 GOTO 110

RUN
```

Das Programm können Sie nun mit dem RUN-Befehl starten, gefolgt von der Return/Enter-Taste...

Die Monitorausgabe der CPU-Temperatur erfolgt jetzt permanent in der 10ten Zeile an der 10ten Spalte, bis eine manuelle Unterbrechung mit der Tastenkombination „CTRL+E“ (für End = Abbruch) erfolgt.

Durch ändern der X- und Y-Position im LOCATE-Befehl, lässt sich die Textausgabe auch an anderer Stelle ausgeben. Das in X-Richtung große Spaltenfeld hat max. 80 Zeichen, das in Y-Richtung 36 Zeilen. Größere Angaben sind nicht erlaubt. Die eingegebene X - und Y - Position zeigt immer auf das erste Zeichen einer Textausgabe.

Hinweis zur Hardware:
Der Temperatursensor befindet sich im 40 pol. DIL-Sockel unterhalb der CPU. Es handelt sich hierbei um einen TMP04 im TO92 Gehäuse der Fa. Analog Device. Der Sensor erzeugt dazu Rechteck-Impulse, die in einer Statemachine vom CPLD-IC in Echtzeit zu einer 32-Bit Zahl zeitlich vorberechnet wird. Die Differenz ergibt anschließend einen 16-Bit Wert, welcher proportional zur aktuellen Temperatur intern zur Verfügung steht. Damit der Wert in Grad Celsius angezeigt werden kann, muss die CPU diesen Wert entsprechend umrechnen. Hierzu wird ein Divisor und ein -Quotient benötigt, welche auch im BIOS-EPROM als Fest-werte zur späteren Kalibration hinterlegt sind.

Der Devisor liegt auf der EPROM-Speicherstelle 0020h, der -Quotient auf 0023h.

Normales Beispiel zur Temperaturausgabe :

```
100 REM CPU-Temperatur ausgeben
110 CLS
120 LOCATE 10,5
130 PRINT „CPU-Temperatur = „; : XGETEMP
140 PRINT
150 XDELAY (10000)
160 GOTO 120
RUN
```

Hinweis zum Programm
Der Befehl XDELAY (arg. < 65535) verzögert den Programmablauf mit einer Variablen zwischen 0 und 65535. Während XDELAY ausgeführt wird, ist die Tastenkombination mit CTRL-E wirkungslos, da die Maschinenroutine in einer geschlossenen Schleife (DJNZ-Loop) ausgeführt wird, bis der Zählerstand erreicht ist. Die Variable wurde nur 16 Bit breit ausgelegt, damit keine unendlichen Verzögerungen programmiert werden können. Benötigt man dennoch längere Zeiten, kann der Befehl mehrfach hintereinander angewendet werden.

Bei einem 20 MHz-System liegt die XDELAY-Verzögerung bei dem Wert:

50000	bei etwa 10 Sekunden
5000	bei etwa 1 Sekunde
1000	bei etwa 200 ms
100	bei etwa 20 ms
10	bei etwa 2 ms

Ausführliches Beispiel: CPU-Temperatur über direkte I/O-Registerabfrage manuell in BASIC berechnen

```
100 REM TMP04 Temperatursensor Messbereich -40...+120 Grad
110 REM Kalibrierte Temperaturkurve
120 :
140 OUT &18,8+4              : REM RESET internal 16bit counter
160 OUT &18,2+1              : REM enable counter + single shot meas
180 OUT &18,0                : REM get counts in 2 register from sensor
200 XDELAY(500)              : REM wait
220 :
240 A = INP(&18)             : REM INP-Reihenfolge nicht veraendern !!!
260 B = INP(&19)
270 C = INP(&1A)
280 IF C > 0 THEN GOTO 140   : REM bei ERROR ODER UEBERLAUF wiederholen
300 TY = A + (B*256)         : REM 16 bit Wert über zwei 8 bit Resister bilden
320 TY = TY / 68             : REM Steilheit der Temperaturkurve (Divisor)
340 TY = TY - 192            : REM Grund-Offset zur Kalibration
350 :
360 PRINT USING „CPU Temp. = ###.#“, TY
380 GOTO 140
```

Quittungton ausgeben

Auf der CPU-Karte befindet sich ein kl. Piezo-Lautsprecher, um beispielsweise einen Quittungston zu erzeugen. Neben dem typischen BELL-Befehl, kann die Tonhöhe und Dauer durch folgenden Advanced-Befehl stattfinden: XPBELL 200,100.

In diesem Beispiel wurde die Tonhöhe 200 und Tondauer 100 vorgegeben. Die variablen Werte werden als Byte dezimal übergeben (0..255).

Geben Sie das nachfolgende BASIC-Programm ein und starten Sie mit **RUN**

```
100 REM Tonausgabe programmieren
110 XPBELL 50,250
120 XPBELL 80,250
130 XPBELL 100,250
140 XPBELL 120,250
150 XPBELL 140,250
160 XPBELL 160,250
170 XPBELL 180,250
180 XPBELL 200,250
190 XPBELL 255,255
```

Uhrzeit und Datum

Wenn Sie eine RTC-Karte verwenden, meldet sich SEPIA beim booten mit Zeit und Datum.

Die RTC-Karte beinhaltet, neben einer COM2 und LPT2 -Schnittstelle, einen Uhren-Chip und eine WDC WatchDog-Schaltung. Damit man den Uhren-Chip auch unter BASIC auslesen kann, wurden zwei Advanced-Befehle integriert: XGETIME und XGEDATE. Diese Befehle erzeugen eine einfache Print-Textausgabe der aktuellen Uhrzeit, oder dem Datum.

Geben Sie das nachfolgende BASIC-Programm ein und starten Sie mit **RUN**

```
100 REM Uhrzeit & Datum unter BASIC ausgeben
110 PRINT "Zeit STD:MIN:SEK = "; : XGETIME
120 PRINT
130 :
140 PRINT "Datum TAG:MONAT:JAHR = "; : XGEDATE
150 PRINT
160 :
170 XDELAY(5000)
180 GOTO 110
```

Hinweis zum Programm
Der Befehl XDELAY(word) ermöglicht eine Ablaufpause.
GOTO 110 startet das Programm ab BASIC-Zeile 110 erneut.

Damit der Anwender die Uhrzeit über eine Variable mit in sein Programm einbinden kann, stehen andere Advanced-Befehle zur Verfügung: XGETSEC, XGETMIN und XGETSTD.

Geben Sie das nachfolgende BASIC-Programm ein und starten Sie mit **RUN**

```
100 CLS
110 A = XGETSEC(0)       : REM Liesst Sekunde von RTC 0
120 B = XGETMIN(0)       : REM Liesst Minute von RTC 0
130 C = XGETSTD(0)       : REM Liesst Stunde von RTC 0
140 :
150 LOCATE 10,10         : REM Cursorposition X 10, Y 10
160 :
170 PRINT "Uhrzeit = "; C; B; A; "     "
180 XDELAY(5000)
190 GOTO 110
```

Textausgabe und Tastatur/Zeichen-Eingabe

Sie können SEPIA im Host- oder Remote-Modus betreiben. Im Host-Betrieb (ähnlich einem PC) werden alle Ein/Ausgaben z.B. auf den VGA-Port der NET-Karte übertragen. Umgekehrt werden Tastatureingaben empfangen und der Konsole zur Verfügung gestellt. Im Remote-Betrieb werden diese Ein/Ausgaben über den COM1-Port der CPU-Karte behandelt. Die Zuweisung der Konsole erfolgt durch Festlegung per DIP-Schalter auf den E/A-Karten. Durch vertauschen der COM-Hardware-Adressen können somit Verbindungen zur Konsole, sowie über andere Karten, stattfinden. Die COM1 Schnittstelle ist dazu für die Konsole reserviert.

Um im Host-Betrieb (NET-Karte muss vorhanden sein) Tastatureingaben unter BASIC zu nutzen, kann man folgende Befehle anwenden:

Geben Sie das nachfolgende BASIC-Programm ein und starten Sie mit **RUN**

```
100 REM INKEY$ und XWAITKEY TEST
110 CLS : LOCATE 2,2
120 PRINT "Eine Taste druecken..."
130 :
140 A$ = INKEY$ : IF A$ = "" THEN GOTO 140
150 :
160 PRINT "Die Taste war: "; A$
170 BEEP
180 PRINT "VARPTR-Zeiger: "; VARPTR(A$)
190 PRINT "Taste fuer weiter..."
200 XWAITKEY
210 GOTO 100
```

Der Advanced-Befehl „XWAITKEY“ unterbricht den Programmablauf und erwartet eine Tasteneingabe durch den User.

Eine andere Variante zum Vergleich:

```
100 REM Tastatur ASCII-Zeichencode anzeigen
110 CLS
120 LOCATE 2,2
130 B = ASC(INKEY$)
140 PRINT „ZEICHEN „; CHR$(B); „ DEZ.:“;B
150 IF B = 5 THEN STOP ELSE GOTO 130
```

Zeichensatz und Datenübertragung (1)

Das nachfolgende Programm zeigt, wie der Zeichensatz auf der Konsole (sprich auf COM1 über die ser. Schnittstelle, oder dem VGA-Bildschirm der NET-Karte) erzeugt werden kann.

Geben Sie das nachfolgende BASIC-Programm ein und starten Sie mit **RUN**

```
100 REM Zeichensatz ausgeben
110 For X = 0 to 23
120 PRINT
130 NEXT X
150 PRINT CHR$(27);"[1J"
160 PRINT CHR$(27);"H"
170 PRINT CHR$(27);"[0,0H"
180 :
190 FOR Z = 30 TO 255
200 PRINT Z; CHR$(Z); "   ",
210 NEXT Z
```

Mit dem Advanced-BASIC-Befehl „XSTROUT(comnr.1..7)“ lassen sich ASCII-Zeichen als String zu einer anderen COM-Schnittstelle einfach umlenken bzw. ausgeben.

Beispiel:

```
100 DIM A$(10)          : REM Konsole -> Ethernet
110 A$ = „Hallo...“     : REM Zeichenkette nach A$
120 XSTROUT(6),A$       : REM String auf COM6 senden
```

Es geht auch umgekehrt:

```
100 DIM A$(80)          : REM Ethernet -> Konsole
110 A$ = XSTRING(6)     : REM lese String auf COM6
120 Print A$            : REM nach <CR> String auf Konsole
```

Oder zur Ausgabe ein einzelnes Zeichen auf COM1...7:

```
100 A = 66              : REM Character ASCII-Zeichen „B“
110 XCOUT(7),A          : REM Zeichen auf COM7 ausgeben
```

Zeichensatz und Datenübertragung (2)

Möchte man die Schnittstelle extern überprüfen, kann man beispielsweise Daten als Echo auf dem gleichen COM-Port zurücksenden.

Geben Sie das nachfolgende BASIC-Programm ein und starten Sie mit **RUN**

```
100 REM Daten von COM2 empfangen und als Echo zuruecksenden
110 :
120 Z = XCIN(2)   : REM Zeichen (Byte) von COM2 empfangen
125 :
130 IF Z > 0 THEN PRINT CHR$(Z); : REM auf Konsole
135 :
140 XCOUT(2),Z    : REM Zeichen auf COM2 zurück senden
145 :
150 IF Z = 13 THEN PRINT : REM LF bei CR ausgeben
155 :
160 GOTO 120      : REM Loop Endlosschleife
```

LOADGO - Befehl

Bei Verwendung der IDE-Karte mit einem Datenspeicher (CF oder SD Card) kann mit dem Direktbefehl „LOADGO(x)“ aus einem laufenden BASIC-Programm ein anderes Programm geladen und gestartet werden. Somit lassen sich beispielsweise auch mehrere Programme über eine eigene Menuestruktur verwalten, oder ggf. größere Programme in mehrere, kleine Unterprogramme zerlegen, die dann je nach Bedarf nachgeladen werden. Da alle BASIC-Programme über Programmnummern verwaltet werden, wird die Zieldatei als Nummer zugewiesen. Unabhängig davon, darf der Dateiname bis zu 32 ASCII-Zeichen beinhalten. Die Dateinummer erfolgt hierbei in hexadezimaler Schreibweise (&00 ... &FF).

Geben Sie das nachfolgende BASIC-Programm ein und starten Sie mit **RUN**

```
100 REM Beispiel für LOADGO(&xx)
110 CLS
120 LOCATE 2,3
130 PRINT "Menue mit Unterprogrammen"
140 PRINT "-------------------------"
150 PRINT
160 PRINT "1 - MASTER-SLAVE         "
170 PRINT "2 - ZIDE6                "
180 PRINT "3 - RELAIS TEST          "
190 PRINT "4 - TONERZEUGUNG         "
200 PRINT
210 :
220 INPUT W$
230 :
240 ON VAL(W$) GOTO 260,270,280,290
250 :
260 LOADGO(&06) : REM Starte Dateinummer
270 LOADGO(&04) : REM Starte Dateinummer
280 LOADGO(&15) : REM Starte Dateinummer
290 LOADGO(&17) : REM Starte Dateinummer
```

Anmerkung:
Die Dateien &06, &04, &15 und &17 müssen auf dem Datenträger vorhanden sein und jeweils ein startfähiges BASIC-Programm beinhalten.

Datei-Verzeichnis ausgeben:
Nutzen Sie den „DIR“-Befehl, um das gesamte Directory des Datenträgers anzuzeigen.

MEM - Segmentadressierung und externe Speicherverwaltung

Damit das SEPIA-BASIC auf den erweiterten Speicher zugreifen kann, wurde zusätzlich eine kleine MMU in das CPLD integriert. Sie ermöglicht eine Segmentadressierung mit 16k-Pages zu je vier Blöcken pro 64k Segment. Die Speicherverwaltung wird über die I/O-Adresse 0000h gesteuert. Hier wurden nun neben den Adressleitungen A16...A19 und BANKEN ein PG0-Bit und PG1-Bit für das Mapping erweitert, welche vier 16k-Pages je nach Bedarf umschaltet. Der Grund für diese Maßnahme war, dass man größere Datenmengen (z.B. Messdaten) nicht in das eigene BASIC-RAM ablegt, sondern in einen Schattenspeicher der genug Platz bietet.

Damit der Anwender sich nicht um die Umschaltung der einzelnen Pages kümmern muss, wurden im Advanced-BASIC weitere Befehle integriert: „XPOKE“ und „XPEEK“. Hiermit lassen sich direkt 64k Schreib- und Leseoperationen je Segment komfortabel handhaben. Die Segementumschaltung erfolgt zuvor über den Befehl „XSEG 1...15“.

Geben Sie das nachfolgende BASIC-Programm ein und starten Sie mit **RUN**

```
100 CLS : LOCATE 2,2
110 PRINT „SEPIA SRAM & MEM-CHECK 32/512/1MB „
120 PRINT „================================ „
130 PRINT
140 POKE (&F000),&A5
150 B = PEEK(&F000)
160 IF B = (&A5) THEN PRINT „32 KB Programmspeicher o.k.“
170 PRINT
200 :
210 FOR X = 1 TO 15
220 XSEG(X)
230 XPOKE (&F000),&A5
240 B = XPEEK(&F000)
250 IF B = (&A5) THEN PRINT „Test 64k Segment „;X
260 NEXT X
```

Anmerkung:
Um die Funktionen der Speichererweiterung nutzen zu können, benötigen Sie die MEM-Karte mit mindestens einem SRAM Speicher-Chip.

Der Befehl „XSEG“ gibt die jeweilige Segmentadresse (64k Block) vor. Die absolute RAM-Adressierung muss gemäß der tatsächlichen Speicherbestückung entsprechen.

WDC Watchdog auf RTC-Karte programmieren

Die gesamte WDC-Programmierung erfolgt über drei I/O-Register: OUT &2C,x (0..15) setzt 16 verschiedene Zeitfenster für den Time-out, OUT &2D,1 erzeugt die Freigabe der WDC-Funktion, und mit OUT &2F,0 wird ein Re-Trigger programmiert, welcher vor Ablauf des Zeitfensters erfolgen muss, damit kein RESET ausgelöst wird. Das Relais schaltet mit einer Haltedauer von ca. 1 Sekunde (bei 8 MHz CPU-Takt) und löscht dabei das Time-out-Flag im CPLD, sodass keine Rückkopplung mit der RESET-Logik entsteht. Die RESET-Schaltung der CPU-Karte übernimmt anschließend alle Aufgaben der Neuinitialisierung (Warmstart). Der WDC-Befehl OUT &2D,0 schaltet die Funktion manuell ab und setzt das Time-Out-Flag wieder auf Null (disable), so dass man gezielt Routinen mit - oder - ohne WDC-Unterstützung programmieren kann.

Bei einer gewollten Unterbrechung durch den Anwender (Tastenkombination „CTRL-E“) wird über das BIOS das WDC-Flag ebenfalls rückgesetzt, da sonst bei jeder Umprogrammierung des BASIC-Quellcodes ein unnötiger CPU-RESET durchgeführt würde.

Geben Sie das nachfolgende BASIC-Programm ein und starten Sie mit **RUN**

```
100 REM WDC-TEST (Watch-Dog)
110 OUT (&2C),6   : REM Timeout Zeitbasis waehlen 0..15
120 OUT (&2D),1   : REM 1=Enable 0=Disable WDC
130 :
140 OUT (&2F),0   : REM Re-Trigger vor Timeout
150 :
160 REM Hier steht eingekapseltes Hauptprogramm
170 XDELAY(2000)  : REM simuliert die Programmdauer
180 :
190 GOTO 140      : REM WDC-Schleife
```

Anmerkung zur Hardware:
Abhängig vom Master-Clock wird ein Frequenzteiler mittels Vergleicher im CPLD intern vorprogrammiert, um verschiedene Time-Out Zeitfenster zu bestimmen. Ein weiteres Kontroll-Register schaltet die WatchDog-Funktion ein bzw. aus. Danach muss auf einem I/O-Port ein regelmäßiger Zugriff durch das Hauptprogramm erfolgen. Unterbleibt der Zugriff innerhalb des programmierten Zeitfensters (bsp. Softwarefehler), schaltet das Relais den RESET-Vorgang ein, damit der Rechner automatisch neu starten kann. Dabei schaltet das Relais mittels Ruhestrom als Öffner (Fail-Safe). Der anschließende RESET-Vorgang setzt die Watch-Dog-Funktion wieder zurück in eine neutrale Ausgangsposition (Rückfallebene), bis erneut eine gewollte Freigabe durch das Programm erfolgt. Der Relaiskontakt kann zudem für eine Weiterschaltung (bsp. als Folgerelais, Alarmkontakt, Meldelinie..) genutzt werden. Die RESET-Leitung (zur CPU) kann zusätzlich durch einen Jumper je nach Bedarf unterbrochen werden, so dass der Relaisausgang für andere Zwecke nutzbar ist.

Relais auf RL8-Karte programmieren

Alle Relais lassen sich auf dieser Erweiterungskarte direkt byte-weise ansteuern, so dass ein OUT-Befehl auf der zugehörigen Kartenadresse die Kontaktstellung entsprechend abändert.

Geben Sie das nachfolgende BASIC-Programm ein und starten Sie mit **RUN**

```
100 REM RELAIS-TEST mit XDELAY (Lauflicht)
110 FOR I = 0 TO 7    : REM Relais 1..8 in Schleife testen
120 POW2 = 2 ^ I
130 :
140 OUT &B0,INT(POW2) : REM Relais gesetzt
150 PRINT „SET RELAIS Nr.:";I+1
160 :
170 XDELAY(500)   : REM 100ms Verzoegern (bei 20MHz CPU)
180 B = INP(&B0)  : REM Kontakt-Zustand zuruecklesen
190 :
200 IF INT(POW2) <> B THEN PRINT „Fehler bei Relais: „;B
210 NEXT I
220 XDELAY(1000)
230 OUT &B0,0     : REM Alle Relais AUS
```

Hinweise zur Hardware:
Die 8-fach Relais-Baugruppe verfügt über 2 Amp. Doppelkontakte. Pro Relais ist ein UM-Kontakt über einen 25 pol. Sub-D Stecker an der Frontplatte zugängig. Der zweite Relais-Kontakt wird der Schaltung, zur Erhöhung der Eigensicherheit, wieder rückgeführt, sodass der Schließer-Zustand, per Software, Bit-weise auf der gleichen I/O-Adresse von jedem Relais wieder zurückgelesen werden kann. Zur zusätzlichen Kontolle dient eine I/O-LED, rechts neben der Power-LED, die bei jedem OUT- Befehl eingeschaltet, sowie bei jedem IN-Befehl wieder ausgeschaltet wird. Damit lassen sich I/O-Zugriffe auf das Relais-Register optisch mitverfolgen. Die Adresslage der Karte ist variabel einstellbar. Dazu wird ein Byte aus 255 möglichen I/O-Adressen für den Zugriff mit DIP-Schalter (DIP-SW1) ausgewählt.

RESET, WatchDog, Rückfallebene:
Ein DIP-Schalter (DIP-SW2) entscheidet mit Schalter Nr.1, ob der System-RESET die Relais-Ausgabe löschen soll, oder der aktuelle Registerinhalt unabhängig vom RESET beibehalten werden soll. Dies kann bei manchen Anwendungen sehr wichtig sein, die nicht durch einen RESET verändert, oder unterbrochen werden dürfen. Die gleiche Möglichkeit besteht ebenfalls, bei einem Time-Out vom WDC (WatchDog) den 8-Bit Registerinhalt zu löschen und damit die Relais in eine ursprünglich stromlose Ausgangsposition zu versetzen. Da der Rechner, selbst nach einem kompletten RESET, den Relais-Zustand der Karte wieder einlesen kann, sind sehr sichere Applikationen realisierbar die sonst nur teuren High-End-Systemen vorbehalten waren.

Optokoppler auf OI8-Karte abfragen

Alle Optokoppler lassen sich auf dieser Erweiterungskarte direkt byte-weise abfragen, so dass ein INP-Befehl auf der zugehörigen Kartenadresse das Eingangs-Register lesen kann.

Geben Sie das nachfolgende BASIC-Programm ein und starten Sie mit **RUN**

```
110 O = INP(&B8)      : REM Register-Eingang lesen
120 :
130 PRINT „OPTO-INPUT HEX = „; HEX$(O)
140 :
150 FOR I = 8 TO 0 STEP -1
160 POW2 = 2 ^ I      : REM Bit (Opto-Kanal) maskieren
165 :
170 IF INT(O AND POW2)=INT(POW2) THEN Bin$=Bin$+"1" ELSE
Bin$=Bin$+"0"
175 :
180 NEXT I
190 PRINT „OPTO-INPUT BIN = „; Bin$
```

Hinweise zur Hardware:
Alle Eingänge werden wie bei der RL8 komfortabel über LEDs angezeigt. Dazu werden die Signale im CPLD intern zwischengespeichert und anschließend erst zur LED-Anzeige gebracht, damit eine sichere, optische Gegenkontolle der erfassten Signalzustände möglich ist. Ein interner Vergleicher prüft alle eingehenden High- Pegel mit der Anzeige auf Übereinstimmung. Bei Zustandsänderung blinkt die zugehörige Kanal-LED so lange, bis der Rechner mit einem Lesebefehl den Zwischenspeicher übernommen hat. Der Lesevorgang wird durch eine weitere LED angezeigt. Somit ist eine zusätzliche Kontolle zur Übergabe des Registers an die Rechner- CPU gegeben. Der hohe Aufwand an Kontollmöglichkeiten bietet dem Anwender erhebliche Vorteile bei der Programmierung, da jedes einzelne Bit, ob high oder low, während der Erfassung mitverfolgt werden kann.

Analoge Messwerte mit AD1 erfassen

Mit der AD1-Karte lassen sich analoge Spannungen zwischen +/- 10 Volt oder kleineren Messbereichen komfortabel erfassen. Zur Messung sind nur zwei Befehle nötig:

XSETAD1(Port),Kanal ... stellt den analogen Messkanal ein
D = XGETAD1(Port) ... gibt das Messergebnis als 16 Bit-Wert an eine Variable zurück

Geben Sie das nachfolgende BASIC-Programm ein und starten Sie mit **RUN**

```
100 REM Messwert auf Kanal 1 von A/D-Karte einlesen
110 REM Basisadresse 90 Hex, Messbereich +/- 10 Volt
120 :
130 XSETAD1(&91),1    : REM A/D Messkanal einstellen
140 :
150 D = XGETAD1(&90)  : REM Wandlung 16BIT Digits nach D
160 :
170 V = (D * 0.0003051804379) - 10.000 : REM Messwert
180 PRINT „Volt = „;  : PRINT USING"##.####"; V
```

Hinweise zur Hardware:
Die AD1-Messkarte dient zur Erfassung analoger Messwerte mit 16-Bit Auflösung. Die A/D-Karte wird in mehreren Genauigkeitsklassen und Geschwindigkeiten angeboten. Je nach Bestückung kann die Messkarte Spannung oder Strom (0...20mA) messen. Dazu sind entsprechende Strom-Shunts vorgesehen, welche auch als Spannungsteiler beschaltet werden können. Als high-speed, low-power 16-bit A/D converter dient ein Wandler vom Typ AD976 (Hersteller: Analog-Device®).

Diese analoge Messkarte benötigt für die gesamte Programmierung nur zwei Bytes im I/O-Adressraum. Um hohe Abtastraten zu erzielen, wurde das Advanced-BASIC mit entsprechenden X-Befehlen erweitert. Die R/C (read-convert) Steuerung, EOC (end-of-conversion) Abfrage, sowie low- und high-byte Bildung werden bereits in Maschinensprache intern vorverarbeitet, so dass ein fertiger 16 Bit Wert an eine Variable übergeben werden kann.

Speziell für den Remote-Betrieb gibt es noch eine kürzere Variante als BASIC-Befehl:

```
100 REM Messwert auf Kanal 5 von A/D-Karte einlesen
110 K = 5
140 D = XRADC(K*256+(&90)) : REM direkte Wandlung nach D
150 :
160 V = (D * 0.0003051804379) - 10.000 : REM Messwert
170 PRINT „Volt = „;  : PRINT USING"##.####"; V
```

Analoge Spannung mit DA1 erzeugen

Mit der DA1-Karte lassen sich 4 analoge Spannungen zwischen +/- 10 Volt oder kleineren Bereichen komfortabel erzeugen. Zur Spannungsausgabe sind nur wenige X-Befehle nötig:

XDAC(Word),Adresse..... setzt den digitalen Spannungswert 0..4095 auf die DAC-Adresse
XDASET(Kartenadresse)... übergibt die DAC-Werte in den Latch und gibt Ausgang frei

Geben Sie das nachfolgende BASIC-Programm ein und starten Sie mit **RUN**

```
100 REM Bipolar-Mode alle DAC-Kanäle 0 Volt:
120 XDAC(&800),&98    : REM setze Wert auf Kanal 1
130 XDASET(&98)       : REM DAC Latch & Output Freigabe
140 XDAC(&800),&9A    : REM setze Wert auf Kanal 2
150 XDASET(&98)       : REM DAC Latch & Output Freigabe
160 XDAC(&800),&9C    : REM setze Wert auf Kanal 3
170 XDASET(&98)       : REM DAC Latch & Output Freigabe
180 XDAC(&800),&9E    : REM setze Wert auf Kanal 4
190 XDASET(&98)       : REM DAC Latch & Output Freigabe

Beispiel 2:
100 REM Kanal 1 und Kanal 2 gleichzeitig ausgeben:
110 XDAC(&0A00),&98   : REM setze +2,5 Volt auf Kanal 1
120 XDAC(&0FFF),&9A   : REM setze +10 Volt auf Kanal 2
130 XDASET(&98)       : REM DAC Latch & Output Freigabe
```

Hinweise zur Hardware:

Die DA1 Karte bietet je nach Bestückung wahlweise zwei oder vier D/A-Kanäle mit je 12 Bit Auflösung an. Je nach Referenz-Chip (REF01 / REF02) und Jumper-Einstellung können Ausgangsspannungen von +/- 10 Volt, 0...10 Volt, oder +/- 5 Volt, 0...5 Volt erzeugt werden. Die Wandler arbeiten intern mit Latch-Zwischenspeicher, so dass alle Kanäle auch zeitgleich übergeben werden können. (siehe Beispiel 2)

Die Grundeinstellung uni. / bipolar erfolgt für jeden Wandler-Baustein getrennt. Da bei dieser Karte Dual-ICs verwendet werden (zwei Wandler in einem DIP-Gehäuse), gilt die Einstellung immer paarweise. Die I/O-Basisadresse ist einstellbar. Dazu wird die Adresslage für den Port-Zugriff mit DIP-Schalter DIP-SW1 ausgewählt.

Insgesamt benötigt die Karte 8 Register. Zwei Lese-Register dienen zur Übergabe der neuen DAC-Werte, sowie einem schaltbaren Ausgang, welcher während der RESET-Phase abgeschaltet bleibt (disable), um keine ungewollten Spannungszustände zu erzeugen. Alle I/O-Kartenzugriffe werden durch eine LED zusätzlich angezeigt. Werkseitig verfügt das Interface über eine 9pol. D-Sub Buchse. Weitere Angaben können Sie dem Datenblatt entnehmen.

Zählerstände (digitale counts) mit CNT-Karte erfassen

Mit der CNT-Karte lassen sich digitale TTL-Impulse komfortabel zählen, oder Positionen an inkrementalen Gebern (bsp. Winkelgeber, Längenmeßstab, Messtaster...) erfassen.
Zur Messung sind nur wenige Input-Befehle nötig.

Geben Sie das nachfolgende BASIC-Programm ein und starten Sie mit **RUN**

```
100 REM Zaehler 1 auf Basisadresse A0 auslesen
110 A = INP(&A0) : REM Low, Read speichert alle Register
120 B = INP(&A1) : REM Medium
130 C = INP(&A2) : REM High
140 D = INP(&A3) : REM nur Vorzeichen-Bit in Data Bit 0
150 :
160 X = A + (B*256) + (C*65535) : REM Zählerstand berechnen
170 PRINT „Alle Register: „
180 PRINT A;B;C;D
190 PRINT „Counter 1 = „;X
```

Hinweise zur Hardware:
Die CNT-Messkarte dient der Erfassung von dig. Impulsen. Beide Zähler können Pulsraten von 0 bis 10 MHz in Echtzeit verarbeiten. Die Impulserfassung erfolgt intern über schnelle Quadraturencoder. Damit keine Impulse ausbleiben werden 3x 8-Bit Latch plus ein Sign-Bit, in einem vierten Register, immer zwischengespeichert. Die Zähler arbeiten vollkommen unabhängig, können jedoch über einen Triggereingang ebenso synchron betrieben werden. Dadurch können beliebig viele Karten parallel betrieben werden, um beispielsweise die Ist-Position bei Längenmessungen in Mehrachsensystemen ohne Zeitjitter zu erfassen. Die anschließende Verarbeitung wird mit absoluten Zählerständen vorgenommen.

Auf Grund der int. 20 MHz Zeitbasis ist eine Impulserfassung (up/down-count) bis 10 MHz möglich. Im 4-fach Inkremental-Mode beträgt die Abtrastrate 5 MHz. Diese Einstellung wird beispielsweise für digitale Drehgeber und Encoder (mit diff. TTL/CMOS-Signal) oder inkrementale Längenmeßstäbe an Mehrachsen-Systeme und CNC-Maschinen benötigt. Ein Zähler-Reset kann synchron oder asynchron erfolgen, sowie über die internen Counter als Zählerstand vorprogrammiert werden. Je Zähler werden vier I/O-Register verwendet. Die Zwischenspeicherung erfolgt bei Lesen des Low-Bytes. Während der Datenübertragung der Registerinhalte, zählt der Counter im Hintergrund weiter, damit kein Verlust an der augenblicklichen Position des Gebers entsteht. Über einen DIP-Schalter werden feste Parameter zu jedem Counter separat eingestellt. Die I/O-Basisadresse ist einstellbar. Dazu wird die Adresslage für den Port-Zugriff mit DIP-Schalter DIP-SW1 ausgewählt.

Optokoppler auf E32-Karte abfragen

Alle Optokoppler-Eingänge lassen sich auf dieser Erweiterungskarte direkt byte-weise erfassen, so dass INP-Befehle auf der zugehörigen Kartenadresse das jeweilige Eingangs-Register einlesen kann.

Geben Sie das nachfolgende BASIC-Programm ein und starten Sie mit **RUN**

```
100 REM Opto-Kanal 1..32 einlesen (vier 8-Bit Register)
120 CLS : LOCATE 2,4
140 :
150 A = INP(&B8)
160 B = INP(&B9)
170 C = INP(&BA)
180 D = INP(&BB)
190 :
200 PRINT „1. BYTE INP = „;A
210 PRINT „2. BYTE INP = „;B
220 PRINT „3. BYTE INP = „;C
230 PRINT „4. BYTE INP = „;D
```

Hinweis zur Hardware:
Da die 32-bit Karte insgesamt 4 Register zu je 8-bit verwendet, erfolgt die Abfrage auf vier unterschiedlichen I/O-Adressen.

Optokoppler-Ausgänge auf A32-Karte programmieren

Alle Optokoppler-Ausgänge lassen sich auf dieser Erweiterungskarte direkt byte-weise erzeugen, so dass OUT-Befehle auf der zugehörigen Kartenadresse das jeweilige Ausgabe-Register überschreiben kann. Das byte wird hierzu invertiert übergeben, damit bei einem Hardware-RESET alle Ausgänge (lachtch-up bedingt) hochohmig (=high-Z) sind.

Geben Sie das nachfolgende BASIC-Programm ein und starten Sie mit **RUN**

```
100 REM Ausgabe Opto-Kanal (1. Register)
110 X = 128           : REM setzte Bit 7 (0x80h)
120 OUT (&BC), 255-X  : REM Output
```

Hinweis2 zur Hardware:
Da die 32-bit Karte insgesamt 4 Register zu je 8-bit verwendet, erfolgt die Ausgabe auf vier unterschiedlichen I/O-Adressen. Jedes 8-bit-Register dient gleichzeitig als Zwischenspeicher, so dass erst durch „überschreiben“ des Bytes ein neuer Wert für das jeweilige Register auf den Optokoppler-Ausgang übertragen wird. Es stehen drei verschiedene Kartenvarianten zur Wahl: P- oder N-schaltend, sowie Photo- MOS Relais bis 350 mA pro Kanal. Es sind auch Mischbestückungen mit P, N und Photo-MOS möglich. Durch Verwendung einer invertierten Logik mit Freigabe-FF (erster Kartenzugriff), werden keine undefinierten Pegelzustände während der Power-Up- und RESET-Phase erzeugt. Ein WatchDog-Reset versetzt alle Ausgänge in eine neutrale Ausgangssituation, wie man sie nach dem Einschalten des Rechners vorfindet (alle Ausgänge = OFF bzw. hochohmig, high-Z).

Die „AUTOBAS“ Startdatei im Hauptverzeichnis (Root)

Eine Besonderheit kommt der ersten Datei auf dem Datenträger (Drive0) zu, denn sie kann, ähnlich einer .bat Batch-Datei, ein BASIC-Programm direkt nach einem RESET ausführen, ohne weitere Benutzereingaben zu tätigen. Dazu muss das Programm zwingend unter dem Namen **AUTOBAS** in **Großbuchstaben** als Datei **01** abgespeichert sein.

Achtung:
SEPIA prüft während des bootens, ob die IDE-Karte nebst Speichermedium (IDE0) und diese Datei vorhanden sind. Falls ja, wird das Programm unmittelbar vom System geladen und im BASIC-Interpreter gestartet. Das AUTOBAS BASIC-Programm ermöglicht somit ein Wiederanlauf von individuellen Programmen, beispielsweise nach einem Stromausfall.

Mit LOADGO können ebenso weitere Programme aus AUTOBAS heraus gestartet werden, sowie COM-Schnittstellen und/oder I/O-Ports neu initialisiert werden. Bei der Nutzung von AUTOBAS gilt die Verwendung des WDC (WatchDog) zu berücksichtigen. Auch hier wird bei einem RESET die AUTOBAS-Datei automatisch geladen und gestartet. Da in AUTOBAS der gesamte Befehlsvorrat des BASIC-Interpreters verwendet werden kann, sind ebenso Protokollausgaben, z.B. auf einem Drucker möglich, welche dann Zeit, Datum, sowie andere Angaben zu diesem Ereignis schriftlich festhalten. Wird keine AUTOBAS Datei im Root-Verzeichnis abgespeichert, startet SEPIA das ursprüngliche Menue zur weiteren Abfrage.

Bildschirm-Befehle bei Verwendung der NET-Karte (mit VGA-Ausgang)

Auf den ersten Seiten wurden VGA-Farb-Befehle nur kurz vorgestellt. Die Handhabung und Wirkungsweise dieser Advanced-BASIC-Befehle sollen nun näher erläutert werden. Bei Textausgaben auf ext. Terminals oder PC-Terminal-Software, kann die Farbausgabe je nach Gerät unterschiedlich ausfallen. Intern erfolgen die verwendeten Steuerkommandos bzw. ESC-Sequenzen in Anlehnung an ein VT100 Terminal.

Beispiel: **`XTCOLOR(0,9)`**
Die gesamte Textausgabe erfolgt mit schwarzen Zeichen auf hell-blauem Hintergrund.

Beispiel: **`XTCOLOR(8,0)`**
Die gesamte Textausgabe erfolgt mit amber Zeichen auf schwarzem Hintergrund.

Beispiel: **`XRSTCOL`**
Der Befehl enthält keine Zusatzparameter. Die anschließende Textausgabe erfolgt in der boot-Einstellung. (Weisser Text auf dunkelblauen Hintergrund)

Beispiel: **`XINVERT`**
Der Befehl enthält keine Zusatzparameter. Die anschließende Textausgabe (ab dem Cursor) erfolgt farblich invertiert, bis die Invertierung wieder aufgehoben wird.

```
; Farbe Vordergrund
; 0 = Schwarz
; 1 = Rot
; 2 = Grün
; 3 = Gelb
; 4 = Blau
; 5 = Rosa
; 6 = Cyan
; 7 = Grau
; 8 = Amber
; 9 = Hell-Lila

; Farbe Hintergrund
; 0 = Schwarz
; 1 = Rot
; 2 = Dunkel-Grün
; 3 = Khaki-Braun
; 4 = Dunkel-Blau
; 5 = Mangenta
; 6 = Dunkel-Cyan
; 7 = Hell-Grau
; 8 = Hell-Blau
; 9 = Orange
```

HDD-DISK-Befehle bei Verwendung der IDE-Karte (1)

Auf den ersten Seiten wurden IDE (Disk)-Befehle nur sehr kurz vorgestellt. Die Handhabung und Wirkungsweise dieser Advanced-BASIC-Befehle sollen nun näher erläutert werden.

Beispiel: **`DRIVE 0`**

Durch Eingabe des Befehls wird der interne Laufwerkzeiger auf Laufwerk 0 (Master) voreingestellt. Alle nachfolgenden Operationen wie DIR, SAVE, LOAD, ERASE... beziehen sich fortan auf dieses Laufwerk. In der Regel handelt es sich dabei um die CF-Karte an IDE0.

Beispiel: **`DRIVE 1`**

Durch Eingabe des Befehls wird der interne Laufwerkzeiger auf Laufwerk 1 (Slave) voreingestellt. Alle nachfolgenden Operationen wie DIR, SAVE, LOAD, ERASE... beziehen sich fortan auf dieses Laufwerk. In der Regel handelt es sich dabei um die SD-Karte an IDE1 oder ein SSD bzw. Flash-Drive.

Beispiel: **`DRIVE S`**

Durch Eingabe des Befehls werden alle aktuellen (vorhandene) Laufwerke angezeigt.

Beispiel: **`DRIVE P`**

Durch Eingabe des Befehls wird das aktuelle Laufwerk geparkt. (Köpfe auf Spur 0)

Beispiel: **`DRIVE T`**

Durch Eingabe des Befehls werden Register-, Fehler-, und Statusmeldungen zum aktuellen Laufwerk angezeigt.

Beispiel: **`RESET`**

Durch Eingabe des Befehls werden alle HDD-Register rückgesetzt bzw. grundinitialisiert.

Beispiel: **`DIR`**

Durch Eingabe des Befehls wird das Disk-Inhaltsverzeichnis angezeigt. Es kann je Laufwerk und Verzeichnisebene bis zu 255 Dateien enthalten. Nach 16 Einträgen pausiert die DIR-Ausgabe. Sie kann durch betätigen der Leertaste fortgeführt oder durch Eingabe J/N auch abgebrochen werden. Die Einträge werden nach Dateinummer 0..FF sortiert ausgegeben.

Beispiel: **`ERASE`**

Durch Eingabe des Befehls wird nach der Dateinummer gefragt, welche anschließend nach J/N-Abfrage/Bestätigung gelöscht werden soll. Die nun freie Dateinummer kann anschließend mit einem neuen BASIC-Program überschrieben werden, oder frei bleiben.

HDD-DISK-Befehle bei Verwendung der IDE-Karte (2)

Fortsetzung:

Beispiel: **LOAD**

Durch Eingabe des Befehls wird zunächst nach einer Dateinummer (0..FF) gefragt. Das Programm mit der angegebenen Dateinummer wird nach Bestätigung in den RAM-Seicher geladen. Ein erfolgreicher „Load“ wird nach Übertragung durch einen kurzen Quittungston signalisiert. Anschließend kann man mit „List“ das Programm sichten, ggf. drucken, nachberabeiten oder mit „RUN“ starten.

Beispiel: **RENAME**

Durch Eingabe des Befehls wird zunächst die Dateinummer eingegeben, welche einen neuen Dateinamen erhalten soll. Im zweiten Abschnitt wird der Dateiname überschrieben und abgespeichert. Alle Eingaben sind dialog-geführt. Der Dateiname kann alle ASCII-Zeichen (32...127 dez.) enthalten und ist bis zu 32 Zeichen lang.

Beispiel: **XROOT(x)**

Durch Eingabe des Befehls wird der Zeiger für die Verzeichnisebene vorbelegt mit der anschließend alle Dateioperationen durchgeführt werden. Abhängig vom Speichermedium (HDD Kapazität) können bis zu 255 Verzeichnisse verwaltet werden. Die Ebenen-Nummer entspricht dem Cylinder-High aus der ATA CHS-Tabelle. Besitzt ein Medium lediglich 15 Cylinder-High-Einträge (siehe CHS-Tabelle), können folglich nur 15 Verzeichnisse genutzt werden. Bei voller Unterstützung mit 255 low und 255 high Cylinder können je Medium bis zu 65280 Dateien verwaltet werden. Es können jedoch nur Dateinummern von 1..255 vergeben werden, da das erste Root-Verzeichnis (0) wichtige Daten für das Directory enthält. Somit ist dieser Cylinder für den Nutzer i.d.R. tabu. Eine Sicherheitsabfrage verhindert zusätzlich den unerlaubten Zugriff auf diesen Bereich.

Beispiel: **LOADGO(x)**

Durch Eingabe des Befehls kann über die Dateinummer (x) ein anderes BASIC-Programm nachgeladen und sofort gestartet werden. Somit lassen sich beispielsweise auch mehrere Programme untereineander verschachteln. Die Dateinummer erfolgt in sedezimaler Schreibweise (&00 ... &FF).

HDD-DISK-Befehle bei Verwendung der IDE-Karte (3)

Fortsetzung:

Beispiel für eine Directory-Ausgabe auf dem Bildschirm. Die DIR-Ausgabe pausiert nach 16 Einträgen. Durch Tastendruck (bsp. „J“ oder Leertaste) wird die Ausgabe weiter gescrollt, oder mit „N“ abgebrochen.

```
Drive: 0  akt. Verzeichnis: 000
 Datei: Nr., Name, Zeit, Datum, Groesse

 01 AUTO........................ 23:32:52  02.06.2012  3628 Bytes
 02 TEST ....................... 11:22:59  25.12.2011  0072 Bytes
 03 CRT-TEST.................... 12:14:29  25.12.2011  0103 Bytes
 04 Tonleiter.BAS............... 20:40:30  27.12.2011  0175 Bytes
 05 TONLEITER.BAS............... 20:46:03  27.12.2011  0175 Bytes
 06 AUTOBAS-BEEP................ 20:54:55  24.05.2012  0072 Bytes
 07 AUTOBAS..................... 23:32:43  02.06.2012  3628 Bytes
 08 16BIT-R/W I/O............... 21:12:25  03.06.2012  0194 Bytes
 09 STEP........................ 13:33:58  03.06.2012  0071 Bytes
 10 EXIT........................ 13:53:18  03.06.2012  0073 Bytes
 11 XPOKE-XPEEK................. 21:42:19  03.06.2012  0283 Bytes
 12 FIX$........................ 20:54:44  03.06.2012  0073 Bytes
 13 XSCALL...................... 21:41:27  03.06.2012  0217 Bytes
 14 OUTBYTE..................... 14:58:52  04.06.2012  0052 Bytes
 FF TEST AUTOBAS................ 12:26:51  24.12.2011  0072 Bytes


 weiter ? Taste!

 BASIC verfuegbar



 Fertig
```

Kurze Erläuterung
Die Anzeige „Drive: 0“ bedeutet, das hier das Laufwerk0 an IDE-0 ausgewählt wurde.
Die Anzeige „Verzeichnisebene: 000“ bedeutet, das die Directory-Ausgaben dem ersten Verzeichnis (root) zuzuordnen sind. Je nach Medium (siehe CHS-Tabelle: Cylinder-High) können bis zu 255 Verzeichnisebenen angelegt (und sogar einzeln formatiert) werden.
Die Formatierung erfolgt über das Debugger-Menue mit „S“ zum SEPIA IDE-Menue mit dem Tastenkürzel „F“. Jede Datei kann zudem beliebig oft mit neuen Programmen überspielt, oder einzeln gelöscht werden.

16 Bit I/O-Adressen direkt ansteuern

Mit SEPIAs Advanced-BASIC können 16-Bit Adressen auf dem Adressbus generiert werden. Dazu wird das BC-Register des Prozessors verwendet, indem die Adressen A8...A15 auf „B“ und A0...A7 auf das „C“-Register eingeschrieben werden. Werden anschließend INP- oder OUT-Befehle verwendet, liegt die Adresse mit dem IORQ-Signal an, sodass eine Dekoderschaltung diese als 16-Bit-Adresse interpretieren kann. Zur Unterscheidung wurden hierzu weitere Advanced-Befehle integriert, die diese Zugriffe behandelt:

```
XOUTWD(word),byte       ; auf 16Bit I/O-Adresse ein byte schreiben
byte = XINPWD(word)     ; auf 16Bit I/O-Adresse ein byte lesen
```

Geben Sie das nachfolgende BASIC-Programm ein und starten Sie mit **RUN**

```
100 REM 16 Bit I/O Test
120 CLS
140 LOCATE2,2
160 XOUTWD(&B0),55          :REM Schreibe 55 nach B0h (Relaiskarte)
180 XDELAY(1000)
200 X = XINPWD(&B0)         :REM Lese den Port zurück, nach X
220 PRINT "Register = ";X
240 :
260 XDELAY(2000)
280 XOUTWD(&FFB0),1         :REM Überschneidung mit B0h !!!
300 XDELAY(2000)
320 Y = XINPWD(&FFB0)       :REM Lese den Port zurück, nach Y
340 PRINT "Register = ";Y
```

ACHTUNG:
Verwenden Sie diese Befehle sehr vorsichtig. Erweiterungskarten besitzen i.d.R. keine volle Adressdekodierung, so dass es bei falscher Programmierung zu Überschneidungen mit Datenkollision kommen kann. Die Anwendung dieser X-Befehle ist erst für Adressen ab E0h vorgesehen, da hier im Normalfall keine Erweiterungskarten mehr installiert sind.

Zu beachten gilt weiterhin, dass der SEPIA-Systembus im full-speed-mode mit 20 MHz betrieben wird. Langsame Port-I/O Bausteine können in diesem Fall nur mittels Wait-State-Zyklen an diese hohe Bus-Geschwindigkeit manuell angepasst werden. Dazu bietet der SEPIA-Bus-Contoller (ispLSI1032) mittels DIP-Schalter mehrere Einstellmöglichkeiten zur IORQ-Verzögerung > siehe techn. Doku: SEPIA-Bus.

Sonderfunktionen: Debugger < > BASIC

Es besteht die Möglichkeit zwischen Debugger- und BASIC-Mode nach belieben hin-und-her zu schalten. Benötigt man beispielsweise den Debugger zur Einsicht von aktuellen RAM Speicherdaten, so kann der BASIC-Mode durch den Befehl „**Bye**“ verlassen werden, ohne das flüchtige Daten oder Zeiger zerstört werden. Im umgekehrten Fall kann der Debugger-Betrieb mit „W“ wie Warmstart verlassen werden, um sich wieder dem BASIC-Programm zu widmen. Wird der Debugger mit „K“ verlassen, werden alle Zeiger und Teile von Variablen wie Initialisierungstabellen neu gesetzt = überschrieben wie RESET-Zustand.

Die Besonderheit dieser Vorgehensweise erlaubt dem Nutzer die optionale Einbringung von Z80 Maschinencode in den BASIC-Speicher**. Durch CALL(Adresse) wird dieser implizierte Code unter BASIC ansprechbar. Bitte nicht vergessen: Sprungadresse am Schluss mit RET beenden. Wird dieses BASIC-Programm anschließend auf einen Datenträger abgespeichert, so wird auch der implizierte Code mit übernommen. Dies hat den Vorteil, das man BASIC und Z80 Maschinensprache miteinander beliebig mischen, speichern und zurückladen kann.

**
In diesem Fall ist es ratsam, sich vorher einen Teilbereich des BASIC-Speichers für die Größe seines Codes zu reservieren, damit keine Überschneidungen von Speicherbereichen stattfinden können. Die Reservierung kann gleich zu Beginn (beim Start) des BASIC-Interpreters erfolgen. Die Return-Taste nimmt keine Speicherreservierung vor und weist den gesamten RAM-Speicher dem Interpreter zu (i.d.R. = 31821 Byte).

Rechenbeispiel:
Das SRAM besitzt urspünglich 32768 byte Speicherzellen. Davon werden i.d.R. 947 Byte für interne Zwecke wie Zwischenparameter und Variablen: Stack-Buffer, Drive-Tabelle, String-Buffer, Uhrzeit- und Sektoren-Buffer verwendet. Um eine genauere Speicherbelegung der einzelnen Bufferbereiche zu ermitteln, kann im Debugger die * Taste weiter Auskunft über die gängigen Labels geben. Darin enthalten sind ebenso BASIC-Anfang und BASIC-Ende-Adresse, sowie aktuelle Stapeladresse und freier RAM-Bereich.

Debugger - Menue

Der Debugger umfasst folgende Befehle:

```
DEBUGGER V2.41

 22:04:59  03.06.2012  Sonntag

 S = IDE Editor
 L = RAM loeschen & Neustart
 D = RAM anzeigen ab Adresse
 I = Befehle interpretieren ab Adresse
 F = RAM fuellen, von, bis, Wert
 A = ASCII-Text eingeben (max. 256)
 G = Starten ab Adresse, Breakpoint
 T = Einzelschritt ab Adresse
 C = Einzelschritt (ohne CALL)
 E = RAM aendern ab Adresse
 Y = Wert1, Wert3, ...suchen
 R = DEBUG-Register, Wert
 X = DEBUG-Register loeschen
 P = Port I/O: I77 oder O77,35
 U = Uhrzeit/Datum stellen
 M = CPU-Temperatur
 H = Protokoll-Drucker: EIN/AUS
 N = XSEG Segmentregister
 * = Label Adressen
 Z = Zurueck, ? = Syntaxhilfe
 <Esc> = Neustart, K = Kaltstart, W = Warmstart

 >
```

Das SEPIA-IDE-Menu kann direkt mit der Taste „S“ angewählt werden. Hier trifft der Anwender weitere Möglichkeiten zur Diagnose, oder kann beispielsweise einzelne Sektoren überschreiben bzw. formatieren (siehe nächste Seite).

Der Debugger beinhaltet einen eigenen Hilfe-Text, welcher mit der Taste „?“ aktiviert werden kann. Alle Monitor-Funktionen sind weitgehend selbsterklärend gestaltet, um einen einfachen und schnellen Einstieg in die maschinennahe Programmierung zu erhalten. Die Schreibweise erfolgt Hexadezimal, die Ausgabe (Dump) kann mit Z80-Labels (quasi wie ein Disassembler) aus dem RAM-Speicher zurück interpretiert werden.

Wer einen separaten Crossassembler für den PC besitzt, kann Z80 Maschinenprogramme extern linken und den Binärcode beipielsweise in ein EPROM übertragen. Da die Speicher- und Port-Schnittstellen alle in der MAPALL.ASM dokumentiert sind, kann der Programmierer hier sein eigenes System, unabhängig von SEPIA, etablieren.

SEPIA IDE - Menue

Das IDE-Menue umfasst folgende Befehle:

```
IDE Editor

 V = DRIVE Master(0) Slave(1)
 I = Identify
 M = MBR lesen
 E = Error/Status zeigen
 X = Drive INIT
 Y = IDE-RESET
 Q = 1 Sektor lesen, ablegen, anzeigen
 R = Sektoren lesen (max. 3Eh)
 T = Sektoren schreiben
 O = Sektoren ueberschreiben
 D = Directory
 J = Verzeichnis Nr.?
 F = Formatieren J/N
 N = Dateiname aendern
 L = Datei laden
 P = Programm starten
 S = Datei speichern
 H = Hardcopy EIN/AUS <CR>
 Z = Zurueck -> Debugger

 <Esc> = Neustart, K = Kaltstart, W = Warmstart
 >
```

Das Menue ist weitgehend selbsterklärend: Mit „V“ 0 oder 1 wird der IDE-Datenträger Master oder Slave aktiviert, mit „I“ wird die Identifikation des Datenträgers als Hex- und ASCII-Dump angezeigt, mit „M“ lässt sich der Master-Boot-Record auslesen. Mit Taste „E“ wird der IDE-Controller angewiesen, das jeweilige Medium auf Fehler zu prüfen. Mit „X“ wird die Drive-Tabelle neu übertragen, mit „Y“ wird das Medium neu gestartet. Mit Taste „Q“ kann man einen bestimmten Sektor direkt vom Medium laden (512 Byte), ins RAM ablegen und anschließend anzeigen lassen. Mit „R“ und „T“ werden Sektoren direkt gelesen/geschrieben oder mit „O“ überschrieben. Die Taste „D“ hat die gleiche Funktion wie der DIR-Befehl in BASIC. Mit „J“ kann die Verzeichnisebene (Zylinder High bis max. 255) gewechselt werden um weitere Dateien zu verwalten. Mit „F“ wird der Datenträger auf dem jeweiligen Zylinder formatiert. Somit kann man einzelne Verzeichnisse formatieren, ohne Daten in anderen Verzeichnissen zu zerstören. Mit „N“ kann ein Dateiname über die Dateinummer wieder neu vergeben werden. „L“ funktioniert wie LOAD unter BASIC und dient zum laden von gespeicherten Dateien in den SEPIA-RAM-Speicher. Mit „P“ lässt sich ein Programm direkt starten, mit „S“ wird eine Datei mit Dateinummer und Name abgespeichert. Alle Text - Ein/Ausgaben lassen sich mit „H“ auf den Drucker mit protokollieren. „Z“ beendet das Menue und kehrt zum Debugger zurück.

SEPIA IDE - Menue: Taste „I“

```
>I = Identify

 Drive: 0
 RAM-Puffer-Adresse:

       .0..1..2..3..4..5..6..7..8..9..A..B..C..D..E..F 0....5....A....F

 8170  03 32 FD EA 00 00 00 22 00 00 00 2C 04 4A 0F 20 .2.j..."...,.J..
 8180  00 00 00 10 7E 00 02 00 00 3F 00 3B 8E 00 00 00 .........?.;....
 8190  37 41 41 46 30 37 31 33 30 45 36 41 30 30 30 30 7AAF07130E6A0000
 81A0  30 35 33 38 00 02 00 02 00 04 56 65 72 36 2E 30 0538......Ver6.0
 81B0  34 20 43 46 20 43 61 72 64 20 20 20 20 20 20 20 4.CF.Card.......
 81C0  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 ................
 81D0  20 20 20 20 20 20 20 20 20 20 80 01 00 00 2B 00 ..............+.
 81E0  40 00 02 00 00 00 00 07 0F 20 00 10 00 3F 8E 00 @............?..
 81F0  00 3B 01 01 8E 00 00 3B 00 00 04 07 00 03 00 78 .;.....;.......x
 8200  00 78 00 78 00 78 40 00 00 00 00 00 00 00 00 00 .x.x.x@.........
 8210  00 00 00 00 00 00 00 00 00 00 00 00 00 00 30 00 ..............0..
 8220  70 2B 54 04 40 00 70 00 14 04 40 00 00 7F 00 1E p+T.@.p...@.....
 8230  00 1E 00 00 FF FE 00 00 00 00 00 00 00 00 00 00 ................
 8240  00 00 00 00 8E 00 00 3B 00 00 00 00 00 00 00 08 .......;........
 8250  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
 8260  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
 8270  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................


 weiter ? Taste!
```

Interessante ROM - CALLs

Es besteht die Möglichkeit, mittels CALL-Befehl, ROM-Routinen direkt aus dem BASIC heraus zu starten. In der folgenden Tabelle sind einige nützliche Sprungadressen aufgelistet, die dem Programmierer, der Erfahrung mit Maschinensprache hat, Arbeit abnehmen können.

Die Eingabe hinter dem CALL-Befehl erfolgt in Hex als 8 oder 16 Bit Adresse.
Beispiel in BASIC: `CALL &0000`

Adresse		Funktionsaufruf
0000h	-	Systemstart
0038 h	-	Interrupt-Register auslesen (ISR0 aufrufen)
0066 h	-	NMI aufrufen, erzeugt Neustart mit Kaltstart auf 0000h
0070 h	-	CPU-Temperatur auf COM1 senden
0074 h	-	Beep Ton ausgeben
0078h	-	Uhrzeit auf COM1 senden
007Bh	-	Datum auf COM1 senden
0080 h	-	Bildschirm löschen und Debugger-Menue starten
0084 h	-	IDE-Medium check Master auf IDE0
0088 h	-	BASIC Programm starten (wenn im SRAM vorhanden)
00FFh	-	CPU Halt auslösen >>> Alarm-Relais testen (rote LED auf CLK-Karte)

... weitere folgen später

Status von Konsole auf VGA-Monitor ausgeben

Um eine Statusmeldung der Konsolenverbindung ein-, oder auszublenden wurden zwei spezielle Direktbefehle in das Advanced-BASIC für die VGA-Konsole eingebunden:

- **XSTSON**

- **XSTSOFF**

Mit Eingabe des Befehls XSTSON wird die Meldung in einem reservierten Ausgabebereich des unteren Bildschirms erzeugt. Alle BASIC-Ein/Ausgaben können weiterhin gescollt werden, unabhängig der Statusmeldungen. Mit XSTSOFF wird die Meldung wieder gelöscht. Ein CLS- Befehl hat die gleiche Wirkung.

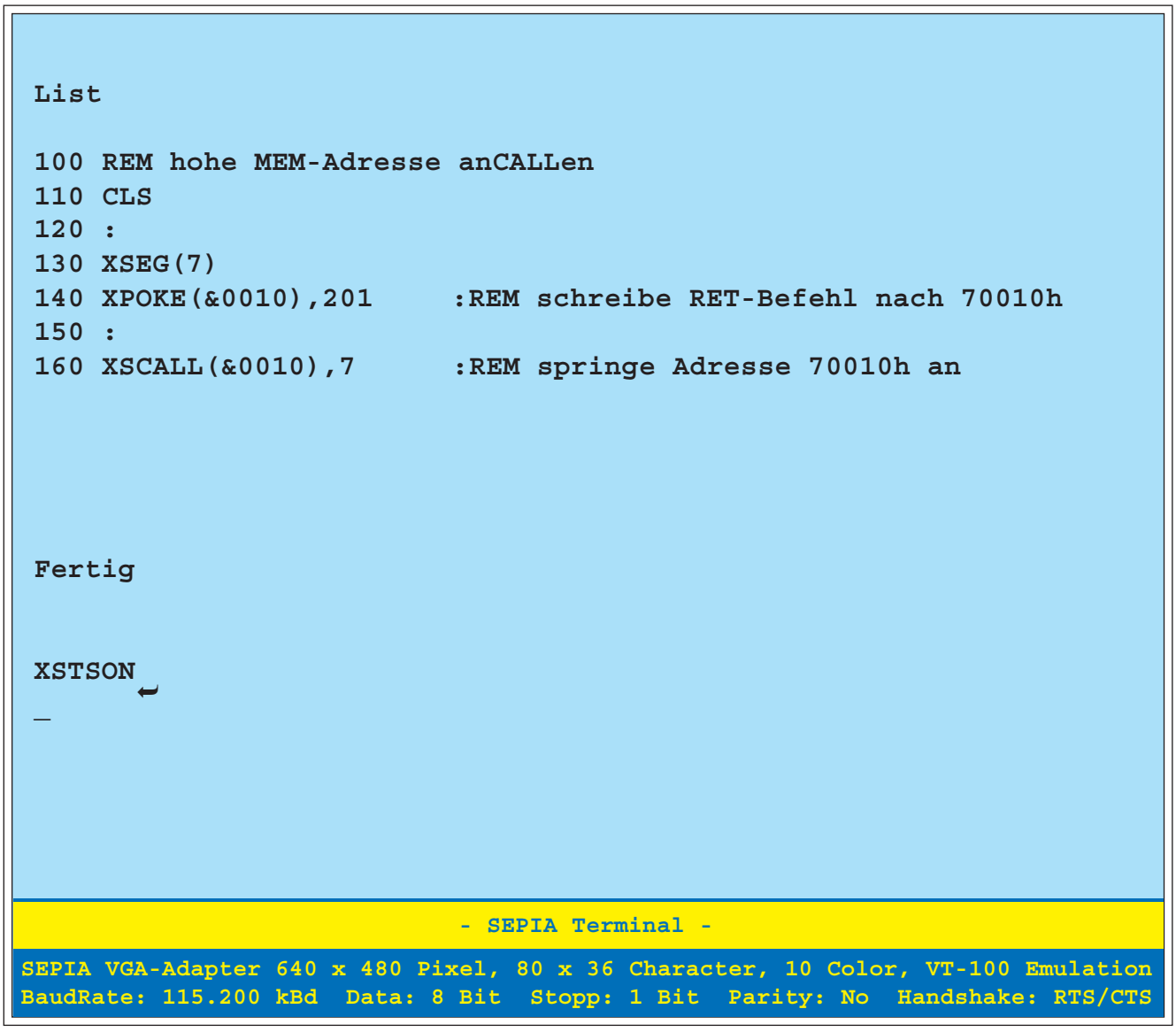
```
List

100 REM hohe MEM-Adresse anCALLen
110 CLS
120 :
130 XSEG(7)
140 XPOKE(&0010),201     :REM schreibe RET-Befehl nach 70010h
150 :
160 XSCALL(&0010),7      :REM springe Adresse 70010h an




Fertig


XSTSON↵
_
```



Grafik-Befehle bei Verwendung der NET-Karte (mit VGA-Ausgang)

Auf den ersten Seiten wurden VGA-Grafik-Befehle nur kurz vorgestellt. Die Handhabung und Wirkungsweise dieser Advanced-BASIC-Befehle sollen nun näher erläutert werden. Da der SEPIA-Rechner in erster Linie für MSR- und Automatisierungszwecke dient, sind die Grafikfunktionen nur auf wenige Befehle beschränkt. Die Auflösung beträgt beispielsweise nur 640 x 240 Pixel mit 2 von 64 Farben. Auf Textausgaben und Kreisfunktionen wurde aus Platzgründen (im EEPROM) ebenfalls verzichtet. Die Kommunikation erfolgt hierzu auf COM1. Intern werden spezielle ESC-Sequenzen für Steuerkommandos verwendet, die der Grafik-Chip mit einem eigenen BIOS interpretiert. Somit wird kein SEPIA RAM-Speicher für den Bildaufbau verwendet und der Hauptprozessor vollkommen entlastet.

Beispiel: **XGRON**
Interne ESC-Sequenz: **ESC[z1**
Schaltet in den Grafik-Mode um. Der Bildschirm wird dabei gelöscht.

Beispiel: **XGROFF**
Interne ESC-Sequenz: **ESC[z2**
Schaltet in den Text-Mode um und inititalisiert den Bildschirm neu.

Beispiel: **XGCLS**
Interne ESC-Sequenz: **ESC[z0**
Löscht den internen Grafik-Bildspeicher (wie CLS).

Beispiel: **XGRTEST**
Interne ESC-Sequenz: **ESC[t@**
Erzeugt ein statisches Testbild als mehrstufige Farbtreppe auf dem Bildschirm.

Beispiel: **XGRESCOL**
Interne ESC-Sequenz: **ESC[z3**
Setzt die org. Farbpalette wieder zurück. (weiß auf blau)

Beispiel: **XGRSTCOL V,H**
Interne ESC-Sequenz: **ESC[<V><H>]**
Setzt Pixel-Farbe für Vordergrund und Hintergrund. (zwei-byte-kompliment)

Beispiel: **XGRPSET X,Y**
Interne ESC-Sequenz: **ESC[<XMSB><XLSB><Y>}**
Setzt einen Pixel an Koordinate X, Y (Xmax. = 640, Ymax. = 240 als zwei-word-kompliment)

Beispiel: **XGRLINE X1,Y1,X2,Y2**
Interne ESC-Sequenz: **ESC[<XMSB1><XLSB1><Y1><XMSB2><XLSB2><Y2>{**
Zeichnet eine Linie von Koordinate X1, Y1 zu X2, Y2. (vier-word-kompliment)

Grafik-Befehle bei Verwendung der NET-Karte (mit VGA-Ausgang) Teil2

Geben Sie das nachfolgende BASIC-Programm ein und starten Sie mit **RUN**

```
100 REM Grafik-Test
110 XGRON                        : REM Grafik EIN
120 XGRTEST                      : REM Testbild
130 XDELAY(40000)                : REM Warten
140 XGRCLS                       : REM Grafik löschen
150 XGRSTCOL(255,2)              : REM Farbe setzen
160 XGRPSET(320,120)             : REM Pixel zeichnen
170 XGRLINE(10,100,630,100)      : REM Linie zeichnen
180 XGRLINE(20,200,630,110)      : REM Linie zeichnen
190 XDELAY(40000)                : REM Warten
200 XGROFF                       : REM Grafik AUS
210 STOP                         : REM Text-Mode
```

Technicher Hinweis:

Alle direkten Advanced-BASIC Ein/Ausgaben der Konsole erfolgen immer über COM1, welche durch div. DIP-Schalterstellungen auf die NET-Karte (VGA-Ausgang) umgelenkt werden. Die Grafik erzeugt ein 8-Kern 32-Bit Parallax®-Prozessor mit 80 MHz interner Taktfrequenz. Die gesamte Datenübertragung erfolgt dazu über eine interne, serielle COM-Verbindung auf Kanal 1 mit 115.200 kBaud als einfache Terminal-Emulation, wie zu einem abgesetzten LCD-Display. Dies ermöglicht ebenso den Einsatz von seriellen Displays.

Diese Schaltungsanordnung hat mehrere Vorteile:

1) Der Hauptprozessor wird komplett mit der Erzeugung von Text und Grafik entlastet.

2) Die Datenübertragung kann auch zu anderen COM-Ports umgelenkt werden.

3) Die Datenaufbereitung erfolgt in einem eigenen Prozess, unabhängig von CPU-Zeiten. Damit sind die Ausgaben immer gleich schnell.

4) Die Auslagerung von Text- und Grafik-Routinen spart viel Platz im EPROM.

5) Die Weiterentwicklung von VGA Hard- & Software erfolgt unabhängig vom Hauptsystem.

6) Durch Nutzung gängiger Auflösungs-Modi sind nahezu alle VGA-Monitore verwendbar.

EMail versenden mit XPort® (auf NET-Karte)

Damit SEPIA eine EMail versenden kann, wurde der Trigger-Eingang des XPort-Konverters mit dem CPLD über ein FlipFlop verknüpft. Mittels BASIC-Befehl wird darauf hin ein negativer Impuls an Trigger-Eingang (CP2) erzeugt. Werden die erforderlichen EMail-Parameter für den XPort über den HTML Browser programmiert (Grundkonfiguration), kann eine Nachricht (bsp. ALARM 123) über das Netzwerk abgesetzt werden. Zur Kontrolle wird ein „@“ Zeichen auf den jeweiligen COM-Port mit ausgegeben.

Geben Sie das nachfolgende BASIC-Programm ein und starten Sie mit **RUN**

```
100 REM EMail-Test
110 XSEMAIL 1                : REM sende auf COM1 = @
120 STOP
```

Wichtiger Hinweis zu XPort®
Damit eine EMail versendet werden kann, muss der Administrator neben der Server-IP Adresse auch den Domain-Namen nebst Unit-Namen fest eingeben. Der EMail-Empfänger ist variabel. Die Einstellungen sind dennoch nicht trivial. Zum bsp. müssen Einstellungen wie der Triggereingang „General Purpose I/O“ als Eingang mit Aktiv „Low“ parametriert - oder eine bestimmte Zeichenfolge für das Auslösen der EMail vorgenommen werden. Weitere Informationen finden Sie in der XPort-Beschreibung des Herstellers unter:
http://www.lantronix.com/support/downloads/?p=XPORT

Anmerkung:
Um erste TCP/IP Winsock Verbindungstests mit XPort durchzuführen, können Programme wie Hyperterminal oder Device-Terminal mit zugewiesener IP-Adresse verwendet werden. Seriell tunneln lassen sich die Daten jedoch nur mit einer Re-Direct-Software (bsp. CRP von Lantronix®), damit ein virtueller COM-Port die Pakete entsprechend umlenkt. Bevor jedoch irgend eine Verbindung zu stande kommt, müssen alle Grundparameter via HTTP-Browser voreingestellt werden. Die erste Konfiguration erfolgt somit i.d.R. über den Internet-Explorer. Als lokale „Webadresse“ nutzt man zunächst die Standard XPort IP: **http://192.168.1.11**

Für den EMail-Versand benötigt man zusätzlich einen Online-Zugang zu einem SMTP-Server mit automatischer Authentifizierung und fest einstellbarer IP-Adresse mit Port 25. Eine gute Lösung ist ein Modem-Router mit DSL-Zugang (bsp. Fritz Box) zu einem Provider, der SMTP ohne weitere Anmeldung erlaubt, welchen man quasi als Remote-IP Server nutzt.

Folgende Webseiten sind zu diesem Thema sehr hilfreich:
http://www.msxfaq.de/internet/inetanbindungsmtp.htm
http://www.albert-rommel.de/ppp-addrconf.htm
http://www.elektronik-kompendium.de/sites/net/0903081.htm
http://www.avm.de/de/Produkte/FRITZBox_an_jedem_Anschluss_Internet.html

XPULSE : Ein einfacher Befehl für Burst-/ Patterngenerator als Winkelgeberemulation

Mit diesem Befehl lassen sich abwechselnd verschiedene Logik-Pattern von 0...FFFF counts auf einer I/O-Adresse in Echtzeit erzeugen. Zur Ausgabe des Signals kann beispielsweise die LPT2 (auf &29h) oder AUX-Schnittstelle (auf &49h) mit TTL-Pegel verwendet werden. Die LPT2-Schnittstelle (mit 8255 Chip) wird bereits beim booten auf Ausgabe initialisiert, um beipielsweise einen zweiten Drucker anzusteuern. Somit lässt sich der Daten-Port (PB) auch für andere Zwecke verwenden, wenn auf diese I/O-Adresse direkt geschrieben wird.

Mit dem erzeugten Signal lassen sich beipielsweise programmierbare Bursts (bis zu 65535 Pulsen) mit versetzten Pegeln wie bei einem Drehgeber simulieren, oder einfach nur eine bestimmte Anzahl von Impulsen zu Prüfzwecken ausgeben. Wendet man den Befehl mehrmals hintereinander an, sind entsprechend höhere Counter-Bursts möglich. Während der XPULSE-Befehl ausgeführt wird, sollte die CPU keinen Interrupt verarbeiten.

Geben Sie das nachfolgende BASIC-Programm ein und starten Sie mit **RUN**

Beispiel Mode 0 (einfacher Impulsgenerator)

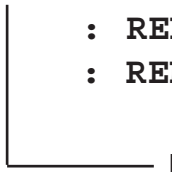
```
100 REM XPULSE-Test Port Toggle = 01010101 -> 10101010
110 XPULSE (&29),65535,0  : REM erzeugt jetzt 65535 Pulse
120 XDELAY (5000)         : REM kleine Pause
130 OUT (&29),0           : REM Port auf Null zurücksetzen
```

Beispiel Mode 1 (Drehgeber-Simulation B vor A Signal voreilend)

```
100 REM XPULSE-Test Bit0=A, Bit1=B, Bit2=/A, Bit3=/B
110 XPULSE (&29),1000,1   : REM erzeugt jetzt 1000 Pulse
120 XDELAY (5000)         : REM kleine Pause
130 OUT (&29),0           : REM Port auf Null zurücksetzen
```

Beispiel Mode 2 (Drehgeber-Simulation A vor B Signal nacheilend)

```
100 REM XPULSE-Test Bit0=A, Bit1=B, Bit2=/A, Bit3=/B
110 XPULSE (&29),2000,2   : REM erzeugt jetzt 2000 Pulse
120 XDELAY (5000)         : REM kleine Pause
130 OUT (&29),0           : REM Port auf Null zurücksetzen
```



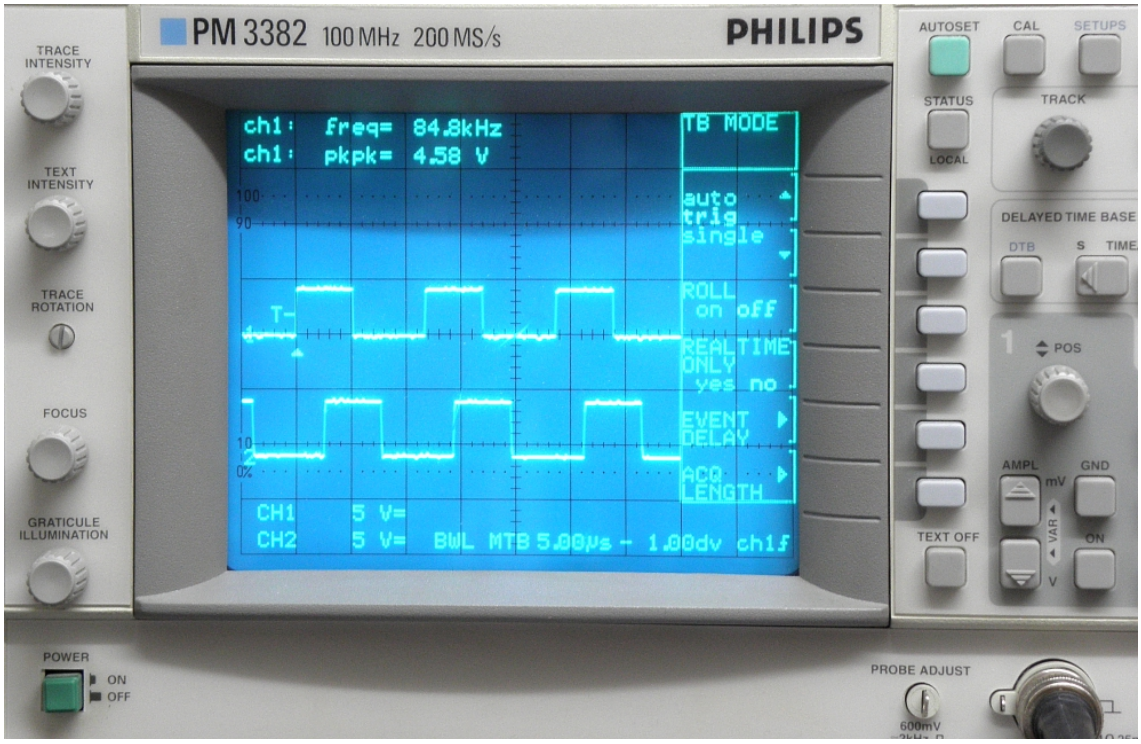
Hinweis zur Geschwindigkeit:
Signal-Taktfrequenz bei 20 MHz CPU mit Bus-Waitstate DIP-Schalter 1 = ON

Mode 0 = 150 kHz
Mode 1 = 85 kHz
Mode 2 = 85 kHz

Verwendet man Data Bit4 und Bit5 für Z und /Z, kann über den OUT-Befehl zusätzlich ein Reset für den empfangenden Quadraturempfänger programmiert werden.

XPULSE

Bild: Drehgeber-Oszillogramm Leitung Bit0 (A) und Bit1 (B) von LPT2 Datenport.



Befehl: XPULSE (&29),1000,2

XIPEAK : Einzel-Impuls mit variabler Dauer erzeugen

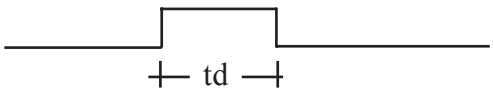
Dieser Befehl erzeugt auf einem TTL I/O-Port einen einzelnen Impuls mit variabler Dauer (Toggle-byte: 01010101 zu 10101010). Somit kann beispielsweise ein Puls-Störgenerator oder Trigger für Spannungsunterbrechungen (mit ext. Leistungsendstufe) realisiert werden. Die Dauer wird als 16-Bit Zahl übergeben, so dass die Impulszeit von ~5µs bis zu 85,2ms einstellbar ist. Mode wird zur Zeit mit 0 übergeben und dient für spätere Erweiterungen. Zur Ausgabe des Signals kann beispielsweise die LPT2 (auf &29h) oder AUX-Schnittstelle (auf &49h) mit TTL-Pegel verwendet werden. Die LPT2-Schnittstelle (mit 8255 Chip) wird bereits beim booten von SEPIA auf Ausgabe initialisiert, um beipielsweise einen zweiten Drucker anzusteuern. Somit lässt sich der Daten-Port auch für andere Zwecke verwenden, wenn auf diese Adresse direkt geschrieben wird. Während der XIPEAK-Befehl ausgeführt wird, sollte die CPU keinen Interrupt verarbeiten. Alle Impulszeiten ergeben sich bei 20 MHz CPU-Takt mit Waitstate DIP-Schalter 1 = ON (DIP-Switch auf Bus-Backplane).

Geben Sie das nachfolgende BASIC-Programm ein und starten Sie mit **RUN**

```
100 REM XIPEAK-Test mit 1,3 ms Impuls
110 :
120 A = (&29)          : REM I/O-Adresse für TTL-Port
140 T = 1000           : REM 16Bit Variable für Impulszeit
160 M = 0              : REM Mode 0
180 :
200 XIPEAK (A),T,M     : REM Befehl ausführen
```

Die folgende Tabelle zeigt den Bezug bei Mode 0 zwischen Impulszeit und Einstelldauer:

Time (word)	~ Dauer	exakte Dauer (td)	
T = 1	5 µs	5,028 µs	(kleinste Verzögerung)
T = 2	6,5 µs	6,430 µs	
T = 5	10 µs	10,23 µs	
T = 10	17 µs	16,83 µs	
T = 20	30 µs	29,83 µs	
T = 50	69 µs	68,83 µs	
T = 100	133 µs	133,83 µs	
T = 767	1000 µs	1,000 ms	
T = 1000	**1,3 ms**	**1,3038 ms**	(wie im Beispiel angegeben)
T = 10000	13 ms	13,003 ms	
T = 16384	21 ms	21,202 ms	
T = 32768	42 ms	42,602 ms	
T = 65535	85 ms	85,199 ms	
T = 0	85 ms	85,200 ms	(längste Verzögerung)



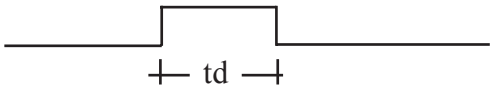
XIPEAK : Fortsetzung mit Mode = 1

Geben Sie das nachfolgende BASIC-Programm ein und starten Sie mit **RUN**

```
100 REM XIPEAK-Test mit Mode = 1 für eine Sekunde Pulse
110 :
120 A = (&29)          : REM I/O-Adresse für TTL-Port
140 T = 312            : REM 16Bit Variable für Impulszeit
160 M = 1              : REM Mode 1
180 :
200 XIPEAK (A),T,M     : REM Befehl ausführen
```

Die folgende Tabelle zeigt den Bezug bei Mode 1 zwischen Impulszeit und Einstelldauer:

Time (word)	~ Dauer	exakte Dauer (td)	
T = 1	3,2 ms	3,205 ms	(kleinste Verzögerung)
T = 2	6,4 ms	6,406 ms	
T = 5	16 ms	16,010 ms	
T = 10	32 ms	32,017 ms	
T = 20	64 ms	64,032 ms	
T = 40	128 ms	128,060 ms	
T = 50	160 ms	160,074 ms	
T = 80	256 ms	256,117 ms	
T = 100	320 ms	320,145 ms	
T = 200	640 ms	640,286 ms	
T = 312	**1 Sekunde**	**998,843 ms**	(wie im Beispiel angegeben)
T = 625	2 Sek.	2000,99 ms	
T = 1000	3,2 Sek.	3201,41 ms	
T = 2000	6,4 Sek.	6402,82 ms	
T = 4000	12,8 Sek.	12805,64 ms	
T = 10000	32 Sek.	32014,09 ms	
T = 0	~ 210 Sek.		(längste Verzögerung)



Hinweis:
Dieser Befehl hat eine sehr hohe Priorität. Während der Ausführung zählt die CPU die programmierte Zeitschleife strikt herab. Somit können keine anderen Funktionen zeitgleich ausgeführt oder abgefragt werden.

XIRQON und XIRQOFF - Interrupt

Diese Befehle veranlassen auf der RTC-Karte einen Interrupt-Generator im 1 Sek. Taktimpuls auszulösen, so dass auf der aktuellen Bus-Nummer ein Interrupt mit Vector 038h erzeugt wird, um beispielsweise eine spezielle ISR-Routine mit hoher Priorität direkt im EPROM anzusteuern.

Die Schaltung (ATMEL-Prozessor) erzeugt nach Freigabe einen Puls mit 1000 ms Dauer, bestehend aus 1 µs neg. Puls + 999,999 ms pos. Puls, so dass im Endeffekt eine Periodizität von einer Sekunde erreicht wird.

Geben Sie das nachfolgende BASIC-Programm ein und starten Sie mit **RUN**

```
100 XIRQON                : REM Generator einschalten
110 XDELAY (50000)        : REM Zeitschleife
120 XIRQOFF               : REM Generator ausschalten
130 STOP                  : REM Programmende
```

Hinweis:

Nach Zeile 100 erfolgt nun eine sekündliche Konsolenausgabe mit: **IRQ: xx**
(xx Steht für Bus-Steckplatz 1...14), bis die Delay-Schleife in Zeile 110 abgearbeitet ist.

Mit XIRQOFF wird in Zeile 120 der Interrupt-Generator wieder abgeschaltet (disable) und anschließend mit Zeile 130 das Programm ganz beendet.

Der Anwender hat nun die Möglichkeit, über eine Assembler-Programmierung mittels neuer Sprungadresse (an 038h im EPROM) eine eigene ISR-Routine im oberen SRAM-Bereich oder im oberen EPROM-Bereich (7FF0-7FFEh) zu hinterlegen. (Programmer erforderlich)

Durch den erzeugten Interrupt vergrößert sich das Zeitverhalten der hier verwendeten Delay-Schleife nur marginal um die Ausgabedauer der erzeugten Meldung auf dem Bildschirm. Diesen Umstand muss man bei der Programmierung stets mit berücksichtigen, wenn ereignis- und/oder zeitkritische Unterprogramme über Interruptanforderungen abgearbeitet werden sollen.

Hardwaretechnisch befindet sich das Enable-Bit auf der I/O-Adresse 02Eh (Data 0).
Mit XIRQON wird gleichzeitig ein EI Befehl zu CPU abgesetzt.

Weitere BASIC-Beispiele

```
100 REM Kreisumfang berechnen
110 INPUT „Radius“;R
120 U=2*PI*R
130 F=PI*R*R
140 PRINT „RADIUS=“;R
150 PRINT „UMFANG=“;U
160 PRINT „FLÄCHE=“;F
```

```
100 REM Cosinuns
110 PRINT COS(PI)
130 D=180
140 R=D*PI/180
150 PRINT COS(R)
```

```
100 REM String Zeichenkette bearbeiten
110 A$="Test123"
120 B$=LEFT$(A$,3)
130 PRINT B$
```

```
100 INPUT"Wort,Länge";A$,L
110 IF A$="E"THEN END
120 A$=FIX$(A$,L)
130 PRINT A$;"Länge";LEN(A$)
```

```
100 REM IDE-Medien scannen mit "DRIVE S" Befehl
110 CLS
120 LOCATE 2,2
130 DRIVE S
```

```
100 REM auf Taste warten
110 XWAITKEY
120 PRINT "o.k."
```

```
100 REM ASCII-Zeichencode zeigen
110 CLS
120 LOCATE 2,2
130 B = ASC(INKEY$)
140 PRINT "ZEICHEN "; CHR$(B); "  DEZ.:";B
150 IF B = 5 THEN STOP ELSE GOTO 130
```

```
100 INPUT "GEBEN SIE EINE ZAHL EIN", A
110 LET B=A*A
120 PRINT "DAS QUADRAT VON "; A; " IST "; B
```

```
100 INPUT "GEBEN SIE EINE POSITIVE ZAHL EIN", A
110 IF A>=0 THEN 140
120 PRINT "DIE ZAHL "; A; "IST NEGATIV!"
130 GOTO 100
140 PRINT "DIE WURZEL VON "; A; " IST "; SQR( A )
```

```
100 INPUT "LAENGE DER 1. KATHETE:", A
110 INPUT "LAENGE DER 2. KATHETE:", B
120 C = SQR( A*A + B*B )
130 PRINT "LAENGE DER HYPOTENUSE: "; C
```

Steuerzeichen lt. ASCII Tabelle

```
00 - NUL   Null character (Nullzeichen)
01 - SOH   Start of heading (Anfang Dokumentkopf)
02 - STX   Start of text (Textanfang)
03 - ETX   End of text (Textende)
04 - EOT   End of transmission (Ende der Übertragung)
05 - ENQ   Enquiry (Anfrage)
06 - ACK   Acknowledgment (Bestätigung)
07 - BEL   Audible bell (Tonsignal)
08 - BS    Backspace (Rückschritt)
09 - HT    Horizontal tab (Horizonaltabulator)
10 - LF    Line feed (Zeilenvorschub)
11 - VT    Vertical tab (Vertikaletabulator)
12 - FF    Form feed (Seitenvorschub)
13 - CR    Carriage return (Wagenrücklauf)
14 - SO    Shift out (Dauerumschaltung)
15 - SI    Shift in (Ende Dauerumschaltung)
16 - DLE   Data link escape (Datenverbindungs-Fluchtsymbol)
17 - DC1   Device control 1 (Gerätekontrollcode 1)
18 - DC2   Device control 2 (Gerätekontrollcode 2)
19 - DC3   Device control 3 (Gerätekontrollcode 3)
20 - DC4   Device control 4 (Gerätekontrollcode 4)
21 - NAK   Negative acknowledgment (Negative Bestätigung)
22 - SYN   Synchronous idle (Synchronisationssignal)
23 - ETB   End of transmission block (Ende des Übertragungsblockes)
24 - CAN   Cancel (Abbruch)
25 - EM    End of medium (Ende des Mediums)
26 - SUB   Substitute character (Ersetzen)
27 - ESC   Escape (Fluchtzeichen)
28 - FS    File separator (Dateitrenner)
29 - GS    Group separator (Gruppentrenner)
30 - RS    Record separator (Datensatztrenner)
31 - US    Unit separator (Einheitentrenner)
127 - DEL  Delete (Entfernen/Löschen)
```

ASCII-Code-Tabelle der ISO-8859-Familie (Byte-Werte 00 bis 127)

00	NUL	**01**	SOH	**02**	STX	**03**	ETX	**04**	EOT	**05**	ENQ	**06**	ACK	**07**	BEL
08	BS	**09**	HT	**10**	LF	**11**	VT	**12**	FF	**13**	CR	**14**	SO	**15**	SI
16	DLE	**17**	DCI	**18**	DC2	**19**	DC3	**20**	DC4	**21**	NAK	**22**	SYN	**23**	ETB
24	CAN	**25**	EM	**26**	SUB	**27**	ESC	**28**	FS	**29**	GS	**30**	RS	**31**	US
32	SP	**33**	!	**34**	„	**35**	#	**36**	$	**37**	%	**38**	&	**39**	‘
40	(	**41**	)	**42**	*	**43**	+	**44**	,	**45**	-	**46**	.	**47**	/
48	0	**49**	1	**50**	2	**51**	3	**52**	4	**53**	5	**54**	6	**55**	7
56	8	**57**	9	**58**	:	**59**	;	**60**	<	**61**	=	**62**	>	**63**	?
64	@	**65**	A	**66**	B	**67**	C	**68**	D	**69**	E	**70**	F	**71**	G
72	H	**73**	I	**74**	J	**75**	K	**76**	L	**77**	M	**78**	N	**79**	O
80	P	**81**	Q	**82**	R	**83**	S	**84**	T	**85**	U	**86**	V	**87**	W
88	X	**89**	Y	**90**	Z	**91**	[	**92**	\	**93**	]	**94**	^	**95**	_
96	‘	**97**	a	**98**	b	**99**	c	**100**	d	**101**	e	**102**	f	**103**	g
104	h	**105**	i	**106**	j	**107**	k	**108**	l	**109**	m	**110**	n	**111**	o
112	p	**113**	q	**114**	r	**115**	s	**116**	t	**117**	u	**118**	v	**119**	w
120	x	**121**	y	**122**	z	**123**	{	**124**	|	**125**	}	**126**	~	**127**	DEL

Übersicht der vorprogrammierten F-Tastenbelegung

Kürzel	Befehl
F1	LIST
F2	RUN
F3	LOAD
F4	SAVE
F5	Renumber 100,20
. . .	
F6	- noch nicht vorbelegt -
F7	- noch nicht vorbelegt -
F8	- noch nicht vorbelegt -
F9	- noch nicht vorbelegt -
F10	- noch nicht vorbelegt -
F11	- noch nicht vorbelegt -
F12	- noch nicht vorbelegt -

Web - Links zu technischen Dokumentationen im PDF-Format:

Info-Flyer	http://www.kolter.de/SEPIA_Flyer01.pdf
SEPIA Datenblatt	http://www.kolter.de/SEPIA_dblatt01.pdf
Übersicht Komponenten	http://www.kolter.de/SEPIA-Konfiguration.pdf
Stromverbrauch	http://www.kolter.de/SEPIA-Stromverbrauch.pdf
CE-Erklärung	http://www.kolter.de/ce-sepia.pdf
SEPIA-Bus Platine	http://www.kolter.de/SEPIA-BUS.pdf
CPU	http://www.kolter.de/SEPIA-CPU.pdf
IDE	http://www.kolter.de/SEPIA-IDE.pdf
NET (inkl. VGA)	http://www.kolter.de/SEPIA-NET.pdf
MEM	http://www.kolter.de/SEPIA-MEM.pdf
RTC (inkl. WatchDog)	http://www.kolter.de/SEPIA-RTC.pdf
CLK	http://www.kolter.de/SEPIA-CLK.pdf
RL8 - Relais	http://www.kolter.de/SEPIA-RL8.pdf
OI8 - Opto Input	http://www.kolter.de/SEPIA-OI8.pdf
AD1 - Analog Input	http://www.kolter.de/SEPIA-AD1.pdf
CNT - Counter	http://www.kolter.de/SEPIA-CNT.pdf
E32 - Opto Input	http://www.kolter.de/SEPIA-E32.pdf
A32 - Opto Output	http://www.kolter.de/SEPIA-A32.pdf
DA1 - DAC Ausgang	http://www.kolter.de/SEPIA-DA1.pdf
PW1...7 DC-Wandler	http://www.kolter.de/SEPIA-PWx.pdf
Netzteil - Datenblatt	http://www.kolter.de/GS120-spec.pdf
- - -	
Foto, 42 TE Gehäuse	http://www.kolter.de/TG42-seitlich-002a.jpg
Foto, 63 TE BGT	http://www.kolter.de/SEPIA_Bild7.jpg
Foto, Frontplatte CLK	http://www.kolter.de/ZMCBTL-001.jpg