

PETER'S ASSEMBLERECKE

Schnelle Linien

Jedesmal, wenn es interessant werde, so beschwerte sich W. Arnold in einem Leserbrief, komme anstelle einer eigenen Lösung eine Routine aus dem Betriebssystem zur Anwendung. Die Assemblerecke solle schließlich darlegen, so schrieb er weiter, wie man ein bestimmtes Problem selbst lösen kann, ohne gleich die Hilfe des Betriebssystems zu benötigen.

Recht hat er! Der Wunsch unserer Leser ist uns selbstverständlich Befehl. Sollten Sie Anregungen für künftige Themen haben, die Sie gern in der Assemblerecke behandelt sehen würden, so schreiben Sie uns. W. Arnold teilte uns noch mit, er sei besonders an einer möglichst schnellen DRAWTO-Routine interessiert. In Ausgabe 1/1986 war ein ähnliches Thema zu finden; allerdings wurden dort die mißliebigen ROM-Routinen verwendet. Nehmen wir nun einmal die Sache selbst in die Hand. Was dabei herauskommt, zeigt Listing 1.

Lassen Sie sich durch seine Länge nicht erschrecken. Wie bei fast allen Assembler-Programmen wird der wesentliche Teil (die Linien-Routine) durch viel Beiwerk wie Variablen-Definitionen, Demo und Utility-Unterprogramme ein wenig in den Hintergrund gedrängt.

Den harten Kern bildet die Unteroutine DRAWTO. Sie übernimmt die Berechnung der Verbindungspunkte zwischen einem gegebenen Anfangs- und Endpunkt. Hierzu wird ein sehr schneller Algorithmus verwendet, den Paul Chabot in der amerikanischen Zeitschrift ANTIC (Ausgabe 6/85) veröffentlichte. Dort war das Programm zwar in Action! codiert, aber die Umsetzung in Assembler macht die Routine noch schneller.

Die DRAWTO-Routine erledigt ihre Aufgabe durch eine

Fallunterscheidung in vier Quadranten. Sie errechnet den Abstand von Anfangs- zu Endpunkt sowohl für die X- als auch die Y-Koordinate. Der längere Weg dient als Zähler für die Menge der zu bestimmenden Punkte. Während der längere Abstand dann einfach linear durchgezählt wird, nimmt eine geschickte Näherung die Korrektur des kürzeren vor.

Auf diese Art erspart man sich die umständlichen und zeitraubenden Multiplikationen bzw. Divisionen, die bei einer direkten Anwendung der Geradengleichung $Y = mx + t$ zwangsläufig durchzuführen wären. Aus diesem Grunde arbeitet diese DRAWTO-Routine auch verhältnismäßig schnell. Wenn Sie Spaß daran haben, können Sie das Demo des Assembler-Programms auch versuchsweise einmal in Basic codieren, was keine großen Schwierigkeiten bereitet. Hier sind deutlich weniger als 10 Zeilen erforderlich, aber der Unterschied in der Ablaufgeschwindigkeit ist einfach enorm.

Gerechterweise muß man aber sagen, daß dies nicht an einer schlechten Programmierung des Betriebssystems liegt, sondern daran, daß die Routinen im OS sehr allgemein gehalten sind. Während die in Listing 1 vorgestellten Routinen nur mit bestimmten Grafikaufösungen funktionieren, ist die im OS für alle Grafikmodi ausgelegt. Das kostet natürlich Rechenzeit.

Hinzu kommt, daß auch der beste Algorithmus zur Berechnung der Linie nichts nützt, wenn die Ausgabe von einzelnen Grafikpunkten nicht ebenfalls optimiert ist. Deshalb wird im Programmbeispiel eine tabellenorientierte PLOT-Funktion verwendet, ebenfalls eine Methode, die sich aufgrund des Speicherplatzbedarfs für ein Betriebssystem in einem 16K-

ROM von vornherein verbietet.

Sie sehen jetzt schon, wo der Trennungsstrich zu ziehen ist. Die Benutzung von OS-Routinen ist überall dort zu empfehlen, wo es nicht auf Geschwindigkeit, sondern auf Vielseitigkeit ankommt. Eigene Routinen sind dagegen für spezielle Anwendungen (wie z.B. Grafikprogramme, Spiele) unentbehrlich.

Noch ein paar Anmerkungen zum Listing: Sie können es mit ATMAS II assemblieren und im Monitor mit G A800 starten. Diesmal wurden (zur Verringerung der Schreiarbeit) zwei Makros, ADD und SUB, eingeführt. Sie erleichtern die

Rechenoperationen mit 16-Bit-Zahlen. Zur Anwendung kommt die zweifarbige Grafikstufe 6, die durch einen CIO-Aufruf eingeschaltet wird (Unterprogramm GRAPHICS).

Wollen Sie andere Grafikstufen benutzen, so müssen die Routinen PLOT und PLOTAB entsprechend modifiziert werden (Bytes pro Zeile usw.). Das PLOT-Unterprogramm arbeitet übrigens im EXOR-Modus; daher ist es möglich, eine Linie einfach durch nochmaliges Zeichnen zu entfernen. Wenn Sie den normalen Zeichenmodus verwenden wollen, muß der EOR-Befehl der PLOT-Routine in eine ORA-Anweisung abgewandelt werden.

P. Finzel

Sourcelisting

```
*****
*           Schnelle Linienroutine
*
```

```
*Assembler: ATMAS-II
```

```
*Peter Finzel                               1987
*****
```

```
* IOCB-Struktur, CIO-Befehle...
```

```
*
ICCOM EQU $342
ICBAL EQU $344
ICBAH EQU $345
ICAX1 EQU $34A
ICAX2 EQU $34B
CIOV EQU $E456 CIO-Vektor
COPEN EQU 3
CCLSE EQU 12
```

```
* Betriebssystem-Variable
```

```
* SAVMSC EQU $58 Bildschirm-Adresse
```

```
* Konstante fuer GRAPHICS 6
```

```
*
YMAX EQU 96 Auflösung vert.
XMAX EQU 160 Auflösung hor.
ZLAENGE EQU 20 Bytes pro Zeile
```

```
* Zeropage-Variable
```

```
*
XNEU EQU $E0 (Byte) Endpunkt X
YNEU EQU $E1 (Byte) Endpunkt Y
XALT EQU $E2 (Byte) Startpunkt X
YALT EQU $E3 (Byte) Startpunkt Y
ZAEHLER EQU $E4 (Byte) Pixelzaehler
DELTA X EQU $E5 (Byte) Abstand X
DELTA Y EQU $E6 (Byte) Abstand Y
XFLAG EQU $E7 (Byte) Merker links/rechts
YFLAG EQU $E8 (Byte) Merker oben/unten
HILFA EQU $E9 (Word) Hilfsregister
HILFB EQU $EB (Word) fuer Naeherung
HILFT EQU $ED (Word)
ZEIGER EQU $EF (Word) Vektor f. Plot
```

```

*
* Zwei Makros fuer 16-Bit Zahlen...
*
ADD     MACRO P1,P2
        CLC
        LDA P2
        ADC P1
        STA P2
        LDA P2+1
        ADC P1+1
        STA P2+1
        MEND

SUB     MACRO P1,P2,P3
        SEC
        LDA P1
        SBC P2
        STA P3
        LDA P1+1
        SBC #0
        STA P3+1
        MEND

*
        DRG $AB00 im res. Bereich
*****
* Demo-Programm 'Highlights'
*****
*
* GRAPHICS 6+16
*
        LDA #6+16    GRAPHICS 7,
        JSR GRAPHICS ganzen Screen
        JSR PLOTAB   Tabellen...
DEMO    LDA #0        Linienstart
        STA XSTART
DEMO2   LDA #160     Linienende
        STA XENDE

DEMO1   LDA XSTART   Anfangspunkt
        STA XALT     setzen (x & Y)
        LDA #0
        STA YALT
        LDY XENDE    Endpunkt setzen
        LDY #95

        JSR DRAWTO   Linie zeichnen

        SEC          neue Koordinaten
        LDA XENDE    berechnen
        SBC #20
        STA XENDE
        BNE DEMD1
        CLC
        LDA XSTART   Startpunkt
        ADC #39
        STA XSTART
        CMP #160
        BCC DEMO2
        JMP DEMO      Endloses Demo

*
* Variablen Demoprogramm
*
XSTART  DFB 0
XENDE   DFB 0
*
*****
* Schnelle Berechnung von Linien
*
* Parameter: XALT,YALT: Startpunkt
*             <X>,<Y> : Endpunkt
*****
DRAWTO  STX XNEU     Endpunkt merken

        STY YNEU
        LDX XALT     Anfangspunkt
        LDY YALT     zeichnen
        JSR PLOT
        LDA #0       Flags ruecksetzen
        STA XFLAG
        STA YFLAG
        LDX XNEU     Endpunkt in Register
        LDY YNEU     (einfacher)

        CPX XALT     Anfang = Ende?
        BNE DR1
        CPY YALT
        BNE DR1
        RTS          ja, fertig!==>

        DR1          CPX XALT     Neuer Punkt ist
        BCC DRX     links von alten P.->
        INC XFLAG   rechts!
        TXA          Delta ausrechnen
        SBC XALT
        JMP DR2

        DRX          DEC XFLAG   links!
        SEC
        LDA XALT     Delta ber.
        SBC XNEU

        DR2          STA DELTAX   Abstand merken
        CPY YALT     neuer Punkt ist
        BCC DRY     oberhalb alten P.
        INC YFLAG   unterhalb
        TYA
        SBC YALT     Delta berechnen
        JMP DR3

        DRY          DEC YFLAG   oberhalb
        SEC
        LDA YALT     Abstand (delta)
        SBC YNEU

        DR3          STA DELTAY   und merken
        LDA XALT     Neuer P.= alter P.
        STA XNEU
        LDA YALT
        STA YNEU
        LDA DELTAX   welches Delta ist
        CMP DELTAY   groesser?
        BCC DRYSTEP  DY ist groesser->

        LDA DELTAY   DX ist groesser!
        ASL
        STA HILFA    Naeherung fuer
        LDA #0       Schrittweite
        ROL
        STA HILFA+1

        SUB HILFA,DELTAX,HILFT
        SUB HILFT,DELTAX,HILFB
        LDA DELTAX   Zaehler fuer
        STA ZAEHLER  Pixels einrichten

        DXSCHL      CLC
        LDA XNEU     X weiterzaehlen
        ADC XFLAG
        STA XNEU
        LDA HILFT+1  Schritt nach Y
        BPL DRX5     erfoderlich? Ja->
        ADD HILFA,HILFT
        JMP DRX4     kein Schritt

        DRX5        CLC
        ADD HILFB,HILFT
        CLC
    
```

```

LDA YNEU      Schritt nach Y
ADC YFLAG     ausfuehren
STA YNEU

DRX4  LDX XNEU      Pixel plotten
      LDY YNEU
      JSR PLOT
      DEC ZAEHLER   ganzes Deltax
      BNE DXSCHL   abgefahren? nein->
      JMP DREND    fertig!==>

DRYSTEP LDA DELTAX   Delta Y war groesser
        ASL
        STA HILFA   Naehung fuer
        LDA #0      Schritt in X-Richt.
        ROL
        STA HILFA+1

        SUB HILFA,DELTAY,HILFT
        SUB HILFT,DELTAY,HILFB

        LDA DELTAY  Zaehler fuer Abstand
        STA ZAEHLER einrichten

DYSCHL CLC
        LDA YNEU   Y weiterzaehlen
        ADC YFLAG
        STA YNEU
        LDA HILFT+1 Schritt nach X
        BPL DRY5   noetig? ja ->
        ADD HILFA,HILFT
        JMP DRY4   keine X-Korrektur

DRY5  ADD HILFB,HILFT
      CLC
      LDA XNEU   X-Korrektur aus-
      ADC XFLAG  fuehren
      STA XNEU

DRY4  LDX XNEU      Pixel ausgeben
      LDY YNEU
      JSR PLOT
      DEC ZAEHLER   alle Punkte?
      BNE DYSCHL   nein-->

DREND LDX XNEU      Ende der Linie
      LDY YNEU      kann Anfang einer
      STX XALT       neuen sein.
      STY YALT
      RTS            fertig!

*****
* GRAPHICS-Unterprogramm
*
* Aufruf: JSR GRAPHICS
* PARAMETER:
* <A> 0 bis 15 (XL/XE)
*      0 bis 11 (400/800)
*****

GRAPHICS PHA          Graphik-Stufe merken
         LDX #$60     IOCB Nr. 6
         LDA #CCLSE   Screen-IOCB zuerst
         STA ICCOM,X  schliessen
         JSR CIOV
         PLA          Graphik-Stufe
         STA ICAX2,X  zurueckholen
         AND #$F0     und passende
         EOR #$10    Bit-Kombination
         ORA #$0C    fuer Handler
         STA ICAX1,X  herstellen
         LDA #COPEN  jetzt den Befehl
         STA ICCOM,X zum Deffen des Screens
         LDA #SDEVICE Zeiger auf Device-
         STA ICBAL,X bezeichnung

LDA #SDEVICE/256
STA ICBAL,X
JMP CIOV
RTS

SDEVICE ASC "S:"      Display-Handler

*****
* HI-SPEED PLOT fuer Einfarb-Modi
*
* Aufruf: JSR PLOT
*
* PARAMETER:
* <X>,<Y> je nach Graphikstufe
*      X,Y werden zerstoert!
*****

PLOT  CPY #YMAX      Grenzen
      BCS PLOTEND    pruefen
      CPX #XMAX
      BCS PLOTEND
      LDA ADRLO,Y    Bildschirm-
      STA ZEIGER     adresse
      LDA ADRHI,Y    in Zeropage
      STA ZEIGER+1
      TXA
      LSR             ;geteilt
      LSR             ;durch 8
      LSR
      TAY             ;Index f. X-Pos
      TXA             X-Position
      AND #7
      TAX
      LDA PIXTAB,X    Welches Pixel
      EOR (ZEIGER),Y  und Pixel manipulieren
      STA (ZEIGER),Y  zurueck in Graphik
PLOTEND RTS

*****
* Erzeugt Adresstabellen fuer Plot
*
* muss vor der ersten Verwendung von
* Plot und nach dem GRAPHICS-Befehl
* stehen!
*****

PLOTAB LDA SAVMSC    Anfangsadresse
        STA ZEIGER    des Video-Rams
        LDA SAVMSC+1
        STA ZEIGER+1
        LDY #0        Index auf 0

NXTADR LDA ZEIGER    Adresstabellen
        STA ADRLO,Y  aufbauen
        LDA ZEIGER+1
        STA ADRHI,Y  MSB-Tabelle
        CLC
        LDA ZEIGER    Adresse des
        ADC #ZLAENGE  naechsten Zeilen
        STA ZEIGER    anfangs berechnen
        LDA ZEIGER+1
        ADC #0
        STA ZEIGER+1
        INY
        CPY #YMAX    schon f. alle Zeilen?
        BNE NXTADR   nein -->
        RTS

*
* ab hier stehen die Tabellen
*
PIXTAB DFB 128,64,32,16,8,4,2,1
ADRLO  ORG ++YMAX    Platz fuer
ADRHI  ORG ++YMAX    Tabellen

```