Harald Astheimer

Grafik in C

Ein schnelles Programm für mc-68000-Computer

Um die Grafikfähigkeiten des mc-68000-Computers voll ausnutzen zu können, müssen diese auch von einer höheren Programmiersprache aus verfügbar sein. Dieser Artikel zeigt nicht nur die Einbindung der entsprechenden Routinen in die Sprache C, sondern stellt auch ein leicht verständliches Beispiel in dieser Sprache dar, die ja im Lieferumfang des CP/M-68k enthalten ist.

In mc 4/1985 wurden Assembler-Routinen vorgestellt, die eine komfortable Benutzung der Grafikmöglichkeiten zulassen. Diese sind ursprünglich für den Monitor-Assembler des mc-68000-Computers geschrieben worden, der jedoch in seinen Grundzügen mit dem Assembler von Digital Research unter CP/M-68k kompatibel ist. So lassen sich die Assembler-Routinen ohne Schwierigkeiten auch unter diesem Betriebssystem weiter verwenden. Will man jedoch auf eine höhere Programmiersprache als Werkzeug nicht verzichten und trotzdem den Geschwindigkeitsvorteil von Programmen in Maschinensprache nutzen, so muß ein Weg gefunden werden, diese Routinen von der höheren Programmiersprache aus aufrufen und ihnen Parameter übergeben zu können.

Der C-Compiler

Durch seinen modularen Aufbau bietet der C-Compiler von Digital Research hierfür gute Voraussetzungen. Übersetzt man das in *Bild 1* abgedruckte kleine C-Programm und betrachtet das so gewonnene Assembler-Programm (*Bild 2*), so kann man den allgemeinen Aufrufmechanismus und die Parameterübergabe verfolgen.
Das C-Programm besteht nur aus einer Typendeklaration und dem Aufruf einer Funktion mit dem Namen „test", wobei die Werte der beiden Integer-Variablen a und b an die Funktion test übergeben werden sollen. Die Zeilen 1...5 in Bild 2 dienen allgemeinen Vereinbarungen. In Zeile 6 wird das Adressregister A6 (die Register können in diesem Assembler auch mit den Namen R0...R14 angesprochen werden) als sogenannter Frame-Pointer mit dem Link-Befehl deklariert. Damit wird Platz im Speicher für Variablen und Parameter geschaffen (eine eingehende Beschreibung findet man in [4]). In den Zeilen 7 und 8 wird den Variablen a und b ein „displacement" zugewiesen, um ihren Werten jeweils einen Speicherplatz innerhalb des reservierten Speicherbereiches zuzuordnen. Der zum Studium der Parameterübergabe interessante Teil folgt nun in den Zeilen 10...12. Dort werden die Variablen a und b in umgekehrter Reihenfolge auf dem Stack abgelegt. Danach erfolgt sofort ein Unterprogramm-Aufruf mit dem JSR-Befehl, wobei sich der Name des Unterprogrammes nur durch eine vorangestellte Unterstreichung vom Namen im C-Programm unterscheidet. Nun kümmert sich der C-Compiler in keiner Weise darum, ob die aufgerufene Funktion innerhalb desselben Programmes definiert ist oder ob es sich um einen reservierten Namen handelt. Das ist erst eine Aufgabe des Linkers, der so aufgerufene Unterprogramme in ein ablauffähiges Programm einbindet. Damit ist für den Anwender die Möglichkeit gegeben, eigene Erweiterungen in ein übersetztes C-Programm einzufügen.

```
1       /*  Beispiel     */
2       #include "stdio.h"
3
4       main()
5       {
6        int a,b;
7        test(a,b);
8       }
```

Bild 1. Nur als Beispiel für Funktionsaufrufe und Parameterübergabe dient dieses kleine C-Programm

```
1       .globl __iob
2       .globl _main
3       .text
4       _main:
5       ßßmain:
6       link R14,#-8
7       ßa=-2
8       ßb=-4
9       *line 7
10      move -4(R14),(sp)
11      move -2(R14),-(sp)
12      jsr _test
13      addq.l #2,sp
14      L1:unlk R14
15      rts
16      .data
```

Bild 2. So sieht das Programm aus Bild 1 nach der Übersetzung durch den C-Compiler aus

Das Maschinenprogramm

Ein solches Maschinenprogramm muß also folgende Bedingungen erfüllen:
○ Als Einsprung muß ein globales Label definiert sein, dessen Name mit dem Funktionsnamen im C-Programm bis auf eine zusätzliche Unterstreichung am Wortanfang identisch ist.
○ Es muß sichergestellt sein, daß alle Registerinhalte nach der Ausführung des Maschinensprache-Programms unverändert sind.
○ Parameter werden über den Stack eingelesen.
Die erste Bedingung bedarf keiner weiteren Erläuterung, hierfür stehen geeignete Assembler-Direktiven zur Verfügung. Um die zweite Bedingung zu erfüllen, sollte nicht der Stack benutzt werden, da dieser ja zur Parameter-Übergabe gebraucht wird. Bei der dritten Bedingung schließlich muß berücksichtigt werden, welche Wortbreite die übergebenen Parameter haben sollen – in den hier aufgeführten Beispielen werden nur Integer-Werte übergeben, also Zahlen mit einer Breite von zwei Bytes. Bei der Berechnung der relativen Position der übergebenen Werte auf dem Stack muß berücksichtigt werden, daß, bedingt durch den JSR-Aufruf, eine Rücksprungadresse mit einer Breite von vier Bytes ebenfalls auf dem Stack liegt.

Der Funktionsaufruf

Im folgenden werden die vier Grafik-Funktionen „punkt", „gerade", „rechteck" und „polygon" beschrieben. Anhand der Namen ist schon zu erkennen, welche Aufgabe die einzelnen Funktionen übernehmen. Allen gemeinsam ist die Möglichkeit, nicht nur das Setzen, sondern auch das Löschen der jeweiligen Grafik-Ausgabe vornehmen zu kön-

nen. Die Form des Aufrufes von C aus ist in der *Tabelle* dargestellt.

Syntax der Funktionsaufrufe

```
Punkt(x,y,clrflag);
gerade(x1,y1,x2,y2,clrflag);
rechteck(x,y,breite,hoehe,clrflag)
polygon(anzahl,x1,y1,x2,y2,...,clrflag)
```

Alle Argumente müssen im C-Programm als Größen vom Typ Integer deklariert werden. In x-Richtung (horizontal) sind Werte im Bereich 0...639 erlaubt, in y-Richtung (vertikal) müssen sie im Bereich 0...199 liegen. Der Ursprung des Koordinatensystems liegt in der oberen linken Ecke. Die Variable clrflag entscheidet, ob gezeichnet (clrflag = 0) oder gelöscht (clrflag = 1) wird; die übergebenen Koordinaten in der Funktion „rechteck" beziehen sich auf den linken oberen Eckpunkt.

Ein Beispiel

Zur Verdeutlichung der Benutzung dieser Funktion ist in *Bild 3* ein kleines C-

```
/*****************************************************************************/
/*         Beispiel-Programm für die Benutzung der Grafik-                   */
/*                        Routinen 18.07.1985                                */
/*****************************************************************************/
/*    Programm-Name: PLOTDEMO.C                                              */
/*****************************************************************************/
#include "stdio.h"

main()
{
int pendown,rubout;
char esc,clrs;
 esc = 27;
 clrs = 12;
printf("%ccg\n%c\n",esc,clrs); /* Grafik-Modus und clear screen          */
 pendown = 0;                  /* Flag für zeichnen                      */
 rubout = 1;                   /* Flag für löschen                       */
 demo1(pendown);               /* Demo für Punkt setzen                  */
 waiting();                    /* Pause                                  */
 demo1(rubout);                /* dasselbe jetzt löschen                 */
 demo2(pendown);               /* Demo für Gerade zeichnen               */
 waiting();
 demo2(rubout);                /* wieder löschen                         */
 demo3(pendown);               /* Demo für Rechteck                      */
 waiting();
 demo3(rubout);                /* wieder löschen                         */
 demo4(pendown);               /* Demo für Polygon                       */
 waiting();
 demo4(rubout);                /* wieder löschen                         */
printf("%cct\n%c\n",esc,clrs); /* zurück in Textmodus und clear screen   */
}

   waiting()                   /* Warteschleife                          */
      {
     float sin();
     int i,a;
     for (i = 1; i <= 2000; i = i + 1)
     a = sin(i);                       /* Beschäftigung für den Rechner*/
         return(1);
      }

demo1(clrflag)                 /* Demo für Punkt zeichnen                */
 int clrflag;                  /* wenn clrflag = 0 dann zeichnen         */
  {                            /* wenn clrflag = 1 dann löschen          */
   int x,y;
   float sin();
    for (x = 1; x <= 639; x = x + 1)
      {
        /* Transformation der Wertebereiche */
        y = 100.0 + sin( x / 639.0 * 4.0 * 3.1415 ) * 100.0;

       punkt(x,y,clrflag);       /* Grafik-Routine                         */
      }
    return(1);
   }

demo2(clrflag)                 /* Demo für das Zeichnen von Geraden      */
  int clrflag;                 /* wenn clrflag = 0 dann zeichnen         */
    {                          /* wenn clrflag = 1 dann löschen          */
   int x1,y1,x2,y2;
    y1 = 0;                    /* die y-Koordinaten bleiben konstant     */
    y2 = 199;
    for (x1 = 0; x1 <= 639; x1= x1 + 10)
      {
      x2 = 639 - x1;                   /* x2 ist gegenläufig zu x1       */
      gerade(x1,y1,x2,y2,clrflag);     /* Grafik-Routine                 */
      }
   return(1);
    }

demo3(clrflag)                 /* Demo für das Zeichnen von Rechtecken   */
  int clrflag;                 /* wenn clrflag = 0 dann zeichnen         */
                               /* wenn clrflag = 1 dann löschen          */
{
   int x,y,breite,hoehe;
   y = 150;
   breite = 10;
   hoehe = 10;
   for (x = 10; x <= 400; x = x + 15)  /* x-Koordinate des oberen        */
    {                                  /* linken Punktes                 */
     y = y - 5;                        /* dazugehörige y-Koordinate      */
     breite = breite + 2;
     hoehe = hoehe + 2;
     rechteck(x,y,breite,hoehe,clrflag); /* Grafik-Routine               */
    }
  return(1);
 }

demo4(clrflag)                 /* Demo für das Zeichnen eines Polygons   */
 int clrflag;                  /* wenn clrflag = 0 dann zeichnen         */
{                              /* wenn clrflag = 1 dann löschen          */
 int anzahl,x1,y1,x2,y2,x3,y3;
 anzahl = 3;                   /* Anzahl der Ecken des Polygons          */
 y1 = 70;                      /* die jeweiligen Koordinaten der         */
 y2 = y1 + 20;                 /* Eckpunkte                              */
 y3 = y1 + 20;
 for (x1 = 10; x1 <= 520; x1 = x1 + 30)
  {
    x2 = x1 - 10;
    x3 = x1 + 10;
    polygon(anzahl,x1,y1,x2,y2,x3,y3,clrflag); /* Grafik-Routine          */
  }
return(1);
}
```

Bild 3. Dieses Demonstrationsprogramm ruft die Grafikroutinen auf

Programm abgebildet, das je ein Beispiel mit den einzelnen Funktionen auf den Bildschirm bringt.
Zunächst wird mit der Funktion „punkt" die Funktionskurve eines Sinus auf den Bildschirm gebracht. Nach einer kurzen Wartezeit wird diese Kurve dann mit der gleichen Funktion wieder gelöscht. Es folgen Demonstrationen der Funktion „gerade" und „rechteck", zum Schluß werden mit der „polygon"-Funktion Dreiecke auf dem Bildschirm ausgegeben.
Bild 4 zeigt das dazugehörige Programm in Maschinensprache. Hier sind die vier Funktionsnamen wiederzufinden, und zwar in der Liste der globalen Labels und als Unterprogramm-Einsprünge. Diese Unterprogramme haben alle die gleiche Struktur: Jeweils am Anfang werden durch die Routine „rescue" alle Registerinhalte in einen Pufferbereich gerettet, dann werden die Parameter vom Stack geholt und in den Registern den jeweiligen Grafikroutinen aus mc 4/1985 übergeben. Am Ende des jeweiligen Unterprogrammes werden die ursprünglichen Registerinhalte von der Routine „restore" wieder hergestellt.
Den Ablauf des Übersetzungsvorganges und das Zusammenbinden der einzelnen Module zeigt *Bild 5*. Zunächst wird das C-Programm von den drei Teilen des C-Compilers in eine Assembler-Quelldatei übersetzt, die dann vom AS68-Assemblerprogramm zu einer Objektdatei weiterverarbeitet wird. Das in Maschinensprache geschriebene Programm wird ebenfalls in eine Objektdatei übersetzt. Im letzten Schritt verbindet dann der Linker diese beiden Objektdateien sowie die benötigten Bibliotheksfunktionen zu einem ablauffähigen Programm. Nach diesem Verfahren ist es möglich, beliebige Erweiterungen des Sprachumfanges von C als Maschinenprogramm auszuführen, also die Laufzeit-Vorteile der Maschinensprache mit den Annehmlichkeiten einer höheren Programmiersprache zu verbinden.

```
*****************************************************************************
*                  Assembler-Routinen für Grafik                            *
*            vorbereitet zum Einbinden in C-Programme                       *
*****************************************************************************
* Programm-Name:  GRAFUTIL.S                                                *
*****************************************************************************
     .globl _punkt
     .globl _gerade
     .globl _polygon
     .globl _rechteck

     .text

_punkt:         jsr     rescue              *rette register
                move.w  #4(SP),D2           *x-Koordinate
                move.w  #6(SP),D3           *y-koordinate
                move.w  #8(SP),clrflag      *wenn clrflag = 1 dann löschen
                jsr     setpnt              *zeichne oder lösche Punkt
                jsr     restore             *stelle Register wieder her
                rts

_gerade:        jsr     rescue              *rette register
                move.w  #4(SP),D2           *x-Koordinate von Startpunkt
                move.w  #6(SP),D3           *y-Koordinate von Startpunkt
                move.w  #8(SP),A2           *x-Koordinate von Endpunkt
                move.w  #10(SP),A3          *y-Koordinate von Endpunkt
                move.w  #12(SP),clrflag     *wenn clrflag = 1 dann löschen
                jsr     draw                *zeichne oder lösche Gerade
                jsr     restore             *stelle Register wieder her
                rts

_rechteck:      jsr     rescue              *rette register
                move.w  #4(SP),x            *x-Koordinate linke obere Ecke
                move.w  #6(SP),y            *y-Koordinate linke obere Ecke
                move.w  #8(SP),breite       *Breite des Rechtecks
                move.w  #10(SP),hoehe       *Höhe des Rechtecks
                move.w  #12(SP),clrflag     *wenn clrflag = 1 dann löschen

                move.w  x,D2                *lade die Koordinaten der
                move.w  y,D3                *ersten Seite in die Register
                movea.w D2,A2               *Startpunkt:        x,y
                movea.w D3,A3               *Endpunkt   : x+breite,y
                adda.w  breite,A2
                jsr     draw                *zeichne oder lösche Gerade

                move.w  x,D2                *zweite Seite
                move.w  y,D3                *
                movea.w D2,A2               *Startpunkt:        x,y
                movea.w D3,A3               *Endpunkt   :       x,y+Höhe
                adda.w  hoehe,A3
                jsr     draw                *zeichne oder lösche Gerade

                move.w  x,D2                *dritte Seite
                move.w  y,D3                *
                movea.w D2,A2               *Startpunkt: x+Breite,y
                movea.w D3,A3               *Endpunkt   : x+Breite,y+Höhe
                add.w   breite,D2
                adda.w  breite,A2
                adda.w  hoehe,A3
                jsr     draw                *zeichne oder lösche Gerade

                move.w  x,D2                *vierte Seite
                move.w  y,D3                *Startpunkt:        x,y+Höhe
                add.w   hoehe,D3            *Endpunkt   : x+Breite,y+Höhe
                jsr     draw

                jsr     restore             *stelle Register wieder her
                rts

_polygon:       jsr     rescue              *rette Register
                move.w  #4(SP),anzahl       *Anzahl der Ecken im Polygon
                move.w  anzahl,D0           *Berechnung der Offset-Adresse
                mulu    #4,D0               *von clrflag
                addq.w  #6,D0               *Adresse =(Anzahl * 4) + 6
                move.w  #0(SP,D0.w),clrflag *holen von clrflag
                subq.w  #2,D0               *Adresse der letzen Koordinate
                movea.w D0,A1               *merken fuer Schleife
                move.w  #6(SP),D2           *erster Startpunkt setzen
                move.w  #8(SP),D3
                movea.w #10(SP),A2          *erster Endpunkt setzen
                movea.w #12(SP),A3
                jsr     draw                *erste Seite des Polygons
                move.w  #12,D5              *Offset fuer Eck-Adressen
loop:           move.w  A2,D2               *Endpunkt wird zu Startpunkt
                move.w  A3,D3
                addq.w  #2,D5               *naechster Endpunkt holen
```

Bild 4. Das Assembler-Programm läßt sich mit wenigen Änderungen auch von C aus aufrufen

Literatur

[1] *Schinz, Helge:* Grafik auf dem mc-68000-Computer mc 4/1985, Seite 68.

[2] *Digital Research:* CP/M-68k Operating System Programmer's Guide, 1983.

[3] *Digital Research:* The C-Language Programming Guide, 1983.

[4] *Kane, Gerry; Hawkins, Doug; Leventhal, Lance:* 68000 Assembly Language Programming, Osborne/McGraw-Hill, Bergeley, California, 1981.

```
                movea.w #0(SP,D5.w),A2
                addq.w  #2,D5
                move.w  #0(SP,D5.w),A3
                jsr     draw                    *zeichne Seite des Polygons
                cmpa.w  D5,A1                   *ist letzte Koordinte erreicht
                bne     loop                    *nein, dann nochmal
                move.w  A2,D2                   *Endpunkt wird zu Startpunkt
                move.w  A3,D3
                movea.w #6(SP),A2               *letzter Endpunkt ist wieder
                movea.w #8(SP),A3               *Anfangspunkt
                jsr     draw                    *letzte Seite zeichnen
                jsr     restore                 *stelle Register wieder her

rescue:         movem.l D0-A6,table             *alle Registerinhalte retten
                clr.l   D2                      *die benutzten Register
                clr.l   D3                      *löschen
                clr.l   A2
                clr.l   A3
                clr.w   clrflag                 *clrflag = 0
                rts

restore:        movem.l table,D0-A6             *alle Registerinhalte wieder-
                rts                             *herstellen
*-------------------------------------------------------------------------
* Routine zum Zeichnen einer Geraden aus MC 4/85 S.68
*          Uebergaberegister:          D2...x-Koordinate  max 639    Start-
*                                      D3...y-Koordinate  max 199    wert
* D2,D3 werden verändert
* A2,A3 bleiben unverändert           A2...x-Koordinate  max 639    End-
* freie Register D5,A1,A6             A3...y-Koordinate  max 199    wert
*-------------------------------------------------------------------------
draw:           clr.l           D4          *D löschen
                movea.w         #$1,A4      *Step in x- und y-Richtung = +1
                movea.l         A4,A5
                move.l          A2,D6       *in D6 dx = x1 - x0 berechnen
                sub.l           D2,D6
                bge             posx        *dx >= 0 ?
                neg.l           D6          *nein, dx := -dx (jetzt dx > 0)
                movea.w         #$ffff,A4   *Step in x-Richtung = -1
posx            move.l          A3,D7       *in D7 dy = y1 -y0 berechnen
                sub.l           D3,D7
                bgt             next        *dy > 0 ?
                beq             dy_0        *dy = 0 ?
                neg.l           D7          *nein, dy := -dy (jetzt dy > 0)
                movea.w         #$ffff,A5   *Step in y-Richtung = -1
                bra             next        *fertig
dy_0:           not.l           D4          *dy = 0 --> D < 0
next            bsr             setpnt      *Punkt bei (x,y) setzen
                cmpa.l          D2,A2       *x-Koordinate = x1 ?
                bne             noend       *nein, weiter
                cmpa.l          D3,A3       *y-Koordinate = y1 ?
                beq             enddraw     *ja, Gerade ist fertig
noend           tst.l           D4          *D >= 0  ?
                bge             ystep       *ja --> y-Step
xstep:          add.l           A4,D2       *x := x + x-Step
                add.l           D7,D4       *D := D + dy
                bra             next        * nächster Punkt zeichnen
ystep           add.l           A5,D3       *y := y + y-Step
                sub.l           D6,D4       *D := D - dx
                bra             next        *nächster Punkt zeichnen
enddraw:        rts
*-------------------------------------------------------------------------
*  Routine zum Zeichnen eines Punktes aus MC 4/85 S.68
*                Uebergaberegister        D2...x-Koordinate max 639
*                                         D3...y-Koordinate max 199
*-------------------------------------------------------------------------
setpnt:         lea     $FF8000,A0      *A0 mit Bildschirmbasis laden
                bsr     calctest        *Offset und Bitnummer berechnen
                bhi     noset           *ausserhalb des Bildschirms ?
                cmp.w   #$1,clrflag     *ist clrflag gesetzt ?
                bne     set             *nein, setze Punkt
                bclr    D1,$0(A0,D0.W)  *ja, Punkt löschen
                bra     noset
set:            bset    D1,$0(A0,D0.W)  *Punkt setzen
noset:          rts
*-------------------------------------------------------------------------
calctest:       cmp.w   #639,D2         *x-Koordinate zu groß ?
                bhi     end             *ja, Abbruch
                cmp.w   #199,D3         *y-Koordinate zu groß ?
                bhi     end             *ja, Abbruch
                move.w  D3,D0           *in D0 2048 * (y MOD8) berechnen
                ror.w   #5,D0           *2048= 2^11 = 2^(16-5)
                and.w   #$3800,D0
                move.w  D3,D1           *in D1 80 * INT(y/8) berechnen
                lsr.w   #3,D1           *8 = 2^3
                mulu    #80,D1
                add.w   D1,D0           *in D0 steht die Summe
                move.w  D2,D1           *in D1 INT(x/8) berechnen
                lsr.w   #3,D1
                add.w   D1,D0           *und zu D0 addieren
                move.w  D2,D1           *in D1 Bitnummer berechnen
                not.w   D1              *Bitnummer = 7 - (x MOD 8)
                cmp.b   D0,D0           *z-Flag im SR löschen
end:            rts

    .data
    .even
table:          ds.l 15                 *diverse Zwischenspeicher
x:              ds.w 1
y:              ds.w 1
breite          ds.w 1
hoehe           ds.w 1
clrflag         ds.w 1
anzahl          ds.w 1
.end
```

```
A>CP68 -I 0: PLOTDEMO.C PLOTDEMO.I

A>CO68 PLOTDEMO.I PLOTDEMO.1 PLOTDEMO.2 PLOTDEMO.3 -F

A>ERA PLOTDEMO.I

A>C168 PLOTDEMO.1 PLOTDEMO.2 PLOTDEMO.S

A>ERA PLOTDEMO.1

A>ERA PLOTDEMO.2

A>AS68 -L -U -S 0: PLOTDEMO.S

A>ERA PLOTDEMO.S

A>AS68 -L GRAFUTIL.S

A>LO68 -R -O PLOTDEMO.68K 0:S.O PLOTDEMO.O GRAFUTIL.O 0:CLIB 0:LIBF.A

A>PLOTDEMO
```

Bild 5. So werden die Programmteile übersetzt und zusammengebunden