Programm- und Dokumentationserstellung: ..... **Friedemann Kienzler**

==========

TI-99/4A plottet Funktionen mit einer Auflösung von 192 X 192 Graphikpunkten (wahlweise auf dem Bildschirm oder auf dem Drucker)

==========

Konfiguration: Micro-Computer TI-99/4A
ROM-Erweiterung TI-Extended-Basic
evtl. TI-32 Zeichen-Thermodrucker

Mit dem Programm „PLOTT“ (Programm- und Datenspeicher zusammen ca. 12 KByte) kann jede beliebige Funktion y=f(x) im Grobraster mit 24 X 24 oder im Feinraster mit 192 X 192 Graphikpunkten nur auf dem Bildschirm oder auf Bildschirm und Drukker gezeichnet werden.

Was muss der Benutzer alles beachten?

Nach Starten des Programms mit RUN (ENTER) werden auf dem Bildschirm alle notwendigen Instruktionen angezeigt:

- Zunächst muss in Programmzeile 480 die zu zeichnende Funktion in der Form ‚DEF Y=f(X)‘ eingegeben werden, z.B.

```
480 DEF Y=SIN(X)/X
```

- Anschließend ist das Programm erneut mit RUN 410 (ENTER) zu starten.

Was muss alles eingegeben werden?

Nach Starten des Programms mit RUN 410 (ENTER) muss folgendes eingegeben werden:

- Definitionsintervall (XU;XO)
  (Achtung: Bei falscher Eingabe oder falls die zu zeichnende Funktion auf dem eingegebenen Intervall nicht definiert ist, springt der Computer zu diesem INPUT-Befehl zurück; XU und XO müssen dann neu gewählt werden).

- „Wollen Sie die Grenzen vom Y-Bereich eingeben?“
  Wird diese Frage mit „J“ beantwortet, muss das Wertebereichintervall eingegeben werden (YU;YO).
  Im anderen Falle ermittelt der Computer selbst YU und YO so, dass alle Extrempunkte auf dem Display erfasst werden.

- „Zusätzlicher Ausdruck vom Printer erwünscht?“
  Wird diese Frage mit „J“ beantwortet, so werden auf dem Drukker Titel und Intervallgrenzen ausgegeben.
  Im anderen Falle wird dieser Programmteil übersprungen.

- „Mit welcher Auflösung soll gezeichnet werden?“
  Wird die „1“ gedrückt, entscheidet man sich für das Feinraster (192 X 192 Punkte).
  Wird die „2“ gedrückt, erfolgt die Ausgabe im Grobraster.

Nach dieser Eingabe wird die Funktion zunächst auf dem Bildschirm gezeichnet, anschließend erfolgt, falls dies zu Beginn eingegeben worden ist, die Ausgabe auf dem Drucker.

Wie arbeitet das Programm (Struktogramm)?

Nachdem sich der Benutzer für den „normalen“ Programmablauf entschieden hat („normal“ heißt: alle Vorzüge des Programms ausnützen!), also XU und XO eingegeben, YU und YO vom Computer berechnen lassen, die zusätzliche Ausgabe auf dem Thermo-Drucker und die hohe Auflösung von 192 X 192 Graphikpunkten gewählt hat, arbeitet das Programm folgendermaßen:

- Eingabe (XU;XO)
- Bestimmung der Extremwerte von Y und zugleich Test, ob die zu zeichnende Funktion auf dem eingegebenen Definitionsintervall definiert ist:
    * erster und zweiter definierter Funktionswert berechnen (falls vorhanden) und miteinander vergleichen (fallende oder steigende Tendenz)
    * f(X3) bis f(XO) werden der Größe nach sortiert
    * Extrempunkte (= Punkte, bei denen eine Änderung des Vorzeichens der Steigung festzustellen ist) werden abgespeichert und sortiert
    * YU und YO werden je nach Zahl der Extrempunkte bestimmt
- Ausgabe des Titels und XU, XO, YU, YO auf dem Drucker
- Wahl der Auflösung (192 X 192)
- Zeichnen der Funktion auf dem Bildschirm:
    * Hauptschleife (Spalte 5 bis 28; 192 X-Werte, 24 Blöcke)
    * Funktionswert von X bilden; 8 X-Werte pro Spalte
    * den einzelnen X-Werten wird ihre Lage in der jeweiligen Spalte zugewiesen (Block 1-24)
    * die ermittelten Koordinaten werden der Größe nach sortiert
    * die Blöcke in einer Spalte, die den 8 X-Werten zugewiesen wurden (mind. 1 Block, höchstens 8 Blöcke) werden nach ihrer Größe sortiert
    * die Zahl der verschiedenen Blöcke wird ermittelt (1-8)
    * die Anzahl der Funktionswerte der 8 X-Werte jeweils pro Block wird ermittelt
    * es beginnt das Codieren der Blöcke ins Binärsystem:
        > jedem Block werden seine Koordinaten zugewiesen
        > die nicht definierten Punkte im Block werden Null gesetzt
        > definierte Punkte im Block werden bestimmt
        > im jeweiligen Block werden die Binär-Codes je Blockreihe ermittelt
        > die Punkte des nächsten Blockes werden zurückgesetzt in Register 1 bis ...
        > die Reihen-Binär-Codes im Block werden in Links- und Rechts-Codes zerlegt
        > Umwandlung der 16 Binär-Codes (8 links, 8 rechts) ins Hexadezimalsystem

```
                > Nullsetzen der Koordinaten des letzten
                  Blockes
        * Ausgabe des 64er-Feldes eines Blockes,
          maximal 8 Blöcke pro Spalte!
        * Nullsetzen der laufenden Variablen
        * NEXT-Befehl der Hauptschleife (wird je Spalte um
          1 erhöht)
  - Drucker-Plott (Zeichnen der Funktion auf dem Drucker):
        * zeilenweise wird jedes Zeichen vom Bildschirm auf
          den Drucker übertragen
  - Ende:
        * akustisches Signal
        * in der rechten unteren Ecke des Bildschirms wird ein
          „E“ angezeigt
        * Endlosschleife
```

Wie lässt sich aus dem Grob- ein Feinraster machen?

Der Bildschirm besteht aus 24 X 28 Feldern (= Blöcken), die sich ihrerseits aus 8 X 8 Einzelpunkten zusammensetzen. Welche Punkte in einem solchen 64er-Feld gesetzt werden sollen, kann mit dem Befehl CALL CHAR festgelegt werden. Insgesamt können 111 verschiedene Blöcke definiert werden (Code 33-143). Werden beim Zeichnen einer Funktion mehr als 111 solche 64er-Felder benötigt, so wird in der rechten unteren Ecke des Bildschirms ein „H“ angezeigt, und die Codes in Verbindung mit dem CALL CHAR-Befehl beginnen wieder bei 33, d.h. die bereits gezeichneten Blöcke mit Code 33, 34, ... werden umdefiniert.

Die Codierung eines 64er-Blockes erfolgt im Hexadezimalsystem:

0000	0	1001	9	
0001	1	1010	A	
0010	2	1011	B	Eine Blockreihe (insge-
0011	3	1100	C	samt 8 Blockreihen) be-
0100	4	1101	D	steht aus zwei solchen
0101	5	1110	E	4er-Gruppen!
0110	6	1111	F	
0111	7	(Code)		1 = Punkt gesetzt
1000	8			0 = Punkt nicht gesetzt

z.B. CALL CHAR (102, „F1000CB401100A01“) definiert das Zeichen 102 als ein Block (64er-Feld) wie folgt:

Reihe 1	1111 0001	
Reihe 2	0000 0000	
Reihe 3	0000 1100	Achtung: vom Drucker wer-
Reihe 4	1011 0100	den die Reihe 1 und
Reihe 5	0000 0001	die Spalten 1, 7 und
Reihe 6	0001 0000	8 nicht ausgegeben
Reihe 7	0000 1010	(8 X 8 Matrix wird
Reihe 8	0000 0001	zur 7 X 5 Matrix!)

1234 5678
-Spalten-

Wie wird der Drucker angesteuert?

Entscheidet sich der Benutzer für die zusätzliche Ausgabe auf Dem Drucker, wird zunächst folgende Datei eröffnet:

```
OPEN #1:“TP.U.S“,OUTPUT
```

Mit der Software-Umschaltoption „.U“ (benutzerdefinierte Zeichen) können die durch CALL CHAR definierten Zeichen auch auf dem Drucker ausgegeben werden. Das „.S“ bei der Dateieröffnung bewirkt, dass der sonst übliche Druckzeilen-abstand auf Null zurückgesetzt wird.

Der Drucker-Plott auf dem TI-32-Zeichen-Thermodrucker ist allerdings wesentlich schlechter als die Bildschirmausgabe, da in Verbindung mit diesem Drucker von dem 64er-Feld nur 35 Zeichenpunkte ausgegeben werden, die anderen Punkte im Block werden nicht gedruckt!

Jeder einzelne Block auf dem Bildschirm wird auf den Drucker übertragen.

Speicherbelegung

```
XU ..... untere Grenze von X (Definitionsbereich)
XO ..... obere  Grenze von X (Definitionsbereich)
DX ..... (XO-XU)/192
MD ..... 1E99
X ..... laufende X-Variable (aus dem Definitionsbereich)
AFP ..... erster definierter Funktionswert
HK ..... Zähler für nicht definierte Funktionswerte von XU ausgeh.
FP ..... Vergleichsvariable zu Y; nach jedem Durchgang wird FP=Y
IND ..... Flag-Variable (0 oder 1)
IND2 ..... Flag-Variable (0 oder 1)
IND3 ..... Flag-Variable (0 oder 1)
IND4 ..... Flag-Variable (0 oder 1)
HU ..... kleinster Funktionswert
HO ..... größter   Funktionswert
ZAE ..... Zähler für die Extremwerte von Y
YU ..... untere Grenze von Y (Wertebereich)
YO ..... obere  Grenze von Y (Wertebereich)
MI ..... Schleifenvariable
VB ..... Austauschvariable beim Sortieren nach Größe
BR ..... Flag-Variable (0 oder 1)
YS ..... 192/(YU-YO)
DY ..... (YO-YU)/192
K ..... Rückmeldevariable bei CALL-KEY-Statement
S ..... Statusvariable    bei CALL-KEY-Statement
P ..... Zähler für die Funktionswerte innerhalb eines Blockes
Z ..... Hauptschleifenvariable
Z1 ..... Schleifenvariable
Z2 ..... Schleifenvariable
Z3 ..... Schleifenvariable
Z4 ..... Schleifenvariable
Z5 ..... Schleifenvariable
Z6 ..... Schleifenvariable
Z7 ..... Schleifenvariable
Z8 ..... Schleifenvariable
Z9 ..... Schleifenvariable
Z10 ..... Schleifenvariable
Z11 ..... Schleifenvariable
Z12 ..... Schleifenvariable
Z13 ..... Schleifenvariable
Z14 ..... Schleifenvariable
Z15 ..... Schleifenvariable
G ..... Reihe bzw. Zeile des Bildschirms, in der f(x) liegt
```

```
V ......... Austauschvariable beim Sortieren nach Größe
REIHE ..... Reihe  im Block, in der ein Punkt gesetzt wird
SPALTE .... Spalte im Block, in der ein Punkt gesetzt wird
Q ......... Schleifenvariable (Umwandlung Binär- in Hexadezimal-Code)
GR ........ ASCII-Wert des gerade definierten Blockes (33-142)
SA ........ Schleifenvariable (Titelunterprogramm)
```

Arrays (Feldvariablen):

```
LT(50) .... Extremwerte von Y (Wertebereich)
B(16) ..... Binär-Codes der 8 linken und 8 rechten 4er-Blockreihen
O(24) ..... Zähler der gesetzten Punkte je Block einer Spalte
BLOCK(8) .. Lage (= Zeile) der verschiedenen Blöcke je Spalte
H(8) ...... Zeile + Blocknummer in der Spalte / 10
F(8) ...... Blocknummer in der jeweiligen Spalte des jeweil. X-Wertes
LTD(8) .... verschieden auftretende Blocknummern je Spalte
U(8) ...... Zeilennummer des jeweiligen Blocks (= INT (H(1-8)))
A(8) ...... Binär-Codes der 8 Blockreihen
```

Strings (Zeichenketten):

```
DF$ ....... „UNTERE GRENZE VON Y = ?“
DG$ ....... „OBERE  GRENZE VON Y = ?“
ST$ ....... „J“ oder „N“ (Eingabe von YU, YO erwünscht?)
ER$ ....... „J“ oder „N“ (Ausgabe auf dem Drucker erwünscht?)
C$ ........ Hexadezimal-Code für eine 4er-Reihe im Block
            (1,2,...,E,F)
D$ ........ Hexadezimal-Code für ein 64er-Feld (D$ = D$&C$ in
            Schleife)
```

Die Werte der unterstrichenen Register müssen (per INPUT-Befehl) manuell eingegeben werden, die übrigen Speicher werden im Programmablauf berechnet bzw. belegt.

Wie läuft das Programm Zeile für Zeile ab (Programmablauf)?

Zeilen	Was wird gemacht?
0100-0250	REM-Titel
0260-0370	Instruktionen zum Starten des Programms werden auf dem Bildschirm angezeigt
0380-0510	Pre-Scan-Aufhebung
0520-0580	Löschen des Bildschirms; Anweisungen für den Computer bei Division durch Null und bei „falschen“ Argumenten, Ausgabe des Titels auf dem Bildschirm
0590-0640	Eingabe des Definitionsintervalles (XU;XO)
0650-0670	Eingabe des Wertebereichintervalles (YU;YO)
0680-0930	Bestimmung der Extremwerte von Y, Sortieren der 192 Funktionswerte und Festlegung des Wertebereichintervalles (YU;YO)
0940-0950	Ausgabe von YU und YO
0960-1030	Ausgabe des Titels und der Koordinatengrenzen auf dem Drucker
1040-1130	Wahl der Auflösung
1140-1170	Löschen des Bildschirms und Wertzuweisungen einiger Variablen
1180-1220	Beginn der Hauptschleife zum Zeichnen mit 36864 Punkten
1230-1330	Blöcke werden berechnet (1-24)
1340-1420	Ermittelte Koordinaten werden nach Größe sortiert
1430-1510	Blöcke werden nach Größe, d.h. Lage sortiert
1520-1560	Zahl der verschiedenen Blöcke wird ermittelt

```
1570-1600   Zahl der gesetzten Punkte der einzelnen Blöcke
            wird ermittelt
1610-1620   es beginnt das Codieren der Blöcke ins Binärsystem
1630-1670   jedem Block werden seine Koordinaten zugewiesen
1680-1750   nicht definierte Punkte im Block werden auf Null gesetzt,
            definierte Punkte im Block werden bestimmt
1760-1830   im jeweiligen Block werden Reihe, Spalte und Binär-
            Code je Reihe ermittelt
1840-1870   Punkte des folgenden Blockes werden in Register
            1, ... zurückgesetzt
1880-1930   die 8 Reihen-Binär-Codes je Block werden in insgesamt
            16 Codes, Links- und Rechts-Codes, zerlegt
1940-2130   Umwandlung der 16 Binär-Codes je Block in einen
            Hexadezimal-Code
2140-2170   Nullsetzen der Koordinaten des letzten Blockes
2180-2280   Ausgabe eines Blockes (es folgt: Sprung nach 1620)
            (der NEXT-Befehl in Zeile 2280 bezieht sich auf die
             Schleife, in welcher alle Blöcke innerhalb einer
             Spalte gezeichnet werden)
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
2290-2370   Nullsetzen der laufenden Variablen (es folgt: Sprung
            nach 1220)
            (der NEXT-Befehl in Zeile 2370 bezieht sich auf die
             Hauptschleife)
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
2380-2410   Einleitung des Programmendes
2420-2500   Ausgabe der gezeichneten Funktion auf dem Drucker
2510-2580   Ende: akustisches und optisches Signal; Endlosschleife
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
2590-2680   Unterprogramm 1: Ausgabe des Titels auf dem Schirm
2690-2760   Unterprogramm 2: Test, ob die Funktion auf dem ein-
                             gegebenen Intervall definiert ist
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
2770-2910   Zeichnen der Funktion im Grobraster (24 X 24 Punkte)
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
2920-2990   Verzweigungsadressen bei auftretenden Fehlermel-
            dungen während des Programmablaufs
```

Was ist sonst noch erwähnenswert?

- Die Pre-Scan-Aufhebung in den Zeilen 0380-0510 bewirkt, dass nach Eingabe von RUN nicht vor Ablauf des Programms Speiplätze reserviert, CALL-Statements festgelegt und etwaige Fehler gesucht werden. Die dadurch gewonnene Zeitersparnis beträgt etwa 7 Sekunden.

  Vor dem Befehl „!@P-“ müssen alle im Programm benutzten Variablen, Strings, Arrays, CALL-Statements und DEF-Statements aufgeführt werden.

- Bei Polstellen von Funktionen, also bei Division durch Null, wird das Zeichnen der betreffenden Funktion ohne Unterbrechung fortgesetzt; dasselbe gilt bei „falschen“ Argumenten wie z.B. bei einer negativen Diskriminante.

  Ist die Funktion überhaupt nicht bzw. auf dem eingegebenen Intervall nicht definiert, so wird dies mit entsprechendem Text angezeigt, die Eingabe muss neu erfolgen.

- Liegt ein Funktionswert außerhalb des festgelegten Wertebereichintervalls bzw. ist ein Funktionswert nicht definiert, so erscheint der „Plott-Punkt“ am oberen oder unteren Bildrand; es kann sich dann ein mehr oder weniger langer Strich ergeben (je nachdem, wieviele Werte außerhalb liegen bzw. nicht definiert sind, und ob sie nach + oder – tendieren!).

- Soll die Funktion in einer anderen Farbe und/oder auf einen anderen Bildschirmhintergrund gezeichnet werden, z.B. weiß auf schwarz, müssen folgende Programmzeilen eingefügt werden:

```
 495 CALL COLOR :: CALL SCREEN
1155 CALL SCREEN(2) :: FOR Z=0 TO 14 ::
     CALL COLOR (Z,16,2) :: NEXT Z
```

  (2 = Zeichen-Code für schwarz, 16 Zeichen-Code für weiß)

- Zeitaufwand:
  Der Benutzer muss zweimal warten, bis ein Ergebnis ausgegeben wird:
  a) Bestimmung der Extremwerte von Y, zugleich Test, ob die Funktion auf dem vorgegebenen Intervall definiert ist (die Funktion wird gezeichnet, wenn mindestens ein Funktionswert von den 192 definiert ist).
     -> Zeitaufwand: 30 – 60 Sekunden
        (Bei auftretenden nicht definierten Argumenten ist dieser Zeitaufwand etwas größer.)
  b) Zeichnen der Funktion mit einer Auflösung von (192 X 192 =) 36864 Graphikpunkten.
     -> Zeitaufwand: 4 – 5 Minuten
        (Grund: Es müssen sehr viele Schleifen durchlaufen werden.)

- Nachteil des Programms: Müssen mehr als 141 Blöcke auf dem Bildschirm ausgegeben werden, werden die anfangs gezeichneten Blöcke neu definiert, d.h. die Zeichnung der Kurve wird von vorne her „zerstört“. Der Grund dafür ist die begrenzte Zahl der zur Verfügung stehenden ASCII-Werten (33-143). Falls dies eintritt, wird es durch ein „H“ in der unteren rechten Ecke des Bildschirms optisch und durch ein Signal akustisch angezeigt.

- Ist die Funktion gezeichnet und, falls erwünscht, auf dem Drucker ausgegeben und erscheint in der rechten unteren Ecke des Bildschirms ein „E“, durchläuft der Computer eine Endlosschleife. Das Programm muss dann mit FCTN CLEAR gestoppt werden, die temporär umdefinierten Zeichen auf dem Bildschirm erscheinen dann wieder in ihrer ursprünglichen Form.

Beispieldurchlauf

Es soll die Funktion y = sin x / x im Intervall (-15; +15) gezeichnet werden (nur auf dem Bildschirm):

```
Tastatureingaben      Bildschirmausgaben/-anweisungen
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
RUN  ENTER            (Instruktionen zum Programmstart
                       werden angezeigt)
480 FCTN E
DEF Y=SIN(X)/X  ENTER
RUN 410  ENTER        (Titel wird angezeigt)
```

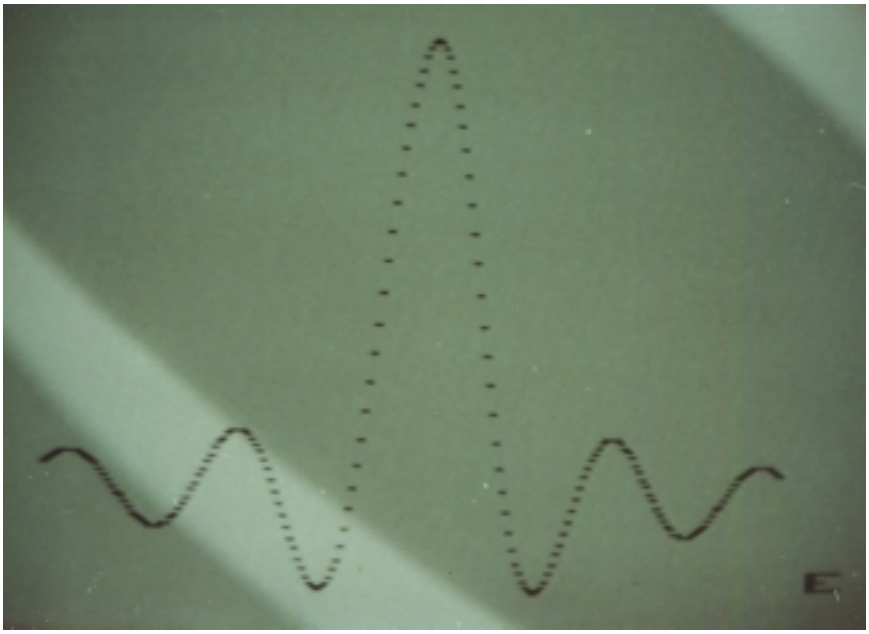
```
                    UNTERE GRENZE VON X = ?
-15  ENTER
                    OBERE  GRENZE VON X = ?
 15  ENTER
                    MOECHTEN SIE DIE GRENZEN VOM
                    Y-BEREICHN EINGEBEN (J/N) ?
 N  ENTER
                    --- BITTE WARTEN !! ---

                    UNTERE GRENZE VON Y =
                    -.2170789834
                    OBERE  GRENZE VON Y =
                     .9959359538

                    ZUSAETZLICHER AUSDRUCK VOM
                    PRINTER ERWUENSCHT (J/N) ?
 N  ENTER
                    MIT WELCHER AUFLOESUNG SOLL
                    GEZEICHNET WERDEN ?

                    DRUECKEN SIE:

                    1   BEI HOHER AUFLOESUNG MIT
                        192 X 192 GRAPHIKPUNKTEN
                    2   BEI GERINGER AUFLOESUNG
                        MIT 24 X 24 GRAPHIKPUNK-
                        TEN
 1
                    (Löschen des Bildschirms;
                     Funktion wird gezeichnet)
```



Hardcopy der fertigen Bildschirm-Ausgabe

Optionale PLOTT-Ergänzungsmodule

(1) Möglichkeit zum Zeichnen von zwei Funktionsgraphen:
Durch geringfügige Programmerweiterungen lassen sich auch zwei Funktionsgraphen in einer Ergebnisausgabe zeichnen; so können auf diese Weise z.B. ein Kreis, eine Ellipse oder zwei um 90 Grad phasenverschobene Sinuskurven gezeichnet werden.

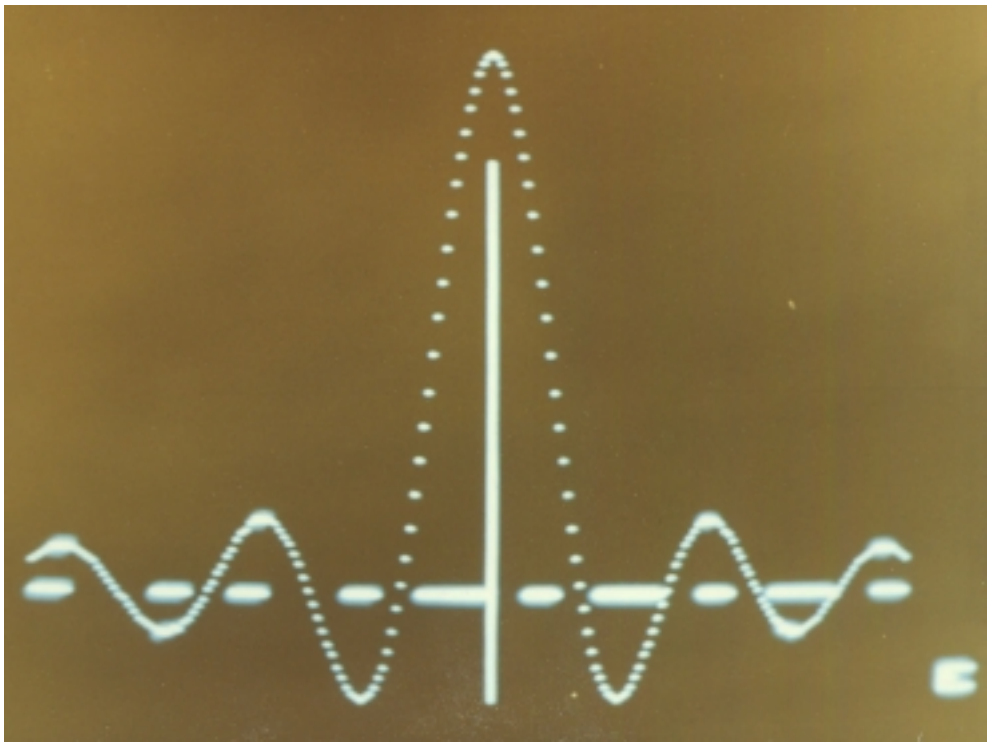
(2) Ergänzende Darstellung von Koordinatenachsen:
Durch geringfügige Programmerweiterungen lassen sich auch - in Abhängigkeit der Definitions- und Wertebereiche - Koordinatenachsen zusätzlich einzeichnen.

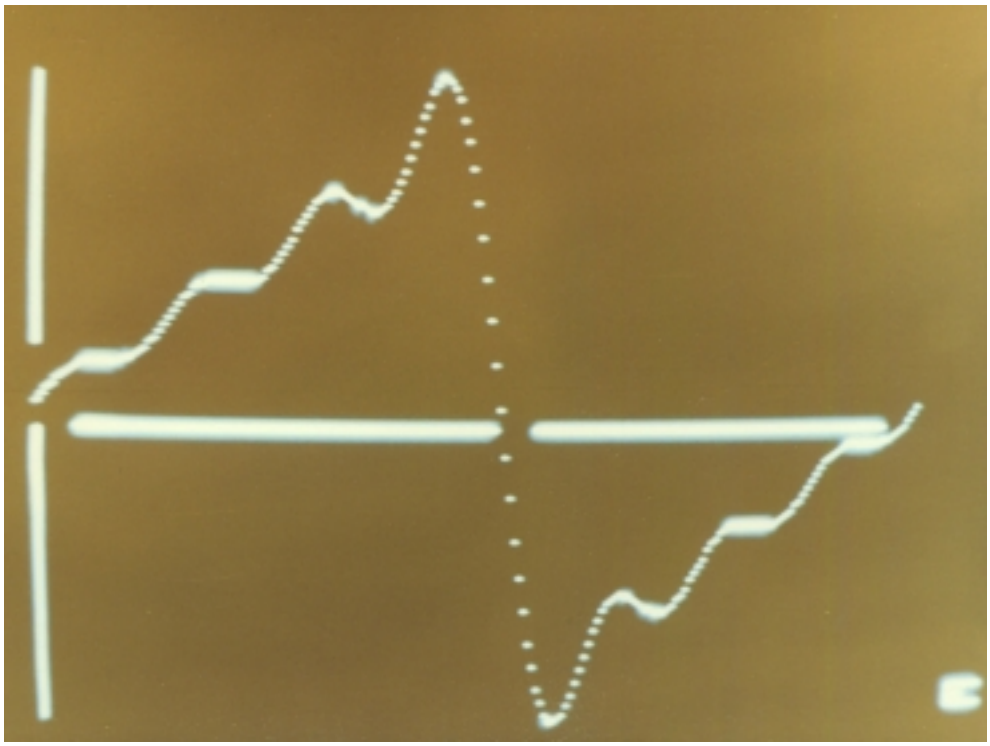
Hinweis zu (1) und (2):
Um „Wischeffekte“ bei Schnittpunkten zweier Funktionsgraphen oder bei Nullstellen und y-Achsen-Schnittpunkten zu vermeiden, werden an diesen Schnittpunkten bzw. –stellen sog. Sprites eingesetzt, deren Zahl allerdings auf 28 beschränkt ist.

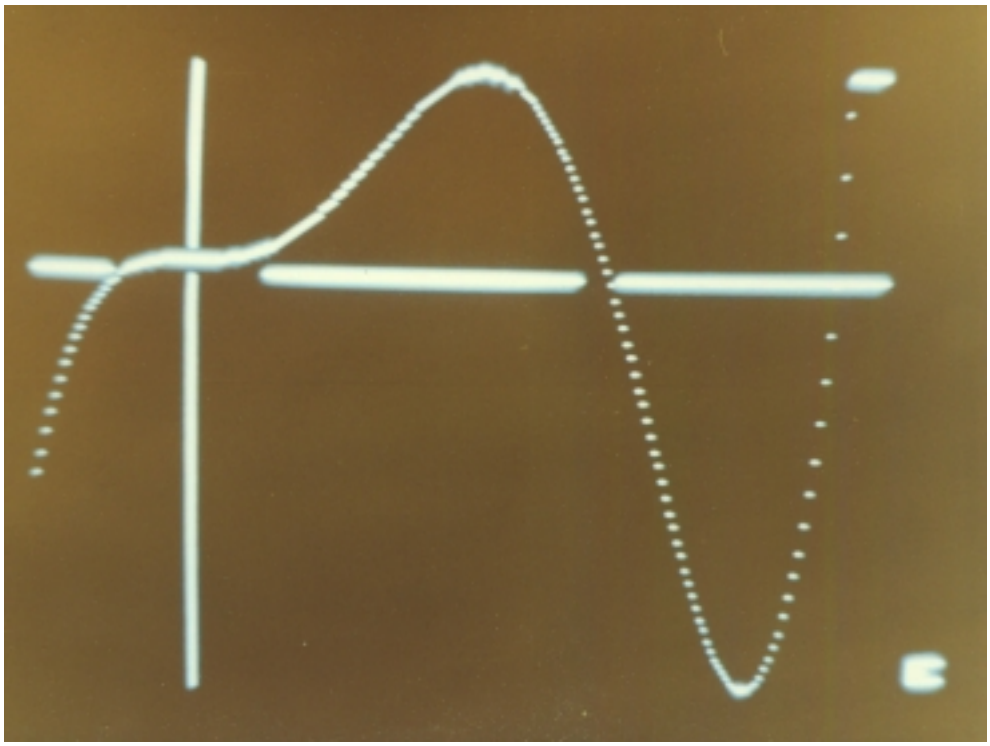
(3) Zusätzliche Druckausgabe auf Matrixdrucker EPSON-FX80

(4) Konvertierung der auf den vorigen Seiten dieser Programmdokumentation beschriebenen TI-Extended-Basic-Version in die TI-Basic-Version:
Näheres hierzu siehe letzte Seite dieser Dokumentation.

Weitere Ausgabebeispiele







==========

Welche Unterschiede weist die TI-Basic-Version des PLOTT-Programms gegenüber der TI-Extended-Basic-Version desselben Programms auf?

==========

- Pre-Scan-Aufhebung entfällt
  Folge: Nach Starten des Programms mit RUN ENTER bzw. RUN 410 ENTER wird das Programm nach möglichen Fehlern abgesucht. Dadurch tritt eine Verzögerung von ca. 20 Sekunden ein.

- Pro Programmzeile ist nur ein Befehl möglich
  (kein Doppelpunkt als Trennzeichen verwendbar)

- Viele Sprünge:
  Die IF-THEN-Anweisungen sind nur in Verbindung mit Sprungadressen Einzusetzen. In TI-Extended-Basic können Wertzuweisungen o.ä. folgen, in TI-Basic muss an eine Stelle im Programm verzweigt werden; dort muss dann z.B. die Zuweisung codiert sein und ggf. sich eine weitere Sprunganweisung anschließen
  Folge: mehr Sprünge, größerer Rechenaufwand, längere Rechenzeiten

- Bei COLOR-Unterprogrammen Satznummern von 1 bis 16
  (in TI-Extended-Basic von 0 bis 14).

- Da die TI-Basic-Version etwas mehr Programmspeicher braucht, wurden zwar in den Listings die REM-Kommentare beibehalten, das PLOTT-Programm ist jedoch ohne diese Kommentare auf Cassette abgespeichert.

- TI-Basic arbeitet (interpretiert) langsamer als TI-Extended-Basic. Dies wird für den Benutzer bemerkbar bei Programmeditierungen, Programmabläufen, Programm-Listings, Datenabspeicherungen.
  Konkrete Beispiele:
  > Zeichnen der Funktion sin x/x im Intervall (-15;15)
  Zeitaufwand bei TI-Basic-PLOTT ca. 7 Minuten
  Zeitaufwand bei TI-Extended-PLOTT ca. 3 Minuten
  > Zeichnen der Funktionen sin x und cos x im Intervall (0;3.5)
  Zeitaufwand bei TI-Basic-PLOTT ca. 12 Minuten
  Zeitaufwand bei TI-Extended-PLOTT ca. 7 Minuten

- Bei auftretenden Fehlerzuweisungen (z.B. bei Null im Nenner oder bei negativer Diskriminante) gibt es in TI-Extended-Basic einige spezielle Befehle wie z.B. ON ERROR oder ON WARNING. Da TI-Basic diesen Vorteil nicht bietet, kann es bei nicht erkannten Polstellen von Funktionen bzw. bei nicht definierten Argumenten Probleme geben. Folgendes kann angezeigt werden:
  Warning Number too big in ... (bei Polstelle)
  Bad Argument in ... (bei nicht definiertem Argument)

- Zusätzliche Unterprogramme (z.B. PLOTT-Ergänzungsmodule) können nicht als MERGE-Unterprogramme abgespeichert/geliefert werden, da der MERGE-Befehl in TI-Basic fehlt.

- Die hier vorliegende PLOTT-Programmdokumentation bezieht sich auf die TI-Extended-Basic-Version von PLOTT. Eine Zusammenstellung der Dokumentationsergänzungen und -korrekturen für die TI-Basic-Version ist ebenfalls vorhanden und beim Autor (F. Kienzler) erhältlich.