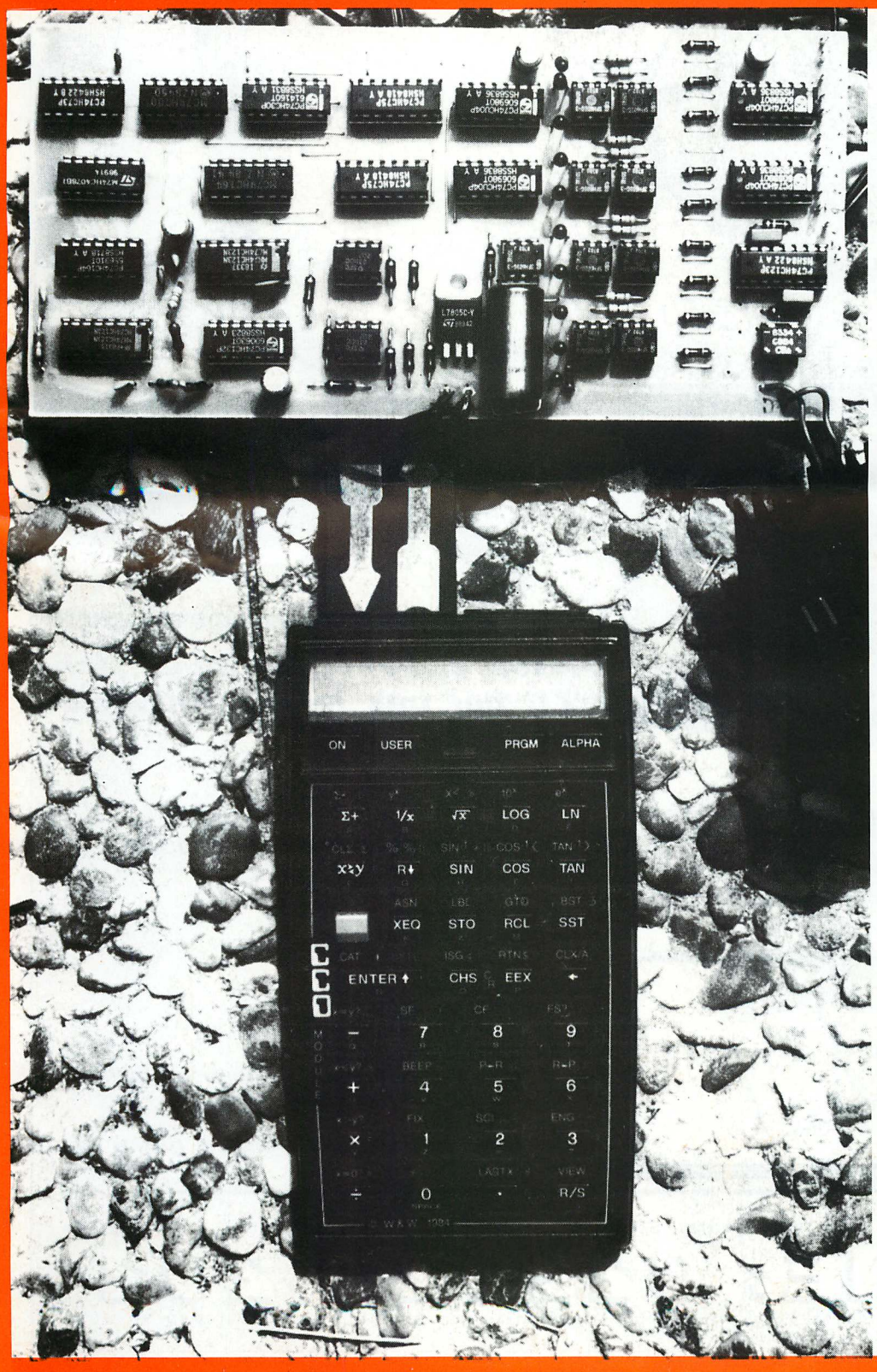
PRISMA

Computerclub Deutschland e.V. · Postfach 11 04 11 · Schwalbacher Straße 50 · D-6000 Frankfurt am Main 1

März / April 1990 Nr. 2 D 2856 F



Das HP-IL-Controlinterface ist eine preiswerte Alternative für Steuerungsaufgaben (Bauanleitung im Heft).

Inhalt

Vorschau

auf die nächsten Hefte

HP-48 Programme

Molmassen beliebiger Verbindungen

Zeichensatz wandeln zwischen
IL-Drucker, ThinkJet und DeskJet

Tastenfeldschablonen auf dem
DeskJet

Elektronischer Einkaufszettel

Haushalt und CCD-Modul

Frischer Wind im CCD

Ein neuer Vorstand, eine neue Zukunft ?

Ich glaube nicht, daß einige wenige neue Personen einen so vielfältig gestalteten und agierenden Club wie den CCD von einem Tag auf den nächsten um 180 Grad drehen können, da kann man getrost alle Zweifel zerstreuen.

Im CCD gehen alle Initiativen, Anregungen und natürlich auch die aktive Mitarbeit von den Mitgliedern selbst aus, nur sie bestimmen über Wohl und Gedeih des Clubs. Kein Vorstand konnte sich bislang dieser massiven Einflußnahme entziehen, d.h. nicht zuschauen lautet die Devise, untergehen wird ein Schiff bekanntlich umso schneller, je mehr Leute zuschauen...

Viele der alten Mitglieder erinnern sich vielleicht noch an die Anfänge des CCD, als die anfängliche Ansammlung von HP41 Begeisterten wahrlich von einer Welle der Kreativität überrollt wurde, die von den scharenweise zuströmenden Mitgliedern ausgelöst wurde. Der Vorstand war damals kein Ruhmesblatt, hatte aber auf die Gestaltung des Clubs und seiner Zeitschrift PRISMA eher geringen Einfluß, die Gestaltung lag bei den Mitgliedern.

Der CCD hat sich in den Jahren seines Bestehens von einem reinen HP41 Anwenderclub hin zu einem HP41 Anwenderclub mit zusätzliche Schwerpunkten im Bereich der PCs, d.h. der sogenannten Personal Computer (MS-DOS kompatible, ATARI und Apple) entwickelt, in den der neue HP48SX wie maßgeschneidert hineinpaßt:
Ein tragbares, kleines, handliches Kraftpaket mit allen nötigen Verbindungen zu den sogenannten PCs, also genau das, was dem "alten" HP41 so gefehlt hat.

Eine neue Fundgrube der Kreativität tut sich da vor uns auf, wir sollten dies nutzen. Auch wenn bereits eine Fülle von fertigen Lösungen in diesem Zwerg stecken, so harrt doch eine breite Palette von Speziallösungen ihrer Verwirklichung.

Die Schnittstelle zum Benutzer ist mit ihrer Grafik und ihren Softkeys *das* Mittel, um bereits existierende Lösungen auf dem HP41/HP71/HP75 und den anderen HP Taschenrechnern wie dem HP28, mit einer dem Fortschritt entsprechenden Bedienungsfreundlichkeit zu versehen, die die Lösung der immer komplexeren Aufgabenstellungen unserer heutigen Zeit ermöglicht.

Mit der Verbindung zu den bei vielen Anwendern inzwischen zum täglichen Werkzeug gehörenden PCs schafft dieser Zwerg ein weiteres großes Feld, in dem sich die Kreativität so richtig schön austoben kann; es gibt viel zu tun, packen wir's an . . .

Die Mitglieder des CCD sind nicht faul und haben bereits, kaum daß der HP48 auf den Markt kam mit seiner Zerlegung begonnen. Beginnt die Kreativität wieder zu sprudeln?
Ich glaube schon. Auch in diesem Heft findet Ihr wieder mehr Details zu diesem "Winzling", einige Programme konnten noch nicht einmal mehr in's Heft kommen, sonst wäre es geplatzt.

Frischer Wind oder Sturm, wir werden es erleben. Die Brücke zwischen den Taschenrechnern und den PCs ist geschlagen.

Martin Meyer
(Redaktion)

Jahreshauptversammlung '90

Liebe Clubfreunde,
auf der letzten Mitgliederversammlung des CCD e.V. in Frankfurt wurde der Vorstand neu gewählt. Wolfgang Fritz und Erich Klee kandidierten nicht mehr für die Funktionen des 1. und 2. Vorsitzenden, weil sie zeitlich durch andere Aufgaben zu sehr belastet sind. Als 1. und 2. Vorsitzender wurden wir, Gerhard Link und Alf-Norman Tietze neu gewählt. Als Schatzmeister arbeitet unverändert Dieter Wolf; als Beisitzer wurden gewählt Werner Dworak und neu Norbert Resch, der in der Regionalgruppe München aktiv ist.

Wir möchten an dieser Stelle Wolfgang Fritz und Erich Klee vor allem den Dank des Vereins für ihre bisher geleistete Arbeit im Interesse unseres Vereines aussprechen. In einer schwierigen Situation hatte sich seinerzeit Wolfgang Fritz bereiterklärt, den Vorsitz im Verein zu übernehmen und ihn in geordnete Bahnen zu lenken. Das ist ihm in Zusammenarbeit mit den anderen Vorstandsmitgliedern hervorragend gelungen, und er hat dafür nicht nur seine großen fachlichen Kenntnisse eingebracht, sondern auch viel Zeit und Mühe aufgewandt.

Durch die bisher vom alten Vorstand geleistete Arbeit sind die Voraussetzungen geschaffen worden, auf denen wir weiter aufbauen können, und wir hoffen dabei auf die Unterstützung aller Mitglieder. Denn unser Verein ist keine Dienstleistungsfirma in dem Sinne, daß einige Vorturner etwas machen, und der Rest der Mitglieder kann dieses dann konsumieren, sondern die teilweise sehr detaillierten fachlichen Kenntnisse vieler unserer Mitglieder sollen allen zu Gute kommen, so daß der Verein eine Organisation zur Verallgemeinerung des in seinen Reihen vorhandenen Wissens darstellt.

In diesem Sinne bitten wir vor allem um Vorschläge aus der Mitgliedschaft für die weitere Verbesserung der Vereinsarbeit. Wir stellen uns vor, daß neben den bestehenden fachlichen Orientierungen weitere Interessengruppen sich bilden, die Anwenderprobleme beraten und lösen, so wie dies in Hinsicht der Ingenieurwissenschaften einer alten Tradition unseres Vereines entspricht. Auch die Regionalgruppenarbeit soll weiter unterstützt und ausgebaut werden. Gerade durch die Regionalgruppen besteht die Möglichkeit, neue Mitglieder an den Verein heranzuführen. Dazu ist es aber auch nötig, in den Regionalgruppen diesen Interessenten etwas zu bieten, was den CCD für sie attraktiv macht. Der Vorstand hat von der Mitgliederversammlung den Auftrag erhalten, ein Info-Paket zur Unterstützung des Aufbaus neuer Regionalgruppen zu erstellen; auch hierfür bitten wir um Vorschläge aus der Mitgliedschaft.

Ein weiteres gilt es zu erwähnen. Unser Verein nennt sich Computerclub Deutschland, und diesem Namen können wir nach der veränderten Situation in Europa und in Deutschland nunmehr auch gerecht werden, während wir bisher ja "nur" ein Computerclub der Bundesrepublik waren. Durch Antworten auf Annoncen in CHIP haben wir festgestellt, daß in der DDR ein großes Interesse an dem Verein besteht. Die Mitgliederversammlung hat daher beschlossen (noch vor Kenntnis der Einzelheiten der Währungs- und Wirtschaftsunion), Interessenten aus der DDR den Beitritt zu erleichtern. Bis 3 Monate nach dem Stichtag der Währungsunion (nach heutiger Kenntnis 2. Juli) wird eine Aufnahmegebühr von neuen Mitgliedern aus der DDR nicht erhoben, und der Beitrag wird bis drei Monate nach dem Stichtag gestundet. Wir glauben, daß der Verein große Möglichkeiten der Unterstützung computerinteressierter Freunde aus der DDR hat, daß aber auch von diesen Freunden gute und wichtige Anregungen kommen können.

Wenn auch die technische Entwicklung in der DDR hinter der im Westen zurückgeblieben war, so hat doch die wissenschaftliche Ausbildung dort ein hohes Niveau. Es erscheint uns nicht als Zufall, daß die verbreitetesten Standardwerke für Mathematik und Ingenieurwissenschaften Lizenzausgaben aus DDR-Produktion sind (z. B. Bronstein/Semendjajew). Und technische Beschränkungen führen ja bekanntlich zu intelligenterer Programmierung, wie man im Vergleich mancher Programme für CP/M und MS-DOS feststellen kann.

In diesem Sinne bitten wir nochmals um die aktive Unterstützung des Vorstandes aus der Mitgliedschaft, um Vorschläge, um Aktivitäten und nicht zuletzt um Werbung für den Verein. Als unsere Aufgabe begreifen wir dabei in erster Linie, diese Aktivitäten der Mitglieder zu unterstützen und zu koordinieren.

Gerhard Link Alf-Norman Tietze

Protokoll

der ordentlichen Mitgliederversammlung des Computerclub Deutschland e.V. vom 21.04.1990

Ort: Intercity Restaurant Frankfurt, Hbf, Frankfurt

Beginn 11.25 Uhr Ende: 18.17 Uhr

Tagesordnung:

1. Begrüßung durch den Vorstand
2. Feststellung der Beschlußfähigkeit und andere Formalitäten
3. Bericht des Vorstandes
4. Bericht des Beirates
5. Bericht der Kassenprüfer
6. Entlastung des Vorstandes
7. Neuwahlen
   a) des Vorstandes
   b) eines Kassenprüfers
   c) ggf. Nachwahlen
8. Haushaltsplan 1990
9. Prisma
10. Anträge
11. Verschiedenes

Zu 1.:
Die erschienenen Mitglieder werden vom 1. Vorsitzenden des Vereins, Herrn Prof. Fritz, begrüßt.

Zu 2.:
Die Beschlußfähigkeit wurde anhand einer Anwesenheitsliste festgestellt (40 anwesende Mitglieder). Der 1. Vorsitzende wies darauf hin, daß u.U. einzelnen Mitgliedern die Einladung zur Mitgliederversammlung im Heft Prisma 1/90 nicht rechtzeitig gemäß Satzung zugegangen sein könnte. Die anwesenden Mitglieder erklärten ausnahmslos per Handzeichen, daß sie hieraus keinerlei Rechte ableiten werden.

Eine Ergänzung der Tagesordnung wurde nicht gewünscht. Gemäß § 14 I Satz 3 wurde ein aus den Mitgliedern Hansmann, Markscheffel und Florstedt bestehender Wahlausschuß gebildet.

Zu 3.:
Der 1. Vorsitzende berichtete über das Vereinsgeschehen und die Entwicklung des Vereins seit der letzten Mitgliederversammlung sowie über Möglichkeiten künftiger Entwicklung.

Der Schatzmeister berichtete über die finanzielle Situation des Vereins, erläuterte den Abschluß 1989 und den Haushaltsvorschlag für 1990, welcher bei vier Enthaltungen ohne Gegenstimme beschlossen wurde.

Zu 4.:
Das Beiratsmitglied Meier berichtete über die Tätigkeit des Beirats; er erwähnte dabei insbesondere die gute und integrative Zusammenarbeit zwischen Beirat und Vorstand.

Zu 5.:
Der Bericht des Kassenprüfers, vorgetragen von dem Mitglied Link, bescheinigte dem Vorstand eine ordnungsgemäße Buchführung; die Belege für Ausgaben und Einnahmen seien korrekt vorgefunden und bei Bedarf zufriedenstellend erläutert worden. Der Kassenprüferbericht schloß mit dem Antrag, dem Vorstand Entlastung zu erteilen.

Zu 6.:
Auf Antrag der Kassenprüfer wurde dem Vorstand ohne Gegenstimmen bei zwei Enthaltungen Entlastung erteilt.

Zu 7.:
Gemäß Satzung standen turnusgemäß Neuwahlen des Vorstandes an

a) Für das Amt des 1. Vorsitzenden kandidierten die Mitglieder Gerhard Link und Norbert Resch. Gewählt wurde mit 30 Stimmen das Mitglied Gerhard Link, auf das Mitglied Norbert Resch entfielen 5 Stimmen; 2 Mitglieder enthielten sich der Stimme.

   Für das Amt des 2. Vorsitzenden kandidierten die Mitglieder Alf-Norman Tietze und Norbert Resch. Gewählt wurde mit 24 Stimmen das Mitglied Alf-Norman Tietze; auf das Mitglied Resch entfielen 8 Stimmen. 3 Mitglieder enthielten sich der Stimme.

   Für das Amt des Schatzmeisters kandidierte lediglich das Mitglied Dieter Wolf, welches mit 36 Stimmen bei einer Enthaltung gewählt wurde.

   Für das Amt des 1. Beisitzenden kandidierten die Mitglieder Norbert Resch und Werner Dworak. Gewählt wurde mit 23 Stimmen Herr Resch; auf Herrn Dworak entfielen 13 Stimmen. 2 Mitglieder enthielten sich der Stimme.

   Für das Amt des 2. Beisitzer kandidierten die Mitglieder Martin Meier und Werner Dworak. Gewählt wurde das Mitglied Werner Dworak mit 22 Stimmen; auf das Mitglied Meier entfielen 12 Stimmen; 4 Mitglieder enthielten sich der Stimme.

b) Anstelle des zum 1. Vorsitzenden gewählten Kassenprüfers Gerhard Link wurde als einziger Kandidat ohne Gegenstimme das Mitglied Reinhard Blum zum weiteren Kassenprüfer gewählt.

c) Weitere Nachwahlen waren nicht erforderlich.

Zu 8.:
Der Haushaltsplan 1990 wurde beschlossen.

Zu 9.:
Es wurde darüber diskutiert, ob für Veröffentlichungen im Prisma den Autoren Honorare bezahlt werden sollen.

Es wurde Übereinstimmung dahingehend erzielt, daß echte Honorare, die eine reale Gegenleistung für die Arbeit der Autoren darstellen, nach wie vor nicht bezahlt werden sollen.

Betreffend die Einsendung von Artikeln wurde seitens der Prisma-Redaktion gebeten, in maschinenlesbarer Form übermittelte Artikel in Form reiner ASCII-Files bereitzustellen.

Zu 10.:
Auf Antrag aus der Mitgliederversammlung wurde bei zwei Gegenstimmen und zwei Enthaltungen beschlossen, daß Mitgliedsbeiträge eines laufenden Jahres jeweils am 15. Februar diesen Jahres fällig sind. Die Beiträge säumiger Mitglieder sollen durch den Justitiar beigetrieben werden.

Auf Antrag des Mitglieds Wolfgang Fritz wurde bei einer Gegenstimme und einer Enthaltung folgendes beschlossen:
Der CCD e.V. trifft für Bürger der DDR, die in der DDR wohnen, folgende Sonderregelung, die drei Monate nach dem Stichtag einer Währungsunion mit der Bundesrepublik Deutschland, spätestens aber mit der Mitgliederversammlung 1991 ausläuft:

1. Auf die Aufnahmegebühr wird verzichtet.
2. Die Bezahlung (in DM) des Mitgliedsbeitrages wird bis drei Monate nach dem Stichtag der Währungsunion gestundet.

Ein weiterer Antrag des Mitglieds Wolfgang Fritz, wonach Sonderbeiträge für die Mitgliedschaft in den Diskettengruppen voraus zu entrichten sein sollen, wurde mit deutlicher Mehrheit abgelehnt.

Zu 11.:
Es stellten sich zwei Gäste des AUGE e.V. vor und schlugen eine Zusammenarbeit der beiden Vereine vor, soweit diese die gemeinsamen Interessen fördere.

Der Vorstand wurde beauftragt, in Gesprächen mit dem Vorstand des AUGE e.V. die Möglichkeiten einer solchen Zusammenarbeit zu prüfen.

Es wurde sodann die Möglichkeit der Förderung der Regionalgruppenarbeit beraten, mit dem Ergebnis, daß allgemeine Zusagen des Vorstandes für Raumkostenübernahme nicht möglich sei, dies vom Vorstand aber im Einzelfall überprüft und entschieden werde.

gez. Widl (Justitiar)

Ortstreffen in Köln

In Köln, im Vereinshaus der Humbolt-Siedlung, organisieren wir aus Köln CCD-Treffen. Diese Treffen finden in unregelmäßigen Abständen alle 3-4 Monate statt. Ich möchte in diesem kurzen Artikel beschreiben, wie es auf diesen Treffen zugeht, für diejenigen im CCD, die an diesen Treffen bisher nicht teilgenommen haben, sei es daß sie bisher nichts davon erfahren haben, oder aber weil sie zu weit von Köln weg wohnen.

Das CCD-Treffen findet in zwei benachbarten Räumen statt, in denen sich mehrere Tische befinden, auf denen Computer samt Zubehör, aufgebaut werden können. Ganz früher gab es nur CPM Rechner, später CPM und MSDOS Rechner und nun fast nur noch MSDOS Rechner und vereinzelte Atari's (bis zu 2).

Inhaltlich wurden bei den letzten beiden Treffen folgende Themen bearbeitet:
Dauerbrenner ist die Datenfernübertragung. Hierzu kann an einem Rechner der Umgang mit BTX oder mit der MEDIA-BOX Köln erprobt werden.

Beim Treffen kurz vor Weihnachten 1989 wurden die ersten SCANNER mitgebracht, und man konnte ausprobieren, wie sich eine Graphik nach dem Scannen mit einem Handscanner verändert, oder sehen, welche Schwierigkeiten Programme zur Texterfassung nach dem Scannen von Textvorlagen auf einem Flachbettscanner haben.

Jürgen's Steckenpferd ist Datensicherheit. Er zeigt die neusten Versuche, wie man Digitale Information sicher verschlüsseln kann. Er zeigt auch, welche Programme für die Datensicherheit nichts taugen, weil sie sehr einfach zu umgehen sind.

Da kaum zwei Besucher den gleichen Computer mitbringen, kann man die verschiedensten Modelle ansehen und von den leidvollen Erfahrungen der Besitzer hören, wenn diese von den Erweiterungen berichten, die fast funktioniert hätten, wenn nicht usw. Ein regelmäßiger Wettbewerb besteht darin, den Laptop mit dem unleserlichsten Display auf den Tisch zu bringen. Nur einer sieht, was auf dem Rechner überhaupt läuft.

Und über den CCD wird immer heftig diskutiert. Auf Details möchte ich hier nicht eingehen. Ich persönlich finde es gut, daß nicht zuletzt durch den CCD diese regionalen Treffen ermöglicht werden.

Nichtmitglieder (oder eventuelle Neumitglieder) dürfen zu Besuch auf diesen Treffen erscheinen und können unverbindlich die Disketten des CCD's begutachten und die letzten Prisma-Hefte lesen.

Computeranfänger werden von allen Seiten mit den verschiedensten guten Ratschlägen versehen. Computerbesitzern mit Problemen bei Hard- und Software kann (manchmal) geholfen werden, oder sie erfahren, warum bestimmte Kombinationen so nicht funktionieren können.

Zum Schluß noch an eine Bitte an diejenigen CCD-Mitglieder, die eine schriftliche Einladung zu diesen CCD-Treffen erhalten möchten. Sendet eine Postkarte entweder an Jürgen Schramm, Frankfurterstr. 855, 5000 Köln 91 oder an mich (siehe unten).

Ach ja, Getränke gibt es zum Selbstkostenpreis. Gegen Abend wird etwas zu Essen bestellt (auf eigene Kosten versteht sich).

Bis Bald, in Köln.

Werner Müller
Schallstr. 6
5000 Köln 41
MBK1:W.MUELLER

P.S.
Wir (Jürgen Schramm und ich) hoffen, daß durch die Einführung der Rubrik Termine die Kölner Termine rechtzeitig im Prisma angekündigt werden. Wer kein Auto hat, könnte sich bei Jürgen oder mir melden, vielleicht läßt sich die eine oder andere Fahrgemeinschaft organisieren.

Nächstes

Ortstreffen in Köln:

am 16. Juni 1990, ab 13 Uhr im Vereinshaus der Humboldt-Siedlung, Frankfurter Straße 855, 5000 Köln 91 (Ostheim)

Verantwortlich und Ansprechpartner:
Dr. Werner Müller,
Schallstraße 6,
5000 Köln 41,
Tel. (0221) 40 23 55

und

Dipl.-Ing. Jürgen Schramm,
Frankfurter Straße 853,
5000 Köln 91,
Tel (0221) 8 90 23 54.

C- Programme - ein Überblick

Dritter Teil

Memory- Modelle, Vektoren und Pointer von C

Bereits im zweiten Teil der Serie haben wir darauf hingewiesen, daß für das Memory-Management unter C Kenntnisse der Hardware unabdingbar sind. Dies hat große Bedeutung für Computer auf Basis der Intel-Prozessorenreihe 80x86, aber auch teilweise für größere UNIX-Maschinen. Bei anderen Computern, wie z.B. dem ATARI ST, gelten diese Besonderheiten nicht oder nur am Rande. Dennoch ist die Kenntnis dieser Umstände generell für C-Programmierer von Interesse, denn sie bestimmen sehr stark die Portabilität von C-Programmen.

1. Memory-Modelle für Intel 80x86-Prozessoren

Die Betrachtungen gelten, und das ist wesentlich, für die Betriebssysteme DOS, OS/2 in seiner jetzigen Fassung als 16-Bit System, aber auch für 286-er UNIX-Versionen, wie XENIX 286 oder Microport System V für IBM-AT. Bekanntlich weisen Intel-Prozessoren der o.g. Baureihe eine sogenannte Segmentstruktur auf: Der adressierbare Speicher für Code und Daten ist in Einheiten, eben Segmente unterteilt. Der Gegensatz dazu sind Prozessoren mit sog. linearem Adressraum, z.B. Motorola-Typen ab 68000 und teilweise 80386.

Die Segmentstruktur, die dem Programmierer viel Kopfzerbrechen bereiten kann, hat eigentlich historische Gründe. Der Prozessor Intel 8086 als Urvater der Baureihe ist ein 16-Bit Prozessor. Mit 16-Bit-Registern kann man 2^{16}= 65536 Byte= 64 KByte adressieren. Da der 8086 als 16-Bit Nachfolger der 8-Bit Prozessoren Intel 8080 bzw. Zilog Z80 dienen sollte, war eine hohe Portierungsfähigkeit von 8-Bit CP/M Programmen anzustreben. CP/M war bis in die frühen 80er Jahre das dominante Mikrocomputer-Betriebssystem. So entsprechen z.B. die Register AL, BX, CX, DX, SI und DI des 8086 den Registern A, HL, BC, DE, IX und IX des Z80, lediglich der Akku des 8086 ist als AX auch 16 Bit lang. Und es sind die "halben" Register (8 Bit) AL, AH, BL, BH usw. verfügbar.

In der Tat konnten so bewährte CP/M-Programme wie WordStar oder VisiCalc offensichtlich sehr einfach portiert werden, was sicher den Erfolg des IBM-PC von Anfang an sicherte. Auf der anderen Seite sollte spürbar mehr Gesamtspeicher als 64 KB bereitgestellt werden: 1 MByte. Dazu sind dann 2^{20}, d.h. 20-Bit Adressen zu nutzen (der 8086 hat in der Tat 20 Adressleitungen).

Um nun diesen Raum zu nutzen, muß ein Trick angewendet werden: Es werden Speichersegmente und entsprechende Segmentregister eingeführt. Mit eine gewöhnlichen Register oder einer 16-Bit-Konstanten wird ein bis zu 64 KByte großer Offset innerhalb eines Segments angegeben und durch ein Segmentregister wird die Startadresse des Segments in Schritten von 16 Byte festgelegt.

Es gibt vier Segmentregister, CS für Codesegment, DS und ES für Datensegment und SS für Stacksegment. Die physikalische Adresse bekommt man beim 8086 (und beim 80286 und 80386 im Real Mode) durch Multiplizieren des Segmentregisters mit 16 und Addieren des Offsets (Tabelle 1). Die selbe physikalische Adresse läßt sich durch viele Kombinationen von Segment und Offset erreichen.

Typische Registerkombinationen sind:

CS:IP Zeiger auf nächsten Programmschritt,
CS= Code-Segmentregister,
IP= Instruction-Pointer

```
xxxx xxxx xxxx xxxx 0000   Segmentregister mal 16
+    xxxx xxxx xxxx xxxx   Offsetregister
xxxx xxxx xxxx xxxx xxxx   20- Bit Adresse
```

Tabelle 1

Listing 1 (b35.c)

```
#include <stdio.h>
#define MAXELE 8000L

void fmul2(long nektor[], long max);
                /* function- deklaration */

int main(void)
{
long nektor[MAXELE];
long i,max;

max= MAXELE;
for(i= 0L; i < max; i++) nektor[i]= i;

fmul2(nektor,max);

printf("\ndas groesste Element
    nektor[%ld] = %ld",max-1L,nektor[max-1L]);
return 0;
}

void fmul2(long nektor[],long max)
{
long i;

for(i= 1L; i < max; i++) nektor[i] *= 2L;
                /* mit 2 multiplizieren */
return;
}
```

Listing 2 (b36.c)

```
#include <stdio.h>

long ladd(void); /* function- deklaration */

int main(void)
{
printf("\n1. Static long i= %ld", ladd());
printf("\n2. Static long i= %ld", ladd());
printf("\n3. Static long i= %ld", ladd());
return 0;
}

long ladd(void)
{
static long i;
i++;
return i;
}
```

Listing 3 (b37.c)

```
#include <stdio.h>

#define MAXELE 16000L

long nektor[MAXELE]; /* extern */
long max= MAXELE;    /* extern */

void fmul2(void); /* function- deklaration */

int main(void)
{
extern long nektor[];
extern long max;
long i;

for(i= 0L; i < max; i++) nektor[i]= i;
fmul2();
```

```
printf("\ndas groesste Element
     nektor[%ld]= %ld",max-1L,nektor[max-1L]);
return 0;
}

void fmul2(void)
{
extern long nektor[];
extern long max;
long i;

for(i= 1L; i < max; i++) nektor[i] *= 2L;
                  /* mit 2 multiplizieren */
return;
}
```

Listing 4 (b38.c)

```
#include <stdio.h>

#define MAXELE 100000L

long huge nektor[MAXELE]; /* extern */
long max= MAXELE;         /* extern */

void fmul2(void); /* function- deklaration */

int main(void)
{
extern long huge nektor[];
extern long max;
long i;

printf("\n%ld Werte mit 2 multiplizieren..",
                                        max);
for(i= 0L; i < max; i++) nektor[i]= i;
fmul2();

printf("\ndas groesste Element
     nektor[%ld]= %ld",max-1L,nektor[max-1L]);
return 0;
}

void fmul2(void)
{
extern long huge nektor[];
extern long max;
long i;

for(i= 1L; i < max; i++) nektor[i] *= 2L;
                  /* mit 2 multiplizieren */
return;
}
```

Listing 5 (b39.c)

```
#include <stdio.h>

#define MAXELE 100000L

long huge nektor[MAXELE]; /* extern */
long max= MAXELE;         /* extern */

int main(void)
{
extern long huge nektor[];
extern long max;
long i;
printf("\nVektor nektor mit i= 1..%ld
                        belegen..",max);
   for(i= 0L; i  max; i++) nektor[i]= i;

printf("\ndas groesste Element
     nektor[%ld]= %ld",max-1L,nektor[max-
1L]);
return 0;
}
```

SS:SP Zeiger auf Stack,
SS= Stacksegement,
SP= Stackpointer

DS:BX Zeiger auf Daten,
DS= Datasegement,
BX= Allzweckregister
(kann auch ein anderes sein)

Damit ist folgendes klar: Sind Programme mit mehr als 64 KByte Code und/oder 64 KByte Daten zu versehen, dann müssen die Segmentregister CS und/oder DS entsprechend nachgeführt werden. Dies besorgen die jeweiligen Sprachcompiler .. oder auch nicht!

Bei manchen Programmiertools, wie z.B. diversen BASIC-Interpretern werden die Segmentregister CS und DS niemals innerhalb des Programmlaufs verändert, somit können derartige Interpreter oder Compiler nur 64 KByte Code und 64 KByte Daten adressieren. Dies galt auch z.B. für Turbo-Pascal 3.0. Ob nun die beiden genannten Segmentregister nachgezogen werden (was natürlich aufwendiger ist als sie fix zu lassen), hängt vom sog. Memory-Modell ab:

Tiny: Code und Daten zusammen 64 KByte

Small: Code 64 KByte, Daten 64 KByte

Medium: Code mehr als 64 KByte, Daten 64 KByte

Compact: Code 64 KByte, Daten mehr als 64 KByte

Large: Code und Daten mehr als 64 KByte, aber pro Array nur 64 KByte

Huge: wie Large, aber einzelne Arrays können mehr als 64 KByte Daten haben

Der Stack ist übrigens immer auf 64 KByte beschränkt, ausgenommen 80386 Protected Mode. Damit müssen wir uns noch beschäftigen.

Diese Memory-Modelle gelten auch für den Intel 80286 im protected-Mode (im Real-Mode sowieso) für die Betriebssysteme OS/2 und 286-UNIX, nur ist durch die virtuelle Adressierung keine Begrenzung auf 20-Bit Adressen gegeben. Was bedeuten Real- Mode und Protected-Mode? Im Real-Mode ist der 80286 kompatibel mit dem 8086, d.h. die Adressen werden w.o. beschrieben als reale Maschinenadressen mit Segment und Offset bestimmt. Im Protected-Mode entfaltet der 80286 seine wahren Stärken: Die Adressen werden als virtuelle Adressen mit Selector und Offset bestimmt. Der Selector zeigt auf eine Descriptor-Table, die das Betriebssystem verwaltet. Erst dort werden aus virtuellen Adressen dann reale Maschinenadressen gebildet aus (intern gespeicherter) Startadresse des Segments und dem Offset. Da die Verwaltung der virtuellen Adressen nicht mehr durch die Programme selbst, sondern durch das Betriebsystem vorgenommen wird, können die Code- und Datenbereiche von parallel laufenden Programmen (was bei OS/2 und UNIX ja möglich ist) voneinander abgeschottet, geschützt werden.

Diese Aussagen gelten wiederum auch für den großen Bruder des 80286, den 80386. Mit einem Unterschied: Der 80386 arbeitet nicht wie der 8086 und der 80286 mit 16-Bit Registern, sondern mit 32-Bit Registern. Mithin können pro Segment nicht nur 64 KByte, sondern bis zu 4 GByte verwaltet werden. Dies führt zu den völlig falschen Aussagen mancher Autoren, daß der 80386 nicht segmentiert sei, sondern einen linearen Adressraum habe. Es ist lediglich so, daß entsprechend große Segmente (innerhalb eines Segments wird immer linear adressiert) zum heutigen Zeitpunkt kein Nachführen der Selektoren erfordern. Prinzipiell können obige Memory-Modelle auf den 80386 übertragen werden. Die Segmentgröße ist jedoch variabel (bis zu 4 GByte).

Es ist also je nach Programmanforderungen hinsichtlich der Größe von Code- und Datenbereich das geeignete Memory-Modell zu wählen. Nicht alle C-Compiler beherrschen die volle Spanne aller denkbaren Memory-Modelle. Manche Compiler lassen nicht das Modell Tiny zu (das ist nicht schlimm), andere vertragen nicht Large und Huge. Wieder andere erlauben vordergründig Huge, also Vektoren mit mehr als 64 KByte, aber nur

für Character- und Integer-Vektoren. Auch das kann zu erheblichen Portierungsproblemen führen, da die Memory-Modelle kein Standard sind und völlig compilerabhängig sind.

```
long huge x[50000]      : ein Vektor mit 200000 KByte,
                          z.B. für DOS
double huge y[1000000]  : ein Vektor mit 8 MByte, geht bei OS/2

long huge *pi           : ein huge Pointer auf Integer
double huge *px         : ein huge Pointer auf Double
```

Achtung: Das Schlüsselwort huge ist vor jeder Variable zu benutzen, also so:

```
double huge x[50000], huge y[50000];
double huge *px, huge *py;
```

nicht so:

```
double huge x[50000], y[50000];
double huge *px, *py;
```

y und *py sind nun nicht huge

Betrachten wir dahingehend rein beispielhaft den C-Compiler von Microsoft, der gern auf PCs verwendet wird. Beim Setup des Compilers kann gewählt werden, welches Memory-Modell standardmäßig genutzt wird: Als Vorgabe wird Small vorgeschlagen. Natürlich kann damit der kompakteste Code bei kürzester Laufzeit erreicht werden, da keine Segmentregister nachgeführt werden müssen. Ich empfehle, hier standardmäßig das Large-Memory-Modell vorzusehen: Die Laufzeitunterschiede sind bei normalen Anwendungsprogrammen (nicht bei irgendwelchen Benchmarks, die man künstlich auf kürzeste Laufzeit trimmen will) vernachlässigbar. Umgekehrt reicht oftmals für z.B. technisch-wissenschaftliche Programme das Small-Memory Modell nicht mehr aus. Man ist mit dem Large-Modell einfach freier. Da bei derartigen Anwendungen nur einige wenige Vektoren größer als 64 KByte sind (dann aber oft richtig groß, mehrere hundert KByte), deklariert man solche Vektoren und Pointer dann mit dem Schlüsselwort huge:

Daneben gibt es die Schlüsselworte far (ein solcher Vektor bzw. ein solcher Pointer erhält zwar eine eigene Segmentadresse, die aber nicht nachgeführt wird: max. 64 KByte, z.B. für Vektoren bis jeweils 64 KByte Größe in Small- und Medium-Memory-Modellen) und near (nur 16-Bit Offsetadresse). Mithilfe dieser Schlüsselworte huge, far und near entstehen dann sog. mixed- Memory-Modelle.

Um diese Feinheiten, über die ein Beginner 100% stolpert, nicht zu beachten: Standardmäßig Large-Memory-Modell, sehr große Vektoren und Pointer auf sehr große Datenbereiche mit huge deklarieren. Man compiliert dann bei MS-C wie folgt unter DOS und OS/2:

```
cl /AL beispiel.c                          : compilieren und linken
cl /c /AL beispiel.c                       : nur compilieren
cl /AL beispiel.c func.c func2.c           : mehrere Module compilieren,
                                             linken
cl /c /AL beispiel.c func1.c func2.c       : nur compilieren mehrerer
                                             Module
als Variante:
cl /AL beispiel.c func1.obj func2.c        : compilieren von beispiel
                                             und func2,mit bereits
                                             compiliertem func2 linken
```

Um nochmal auf das Setup des MS-C Compilers einzugehen: Beim Einrichten wird abgefragt, wie der Coprozessor 80x87 zu behandeln ist: garnicht, optional oder zwingend verwenden. Es gibt hier verschiedene Varianten. Die mit Abstand schnellste ist zwingende Coprozessorverwendung mit Inline-Code (der Code für den 80x87 wird direkt in den Objekt-Code eingefügt). Dazu ist der Switch /FPi87

Listing 6 (b40.c)

```
/* programm b40.c fuer UNIX System V      */
/* entspricht funktionell b38.c           */
/* mit cc -ob40 b40.c compilieren/linken */

#include "/usr/include/stdio.h"

#define MAXELE 1000000L

long nektor[MAXELE];        /* extern */
long max= MAXELE;           /* extern */

main()
{
extern long nektor[];
extern long max;
long i;

printf("\n%ld Werte mit 2 multiplizieren..",
                                        max);
for(i= 0L; i < max; i++) nektor[i]= i;
fmul2();

printf("\ndas groesste Element
   nektor[%ld]= %ld", max-1L,nektor[max-1L]);
return 0;
}

fmul2()
{
extern long nektor[];
extern long max;
long i;

for(i= 1L; i < max; i++) nektor[i] *= 2L;
                /* mit 2 multiplizieren */
return;
}
```

Listing 7 (b41.c)

```
#include <stdio.h>
#include <alloc.h>

int main(void)
{
long huge *pnektor;
long max;
long i;

/* Speicher dynamisch anfordern */

printf("\nWieviel Elemente soll der
                        Vektor enthalten:");
scanf("%ld",&max);
pnektor=(long huge *)halloc(max,sizeof(long));
if(pnektor != NULL)
   printf("\nVektor erfolgreich angelegt..");
else
   {
   printf("\nVektor nicht angelegt ..stop\n");
   return 1;
  }
 printf("\nVektor nektor mit i= 1..%ld
                          belegen..",max);
 for(i= 0L; i < max; i++) *(pnektor+i)= i;

printf("\ndas groesste Element
                        nektor[%ld]= %ld",
max-1L,*(pnektor+max-1L));
return 0;
}
```

Listing 8 (b42.c)

```
#include <stdio.h>
#include <malloc.h>

int main(void)
{
double *pvektor;
long lmax;
int max;
int i;

/* Speicher dynamisch anfordern */

printf("\nWieviel Elemente soll der Vektor
                                  enthalten:");
scanf("%ld",&lmax);
if(lmax < 64000L/sizeof(double))
  {
  printf("\nmax ist zulaessig..");
  max= (int) lmax;
  }
else
  {
  printf("\nmax ist zu gross..stop");
  return 1;
  }
pvektor=(double *)calloc(max,sizeof(double));
if(pvektor != NULL)
  printf("\nVektor erfolgreich angelegt..");
else
  {
  printf("\nVektor nicht angelegt ..stop\n");
  return 1;
  }

printf("\nVektor vektor mit i= 1..%d
                              belegen..",max);
  for(i= 0; i < max; i++) *(pvektor+i)=
                                  (double)i;
printf("\ndas groesste Element
                            nektor[%d]= %lf",
max-1,*(pvektor+max-1));
return 0;
}
```

Listing 9 (b43.c)

```
#include <stdio.h>

int main(void)
{
static  double  hektor[10]= {10.,20.,30.,40.,
          50.,60.,70.,80.,90.,100.};
static int      iektor[10]= {0,1,2,3,4,5,
                                  6,7,8,9};
static int      jektor[10]= {9,8,7,6,5,4,
                                  3,2,1,0};
int    i;

do
  {
  printf("\ngib i (von 0..9): ");
  scanf("%d",&i);
  }
while( i < 0 || i > 9);

printf("\nhektor[iektor[jektor[i]]]= %lf\n",
hektor[iektor[jektor[i]]]);
return 0;
}
```

beim Compilieren anzugeben:
cl /AL /FPi87 beispiel.c : gibt kompaktesten, schnellsten Code, wenn Gleitkomma-Operationen im größeren Stil gefragt sind.

Dazu mußte beim Setup die C-Library LLIBC7.LIB erzeugt werden (das geht automatisch, man muß es nur dem Setup-Programm sagen). Eine Alternative ist LLIBCE.LIB, die Emulator-Library: Wenn im System ein Coprozessor vorhanden ist, wird er genutzt, sonst wird er softwaremäßig durch LLIBCE emuliert. Dazu muß beim Compilieren kein Switch angegeben werden (es wird standardmäßig der Switch /FPi genutzt).

Bei der Programmentwicklung sind immer folgende Switches gut
/Od : Disable Optimizing
/W3 : höchster Warning-Level
Also
cl /c /AL /Od /W3 beispiel.c, cl /AL /FPi87 /Od /W3 beispiel.c usw. Die Reihenfolge der Switches ist egal.

Bei Turbo-C in der integrierten Entwicklungsumgebung werden diese Angaben über Menüs gesteuert.

Diese Betrachtung gilt nicht nur für DOS,sondern auch für OS/2 und 286-er UNIX-Systeme, da die Adressbildung mit Segment- und Offsetregister (bzw. Selector und Offset) prozessorspezifisch und nicht betriebssystemspezifisch ist. Sie gilt ferner auch für andere Programmiersprachen: MS-FORTRAN läßt die Wahl zwischen Medium- und Large-Memory-Modell (Standard ist Large). Auch hier müssen größere Vektoren mit FAR bzw. HUGE deklariert werden. Will man das vermeiden, um nicht in Portabilitätsprobleme zu laufen, deklariert man derartige Vektoren als COMMON- Blöcke, z.B. DOUBLE PRECISION X(100000) COMMON /CX/ X, damit steht X in einem gelabelten COMMON-Block namens CX.

Andere Compiler wie Lahey-FORTRAN 77 operieren standardmäßig mit Huge-Memory-Modellen.

Was hat dieser Exkurs in einer C-Übersicht verloren? Sehr viel, denn in C gibt es Äquivalentes, das intern ebenso arbeitet. Kennzeichnet man eine Variable bzw. einen Vektor als COMMON, so wird er nicht im DATA-Segment (das 64 KByte nicht überschreiten kann) eingelagert, sondern erhält Segmente nur für sich. Mit der Deklaration far in C oder FAR in FORTRAN erhält er ein Segment, mit huge/HUGE oder COMMON beliebig viele Segmente.

Normalerweise verwendet man solche COMMON-Blöcke zum Datenaustausch zwischen Hauptprogramm / Unterprogrammen bzw. Unterprogrammen / Unterprogrammen, aber das muß man nicht. Man kann dieses Feature auch verwenden, um sehr große Speicherflächen anzulegen. Ein ähnliches Feature gibt es in C, die Speicherklasse "extern". Damit sind wir bei

2. C-Speicherklassen

Betrachten wir ein kleines C-Programm (Listing 1, b35.c).

Hier wird ein Vektor mit 8000 long-elementen definiert, belegt und ganz C-typisch an eine function übergeben. Da der Vektor nektor innerhalb von main definiert ist und ohne weitere Schlüsselworte deklariert ist, ist er der Speicherklasse "auto" zugehörig: Die Klasse "auto" bezeichnet lokale Variable, deren Speicherplatz auf dem Stack angelegt wird. Damit sind wir wieder beim Thema Memorymodelle: Wie wir wissen, kann der Systemstack nie größer als 64 KB sein. Für die Speicherklasse "auto" werden sogar nur 32 KB erlaubt. Mithin sind wir mit unseren 8000 x 4 = 32000 Bytes ohnehin praktisch an der Grenze. Übrigens sind im Hauptprogramm max und i ebenfalls "auto", desgleichen i in function fmul2. Normalerweise sind die Stacks, wenn von MS-C oder MS-FORTRAN angelegt, nur 2 KB groß. Es muß obiges Beispiel dann so übersetzt und gelinkt werden:
cl /c /Od /AL /FPi87 /W3 b35.c
link /STACK:50000 b35;
Der Stack wurde sicherheitshalber gleich auf 50000 Bytes gesetzt. Es ist klar ersichtlich, daß mit diesem Vorgehen keine großen Arrays angelegt werden können. Die Speicherklasse "auto" wird typi-

scherweise für Skalare und sehr kleine Vektoren angewendet. Eine Variable, innerhalb einer Funktionseinheit mit dem Schlüsselwort "auto" oder ohne Schlüsselwort einer Speicherklasse, d.h. auto, ist nicht initialisiert, lebt nur innerhalb der Funktionseinheit und stirbt nach Verlassen der Funktionseinheit. Mit Funktionseinheit sind gemeint: functions, main, aber auch Blockstrukturen innerhalb von functions/ main.

Die Speicherklasse "static" bildet eigentlich das Gegenstück zu "auto". Eine mit static deklarierte Variable behält ihren Wert bei, auch nach Verlassen der Funktionseinheit und entspricht etwa dem FORTRAN SAVE. Beispiel b36.c (Listing 2).

Im Gegensatz zu "auto" werden bei "static" Variable vom System mit 0 initialisiert, sodaß i bei der Ausgabe die Werte 1, 2 und 3 hat. Die Verwendung der Speicherklasse "static" ist ansich nicht häufig und wird mitunter in fortgeschrittenen C-Programmen, z.B. in Presentation-Manager-Programmen für bestimmte Handles, gebraucht.

Die dritte Speicherklasse ist "register". Die Angabe von "register int i,j;" in einem Programm besagt, daß die Variablen i und j sehr häufig gebraucht werden, z.B. für Zähler in for-Loops und daher direkt in Maschinenregistern abgebildet werden sollen. Es ist leicht ersichtlich, daß so etwas extrem maschinenabhängig wirkt: Hängt von der Anzahl der vorhandenen, freien, Maschinenregistern, von der Wortlänge und mehr ab. Bei den Intel 80x86 werden hier gerne intern die Register SI und DI hergenommen. Auf der anderen Seite setzen optimierende Compiler von sich aus Zählervariable und dgl. auf Maschinenregister, sodaß diese Speicherklasse für normale C-Programme fast entbehrlich ist. Die Angabe "register" macht ansich nichts kaputt, denn nach Definition C muß diese Anforderung nicht befriedigt werden. Wenn der Compiler sämtliche Register für andere Aufgaben verplant hat, kann man soviel "register" angeben, wie man will: Wird beim Übersetzen ignoriert.

Die vierte und letzte Speicherklasse ist "extern". Sie hat einen ähnlich hohen Stellenwert wie "auto". Nur mit ihr ist es möglich, große statische Vektoren zu erzeugen, gleichzeitig erspart sie die Parameterübergabe (extrem schnell): Extern deklarierte Variable können allen möglichen Programmeinheiten zugänglich gemacht werden.. wie FORTRAN COMMON. Beispiel b37.c (Listing 3).

Hier nutzen wir zwei Eigenschaften der Speicherklasse "extern": Die Variablen nektor und max werden vor dem Hauptprogramm deklariert, und zwar ohne das Schlüsselwort "extern" (weil sie vor dem Hauptprogramm stehen, sind sie extern). Da wir das Large-Memory-Modell benutzen, kann jeder Vektor 64 KB groß sein. Damit kann die Speicheranforderung von nektor befriedigt werden. Die Variable max braucht zwar nur 4 Byte Speicher; sie ist extern, um sie ohne Parameterliste an fmul2 "übergeben" zu können. Durch die Schlüsselworte "extern" in main und fmul2 können beide auf die gemeinsame externe Speicherfläche, umfassend nektor und max, zugreifen.

Pascal-Programmierer und Anhänger "reinsten" Programmierens werden nicht von solchen Möglichkeiten angetan sein. In der Tat raten auch praktisch alle C-Lehrbücher von der Ausnutzung dieses Features ab: Besonders wenn mehrere Programmierer an einem Projekt arbeiten, muß so etwas sehr deutlich dokumentiert sein, sonst kann das Chaos enorm werden. Gleichwohl ist diese Methode der Parameterübergabe die schnellste und kompakteste, die denkbar ist. Im Falle der üblichen Parameterübergabe werden die Werte bzw. Adressen auf den Stack geschoben ("gepusht") und im Unterprogramm/in der Function dann vom Stack geholt ("gepopt"). Das kann, wenn eine größere Anzahl Parameter übergeben wird, durchaus nicht unbeträchtlich Zeit brauchen. Gleichzeitig kann man mit "extern" , wie ersichtlich, auch

Listing 10 (b44.c)

```
#include <stdio.h>

int main(void)
{
static double hektor[10]= {10.,20.,30.,40.,50.,
60.,                          70.,80.,90.,100.};
static int    iektor[10]= {0,1,2,3,4,5,
                                    6,7,8,9};
static int    jektor[10]= {9,8,7,6,5,4,
                                    3,2,1,0};
double *phektor;
int    *piektor,*pjektor;
int    i;

/* initialisieren der pointer */
phektor= hektor;
piektor= iektor;
pjektor= jektor;

do
  {
  printf("\ngib i (von 0..9): ");
  scanf("%d",&i);
  }
while( i < 0 || i > 9);

printf("\n *(phektor  + *(iektor  +
   *(pjektor + i)))= %lf\n", *(phektor  +
              *(iektor  +  *(pjektor + i))));
return 0;
}
```

Listing 11 (b45.c)

```
#include <stdio.h>

int main(void)
{
static double hektor[10]= {10.,20.,30.,40.,50.,
                    60.,70.,80.,90.,100.};
static int    iektor[10]= {0,1,2,3,4,5,
                                    6,7,8,9};
static int    jektor[10]= {9,8,7,6,5,4,
                                    3,2,1,0};
double *phektor;
int    *piektor,*pjektor;
int    i;

/* initialisieren der pointer */
phektor= hektor;
piektor= iektor;
pjektor= jektor;

do
  {
  printf("\ngib i (von 0..9): ");
  scanf("%d",&i);
  }
while( i < 0 || i > 9);

printf("\nphektor[piektor[pjektor[i]]]= %lf\n",
                phektor[piektor[pjektor[i]]]);
return 0;
}
```

Listing 12 (b46.c)

```
#include <stdio.h>
#include <malloc.h>

int main(void)
{
double *pvektor;
long lmax;
int max;
int i;
```

```
/* Speicher dynamisch anfordern */

printf("\nWieviel Elemente soll der Vektor
                                    enthalten:");
scanf("%ld",&lmax);
if(lmax < 64000L/sizeof(double))
  {
  printf("\nmax ist zulaessig..");
  max= (int) lmax;
  }
else
  {
  printf("\nmax ist zu gross..stop");
  return 1;
  }
pvektor= (double *) calloc(max,sizeof(double));
if(pvektor != NULL)
  printf("\nVektor erfolgreich angelegt..");
else
  {
  printf("\nVektor nicht angelegt ..stop\n");
  return 1;
  }

printf("\nVektor vektor mit i= 1..%d
                                belegen..",max);
  for(i= 0; i < max; i++) pvektor[i]= (dou-
ble)i;

printf("\ndas groesste Element
    nektor[%d]= %lf",max-1,pvektor[max-1]);
return 0;
}
```

Listing 13 (b47.c)

```
#include <stdio.h>
#include <string.h>

int main(void)
{
/* definition einer structure */
struct DATENSCHUTZ
      {
      char cvname[6];
      int ialter;
      };

/* speicherplatz anfordern */
struct DATENSCHUTZ spersdaten[5];
int i;

/* belegen */
strcpy(spersdaten[0].cvname, "Harry");
spersdaten[0].ialter= 21;
strcpy(spersdaten[1].cvname, "Heinz");
spersdaten[1].ialter= 50;
strcpy(spersdaten[2].cvname, "Hugo");
spersdaten[2].ialter= 16;
strcpy(spersdaten[3].cvname, "Maria");
spersdaten[3].ialter= 25;
strcpy(spersdaten[4].cvname, "Petra");
spersdaten[4].ialter= 9;

do
  {
  printf("\nGib Personen-Nummer (0..4):");
  scanf("%d",&i);
  }
while(i < 0 || i > 4);

printf("\nVorname:%s   Alter:%d\n",
spersdaten[i].cvname,spersdaten[i].ialter);
return 0;
}
```

sehr große statische Vektoren anlegen (b38.c, Listing 4).

Wir können, da der Vektor nektor nunmehr huge und obendrein extern ist, ganz normal compilieren und linken, ohne auf Stackgröße etc. Rücksicht nehmen zu müssen. Wir nutzen dieses Feature nun, um lediglich im Hauptprogramm mit einem derart großen Vektor operieren zu können und nehmen keine Parameterübergabe an eine Function vor (b39.c, Listing 5).

Das ist der oben geschilderte Fall, daß man die Speicherklasse "extern" wie FORTRAN COMMON lediglich dazu nutzt, um sehr große Vektoren zu definieren. Wir probieren das Programm b38.c nun auf einem größeren UNIX System V aus, (IBM AIX mit 32-Bit RISC Prozessor), wobei keine Segmentstruktur zu beachten ist (b40.c, Listing 6).

Das Schlüsselwort "huge" entfällt, soweit o.k. Unangenehm ist, daß der Standard-UNIX-Compiler cc nichts mit den feschen Function-Deklarationen (b38.c, Zeilen 8,10 und 26) anfangen kann.. man nimmt hier einfach den Defaulttyp int implizit an. Bei den Header-Files vom Typ *.h ist hier ein Pfad anzugeben. Wenn auch b38.c und b40.c funktionell völlig identisch sind (abgesehen von MAXELE 100000 für b38.c und MAXELE 1000000 für b40.c), ist einige Anpassungsarbeit erforderlich .. soviel zum Thema "C-Programme sind höchst portabel".

3. Pointer

Über Pointer hatten wir schon früher gesprochen und festgestellt, daß Pointer und Vektoren äquivalent sind. Wir wissen, daß gilt:

px= &x[0]
oder treffender px= x
und *(px + i) = x[i]
mit px ist pointer, x ist Vektor.

Neben vielen anderen Möglichkeiten, bei denen man Pointer vorteilhaft einsetzt, soll hier die der dynamischen Speicheranforderung behandelt werden. In obigen Beispielen b37.c bis b40.c hatten wir mit sog. statischen Speicheranforderungen (hat nichts mit der Speicherklasse "static" zu tun!) Vektoren definiert. Betrachten wir Beispiel b40.c: Der Vektor nektor wird zur Übersetzungszeit mit Anzahl MAXELE Elementen definiert, in diesem Fall 1 Million Elemente. Zur Laufzeit fordert das Programm dann unabdingbar 4 MByte Speicher für nektor an .. ich kann diese Speicheranforderung nicht im Programmlauf verändern. Dieses Vorgehen der statischen Speicheranforderung hat einen großen Vorteil: es ist ausgesprochen sicher. Entweder das Betriebssystem stellt beim Starten des Programms diesen angeforderten Speicher zur Verfügung oder nicht. Kann der Speicher nicht bereitgestellt werden, so wird das Programm sofort vom Lader abgebrochen mit einer Meldung wie "Programm does not fit in Memory" oder "Memory exhausted". Dieses Vorgehen wird z.B. generell von Standard-FORTRAN 77 genutzt (obwohl es F77- Compiler gibt, die eine dynamische Speicheranforderung zulassen). Nachteilig ist, daß Programme, die solchermaßen sehr große Vektoren anfordern, auf die Maschinengröße angepaßt sein müssen, wenn das Betriebssystem real adressiert: DOS und TOS. Bei virtuellen Betriebssystemen wie OS/2 oder UNIX ist eine Anpassung nicht nötig.

Es wäre doch sehr praktisch, wenn man Speicher je nach Bedarf anfordern könnte: Das Programm fragt vom Bediener ab, wieviel Platz für definierte Vektoren bereitgestellt werden soll oder aber das Programm ermittelt selbst, wieviel Kernspeicher momentan zur Verfügung steht und paßt seine Speicheranforderung dahingehend an.

In der Tat ist dies mit C sehr elegant möglich. Ehe wir jedoch das Vorgehen erläutern, sei festgestellt: Die Funktionen, die gleich erläutert werden, greifen auf hochentwickelte Betriebssystem-Routinen zu. Diese Routinen funktionieren beim einen Betriebssystem gut, beim nächsten weniger zuverlässig und beim dritten in der Praxis gar nicht! Halten wir sicherheitshalber nicht fest, bei welchen Systemen und Compilern das problematisch ist, sondern sa-

gen, wo das ordentlich (d.h. immer zuverlässig) funktioniert: Bei UNIX und OS/2. Es ist einfach so, daß derartige virtuelle Betriebssysteme systemimmanent genau auf solche Anforderungen ausgerichtet sind.
Genug der Vorrede, zu Beispiel b41.c (Listing 7).

Dieses Programm ist funktionell identisch zu b39.c. Hier wird zunächst abgefragt, wieviel Elemente der Vektor enthalten soll. Sodann wird die Funktion halloc() aufgerufen (steht für huge alloc), die einen Pointer auf einen long huge Pointer pnektor zurückgibt. Ist dieser Pointer nicht NULL, so konnte die Speicheranforderung bedient werden, ansonsten wird das Programm gestoppt.. so eine Sicherheitsabfrage sollte immer vorgesehen werden. Sodann wird der Pointer mit Werten belegt. Beachten, daß das, was bei einem Vektor [i] heißt, bei einem Pointer *(pointer+i) ist.

Das sieht alles sehr einfach aus und das ist es auch prinzipiell. Die generische Funktion zur Speicheranforderung heißt malloc(), deshalb wird auch das Header- File alloc.h eingefügt. malloc selbst stellt Speicherplatz für eine Anzahl von Bytes bereit (malloc heißt memory allocate = Speicher zuteilen). Wir greifen hier auf eine höherentwickelte Form von malloc() zurück, das ist calloc(), was besonders zum Anlegen von Vektoren geeignet ist. Da wir gleich in die Vollen gehen wollen, lassen wir uns mal ordentlich Speicher kommen..auch über 64 KB, daher wird die Sonderform halloc() gewählt. Verwirrend? In der Tat: Was wir wollen, ist eigentlich einen beliebig großen Vektor anfordern. Genau an diesem Punkt wird es wieder compilerabhängig:
Bei MS-C für DOS und OS/2:
halloc()
Bei Turbo-C für DOS :
farcalloc()
Bei UNIX System V :
calloc()

Das bedeutet in der Praxis: Im jeweiligen Compilerhandbuch unter der Rubrik "malloc" oder "calloc" nachsehen, was da im Einzelnen geboten wird. Bleiben wir hier einfach bei halloc() (bei anderen Compilern dieses Wort austauschen):

```
pnektor= (long huge *)
        halloc(max,sizeof(long));
```

Was passiert im Einzelnen? halloc() stellt zunächst einen Pointer vom Typ void zur Verfügung; die Speicheranforderung bezieht sich auf max Elemente der Größe long (sizeof(long)). Da pnektor ein Pointer vom Typ long huge ist, erfolgt der cast (die zwangsweise Umsetzung) vom Typ void auf den Typ (long huge *). Noch ein Beispiel: Wir fordern einen nicht sehr großen Vektor unter 64 KB vom Typ double an (b42.c, Listing 8).

Wie erwähnt, sind Funktionen wie calloc maschinen- und compilerabhängig. Der damit bereitgestellte Speicher wird bei Programmende von selbst freigegeben. Will man dies explizit tun, um beispielsweise während des Programmlaufs einen Vektor freizugeben und so Platz für einen neuen Vektor schaffen, so kann dies mit der generischen Funktion

```
void free(void *puffer)
```

geschehen. Auch hier compiler- und maschinenabhängig. Beispielweise wird bei MS-C ein huge vektor mit void hfree(void huge *puffer) freigegeben. Mit obigen Beispielen b41.c und b42.c experimentieren und die Aufrufe im Compilerhandbuch unter "malloc" und "free" nachsehen.

Wir hatten gesehen, daß ein Pointer, der einen Vektor darstellt, immer zunächst auf die Adresse des ersten Elements zeigt. Wenn nun gilt:
px = &x[0];
dann ist folgende Adressrechnung zulässig:
px++;
Nunmehr gilt:
px entspricht &x[1]

Nehmen wir nun wieder an, daß px = &x[0] oder kürzer px= x. Dann gilt für die Werte:

Vektor x	Pointer px
x[0]	*(px+0) oder kurz *px
x[1]	*(px+1)
..	..
x[n]	*(px+n)

Diese Schreibweise für die Werte von px ist logisch und einsichtig. Sie wird dann unübersichtlich, wenn mehrere Vektoren aufeinander zeigen, was in Vektorschreibweise sehr simpel ist (b43.c, List. 9).

Die Vektoren hektor,jektor und iektor müssen als "static" deklariert werden, damit sie innerhalb einer function (und main() ist eine function) mit Werten initialisiert werden können. Dieser Fall, daß ein Vektor auf einen anderen zeigt, kommt in der Praxis sehr häufig vor. In der uns bekannten Pointerschreibweise würde b43.c wie folgt aussehen (b44.c, Listing 10).

Etwas umständlich und undurchsichtig, oder ? Das geht aber viel eleganter, denn es gilt folgende Äquivalenz:
*(px + i) entspricht px[i]

Gut merken! Damit ist nun entgültig deutlich, daß Vektoren und Pointer in C praktisch ein und dasselbe bedeuten. Wir können unser Beispiel b44.c also wie folgt formulieren (b45.c, Listing 11).

Diese Aussage
*(px + i) entspricht px[i]
gilt grundsätzlich, also auch dann, wenn gar keine Vektoren vorliegen, sondern nur Pointer, die auch mit malloc() angelegt sein können (b46.c, Listing 12).

4. Structures

Das sind eigentlich Sonderfälle von mehrdimensionalen Vektoren, wobei die diversen Vektorelemente nicht vom gleichen Datentyp sein müssen. In Pascal entsprechen den C-structures in etwa die records, in üblichen BASICs und in FORTRAN 77 gibt es kein Äquivalent, dort muß mit mehreren eindimensionalen Vektoren gearbeitet werden. Ein typisches (und gern gebrauchtes) Beispiel, Personen und deren Alter (b47.c, Listing 13).

Das Beispiel spricht hoffentlich für sich selbst und über den Umgang mit structures. Structures können, wenn man das möchte, meist sehr leicht durch mehrere getrennte Vektoren abgebildet werden. Structures haben immense Bedeutung beim Programmieren von C-Programmen unter MS-Windows oder noch mehr unter dem IBM Presentation Manager von OS/2. Beispielsweise werden bei PM-Programmen x,y- Koordinaten als structures vom Typ POINTL abgebildet, was dann z.B. so aussieht:

```
POINTL ptl;

ptl.x=   100;
ptl.y=   200;
GpiMove(hps,&ptl);
ptl.x=   200;
ptl.y=   300;
GpiLine(hps,&ptl);
```

Dies nur als kleiner Ausblick (dabei ist POINTL in <os.h> definiert; GpiMove und GpiLine sind Grafikfunktionen des PM und hps ist ein handle auf einen sog. Presentation-Space).

In der vierten Folge werden wir das Vorgehen bei größeren Programmier-Projekten mit getrennten C-Sourcen zeigen und auf die entsprechenden Utilities wie MAKE und TOUCH eingehen.

Dr.-Ing. Frank Rieg
2002
Am Elfengrund 12
6100 Darmstadt

Fehler in den Listings der Folge 2!

Leider hat uns der Ventura Publisher einen bösen Streich gespielt.

Durch eine Eigenart von Ventura werden ">" und "<" beim Einlesen von Fremdformaten als Vorzeichen für Befehlssequenzen interpretiert. Um ">" und "<" in Ventura einlesen zu können, müssen diese Zeichen verdoppelt werden.

Dies ist uns erst bei dieser Folge bewußt geworden, sodaß in den Listings der Folge 2 die ">" und "<" fehlen. Die Listings sind daher nicht ohne Überarbeitung zu gebrauchen.

Wir bitten um Entschuldigung.

Euer Satzteam

PS.:

Die Listings erscheinen demnächst auf den MS-DOS-Disketten. Die Mitglieder der MS-DOS-Gruppe können sich also die Abtipperei ersparen.

LITTLE PAINTER Version 4.32

LITTLE PAINTER ist ein Zeichenprogramm für den Atari-ST. Die Shareware-Gebühr für dieses Programm beträgt 20 DM.

Autor: Markus Dheus
Hirtenstr. 9
8139 Bernried
Tel.: 08158/1898

Bei der Bezahlung ist der Absender nicht zu vergessen. Jeder registrierte Benutzer erhält dann eine Anleitung zugeschickt. Dieser Shareware-Autor bietet also ein komplettes funktionsfähiges Progamm in seiner neusten Version an, fügt aber keine Anleitung bei.

Nun zeichnet sich dieses Malprogramm durch seine benutzerfreundliche Bedienung aus (Figur 1). Die meisten Parameter können an dem Rand des Zeichenbretts mit der Maus eingestellt werden. Zusätzliche Optionen werden einfach über die GEM Menüleiste angewählt.

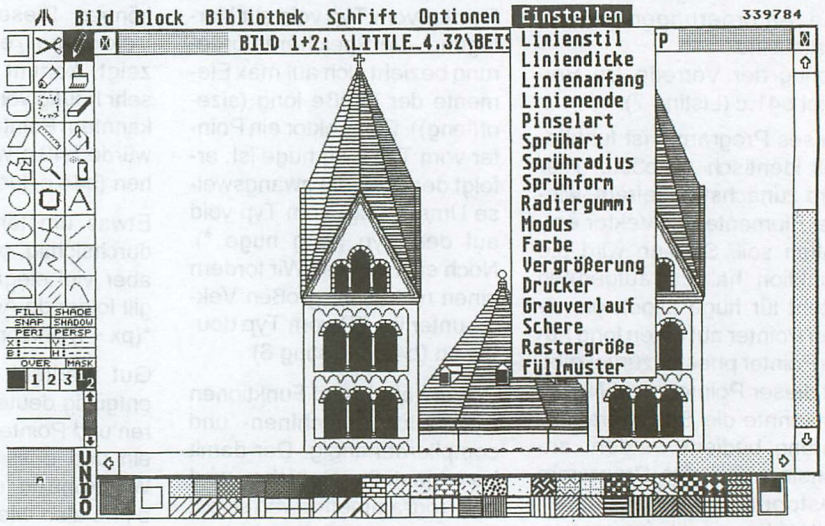


Bild 1: Hardcopy des Arbeitsbildschirmes

Das Programm arbeitet pixelorientiert, das heißt, die Auflösung der Zeichnung wird durch die Auflösung der Bildschirmdarstellung bestimmt. Da die Auflösung eines Bildschirms für größere Zeichnungen nicht immer ausreicht, kann das Programm mit Bildern der doppelten Bildschirmgröße arbeiten. Der angezeigte Bereich kann einfach mit der Maus durch Bewegen des schwarzen Blockes unten rechts im Bild eingestellt werden.

LITLLE PAINTER speichert Bilddateien im eingenem Format, kann aber auch mit DEGAS und STAD Dateiformat arbeiten. Um das Zeichnen z.B. von Schaltplänen zu Beschleunigen, kann man eine Bibliothek von Blöcken anlegen und diese einfach in der Zeichnung einbinden. Innerhalb des Programms können proportionale Zeichensätze vewendet werden. Ein Zeichensatzeditor für proportionale Zeichensätze (Figur 2) ist enthalten. Dieser Zeichensatzeditor verarbeitet das LITTLE PAINTER eigene Format LTP und GEM Zeichensätze. Als besonderes Bonbon liegt ein Programm bei, das Signum Fonts in GEM bzw. LITTLE PAINTER Fonts umwandelt. Damit kann man nun auf die große Zahl der SIGNUM Fonts zurückgreifen.

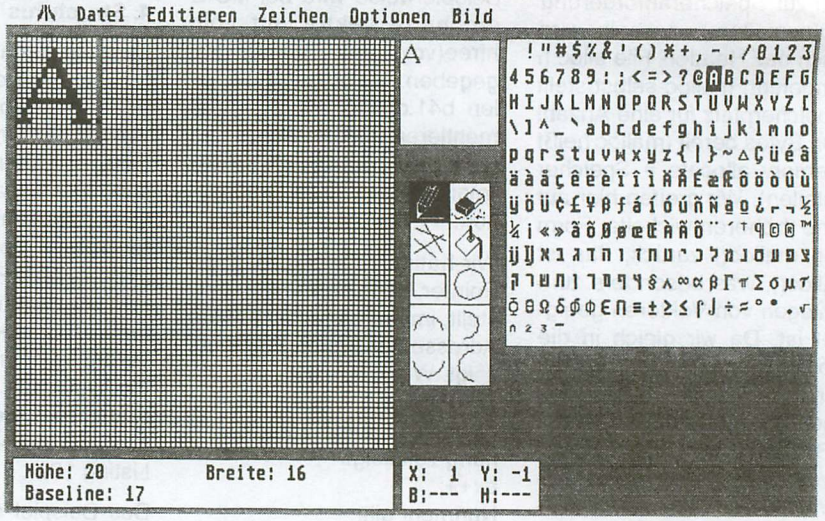


Bild 2: Hardcopy des Fonteditor-Schirmes

Noch ein Wort zur Druckerausgabe: LITTLE PAINTER kann bisher nur entweder auf 9-Nadel-Drucker (Epson kompatibel) oder auf 24-Nadel-Drucker (Nec P6 kompatibel) ausdrucken.

FAZIT:
Die einfache Benutzeroberfläche des Progamms LITTLE PAINTER macht dieses Programm gerade für die ersten Versuche auf dem Atari sehr geeignet. Es wird sehr lange dauern, bis man ein leistungsfähigeres Zeichenprogamm (z.B. mit Vektorgraphik) benötigt.
Obwohl das Programm auch ohne zusätzliche schriftliche Anleitung bedient werden kann, hoffe ich, daß diejenigen, die dieses Programm regelmäßig benutzen, auch die gewünschten 20 DM an den Autor zahlen. Änderungs- und Erweiterungswünsche zu dem Progamm sind erwünscht.

Die Version 5.0 ist in Arbeit.

ERHÄLTLICH:
Auf CCD-Diskette Nr.24

Werner Müller
Schallstr. 6
5000 Köln 41

HP-IL Controlinterface

(für den HP41)

Steuern mit dem HP41

Das vorliegende Controlinterface ist, in Verbindung mit einem vorhandenen HPIL-Modul, eine sehr preiswerte Alternative zum HPIL-Converter HP82166A für die Realisierung von Steuerungsaufgaben mit dem Taschencomputer HP41.
Gerade der neue HP41CX bietet sich mit seinen internen Zeitfunktionen für Steuerungsaufgaben an. Weiterhin können unter Verwendung der X-Funktionen und der Möglichkeit, ASCII-Files im Extended Memory zu speichern, einfache Steuerprogramme erstellt bzw. große Datenmengen für umfangreiche Steuerungsaufgaben mit dem Controlinterface ausgegeben werden.
Aus diesem Grund ist die im folgenden vorgestellte Prüf- und Steuersoftware speziell für den HP41CX verfaßt. Mit entsprechenden Programmen können selbstverständlich auch die Taschencomputer HP41C und HP41CV als Controller fungieren.
Bei dem Controlinterface handelt es sich, im Gegensatz zum HPIL-Converter HP82166, um ein Interface mit nonbidirektionalem Datenbus, d.h. Daten können nur gelesen werden.
Dabei kann das Datenwort den Wert 0000 $0001_{bin} \hat{=} 1_{dez}$ bis 1111 $1111_{bin} \hat{=} 255_{dez}$ annehmen, der Wert 0000 $0000_{bin} \hat{=} 0_{dez}$ kann aufgrund der speziellen Interfacestruktur nicht übertragen werden.
Schließlich kann das Controlinterface im Gegensatz zum HPIL-Converter innerhalb der Loop nicht adressiert werden. Dies kann bei der Verwendung mehrerer IL-Geräte zu ungewollten Datenausgaben des Controlinterfaces führen.

Dafür bietet das Controlinterface sehr preisgünstig (Materialkosten unter 100.-DM) einen potentialfreien, parallelen 8 Bit TTL-Datenausgang mit zugehörigem Taktsignal. Durch die optoelektronische Potentialtrennung von Controlinterface und nachfolgender Steuerlogik wird höchste Sicherheit für das gesamte System erreicht!
Die maximale Datenübertragungsrate liegt bei ca. 77 $^{Byte}/_{sec}$ ($\hat{=}$ 616 $^{Bit}/_{sec}$) und dürfte so auch für komplexere Steueranwendungen ausreichen.

Das Controlinterface ist eine Weiterentwicklung des Schaltinterfaces von Wolfgang Ressel, beschrieben im PRISMA 82.7.9. Mein besonderer Dank gilt dabei Wolfgang Ressel für die Beratung bei der Hardwareentwicklung sowie Helge Hörnis für die Bereitstellung des HP-IL Moduls und der Mitarbeit bei der Softwarentwicklung.

Erweiterung der Steuerfunktionen

Mit der im **IL-Modul** enthaltenen Funktion [OUTA] können nur die Datenworte mit dem Wert 0000 $0001_{bin} \hat{=} 1_{dez}$ bis 1111 $1111_{bin} \hat{=} 255_{dez}$ gesendet werden.

Die Funktion [OUTAN] des **EXTENDED-I/O Moduls** ermöglicht auch die Ausgabe des Wertes 0000 $0000_{bin} \hat{=} 0_{dez}$. Dazu muß ein "Dummy-Byte" gefolgt von dem Steuercode [ALPHA] [SHIFT] [N] [ALPHA] {↑} und dem Datenbyte im ALPHA-Register abgelegt werden. Mit 2 im X-Register wird dann [OUTAN] ausgeführt.
Mit dem **CCD-Modul** kann das Datenbyte **direkt** in Dezimal- oder Hexadezimalformat eingegeben werden.

Hardware Aufbauhinweise

Bei Aufbau des Controlinterface ist für dessen korrekte Funktion folgendes zu beachten:

- Richtige Polung der ICs ist notwendig, die Verwendung von IC-Fassungen erleichtert den Austausch defekter Schaltkreise.
- Die Drahtbrücken sollten zuerst eingelötet werden, danach alle übrigen Bauteile.
- Vor allem bei den Kondensatoren C_1 und C_2 (=100pF keramisch) und den Widerständen R_8 und R_9 (=39kΩ, metallschicht) sind Wertetoleranzen zu vermeiden, um eine einwandfreie Funktion des Interfaces zu erreichen.
- Der Spannungsregler 7805 benötigt einen angemessenen Kühlkörper, wenn man keine HC-MOS Bausteine verwendet, diese benötigen nur einen Bruchteil des Stromes von TTL-Bausteinen.
- Auf korrekte Polung der ICs achten.
- Die Optokoppler und der Gleichrichter müssen, wie im Bestückungsplan angegeben, in die Fassungen eingesetzt werden, es bleiben jeweils die mittleren Pins frei.
- Auch der richtige Anschluß des IL-Kabels sowie der 15-poligen MIND-Stiftleiste (Interfacebus) geht aus dem Bestückungsplan hervor.
- Für den Netztransformator sollte ein zweipoliger Ein/Aus-Schalter vorgesehen werden. Es empfiehlt sich der Einbau der Platine und des Netztransformators in ein schützendes Flachgehäuse.

Hardware Spezifikationen

Das Controlinterface arbeitet mit TTL-LS Schaltkreisen und benötigt eine Spannungsversorgung von 5 Volt.
(Bei HC-Logik wären auch andere Versorgungsspannungen möglich, dies würde allerdings einen Neuabgleich der 74HC123 erfordern, da die dort produzierten Zeitkonstanten spannungsabhängig sind.)
Die auf der Platine befindliche Spannungsstabilisierung muß für den Betrieb mit einem externen Netztransformator beschaltet werden (eventuell HP Akkuladenetzteil). Die Betriebsbereitschaft wird mit der LED "Power" angezeigt.

Das Controlinterface hat eine Stromaufnahme von ca. 300mA (60mA bei HC-Logik), die potentialgetrennten TTL-Ausgangstreiber benötigen noch einmal etwa 10mA.
Wahlweise kann eine auch unstabilisierte Versorgungsgleichspannung von etwa 8-10V an die Pins 14/15 angelegt werden, dies ist in Bild 1 zu erkennen:

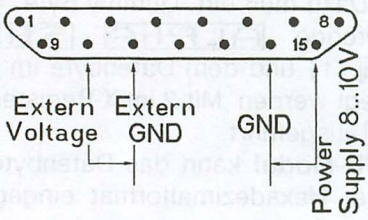


Bild 1: 25poliger Interface-Stecker

Wird eine Potentialtrennung von Controlinterface und nachfolgender Logikschaltung gewünscht, so muß das Controlinterface mit einem separaten Netzteil gespeist werden.
Der Einsatz von Optokopplern ermöglicht die Potentialtrennung der Daten- und Taktausgänge (Clock) vom Controlinterface. Dazu muß eine Interfaceausgangsstufe mit einer externen Versorgungsspannung von 5 Volt an Pin 10 (Extern Voltage) gegen Pin 12 (Extern GND) eingespeist werden.
Wird dagegen keine Potentialtrennung gewünscht, dann kann die Spannungsversorgung der TTL-Ausgangstreiberstufen mit der internen Interfacespannung erfolgen. Dazu sind Brücken von Pin 10 gegen Pin 11 und von Pin 12 nach Pin 13 zu schalten, siehe dazu Bild 2:

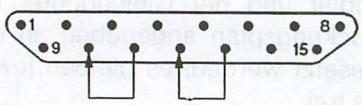


Bild 2: Kurzschlußbrücken

Die Datenausgänge des Controlinterfaces belegen Pin 1-8 des Interfacebusses. Pin 1 ≙ MSB (Most Significant Bit = höchstwertiges Bit) und Pin 8 ≙ LSB (Least Significant Bit = niederwertigstes Bit). An Pin 9 liegt das zu jedem Datenbyte gehörende Taktsignal (Clock), siehe Bild 3:

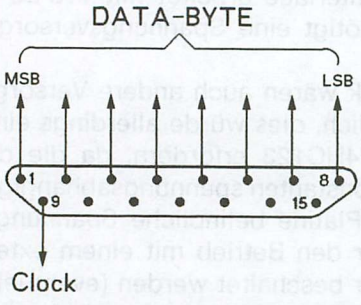


Bild 3: Datenausgänge

Die maximale Datenübertragungsrate (z.B. durch Auslesen des vollständig belegten ALPHA-Registers ≙ 12 Datenbytes) liegt bei ca. 77 $^{Byte}/_{sec}$ (≙ 616 $^{Bit}/_{sec}$). D.h alle 13ms wird ein neues Datenbyte übertragen.
Somit ergibt sich das in Bild 4 zu sehende Timingdiagramm.

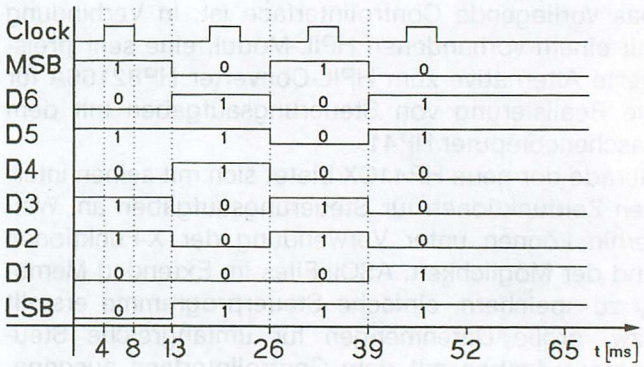


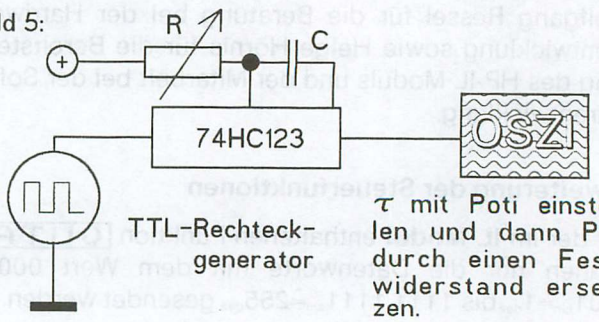
Bild 4: Timingdiagramm

Zeitkonstanten der Monoflops 74123:

$C_1 = 100pF$ $R_8 = 39k\Omega$	$\tau_1 = 1{,}8\mu s$	$C_1 = 100pF$ $R_8 = 15k\Omega$
$C_2 = 100pF$ $R_9 = 39k\Omega$	$\tau_2 = 1{,}8\mu s$	$C_2 = 100pF$ $R_9 = 15k\Omega$
$C_3 = 1{,}5nF$ $R_{10} = 10k\Omega$	$\tau_3 = 6\mu s$	$C_3 = 1{,}8nF$ $R_{10} = 3{,}0k\Omega$
$C_4 = 100pF$ $R_{11} = 10k\Omega$	$\tau_4 = 0{,}5\mu s$	$C_4 = 100pF$ $R_{11} = 2{,}2k\Omega$
$C_{10} = 100nF$ $R_{17} = 82k\Omega$	$\tau_{10} = 4ms$	$C_{10} = 100nF$ $R_{17} = 100k\Omega$
$C_{11} = 100nF$ $R_{18} = 82k\Omega$	$\tau_{11} = 4ms$	$C_{11} = 100nF$ $R_{18} = 100k\Omega$
Logikfamilie LS		Logikfamilie HC

Je nach IC-Hersteller können die erforderlichen RC-Glieder für die benötigten τ **sehr verschieden** ausfallen; wichtig ist, daß die Zeiten stimmen.
Deswegen ist ein Ausmessen mit einem Frequenzgenerator und einem Oszilloskop in einem Versuchsaufbau unbedingt erforderlich, das ist eben doch noch die gute alte Analogtechnik...
Siehe dazu Bild 5 für den Abgleich.

Bild 5:



τ mit Poti einstellen und dann Poti durch einen Festwiderstand ersetzen.

Obige Bauteilewerte (rechte Spalte der Tabelle) sind deshalb nur als Anhaltspunkt aufzufassen.
Auch bei einer Abweichung der Versorgungsspannung von nominal 5 Volt ist ein Abgleich der RC-Glieder erforderlich.

Software Spezifikationen

Das Controlinterface wird im ausgeschalteten Zustand mit dem im HP41CX einegsetzten IL-Modul verbunden. Nach dem Einschalten der Interfacespannung wird der Rechner in Betrieb genommen.

Die Interfacesteuerung und die Datenausgaben erfolgt mit dem ALPHA-Registerinhalt. Dabei entspricht jedes ALPHA-Symbol einem speziellen Bytewert. Im Folgenden werden "Steuerbytes" und "Datenbytes" unterschieden. Die Bytewerte werden mit ihrem äquivalenten "ALPHA-Symbol" bzw. dezimal dargestellt.

Das Interface wird mit dem Steuerbyte "↑" angesprochen. Das Steuerbyte wird mit folgenden Tastenfunktionen im ALPHA-Register abgelegt:
[ALPHA] [SHIFT] [N] [ALPHA]
Diesem Steuerbyte folgt das mit dem Controlinterface auszugebende Datenbyte. Prinzipiell muß jedem zu übertragenden Datenbyte das Steuerbyte vorangehen. Das Datenbyte kann den Wert 0000 $0001_{bin} \hat{=} 1_{dez} \hat{=}$ ↑$_{ALPHA}$ bis 1111 $1111_{bin} \hat{=} 255_{dez} \hat{=}$ ▒$_{ALPHA}$ annehmen.

Die Erzeugung des dem Dezimalwert äquivalenten ALPHA-Symbols erfolgt mit der Funktion [XTOA]. Diese Funktion übersetzt die im X-Register abgelegte Dezimalzahl in das äquivalente ALPHA-Symbol. Dabei wird das ALPHA-Symbol dem ALPHA-Registerinhalt angehängt.

Mit dem Extended I/O-Modul können auch Datenbytes mit dem Wert 0=Null mit Hilfe der Funktion [OUTAN] in's ALPHA-Register gebracht werden.
Mit dem CCD-Modul kann das Datenbyte direkt im Dezimal- oder Hexadezimalformat in das ALPHA-Register eingegeben werden.

Sind Steuerbyte und Datenbyte im ALPHA-Register abgelegt ([ALPHA] [SHIFT] [N] [ALPHA], Dezimalwert im X-Register, [XTOA]), erfolgt die Datenausgabe bei gesetztem Flag 17 ([SF] 17) mit dem IL-Modul Befehl [OUTA]. Mit dem Befehl [OUTA] können ein Datenbyte, maximal aber 12 Datenbytes (≙24 ALPHA-Symbolen im ALPHA-Register) nacheinander mit dem Controlinterface ausgelesen werden. Die Datenübertragungsrate liegt bei maximal 77 $^{Byte}/_{sec}$.
Für größere Datenmengen empfiehlt sich deren Speicherung in ASCII-Files im Extended-Memory.

Software Prüfprogramm "CICOUNT"

Das Programm CICOUNT dient der Überprüfung der Datenübertragung des Controlinterfaces. Dazu werden alle möglichen zulässigen Bytes von 0000 $0001_{bin} \hat{=} 1_{dez} \hat{=}$ ↑$_{ALPHA}$ bis 1111 $1111_{bin} \hat{=} 255_{dez} \hat{=}$ ▒$_{ALPHA}$ nacheinander ausgegeben.

Wird der Wert 255_{dez} erreicht, so beginnt die Datenausgabe erneut mit dem Wert 1_{dez}. Der Programmstop erfolgt mit der Taste R/S.

```
01  LBL "CICOUNT"
02  FIX 0
03  SF 17
04  1
05  STO 00
06  LBL 01
07  RCL 00
08  CLA
09  "↑"
10  XTOA
11  OUTA
12  "├"
13  ARCL X
14  AVIEW
15  RCL 00
16  255
17  X=Y?
18  GTO "CICOUNT"
19  1
20  ST+ 00
21  FS? 49
22  OFF
23  GTO 01
24  END
```

Software Prüfprogramm CILOOP

Das Programm CILOOP dient der Überprüfung der maximalen Übertragungsrate (oder Fehlerverfolgung) des Controlinterfaces.
Dazu werden die Datenausgänge D_7 bis D_0 abwechselnd auf 1-Pegel und 0-Pegel geschaltet. Das Programm lädt das ALPHA-Register mit den Bytes 0101 $0101_{bin} \hat{=} 85_{dez} \hat{=} U_{ALPHA}$ und 1010 $1010_{bin} \hat{=} 170_{dez} \hat{=}$ ▒$_{ALPHA}$.
Die zwölf Datenbytes erzeugen am Interfaceausgang folgendes Pulsdiagramm:

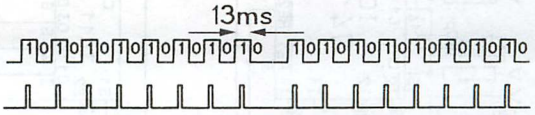


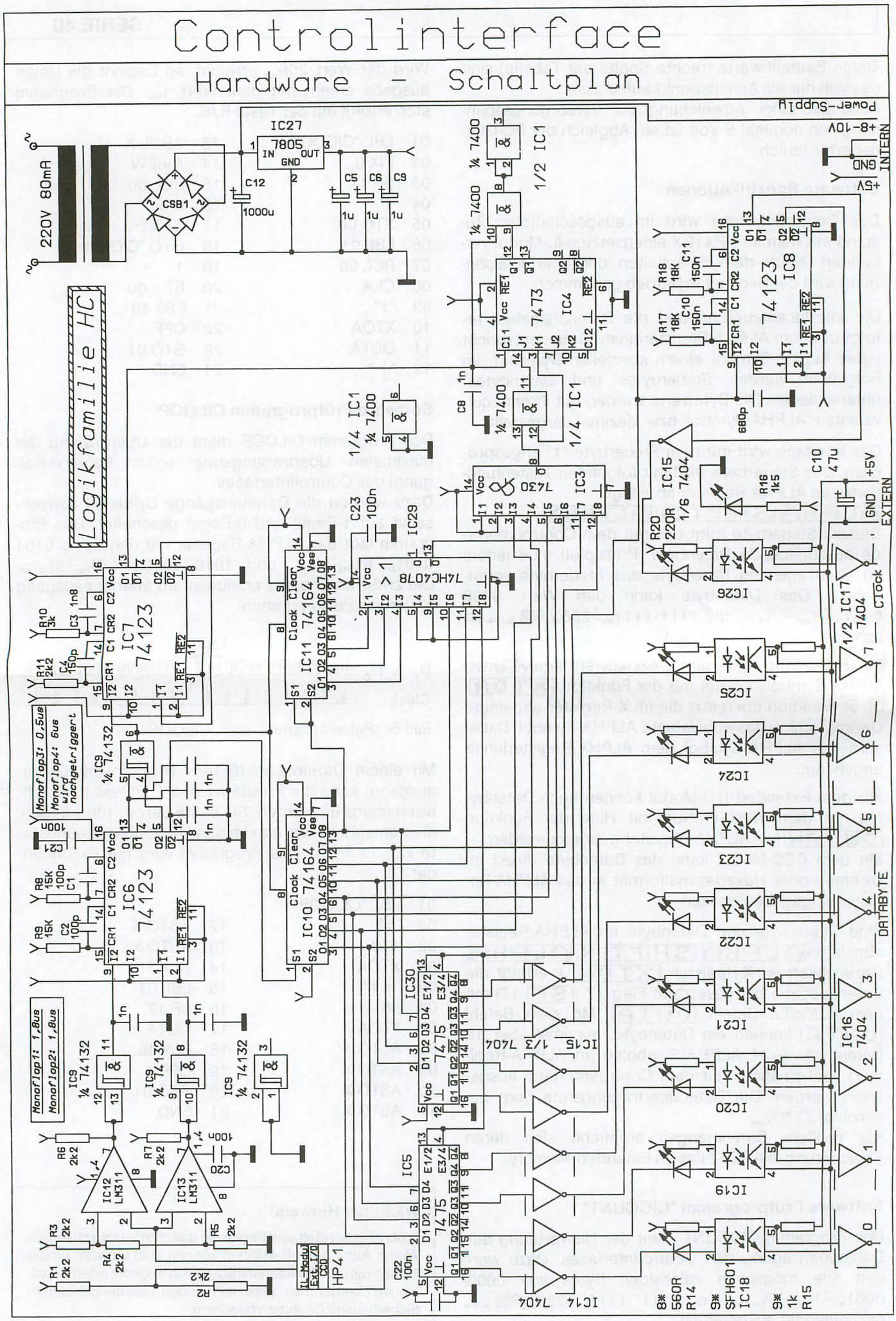
Bild 6: Pulsdiagramm von CILOOP

Mit einem Oszilloskop (Extern Triggern mit Datenausgang) kann die Pulsdauer des 1. Pulses meßtechnisch überprüft werden. Sie sollte bei ca. 13ms liegen, dies entspricht der maximalen Datenübertragungsrate von 77 $^{Byte}/_{sec}$. Das Programm wird mit R/S beendet.

```
01  LBL "CILOOP"
02  "↑"
03  170
04  XTOA
05  "├↑"
06  85
07  XTOA
08  ASTO X
09  ASTO X
10  ASTO X
11  ASTO X
12  ASTO X
13  ASTO X
14  CLX
15  LBL 01
16  SF 17
17  OUTA
18  FS? 49
19  OFF
20  GTO 01
21  END
```

Wichtiger Hinweis!

Trotz aller Sorgfalt sind Fehler nie gänzlich auszuschließen. Weder Autor noch Redaktion können eine Gewähr für die Richtigkeit und Funktionsfähigkeit von abgedruckten Schaltungen übernehmen. Bitte beim ersten Test die gebotenen Sicherheitsmaßnahmen beachten.


Controlinterface
Hardware - Schaltplan
Logikfamilie HC
Power-Supply
+8-10V
INTERN
220V 80mA
CSB1
IC27
7805
IN
GND
OUT
C12
1000u
C5
C6
C9
1u
1/2 IC1
1/4 7400
1/4 IC1
C8
1n
IC4
7473
IC8
74123
R18
18K
C11
150n
R17
C10
680p
IC3
7430
IC15
7404
R16
1k5
C10
47u
+5V
GND
EXTERN
R20
220R
1/6
IC26
IC25
IC24
IC23
IC22
IC21
IC20
IC19
IC16
7404
1/2 IC17
7404
Clock
DATABYTE
IC7
74123
IC11
74164
C23
100n
IC29
74HC4078
IC9
1/4 74132
Monoflop3: 0,5us
Monoflop4: 6us
wird
nachgetriggert
IC6
74123
IC10
74164
R8
15K
100p
C1
R9
15K
C2
100p
C21
100n
Monoflop1: 1,8us
Monoflop2: 1,8us
IC12
LM311
IC13
LM311
R6
2k2
R7
2k2
R3
2k2
R4
2k2
R5
2k2
R1
2k2
R2
2k2
C20
100n
IC30
74175
IC5
74175
IC15
7404
IC14
7404
C22
100n
IL-Modul
Ext. I/O
CCD
HP41
8*
560R
R14
9*
SFH601
IC18
9*
1k
R15

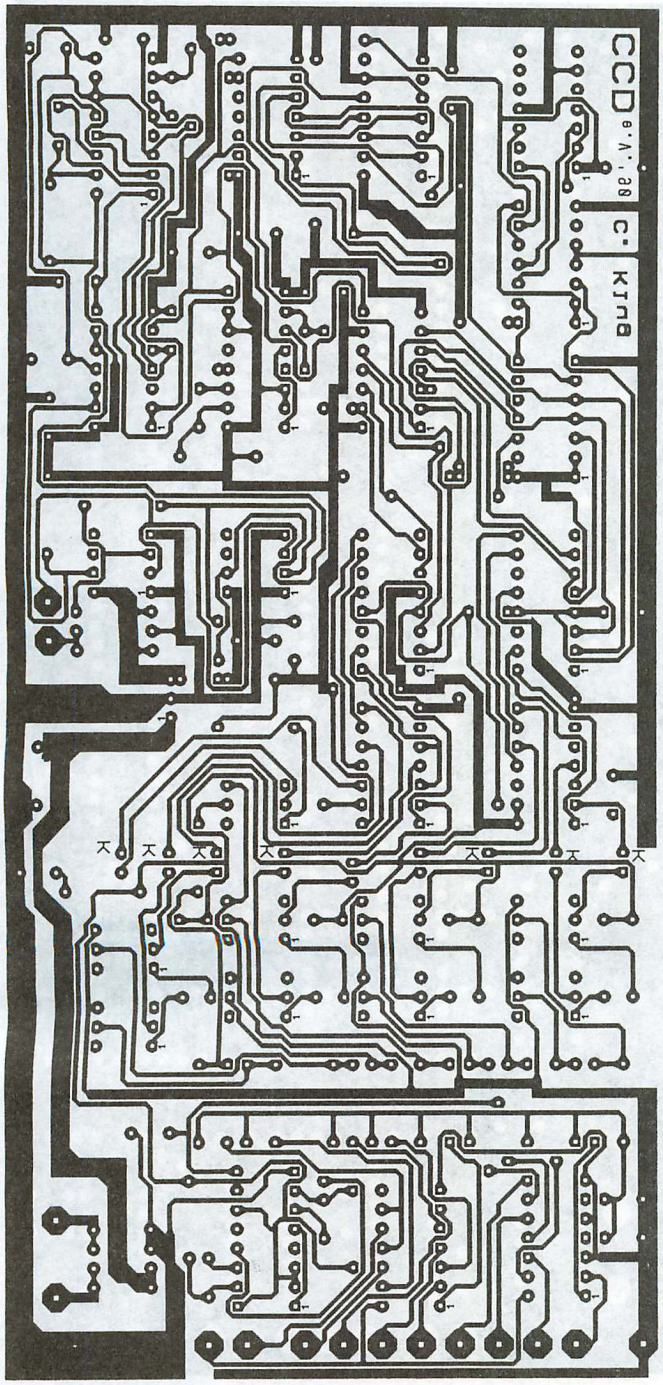


Bild 7: Lötseite

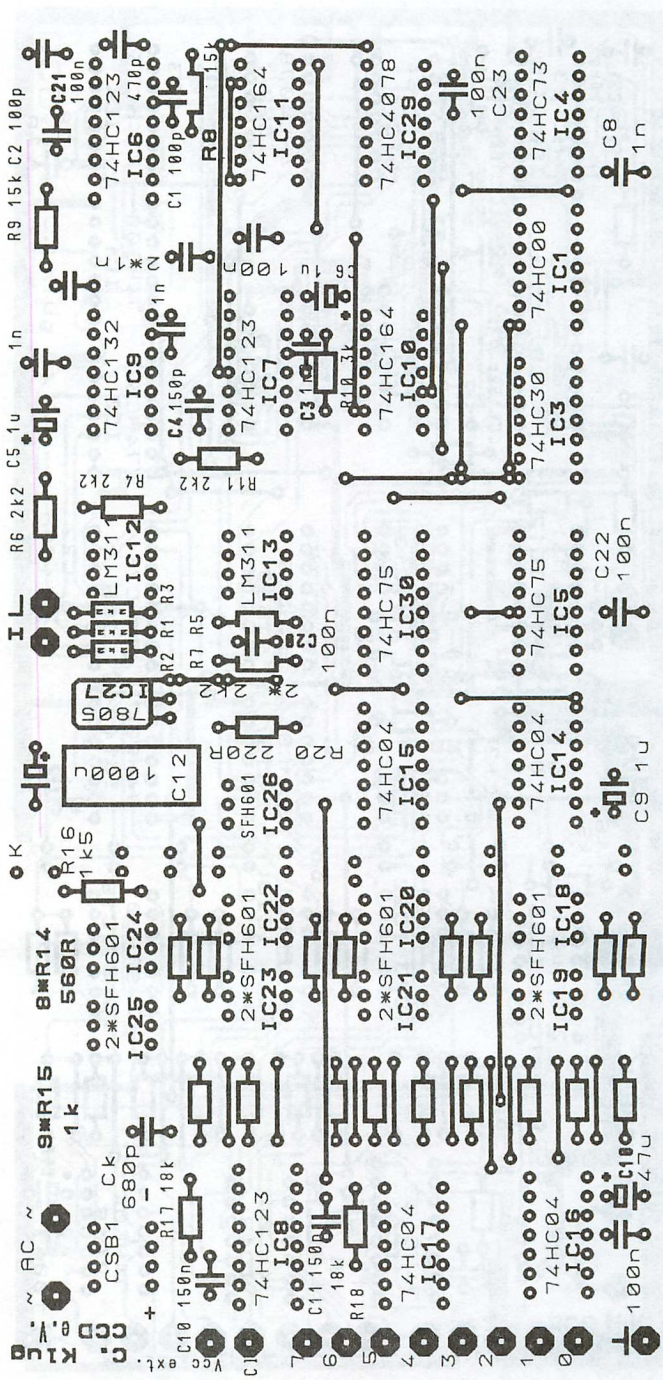


Bild 8: Bestückungsseite

Stückliste

Stk.	Teil	Bezeichn.	Kommentar
006	IC14	1/6_7404	*
003	IC15	1/6_7404	*
006	IC16	1/6_7404	*
003	IC17	1/6_7404	*
001	IC6	74123	*
009	R15	1k	*
001	C10	47u	16V
001	CSB1	8406	DIL-Brücken
001	IC27	7805	5V-Regler
001	C12	1000u	25V
001	C5	1u	16V
001	C6	1u	16V
001	C9	1u	16V
001	C3	1n8	*
001	R1	2k2	*
001	R3	2k2	*
001	R2	2k2	*
001	IC12	LM311	*
001	IC13	LM311	*
001	R4	2k2	*
001	R5	2k2	*
001	R7	2k2	*
001	R6	2k2	*
001	C1	100p	*
001	C2	100p	*
001	R8	15k	*
001	R9	15k	*
005	Ck	1n	keramisch
001	IC7	74123	*
001	C4	150p	*
001	R10	3k	*

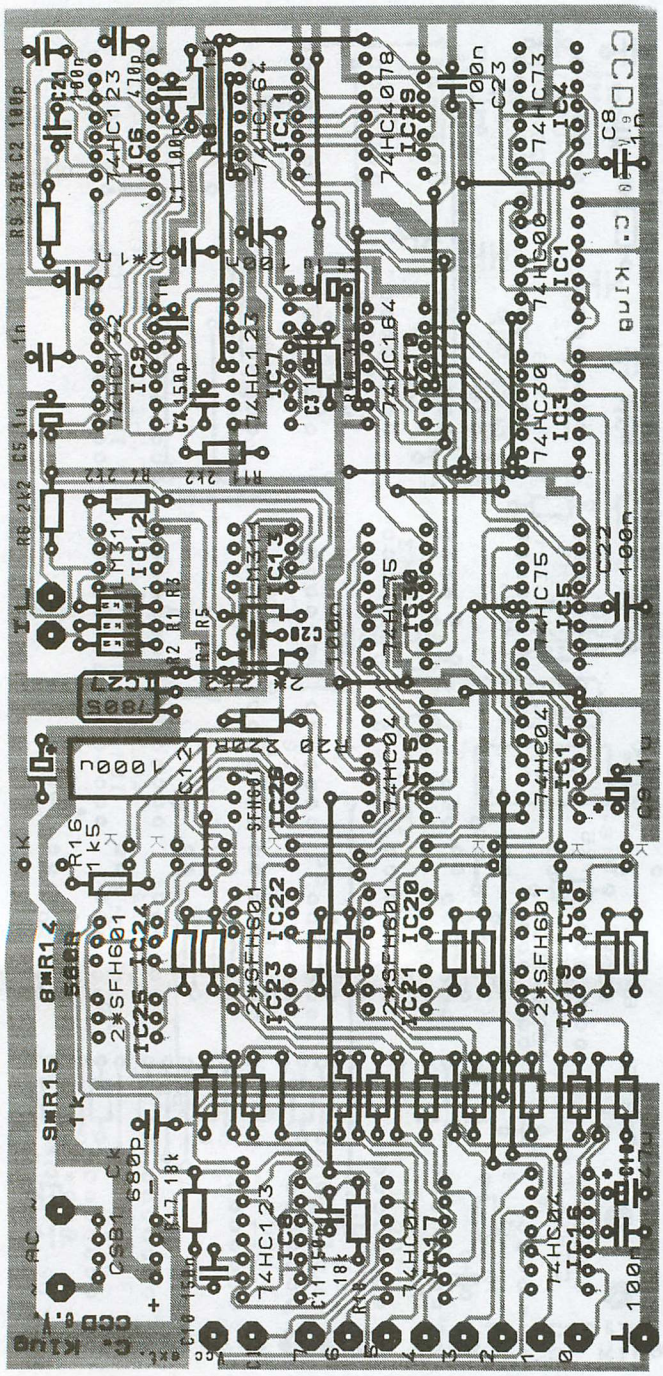


Bild 9: Platinenübersicht von oben gesehen

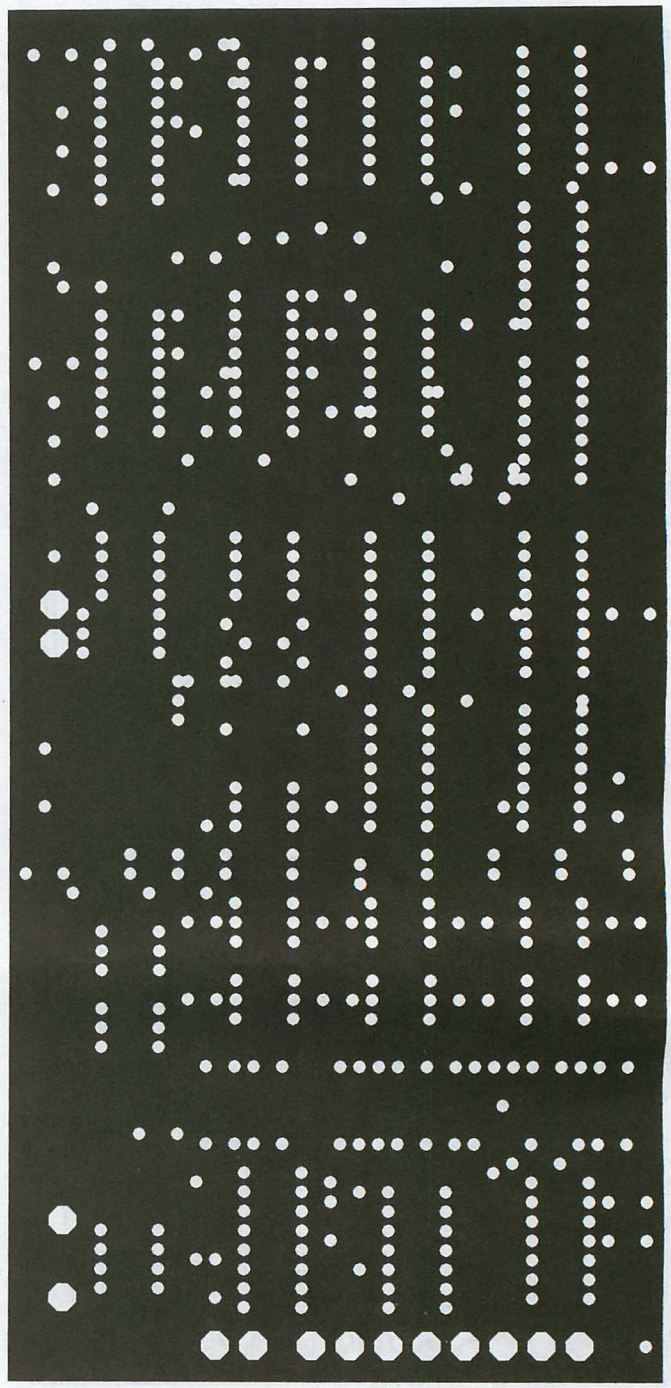


Bild 10: Lötstoppmaske

001	R11	2k2	*
001	IC11	74164	*
001	IC5	7475	*
001	IC30	7475	*
001	IC29	74HC4078	*
004	IC9	1/4_74132	*
004	IC1	1/4_7400	*
001	IC4	7473	*
001	C8	1n	*
001	IC8	74123	*
001	C11	150n	*
001	C10	150n	*
001	R17	18k	*
001	R18	18k	*
001	Ck	680p	*
001	IC18	SFH_601	*
001	IC19	SFH_601	*

001	IC20	SFH_601	*
001	IC21	SFH_601	*
001	IC22	SFH_601	*
001	IC23	SFH_601	*
001	IC24	SFH_601	*
001	IC25	SFH_601	*
001	IC26	SFH_601	*
008	R14	560R	*
001	R19	220R	*
010	LDx	LED_rot	*
001	R16	1k5	*
001	IC10	74164	*
001	IC3	7430	*
001	C20	100n	*
001	C21	100n	*
001	C23	100n	*
001	C22	100n	*

Christoph Klug
Leibnitzstraße 19
3200 Hildesheim

Emulation des
TI 58/59 auf HP-41

von Dipl.-Ing. R. Beucamp

In verschiedenen Veröffentlichungen finden sich Programme für die programierbaren Taschenrechner TI 58 und TI 59 von Texas Instruments. Sie lassen sich verhältnismäßig leicht auf den HP-41 umschreiben.

Einige Besonderheiten sind zu beachten. Die Formeln sind in mathematischer Schreibweise (AOS) mit Benutzung von Klammern und Gleichheitszeichen geschrieben. Klammern und Gleichheitszeichen müssen entfernt und die Formeln mit enter und unter Beachtung der Prioritäten in UPN umgestellt werden. Dabei sind auch bei TI 58 und TI 59 monadische Funktionen wie sin, 1/x etc. ihrem Argument nachgestellt und brauchen nicht umgestellt werden.

Sprungziele von bedingten oder unbedingten Sprüngen können die Programmzeilennummer oder Marken (Labels) sein.

Nach Labels (Befehl LBL) stehen als Adressen im Programmausdruck ähnlich dem HP-41 die Buchstaben A bis E und A'bis E', abweichend vierstellige Zahlen 0001 bis 0999 (entsprechende Programmschrittnummer) oder irgendwelche Funktionen wie z.B. 1/x, x, sin , +, = etc. Sie müssen durch die LBL-Adressen des HP-41 ersetzt werden, d.h. durch A bis J, a bis j oder 00 bis 99. A bis E und A' bis E' sind wie beim HP-41 auch vom Tastenfeld anwählbar.

Da der HP-41 keine Programmzeilennummern als Ziele kennt, muß vor solchen Zielpunkten ein Label eingefügt werden. Mehrfache Verwendung eines Labels kennen der TI 58 und TI 59 nicht.

Der Befehl INV kehrt die nachfolgende Funktion oder Abfrage in das Gegenteil um.

OP mit nachfolgender zweiziffriger Zahl sind Druck-, Statistik-, Signum oder Inkrementierungs- bzw. Dekrementierungsbefehle der Register 00 bis 09. Vor den OP-Druckbefehlen stehende bis zu 5 zweistellige Zahlen sind bis zu fünf codierte Alphazeichen (siehe Tabelle), die zu 4 x 5 Zeichen im Druckpuffer zusammengestellt und mit dem Befehl OP 05 als Zeile gedruckt werden. Mit dem Befehl OP 06 wird das Ergebnis einer Rechnung oder eine Eingabe mit bis zu 4 Schriftzeichen (rechts vom Ergebnis) ausgedruckt.

Mit OP 17 wird die Aufteilung des Speichers in Zahl der Datenregister und in Zahl der Programmschritte vorgenommen (Syntax: Zahl der Datenregister/10, OP 17).

Es können beim TI 59 bis zu 100 Datenregister in Gruppen zu je 10 eingeteilt werden (beim TI 58 bis zu 60). Dann sind noch 160 Programmschritte möglich. Jedes Datenregister belegt 8 Speicherplätze. Auf die Datenregister kann mit RCL und STO direkt und indirekt zugegriffen werden. Indirekte Zugriffe sind im Programmausdruck mit * gekennzeichnet. Speicherarithmetik ist möglich.

Vergleiche bei bedingten Sprüngen werden zwischen dem x-Register (=Display) und dem Inhalt eines Vergleichsregisters t vorgenommen, in das vorher der Vergleichswert oder durch den CP-Befehl eine Null zu speichern ist. Nach dem Vergleich wird bei "wahr" der nachfolgende Befehl, der nur eine Sprungadresse sein darf, ausgeführt. Bei "falsch" wird der nachfolgende Befehl igniriert. Das entspricht dem HP-41.

HIR-Befehle mit 2 nachfogenden Zahlen sind Datenspeicherbefehle mit zusätzlichen Registern. Im einzelnen gilt für die Befehle folgendes. Die Emulatuion sieht schwieriger aus als sie ist. Bei einem Programm kommen gewöhnlich nur wenig zu ändernde Befehle vor.

a) Nicht geändert zu werden brauchen folgende Befehle:
LBL A bis LBL E, STO 00 bis STO 99, RCL 00 bis RCL 99, R/S, x^2, $\sqrt{x}$, 1/x, COS, SIN, TAN, π, DEG, RAD, A bis E, LOG, RTN, +, -, *, ÷ (jedoch UPN beachten!).

b) Zu streichen sind Klammern und = Zeichen. Ersatz durch UPN mit Beachtung der Prioritätsregeln (1. Sonderfunktionen, 2. Potenzen, 3. Multiplikatoren und Division, 4. Addition und Subtrktion. Klammerausdrücke je für sich).

c) Abgeändert werden müssen folgende Befehle:
A' bis E' in a bis e
ADV in XEQ "ADV"
CMS in XEQ "CLRG" (Datenregisterlöschung)
CLR in CLX oder XEQ "CLST"
D.MS in "HR"
EE INV EE in XEQ "RND"
EE in EEX
ENG in ENG 7
FIX 03 in FIX 3
GO* in GTO ❐ nn (indirektes GOTO)
GRD in XEQ "GRAD"
GTO Name in GTO nn oder Name
HIR 01 bis HIR 08 (STO-Befehle in Stackregister 1-8) in STO Befehle besonderer Datenspeicher ändern
HIR 11 bis HIR 18 (RCL-Befehle für Stackregister 1-8) in RCL Befehle besonderer Datenspeicher wandeln
HIR 21 bis HIR 28 keine Operation
HIR 31 bis HIR 38 (SUM mit Stackregister 1-8) in STO + Befehle wandeln
HIR 41 bis HIR 48 (PRD mit Stackregister 1-8) in STO * Befehle wandeln
HIR 51 bis HIR 58 (INV SUM mit Stackregister 1-8) in STO - Befehle wandeln
HIR 61 bis HIR 98 (INV PROD mit Stackregister 1-8) in STO ÷ Befehle wandeln
INT in XEQ "INT"
ICOS = INV COS in COS^{-1}
IDMS = INV D.MS in "HMS"
IENG = INV ENG in FIX 9
IFF 00 in FS? 00
IFIX = INV FIX in FIX 9
IIFF = INV IFF in XEQ "FC?"
IINT = INV INT in XEQ "FRC"
ILOG = INV LOG in 10^x
ILNX = INV LNX in e^x
IPD* = INV PRD indirekt in STO ÷ ❐ nn
IPRD = INV PRD in STO ÷ nn
IP/R = INV P→R (x in t,y in Anzeige, dann φ in t,R in Anzeige) in R→P
ISBR = INV SBR in RTN
ISM* = INV SUM indirekt in STO - ❐ nn
ISIN = INV SIN in SIN^{-1}
ISUM = INV SUM in STO - nn
ISTF = INV STF in CF
ITAN = INV TAN in TAN^{-1}
INV x in Standardabweichung
IXI in XEQ "ABS"
a Iy^x b = a INV y^x b in a enter b 1/x $y^x = \sqrt[b]{a}$
LBL A' bis E' in LBL a bis e
LBL mit beliebiger nachfolgender Funktion z.B. LBL+, LBL GTO, LBL STO, LBL SIN, etc. in LBL 00 bis 99
LNX in LN
NOP in kein Befehl
OP 00 in CLA
OP 01 bis OP 04, Codes davor ins Alpharegister tippen
OP 05 in XEQ "PRA"
OP 06 (Ausdruck Ergebnis und bis 4 Zeichen) in ARCL.X und PRA
OP 10 in XEQ "SIGN"
OP 11 bis 15 Statistik (Varianz, Gerade, KorrKo., Schätzw.)

4 OP 17 (40 Datenspeicher vorsehen) in XEQ "SIZE" 040
OP 18 (setzt Flag 7, wenn kein Fehler) in entsprechende Anweisung
OP 19 (setzt Flag 7, wenn Fehler eingetreten) in entsprechende Anweisung
OP 20 bis 29 in 1 STO + 00 bis 09, CLX (Inkrementieren)
OP 30 bis 39 in 1 STO - 00 bis 09, CLX (Dekrementieren)
PAU in XEQ "PSE"
PD*07 in STO* ❒ 07
PRT in XEQ "PRX"
PGM 03 Einlesen eines Modulprogramms:
Für Standardmodul gilt:
01 und SBR CLR = lineare Regression
02 Matrix Inversion und Determinanten
03 Matrix Addition und Multiplikation
04 komplexe Arithmetik
05 komplexe Funktionen
06 komplexe trigonometrische Funktionen
07 Polynomentwicklung
08 Nullstellen von Funktion
09 Simpson (stetig)
10 Simpson (unstetig)
11 Dreiecksberechnungen (gegeben SSS, SSW, SWS)
12 Dreiecksberechnungen (gegeben WSW, SWW und Flächenberechnung)
13 Kreisbogenberechnungen
14 Normalverteilung
15 Zufallsgenerator
16 Kombination, Variation, Fakultät
17 gleitender Durchschnitt
18 Zinseszins, 19 Renten, 20 Kalender
23 Grad-Minuten-Stunden Addition, Subtraktion etc.
24, 25 Maß-Umrechnungen
PRD 07 in STO * 07 (Speicherarithmetik)
P/R (R in t,φ in Anzeige, dann y in Anzeige, x in t) in P→R
RAD in XEQ "RAD"
RC*07 in RCL ❒ 07 (indirekt)
RST in Rücksprung auf Programmanfang, Löschen aller Programmflags und Unterprogrammrücksprungadressen
SBR Name in XEQ nn oder XEQ "Name" (UP-Aufruf)
SM*07 in STO+ ❒ 07 (indirekte Speicherarithmetik)
ST*07 in STO ❒ 07 (indirekt)
SUM 07 in STO + 07 (Speicherarithmetik)
IXI in XEQ "ABS"
$\bar{x}$ = Mittelwert (01=Σ y, 02= Σ y^2, 03=N, 04= Σ x, 05= Σ x^2, 06= Σ xy)
X↔T = Austausch Anzeige mit Testregister siehe d) EXC und X↔T
a y^x b in a enter b y^x =a^b
a $1y^x$ b = a INV y^x b in a enter b 1/x y^x = $\sqrt[b]{a}$
+/- in CHS
= in Weglassen und UPN-Wandlung
( ) in Weglassen und UPN-Wandlung

d) Logische Umstellungen müssen bei folgenden Befehlen vorgenommen werden:
CP löscht Vergleichsregister t um Vergleiche mit 0 anzustellen. Wird ersetzt durch 0 in y-Register und Vergleiche x = y? etc.
DSZ 6 136 (DSZ, Register 0 bis 9, Schrittnummer oder Name) dekrementiert Register 0-9 um 1 und prüft auf Null. Wenn Register Ri = 0 wird die Programmschrittnummer übersprungen. Wenn Ri ≠ 0 wird in diesem Fall mit Schritt 136 fortgefahren. Ersetzt in diesem Fall durch 1, STO - 06, RCL 06, XEQ "X ≠ 0?" GTO mm und evtl. Label mm einbauen vor dem betreffenden Programmschritt 136 als Zielpunkt.
EQ Label oder nnnn = Vergleich auf Gleichheit zwischen Anzeige und t-Register. Wenn wahr Sprung zu Label oder Programmschrittzahl nnnn. Durch entsprechenden Vergleich ersetzten.
EXC nn tauscht Register nn mit der Anzeige aus, ersetzt durch: RCL nn, x↔y, STO nn
EX* in RCL ❒ nn, x↔y, STO ❒ nn
GE Label oder nnnn = Vergleich der Anzeige mit t-Register x ≥ t und wenn wahr Sprung zu Label oder nnnn, ersetzt durch entsprechenden Vergleich
INV DSZ wie DSZ, jedoch Überspringen bei nicht Null, ersetzt sinngemäß wie vor, aber XEQ "X = 0?"
INV EQ = Vergleich der Anzeige mit t-Register X ≠ t, ersetzt durch entsprechenden Vergleich
INV GE = Vergleich der Anzeige mit t-Register x < t, ersetzt durch entsprechenden Vergleich
x↔t = Austausch Anzeige = Testwert mit t-Register. Ersetzen durch Bringen des Testwertes in beliebiges Register und Rückholen diese Testwertes zum Testzeitpunkt. Dann Test mit x- oder y-Register, sowie löschen des Testwertes falls nötig.
Druckercodes (bis zu 5 zweistellige Zahlen vor den Befehlen OP 01 bis 04 und 06)

		zweite Ziffer							
		0	1	2	3	4	5	6	7
1.	0		0	1	2	3	4	5	6
Z	1	7	8	9	A	B	C	D	E
i	2	-	F	G	H	I	J	K	L
f	3	M	N	O	P	Q	R	S	T
f	4	.	U	V	W	X	Y	Z	+
e	5	x	*	√	∫	e	(	)	,
r	6	↑	%	▶	/	=	'	x	$\bar{x}$
	7	²	?	÷	!	□	△	∫	Σ

R. Beaucamp (1071)
Brockhoffstraße 4
4400 Münster

Schutz vor Programmverlust
beim HP48SX

Bei meiner Beschäftigung mit dem HP48SX konnte ich mich nur auf die Informationen stützen, die in den Handbüchern zu finden waren. Dabei stieß ich darauf, daß es scheinbar keinen Bereich zu geben scheint, der sich schreibschützen läßt.

Folgender Fall passierte mir vor kurzem:
Mit Stolz wird die Variable PGN mit einem neuen Programm belegt; alles scheint in bester Ordnung zu sein.
Eines Tages wird vor der Menü-Taste, die mit PGN belegt ist, versehentlich "left-shift" gedrückt. PGN wird mit dem Inhalt von Level 1 überschrieben, das Programm ist verloren...

Es ist nun deshalb nicht nötig, sich gleich eine RAM-Karte zu kaufen. Vielmehr läßt sich das Programm in Port 0 ablegen, wie auf der Seite 647 des englischen Handbuchs ganz unten fast verschämt bemerkt ist.

Dr. G. Heilmann
Obernhofer Straße 15
5408 Seelbach

Fadenpendel

mit 'großer' Amplitude von Dr. Heilmann

HP-15C
HP-41, 63 Zeilen, 107 Bytes, 16 Regs.

Für das reibungsfreie Fadenpendel der Länge l und der Masse m gilt mit Gravitationsfeldstärke g bei einer Auslenkung α der Energiesatz:

$$E = \frac{1}{2} m l^2 \dot{\alpha}^2 + m g l (1 - \cos \alpha)$$

Wird die Energie E durch die Maximalamplitude α_m ausgedrückt, so ergibt sich mit

$$1 - \cos \alpha = 2 \sin^2 \frac{\alpha}{2}$$

$$\dot{\alpha} = \frac{4g}{l} \cdot \left(\sin^2 \frac{\alpha_m}{2} - \sin^2 \frac{\alpha}{2} \right)$$

Durch die Integration ergibt sich als Schwingungszeit

$$(1)\ T = T(\alpha_m) = \frac{T_o}{a}$$

$$T_o = 2\pi \sqrt{\frac{l}{g}}$$

und mit der Substitution

$$\sin \frac{\alpha}{2} = \sin \frac{\alpha_m}{2} \cdot \sin x$$

$$\frac{\pi}{2\alpha} = \int_0^{\pi/2} \frac{dx}{\sqrt{1 - \sin^2 \frac{\alpha_m}{2} \cdot \sin^2 x}}$$

Dieses Integral , ein elliptisches Integral erster Ordnung, ist der Grenzwert der monoton absteigenden Folge

$$a_{k+1} = \frac{a_k b_k}{2}$$

und der monoton aufsteigenden Folge

$$b_{k+1} = \sqrt{a_k b_k}$$

mit $a_o = 1$ und $b_o = \cos \frac{\alpha_m}{2}$. Die Folgen werden mit

$$(2)\ a_n - b_n \leq \delta$$

vorgegebenes $\delta < 0$ abgebrochen. Wird der Faden durch eine gegen m gewichtslose Stange ersetzt gedacht, so ist das Verfahren für Amplituden bis 180° gültig.

Literatur:

Artikel von Milne Thompson, L. M. Formeln 17.3.1., s.590; 17.6.3., s.598 aus: Abramowitz, M. und Stegun, J. A. (Hrsg.): Handbook of mathematical functions. Dover Publications, New York 1965.

Schlußbemerkung:

Mit $\delta = 2 \cdot 10^{-10}$ führt das Verfahren auf dem HP 15C und auf dem HP 41 programmiert in wenigen Sekunden zum Ergebnis.

Programmbedienung auf dem HP 41:

1. Starten mit XEQ 'PD'

Anzeige:

L ↗ G

Pendellänge l in m und nach ENTER Gravitationsfeldstärke g in ms^{-2} eingeben.

R/S Führt zu

2. a(MAX)?

Die maximal-Amplitude des Pendels in Grad eingeben.

R/S

nach wenigen Sekunden

HO = Z.ZZZZZZ

Periodendauer des Pendels für 'kleine' Ausschläge in Sekunden

R/S

PD = Z.ZZZZZZ

Periodendauer des Pendels für 'große' Ausschläge in Sekunden

R/S

D = Z.ZZZZZZZ

Differenz H0 und PD in Sekunden

3. Wiederholung für gleiche l und g:

XEQ A

nach 2.

Programm für den HP-15C:

```
LBL C
L ↗ G          RI: 2·10^-10
/
SQRT           [L]=m
2
STO 2          [G]=m·s^-2
*
π
*
STO 3
RTN            H0,[40]=δ
LBL A          a(MAX)  0°<a<180°
RCL 2
/
STO 1
EEX
STO 0
LBL 0          H0 in Y-Register
RCL 1
RCL * 0        PD in X-Register
SQRT
X<>1
RCL + 0
RCL 2
STO 0
RCL -1
RCL I
XY?
GTO 0
RCL 3
ENTER
RCL / 0
RTN            PD [PD]=s
```

```
01*LBL "PD"
02 SF 27
03 5
04 PSIZE
05 2
06 STO 02
07 "L^G?"
08 PROMPT
09 /
10 SQRT
11 ENTER^
12 +
13 PI
14 *
15 STO 03
16*LBL A
17 SF 21
18 FIX 6
19 "a(MAX)?"
20 PROMPT
21 RCL 02
22 /
23 COS
24 STO 01
25 SIGN
26 STO 00
27 E1
28 CHS
29 10^X
30 ENTER^
31 +
32 STO 04
33*LBL 00
34 RCL 01
35 RCL 00
36 *
37 SQRT
38 X<> 01
39 ST+ 00
40 RCL 02
41 ST/ 00
42 RCL 00
43 RCL 01
44 -
45 RCL 04
46 X<=Y?
47 GTO 00
48 RCL 03
49 RCL X
50 RCL 00
51 /
52 "HO= "
53 ARCL Y
54 AVIEW
55 "PD= "
56 ARCL X
57 AVIEW
58 -
59 FIX 7
60 "D= "
61 ARCL X
62 AVIEW
63 END
```

Dr. G. Heilmann (3489)
Obernhofer Straße 15
5408 Seelbach

Neues vom HP 48 von Matthias Rabe

Im letzten Heft habe ich behauptet, der HP-48 könne nur Polynome symbolisch integrieren. Das ist falsch, es können "beliebige" Funktionen integriert werden. Das erste Integral, das ich eingefüttert habe, hat er zwar nicht geknackt, aber ich muß zugeben, daß es wohl doch etwas hinterhältig war; ich habe es jedenfalls auch nicht zu Fuß ausrechnen können - es gibt ja Integrale, die sich nicht analytisch lösen lassen. Wie leistungsfähig der 48'er in dieser Hinsicht wirklich ist, bedarf noch näherer Untersuchungen.

Die Kompatibilität zum HP-28 ist recht hoch, aber nicht vollkommen. Ich habe zum Vergleich den HP-28C herangezogen. Es gibt dort etwa zwei dutzend Funktionen, die beim HP-48 nicht in dieser Form oder unter identischem Namen vorhanden sind. Z.B. "!" anstatt "FACT" oder "KEY" gibt eine TastenNUMMER und keinen String zurück und "ABORT" muß durch "0 DOERR" ersetzt werden. Von solchen Kleinigkeiten abgesehen, beherrscht der HP-48 jedoch alle Möglichkeiten des HP-28, kann aber noch viel mehr als dieser. Ich vermisse eigentlich nur Suchpfade, ähnlich wie bei MS-DOS, *lokale Funktionen und Konstanten (ein Programm muß jetzt mit allen dazu gehörenden Routinen und Daten in einem Directory gehalten werden, wodurch man unnötigerweise Zugang dazu hat)*, es gibt keine "DO ... LOOP" Schleife und neudefinierte Einheiten müssen entweder dimensionslos sein oder auf dem SI-System basieren. Sollen z.B. Währungseinheiten definiert werden, so sind diese entweder dimensionslos, sodaß z.B. 5DM+3 berechnet werden kann, was dann 8 ohne Einheit ergibt. Oder man definiert eine Währungseinheit vielleicht als Arbeit. Dann kann man sofort ausrechnen, wieviele Stunden fernsehen für den Anschaffungspreis des neu erworbenen HP-48 drin gewesen wären (ein fragwürdiger Nutzen).

Den Nutzen des EquationWriters habe ich anfangs wegen seiner "Geschwindigkeit" bezweifelt. Wer jedoch schon mal fünf Minuten vergeblich versucht hat, in einer komplizierteren Formel die Klammern richtig zu sortieren, weis die symbolische Darstellung sehr zu schätzen.

Die Taktfrequenz des Rechners ist erfreulicherweise nicht die gleiche, wie beim HP-28. Bei einem der Selbsttests wird sie angezeigt: Mit rund zwei Megahertz ist der HP-48 etwa doppelt so schnell wie der HP-28S und dreimal so schnell wie der HP-28C und der HP-71B.

Der HP-48 kennt offiziell zwanzig Datentypen, doppelt so viele wie der HP-28. Und wer das Betriebssystem mit dem eingebauten HEX-Monitor durchforstet, findet noch weitere. So gibt es z.B. einen Datentyp für Assembler-Inline-Code oder LONG-REAL und LONG-COMPLEX

Die serielle Schnittstelle bedarf eigentlich keines "Schnittstellenmoduls", wie HP das entsprechende Kabel nennt. Ein einfaches Kabel mit zwei passenden Stekkern reicht aus. Da fängt aber auch schon das Problem an: Der Abstand der Kontaktstifte im HP-48 liegt mit 2mm abseits jeglicher Norm. Nach langer Suche habe ich den Hirschmann SPB-4 gefunden. Fragt jetzt bloß nicht, wo der zu bekommen ist, da niemand diesen Stecker auf Lager hat und ab Fabrik eine Verpakkungseinheit 200 Stück enthält. Wer nun doch so einen Stecker gefunden hat, wird dann ganz enttäuscht feststellen, daß er viel zu locker sitzt und keinen sicheren Kontakt gewährleistet. Das ist aber kein Problem, da sich die Kontakte mit einer Zange leicht aus dem Kuststoffträger herausziehen lassen und dann etwas zusammengedrückt werden können. Ich habe mir noch ein paar Plastikschnipsel mit Heißkleber auf den Stecker geklebt, um ihn auf die richtige Größe zu kriegen und das Ganze mit einem Schrumpfschlauch überzogen. Alles in allem eine ziemliche Bastelei, nur um knapp einhundert Mark zu sparen. Für alle, die dennoch ein Kabel selbst anfertigen wollen, hier die Kontaktbelegung. Von links nach rechts (Draufsicht auf die Kontakte, Tastatur nach oben): Signalmasse, Empfangsdaten, Sendedaten und Schutzmasse. Der Pegel, den der HP-48 abgibt, liegt bei gut +/- 4V, nach Norm müßte er aber zwischen +/- 5V und +/- 15V liegen. Es funktioniert aber auch so ganz gut, da der Empfänger einen Pegel von +/- 3V akzeptieren muß. Nur der Bereich zwischen +3V und -3V ist nicht definiert.

Wer nun ein Schnittstellenkabel hat, kann es nicht nur nutzen, um mit einem PC Daten auszutauschen. Wer die Hardcopy-Möglichkeit verschmerzt, kann auch einen normalen Drucker mit serieller Schnittstelle direkt benutzen. Die Zeilenbreite des Druckers kann in der PRINT-PAREMETER-Variable eingestellt werden, sodaß jeder Drucker optimal ausgenutzt wird. Listings sehen dann viel übersichtlicher aus als auf dem IR-Klopapierdrucker. Beim QuietJet wird der Latin-1 Zeichensatz überigens mir "ESC (0 N" eingestellt. Das wurde im Handbuch des QuietJet vergessen.

Anfangs haben mich die 32 kBytes RAM, die in der Grundausstattung vorhanden sind, gelinde gesagt, nicht vom Hocker geworfen. Der HP-48 geht aber viel sparsamer mit dem Speicher um, als der HP-71 mit seinem BASIC, wo Programme schnell mehrere kBytes groß werden. Auch benötigen Daten beim 71'er bei der Bearbeitung nochmal soviel freien Speicher, wie in der Variable, in der sie gespeichert sind. Werden zwei lange Zeichenketten auf Gleichheit verglichen, so werden sie erst auf den sogenannten Mathestack kopiert (was auch Zeit braucht), wogegen beim 48'er nur zwei Zeiger zu den entsprechenden Speicherstellen auf den Stack gelegt werden. Da beim HP-48 die Daten aber nur "irgendwo" hin geschrieben werden, muß der Speicher gelegentlich aufgeräumt werden, was etwas Zeit braucht und manchmal komische Effekte hervorrufen kann. Beim Programm "UHR" z.B. legt der Sekundenzeiger ab und zu eine kleine Gedenkpause ein, um dann die verlorene Zeit schnell wieder aufzuholen.

In den letzten drei Wochen fiel es mir garnicht so leicht, den HP-48 zu programmieren, obwohl Algorithmen sich mir vor ein paar Jahren geradezu kanonisch in UPN-Logik offenbarten. Jetzt habe ich mich aber schon wieder ganz gut daran gewöhnt. Wer direkt vom HP-41 umsteigt, dürfte keine großen Probleme haben und HP-28-Benutzer merken wahrscheinlich zuerst garkeinen Unterschied.

Mit dem HP-41 war ich eigentlich nie so ganz glücklich. Es gab zu viele Sachen, die er nicht konnte, oder die zu schlecht gelöst waren. So konnte das Mathe-ROM garnicht überzeugen. Als ich dann den HP-71 mit Mathe-ROM hatte, dachte ich, ich hätte den ultimativen Taschenrechner. Wie lange ich das wohl vom HP-48 glauben kann?

Bereits seit 1986 läuft auf meinem HP-71 ein Datenbank-Programm, welches nun schon in der dritten, von Grund auf neu geschriebenen Version existiert. Wegen meines Perfektionswahns und einiger Unzulänglichkeiten des HP-71 habe ich es bisher nicht geschafft, dieses Programm zu veröffentlichen (obwohl einige Leute mit den existierenden Versionen zufrieden arbeiten). Nun schwebt mir eine HP-48-Version vor. Der 48'er ist mit seinem achtzeiligen Display geradezu für solche Aufgaben prädestiniert. Dann kann ich demnächst endlich mit einer Liste meiner Schallplatten einkaufen gehen und im Zweifel schnell mal im Rechner nachsehen, ob ich eine Platte schon habe (es gibt Leute, die wären glücklich, so viele Schallplatten zu besitzen, wie ich doppelt habe). Dank der eingebauten RS232 ist es nicht mit zusätzlichem Hardwareaufwand für mehrere hundert Mark verbunden, Dateien z.B. mit DBase auszutauschen. Wegen des unmöglichen Datumsformates (tt.mmjjjj bzw. mm.ttjjjj) ist leider ein Größenvergleich zweier Daten schwierig geworden (um nach Datum zu sortieren, oder nach Datensätzen innerhalb eines Zeitintervalles zu suchen).

Matthias Rabe

VerHEXt nochmal!

Eines der Dinge, mit denen ich beim HP-48 SX als erstes Bekanntschaft machte, war der HEX-Monitor (Hallo Thomas!). Die Tastenkombination ON-D; Backspace; EVAL ergibt bekannterweise die Versionsnummer nebst Copyright.

Irgendwann habe ich dann mal eine andere als die von HP empfohlene Taste gedrückt, und siehe da, in der Anzeige tat sich was. Zunächst noch recht willkürlich, dann aber recht gezielt und mit der informellen Hilfe einiger CCD-Freaks tastete ich mich durch den Zahlendschungel. 1 Meganibble sind ja auch kein Pappenstiel, wenn's um das Erforschen derselben geht.

Kurz und gut, es ist ein vollwertiger ROM/RAM-Editor. Hier also in der Tabelle 1 die mir bisher bekannten und bestätigten Tastenbelegungen.

Taste	Bereich	
ENTER	00100	hier ist RAM!
	00101	Display Contrast Nibble
EEX	80000	Start Port 1
DEL	C0000	Start Port 2
+/-	F000A	Display RAM
	F0857	dto.
1/X	F08AC	9. Reihe
	F097B	Display RAM
	F10EE	dto.
UP	springt	um #1000h aufwärts
DOWN	dto.	um #1000h abwärts
/	dto.	um #100h ab
*	dto.	um #100h auf
-	dto.	um #1h ab
+	dto.	um #1h auf
	Druckerschnittstelle	
	und inkrementiert das Display	
SPC	dto., jedoch ab mom. Adr.	
	fortlaufend...	
0-9	HEX-Tastenfeld	
A-F	dto.	

Mit ON-C wird der HEX-Monitor verlassen.

Tabelle 1

Dem HP48 auf die Finger geschaut .

Was soll das Ganze ?! Naja , man will den HP48 doch mal an den Zehen kitzeln, oder ?! Nun, ähnlich wie beim seligen HP28 C/S wird die zuletzt im HOME-Dir erzeugte Variable am Top of RAM (ToR) abgespeichert. Genau 5 Nibbles unter #FFFFFh, denn diese wiederum sollten zunächst immer Null sein (Format s. Datentypen weiter unten), in einem Sub-Dir am Top of Dir (ToD), jedoch nicht zwingend am ToR!

Hierzu ein Beispiel:
Zuerst HOME drücken, dann folgendes PRGM eintippen:
(PRGM-Begrenzer)
RCL SWAP "NAJA!"
ENTER drücken.

Nun im HEX-ED zuerst Backspace, ENTER, dann '/' und danach sooft '-', bis links oben im Display FFFBF:30 ... zu sehen ist. Mit '+' oder '-' schrittweise vor- und zurückgehen. Die sichbar werdenden Codes sind in Tabelle 2 dargestellt.

Der größte Teil des Programms besteht aus Definitions-Adressen . Diese Adressen definieren entweder die Startadresse einer Funktion, z.B. SWAP;#1FBBDh oder einen Objekttyp sowie dessen Dimensionen, im vorliegenden Beispiel also TYPEDEF:String: #02A2Ch (Konstanten/Adressen werden von rechts nach links gespeichert). Nach #02A2Ch muß sich ein Längenfeld von 5 Nibbles anschließen, dann kommt erst der ASCII-Code. Das PRGM wiederum ist mit einer Anfangsdefinition (#2361Eh) und einer Ende-Definition (#23639h) versehen, die ihrerseits von einer BEGIN SUB-Adress-List-Definition (02D9Dh) und einer END SUB-Adress-List-Definition eingeschlossen ist. Und so weiter , und so weiter... Naja, so kompliziert , wie es klingt , ist es gar nicht! Nun soll das kleine Programm ein wenig modifiziert werden (Tabelle 3).

Die verwendeten Adressen sind natürlich nur beispielhaft, und der Code verschiebt sich bei Eingabe einer neuen Variablen.

Das RAM, welches in HEX-ED ab Adr. #F7FFFh - #FFFFFh angezeigt wird, be-

AdressCode

```
FFFBF : 30151515          ! Länge(Name)=03 ; Name='QQQ'
FFFC7 : 30                ! Nochmal Länge(Name)
FFFC9 : D9D20             ! DEF: SUB Adress List
FFFCE : E1632             ! DEF: USER-Code-PRGM Adr List
FFFD3 : 04B02             ! Adr.
FFFD8 : C2A20             ! DEF: STRING
FFFE2 : F0000             ! Länge(Str)=#0000Fh
FFFE7 : E414A41412        ! "NAJA!"
FFFF1 : 93632             ! DEF: END USER-Code
FFFF6 : B2130             ! DEF: END SUB Adr List
FFFFB : 00000             ! Wohl immer Null
```

Tabelle 2

- Hier bleibt noch alles beim alten -

```
FFFBF : 30151515          ! Länge(Name)=03 ; Name='QQQ'
FFFC7 : 30                ! Nochmal Länge(Name)
FFFC9 : D9D20             ! DEF: SUB Adress List
FFFCE : E1632             ! DEF: USER-Code-PRGM Adr List
- Ab hier wird folgendes eingegeben
- (Natürlich nur die Hex-Codes rechts des Doppelpunktes !!)
FFFD3 : C2A20             ! DEF: STRING
FFFD8 : 91000             ! Länge(Str)=00019h
FFFDD :8414C4C4F40255355425        ! "HALLO USER"
- Die obige Zeile war die Letzte !
- Also ab hier erstmal nichts mehr verändern !
FFFF1 : 93632             ! DEF: END USER-Code
FFFF6 : B2130             ! DEF: END SUB Adr List
FFFFB : 00000             ! Wohl immer Null
Nun bitte HEX-ED verlassen und das PRGM angeschaut .
```

Tabelle 3

Code:

```
CCD20  CON(5)      ! DEF: In-Line Machine Code
7B000  REL(5)      ! Länge in Nibbles (=B7h)
20     P=0
31A3   LC(2) 3Ah
20     P=0
10B    R3=C        ! R3= 59d
147    C=DAT1 A    ! C = Adr(Obj(Level 1))
133    AD1EX       ! A = Adr(L1)
1FC2A20  D1=(5)    ! STRING-Definition
137    CD1EX       ! C = 02A2C ; D1= Adr(Obj(L1))
145    DAT1=C A    ! Adr(Obj(L1))= C2A20 (= DEF: STRING)
101    R1=A        ! R1= Adr(L1)
174    D1=D1 + 5   ! D1= Adr(Len(Str))
147    C=DAT1 A    ! C = Len(Str)
137    CD1EX       ! C = Adr(Len(Str)) ; D1= Adr(Len(Str))
1C5    D1=D1 - 6   ! D1= Len-6
137    CD1EX       ! C = Len-6 ; D1= Adr(Len)
81E    CSRB        ! C = (Len-6)/2
108    R0=C        ! R0= (Len-6)/2
174    D1=D1 + 5   ! D1= Adr(POKE)
133    AD1EX       ! A = Adr(POKE) , D1= Adr(L1)
174    D1=D1 + 5   ! D1= Adr(L2)
147    C=DAT1 A    ! C = Adr(Obj(L2))
137    CD1EX       ! C = Adr(L2) , D1= Adr(Obj(L2))
179    D1=D1 + A   ! D1= Adr(Data(Obj(L2)))
147    C=DAT1 A    !C = Data(Obj(L2))
137    CD1EX       ! C = Adr(Data(Obj(L2))) ; D1= Adr(PEEK)
-LOOP-
AE2    C=0  B      ! C = 0 Exponent
1570   C=DAT1 P    ! C = Data(Adr(PEEK))
170    D1=D1 + 1   ! D1= Adr(PEEK)+1
137    CD1EX       ! C = Adr(PEEK)+1 ; D1= Data(Adr(PEEK))
```

```
17F     D1=D1 + F       ! D1= Data(Adr(PEEK))+16d
17F     D1=D1 + F       ! D1= Data(Adr(PEEK))+16d
17F     D1=D1 + F       ! D1= Data(Adr(PEEK))+16d
137     CD1EX           ! C = Data(Adr(PEEK))+48 ;
                          D1= Adr(PEEK)+1
123     AR3EX           ! A = 3Ah ; R3= Adr(POKE)
9E6     ?C>A  B         !
B0      GOYES + B       ! YES: 0<=Data<=9
137     CD1EX           ! C = Adr(PEEK)+1 ;
                          D1= Data(Adr(PEEK)+48
176     D1=D1 + 7       ! D1= Data(Adr(PEEK))+55d
137     CD1EX           ! C = Data(Adr(PEEK))+xx ;
                          D1= Adr(PEEK)+1
123     AR3EX           ! A = Adr(POKE) ; R3= 3Ah
133     AD1EX           ! A = Adr(PEEK)+1 ; D1= Adr(POKE)
1556    DAT1=C  B       ! Adr(POKE)= Data(Adr(PEEK))+xx
171     D1=D1 +2        ! D1= Adr(POKE)+2
133     AD1EX           ! A = Adr(POKE)+2 ; D1= Adr(PEEK)+1
118     C=R0            ! C = Len(Data)
CE      C=C-1  A        ! C = Len-1
480     GOC   +8        ! EXIT
108     R0=C            ! R0= Len-1
5AB     GONC  -70d      ! LOOP
-EXIT-
E7      D=D+1  A        ! Free-Item-Counter
119     C=R1            ! C = Adr(L1)
137     CD1EX           ! C = Adr(Data(PEWEK)+n) ; D1= Adr(L1)
147     C=DAT1  A       ! C = Adr(Obj(L1))
174     D1=D1 + 5       ! D1= Adr(L2)
145     DAT1=C  A       ! Adr(L2)= Adr(Obj(L1))
-Der folgende Code sollte am Ende fast jeden In-Line Codes
-stehen . Ausnahmen bestätigen die Regel .
142     A=DAT0  A       ! A = Obj(Adr(Instruction Pointer))
164     D0=D0 + 5       ! D0= NEXT Instruction
808C    PC=(A)          ! PC= Momentaner Befehl + n Nibbles (=NEXT)
```

Tabelle 4

findet sich tatsächlich aber im Bereich ab #77FFFh.

Ein kleiner Leckerbissen zum Schluß:
Ein PEEK-Prgm, welches n Nibbles ab Adr.#AAAAAh als String nach Ebene 1 zurückgibt. n sollte nicht größer als der momentane freie Speicherplatz in Nibbles (=Bytes/2) abzüglich von etwa 250 Bytes für Systemzwecke sein.

Eingabe:
2: #AAAAAh
1: n
PEEK drücken
1: "CCCC...n"

Das PRGM braucht für 10000 Nibbles ca. 2 Sekunden! Die Verzögerung ist auf die relativ lange Schleife (LOOP) zurückzuführen, die leider etwas lahm ist. Sonst wäre die benötigte Zeit unter einer Sekunde!

Hier ist der User-Code Teil :
<< SWAP 2 6 ^ STWS RR RL OVER ABS 5 - R->B #1h SWAP BLANK Code 1 ROT SUB >>

Code ist bei der PRGM-Eingabe zunächst ein String mit 89 Zeichen Länge. Erst nachdem im HEX-ED der unten angegebene Code eingetippt wurde, erscheint bei 'PEEK' VISIT/RCL Code in der Anzeige. Das fertige PRGM keinesfalls editieren, da beim dekompilieren und kompilieren die Kennung 'Code' in ein Namen-Objekt umgewandelt wird.

So, wer will, kann eine kürzere, schnellere Version schreiben oder noch besser eine äquivalente POKE-Funktion! Kleiner Tip: Ein paar kleine Änderungen bei PEEK führen zur Lösung.

Das wars für diesmal, happy PEEKing!

Raymond Hellstern
Liebigstr.8
3000 Hannover 1
(0511)661011
#2853

Tag & Jahr

von Rainer Gillmann

216 Bytes, SIZE 005, HP-41C, TIME

Ich habe das interesante Programm "D?Y" von Dr. M. Hochenegger überarbeitet und möchte gerne meine Version davon vorstellen.

Meine Version ist ohne CCD-, IL- und EXT I/O-Modul lauffähig. Es kann wahlweise mit oder ohne Drucker betrieben werden. Außerdem ist es möglich , z.B. jeden Freitag den 13. zu berechnen. Hierzu muß lediglich bei der Datumsangabe der Monat weggelassen werden. Wenn das Anfangsjahr der Berechnung das laufende Jahr ist, muß bei dem Prompt z.B. "AB 1989 ?" nur R/S gedrückt werden. Die Druckausgabe ist zugegebenermaßen nicht so elegant, dafür wurde das Programm jedoch lediglich um ganze 19 Bytes länger und paßt noch locker auf eine Magnetkarte.

Übrigens der Wochentag wird in der Form "SO", "MO", DI" ... direkt in das Alpha-Register eingegeben.

Beispiel:

Wann fällt der Heiligabend auf einen Mittwoch ?
MI 24.12.1997
MI 24.12.2003
MI 24.12.2008
MI 24.12.2014
MI 24.12.2025
MI 24.12.2031

Wann fällt der Freitag auf einen 13. des Monats ?
FR 13.01.1989
FR 13.10.1989
FR 13.04.1990
FR 13.07.1990
FR 13.09.1991

```
01◆LBL "D?Y"
02◆LBL E
03 FIX 0
04 CF 29
05 CF 01
06 "DD,MM=?"
07 PROMPT
08 INT
09 STO 00
10 LASTX
11 FRC
12 STO 01
13 X=0?
14 SF 01
15 .01
16 FS? 01
17 STO 01
18 AON
19 "WOCHENTAG?"
20 PROMPT
21 AOFF
22 ASTO 02
23 "SO"
24 CLX
25 XEQ 01
26 "FR"
27 5
28 XEQ 01
29 "MO"
30 1
31 XEQ 01
32 "DI"
33 2
34 XEQ 01
35 "MI"
36 3
37 XEQ 01
38 "DO"
39 4
40 XEQ 01
41 "SA"
42 6

43◆LBL 01
44 RCL 02
45 ENTER↑
46 ASTO X
47 X≠Y?
48 RTN
49 R↑
50 R↑
51 STO 03
52 CF 22
53 DATE
54  E2
55 *
56 FRC
57  E4
58 *
59 STO 04
60 "AB "
61 ARCL 04
62 "├ ?"
63 PROMPT
64 FS? 22
65 STO 04

66◆LBL 02
67 RCL 04
68  E6
69 /
70 RCL 00
71 +
72 RCL 01
73 +
74 DOW
75 RCL 03
76 X≠Y?
77 GTO 03
78 CLA
79 ARCL 02
80 "├ "
81 ARCL 00
82 "├."
83 RCL 01
84  E2
85 *
86 10
87 X>Y?
88 "├0"
89 ARCL Y
90 "├. "
91 ARCL 04
92 AVIEW
93 FC? 21
94 STOP

95◆LBL 03
96 FS? 01
97 GTO 05
98◆LBL 04
99 1
100 ST+ 04
101 GTO 02

102◆LBL 05
103 .01
104 ST+ 01
105 RCL 01
106 .12
107 X<=Y?
108 X=Y?
109 GTO 02
110 -
111 STO 01
112 GTO 04
113 END
```

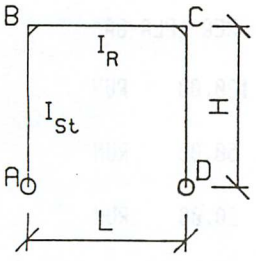
Rainer Gillmann
Möllers Kamp 11
2050 Hamburg 80

Statik:

Zweigelenkrahmen von Alfred Büttner

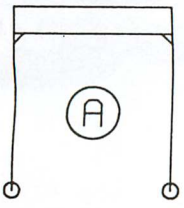
Ich möchte im vorliegenden Beitrag das Thema "Zweigelenkrahmen" einmal von der praktischen Seite beleuchten, wobei nicht viel Zeit mit langen theoretischen Abhandlungen verbracht werden soll. Im Folgenden erst einmal die grundlegenden Einzelkonstruktionen für die Zweigelenkrahmen, die Variablennamen in den Gleichungen finden sich auch fast identisch im Programm bei den Abfragen wieder.

Grundwerte:



$$C=K=\frac{I_R * H}{I_{St} * L} \qquad N=2K+3$$

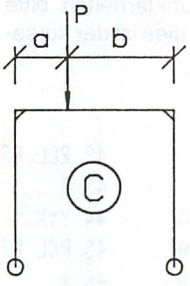
Gleichlast: Riegel



$$M_B / M_C = -\frac{Q/M * L^2}{4N} \qquad A=B=\frac{Q/M * L}{2}$$

$$H_A / H_D = -\frac{M_B}{H} \qquad M_F = \frac{Q/M * L^2}{8} + M_B$$

Einzellast: Riegel

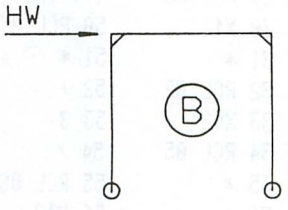


$$M_B = M_C = -\frac{P*a*b}{L} * \frac{3}{2N} \qquad V_A = \frac{P*b}{L}$$

$$H_A / H_D = -\frac{M_B}{H} \qquad V_D = \frac{P*a}{4}$$

$$M_F = \frac{P*a*b}{L} + M_B$$

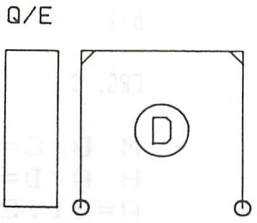
H von links: einseitig



$$V_A = -V_D = -\frac{HW*H}{L} \qquad H_A = H_D = -\frac{HW}{2}$$

$$M_B = -M_C = +\frac{HW*H}{2}$$

Gleichlast: einseitig Stiel

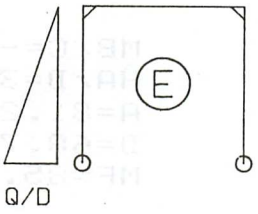


$$M_B = M_C = \frac{Q/E*H^2}{4}\left[-\frac{K}{2N} \pm 1\right]$$

$$H_D = -\frac{M}{H} \qquad H_A = -(Q/M*H - H_D)$$

$$V_A = -V_D = -\frac{Q/M*H^2}{2L}$$

Dreieckslast: einseitig Stiel

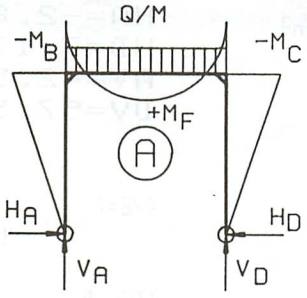


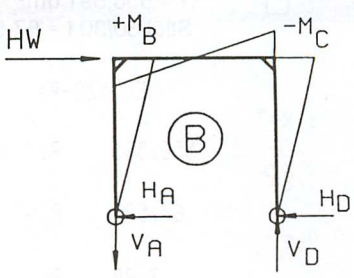
$$M_B = M_C = \frac{Q/D*H^2}{12}\left[-\frac{7k}{10N} \pm 1\right]$$

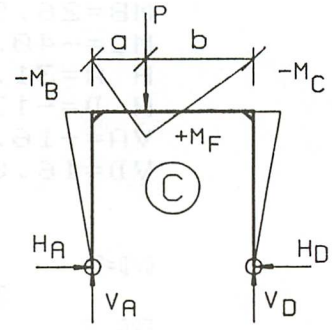
$$H_D = -\frac{M_C}{H} \qquad H_A = \frac{Q/M*H}{2} - H_D$$

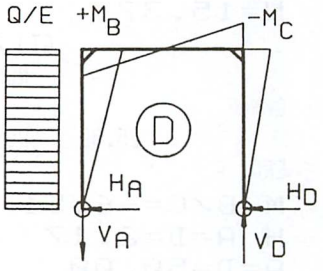
$$V_A = -V_D = \frac{Q/M*H^2}{GL}$$

5 Lastfälle









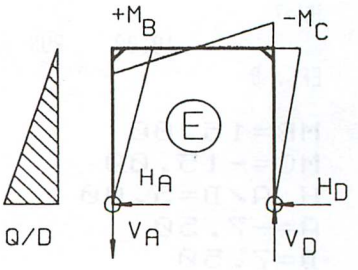
A+C

A+B

D+E

C=k=

N=



Nach der vielen trockenen Theorie ein einfaches Beispiel anhand einer Garagenabfahrt, dies dürfte wohl noch am leichtesten einleuchten.

Beispiel:

Garagenabfahrt

Lasten:

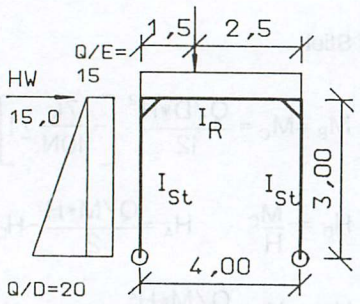
Erdüberschüttung und

Nutzlast	Q/M	=	25 kN/m²
Bremslast	HW	=	15 kN/m²
Raddruck	P	=	50 kN

Erdruck einseitig aus:

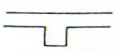
Nutzlast	Q/E	=	15 kN/m²
Erde	Q/D	=	20 kN/m²

Lastbild:



Grundwerte:

Riegel: Plattenbalken



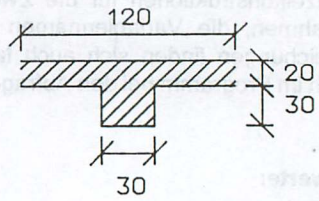
120*30+20+30 cm

I = 556,591 dm³

Stiel 30/30 I = 67,5 dm³

```
                XEQ "ZG-Ra"
I/R=?
          556.591     RUN
I/S=?
           67.500     RUN
H=?
            3.00     RUN
L=?
            4.00     RUN
C=K=6.18
N=15.37
                    GTO A
                      RUN
Q/M=?
           25.00     RUN
ERG. A
M  B/C=-6.51
H  A=D=2.17
A=D=50.00
M/F=43.49
                    GTO B
                      RUN
HW=?
           10.00     RUN
ERG. B
MB=15.00
MC=-15.00
H  A/D=5.00
A=-7.50
D=7.50
                    GTO C
                      RUN
P=?
           50.00     RUN
a=?
            1.50     RUN
b=?
            2.50     RUN
ERG. C
M  B/C=-4.58
H  A/D=1.53
A=31.25
D=18.75
MF=42.30
                      RUN
ERG. A+C
MB/C=-11.08
HA/D=3.69
A=81.25
D=68.75
MF=85.79
                      RUN
ERG. A+B
MB=8.49
MC=21.51
HA=-2.83
HD=7.17
AV=42.50
DV=57.50
                    GTO D
                      RUN
Q/E=?
           15.00     RUN
ERG. D
MB=26.96
MC=-40.54
H  A=31.49
H  D=-13.51
VA=-16.88
VD=16.88
                    GTO E
                      RUN
Q/D=?
           20.00     RUN
ERG. E
MB=10.77
MC=-19.23
HA=23.59
HD=-6.41
VA=-7.50
VD=7.50
                      RUN
ERG. D+E
MB=37.73
MC=-59.77
HA=55.08
HD=-19.92
A=-24.38
D=24.38
```

Die Endüberlagerung findet dann durch einfache Addition statt.




```
                XEQ "PLA-BA"
B/1=?
          120.00     RUN
D/0=?
           50.00     RUN
D=?
           20.00     RUN
D/1=?
           30.00     RUN
B/0=?
           30.00     RUN

e/O=16.82
e/U=33.18
F=3,300.00

J/Y=556,590.91

W/O=33,094.59
W/U=16,773.97
```

Bei Rückfragen und Unklarheiten bitte direkt an mich wenden, dies ist der kürzeste Weg.

```
01♦LBL "PLA▪BA"
02 "B/1=?"    22 *          42 RCL 03
03 PROMPT     23 +          43 3
04 STO 01     24 STO 06     44 Y↑X
05 "D/0=?"    25 RCL 01     45 RCL 07
06 PROMPT     26 RCL 05     46 *
07 STO 02     27 -          47 RCL 02
08 "D=?"      28 STO 07     48 3
09 PROMPT     29 RCL 03     49 Y↑X
10 STO 03     30 X↑2        50 RCL 05
11 "D/1=?"    31 *          51 *
12 PROMPT     32 RCL 02     52 +
13 STO 04     33 X↑2        53 3
14 "B/0=?"    34 RCL 05     54 /
15 PROMPT     35 *          55 RCL 00
16 STO 05     36 +          56 X↑2
17 RCL 01     37 2          57 RCL 06
18 RCL 03     38 /          58 *
19 *          39 RCL 06     59 -
20 RCL 04     40 /          60 STO 09
21 RCL 05     41 STO 08     61 RCL 02
```

```
62 RCL 08
63 -
64 STO 10
65 RCL 09
66 RCL 08
67 /
68 STO 11
69 RCL 09
70 RCL 10
71 /
72 STO 12
73 ADV
74 "e/O="
75 ARCL 08
76 AVIEW
77 "e/U="
78 ARCL 10
79 AVIEW
80 "F="
81 ARCL 06
82 AVIEW
83 ADV
84 "J/Y="
85 ARCL 09
86 AVIEW
87 ADV
88 "W/O="
89 ARCL 11
90 AVIEW
91 "W/U="
92 ARCL 12
93 AVIEW
94 ADV
95 ADV
96 END
```

```
01♦LBL "ZG-Ra"
02 "I/R=?"
03 PROMPT
04 STO 01
05 "I/S=?"
06 PROMPT
07 STO 02
08 "H=?"
09 PROMPT
10 STO 03
11 "L=?"
12 PROMPT
13 STO 04
14 RCL 01
15 RCL 03
16 *
17 RCL 02
18 RCL 04
19 *
20 /
21 STO 00
22 2
23 *
24 3
25 +
26 STO 05
27 ADV
28 SF 12
29 "C=K="
30 ARCL 00
31 AVIEW
32 "N="
33 ARCL 05
34 AVIEW
35 CF 12
36 STOP
37 ADV
38♦LBL A
39 "Q/M=?"
40 PROMPT
41 STO 06
42 RCL 04
43 X↑2
44 *
45 4
46 /
47 RCL 05
48 /
49 CHS
50 STO 07
51 2
52 /
53 RCL 05
54 *
55 CHS
56 RCL 07
57 +
58 STO 10
59 RCL 07
60 RCL 03
61 /
62 CHS
63 STO 08
64 RCL 06
65 RCL 04
66 *
67 2
68 /
69 STO 09
70 "ERG. A"
71 PRA
72 ADV
73 SF 12
74 "M B/C="
75 ARCL 07
76 AVIEW
77 "H A=D="
78 ARCL 08
79 AVIEW
80 "A=D="
81 ARCL 09
82 AVIEW
83 "M/F="
84 ARCL 10
85 AVIEW
86 CF 12
87 STOP
88 ADV
89♦LBL B
90 "HW=?"
91 PROMPT
92 STO 11
93 RCL 03
94 *
95 2
96 /
97 STO 12
98 2
99 *
100 RCL 04
101 /
102 STO 14
103 RCL 11
104 2
105 /
106 STO 13
107 "ERG. B"
108 PRA
109 ADV
110 SF 12
111 "MB="
112 ARCL 12
113 AVIEW
114 "MC=-"
115 ARCL 12
116 AVIEW
117 "H A/D="
118 ARCL 13
119 AVIEW
120 "A=-"
121 ARCL 14
122 AVIEW
123 "D="
124 ARCL 14
125 AVIEW
126 CF 12
127 STOP
128♦LBL C
129 "P=?"
130 PROMPT
131 STO 15
132 "a=?"
133 PROMPT
134 STO 16
135 "b=?"
136 PROMPT
137 STO 17
138 RCL 16
139 *
140 RCL 15
141 *
142 1.5
143 *
144 RCL 04
145 RCL 05
146 *
147 /
148 CHS
149 STO 18
150 RCL 03
151 /
152 CHS
153 STO 19
154 RCL 17
155 RCL 04
156 /
157 RCL 15
158 *
159 STO 20
160 LASTX
161 RCL 16
162 *
163 RCL 04
164 /
165 STO 21
166 RCL 15
167 RCL 16
168 *
169 RCL 17
170 *
171 RCL 04
172 /
173 RCL 18
174 +
175 STO 22
176 "ERG. C"
177 PRA
178 ADV
179 SF 12
180 "M B/C="
181 ARCL 18
182 AVIEW
183 "H A/D="
184 ARCL 19
185 AVIEW
186 "A="
187 ARCL 20
188 AVIEW
189 "D="
190 ARCL 21
191 AVIEW
192 "MF="
193 ARCL 22
194 AVIEW
195 CF 12
196 STOP
197 ADV
198♦LBL "A+C"
199 RCL 07
200 ST+ 18
201 RCL 09
202 ST+ 20
203 ST+ 21
204 RCL 08
205 ST+ 19
206 RCL 10
207 ST+ 22
208 "ERG. A+C"
209 XEQ "PRA"
210 ADV
211 SF 12
212 "MB/C="
213 ARCL 18
214 AVIEW
215 "HA/D="
216 ARCL 19
217 AVIEW
218 "A="
219 ARCL 20
220 AVIEW
221 "D="
222 ARCL 21
223 AVIEW
224 "MF="
225 ARCL 22
226 AVIEW
227 CF 12
228 STOP
229♦LBL "A+B"
230 RCL 07
231 RCL 12
232 +
233 STO 06
234 RCL 07
235 ST- 12
236 RCL 08
237 RCL 13
238 -
239 STO 18
240 RCL 08
241 ST+ 13
242 RCL 09
243 RCL 14
244 -
245 STO 20
246 RCL 09
247 ST+ 14
248 ADV
249 "ERG. A+B"
250 PRA
251 SF 12
252 "MB="
253 ARCL 06
254 AVIEW
255 "MC="
256 ARCL 12
257 AVIEW
258 "HA="
259 ARCL 18
260 AVIEW
261 "HD="
262 ARCL 13
263 AVIEW
264 "AV="
265 ARCL 20
266 AVIEW
267 "DV="
268 ARCL 14
269 AVIEW
270 CF 12
271 STOP
272♦LBL D
273 "Q/E=?"
274 PROMPT
275 STO 06
276 RCL 03
277 X↑2
278 4
279 /
280 *
281 STO 07
282 RCL 00
283 2
284 /
285 RCL 05
286 /
287 CHS
288 1
289 +
290 STO 08
291 2
292 -
293 RCL 07
294 *
295 STO 09
296 RCL 08
297 RCL 07
298 *
299 STO 10
300 RCL 09
301 RCL 03
302 /
303 STO 11
304 RCL 06
305 RCL 03
306 *
307 RCL 11
308 +
309 STO 12
310 RCL 03
311 X↑2
312 RCL 06
313 *
314 2
315 /
316 RCL 04
317 /
318 STO 13
319 ADV
320 "ERG. D"
321 PRA
322 ADV
323 SF 12
324 "MB="
325 ARCL 10
326 AVIEW
327 "MC="
328 ARCL 09
329 AVIEW
330 "H A="
331 ARCL 12
332 AVIEW
333 "H D="
334 ARCL 11
335 AVIEW
336 "VA=-"
337 ARCL 13
338 AVIEW
339 "VD="
340 ARCL 13
341 AVIEW
342 CF 12
343 STOP
344 ADV
345♦LBL E
346 "Q/D=?"
347 PROMPT
348 STO 06
349 RCL 03
350 X↑2
351 12
352 /
353 *
354 STO 07
355 RCL 00
356 7
357 *
358 10
359 /
360 RCL 05
361 /
362 CHS
363 1
364 +
365 STO 08
366 2
367 -
368 RCL 07
369 *
370 STO 14
371 RCL 08
372 RCL 07
373 *
374 STO 15
375 RCL 14
376 RCL 03
377 /
378 STO 16
379 RCL 06
380 RCL 03
381 *
382 2
383 /
384 RCL 16
385 +
386 STO 17
387 RCL 03
388 X↑2
389 RCL 06
390 *
391 6
392 /
393 RCL 04
394 /
395 STO 18
396 "ERG. E"
397 PRA
```

Uhr

von Matthias Rabe

Das Programm UHR bringt - na was wohl- eine laufende Analoguhr in das Display des HP-48. Das ist zwar nichts sinnvolles, aber vieleicht findet jemand Gefallen daran. Es war einer meiner ersten Programmierversuche auf dem HP-48.

Das etwa ein halbes kByte große Grafikobjekt habe ich natürlich nicht per Hand eingehackt. Das Zifferblatt wurde zum Teil mit einem Programm berechnet, teils interaktiv mit dem Grafikeditor erstellt und dann mit dem sogenannten interaktiven Stack in das Programm integriert. Ebenso die Konstanten für XRNG, YRNG und sec.

Einmal pro Minute werden Minuten- und Stundenzeiger gezeichnet, die Tageszeit berechnet und dann auf diese Grafik jede Sekunde neu der Sekundenzeiger gezeichnet.

Die Zeiger zu berechnen geht übrigens ganz einfach: Ich habe das Zifferblatt als Einheitskreis gezeichnet, also mit Radius eins und Ursprung im Zentrum. Da die Exponentialfunktion sich im Komplexen mit einer Preiode von 2*π um den Einheitskreis dreht, bietet sie sich geradezu dazu an. Der Imaginärteil der Konstanten sec beträgt 2*π/60, sodaß 60 Sekunden (Minuten) gerade eine Umdrehung ergeben. Damit die Zeiger sich richtig herum drehen, muß der berechnete Wert nur konjugiert und dann noch um 15 Sekunden bzw. Minuten verschoben werden.

```
UHR     # D9EBh    1287 Bytes

« ERASE -2,09677419355 2,09677419355 XRNG -1 1,03225806452 YRNG
GROB 63 63 000000CFF100000000000CFFFF10000000008F300EF000000000E30370E300000008F
002408F0000000C1002700C1000000E1002100C3000008320077002E00000C100000000C10000E00
0000000830000700000000007000830000000000E000810000000000C000C0000000000081006000
0000000003007000000000000700700000000000070890000000000008C0810000000000000C0C1000
000000000C1C000003237000081C000005551000081600000373700000360000055510000036000000
05537000003600000000000003300000000000006300000000000000630000000000000006B3000
00C100000E6B20000022000086B300000220000E6320000022000086B300000C100000E630000
0000000006300000000000006300000000000000660000000000000036000000000000003600000
0000000003600000000000003C000000000000081C000000000000081C10000EFF30000C181000
020020000C0890000AB820008C0070000AAD2000070070000ABA2000070060000A8820000300C000
0A882000810081000200200C00083000EFF3000E000070000000000700000E0000000008300000C10
0000000C10000832008100 2E000000E1004000C3000000C100C100C10000008F004108F000000000E
30C10E3000000008F300EF0000000000CFFFF100000000000CFF1000000
(0;,10471975512) → c0 sec
  « { # 0h # 0h } PVIEW
    WHILE 1
    REPEAT TIME DUP HMS→ 5 * DUP sec * EXP CONJ ,08 * DUP NEG ROT 15 - sec * EXP
CONJ ,7 * DUP ROT 5 PICK FP 100 * IP DUP sec * EXP CONJ ,06 * DUP NEG ROT 15 -
sec * EXP CONJ DUP ROT PICT { # 22h # 1h } c0 REPL LINE LINE LINE LINE
      IF DUP 12 <
      THEN PICT { # 3Dh # 2Eh } "A" 1 →GROB GOR
      END 100 * IP PICT { # 22h # 1h } { # 61h # 40h } SUB → t c1
      «
        WHILE TIME 100 * DUP IP t ==
        REPEAT FP 100 * IP DUP → t
          « 15 - sec * EXP CONJ DUP -,2 * PICT { # 22h # 1h } c1 REPL LINE
            WHILE t TIME 100 * FP 100 * IP ==
            REPEAT
            END
          »
        END DROP
      »
    END
  »
»
```

Matthias Rabe

```
398 ADV         413 "VA=-"      428 RCL 11      443 "MC="       452 "A=-"
399 SF 12       414 ARCL 18     429 ST+ 16      444 ARCL 14     453 ARCL 18
400 ADV         415 AVIEW       430 RCL 13      445 AVIEW       454 AVIEW
401 "MB="       416 "VD="       431 RCL 18      446 "HA="       455 "D="
402 ARCL 15     417 ARCL 18     432 +           447 ARCL 17     456 ARCL 19
403 AVIEW       418 AVIEW       433 STO 19      448 AVIEW       457 AVIEW
404 "MC="       419 CF 12       434 RCL 13      449 "HD="       458 CF 12
405 ARCL 14     420 STOP        435 ST+ 18      450 ARCL 16     459 STOP
406 AVIEW       421◆LBL "D+E"   436 "ERG. D+E"  451 AVIEW       460 END
407 "HA="       422 RCL 10      437 XEQ "PRA"
408 ARCL 17     423 ST+ 15      438 ADV
409 AVIEW       424 RCL 09      439 SF 12
410 "HD="       425 ST+ 14      440 "MB="
411 ARCL 16     426 RCL 12      441 ARCL 15
412 AVIEW       427 ST+ 17      442 AVIEW
```

Alfred Büttner
Ernst Kuhn Straße 5
8000 München 50

Verspätet

erscheint auch diese Ausgabe von PRISMA.

Es ist recht problematisch, neben den beruflichen und privaten Anforderungen immer sicherzustellen, daß PRISMA pünktlich erscheint.

Wir könnnen zwar garantieren, daß im Jahr sechs Hefte erscheinen, aber leider nicht absolut gleichmäßig übers Jahr verteilt.

Dies scheint uns erträglicher, als auf Kosten der Qualität pünktlicher zu werden. Den erreichten Standard wollen wir auf jeden Fall bewahren.

dw

Sonne, Mond und die inneren Planeten Merkur, Venus und Mars

Ephemeridenprogramm IP

von Dr. G. Heilmann

HP71B, 5087 Bytes beschriftet, 3855 Bytes unbeschriftet

IP ist ein einfaches astronomisches Programm und als Hilfe für den beobachtenden Sternfreund gedacht.

Es lassen sich berechnen:

1. die geozentrisch-äquatorialen Koordinaten,
2. die topozentrisch-äquatorialen Koordinaten des Mondes,
3. die azimuthalen Koordinaten aller genannten Körper für jeden Beobachtungsort der Erde (ausgenommen exakter Nord- und Südpol),
4. die Phasen der Planeten und des Mondes und
5. die Sternzeit der gewählten Epoche (Beobachtungsort).

Diese Fassung ist gültig für alle Epochen zwischen 01.01.1961 und 31.12.2060. Das Äquinoktium ist nicht beschränkt.

Nutation, Refraktion, geozentrische Aberration, Abweichung des Baryzentrums vom Erdmittelpunkt und die Abweichung des Geoiden von der Kugelgestalt bleiben unberücksichtigt (4).

Der Sonnenort ist für Bahnbestimmungen (Kometen) ungeeignet (siehe Newcombsche Sonnentheorie in (1)).

Der Mondort ist auf maximal 5 Bogenminuten unsicher. Für topozentrische Koordinaten entspricht dies einer Unsicherheit des Beobachtungsortes von höchstens 500 km in unseren Breiten. Damit bleibt der Aufwand im Vergleich zum Zweck in vertretbaren Grenzen.

Programmbedienung

Die Anweisung TIME muß die Greenwich-Zeit (Universal-Time) ausgeben.

Nach der Initiierung setzt IP Flag 0. Jeder weitere Programmdurchgang folgt dem festgelegten Muster, bis Flag 0 wieder gelöscht wird. Soll ein anderes Programm zwischenzeitlich benutzt werden, so muß vor erneuter Ausführung von IP Flag 0 gelöscht sein. Es empfiehlt sich, CFLAG 0 als direkt auszuführende Anweisung auf eine geeignete Taste zu legen.

Programmausführung

1. RUN IP
   Flag 0 gelöscht, nach 2.
   Flag 0 gesetzt, nach 8.
2. Display: Ort: 50.06,-8.40
   Eingabe der geographischen Koordinaten des Beobachtungsortes in Grad, Minuten und Sekunden, südliche Breite und östliche Länge negativ.
   Voreinstellung: Koordinaten von Frankfurt am Main.
   ENDLINE
3. Display Zeit: jetzt? JN?
   Mit
   "N" ENDLINE
   wird Flag 1 gesetzt. Dann fordert das Programm zu Datum und Zeit der gewählten Epoche auf.
   ENDLINE ohne Eingabe veranlaßt den Computer Datum und Zeit nach der inneren Uhr zu verwenden.
4. Display: azimuthal? JN?
   "N" ENDLINE führt nach 5.
   ENDLINE ohne Eingabe setzt Flag 4 und beendet die Initiierung.
   Der Ort des Körpers wird in lokalen azimuthalen Koordinaten ausgegeben:
   Azimuth A von Nord über Ost,
   Höhe H vom Horizont zum Zenith positiv,
   beide Koordinaten in dezimalen Grad mit einer Nachkommastelle.
   Weiter nach 8.
5. Display Mond topoz.? JN?
   Nur für den Mond wird über den Ursprung des äquatorialen Koordinatensystems entschieden, in dem der Ort ausgegeben wird:
   topozentrisch: Ursprung ist der Beobachtungsort,
   geozentrisch: Ursprung ist der Erdmittelpunkt.
   Hier sei darauf hingewiesen, daß das natürliche Bezugssystem für die Position der Sonne der Schwerpunkt des Systems Erde-Mond ist. Der Schwerpunkt liegt etwa 450 km vom Erdmittelpunkt in Richtung Mondmittelpunkt entfernt. Die dadurch auftretenden Fehler in den Orten bleiben unberücksichtigt.
   "N" ENDLINE setzt Flag 5 und führt wie
   ENDLINE ohne Eingabe nach 6.
6. Display Äquinoktium? 1950?
   Die Jahreszahl bestimmt den Anfang des zugehörigen Besseljahres als Bezugs-Äquinoktium.
   Mit der Eingabe 0 ist die Epoche Äquinoktium.
   Vorgabe: 1950
   ENDLINE
7. Display Min,Sek? JN?
   Die Ausgabe äquatorialer Koordinaten ist für zwei Formate eingerichtet.
   "N" ENDLINE
   wählt:
   Rektaszension a in Stunden und Minuten,
   Deklination d in dezimalen Grad mit einer Nachkommastelle.
   ENDLINE ohne Eingabe
8. Display S,Me,V,Ma,L: 0...4?
   Himmelskörper mit zugeordneter Zahl bestimmen.
   ENDLINE

8.1. Anzeige der vorgewählten Koordinaten.
   CONT

8.2. Anzeige der Entfernung Erde-Himmelskörper.
   CONT

8.3. Anzeige der Phase in Bruchteilen der beleuchteten Scheibe, außer Sonne,
   CONT

8.4. Anzeige der Sternzeit zur gewählten Epoche (Beobachtungsort).
   Rechenzeit je nach Status und Körper: 6,2-15,5 Sekunden.
   Die Orte von Sonne und Planeten werden besser als 0,1 Minuten in Rektaszension 30 Sekunden in Deklination ausgegeben.

Schon geringe Verbesserungen der Genauigkeit erfordern einen Aufwand, der eine Darstellung der hier geschlossenen Art verbietet. Die Bahnen der äußeren Planeten können mit vergleichender Genauigkeit nicht in dieser Form behandelt werden. Jedoch besteht beim Autor schon ein solches Programm für Jupiter und Saturn (HP41CY turbo).

Im Übrigen weise ich auf die besonderen Möglichkeiten der Anweisung ANGLE bei Transformation von räumlichen Koordinaten hin.

Literatur:

1. O. Montenbruck:
   Grundlagen der Ephemeridenrechnung. Verlag Sterne und Weltraum. München 1985.
2. Kenneth R. Lang:
   Astrophysical Formulae.
   Springer. Berlin, Heidelberg, New York 1980.
3. G. Heilmann:
   Prisma 22.4.89.
   (versehentlich unter T. Gehrmann).
4. Robin M. Green:
   Spherical Astronomy.
   Cambridge University Press 1985.
5. M. Schneider:
   Himmelsmechanik.
   BI Wissenschaftsverlag. Mannheim, Wien, Zürich 1981.

```
 10 ! IP.  Ephemeriden von Sonne, Merkur, Venus, Mars und Mond für jedes
 20 ! Äquinoktium, die azimuthalen Koordinaten jedes Beobachtungsortes
 30 ! (Default Frankfurt/Main), die Phase von Mond und Planeten und die
 40 ! Sternzeit für die Epoche. Die Uhr muß UT anzeigen. Gültig: 1961 bis 2060.
 50 ! Quelle: O.Montenbruck, Grundlagen der Ephemeridenrechnung. München 1985.
 60 ! Anordnung und Keplerroutine: Dr.Heilmann (3489), 5408 Seelbach, 1989.
 70 !
 80 ! Initierung (Beobachtungsort, Äquinoktium). Setzt sperrendes Flag 0.
 90 IF FLAG(0) THEN 210 ELSE CFLAG ALL
100 OPTION BASE 0
    @ DESTROY ALL
    @ DIM A,C,G,H,J,K,T,S,V,M,N,X,E$[1]
110 INTEGER I
    @ G=3600
    @ H=36525
    @ V=360
120 INPUT 'Ort:B,L ','50.06,-8.40';M,N
    @ M=FNR(M)
    @ N=FNR(N)
130 DISP 'Zeit:jetzt? JN';
    @ LINPUT E$
    @ IF E$="N" THEN SFLAG 1
140 DISP 'azimuthal? JN';
    @ LINPUT E$
    @ IF E$#"N" THEN SFLAG 4
    @ GOTO 210
150 DISP 'Mond topoz.? JN';
    @ LINPUT E$
    @ IF E$="N" THEN SFLAG 5
160 INPUT 'Äquinoktium?','1950';K
    @ IF K=0 THEN 180
170 K=(.31352+365.242198781*(K-1900))/H
    @ SFLAG 2
180 DISP 'Min,Sek? JN';
    @ LINPUT E$
    @ IF E$#"N" THEN SFLAG 3
190 !
200 ! Körper, Epoche.
210 SFLAG 0
    @ DIM D$[8]
    @ DISP 'S,Me,V,Ma,M: 0...4';
    @ INPUT I
220 I=ABS(I)
    @ IF 4-I<0 THEN GOTO 210
230 IF FLAG(1) THEN 240 ELSE D$=DATE$
    @ C=TIME/6
    @ GOTO 250
240 INPUT 'Datum?(jj/mm/dd)';D$
    @ INPUT 'UT?';C
    @ C=FNR(C)
250 J=FNJ(D$)-2415020
    @ T=J/H
260 S=MOD(((.0929*T+8640184.542)*T+23925.836)/G+C*1.0027379,24)
270 T=(J+C/24)/H
    @ REAL R,L,B,R0,L0,B0,R1,L1,B1
280 !
290 ! Ort.
300 DIM D$[7],F$[7]
310 IF I AND I<4 THEN GOSUB 'P' ELSE GOSUB 'S'
320 IF I=4 THEN GOSUB 'M'
    @ GOTO 340
```

```
330 CALL Q(T,L,B)
340 IF FLAG(2) THEN CALL R(T,K,L,B,G)
350 IF FLAG(4) THEN CALL H(L,B,(M),S*15-N-L,V)
    @ GOTO 400
360 L=FNS(MOD(L,V)/15)
    @ IF FLAG(3) THEN 380
370 PRINT USING '7A,"a=",2D.2D,",","d=",M2D.D';F$,L,B
    @ PAUSE
    @ GOTO 410
380 B=FNS(B)
    @ PRINT USING 390;F$[1,2],L,B
    @ PAUSE
    @ GOTO 410
390 IMAGE 2A,': ',2D.4D,', ',M2D.4D
400 PRINT USING '7A,"A=",3D.D,",","H=",M2D.D';F$,L,B
    @ PAUSE
410 IF I=4 THEN PRINT USING '7D," km"';R
    @ PAUSE
420 IF 4-I THEN PRINT USING 'D.3D,"AE= ",4D.D,"Mill.km"';R,149.6*R
    @ PAUSE
430 IF I THEN PRINT USING '"Bel.Scheibe:",D.3D';C
    @ PAUSE
440 C=FNS(MOD(S-N/15,24))
450 PRINT USING '"Sternzeit: ",2D.2D';C
    @ END ALL
460 !
470 ! Planet, heliozentrisch, Phase.
480 'P': CALL K(F$,I,T,R,L,B)
    @ CALL K(D$,0,T,R1,L1,0)
490 R0=R
    @ CALL T(R,L,B,R1,L1,0)
    @ C=FNP(R,R0,R1)
    @ RETURN
500 !
510 ! Sonne, geozentrisch-ekliptisch.
520 'S': CALL K(F$,0,T,R,L,B)
    @ L=L+180
    @ RETURN
530 !
540 ! Mond, geozentrisch/ topozentrisch, Phase.
550 'M': CALL M(R,L,B,T,G,V)
    @ CALL K(F$,0,T,R1,L1,B1)
560 F$='Mond'
    @ A=R
    @ C=L
    @ X=B
    @ L1=L1+180
    @ R1=149597892*R1
570 CALL T(R,L,B,R1,L1,B1)
    @ R0=R
    @ R=A
    @ L=C
    @ B=X
    @ C=FNP(R,R0,R1)
580 CALL Q(T,L,B)
    @ IF FLAG(5) THEN RETURN
590 A=R
    @ CALL T(R,L,B,6378.4,S-N,M)
    @ R=A
    @ RETURN
600 !
```

```
610 ! Dezimale Grad-Grad-Minuten Sekunden.
620 DEF FNS(X)
    @ A=FP(X)*60
    @ FNS=IP(X)+IP(A)/100+FP(A)*3/500
    @ END DEF
630 !
640 ! Grad, Minuten Sekunden- dezimale-Grad.
650 DEF FNR(X)
    @ A=FP(X)*100
    @ FNR=IP(X)+IP(A)/60+FP(A)/36
    @ END DEF
660 !
670 ! Datum jj/mm/dd - Julianische Tageszahl.
680 DEF FNJ(D$)
    @ A=VAL(D$[1,2])
    @ IF A>60 THEN A=A+1900 ELSE A=A+2000
690 S=VAL(D$[4,5])
    @ IF S<3 THEN S=S+12
    @ A=A-1
700 FNJ=IP(365.25*A)+IP(30.6001*(S+1))+VAL(D$[7])+1720981.5
    @ END DEF
710 !
720 ! Phase.
730 DEF FNP(R,R0,R1)=.5+(R^2+R0^2-R1^2)/(4*R*R0)
740 !
750 ! Kepler-Routine.
760 SUB K(B$,K,S,O,U,V)
    @ REAL C(10),E,M,X,Y,Z
770 RESTORE CHR$(65+K)
    @ READ B$,C()
780 'A': DATA Sonne:,1+3.8E-6,.016751,-4.2E-5,358.4758,35999.0498
790 DATA 101.2208,1.7192,0,0,0,0
800 'B': DATA Merkur:,.387099,.205614,2E-5,102.2794,149472.5123
810 DATA 28.7537,.3705,7.0029,.0019,47.146,1.185
820 'C': DATA Venus:,.723332,.006821,-4.8E-5,212.6032,58517.8039
830 DATA 54.3838,.508,3.3936,.001,75.78,.9
840 'D': DATA Mars:,1.523692,.093313,9.2E-5,319.5294,19139.8585
850 DATA 285.4322,1.0698,1.8503,-7E-4,48.786,.771
860 E=C(1)+C(2)*S
    @ RADIANS
    @ M=RAD(MOD(C(3)+C(4)*S,360))
870 !
880 ! Verfahren des Verfassers.
890 IF M=0 EXOR M=PI THEN U=M
    @ GOTO 990
900 IF M>PI THEN SFLAG 7
    @ M=2*PI-M
910 U=M
    @ V=PI
    @ IF M+E<PI THEN V=M+E
920 U=U+(U-V)/(FNK(V)/FNK(U)-1)
    @ IF ABS(FNK(U))>.0000001 THEN GOTO 920
930 IF FLAG(7,0) THEN U=2*PI-U
940 !
950 ! Kepler-Gleichung.
960 DEF FNK(X)=X-E*SIN(X)-M
970 !
980 ! Transformation Bahnebene- Ekliptik.
990 DEGREES
    @ U=DEG(U)
    @ V=C(0)*SQR(1-E^2)*SIN(U)      @ U=C(0)*(COS(U)-E)
```

```
1000 O=SQR(U^2+V^2)
     @ U=ANGLE(U,V)+C(5)+C(6)*S
     @ V=C(7)+C(8)*S
1010 X=COS(U)
     @ Y=SIN(U)*COS(V)
     @ Z=SIN(U)*SIN(V)
1020 U=ANGLE(X,Y)+C(9)+C(10)*S
     @ V=ANGLE(SQR(X^2+Y^2),Z)
     @ END SUB
1030 !
1040 ! Mondroutine.
1050 SUB M(R,L,B,T,Q,V)
     @ REAL A,C,D,E,F,G
1060 C=MOD(270.434164+(481267.883142-.001133*T)*T,V)
1070 D=MOD(296.104608+(477198.849108+.009192*T)*T,V)
1080 E=MOD(259.183275-(1934.142008-.002078*T)*T,V)
1090 A=MOD(279.696678+(36000.768925+.000303*T)*T,V)
1100 G=MOD(358.475833+(35999.04975-.00015*T)*T,V)
     @ A=2*(C-A)
1110 L=22640*SIN(D)+769*SIN(2*D)+36*SIN(3*D)-125*SIN(A/2)+2370*SIN(A)-668*SIN(G)
1120 L=L-412*SIN(2*(C-E))+212*SIN(A-2*D)+4586*SIN(A-D)+192*SIN(A+D)+165*SIN(A-G)
1130 L=MOD((L+206*SIN(A-D-G)-110*SIN(D+G)+148*SIN(D-G))/Q+C,V)
1140 P=(3423+187*COS(D)+10*COS(2*D)+34*COS(A-D)+28*COS(A)+3*COS(A+D))/Q
1150 A=C-A-E
     @ B=18520*SIN(L-E+(412*SIN(2*(C-E))+541*SIN(G))/Q)-526*SIN(A)
1160 B=B+44*SIN(A+D)-31*SIN(A-D)-23*SIN(A+G)+11*SIN(A-G)
1170 A=C-E-D
     @ B=(B-25*SIN(A-D)+21*SIN(A))/Q
     @ R=6378.14/SIN(P)
     @ END SUB
1180 !
1190 ! Transformation ekliptisch-äquatorial.
1200 SUB Q(S,U,V)
     @ REAL E,X,Y,Z
     @ E=23.4393-.013*(S-1)
     @ X=COS(U)*COS(V)
1210 Y=COS(E)*SIN(U)*COS(V)-SIN(E)*SIN(V)
     @ Z=SIN(E)*SIN(U)*COS(V)+COS(E)*SIN(V)
1220 U=ANGLE(X,Y)
     @ V=ANGLE(SQR(X^2+Y^2),Z)
     @ END SUB
1230 !
1240 ! Transformation äquatorial-horizontal.
1250 SUB H(A,D,M,S,V)
     @ REAL X,Y,Z
     @ X=COS(M)*SIN(D)-SIN(M)*COS(D)*COS(S)
1260 Y=-COS(D)*SIN(S)
     @ Z=SIN(M)*SIN(D)+COS(M)*COS(D)*COS(S)
1270 A=MOD(ANGLE(X,Y),V)
     @ D=ANGLE(SQR(X^2+Y^2),Z)
     @ END SUB
1280 !
1290 ! Vektordifferenz:(A,B,C)-(P,Q,R) (räumliche Polarkoordinaten)
1300 SUB T(A,B,C,P,Q,R)
     @ REAL X,Y,Z
     @ Z=A*SIN(C)-P*SIN(R)
1310 X=A*COS(C)*COS(B)-P*COS(R)*COS(Q)
     @ Y=A*COS(C)*SIN(B)-P*COS(R)*SIN(Q)
1320 A=SQR(X^2+Y^2+Z^2)
     @ B=ANGLE(X,Y)
     @ C=ANGLE(SQR(X^2+Y^2),Z)     @ SUB END
1330 !
1340 ! Präzession.
1350 SUB R(L,M,P,Q,G)
     @ DIM A,B,C,X,Y,Z
1360 M=M-L
     @ L=L-1
1370 A=((.018*M+.302)*M+2306.218+1.397*L)*M/G
1380 B=A+.00022*M^2
     @ C=((-.042*M-.427)*M+2004.311-.853*L)*M/G
1390 X=COS(C)*COS(Q)*COS(P+A)-SIN(C)*SIN(Q)
     @ Y=COS(Q)*SIN(P+A)
1400 Z=SIN(C)*COS(Q)*COS(P+A)+COS(C)*SIN(Q)
1410 P=B+ANGLE(X,Y)
     @ Q=ANGLE(SQR(X^2+Y^2),Z)
     @ END SUB
```

Dr. Heilmann (3489)
Obernhofer Straße 15
5408 Seelbach

HP28S SYSEVALs

Maschinensprache auf dem HP28S

von Peter Röhl

Wäre es nicht schön, wenn man den HP28 zu Leistungen bringen könnte, die ihm mit normaler Programmierung nicht zur Verfügung stehen?

Beispielsweise scheint es nicht möglich zu sein, COMMAND, UNDO, EDIT, VISIT o.ä. von einem Programm ausführen zu lassen. Dabei erscheint gerade folgender Fall sinnvoll, um UNDO zu verwenden:

```
<<....IFERR....THEN "Fehler" 1
DISP UNDO  ELSE....END >>
```

Tritt ein Fehler auf, so wird der Stack durch UNDO wieder so aufgebaut, wie er vor der Ausführung des Programmes war, egal, welche Stackmanipulation dasselbe durchgeführt hat.

Oder man versuche doch einmal einen Text und gleichzeitig ein neues Menü anzeigen zu lassen:

```
<<{A B C} MENU "Wähle A,B oder
C" 1 DISP>>
```

Scheinbar ist es nicht möglich, den Text "Wähle A,B oder C" und gleichzeitig das Menü A, B oder C anzeigen zu lassen. Scheinbar !!!!

Bevor ich jedoch verrate, wie man dies und anderes mehr auf dem HP28S realisieren kann, muß erst einmal eine Buchbesprechung her:

Customize your HP28

by W.A.C. Mier-Jedrzejowicz
ISBN 0951073311
$16.95 228 Seiten

Das Buch ist in wirklich leicht verständlichem Englisch geschrieben und in 5 Kapitel und 5 Anhänge gegliedert. Der Autor verfolgt konsequent das Ziel dem Leser zu zeigen, wie er das meiste aus seinem HP28 herausholen kann.

Das Buch behandelt alle 3 auf dem Markt befindlichen Versionen:

HP28C	#1BBh	#1CCh
HP28S	#2BBh	

Kapitel 1 beschäftigt sich mit dem allgemeinen Umgang mit dem HP28 und stellt eine recht gelungene Darstellung der Arbeitsweise des Rechners dar. Dieses Kapitel wendet sich in erster Linie an den Anfänger auf diesem Gerät.

Kapitel 2 zeigt, wie Programme kurz und effizient geschrieben werden können.

Es wird gezeigt, wie einige Probleme bei der Realisierung von Programmen behoben werden können (Bsp. Generierung von `<<..."":"...>>` auf dem HP28C).

Kapitel 3 ist dem Befehl SYSEVAL gewidmet.

Der Autor stellt dabei die allgemeine Anwendbarkeit des Befehls dar und bringt einige Beispiele (auch die SYSEVAL-Adresse von UNDO).

Kapitel 4 behandelt die Maschinenprogrammierung und gibt einen tiefen Einblick in die Wirkungsweise des Rechners.

Da der Autor dabei als erste Maschinenprogramme die Befehle PEEK und POKE realisierte, wundert nicht. Der Erforschung des HP28 steht somit nichts mehr im Wege.

Kapitel 5 behandelt die Hardware des HP28 und zeigt, wie der HP28C auf 64kByte aufgerüstet werden kann.

Anhang A listet Clubadressen, Bezugsquellen u.ä. auf.

Anhang B führt 22 Bugs auf.

Anhang C erläutert, wie Objekte und Programmstrukturen im HP28 aufgebaut sind.

Anhang D widmet sich nochmals der Maschinenprogrammierung und der Speicheraufteilung (RAM-Adressen) sowie der CPU.

Anhang E beinhaltet eine Liste der SYSEVAL-Adressen und stellt die Unterschiede der 3 Versionen dar.

Mir hat das Buch sehr gefallen, die Ausführungen sind trotz teilweise höherem Schwierigkeitsgrad der Materie leicht verständlich.

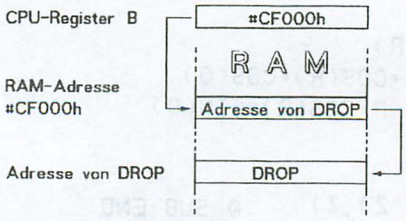
Mit dem Wissen, das dieses Buch vermittelt, fällt es dem Leser nicht schwer, seine eigenen Maschinenprogramme zu schreiben.

Folgende Mängel möchte ich an dieser Stelle kurz festhalten:

a.) Der Autor geht nicht auf die Aufgabe des CPU-Registers B ein. Dies möchte ich nun nachholen:
Im CPU-Register B ist die Adresse des Programmreturnstacks vermerkt. (Bis zur Textstelle "Zurück zum Beispiel" sollte man zum Verständnis des nun folgenden das Buch studiert haben)
Ein Beispiel verdeutlicht dies:

```
'X' <<SWAP Y DROP>>
'Y' <<+>>
```

Das Programm X ruft das Programm Y auf. Wenn Y beendet ist, kehrt der Programmzeiger zurück nach X und führt den Befehl DROP aus.
Dabei passiert folgendes:
In dem Moment, wo X Y aufruft, wird die Adresse, in der der Befehl DROP steht, in dem Register abgespeichert,



das durch das CPU-Register B spezifiziert wird, d.h. dessen Adresse im RAM in diesem CPU-Register steht.

b.) Entgegen der im HP28 üblichen Definition darf bei dem Maschinenbefehl 7hhh die Sprungweitenangabe nicht in die Sprungweite mit einbezogen werden. Dies erwähnt der Autor nicht.

c.) Will man auf dem HP28S einen Short Integer erzeugen, so muß #C53Bh SYSEVAL ausgeführt werden und nicht #71A9h SYSEVAL.

Zurück zum Beispiel
Vorweg: Ich möchte jedem empfehlen, zuerst diesen Artikel vollständig zu lesen und danach mit dem Abtippen zu beginnen; ein MEMORY LOST zerstört unwiederbringlich alle Daten im HP28, die auch nicht mehr eingelesen werden können!

Gesucht wird die Möglichkeit das Menü `{A B C}` anzeigen zu lassen, wenn gleichzeitig ein Text dargestellt wird.

Suchen wir also nach einem bekannten Befehl des Rechners, der dies bewerkstelligen kann. Nach etwas Überlegung stößt man auf den Befehl CLMF.

Dieser löscht bekanntlich die momentane Anzeige und baut anschließend die Anzeige des Stacks und danach die Menüleiste neu auf. Also ist die letzte Routine des Befehls CLMF die gesuchte, sie ist unter der Adresse #18AAAh zu finden.

Alle in diesem Artikel angegebenen SYSEVAL-Adressen gelten nur für den HP28S !

(Im oben beschriebenen Buch wird gezeigt, wie man sich den CLMF, Adresse #A792h ansehen und analysieren kann)

Führt man also #18AAAh SYSEVAL aus, so erreicht man die Anzeige des momentanen Menüs.

Ändert man das Programm also zu

```
<<{A B C} MENU "Wähle A,B oder
C" 1 DISP #18AAAh SYSEVAL>>
```

so erscheint nach dessen Ausführung der Text und das Menü.

Es wäre auch ganz praktisch, wenn man in der Anzeige eine weiß auf schwarz Darstellung ähnlich der der Kataloge realisieren könnte.

Nun ist es ja, wie im PRISMA bereits veröffentlicht, möglich, dies mit der Befehlsfolge

```
...LCD→....NOT...→LCD...
```

zu verwirklichen. Dies hat jedoch den Nachteil, daß in der Anzeige des Rechners ein wildes Geflimmer auftritt, was bei den Katalogen nicht der Fall ist.

Eine Suche nach den Katalogen im 128kByte-ROM wäre jedoch recht aufwendig, etwas überlegen hilft hier schneller zum Ziel kommen.

Die Anzeige der Betriebssystemversion geschieht mit #Ah SYSEVAL und erscheint weiß auf schwarz.

Man findet schließlich, daß #18778h SYSEVAL alle Pixel in Ebene 1 invertiert und #18789h SYSEVAL diese Invertierung wieder aufhebt.

Das Programm IDISP nutzt dies aus, es kann genauso wie der Befehl DISP verwendet werden.

Jetzt aber zur

Maschinenprogrammierung

Es gibt zwei Wege, Maschinenprogrammierung auf dem HP28 zu realisieren. Die einfachere und bequemere möchte ich nun beschreiben.

Form eines Maschinenprogramms:

69C20 Definition des Anfangs eines INLINE-MCodes

abcde Länge des Programms in Nybbles (Halbbyte); diese 5 müssen mitgezählt werden !

xy
.
.
.
.
kl } Maschinenbefehle

Unter der Adresse #02C96h steht die Definition eines INLINE-MCodes.

INLINE-MCode bedeutet, daß dieses Maschinenprogramm nicht an einer absoluten (genau festgelegten) Adresse stehen muß, sondern an jeder beliebigen Stelle im RAM oder ROM stehen darf. Somit darf ein INLINE-MCode Programm irgendwo in einem normalen Programm als Maschinensequenz auftauchen.

Wenn wir nachher ein Maschinenprogramm schreiben ist uns diese Tatsache sehr von Nutzen, denn wir können es dann dort im USER-Menü speichern, wo es uns gerade gefällt. Auch brauchen wir keine Angst zu haben, daß die Speicherorganisation verändernde Befehle wie ORDER, STO, PURGE oder CLUSR Unheil anrichten.

Die zweite Möglichkeit ein Maschinenprogramm zu schreiben besteht darin, dies unter einer festen Adresse zu speichern. Im ROM ist diese Technik sinnvoll, da sich die Organisation eines ROM (Read Only Memory) schwerlich während des Rechnerbetriebs ändern wird. Würden wir diese Technik verwenden und dabei einmal eine falsche Adresse angeben, so würde der Rechner gnadenlos abstürzen.

Jetzt ist sicher schon aufgefallen, daß unter "Die Form eines Maschinenprogrammes" 69C20 steht, während ich im anschließenden Text die Adresse #02C96h (=#2C96h, führende Nullen können weggelassen werden) für die Definition des INLINE-MCodes angegeben habe.

Ein Widerspruch? Nein, keineswegs, dabei ist einfach die Reihenfolge der Ziffern umgekehrt.

Dies liegt daran, daß der HP28 die Reihenfolge der Nibbles bei einer Abspeicherung in's RAM umdreht. Sobald ich ein "#" vor eine Zahl schreibe ist grundsätzlich die nicht verdrehte Reihenfolge gemeint.

Beispiel:
#18AAAh zeigt, wie bereits erwähnt, das Menü an.
Die Routine steht also unter der absoluten Adresse #18AAAh. Im RAM wird diese aber als AAA81 gespeichert.
Also: #18AAA≙AAA81

Und da wir ein Maschinenprogramm schreiben wollen, müssen wir 69C20 in's RAM schreiben. Liest dann die CPU bei der Ausführung des Programms diese Informationen ein, so wird durch den Lesevorgang die Reihenfolge wieder umgedreht und in der CPU steht die Information #02C96h, also richtig herum.

Da wir ein Programm schreiben wollen verwende ich hier nur die Reihenfolge, wie sie in's RAM soll: 69C20.

Unter dem 69C20 steht eine 5 Nibble lange Längenangabe des nachfolgenden Maschinenprogramms. Diese 5 Nibble müssen auch zu der Länge des Maschinenprogramms dazuaddiert werden!

Ist das Maschinenprogramm beispielsweise 10 Nibbles lang, so ergibt sich für die Längenangabe:
10+5=15

Die Längenangabe muß aber in HEX angegeben werden:
15d=#Fh

Auch diese Längenangabe muß wieder falsch herum angegeben werden:
#Fh=#0000Fh ≙ F0000

Es muß also für abcde die Ziffernkette F0000 eingesetzt werden.

Nun füge ich noch einige Maschinenbefehle hinzu und fertig ist das Maschinenprogramm!

```
          69C20  INLINE-MCode
          F0000  LEN (Länge)
          142    A=DAT0 A
10 Nibbles 164   D0=D0+5
          808C   PC=(A)
```

Zeile 1-2 Maschineneinstiegsroutine

Zeile 3-5 Maschinenausstiegsroutine

Obiges Programm besteht also nur aus einer Maschineneinstiegsroutine und einer Maschinenausstiegsroutine, hat aber sonst keinerlei Wirkung.

Bevor nun aber ein Programm mit Wirkung eingetippt werden kann brauchen wir noch eine Methode, wie wir das Programm in einer für den Rechner verständlichen Art und Weise in's RAM bekommen.

(Vorweg: sollte bei Eurem Rechner etwas anders laufen, als von mir beschrieben, so fangt hier noch einmal von vorne an!)

Dafür benötigen wir die Programme COD und ESNL:
COD codiert einen String in Ebene 1
ESNL entfernt SPACEs und NEWLINEs aus einem String

Beispiel a.)
"1424" COD führt zu "AB" (65/66d oder 41/42h)
Daß COD die Bytes verdreht ist korrekt, der Grund steht im erwähnten Buch.

Beispiel b.)
"51424" COD führt zu "PAB" (50/41/42h), weil COD vor den String in Ebene 1 (hier "51424") eine Null (="0") setzt, wenn der String eine ungeradzahlige Länge hat.

COD erwartet einen String, der rein aus Ziffern besteht, die den Programmcode darstellen. Im obigen Beispiel heißt dies "69C20F0000142164808C".

Nun kann man sich leicht vorstellen, wie schwer es sein kann, einen String von 100 Ziffern Länge fehlerfrei (!!!) einzugeben. Besser wäre die Möglichkeit beliebig viele SPACEs und NEWLINES zur Trennung in den String einfügen zu können.

Die SPACEs und NEWLINEs müssen aber vor der Ausführung von COD wieder aus dem String entfernt werden. Genau dies tut ESNL (Erase Spaces and NEWLINEs).

Zum Testen von ESNL wird einfach ein String mit einigen SPACEs und NEWLINEs sowie beliebigen anderen Zeichen eingegeben und dann ESNL ausgeführt. Danach dürfen in dem String nur noch die beliebigen anderen Zeichen vorhanden sein.

Die Programme COD und ESNL sollen im HOME-Menü stehen. Nun geben wir den String ein, der unter SCMEN im Ausdruck aufgeführt ist und speichern ihn unter dem Namen 'SCMEN' im HOME-Menü ab. Jetzt bringen wir den Text wieder in den Stack und führen ESNL aus: Zurückgegeben wird der String ohne SPACEs und NEWLINEs. Nun fertigen wir noch eine Kopie von diesem String an, indem wir einfach ENTER ausführen, der String steht nun zweimal im Stack.

Nach drücken von COD wird der String in Ebene 1 kodiert. Diesen kodierten String speichern wir nun unter dem Namen CCMEM im HOME-Menü. CCMEM muß jetzt ganz links im HOME-Menü stehen, das ist ganz wichtig ! Wenn dies der Fall ist, so steht CCMEM im TOP OF RAM, dessen Adresse #CFFFFh ist.

Um nun herauszufinden, wo der Code 69A20 im RAM steht (d.h. wo unser eingegebener kodierter Code beginnt), müssen wir nun vom TOP OF RAM plus 1 die Länge des Codes abziehen:

```
#D0000h  SWAP
```

Der Stack sieht jetzt wie folgt aus:

- in Ebene 1 steht #D0000h und
- in Ebene 2 steht der **nicht** kodierte String.

Jetzt führen wir SIZE aus, das Ergebnis in Ebene 1 muß 85 sein, ansonsten haben wir etwas falsch gemacht.

Wenn wir jetzt "-" (minus) drücken erhalten wir die Adresse , wo der Code 69A20 beginnt.

Ergebnis: #CFFABh

Jetzt führen wir SYSEVAL aus. Das Ergebnis ist eine Liste im Stack mit folgendem Aussehen:

```
{0 System Object}
```

Jetzt prüfen wir, ob die Liste im Stack wirklich nur ein Objekt, nämlich unser Maschinenprogramm, enthält.

ENTER SIZE muß Eins ergeben. Die Eins brauchen wir nicht mehr, deshalb führen wir DROP aus.

Nun setzen wir unser Maschinenprogramm frei: LIST→

Die Eins brauchen wir nicht mehr, deshalb führen wir DROP aus.

Im Stack steht jetzt nur noch unser Maschinenprogramm mit folgendem Aussehen:

```
0 System Objekt
```

'CMEN' STO speichert es unter dem Namen CMEN.

CMEN ist dem Befehl VARS vergleichbar, nur gibt es den Inhalt des Custom*menüs in Form einer Liste* zurück.

CCMEN und SCMEN brauchen wir nun nicht mehr und können sie deshalb löschen; fertig ist das Maschinenprogramm!

Aufgerufen wird es wie jedes andere Programm.

Erläuterungen:

1. TOP OF RAM ist #CF000h. Ist ein Code 1 Nibble lang und steht im TOP OF RAM, so steht er unter der Adresse #CF000h.
   Um eine Adresse zu berechnen müssen wir also #D0000h-1 = #CFFFFh rechnen. Weil der Rechner in Nibbles (1 Nibble = 4Bit = Byte) adressiert wird ergibt sich für die Adresse, in der ein n-Nibble langes Programm beginnt, zu #D0000h-n.
2. 0 System Objekt sieht zwar so aus, als ob es zwei Objekte wären; hier ist es jedoch ein einziges Objekt, da es von "unsichtbaren" Klammern zusammengehalten wird (76C20 und 09F20). Um dies verständlich zu machen gucken wir uns doch einmal an, was wir unter SCMEN eingegeben haben:

69A20 LISTE
definiert den Anfang einer Liste (das ist die, die nach #CFFABh SYSEVAL im Stack lag, weil mit dem SYSEVAL genau diese Liste ausgeführt wurde.

76C20 RPL-Prgm.
definiert den Anfang eines RPL-Programms (RPL= Reverse Polnish Lisp= Sprache des Betriebssystems des HP28).
Alle Programme beginnen damit.
Hier ist bereits die erste unsichtbare Klammer.

4B211 0
#112B4h ist die Adresse, unter der die Null definiert ist. Also legt #112B4h SYSEVAL die Null in den Stack. Der Code 4B211 wird bei der Ausführung von CMEN in den Stack gelegt. Im Stack stehen also nur die Adressen der Stackobjekte, nicht jedoch die Stackobjekte selbst, die stehen woanders.

69C20 INLINE-MCode
definiert den Anfang eines INLINE-MCode-Programms.

73000 LEN
Länge des MCodes-Programms

342D00C C=#C00D2
lädt CPU-Register C mit dem Code #C00D2h steht die Adresse der Liste, die den Inhalt des Custommenüs bestimmt.
Vergleiche: {A B C} MENU nimmt die Liste vom Stack, speichert sie, wo Platz ist und legt diese Adresse in der Adresse #C00D2h ab.
Zu bemerken ist, daß in eine Adresse nur ein Nibble paßt. Wenn man einen 5 Nibble langen Code abspeichert, so liegt er folglich verteilt auf 5 Adressen. Hier ist es #C00D2h bis #C00D6h .
Um die Beschreibung einfach zu halten rede ich nur von der Adresse, wo ein Objekt beginnt. Zum anderen verlangt der Rechner diese Anfangsadresse.
Die Längenangabe wird in diesem Programm durch den sogenannten Modifier A (das zweite A in einigen Befehlen; A steht für Adressfeld = 5 Nibbles lang) bewerkstelligt.

136 C D0 EX
vertauscht die CPU-Register C und D0.
D0 ist der Befehlszeiger.
Bsp.: Bei der Ausführung von <SWAP DROP> zeige D0 zum Beispiel auf SWAP. Bei der Ausführung von SWAP ist der Befehlszeiger D0 bereits auf DROP gesetzt.

142 A=DAT0 A
speichert im Adressfeld (das zweite A) des CPU-Registers A (das erste A) den Inhalt des Registers, das durch den Inhalt von D0 spezifiziert wurde. In D0 steht gerade #C00D2h, weshalb in A jetzt der Inhalt von #C00D2h steht.

136 C D0 EX
Zurückspeichern des ursprünglichen Inhalts von D0.

3497C81 C=#18C79
Lädt Register C mit #18C79h.
Unter dieser Adresse steht eine Liste mit 6 System Objects, die die Adresse der Routine darstellen, die den Ton einer unbelegten Taste erzeugen.
Ist das Custommenüs "Leer", so definiert diese Liste den Inhalt des Custommenüs.

8A6B0 ?A≠C A
Ist A≠C im Adressfeld der Register A und C ?
Wenn ja, dann springe #0Bh Nibbles weiter. #0Bh=11d, die 2 Ziffern "0" und "B" müssen mitgezählt werden.
B03498040DA=11 Nibbles..
Wenn nein, dann mache weiter mit der nächsten Zeile.
Wenn die Adresse in A #18C79h ist, dann muß diese Adresse durch die Adresse einer leeren Liste (#04089h) ersetzt werden, weil das Programm sonst die Liste mit 6 System Objects in den Stack legt, obwohl das Custommenü "leer" ist.

3498040 C=#04089h
lade C mit #04089h

DA A=C
kopiere C (#04089h) nach A

141 DAT1=A A
Speichere das Adressfeld (zweites A) des CPU-Registers A (erstes A) in der durch den Inhalt des Registers D1 spezifizierten Adresse. D1 ist der Stackpointer und zeigt auf das Objekt in Ebene 1.
Da wir weiter oben die Null in den Stack gelegt haben, zeigt der Pointer auf diese Null. Durch diesen Befehl (141) wird also die Adresse #112B4h in Ebene 1 durch die Adresse der Customliste überschrieben.
Wir haben die Null also nur als Dummy für das Überschreiben benötigt!

142 A=DAT0 A
schreibt nach A die Adresse des nächsten RPL-Befehls.

164 D0=D0+5
erhöht D0 um 5

808C PC=(A)
setzt den Programmcounter (Zähler) auf die Adresse, die durch den Inhalt der durch den Inhalt von A spezifizierten Adresse vorgegeben wird (Erläuterung siehe unten).

09F20 END Prgm.
Dies ist die zweite unsichtbare Klammer, die unser Programm zusammenhält.

09F20 Ende der Liste

09F20 bezeichnet allgemein das Ende eines zusammengesetzten Objekts.

Durch den Befehl 808C wird der Programm-Counter auf die Adresse gesetzt, die in der Adresse #02F90h (END Prgm. = Ende des RPL-Programms) gespeichert ist, da D0 auf die Adresse zeigte, in der der Code 09F20 stand.

(Im Buch wird dies sehr schön und lang beschrieben)

Durch unseren #CFFABh SYSEVAL führten wir die obige Liste aus. Würde man die Listendefinition (69A20 und 09F20) weglassen, (dann müßte man #CFFABh um 10d erhöht werden, da 69A20 und 09F20 genau 10d Nibbles verbrauchen) dann würde #CFFBBh SYSEVAL gleich das MCode-Programm ausführen.

Nicht sehr sinnvoll, den wir wollen das Programm unter dem Namen CMEN speichern, um es jederzeit mit diesem Namen aufrufen zu können.

Deswegen der Weg mit der Liste.

Normale Programme

a.) OSIZE gibt die Länge eines unter einem Namen gespeicherten Objektes in Bytes in den Stack zurück. Vor der Ausführung von OSIZE muß der entsprechende Name in Ebene 1 stehen. Der Zustand von LAST, COMMAND und UNDO ist dabei egal, das Ergebnis von OSIZE ist immer korrekt.

b.) Die Programme zur Linearen Regression sind, verglichen mit meiner früher in PRISMA erschienen Version, wesentlich verbessert worden. Die Bedienung ist die gleiche geblieben.
Die einzige Ausnahme bildet das Programm WAHL: Es ersetzt das Programm ANDRE.
Führt man WAHL aus, so erscheint ein Custommenü mit 6 Feldern; durch sie werden direkt die Kombinationen von
X, INV(X), LN(X)
Y, INV(Y), LN(Y)
gewählt.
Hinter einem der "X"-Felder steht ein kleines Quadrat, ebenso hinter einem der "Y"-Felder. Diese Quadrate zeigen an, welche XY-Kombination gerade eingestellt ist. Will man diese ändern, so drückt man die entsprechende Menütaste.
Ist beispielsweise X mit einem Quadrat versehen und will man LN(X) wählen, so drückt man die Menütaste unter LNX. Daraufhin erlöscht das Quadrat hinter dem X und es erscheint hinter dem LNX.
Um die zugehörige Funktion berechnen zu lassen wird einfach eine Menütaste gedrückt, deren Feld ein Quadrat enthält. Anschauen, es ist wirklich trivial.

Für INIT sind zwei verschiedene Versionen angegeben, Ihr könnt Euch die schönere von beiden aussuchen.
NOP (No Operation) erzeugt Tastennops. Das sind leere Menüfelder, die beim Drücken derselben den Ton einer nicht belegten Taste erzeugen. So können Menüeinträge voneinander getrennt werden.
Beispiel:

```
2 'A' STO NOP '6'B STO
```

in einem vorher leeren Menü ausgeführt bringt ein Menü, in dem A und B durch ein leeres Feld voneinander getrennt sind. Wird NOP mehrfach ausgeführt, so liegen mehrere leere Felder nebeneinander.
PNOP (Purge NOP) löscht diese leeren Felder einzeln wieder.
Für NOP und PNOP habe ich zwei verschiedene Versionen ausgedruckt, die aber beide gleich arbeiten, schaut sie Euch also gut an.
LN04 erzeugt man wie folgt: HOME PNOP, bis alle Tasten-nops weg sind, dann

```
#1E791h SYSEVAL DUP DUP2 4
→LIST 'LN04' STO
```

Ein anderer Weg: `LNOP 2 2 SUB DUP + DUP + 'LN04' STO`

GRUNDSÄTZLICHES

a.) Trotz großer Mühe Fehler zu vermeiden können diese niemals ausgeschlossen werden. Deshalb sind jegliche Haftung und Gewährleistung grundsätzlich ausgeschlossen. Alle Ausdrucke sind vorher probegelaufen und haben sich bei mir als fehlerfrei herausgestellt.

b.) CMEN, NOP und PNOP können meiner Erfahrung nach grundsätzlich immer aufgerufen werden, sie sind also harmlos. Es muß aber darauf geachtet werden, daß alles so eingegeben wird, wie ich es vorgegeben habe.
Dies gilt insbesondere für die SYSEVAL-Adressen, LNOP uns SCMEN.

c.) LN04, NOP, CMEN und die mit #3D43h SYSEVAL zusammengesetzten NOP und PNOP dürfen nicht mit VISIT ENTER betrachtet werden, da sonst der Interpreter aus System Object die zwei Namen 'System' 'Object' macht und den Namen " unterdrückt, d.h. die ursprüngliche Information vernichtet!!
Sollte doch einmal versehentlich VISIT aufgerufen werden, so ist der Vorgang mit ON abzubrechen, da dann der Interpreter nicht in Aktion tritt.
Gleiches gilt für EDIT.

d.) Sollte Euch einmal etwas Ungewöhnliches vorkommen (Stack voller System Objects, gestörte Anzeige etc...), so könnt Ihr versuchen, mittels SYSTEM TEST (nicht aber SYSTEM HALT, dies ist zu lasch) den Inhalt des RAM zu retten.
Meistens erfolgt kein Memory Lost und alle Daten bleiben somit erhalten. Tritt Memory Lost dennoch auf, so war es auch in keinem Fall mehr zu verhindern.
Der System Test hat "wunderbar reinigende" Wirkung, da er alle Zeiger neu justiert und somit sicher weitergearbeitet werden kann.

So, und jetzt viel Spaß

```
COD
«
  IF DUP SIZE 2 MOD
  THEN "0" SWAP +
  END "" SWAP
  WHILE DUP SIZE
  REPEAT DUP 3 OVER
SIZE SUB "#" ROT 1 2
SUB + RCLF 8 STWS
HEX SWAP STR→ RR RR
RR RR B→R CHR SWAP
STOF ROT SWAP + SWAP
  END DROP
»

157 BYTES

ESNL
« -12 SWAP 1 2
  START SWAP 22 +
SWAP
    WHILE DUP 3 PICK
CHR POS
    REPEAT LAST DUP2
1 - 1 SWAP SUB ROT
ROT 1 + OVER SIZE
SUB +
    END
  NEXT SWAP DROP
»

127 BYTES

IDISP
« SWAP # 18778h
SYSEVAL SWAP DISP
# 18789h SYSEVAL
»

58 BYTES

CMEN
0 System Object

46 BYTES

OSIZE
« RCL LAST '""' DUP
CRDIR EVAL STO MEM
CLUSR MEM - BACK
'""' PURGE NEG 16 +
»

91 BYTES

BACK
« PATH DUP SIZE 1 -
DUP SUB LIST→ DROP
EVAL
»

43.5 BYTES

FF
« 23 MENU # 25AFCh
SYSEVAL " DIR: "
PATH 2 OVER SIZE SUB
DUP SIZE 1 - DUP SUB
LIST→ DROP DUP EVAL
# 3CF16h SYSEVAL +
# 18A85h SYSEVAL
# 2385Ah SYSEVAL
# 18AAAh SYSEVAL
»

158 BYTES
```

```
SCMEN
"69A20 76C20 4B211
69C20 73000
342D00C 136 142 136
3497C81 8A6B0 3498040
DA 141 142 164 808C
09F20 09F20"
```

Lineare Regression

```
INIT
«
  IF 1 FS?C
  THEN ( →XY Σ- ΣDAT
▶ ▶▶ ) MENU "
  ELSE CLLCD CLMF 1
SF ( INIT UNDO )
MENU
"Regressionsanalyse
(lin.) der Form
Y'=m*X'+c"
1 DISP MDISP
  END
»
```

182 BYTES

```
DIR " =( ▶▶ ▶ X. INVX
LNX Y. INVY LNY X.à
INVXà LNXà Y.à INVYà
LNYà ~P ;P RO A B
WAHL BEST OUT !PKT!
PKT )

▶▶
« 1 RCLΣ DUP 'S' STO
SIZE 1 GET '"P'
CRDIR
  START " Σ- ARRY→
DROP OVER INV 3 PICK
LN ROT DUP INV OVER
LN 6 →ARRY "P Σ+
  NEXT RCLΣ CLΣ "
'"P' PURGE STOΣ ~P
»
```

142 BYTES

```
▶
« ( ) 1 RCLΣ DUP 'S'
STO SIZE 1 GET
  START Σ- ARRY→
DROP OVER INV 3 PICK
LN ROT DUP INV OVER
LN 6 →LIST +
  NEXT LIST→ 6 / 6 2
→LIST →ARRY STOΣ ~P
»
```

118 BYTES

```
X.
« 1 ;P
»
```

24.5 BYTES

```
INVX
« 2 ;P
»
```

26.5 BYTES

```
LNX
« 3 ;P
»
```

25.5 BYTES

```
Y.
« 4 ;P
»
```

24.5 BYTES

```
INVY
« 5 ;P
»
```

26.5 BYTES

```
LNY
« 6 ;P
»
```

25.5 BYTES

```
X.à
'RO'
```

13 BYTES

```
INVXà
'RO'
```

15 BYTES

```
LNXà
'RO'
```

14 BYTES

```
Y.à
'RO'
```

13 BYTES

```
INVYà
'RO'
```

15 BYTES

```
LNYà
'RO'
```

14 BYTES

```
~P
« ( BEST WAHL CORR
ISOL STEQ RCEQ PKT
!PKT! PPAR ) MENU
»
```

71.5 BYTES

```
;P
« ( A B ) OVER 4 /
IP 1 + GET STO WAHL
»
```

58 BYTES

```
RO
« ~P A B DUP2 COLΣ
OUT
»
```

42.5 BYTES

```
A
2
```

16 BYTES

```
B
4
```

16 BYTES

```
WAHL
« ( X.à INVXà LNXà
Y.à INVYà LNYà ) (
X. INVX LNX Y. INVY
LNY ) A 3 PICK OVER
GET PUT B ROT OVER
GET PUT MENU
»
```

146.5 BYTES

```
BEST
« 0 0 0 1 3
  FOR Q 4 6
    FOR R Q R COLΣ
CORR ABS DUP
      IF 5 PICK >
      THEN 4 ROLLD 3
DROPN Q R
      ELSE DROP
      END
    NEXT
  NEXT ROT DROP DUP2
COLΣ DUP2 'B' STO
'A' STO OUT
»
```

166 BYTES

```
OUT
« 3 - ( Y 'INV(Y)' '
LN(Y)' ) SWAP GET (
X 'INV(X)' 'LN(X)' )
ROT GET LR ROT *
SWAP + =
»
```

114.5 BYTES

```
!PKT!
« ΣPAR 1 4 COLΣ SCLΣ
CLLCD DRWΣ LIST→
DROP DROP2 COLΣ DRAW
DGTIZ
»
```

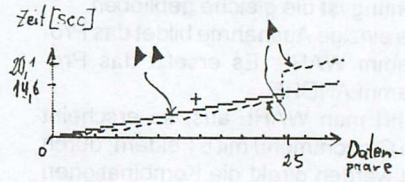
57 BYTES

```
PKT
« ΣPAR 1 4 COLΣ
CLLCD DRWΣ LIST→
DROP DROP2 COLΣ DRAW
DGTIZ
»
```

52.5 BYTES

Σ = 1431.5 BYTES

▶▶ ist ▶ vorzuziehen, wenn mehr als 24 (dies ist ein statist. Mittelwert) Datenpaare zu bewältigen sind. ▶▶ ist dann sehr viel schneller als ▶. ▶▶ braucht konstant 0.47 Sekunden pro Datenpaar.Bei bis zu etwa 10 Datenpaaren ist ▶ etwa 1.5 mal schneller als ▶▶, jedoch stimmt diese Verhältnismäßigkeit bei mehreren Datenpaaren nicht mehr, weil der Stack dann zusätzlichen Speicher braucht und somit das RAM 'gepackt' werden muß.
Folgende Graphik verdeutlicht dies.



Die Kombination von ▶ und ▶▶ :

```
▶
« 1 RCLΣ DUP 'S' STO
SIZE 1 GET
  IF DUP 10 ≥
  THEN ( ) ROT ROT
    START ▶: →LIST +
    NEXT LIST→ 6 / 6
2 →LIST →ARRY
  ELSE '"P' CRDIR
    START " ▶: →ARRY
"P Σ+
    NEXT RCLΣ CLΣ "
'"P' PURGE
  END STOΣ ~P
»
```

195 BYTES

```
▶:
« Σ- ARRY→ DROP OVER
INV 3 PICK LN ROT
DUP INV OVER LN 6
»
```

51.5 BYTES

Neues INIT :

```
INIT
«
  IF 1 FS?C
  THEN { →XY Σ- ΣDAT
▶ } MENU " CLLCD
    IFERR 'S' RCL
    THEN " S leer" 2
DISP DROP
    ELSE STOΣ Σ- →XY
    END MDISP
  ELSE CLLCD 1 SF {
INIT } LNO4 + { UNDO
} + MENU
"Regressionsanalyse
(lin.) der Form
Y'=m*X'+c"
1 DISP MDISP
  END
»
```

270.5 BYTES

Dazu stehen im HOME-Menu:

```
LNO4
{ }
```

23.5 BYTES

```
System Object ""
System Object
System Object
System Object
System Object
System Object
```

26 BYTES

```
PNOP
« LNOP 2 GET PURGE
»
```

33.5 BYTES

```
NOP
« LNOP LIST→ DROP
# 64BDh SYSEVAL
»
```

45.5 BYTES

```
LNOP
{ System Object }
```

18.5 BYTES

```
→XY
« DUP TYPE 3 -
  IF
  THEN LAST
    IF 3 +
    THEN C→R
    END 2 →ARRY
  END CLLCD
  IFERR Σ-
  THEN DUP Σ+ 1 DISP
1 2 DISP
  ELSE DUP Σ+ SWAP
DUP Σ+ 2 DISP 1 DISP
NΣ 3 DISP
  END MDISP
»
```

153.5 BYTES

```
UNDO
« 1 CF FF
»
```

29 BYTES

```
MDISP
« # 18AAAh SYSEVAL
»
```

35 BYTES

Erläuterung zu LNO4, den 6 System Objekts, PNOP, NOP, LNOP und MDISP :

In LNOP steht der RPL-Code des Tons einer unbelegten Taste, sowie der globale Name ohne Buchstaben('').
Erzeugung von LNOP :

```
# 18C79h SYSEVAL
1 1 SUB
# 1E791h SYSEVAL +
HOME 'LNOP' STO
```

NOP speichert diesen Ton unter dem Namen ''. Dabei wird die STO-Routine des Befehles STO verwendet. Diese prüft nichts. Es muß deshalb bei der Verwendung von #64BDh SYSEVAL sichergestellt werden, daß sich mindestens 2 Elemente im Stack befinden, und daß das Objekt in Ebene 1 ein Name ist.Dies ist bei NOP gewährleistet. Wird NOP mehrfach ausgeführt, so steht auch der Name '' mehrfach im USER-Menu.

PNOP löscht diese Einträge wieder.

Wird PRVAR auf den Namen '' angewendet, so wird weder der Name gedruckt noch ein anschließendes CR ausgeführt.Es wird nur das unter '' gespeicherte Objekt gedruckt.
Das in der Liste LNOP stehende System Objekt ist die Adresse #2242Ch.
(#2242Ch SYSEVAL erzeugt den Ton einer nicht belegten Taste.)
Unter #2242Ch befinden sich 9 RPL-Adressen :

```
76C20 Anfang von einem
      RPL-Programm
72650 ???
F7040 (leerer String)
72650 ???
5A0E1 ???
E0CE3 Short Integer#70d
08DE3 Short Integer
      #337d
7AA22 BEEP-Routine
09F20 Ende des RPL-Prgm
```

76C20 und 09F20 werden beim Druck unterdrückt. Somit sind die 6 System Objekts mit dem leeren String und der Länge 26 Bytes das Resultat von NOP

Die Liste in LNO4 beinhaltet 4 mal den Namen ''.

MDISP zeigt das aktuelle Menu an.

Mit einem Trick kann man LNOP und NOP bzw LNOP und PNOP zu einem Objekt zusammenfügen. Das sieht dann so aus:

```
NOP
{ System Object }
« LIST→ DROP # 64BDh
SYSEVAL
»
```

48 BYTES

```
PNOP
{ }
« 1 GET PURGE
»
```

33.5 BYTES

Das erreicht man wie folgt:

```
2: « LIST→ DROP #64BD
   SYSEVAL»
1: {System Objekt}

   (dies ist die Liste
   aus LNOP)
```

```
#3D43h SYSEVAL
```

ergibt :

```
1: { System Object }
   « LIST→ DROP # 64BDh
   SYSEVAL »
```

```
'NOP' STO
```

und

```
    LNOP 2 2 SUB
    «1 GET PURGE » SWAP
    #3D43h SYSEVAL
    'PNOP' STO
```

erzeugt PNOP.

#3D43h SYSEVAL ist eigentlich dafür bestimmt, 2 Listen zu verketten.
Also :

```
2: {LISTE2}
1: {LISTE1}
+
```

ergibt

```
1: {LISTE2 + LISTE1}
```

Durch Drücken von + wird die Routine unter #3D43h ausgeführt.
Deshalb nenne ich die Routine unter #3D43h

```
"PLUS für 2 Listen"
```

LNOP wird jetzt nicht mehr gebraucht.

Peter Röhl
Osterfeuerbergstraße 70
2800 Bremen 1

ZG_RA

Zeile 1 von ZG_RA (1-2) CCD-Barcodes Alfred Büttner

Zeile 2 von ZG_RA (3-8) CCD-Barcodes Alfred Büttner

Zeile 3 von ZG_RA (9-15) CCD-Barcodes Alfred Büttner

Zeile 4 von ZG_RA (16-28) CCD-Barcodes Alfred Büttner

Zeile 5 von ZG_RA (29-33) CCD-Barcodes Alfred Büttner

Zeile 6 von ZG_RA (34-39) CCD-Barcodes Alfred Büttner

Zeile 7 von ZG_RA (40-51) CCD-Barcodes Alfred Büttner

Zeile 8 von ZG_RA (52-64) CCD-Barcodes Alfred Büttner

Zeile 9 von ZG_RA (65-71) CCD-Barcodes Alfred Büttner

Zeile 10 von ZG_RA (72-75) CCD-Barcodes Alfred Büttner

Zeile 11 von ZG_RA (76-80) CCD-Barcodes Alfred Büttner

Zeile 12 von ZG_RA (81-84) CCD-Barcodes Alfred Büttner

Zeile 13 von ZG_RA (85-91) CCD-Barcodes Alfred Büttner

Zeile 14 von ZG_RA (92-104) CCD-Barcodes Alfred Büttner

Zeile 15 von ZG_RA (105-110) CCD-Barcodes Alfred Büttner

Zeile 16 von ZG_RA (111-114) CCD-Barcodes Alfred Büttner

Zeile 17 von ZG_RA (115-119) CCD-Barcodes Alfred Büttner

Zeile 18 von ZG_RA (120-125) CCD-Barcodes Alfred Büttner

Zeile 19 von ZG_RA (126-132) CCD-Barcodes Alfred Büttner

Zeile 20 von ZG_RA (133-138) CCD-Barcodes Alfred Büttner

Zeile 21 von ZG_RA (139-148) CCD-Barcodes Alfred Büttner

Zeile 22 von ZG_RA (149-158) CCD-Barcodes Alfred Büttner

Zeile 23 von ZG_RA (159-167) CCD-Barcodes Alfred Büttner

Zeile 24 von ZG_RA (168-176) CCD-Barcodes Alfred Büttner

Zeile 25 von ZG_RA (177-180) CCD-Barcodes Alfred Büttner

Zeile 26 von ZG_RA (181-183) CCD-Barcodes Alfred Büttner

Zeile 27 von ZG_RA (184-189) CCD-Barcodes Alfred Büttner

Zeile 28 von ZG_RA (190-196) CCD-Barcodes Alfred Büttner

Zeile 29 von ZG_RA (197-202) CCD-Barcodes Alfred Büttner

Zeile 30 von ZG_RA (203-208) CCD-Barcodes Alfred Büttner

Zeile 31 von ZG_RA (209-211) CCD-Barcodes Alfred Büttner

Zeile 32 von ZG_RA (212-215) CCD-Barcodes Alfred Büttner

Zeile 33 von ZG_RA (216-221) CCD-Barcodes Alfred Büttner

Zeile 34 von ZG_RA (222-227) CCD-Barcodes Alfred Büttner

Zeile 35 von ZG_RA (228-234) CCD-Barcodes Alfred Büttner

Zeile 36 von ZG_RA (235-244) CCD-Barcodes Alfred Büttner

Zeile 37 von ZG_RA (245-249) CCD-Barcodes Alfred Büttner

Zeile 38 von ZG_RA (250-254) CCD-Barcodes Alfred Büttner

Zeile 39 von ZG_RA (255-259) CCD-Barcodes Alfred Büttner

Zeile 40 von ZG_RA (260-265) CCD-Barcodes Alfred Büttner

Zeile 41 von ZG_RA (266-272) CCD-Barcodes Alfred Büttner

Zeile 42 von ZG_RA (273-279) CCD-Barcodes Alfred Büttner

Zeile 43 von ZG_RA (280-290) CCD-Barcodes Alfred Büttner

Zeile 44 von ZG_RA (291-297) CCD-Barcodes Alfred Büttner

Zeile 45 von ZG_RA (298-310) CCD-Barcodes Alfred Büttner

Zeile 46 von ZG_RA (311-315) CCD-Barcodes Alfred Büttner

Zeile 47 von ZG_RA (316-320) CCD-Barcodes Alfred Büttner

Zeile 48 von ZG_RA (321-325) CCD-Barcodes Alfred Büttner

Zeile 49 von ZG_RA (326-331) CCD-Barcodes Alfred Büttner

Zeile 50 von ZG_RA (332-338) CCD-Barcodes Alfred Büttner

Zeile 51 von ZG_RA (339-345) CCD-Barcodes Alfred Büttner

Zeile 52 von ZG_RA (346-357) CCD-Barcodes Alfred Büttner

Zeile 53 von ZG_RA (358-370) CCD-Barcodes Alfred Büttner

Zeile 54 von ZG_RA (371-380) CCD-Barcodes Alfred Büttner

Zeile 55 von ZG_RA (381-388) CCD-Barcodes Alfred Büttner

Zeile 56 von ZG_RA (389-394) CCD-Barcodes Alfred Büttner

Zeile 57 von ZG_RA (395-399) CCD-Barcodes Alfred Büttner

Zeile 58 von ZG_RA (400-405) CCD-Barcodes Alfred Büttner

Zeile 59 von ZG_RA (406-411) CCD-Barcodes Alfred Büttner

Zeile 60 von ZG_RA (412-416) CCD-Barcodes Alfred Büttner

Zeile 61 von ZG_RA (417-425) CCD-Barcodes Alfred Büttner

Zeile 62 von ZG_RA (426-428) CCD-Barcodes Alfred Büttner

Zeile 63 von ZG_RA (429-433) CCD-Barcodes Alfred Büttner

Zeile 64 von ZG_RA (434-438) CCD-Barcodes Alfred Büttner

Zeile 65 von ZG_RA (439-444) CCD-Barcodes Alfred Büttner

Zeile 66 von ZG_RA (445-451) CCD-Barcodes Alfred Büttner

Zeile 67 von ZG_RA (452-452) CCD-Barcodes Alfred Büttner

PD

Zeile 1 von PD (1-6) CCD-Barcodes Dr.G.Heilmann

Zeile 2 von PD (7-15) CCD-Barcodes Dr.G.Heilmann

Zeile 3 von PD (16-19) CCD-Barcodes Dr.G.Heilmann

Zeile 4 von PD (20-30) CCD-Barcodes Dr.G.Heilmann

Zeile 5 von PD (31-41) CCD-Barcodes Dr.G.Heilmann

Zeile 6 von PD (42-51) CCD-Barcodes Dr.G.Heilmann

Zeile 7 von PD (52-55) CCD-Barcodes Dr.G.Heilmann

Zeile 8 von PD (56-62) CCD-Barcodes Dr.G.Heilmann

Zeile 9 von PD (63-63) CCD-Barcodes Dr.G.Heilmann

PLAMBA

Zeile 1 von PLAMBA (1-2) CCD-Barcodes Alfred Büttner

Zeile 2 von PLAMBA (3-7) CCD-Barcodes Alfred Büttner

Zeile 3 von PLAMBA (8-12) CCD-Barcodes Alfred Büttner

Zeile 4 von PLAMBA (13-20) CCD-Barcodes Alfred Büttner

Zeile 5 von PLAMBA (21-33) CCD-Barcodes Alfred Büttner

Zeile 6 von PLAMBA (34-46) CCD-Barcodes Alfred Büttner

Zeile 7 von PLAMBA (47-59) CCD-Barcodes Alfred Büttner

Zeile 8 von PLAMBA (60-72) CCD-Barcodes Alfred Büttner

Zeile 9 von PLAMBA (73-77) CCD-Barcodes Alfred Büttner

Zeile 10 von PLAMBA (78-84) CCD-Barcodes Alfred Büttner

Zeile 11 von PLAMBA (85-89) CCD-Barcodes Alfred Büttner

Zeile 12 von PLAMBA (90-96) CCD-Barcodes Alfred Büttner

Zeile 13 von PLAMBA (97-96) CCD-Barcodes Alfred Büttner

DANDY

Zeile 1 von DANDY (1-3) CCD-Barcodes Rainer Gillmann

Zeile 2 von DANDY (4-7) CCD-Barcodes Rainer Gillmann

Zeile 3 von DANDY (8-16) CCD-Barcodes Rainer Gillmann

Zeile 4 von DANDY (17-19) CCD-Barcodes Rainer Gillmann

Zeile 5 von DANDY (20-26) CCD-Barcodes Rainer Gillmann

Zeile 6 von DANDY (27-32) CCD-Barcodes Rainer Gillmann

Zeile 7 von DANDY (33-37) CCD-Barcodes Rainer Gillmann

Zeile 8 von DANDY (38-44) CCD-Barcodes Rainer Gillmann

Zeile 9 von DANDY (45-54) CCD-Barcodes Rainer Gillmann

Zeile 10 von DANDY (55-61) CCD-Barcodes Rainer Gillmann

Fortsetzung Seite 48

Verkaufe: HP-41CY (HP-41CX mit eingebauter 64k-Rambox von W&W Software Products), wenig gebraucht, für DM 790.-.
Joachim Wetzlar (2271),
✆ 0241/87 24 06

Suche dringend für HP-41 **Magnetkartenleser**, Preis VS.
✆ 05141/835 13

Suche dringend **Navigationsmodul** für HP-41.
Georg Klare, Tarpenbekstraße 90, 2000 Hamburg 20, ✆ 040/460 61 69 oder 06102/538 70.

Verkaufe CPU 80 387 16/20/25 MHz, DM 550.-/650.-/900.-.
Georg Klare, ✆ 040/460 61 69 oder 06102/538 70.

Hilfe:
Wer kann mir die Programmsammlung "**Fluid Dynamics and Hydraulics**" (00041-90139) für kurze Zeit zur Verfügung stellen?
Suche für HP-41CX Programm zur Berechnung von **Einfeldträgern** mit beliebiger Belastung (auch Momente im Feld!), Stabendmomenten und Ergebnisausgabe an beliebiger ~~Stelle~~.
Jürgen Mang (3510), Lindener Straße 7, 6751 Krickenbach,
✆ 06307/525, abends.

Verkaufe HP-110 mit DiscDrive und ThinkJet, mit HP-IL-Kabel; VB DM 4.000.-. Bitte melden unter ✆ 05941/83 74.

Suche für HP-75:
1. Assembler mit dazugehörenden Source-/LEX-Files auf Diskette (9114 B) oder Magnetkarten + Anleitungen
2. sämtliche für diesen Rechner erhältlichen LEX-Files auf Diskette oder Magnetkarten

Preisvorstellung für 1. und 2. (nur Originale) max. DM 120.-.
Wolfgang Schäfer (3151),
✆ 06841/713 69 ab 19.00 Uhr.

Verkaufe (Systemwechsel):

HP-71B mit IL-Modul	DM	600
32 kB RAM	DM	150
Mathemodul	DM	150
Forth-Assembler	DM	150
Kassettenlaufwerk	DM	250
HP 82973A (PC Karte)	DM	300
ThinkJet (IL)	DM	450
IDS Band 1	DM	50
PPC-Journal ('85-Mai '87)	DM	100
komplett mit jeder Menge Software für	DM	2000

Stephan Trzeciak, Eginhardstraße 22, 5100 Aachen, ✆ 0241/15 39 25.

HP-41CX	DM	300.-
64 kB Eramco RAM-Storage-Unit für 41er	DM	400.-
IL-Modul HP 82160 A	DM	100.-
PC-IL-Schnittstellenkarte HP 82973 A inkl. LINK	DM	250.-
IL-Kassettenlaufwerk HP 82161 A (Akku neu)	DM	350.-
IL-Thermodrucker HP 82162 A (Akku neu)	DM	300.-
IL-Video-Interface HP 82163 A	DM	100.-
Magnetkartenleser HP 82104 A	DM	150.-
12" Phillips Monitor 80	DM	100.-
Barcode-Lesestift HP 82153 A	DM	100.-
W&W CCD-Modul	DM	100.-
Mathematik-Modul HP 5061-5261	DM	20.-
Advantage Pac HP 82181 A	DM	100.-
Doppel-XMemory-Modul (1 * OK, 1* defekt)	DM	70.-

Literatur von Dearing, Jarret, Meschede, Albers, Wickes.
Peter Farwig, Schwanenburgstraße 50, D-4500 Osnabrück, ✆ 0541/70 70 85 ✆

Suche HP-41CX, suche Modul Advantage Pac 00041-15057.
Günther Schwarz,
✆ 05527/22 86.

HP150 Komplettausstattung (MS-DOS):
HP150-2, 512k , Touch-Screen-Option, Hard-Disc 20 MByte mit 3,5" Floppydrive, viel Software worunter Memomaker, Personal Card File und (original verpackt:) 'Picture Perfect", ThinkJet-IB-Printer.
Zusammen DM 2.500.- VB.
Weiterhin HP-71B, Version 2CDCC, DM 400.-.
HP-41CV mit Akku DM 150.-.
HP-16C DM 50.-.
Kassettenlaufwerk HP 82161 A defekt DM 100.-.
HP-125 mit Dual-Floppydrive mit kleinen Fehlern DM 150.-.
Gysbert Hagemann, Alter Weg 1, D-6653 Blieskastel 2, geschäftszeitlich ✆ 06842/30 41/30 42, privat ✆ 06842/28 05

HP-71B System:

HP-71B mit IL-Modul und 24 KB-RAMs	DM	780
Kassettenlaufwerk mit Netzteil	DM	500
Advanced PAC Screen mit Netzteil	DM	250
Advanced PAC Screen 2.0 mit Netzteil und Maus	DM	400
12" Monitor bernstein passend für PAC-Screen	DM	150
2 IL-Konverter je	DM	250
1 Paar IL-Kabel 5m	DM	35
HP41 Translator ROM	DM	120
Workbook 71 ROM	DM	180
2 weitere Netzteile für IL-Geräte je	DM	25

Alle IL-Geräte mit original Handbüchern und je 1 IL-Kabel.
Walter Becker, Buschkampen 41, 2850 Bremerhaven,
✆ 0471/565 53 (abends).

Verkaufe: Taschenrechner Texas Instruments **TI** Programmable **58** C, incl. Standard Software Modul, mit neuen NICd Accus. Magnetkarten für TI 59.
A. Thesen, Im Schwalg 14, 6530 Bingen, ✆ 06721/327 53

Hp-41CV, CCD-Modul, IL-Modul, Digitalkassettenlaufwerk + 15 Kassetten für DM 550.-.
Dennis Föh (2374), Hermann-Hanker-Straße 17, 3400 Göttingen, ✆ 05551/79 28 57

Verkaufe:
IL Development Modul DM 70.-
Suche:

IL Digitalmultimeter	HP 3468
IL Video-Interface	HP 82163
IL Kassettenlaufwerk	HP 82161
IL Diskettenlaufwerk	HP 9114
Extended I/O-Modul	*HP 82183*

Karlheinz Jünemann (1072), Rugewisch 18, 2000 Hamburg 63,
✆ 040/532 11 11

Verkaufe Nixdorf LapTop 8810 M15 80286 CPU, 20 MB HD, 3 1/2" FD, 640 KB RAM etc.
Georg Klare, ✆ 040/460 61 69 oder 06102/538 70.

Suche: "Tricks, Tips und Routinen für Taschenrechner der Serie HP-41" von J. Dearing.
Bernd Koschel, Jakob-Henle-Straße 1, 3400 Göttingen.

Suche Barcodeleser für HP-41.
Joachim v. Sprang,
✆ 0212/59 23 44.

DDR-Kontakte DDR-Kontakte
Interesse an den Betriebssystemen MS-DOS und UNIX sowie Naturwissenschaften:
Rolf Wutzler
Friedrich-Staude-Straße 25
DDR 9591 Zwickau

PC-MS-DOS-Anwender sucht Kontakte zwecks Erfahrungsaustausch zum Einsatz in der BRD-"gängigen" Software, speziell, aber nicht nur, E-Technik:
Horst Schreiter
Clara Zetkin Straße 04
DDR 7263 Mueglen

Erfahrungsaustausch und Software sucht:
Andreas Heidrich
Leninstraße 30
DDR 9580 Zwickau

DDR-Kontakte DDR-Kontakte

Funktion:

CONFMT (=Concurrent Formatter) ist ein leistungsfähiger Diskettenformatierer. CONFMT ist speicherresident geladen, kann per Tastendruck aktiviert werden und formatiert dann im Hintergrund.

Typ:

Speicherresident, wieder entfernbar

Format:

```
CONFMT [S]

CONFMT R
```

Hinweise:

Vor der Benutzung muß CONFMT installiert werden, dies geschieht mit dem Programm INSTALL (ohne Parameter aufrufen). XTs, die mit einem Kontroller ausgestattet sind, der High Density Laufwerke bedienen kann, sollten bei der Installation als AT angegeben werden. Ändert sich die Laufwerkskonfiguration, muß CONFMT neu installiert werden.
Der Parameter S bewirkt, daß die COPYRIGHT-Meldung beim Start übergangen wird (nur bei lizensierten Versionen).
CONFMT R entfernt CONFMT wieder aus dem Speicher.

Quelle:

CONFMT Version 1.04 ist ein SHAREWARE-Programm. Copyright (C) 1989 by Sydex

Funktion:

COPYQM ist ein leistungsstarkes Diskettenkopierprogramm.
COPYQM formatiert automatisch, kann Mehrfachkopien erstellen, erkennt Diskettenwechsel automatisch, arbeitet mit allen IBM-Formaten, auch 1.2 Mb und 1.44 Mb.
COPYQM kann in expanded und extended Memory sowie auf Platte zwischenspeichern, wenn die Diskette nicht in den Hauptspeicher passt.
COPYQM kann Imagedateien von Diskette auf Platte speichern und wieder abrufen.

Format:

COPYQM Laufwerke [Optionen]
COPYQM PROMPT
COPYQM HELP

Hinweise:

Vor der Benutzung muß COPYQM installiert werden, dies geschieht mit dem Programm CQINSTAL (ohne Parameter aufrufen). Ändert sich die Laufwerkskonfiguration, muß COPYQM neu installiert werden.
COPYQM HELP gibt einen Hilfebildschirm aus.
COPYQM PROMPT fragt interaktiv nach Laufwerken und einigen Optionen.
COPYQM Laufwerke [Optionen] ist der normale Aufruf.
Es können mehrere Laufwerke angegeben werden. Es gibt kein Quell- und Ziellaufwerk, wie bei anderen Programmen. COPYQM liest immer die ganze Quelldiskette und schreibt dann die angegebene Zahl Zieldisketten. Werden mehrere Laufwerke angegeben geht COPYQM sie beim Schreiben der Reihe nach durch.
Folgende Optionen sind möglich:
COUNT=n (Abkürzung: C=n) gibt die Zahl der Kopien an, die von der Quelldiskette gezogen werden, wobei n für die Anzahl der Kopien steht.
QUICK (Abk.: Q) bewirkt, daß nur der von Dateien benutzte Bereich auf der Zieldiskette formatiert wird. Dadurch wird das Kopieren schneller, die Zieldiskette ist aber nur mit Einschränkungen verwendbar !
VERIFY=NONE (Abk.: V=N) bewirkt, das die Zieldiskette nicht überprüft wird.
VERIFY=DATA (Abk.: V=D) bewirkt, daß die Zieldiskette nur im benutzten Bereich überprüft wird (Voreinstellung).
VERIFY=ALL (Abk.: V=A) bewirkt, daß die gesamte Zieldiskette überprüft wird.

OVERFLOW=E (Abk.: O=E) bewirkt, daß bei großen Disketten der EMS-Speicher zur Pufferung benutzt wird (damit auch große Disketten auf EINMAL eingelesen werden können).
OVERFLOW=X (Abk.: O=X) bewirkt, daß der extended memory (AT >1Mb) zur Pufferung benutzt wird.
OVERFLOW=Hd (Abk.: O=Hd) bewirkt, daß die Festplatte (Laufwerk d) zu Pufferung benutzt wird.
SILENT (Abk.: S) schaltet die Piepstöne zum Diskettenwechsel ab.
NORULER (Abk.: N) schaltet den Balken ab, der das Fortschreiten der Kopie anzeigt.
RECORD=dateiname
PLAYBACK=dateiname
Mit diesen beiden Parametern kann man Disketten in eine Imagedatei ausgeben und von einer Imagedatei eine Diskette erstellen. RECORD=dateiname liest eine Diskette in die angegebene Datei, PLAYBACK=dateiname erstellt eine Diskette aus der angegebenen Datei.
Auf diese Weise kann man z.B. Disketten 1:1 über DFÜ verschicken, indem man eine Imagedatei verschickt (Das geht natürlich nur dann, wenn der Partner auch über COPYQM verfügt).
COPYQM kann die Zieldisketten auch mit Seriennummern versehen. Näheres steht in der englischen Originaldokumentation.

Beispiele:

COPYQM A: V=A COUNT=100 O=HC

Kopiert auf dem Laufwerk A:, überprüft die gesamte Diskette, erstellt 100 Kopien und benutzt die Festplatte C: zur Pufferung.

COPYQM A: RECORD=C:\TEST\XY_DISK.IMG

Liest eine Diskette von Laufwerk A: in die Datei C:\TEST\XY_DISK.IMG ein.

COPYQM PLAYBACK=D:INFO46.IMG O=HC COUNT=290

Liest aus der Datei D:INFO46.IMG und erstellt davon 290 Kopien, wobei die Festplatte C: zur Pufferung benutzt wird.

Funktion:

FORMATQM ist ein leistungsfähiger Diskettenformatierer.

Format:

FORMATQM Laufwerke Format [N] [S]

Hinweise:

Vor der Benutzung muß FORMATQM installiert werden, dies geschieht mit dem Programm FQINSTAL (ohne Parameter aufrufen). Bei XTs, die mit einem Kontroller ausgestattet sind, der High Density Laufwerke bedienen kann, sollten bei der Installation als AT angegeben werden. Ändert sich die Laufwerkskonfiguration, muß FORMATQM neu installiert werden.
Es können mehrere Laufwerke angegeben werden. FORMATQM formatiert dann der Reihe nach die Laufwerke durch.
Folgende Formate können angegeben werden:

160K	DOS 1.x, einseitig
180K	DOS ab 2.0, einseitig
320K	DOS 1.x, zweiseitig
360K	DOS ab 2.0, zweiseitig
720K	Microsoft 720k Format (2 Sektoren/Cluster)
I720K	IBM DOS 3.3, 720k Format (1 Sektor/Cluster)
1200K	DOS ab 3.0, High Density
1440K	DOS ab 3.3, 3.5" High Density

Es können natürlich nur solche Formate angegeben werden, die auch vom Laufwerk unterstützt werden. Bei 720K gibt es zwei Formate, wobei das Microsoft-Format das gebräuchlichere ist.
Der Parameter N bewirkt, das die Diskette nicht überprüft wird.
Der Parameter S bewirkt, daß die COPYRIGHT-Meldung beim Start übergangen wird (nur bei lizensierten Versionen).

Beispiele:

FORMATQM A: 1440K
Formatiert 1,44 Mb Disketten in Laufwerk A:

FORMATQM A: B: I720K N
Formatiert 720k Disketten im IBM-Format abwechselnd in den Laufwerken A: und B:, wobei die formatierten Disketten nicht überprüft werden.

SERVICELEISTUNGEN

BEST OF PRISMA
Schutzgebühr: 30,- DM

Nachsendedienst PRISMA
Schutzgebühr: 5,- DM pro Heft für Jahrgänge 1982-86
10,- DM pro Heft für Jahrgänge ab 1987

Inhaltsverzeichnis PRISMA
Schutzgebühr: 3,- DM in Briefmarken

Programmbiliothek HP-71
Die bislang in PRISMA erschienenen Programme können durch Einsenden eines geeigneten Datenträgers (3,5" Diskette, Digitalkassette oder Magnetkarte) und eines SAFU angefordert werden.

MS-DOS Inhaltsverzeichnis
Kann durch das Einsenden einer formatierten 360 kB oder 1,2 MB 5,25"-Diskette oder einer formatierten 720 kB oder 1,44 MB 3,5"-Diskette und einem SAFU angefordert werden.

ATARI Inhaltsverzeichnis
Kann durch das Einsenden einer 3,5"-Diskette + SAFU bei Werner Müller angefordert werden.

UPLE
Das UPLE-Verzeichnis mit der Kurzbeschreibung der einzelnen Programme sowie den Bezugsbedingungen kann gegen Einsenden von DM 10,- in Briefmarken angefordert werden.

Programme aus BEST OF PRISMA

a) Eine Kopie der Programme von BEST OF PRISMA auf Kassette erfordert das Beilegen einer Leerkassette und eines SAFU.

b) Für Barcodes von BEST OF PRISMA-Programmen gibt es folgendes Verfahren:
Schickt eine Liste mit den Namen und der Seitenangabe (der Barcodeseiten) an die Clubadresse, pro Barcodeseite legt bitte 40 Pf., plus 2,40 DM für das Verschicken, in Briefmarken bei. Die Liste der verfügbaren Programme ist in Heft 3/88 auf der Seite 35 abgedruckt, sie kann gegen einen SAFU angefordert werden.

Der Bezug sämtlicher Clubleistungen erfolgt über die Clubadresse, soweit dies nicht anders angegeben ist, oder telefonisch bei Dieter Wolf:

(069) 76 59 12

Die eventuell anfallenden Unkostenbeiträge können als Verrechnungsscheck beigelegt werden, Bargeld ist aus Sicherheitsgründen nicht zu empfehlen; ist dies nicht der Fall, so wird Rechnung gestellt, dies macht die Sache natürlich nicht unbedingt einfacher, bzw. schneller.
Formvorschriften für Schreiben an die Clubadresse gibt es keine; das Schreiben kann durchaus handschriftlich verfasst sein, ein normaler Sterblicher sollte es noch lesen können. Vor allem den Absender und die Mitgliedsnummer deutlich schreiben !
(SAFU = Selbst Adressierter Freiumschlag)

CLUBADRESSEN

1. Vorsitzender
Gerhard Link (3107),
Postfach 1615, 6090 Rüsselsheim,
✆ (06142) 81 51 0, Fax: (06142) 81 57 9, GEO1:G.LINK

2. Vorsitzender
Alf-Norman Tietze (1909),
Sossenheimer Mühlgasse 10,
6000 Frankfurt 80, ✆ (069) 34 62 40, GEO1:A.N.TIETZE

Schatzmeister
Dieter Wolf (1734),
Pützerstraße 29, 6000 Frankfurt 90,
✆ (069) 76 59 12, GEO1:D.WOLF

1. Beisitzer
Norbert Resch (2739),
Tsingtauerstraße 69, 8000 München 82

2. Beisitzer
Werner Dworak (607),
Allewind 51, 7900 Ulm,
✆ (07304) 32 74, GEO1:W.DWORAK

MS-DOS Service / Beirat
Alexander Wolf (3303),
Pützerstraße 29, 6000 Frankfurt 90,
✆ (069) 76 59 12

ATARI Service / Beirat
Dr. Werner Müller (1865),
Schallstraße 6, 5000 Köln 41,
✆ (0221) 40 23 55, MBK1:W.MUELLER

Regionalgruppe Berlin
Jörg Warmuth (79), Wartburgstraße 17, 1000 Berlin 62

Regionalgruppe Hamburg
Alfred Czaya (2225), An der Bahn 1, 2061 Sülfeld,
✆ (040) 43 36 68 (Mo.-Do. abends)
Horst Ziegler (1361), Schüslerweg 18b, 2100 Hamburg 90,
✆ (040) 79 05 67 2

Regionalgruppe Karlsruhe / Beirat
Stefan Schwall (1695), Rappenwörthstraße 42,
7500 Karlsruhe 21, ✆ (0721) 57 67 56, GEO1:S.SCHWALL

Regionalgruppe Rheinland/Ruhrgebiet
Jochen Haas (2874), Roßstraße 27, 5000 Köln 30,
✆ (0221) 51 98 70

Regionalgruppe München / Beirat
Victor Lecoq (2246), Seumestraße 8, 8000 München 70,
✆ (089) 78 93 79

Regionalgruppe Rhein-Main
Andreas Eschmann (2289), Lahnstraße 2, 6906 Raunheim,
✆ (06142) 46 64 2

Beirat
Manfred Hammer (2742), Oranienstraße 42, 6200 Wiesbaden

Beirat
Peter Kemmerling (2466), Danziger Straße 17, 4030 Ratingen

Beirat
Martin Meyer (1000), Robert-Stolz-Straße 5, 6232 Bad Soden 1

CP/M-80
Peter-C. Spaeth, Michaeliburgstraße 4, 8000 München 80

E-Technik
Werner Meschede (2670), Sorpestr. 4, 5788 Siedlingshausen

Grabau GR7 Interface
Holger von Stillfried (2641), Am Langdiek 13, 2000 Hamburg 61

Hardware 41
Winfried Maschke (413), Ursulakloster 4, 5000 Köln 1,
✆ (0221) 13 12 97

HP-71 Assembler (LEX-Files)
Matthias Rabe (2062), Teichsheide 13, 4800 Bielefeld,
GEO1:M.RABE

Mathematik
Andreas Wolpers (349), Steinstraße 15, 7500 Karlsruhe

Naturwissenschaften
Thor Germann (3423), Hobeuken 18, 4322 Spockhövel 2,
✆ (02339) 39 63

Serie 80
Klaus Kaiser (1661), Mainzer Landstr. 561, 6230 Frankfurt 80,
✆ (069) 39 78 52

Vermessungswesen
Ulrich Kulle (2719), Memeler Straße 26, 3000 Hannover 51,
✆ (0511) 60 42 72 8

Programmbibliothek HP-71
Henry Schimmer (786), Homburger Landstr. 63,
6000 Frankfurt 50

"Clubadresse"
CCD e.V., Postf. 11 04 11, 6000 Frankfurt 1, ✆ (069) 76 59 12

Postvertriebsstück
Gebühr bezahlt

D 2856 F

CCD - Computerclub Deutschland e.V.
Schwalbacher Straße 50
D-6000 Frankfurt am Main 1

CCD
ISSN 0176-8735
PRISMA
März / April 1990 Nr. 2

Ortsgruppe Köln

Die bisherige Ortsgruppe Köln hat ihren Sitz nach Essen verlagert; nun ist die Bezeichnung treffender Regionalgruppe

Rheinland/Ruhrgebiet

Die nächsten CCD-Treffen finden jeweils am zweiten Sonntag eines jeden Monats um 15.00 Uhr statt.

Die genauen Termine lauten:
10.06.90 Juni
08.07.90 Juli
12.08.90 August
09.09.90 September
14.10.90 Oktober
11.11.90 November
09.12.90 Dezember

Die Anschrift unseres Treffpunktes ist:
Essen 1 - Rüttenscheid
Witteringstraße 24 (Hinterhof)

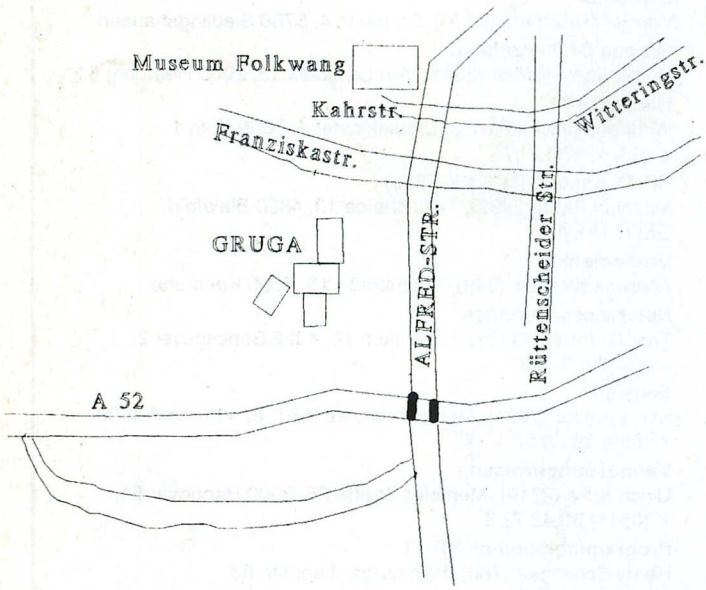
Für die Anfahrt ist euch hoffentlich die grobe Skizze behilflich.

Aufgrund der Nähe zu GRUGA und Folkwang Museum besteht auch bei Benutzung öffentlicher Verkehrsmittel gute Verbindung.

Die jeweils beste Anbindung bei eigenem PKW, müßt Ihr wegen der Vielzahl möglicher Routen selbst ermitteln.

Es empfiehlt sich z.B. über A3 von Köln kommend, auf die A52 zu fahren, und die Autobahn an der Abfahrt Essen/Rüttenscheid zu verlassen.

Fahrt dann über **Alfredstraße**, vorbei an der **GRUGA**. In Höhe des **Folkwang Museums**, fahrt Ihr **rechts** in die **Kahrstraße** ab. Ihr trefft auf die **Witteringstraße** als Verlängerung dieser Straße. Der eigentliche Treffpunkt liegt im **Hinterhof** (bitte geht durch die Einfahrt) des Hauses **24**.



Unsere Ortsgruppe befaßt sich mit HP-Taschencomputern, behandelt aber mittlerweile auch sehr viel MS-DOS, also IBM-Maschinen.

Impressum

Titel:
PRISMA

Herausgeber:
CCD - Computerclub Deutschland e.V.

Postfach 11 04 11
Schwalbacher Straße 50
6000 Frankfurt 1

Verantwortlicher Redakteur:
Alf-Norman Tietze (ant)

Redaktion:
Klaus Kaiser (kk)
Michael Krocker (mik)
Martin Meyer (mm)
Dieter Wolf (dw)

Herstellung:
CCD e.V.

Manuskripte:
Manuskripte werden gerne von der Redaktion angenommen. Honorare werden in der Regel nicht gezahlt. Die Zustimmung des Verfassers zum Abdruck wird vorausgesetzt. Für alle Veröffentlichungen wird weder durch den Verein noch durch seine Mitglieder eine irgendwie geartete Garantie übernommen.

Druck und Weiterverarbeitung:
Reha Werkstatt Rödelheim
Biedenkopfer Weg 40 a, 6000 Frankfurt

Anzeigenpreise:
Es gilt unsere Anzeigenpreisliste 3 vom Juni 1987

Erscheinungsweise:
PRISMA erscheint jeden 2. Monat

Auflage:
2500

Bezug:
PRISMA wird allen Mitgliedern des CCD ohne Anforderung übersandt. Ein Anspruch auf eine Mindestzahl von Ausgaben besteht nicht. Der Bezugspreis ist im Mitgliedsbeitrag enthalten.

Aufgrund der zentralen Lage, dem spannenden Gespräch, einer Vielzahl von Programmen und, je nach Teilnehmerzahl, verschiedenen Referaten zu Problemen durch Hard + Software, lohnt sich dieser Sonntag auf jeden Fall.

Bei Fragen, wendet Euch am besten an:
Jochen Haas
Roßstraße 27
5000 Köln 30
0221/51 98 70

Fortsetzung von Seite 43