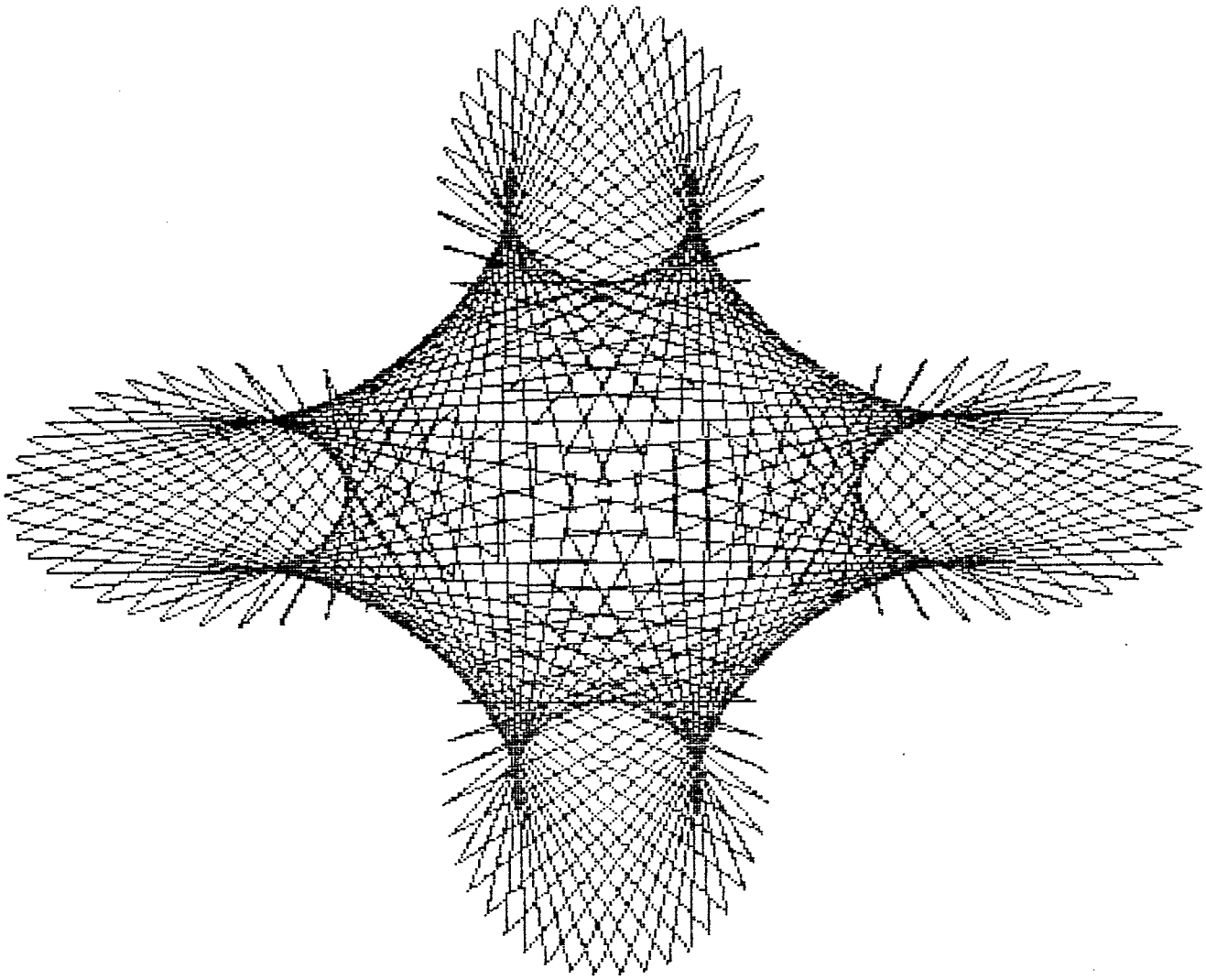
CLUBZEITUNG



16. AUSGABE

AUS DEM INHALT:

TERMINE FÜR CLUBTREFFEN

Mittwoch 27.10.82 19°°
Gaststätte Ludwigsvorstadt
Kazmairstr. 44 8000 München 2

Mittwoch 24.11.82 19°°
Gaststätte Ludwigsvorstadt
Kazmairstr.44 8000 München 2

Im Dezember findet kein Clubtreffen statt !!!

Zeitschriften-Umlauf:

Immer wieder wird geklagt, daß die Zeitschriften der Umläufe nicht mit der erforderlichen Sorgfalt weitergeschickt werden. Offensichtlich wird hier von jemanden der Sinn eines solchen Umlauf's gründlich mißverstanden.
Ziel ist es, daß die jeweilige Ausgabe des Magazins, bei mir beginnend jeweils zum nächsten Teilnehmer geschickt wird. Das Heft müßte also zum Schluß wieder bei mir landen.
Ich lege jeder Zeitschrift vorgedruckte Adressaufkleber bei, um es den Teilnehmern so einfach wie möglich zu machen.
Sollte einem die trotzdem noch damit verbundene Arbeit zuviel sein, so kann er sich jederzeit aus der Umlaufliste wieder austragen lassen.
Der Spaß mit den Zeitschriftenumläufen hat den Club alleine dieses Jahr schon mehr als DM 200.- gekostet. Ich bin nicht bereit nachzuforschen, bei wem die Hefte nun hängengeblieben sind. Wenn es in Zukunft nicht besser klappt, wird der Umlauf eingestellt.
Da sich die Versandform "Büchersendung" für das 80-Micro als unvorteilhaft erwiesen hat, werde ich künftig immer zwei Zeitschriften zusammenkommen lassen, und diese dann als Päckchen weiterschicken.
Das Päckchen hat gegenüber der Büchersendung den Vorteil, daß es bis 2 kg schwer und fest verschlossen sein darf. Außerdem geht der Inhalt den Postbeamten nichts an.

Liebe Clubfreunde,

nachdem ich in der letzten Ausgabe unserer Clubzeitung das ganze Material untergebracht hatte, ist nun schon seit Monaten eine Flaute eingetreten.
Ich brauche also dringend neue Beiträge für die Clubzeitung. Andererseits sollen nicht immer nur die selben Leute zu Wort kommen, sondern auch diejenigen, die sich bisher mehr im Hintergrund hielten.

Der Mangel an Beiträgen ist ein Problem mit dem ich mich seit der Gründung des Clubs herumschlagen muß. Ich glaube aber nicht, daß die vielzitierte Schreibfaulheit der Grund für diesen Engpaß ist. Vielmehr bin ich der Ansicht, daß die meisten nur aus Unsicherheit und Angst davor zurückschrecken selbst zur Feder zu greifen.
Ich möchte an dieser Stelle noch einmal betonen, daß redaktionelle Fähigkeiten absolut zweitrangig sind. Außerdem muß ein Thema nicht immer brandneu oder außergewöhnlich sein um bei den Mitgliedern anzukommen.
Wie immer im Leben sind auch in der Computerei oft Nebensächlichkeiten und Unscheinbares von größter Bedeutung. Also auch die Kleinigkeit oder den Programiertrick, den Sie erst letzte Woche entdeckt haben, und bei dem Sie davon ausgegangen sind, daß ihn die "Anderen" schon lange kennen. Aber - wir kennen Ihn noch nicht und würden gerne mehr darüber wissen.

Es gibt, und darüber bin ich sehr froh, einige Spezialisten im Club die Außerordentliches leisten. Die breite Masse aber sind Hobbyisten wie Du und ich. Leute, die in ihrer Freizeit ab und zu vor dem Computer hocken, und durch gezieles Betätigen der Tasten, versuchen ihm ihren Willen aufzuzwängen.
Gerade bei diesen Leuten findet man häufig selbstgestrickte Programme, die zwar nicht weltbewegend sind, aber mit Sicherheit als Beitrag für die Clubzeitung geeignet wären.
Also - Liebe Leute, was hindert euch noch daran euere eigenen oder abgetippte und verbesserte Programme zusammen mit ein wenig Dokumentation in der nächsten Clubzeitung zu veröffentlichen?

Bei dieser Gelegenheit noch ein Ansporn an die neuen Mitglieder. Es besteht die Möglichkeit, und - vereinzelt wurde auch schon Gebrauch davon gemacht, sich und sein Betätigungsfeld, in Form einer kurzen Beschreibung, den anderen Mitgliedern vorzustellen. Siehe auch die Vorstellung von Hrn. Wirtz auf Seite 3.

Euer *Gregor*

****
****** Wolfgang Wirtz *******
** Schanderlweg 7, 8000 München 82 **
****** Tel. 089/4304324 *******
****

Hallo Computerfreaks,

obwohl ich bereits seit September 1980 im Club bin, habe ich erst jetzt Zeit und Muße (oder war es Faulheit?) gefunden, mich vorzustellen.
Ich bin 1948 geboren und von Beruf kaufm. Angestellter in einem Industrieunternehmen. 1979 hatte ich für kurze Zeit ein Modell I besessen, welches ich aufgrund fehlender Anwendungsmöglichkeiten, prellender Tasten und den bekannten Kassettenladeproblemen wieder abgeschafft habe. Seit Dezember vergangenen Jahres bin ich nun stolzer Besitzer eines Modell III, 48K mit 2 x 40 Track Shugart - Laufwerken, Newdos und einem (übrigens sehr hervorragenden) Itoh 8510 A Drucker.

Der Computer ist hauptsächlich für die Lager-, Kunden-, Rechnungsverwaltung, Buchhaltung und Textverarbeitung eines Versandhandels im Einsatz. Natürlich läuft auch ab und zu das eine odere andere Spielchen darauf.
Da ich leider aus zeitlichen Gründen noch keine tiefschürfenden Basic - Kenntnisse sammeln konnte - geschweige denn Maschinensprache - erstreckt sich meine Arbeit am Computer mehr oder weniger nur auf Änderungen von bestehenden Programmen.

An Programmen bin ich recht gut bestückt, da eine Menge Modell I - Programme auch auf Modell III laufen.
Nach Durchsicht der bisher erschienenen Clubzeitungen scheine ich mit dem Modell III ja so ziemlich allein auf weiter Flur zu stehen, oder nicht?

Gerne bin ich mal zu einem Erfahrungsaustausch bereit.

Mit freundlichem Gruß

Wolfgang Wirtz

Kurt Trappschuh

8036 Herrsching, 03.06.82
Schloß Mühlfeld
(08152) 8127

Zusammenstellung der mir bekannten oder von mir entwickelten Zaps. Es kann keine Garantie auf Fehlerlosigkeit gegeben werden.

********** TRAPPER-ZAP 01 ********** 18.08.81 **********

Änderung der Bildschirmweite bei SCRIPSIT (auch USCRIPS), die bei Initialisierung auf 60 (3C Hex) steht. Die von mir hier vorgeschlagene Weite von 64 (40 Hex) kann natürlich individuel angepaßt werden.

SCRIPSIT/CMD,00,A5 change 0E 3C FD to 0E 40 FD

********** TRAPPER-ZAP 02 ********** 07.09.81 **********

Unterdrückung der Fage nach dem Druckercode bei USCRIPS2. Wenn dieser Zap installiert ist, fährt das Programm beim Laden gleich bis zum Anfang durch. Wenn dabei die Kleinschreibung nicht eingeschaltet ist, kann ein Festfahren des Programms erfolgen.

Code 0: USCRIPS2/CMD,43,81 change 30 28 2E to 30 18 2E
Code 1: USCRIPS2/CMD,43,85 change 31 CA 65 to 31 C3 65
Code 2: USCRIPS2/CMD,43,8A change 32 28 02 to 32 18 02

********** TRAPPER-ZAP 03 ********** 06.81 **********

PROFILE gibt die Druckdaten immer im Format 66 Zeilen/Seite aus. Mit diesem Zap kann dies an den jeweiligen Drucker angepaßt werden. In der Datei INIT steht im Sektor 0, Byte 4B (oder im Byte 3C bei älteren Versionen) die hexadezimale Anzahl der Zeilen pro Seite. Normalerweise steht dort 42 für 66 Zeilen, für 72 Zeilen muß dort 48 stehen.

INIT,0,4B change 42 to 48 oder
INIT,0,3C change 42 to 48

Kurt Trappschuh

8036 Herrsching, 03.06.82
Schloß Mühlfeld
(08152) 8127

********** TRAPPER-ZAP 04 ********** 27.02.82 **********

SUPERZAP und DEBUGGER wandeln für die ASCII-Darstellung am rechten Bildschirmrand alle Zeichen, die nicht zwischen 21 und 7F (hex) liegen in Punkte um. Die ersten beiden Zaps lassen eventuelle Leerzeichen (20 hex) wieder erscheinen, die zweiten beiden Zaps holen die Grafikzeichen (80 - 8F hex) zurück. Wenn keine Kleinschreibung installiert ist, werden die Kleinbuchstaben falsch (zum Teil als Ziffern) dargestellt.

```
SYS5/SYS,03,6D        change  D6 21 B9  to  D6 20 B9
SUPERZAP/CMD,12,D7    change  D6 21 B9  to  D6 20 B9

SYS5/SYS,03,69        change  C6 3F 4F  to  C6 80 4F
SUPERZAP/CMD,12,D4    change  0E 5F 1A  to  0E A0 4F
```

******* Newdos/80 Vers. 2.0 Zap ***** aus CompuTROning *******

Wenn dieser Zap eingebaut wird, gibt JKL Bildschirmgrafik auf dem EPSON MX-80 in Standard-Einstellung richtig aus. Es ist zusätzlich erforderlich SYSTEM,0,AX=127 einzustellen.

```
SYS3/SYS,4,C0     change   02 3E 2E CD   to   02 C6 20 CD
```

Kurt Trappschuh

8036 Herrsching, 03.06.82
Schloß Mühlfeld
(08152) 8127

********** TRAPPER-ZAP 005 ********** 03.06.82 **********

Diese Zaps ermöglichen einige Änderungen im Programm LPBUFFER/CMD Version 1.1 von mir. Die 3 Einzelzaps können beliebig miteinander kombiniert werden.

Zur Unterdrückung des Titelbildes mit Abfrage:

LPBUFFER/CMD,1,5E change 0D 21 00 3C 36 to 0D C3 FA FE 36

Zur Unterdrückung der Abfrage und Einstellung einer Fixgröße (XX ist der hexadezimale ASCII-Wert der gewünschten Eingabe, z.B. 3 = 33H):

LPBUFFER/CMD,1,7A change 31 38 F9 FE 3A 30 F5
                      to 31 00 00 FE 3A 3E XX

Zur Unterdrückung der Anzeige des Eingabewertes (oder der Fixgröße):

LPBUFFER/CMD,1,81 change 32 B6 3D 06 to 00 00 00 06

****
***** WOLFGANG WIRTZ *****
*** POSTFACH 1372, 8013 HAAR ***
***** TEL. 089/4304324 *****
****

Hallo Computerfreaks,

am Samstag war ich auf Einladung eines Bekannten als Gast beim Clubtreffen des AMMS e.V. - Arbeitsgemeinschaft Microprozessor & Minicomputer e.V. in Stuttgart.

Es handelt sich hier um einen Verein, der sich in 5 Untergruppen teilt, und zwar: 1.) PET, CBM, KIM, AIM, SYM etc., 2.) 8080, 8085, Z 80 Systeme, 3.) TRS-80, 4.) 6800 Systeme, 5.) APPLE. Jede der 5 Untergruppen hat ihren eigenen Obmann, der Ansprechpartner für sein System ist und alle ihn betreffenden Clubinfos bereithält.

Der Verein hat laut Mitgliederliste 133 Mitglieder. Davon waren ca. 50 anwesend zzgl. 12 Gäste.

Die Mitglieder waren bunt gemischt durch alle Computersysteme. Eine Clubzeitung wie die unsrige (wir können stolz darauf sein) gibt der AMMS nicht heraus, weil "die Laute zu faul zum Schreiben" sind.

Und nun zum Kern der Angelegenheit:

Es wäre doch sicher sehr interessant, einen Erfahrungsaustausch von Club zu Club (hier natürlich speziell die TRS-80 und verwandten Systeme) vorzunehmen.

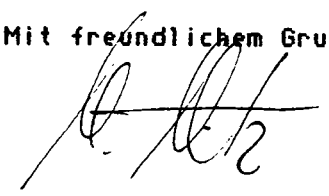
- Welche Fragen und Probleme tauchen immer wieder auf?
- Welche Hardware- bzw. Softwarelösungen sind auf Betreiben von Vorstand oder Mitgliedern realisiert worden? Ich denke da nur mal an den Umlautbausatz von H. Thalmeier.
- Ist eine Einkaufsgemeinschaft (bei Zusammenschluß mehrerer Clubs natürlich noch effizienter) für Hard- und Software denkbar?
- Softwaretausch von Club zu Club, um das Angebot zu erweitern?
- Lektüren- und Manualtausch?
- Zusammenarbeit bei größeren Projekten?

Die Liste ließe sich sicher noch weiterführen.

Im Laufe der Gespräche beim AMMS kam zur Sprache, daß sich etwa 30 Leute (incl. ihrer Frauen) zu einer gemeinsamen Fahrt zur Elektronika Anfang November nach München entschließen wollen. Hier wäre doch evtl. der erste Ansatzpunkt, um möglicherweise ein erstes Treffen an dem entsprechenden Tag zu vereinbaren.

Diese Ausführungen nur mal als Denkanstoß.
Gerne bin ich zu weiteren Hilfeleistungen in dieser Richtung bereit.

Mit freundlichem Gruß



Andreas Voss

Pickelstr. 19
8000 München 19

30.06.82

B e r i c h t i g u n g

In der Abschrift der DIN-Norm 66030 sind mir leider bei der Korrektur zwei Fehler entgangen.

1. Auf Seite 3 (Heftseite 46) fehlt bei der abgeleiteten SI-Einheit Ohm das Zeichen nach DIN 1301, ein großes Omega (Ω).

2. Auf Seite 6 (Heftseite 48) fehlt hinter den Vorsätzen folgender Satz:
"Exa und Peta wurden gegenüber der Vornorm DIN 66030 01/73 neu aufgenommen."

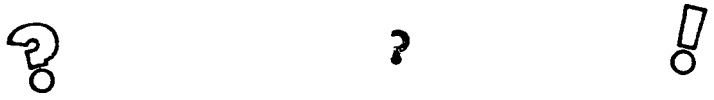
MYSTERY PROGRAM

By Charley Butler

(What does it do? One may never know, unless one enters it into their TRS-80! This program was written by Charley in one of his more sadistic moods. We hope you find it interesting!)

PROGRAM LISTING

```
0 'THE ALTERNATE SOURCE MYSTERY PROGRAM
5 'IT'S EASIER TO DEMONSTRATE THAN EXPLAIN!!
10 FOR X = 28707 TO 28814 : READ A : POKE X,A : NEXT
20 POKE 16633,143 : POKE 16635,143 : POKE 16637,143
30 POKE 16634,112 : POKE 16634,112 : POKE 16634,112
40 POKE 16548,36  : POKE 16549 ,112
50 CLS:LIST
60 DATA0,68,112,10,0,76,69,65,82,78,32,65,76,76,32
70 DATA65,66,160,32,83,69,76,70,206,77,79,68,143,89,73
75 DATA78,71,0,102,112,20,0,89,79,85
80 DATA82,32,79,87,78,32,80,82,79,71,82,65,77,83,206
90 DATA206,67,79,77,73,78,71,32,83,79,161,0,140,112,30
100 DATA0,73,78,32,58,147,251,84,72,69,32,65,76,84,69
110 DATA82,78,65,84,69,32,83,79,85,82,67,69,32,78,69
120 DATA87,83,39,33,0,0,0,28
130 'DOUBLE CHECK DATA BEFORE RUNNING!
140 'IF ALL ELSE FAILS, TRY LISTING THE PROGRAM AGAIN!
```



Enjoy...

UMLAUT-CHIP

Es gibt ihn noch - den Zeichengenerator für den deutschen Zeichensatz - zum vernünftigen Preis. Es handelt sich hierbei um eine Clubeigene Entwickluog und nicht um einen Nachbau. Hier das Wichtigste in Stichpunkten:

- Umlaute und echte Unterlängen auf dem Bildschirm.
- zwei Zeichensätze deutsch/international umschaltbar.
- betriebsfertig - komplett mit Schalter.
- ideal für Umlaut-SCRIPSIT Anwender.
- einfach einzubauen - alten Zeichengeneratorchip aus Steckfassung (Tastatur) entfernen, neuen Ummautchip hineinstecken. Nur bei sehr alten Geräten ist für den Character-Generator (Z29) keine Fassung vorhanden.
- in begrenzten Umfang kann ich auf Sonderwünsche bei der Zeichendarstellung eingehen.
- voll ausnutzbar nur mit Kleinschrift-Umbausatz.
- Preis DM 50.-

Club-Beiträge:

Da ich immer wieder danach gefragt werde, hier noch einmal das Clubkonto.

Postscheckamt München
BLZ: 700 100 80
Kontonr. 3452 35-800
G.Thalmeier

Achtung: *Alle Einzahlungen auf dieses Konto (auch zugesandte Schecks) müssen meinen Namen tragen. Einzahlungen, die nur an den Club adressiert sind, werden von der Post zurückgewiesen.*

Der Clubbeitrag beträgt DM 3.- pro Monat. Wie lange Ihr Beitragspolster noch ausreicht, können Sie auf dem Adressaufkleber ablesen, mit dem Sie diese Zeitung erhalten haben.

Die Entscheidung, wer noch Clubzeitungen erhällt, trifft ein Programm. Wer sein Beitragskonto um mehr als zwei Monate überzogen hat, bekommt keine Zeitung mehr. Neuen Mitgliedern, die noch nie Beitrag bezahlt haben, wird die zweimonatige Frist ebenfalls gewährt.

GRAFTRAX-80:

Vor kurzem habe ich in meinen EPSON MX-80 die EPROM-Erweiterung GRAFTRAX-80 eingebaut.

GRAFTRAX unterstützt folgende Schriftarten und Druckstärken:

B r e i t s c h r i f t
Kursivschrift
Normaldruck
hervorgehobene Schrift
Doppeldruck

Diese Schriftarten können b e l i e b i g *kombiniert werden.*

Durch Einzelpunktansteuerung ermöglicht GRAFTRAX hochauflösende Grafik und die Erstellung von speziellen Sonderzeichen.

Kunt ☎ √ π ¶ [ ] { }

Zusammen mit *Kunt* Trappschuh's TSCRIPS (stark erweitertes Umlaut-SCRIPSIT) wird GRAFTRAX-80 ein hervorragendes Werkzeug zur Textverarbeitung. Es ist sogar möglich die TRS-80 Block-grafik im Text einzusetzen.

Einen Nachteil hat GRAFTRAX allerdings. Während der MX-80 normalerweise ein Papierformat von 72 Zeilen/Seite annimmt, sind im GRAFTRAX nur 66 Zeilen eingestellt.
Bei einem Seitenvorschub (FF-Taste oder Lprintchr$(140)) verpositioniert sich der Drucker.
Dies kann umgangen werden, wenn man den Drucker zu Beginn mit

```
LPRINTCHR$(27);"C";CHR$(72)
```

auf das aktuelle Papierformat einstellt.

COMPARISONS AMONG ZBASIC, ACCEL2, AND BASCOM

by Bruce Douglass

There are now two compilers on the market for the TRS-80 besides the infamous BASCOM by Microsoft: they are the ZBASIC compiler from SIMUTEK and ACCEL2 by Gelder Software. As with all software, each of these has its strong and weak points. Depending on your application, one may well suit your needs better than another.

Let's review some of the basic (pun intended) aspects of the compilers separately.

The ZBASIC compiler has the following important aspects:

ZBASIC SPECIFICATIONS

(1) interactive between BASIC and compiler
(2) does not support arrays or floating point
    (integers and strings only!)
(3) fixed memory locations for object program and
    variables and fixed variable names
(4) AWESOME compilation speed! (1-5 seconds!)
(5) VERY fast run speed!
(6) does not support transcendental functions,
    although they may be simulated
(7) must use Level II USR function to interface
    with machine language programs
(8) run-time subroutine package of 1K
(9) few ROM calls
(10) if any part of the program will not compile,
    then none of it will compile
(11) documentation is fairly good and presents
    methods of simulation of many commands
    ZBASIC does not support
(12) compiles entire program into machine language
(13) does not support disk or tape I/O
(14) runs in 16K LEVEL II BASIC or DISK BASIC
    but requires you to buy the program to
    suit the size of your requirements
(15) object program loads as a system program
(16) you needn't pay them when you sell the
    compiled program!!
(17) compiles into self-contained system tape
    or disk CMD file
(18) compiled programs may be called via USR function
(19) compatible with TRSDOS or NEWDOS/80
(20) cost is $79.95 (tape version) and $89.95
    (disk version) or $99.95 (both versions) --
    manual available separately for $25

The ACCEL2 compiler has different attributes:

(1) PROGRAM FLAWED! INCORRECT READING OF DATA
    STATEMENTS WHEN COMPILED PROGRAM IS
    CALLED FROM DISK!
(2) not interactive to any great extent
    (limited to some error messages)
(3) compiles some floating point functions
    (+, -, *, /)
(4) supports arrays

(5) compile time is significantly longer than ZBASIC (10 seconds to over 10 minutes)
(6) compiles some string functions
(7) documentation is poor to mediocre
(8) run time is significantly longer than ZBASIC, depending on program
(9) will compile even if the program contains some portions that will not compile since non-compiled statements remain in BASIC
(10) essentially all ROM calls
(11) compiles into REM statements. Any part that cannot be compiled remains in BASIC format, but no part of the compiled program may be edited
(12) supports disk or tape I/O by keeping the pertinent statements in BASIC
(13) runs in 16K LEVEL II or DISK BASIC; one size fits all
(14) you needn't pay when you sell the compiled program!!
(15) compiler is relocatable
(16) to save SYSTEM tapes, you must purchase TSAVE for $10, but you may save on disk directly using the compiler
(17) requires ACCEL2 run time system to load compiled program
(18) compatible with TRSDOS or NEWDOS/80
(19) cost is $89

As for BASCOM:

(1) Minimum 32K to run and 48K to compile
(2) more extensive error messages
(3) MANY more options available for compilation
(4) user must play "musical diskettes" unless he has 3 or more disk drives
(5) ROYALTIES of 9% per sale or $195 per year REQUIRED on sold programs
(6) provides REL (relocatable module) and LST (Z80 listing of object program) if desired
(7) BRUN, the run time system, occupies 5200H to 8A00H (that's 14336 bytes!)
(8) compilation is SLOW and TEDIOUS in comparison with either ZBASIC or ACCEL2
(9) speed is usually intermediate between ZBASIC and ACCEL2
(10) cannot use variables to DIM arrays
(11) compiles superset containing most of DISK BASIC commands, and several that are not supported by DISK BASIC
(12) gives double precision transcendental functions
(13) easy to interface with machine language or Fortran-80 or Macro-80
(14) supports disk I/O, but not tape I/O
(15) manual is extensive and comes in two parts; compiler manual and BASIC 80 manual
(16) comes with two SYSTEM diskettes containing the programs necessary to compile (BASCOM, L80, BASLIB, and BRUN)

(17) compatible with TRSDOS or NEWDOS/80, although it comes with TRSDOS 2.3 on both diskettes
(18) optional use of long variable names with forty significant characters
(19) cost is $195

As you can imagine, with these different attributes, the programs really are for different purposes. ZBASIC handles only integers and strings, while ACCEL2 handles all three numerical types. Those functions that ACCEL2 cannot compile will be left in BASIC form, and executed by the BASIC interpreter. BASCOM executes a superset of DISK BASIC, containing most of DISK BASIC commands and some commands not supported by DISK BASIC.

Some of these functions may be simulated in ZBASIC, for instance, the manual presents routines that will return 1000 times the value for COS, SIN, and ARCTAN. BASCOM will return double precision transcendental functions, while ACCEL2 supports the transcendental functions to the extent that it leaves them in the BASIC program, but does not compile them. They are not supported in ZBASIC, although they may be simulated, after a fashion.

This brings up an interesting point. Since ACCEL2 compiles full floating point for addition, subtraction, multiplication, and division, double precision values for the transcendental functions of LOG, SIN, etc. can be obtained using ACCEL2 by compiling a finite power series. An example of this was used as a bench marker for BASCOM vs ACCEL2 vs BASIC (Program 7 below). This is a capability, while possible in ZBASIC, that would be MUCH more trouble than it is worth. ZBASIC supports only 16 bit arithmetic. If you need multiple precision for your application, DON'T use ZBASIC.

I stated that arrays are not supported in ZBASIC. This is strictly true, but they may be simulated reasonably easily, and with a savings of memory space over the BASIC format. What is done is to dedicate a block of memory in RAM for the array, and access the data points using PEEK and POKE. This is not too difficult, but it is a bit of a pain to rewrite all the programs you own in order to compile them (I checked; ALL my programs used arrays!). One game I wrote required a 4000 element array of 8-bit integers. It was fairly straightforward to rewrite it using a 4000 byte block of memory below the location of the object program. Pointers may be kept going in a similar manner as in BASIC. You must be cautious with regards to the size of your numbers. If the number exceeds 256, only the LSB (least significant byte) will be POKEd into the memory location. If the numbers you will be using are larger than 256, or are POTENTIALLY larger, then you must use two-byte locations for the array elements, and be sure your pointers are updated by two bytes at a time -- the POKE (or PEEK) the MSB in the first byte, and the LSB in the second byte. Software conquers all!

A short routine to do this follows:

```
10 L=25000:FORI=1TO10 'L=STARTING ARRAY ADDRESS
20 READ A 'GET THE NUMBER TO POKE
30 A1=A/256 'MSB; NOTE IT IS AN INTEGER ALREADY DUE
           TO INTEGER ARITHMETIC
40 POKEL,A1:L=L+1 'POKE IT IN AND INCREMENT L
50 POKEL,A-A1*256 'NOW THE LSB INTO PLACE
60 L=L+1 'NOW L IS INCREMENTED THE RIGHT AMOUNT
70 NEXTI
```

Reading the data out of the array is straightforward as well, consisting essentially of replacing the POKE with PEEK statements, and reconstructing the integer rather than dissect it.

ACCEL2 supports arrays, and you may choose to compile, or not to compile, them. I used ACCEL2 to compile an OTHELLO game I have that uses several arrays. I have not used

ZBASIC due to the amount of rewriting that would be necessary. One of these days, when I have several hours with nothing to do, I will use it to see how much faster it will run. The time for the computer to decide on its move went from 45 to 60 seconds (BASIC) to 6-10 seconds (ACCEL2). When the same program is compiled with BASCOM, the response time for the computer to move is 3-5 seconds. Most of the savings in time between BASCOM and ACCEL2 is due to increased speed of redrawing the screen (remember, ACCEL2 does not compile PRINT statements).

A note here should be made. The Othello game is the only program I have that uses DATA statements. When ACCEL2 compiles and runs it, it works beautifully. You can then save it on disk. But when you call the compiled program off disk and try to run it, it gives you an OUT OF DATA error! I tried several fixes, but none seemed to work. Thus, my conclusion is that if you use DATA statements in your programs, you must compile the program each time you run it. This is based on one program, however, and I am open to comments from other readers.

This compilation takes from 10 seconds (very short program) to five minutes (6K Othello program) to 30 minutes (long program). Compared with ZBASIC, these compile times are like eternity. The longest compile time I've had with ZBASIC was about 3 seconds. BASCOM, due to the necessity of switching disks, and the long link-loading process, takes much longer -- 5 minutes for short programs, 10 minutes for 6K Othello, and much longer for long programs.

A major difference among the programs is how the programs are stored. ZBASIC takes the entire program and compiles it into a set memory location. You are limited to use only certain variable names (A-Z, A1-Z1, A2-Z2, A$-Z$), which fit into pre-specified memory addresses (in 16K, the fixed variable locations begin at 7C00H). With ACCEL2, you may use any normal variable names except QX, which is reserved. With BASCOM, you may use any ordinary name, or if you wish, you may set a switch and use names up to 40 characters in length, with each character significant.

A nice capability found in BASCOM that ZBASIC lacks is to compile into variable memory locations. If the program isn't too large, it may be possible to disassemble it using TLDIS or DLDIS into an EDTASM source file, either for DISK*MOD or Apparat EDTASM. Then it may be reassembled into a new location. You will need a lot of memory to do this. Of course, with a run time system of over 14000 bytes, there is much more of a need to be able to relocate the object program than with a run time system of 1K, such as for ZBASIC or ACCEL2.

ACCEL2 puts in REM statements into the BASIC program, which with the help of the run time package, call ROM routines to execute the compiled program. Non-compiled sections of the program remain in BASIC format. The BASIC program, once compiled, may not be edited. If your program has variable dimensioned arrays (your program contains something of the form 10 INPUTN:DIMA(N)), then you must precede the statement with REM NOARRAY to tell the program not to compile the arrays. With BASCOM, you may not use variables to DIM arrays. If you have an interactive program, as in many scientific applications, that require an equation or statement to be placed into the program, you have a problem, since once compiled, you may not change the program using normal BASIC modes of operation. You MAY be able to read in your equations as strings and POKE them in a line somewhere. This requires two lookup tables for defined BASIC words and operations, one for the tokens that stand for one ASCII character (e.g., +,/,*,=) and one that looks up those that stand for three ASCII characters (e.g., SIN, LOG, COS, ABS, SQR). The pointer for the parser will have to be incremented by one if it finds a token in the first category, and by three if it finds one in the latter category. If the character is not in either list, then POKE the ASC of that character into the line, and increment the parser by one. If the reserved operations are not entered as tokens, you will get a SYNTAX ERROR when the program attempts to execute them. Remember, you cannot edit that program once compiled!

I have played around with this idea with ACCEL2, but have not yet made it workable.

I think it will work, however. Just remember to precede the line with REM NOEXPN so that the line won't be compiled, and make it something like 300 XXXXXXXXXXXXXXXXXX-1. Then you can look for the line with the X's by using a FOR-NEXT loop with PEEKs (for example, PEEK(I)=88 AND PEEK(I+1)=88 THEN GOTO ....). Follow the POKEd in string by POKEing in a 58 (":") and and 147 ("REM"). That way you can forget about the extra X's in the line. Well, that about exhausts my ideas on the subject.

BASCOM takes a different approach to compiling. The source program must be stored on disk as an ASCII file (using the SAVE "filename/ext",A format). The program BASCOM itself calls the source program off disk and writes LST (optional) and REL files. The LST is an ASCII file of the object program in Z-80 opcodes. The LST file may optionally be sent to a printer. The actual BASIC lines are listed in the LST file as comments.

```
BASCOM 5.23 - COPYRIGHT 1979, 80 (C) BY MICROSOFT - 13512 BYTES FREE

0014 0007       10 CLS
        ** 0014'            CALL        $4.0
        ** 0017'L00010:     CALL        $CLS
001A 0007       20 PRINT "OK, BOSS"
        ** 001A'L00020:     CALL        $PROA
        ** 001D'            LD          HL,<CONST>
        ** 0020'            CALL        $PV2D
0023 0007       30 FOR I=1 TO 1000
        ** 0023'L00030:     CALL        $FASA
        ** 0026'            DW          I!
        ** 0028'            DW          <CONST>
        ** 002A'I00000:
002A 000B       40 NEXT I
        ** 002A'L00040:     CALL        $FADA
        ** 002D'            DW          I!
        ** 002F'            DW          <CONST>
        ** 0031'            CALL        $FASO
        ** 0034'            DW          I!
        ** 0036'            CALL        $LEJA
        ** 0039'            DW          I!
        ** 003B'            DW          <CONST>
        ** 003D'            DW          I00000
003F 000B       50 $INCLUDE SET/BAS
        ** 003F'L00050:
003F 000B       51 CLS
        ** 003F'L00051:     CALL        $CLS
0042 000B       52 FOR I=1 TO 127:FOR J=1 TO 47
        ** 0042'L00052:     CALL        $FASA
        ** 0045'            DW          I!
        ** 0047'            DW          <CONST>
        ** 0049'I00001:
        ** 004C'            DW          J!
        ** 004E'            DW          <CONST>
        ** 0050'I00002:
0050 000F       53 SET(I,J)
        ** 0050'L00053:     LD          HL,I!
        ** 0053'            CALL        $CINA
        ** 0056'            PUSH        HL
        ** 0057'            LD          HL,J!
        ** 005A'            CALL        $CINA
        ** 005D'            POP         DE
        ** 005E'            CALL        $GST
0061 000F       54 NEXT J,I
        ** 0061'L00054:     CALL        $FADA
        ** 0064'            DW          J!
```

```
      ** 0066'          DW        <CONST>
      ** 0068'          CALL      $FAS0
      ** 006B'          DW        J!
      ** 006D'          CALL      $LEJA
      ** 0070'          DW        J!
      ** 0072'          DW        <CONST>
      ** 0074'          DW        I00002
      ** 0076'          CALL      $FADA
      ** 0079'          DW        I!
      ** 007B'          DW        <CONST>
      ** 007D'          CALL      $FAS0
      ** 0080'          DW        I!
      ** 0082'          CALL      $LEJA
      ** 0085'          DW        I!
      ** 0087'          DW        <CONST>
      ** 0089'          DW        I00001
008B 000F      60 END
      ** 008B'L00060:   CALL      $END
008E 000F
      ** 008E'          CALL      $END
00AF 0019

00000 FATAL ERROR(S)
13138 BYTES FREE
```

The production of the LST file is optional. The REL file is a relocatable object file that must now be fed to the L80, or linking loader.

The L80 reads in the REL file, and will place the program directly above the BRUN, or run time module, unless another location is specified. The data area location may also be specified by the user.

Now you must link the L80-REL with BASLIB, which calls routines in this library off disk to link with the object program now in memory. This is the most time consuming step. Then you call the REL program off disk again (be sure to use the "switches" when loading this time so that L80 will save the object program and exit to DOS). It performs the final linking operations and writes the object program to disk.

You may link Microsoft Fortran-80 or Microsoft Macro-80 files with your program as well. This is extremely handy for those people having those programs -- you don't have to recode your favorite Fortran routines!

To run your program, load BRUN off disk. It occupies locations 5200H to 8A00H (as you can see, even the run time system won't fit into 16K!). When prompted, tell it which program to load, and BRUN will load and execute the program. When it executes a STOP or END, control is transferred to DOS operating system (in ZBASIC, STOP will exit to BASIC and END will exit to DOS).

A handy feature of BASCOM is the error messages. It not only gives you the line number, but also the type of error and an arrow pointing to the exact location of the error in the line. A feature that some (such as myself) will not care for is the necessity of playing musical disks. The compilation process requires switching disks around unless you have three or more drives! This is due to the size and complexity of the system itself, and the large number of compilation options at your disposal. It is still a pain, especially if you only have one disk drive. The BASCOM system comes with four large programs that won't fit on one disk (BASCOM, L80, BASLIB and BRUN). And then you need a disk for the object and source programs. For small programs, you can kill FORMAT, BACKUP, and BASIC (rename BASICR to BASIC) on one of the BASCOM disks that you have backed up. That should give you enough room for a SMALL program. You'll need to remember the TRSDOS master password, F3GUM, to make those changes.

BASCOM will give you useful information during the compilation process, such as errors in lines, the type of error, memory left, and starting and ending address for the object program. And it will compile certain errors without blowing up, and give you warning messages. If it encounters an array that has not been dimensioned, it will give it a default dimension of 10, and advise you of the fact.

For just a moment, let's mention some other aspects that are peculiar to BASCOM. REM statements are supported, but (even though the manual doesn't give you this information) the REM statements must start the line. If they occur at the end of a line containing a compiled BASIC statement, BASCOM will give you a fatal error message.

BASCOM gives you an additional conditional construct in the form of WHILE - WEND. Often you must perform loops until a statement is true or false, which requires a comparison. This does it for you. WHILE "expn" is true, when the execution gets to the accompanying WEND, control will loop back to the first statement following "WHILE expn". When it is no longer true, control will fall through the loop. This is a nice feature of the compiler, although it can easily be simulated in either FOR - NEXT or GOTO style loops.

One of the most useful compile time features is the ability to call other BASIC ASCII format files with the %INCLUDE command. This command effectively allows you to merge BASIC ASCII files, and so allows for easy access to BASIC subroutines. Although you may have some trouble with SQ errors (line number sequence errors), this can be remedied by setting the compile time switches -5-C, which tell the compiler to use the BASIC80 5.0 language set, and to relax line number constraints.

If you use the BASIC80 5.0, which allows for variable names of up to 40 characters, you must be careful not to use such statements as GOSUB1000, since the compiler will think it is a variable, and not a command. With such long variable names, they had to allow the appearance of BASIC reserved words in the variable names, so GOSUB1000 looks like a variable. GOSUB 1000 is okay, though. If you use the default lexical conventions (4.51, or normal DISK BASIC), you can use GOSUB1000 and it will compile and execute correctly.

During compile time, you can set your input and output sinks easily. You can input a command from the keyboard and have it compile to disk! Or, you can output to the printer. The options are many.

One of ZBASIC's extremely useful features is that it is interactive. Since your BASIC program and the compiled program are in different locations, you can jump back and forth and easily modify your BASIC program until you get it just right. You can compile, run, and save your program without disturbing the resident BASIC program. Just remember to put in a STOP command to return to BASIC, or put in a loop that PEEKs at location 14400 and executes a STOP if it equals four (e.g., if the <BREAK> key is pressed). Unless you POKE into the BASIC program space with your compiled program, your BASIC program will stay there, safe and sound.

Not so with either BASCOM or ACCEL2. As previously stated, ACCEL2 eats the source program resident in memory. So make sure you save the BASIC program on disk or tape BEFORE you compile. Further, if it starts to compile, and finds something that it doesn't like, it will give you an error message, but your program has hitten the big one! You must type NEW, and re-enter or reload the program. This is annoying, but once you get used to it, it is tolerable.

With BASCOM, the program MUST be on disk, so that you don't have to remember to save it. Interactive compilation is a really major point in ZBASIC's favor.

Another important difference is the run speed of the programs. After all, that's why one spends a small fortune on a compiler in the first place, eh? I bench marked several small programs to indicate loop speed, arithmetic speed, and jumping (GOTO)

speed. The programs and results are given below. Note that this is only for integer values, except for Program 8, which has some double precision variables.

PROGRAM NUMBER		BASIC TIME	ACCEL2 TIME	ZBASIC TIME	BASCOM TIME
1		3:40	3:40	2:15	2:49
2		2:50	2:35	1:02	1:21
3		0:25	0:20	0:07	0:10
4	(SET)	0:40	0:05	0:03	0:03
5		0:57	0:46	0:06	0:45
6		1:15	0:38	0:05	0:26
7	(primes)	15:26	5:44	2:13	6:46
8	(dblcos)	10:07	4:39	----	2:34
9	(psuedo)	3:53	2:07	0:53	1:26

---- indicates the program used greater than integer arithmetic and so could not be compiled by ZBASIC.

Notice the differences in some of the speeds. The programs are listed at the end of the article. I suggest you look at them, and note that for some applications BASIC and ACCEL2 are the same (or close), particularly when there are many PRINT statements, which ACCEL2 doesn't compile. But for the programs that tests SET, the run times are quite fast, comparable to ZBASIC. As you can see, BASCOM mostly falls in between ZBASIC and ACCEL2 in terms of speed.

The jumps within the compilers are extremely fast when compared to BASIC, since BASIC uses a lookup table for GOTOs and GOSUBs, whereas the compiled programs can jump directly to the address.

I would like to mention something about one of the programs now. Program 7 is a simple one to calculate the prime numbers from 3 to 1000. Originally, the program went like this:

```
10 CLS
20 DEFINT A-Z
30 FOR I=3 TO 1000 STEP 2
40   FOR J=2 TO I/2
50     A=I/J REM A IS AN INTEGER
60     B=A*J REM SO IS B
70     IF B=I THEN 100 REM NOT PRIME
80   NEXT J
90   PRINT I; REM PRIME
100 NEXT I
110 STOP
```

But, much to my amazement, ACCEL2 compiled the program and the indexing variable "I" began incrementing by hundreds and then thousands! Where is the problem?? Well, after a couple of hours of threatening to play frisbee with my box of diskettes, I finally discovered that you can't jump out of a current FOR - NEXT loop. Things go to Never - Never Land right away if you do. So, once realizing that was the problem, I changed line 70 to read 70 IF B=1 THEN J=I/2 and line 90 to read 90 IF B=1 THEN 100 ELSE PRINT I; which gets the same effect. But the ACCEL2 manual advised that you could jump out on ongoing FOR - NEXT loops. It lies. Generally, that's not something you want a manual to do.

A capability that would greatly enhance the usability of ZBASIC is a straightforward method of interfacing with other machine language programs, either by compiling &FUSR statements or creating two new commands JUMP and CALL, which would be simple to

implement in the program. It does compile the Level II format of USR calls, but not the disk BASIC version. ACCEL2 has this capability since what it doesn't like, it won't compile, but leaves in BASIC thus allowing use of USR calls. BASCOM compiles DEFUSR and USRn commands, as well as CALL. With the former language commands, the only way to pass arguments to your machine language routines is to dedicate a block of memory for this purpose and POKE and PEEK arguments back and forth. The CALL command only works with Fortran-80 or Macro-80 programs, according to the manual. Alternatively, you can define machine language routines in strings (using CHR$) and find where they are using VARPTR, and then jump to the address with DEFUSR and A=USR0(0). You may, during the compilation process, dedicate memory for the purpose of placing machine language routines there.

In conclusion, I draw the following facts regarding these three compilers:

RESULTS OF COMPARISON

(1) ZBASIC is optimal in speed and integer arithmetic: best for fast action game or real-time controlling applications. BASCOM is often as fast, but more often is slower. ACCEL2 is the slowest of the three.

(2) ACCEL2 is optimal in memory space utilization, but runs more slowly than ZBASIC or BASCOM.

(3) ACCEL2 and BASCOM have more uses in scientific or technical applications which require greater than integer precision; BASCOM supports only disk I/O, whereas ACCEL2 supports both disk and tape I/O.

(4) ACCEL2 is the most generally applicable compiler since: (1) it will leave the uncompiled sections of the program intact, and speed up the other portions, although the increase in speed is not as great, and (2) it will compile and run in 16K.

(5) ZBASIC can be tedious to use, because often many statements (such as arrays) must be re-written, but since the compiler itself is much more interactive this problem is greatly mitigated. BASCOM is most tedious during compile time, but less so during the writing of the program itself.

(6) The ZBASIC manual is far superior to the ACCEL2 manual irrespective of the capability of the compilers themselves. Neither is NEARLY as extensive as the BASCOM manual.

(7) BASCOM is by FAR the most complete and thorough implementation, but for many applications, it may not be cost-efficient, since (1) not even the run time system will fit into 16K and (2) it costs over twice as much as ACCEL2 or ZBASIC. Microsoft requires royalty payments on sold object programs; neither ACCEL2 or ZBASIC do.

(8) ACCEL2 does contain an error in handling of READ statements when compiled programs are called off disk. This greatly mitigates my recommendations of it.

About this stage of the game, I'm supposed to tell you which compiler is best. Well, that's up to you, friend. None of them is the ideal compiler, though all have good points. And their good points are often different good points. ACCEL2 isn't as fast as or as interactive as ZBASIC. But one nicety about ACCEL2 is that generally, the amount of rewriting to get compiled programs to run is relatively minimal, once you understand some of the things the manual doesn't tell you (see above). BASCOM compiles a superset of the DISK BASIC, but requires more memory, and at least one disk. You cannot sell programs compiled with BASCOM to LEVEL II owners, and when you do sell programs, you must pay royalties to Microsoft. Compilation can be tedious, and it takes orders of magnitude longer than compilation with ZBASIC. For these reasons, I do not recommend BASCOM for the average user. For a professional writing professional programs for disk based systems, it is an excellent program. My opinion is that it is not cost-efficient for general use. ACCEL2 is the best alternative in that it fits into 16K and will compile most BASIC programs. The speed increase is not as great as with ZBASIC or BASCOM,

out what it doesn't like, it will leave for the BASIC interpreter. ZBASIC is the fastest, but often it takes some creative programming to rewrite programs so that they will work in the ZBASIC syntax. If all you want is to compile games and programs not requiring better than integer precision, ZBASIC is the one you want. The recommendation for ACCEL2 is mitigated greatly by the flaw in reading data statements.

(Editor's Comment: On the day we were printing the final draft of this article, we received in the mail a revised edition of ACCEL2, with a note stating that the enclosed version was easier to use, faster at compiling, and that it had incorporated some improvements suggested by users, thus making it an "interactive" compiler. We were also informed by the creators of ZBASIC that they were coming out with a new version, one which will be relocatable, normal variables, sequential tape and disk I/O, etc. We mention this in fairness, and regret that we hadn't the time to explore the improvements for inclusion herein. If enough interest is generated, we'll consider a follow up report on these new versions. jmk)

PROGRAM ONE

```
10 CLS
20 FOR I=1 TO 2000
30   PRINT I
40   PRINT I
50   PRINT I
60   PRINT I
70 NEXT I
80 STOP
```

PROGRAM TWO

```
10 CLS
20 DEFINT I
30 FOR I=1 TO 10000
40   PRINT @ 500,"*"
50   PRINT @ 500," "
60 NEXT I
70 STOP
```

PROGRAM THREE

```
10 CLS:DEFINT I
20 FOR I=1 TO 1000
30   PRINT @ 450,"*"
40   J=I*I
50   PRINT @ 450," "
60 NEXT I
70 STOP
```

PROGRAM FOUR

```
10 CLS:DEFINT I,J
20 FOR I=0 TO 127
30   FOR J=0 TO 47
40     SET(I,J)
50   NEXT J,I
60 STOP
```

PROGRAM FIVE

```
10 CLS:DEFINT H-J
20 FOR H=1 TO 2000
30   I=RND(127)
40   J=RND(47)
50   SET(I,J)
60 NEXT H
70 CLS
80 STOP
```

PROGRAM SIX

```
10 CLS:DEFINT I
20 FOR I=1 TO 2000
30   J=I+I+I
40   J=J*J
50   J=J/I
60   J=J+I+I-I-I
70 NEXT I
80 STOP
```

PROGRAM SEVEN

```
10 CLS REM PRIMES TO 1000
20 DEFINT A-Z
30 FOR I=3 TO 1000 STEP 2
40   C=I/2
50   FOR J=2 TO C
60     A=I/J
70     B=A*J
80     IF B=I THEN J=C
90   NEXT J
100  IF B=I THEN 110 ELSE PRINT I;
110 NEXT I
120 END
```

PROGRAM EIGHT

```
10 CLS REM DOUBLE PRECISION COSINE
          ACCURATE TO < 2*10[(-9)
20 DEFDBL A-H,X:DEFINT I,S:S=1
30 A=-.49999 99963:B=.04166 66418:
   C=-.00138 88397:D=.00002 47609:
   E=-.00000 02605:G=1.5707 6327:
   G1=6.2831 85307:G2=3.1415 92564
40 FOR I=1 TO 300:X=I
50   IF X>G1 THEN X=X-G1:GOTO 50 'ANGLE REDUCTION
55   IF X>G THEN X=G2-X:S=-S:GOTO 55 'DITTO
60   F=1+X*X*(A+X*X*(B+X*X*(C+X*X*(D+X*X*E))))
70   PRINT "I=";I,"COS(I)=";F*S
80 NEXT I
90 END
```

PROGRAM NINE

```
5 DEFINT A-Z REM PROGRAM FROM ZBASIC
                MANUAL, ADAPTED SLIGHTLY: CALCULATES
                1000*COS(X) WHERE X IS IN DEGREES
10 FOR I=1 TO 1000:X=I
20   IF PEEK(14400)=4 THEN STOP REM BREAK KEY?
30   GOSUB 9000
40   PRINT "I=";I,"1000*COS(I)=";X
50 NEXT I
60 STOP
9000 REM THIS IS IT!
9010 X=X+90
9020 IF X>359 THEN X=X-360:GOTO9020
9030 S=1:IFX>170 THEN X=X-180:S=-1
9040 IF X>89 THEN X=180-X
9050 IF X>45 THEN 9090
9060 X=X*174/10:R=X/10
9070 X=X-R*R/200*R/30+R*R/200*R/100*R/250*R/240
9080 X=X*S:RETURN
9090 X=90-X
9100 X=174*X/10:R=X/10
9110 X=1000-R*R/20+R*R/200*R/100*R/120
9120 X=X-R*R/200*R/100*R/250*R/200*R/720
9130 X=X*S:RETURN
```

★★★★

BIT KICKIN' WITH JESSE BOB
(Copyright (c) 1981 by The Circle J Software Ranch)

The visage at left belongs to the legendary Jesse Bob Overholt, proprietor of the famed Circle J Software Ranch. Nestled in the fertile plains of North Texas, the Circle J occupies 180,000 microacres of prime bit grazing land. From his herd of over 90,000 bits Jesse Bob produces fine software for computers all over the world.

Dear Jesse Bob,

Does a write-protect tab on a diskette really and truly mean that it can't be written on? I have heard tales of diskettes being clobbered, even though they were supposedly write-protected. So what's the deal, are these diskettes being written on, or are the stories I've heard giving me nightmares for no reason?

Overwrought in Oregon

Dear Overwrought,

Well, this isn't the kind of problem you should lose sleep over, but yes, it is possible to lose data on a write-protected disk. How is this possible? To answer that we need to study how the disk write-protect actually works.

The write-protect sensor (or sensors if you have a "flippy") is simply a switch. If a drive contains a write-protected diskette, a signal is sent back to the disk controller, when the drive is selected, indicating this fact. One very important fact should be mentioned here. Just because the write-protect sensor is a switch, there is no guarantee that it is mechanical. Many of the newer disk drives now on the market use an optical sensor instead. This is more reliable, but does pose a new risk. What happens if a disk is protected by transparent tape instead of a write-protect tab? If you said it gets clobbered, you are absolutely right -- go to the head of the class! Light can shine right through transparent tape, so the drive considers itself to be unprotected! The moral here is don't use clear tape to write-protect diskettes that you want to keep.

Now back to our story of how a protected disk can be written on in spite of write-protect tabs. In addition to providing a line to tell the controller that the diskette is protected, the write-protect sensor being "on" also "locks out" the write circuitry of the disk drive, preventing it from writing by accident. This would seem to be foolproof, but being computer owners we know better, right?

One of the fundamental laws of the universe is that an electrical current through a wire will generate a magnetic field around that wire. This handy little principle is what makes a disk work in the first place. So what will happen if electrical current should flow through the write head? If the current is sufficiently large, it will generate a magnetic field that will alter the information on the disk directly below the head, EVEN IF THE DISK IS WRITE-PROTECTED! Currents of this type can be generated by "spikes" in the line, power failures, or general ineptitude on the part of the operator.

==> PROGRAMMBIBLIOTHEK TRS-80 USER CLUB <==

PROGRAMM	EINSENDER	PREIS	LAENGE	BESCHREIBUNG
AUTOCOPY	THALMEIER	5.00	1259	UTILITY - PROGRAMM ZUM AUTOMATISCHEN KOPIEREN VON DISK-FILES
AUTOKILL	THALMEIER	5.00	722	DISK-FILES MIT DEM ZUSATZ /DEL WERDEN AUF DER ANGEGEBENEN DISKETTE GELOESCHT
BABEL	GOEBL	5.00	2646	TURMBAU-SPIEL
BALL	SCHUELER	5.00	1501	SPIEL AUS LEVEL II - HANDBUCH
BANDWURM	TRAPPSCHUH	5.00	3393	GESCHICKLICHKEITS-SPIEL
BIO	GOEBL	5.00	4218	BIORYTHMUS
BIORYTH 4	SCHLADEBACH	5.00	3345	EINFACHER GRAFIC-BIORYTHMUS
BIORYTHM 2	MADER	5.00	4896	BIORYTHMUS-PROGRAMM MIT KURVENGRAFIC
BIORYTMUS 3	TRAPPSCHUH	5.00	4541	BIORYTHMUS-PROGRAMM MIT GUTER DARSTELLUNG DER KURVEN UND STEUERUNG DES ABLAUFES
BIRTHDAY	MADER	5.00	4168	GEBURTSTAGS-GRUSS MIT GRAFIC+TON
CALEND	BRUEHBACH	5.00	1770	EWIGER KALENDER
CARD 51	TRAPPSCHUH	7.50	6398	BASIC - PROGRAMM DES BEKANNTEN KARTENSPIELS " 51 " MIT GUTER GRAFIK
CINIT	BRUEHBACH	5.00	1481	UTILITY ZUM GENERIEREN VON CHAIN-FILES
COMPRESS	BRUEHBACH	10.00	14353	UTILITY ZUM VERKUERZEN VON BASIC-PROGRAMMEN (AEHNLICH PACKER)
CRUPS	SCHLADEBACH	5.00	2513	WUERFEL-SPIEL
DEMOGRAF	MADER	10.00	11353	VERSCHIEDENE GRAFIC-PROGRAMME (NUR MIT BASIC+)
DOGRACE	MADER	5.00	1564	HUNDERENNEN
ELIZA	GOEBL	7.50	6551	TRS-80 ALS PSYCHOTHERAPEUT
FORMELN	MADER	10.00	10931	VERSCHIEDENE MATHE.PROGRAMME
GROSS-SCHRIFT	SCHLADEBACH	5.00	3345	STELLT TASTATUR-EINGABEN IN GRAFIC-LETTERN DAR. FORMAT: 6 ZEILEN MIT JE 16 ZEICHEN
HANGMAN	SCHLADEBACH	5.00	2664	WORTE RATEN MIT GRAPHIC-AUSWERTUNG (NEUE VERSION)
HAUSKAUF	THALMEIER	5.00	769	MONATLICHE BELASTUNG
HEXUM	THALMEIER	5.00	784	UMRECHNUNGEN DEZ/HEZ UND HEX/DEZ
HOELZER	MADER	7.50	5500	NIM-SPIEL
INHALT	THALMEIER	5.00	2050	RESIDENTES KASSETTEN-INHALTSVERZEICHNIS
JAEGER	MADER	5.00	2085	RAUMJAEGER
JKLMID	TRAPPSCHUH	5.00	578	GIBT BEI "JKL"-BEFEHL DEN ILDSCHIRM-INHALT AUSGEMITTELT AUS. SIEHE CZ NR.12
KAFO	MADER	7.50	9420	KANIBALEN-SPIEL
KALAH	MADER	5.00	4061	ORIENTALISCHES BRETTSPIEL
KALORIE	GOEBL	5.00	4774	KALORIEN-TABELLE
KREDIT	THALMEIER	7.50	5477	ZINSBERECHNUNG
KURVENANPASSUNG	DUMKE	7.50	8737	BERECHNEN UND ZEICHNEN EINER NAEHERUNGS-KURVE DURCH EINE GEGEBENE DATENMENGE. (METH
KUSTAMM	SCHUELER	5.00	4425	KUNDENSTAMM-DATEI
LIN-GLEICH	SCHLADEBACH	5.00	1005	BERECHNET LINEARE GLEICHUNGSSYSTEME
LLIST	TRAPPSCHUH	5.00	706	GIBT PROGRAMMLISTINGS AUSGEMITTELT AUF DRUCKER AUS. SIEHE CZ NR.12
MAT-GLEICH	SCHLADEBACH	5.00	3454	DURCHFUEHRUNG VON MATRIX-OPERATIONEN
MAULWURF	SCHLADEBACH	5.00	4313	MAULWURF SUCHEN. EINFACHES LOGIG-SPIEL AUS PCC-GAMES
MENUE	SCHUELER	5.00	3067	AUS LEVEL II - HANDBUCH
MONA/PRT	MADER	7.50	8782	GIBT BILD "MONA-LISA" AUF PRINTER AUS (MX-80)
NUCODE	BRUEHBACH	5.00	1234	KOMFORTABLES UMRECHNEN VON DEZ-HEX-DUAL
OTHELLO	WALDAMERO	7.50	9379	LOGIK-SPIEL AEHNLICH REVERSI
PHONE	BRUEHBACH	5.00	2317	ZAEHLT ZEIT UND GEBUEHRENEINHEITEN BEIM TELEFONIEREN
PIDELTA	SCHLADEBACH	5.00	1618	FILTER-BERECHNUNGEN PI IN DELTA UND ZURUECK
PLOT	SCHLADEBACH	7.50	5422	ZEICHNET KURVEN MIT LIN. ODER LOG. ACHSENTEILUNG
PRIMZAHL	SCHLADEBACH	5.00	2511	PRIMZAHL-ZERLEGUNG (NEUE VERSION)
QUIZ	SCHUELER	5.00	3964	QUIZ MIT AUSWERTUNG DER ANTWORTEN
REVERSI/CMD	MADER	10.00	21000	BRETTSPIEL GEGEN TRS-80 (NUR DISK)
ROBOTER	BRUEHBACH	5.00	4004	ROBOTER-JAGD MIT TON
SHOW	MADER	10.00	12085	DEMO-PROGRAMM
SNOOPY	MADER	5.00	1925	ZEICHNET BILD VON SNOOPY
SPACE-INVADERS	TRAPPSCHUH	10.00	12288	BEKANNTES WELTRAUM-SPIEL MIT GUTER GRAFIK UND TON **NEU AUCH ALS DISK-VERSION***
SPANNUNG	SCHLADEBACH	5.00	3848	BERECHNET BELASTETEN SPANNUNGSTEILER (NEUE VERSION)
SPRINGER	MADER	5.00	3246	DIE SPRINGER-TOUR
SPRIT	THALMEIER	5.00	2665	BENZINVERBRAUCH

==> PROGRAMMBIBLIOTHEK TRS-80 USER CLUB <==

PROGRAMM	EINSENDER	PREIS	LAENGE	BESCHREIBUNG
STARWARS	SCHLADEBACH	7.50	8706	STARTREK AUS EACA-HANDBUCH
STERNE	SCHLADEBACH	5.00	1137	EINFACHES ZAHLENRATE-SPIEL
SUPERHRN	BRUEHBACH	5.00	2007	SUPERHIRN-ZAHLENRATE-SPIEL
TICTAC3D	MADER	7.50	6739	DREIDIMENSIONALES TIC-TAC-TOE - SPIEL
TIMER 2	SCHLADEBACH	5.00	2986	BERECHNET TIMER-IC NE 555
TUERME	MADER	5.00	1899	TUERME VON HANOI - GRAFIC-SPIEL
UHR	MULIA	5.00	2602	DIGITALUHR/ DIGITALZAEHLER IN GROSSBUCHSTABEN
UMRECH	SCHUELER	5.00	2181	UMRECHNEN LAENGENMASSE ENGL./DEUTSCH
UNICOP	TRAPPSCHUH	5.00	2317	==VERSION 1.1 == UNIVERSAL-KOPIERER S. CLUBZ. NR.10
VERSENK	TRAPPSCHUH	10.0	12500	SCHIFFE VERSENKEN (GRAFIC,TON) TRAPPSCHUH-QUALITAET
VOKABEL	SCHLADEBACH	5.00	3327	EINLESEN VON 2 STRINGS VON KASSETTE UND DARSTELLUNG IN GRAFIC-GROSSLETTERN
WURM	PIETZSCH	5.00	2733	GRAFIC-SPIEL
ZDIODE	SCHLADEBACH	5.00	1065	BERECHNET VORWIDERSTAND UND LEISTUNG VON ZENER-DIODEN
ZGRAF	SCHUELER	5.00	696	ZUFALLS-GRAFIC