TRS8BIT

PRODUCED AT WWW.TRS-80.ORG.UK

MARCH 2025 - VOLUME 19 ISSUE 1

PAGE 03 2025 COMPETITION - IMPOSSIPUZZE25
ONCE AGAIN, GREAT PRIZES SO GET YOUR ENTRIES IN QUICK

PAGE 04 TA25 NEWS - IAN MAVRIC
PLANNING THE 2025 TANDY ASSEMBLY

PAGE 09 DISK FILES (PART 1) - JIM NORRIS
A RARE FIND AMONGST A BATCH OF OLD TANDY PAPERS

PAGE 22 DEBUG ROUTINES - KEN BROOKNER
CODED ROUTINES FOR TROUBLESHOOTING

PAGE 28 NEWS FROM FRANCE - PASCAL HOLDRY
PASCAL'S LATEST SUPERB RE-ENGINEERING PROJECTS

PAGE 39 SCRABBLE - E.T. FONEHUME
A ONE-LINER BASIC PROGRAM TO ASSIST YOU SCRABBLE FANS

PAGE 41 MODE 16 'ROLLOUT' - KEN BROOKNER
A ROUNDTABLE CONVERSATION WITH KEN, SAM SAWYER AND JOHN ELLIOTT.

PAGE 60 THE 'COMPETITION-COMPETITION'
YET ANOTHER CHANCE TO WIN. A RARE AND UNIQUE PRIZE. JUST A SIMPLE EMAIL REQUIRED.

PAGE 62 IN MAV'S WORKSHOP - IAN MAVRIC
MODEL 3 VIDEO BOARD REPAIRS

PAGE 70 SERVICING YOUR TANDY/RADIO SHACK - KEN BROOKNER
A CASE FOR CONTAINMENT

PAGE 76 U.K. EBAY FINDS
WERE YOU ONE OF THE LUCKY ONES?

PAGE 79 THE WAY WE WERE
LITTLE SNIPPETS FROM THE DAY

PAGE 83 'AMAZED'
GIVE YOURSELF A BREAK AND HAVE A GO. TRY FOR 10 SECS.

PAGE 85 SURFIN' THE NET
STRANGE OR INTERESTING POST THAT HAVE CAUGHT MY EYE

PAGE 89 RANDY KINDIG INTERVIEWS
TRANSCRIPT OF RANDY'S INTERVIEW WITH DON FRENCH AND STEVE LEININGER

PAGE 111 POWER OF THE ROM - VINCE OTTEN
VINCE START A NEW SERIES USING DR. DAVID LIEN'S FAMOUS BOOK 'IT'S EASIER THEN YOU THINK'

PAGE 125 SOFTWARE TOOLS IN BASIC - EPISODE 6 - VINCE OTTEN
THE SHELL GAME

PAGE 142 MAV'S HISTORY WORKSHOP - IAN MAVRIC
WHAT WAS THE MODEL 12?

LATEST UPGRADE NOW AVAILABLE
VERSION 2.5.4

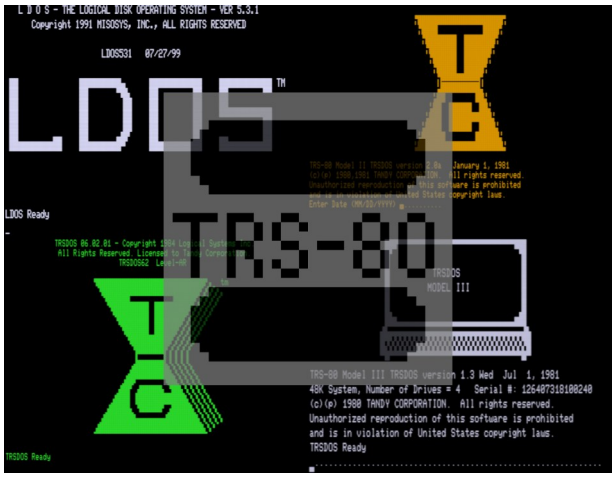
DON'T HAVE A TRS-80?

CRAZY PRICES ON EBAY?

TIRED OF RIFA CAPS BLOWING?

OF FLOPPIES SHEDDING OXIDE?

OR JUST PLAIN DEAD CHIPS?





THEN WHAT YOU NEED IS TRS80GP.
THE ULTIMATE TRS-80 EMULATOR



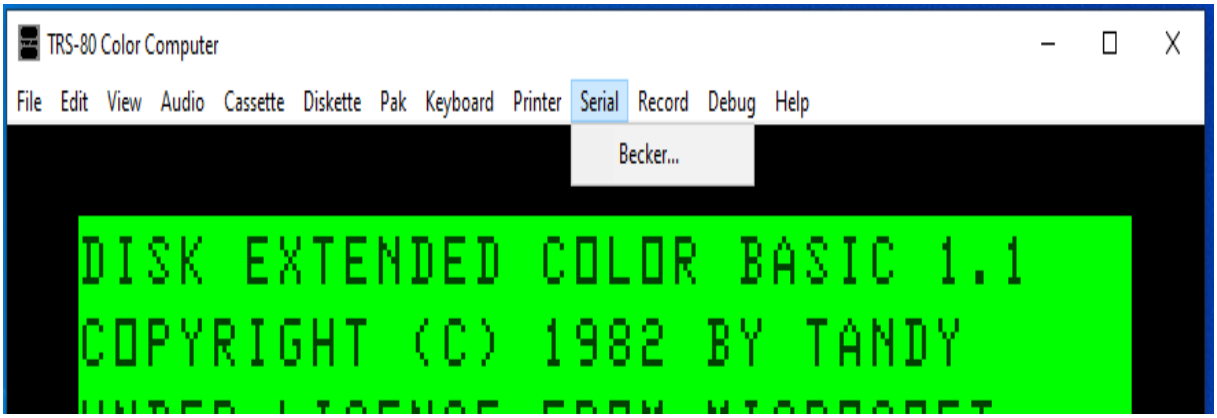
RUNS ON WINDOWS, MACOS, LINUX & RASPBERRY PI.
EMULATES MODELS 1, 2, 3, 4, 12, 16, 6000, MC-10,
COLOR COMPUTER, DT-1, VIDEOTEX AND NOW THE **TANDY 10.**

JUST LIKE A REAL TRS-80 WITH NONE OF THE HASSLE.
BUILT-IN DEBUGGER AND OTHER FEATURES, JUST GREAT FOR PROGRAMMERS.

GET IT NOW AT HTTP://48K.CA/TRS80GP.HTML

UPDATED VERSION, 2.5.4, NOW OFFERS :

BETTER BREAKPOINT GUI WITH SINGLE INPUT WINDOWS.
BECKER PORT SUPPORT FOR THE COLOR COMPUTER.
HARDWARE SERIAL PORT ACCESS ON LINUX.
** TANDY 10 EMULATION **



TRS8BIT's IMPOSSiPUZZLE25

A U.K. quiz program called 'Countdown' has a fascinating numbers game.
You select 6 unseen numbers comprising of a group from 1 to 10 (small numbers) and the four large numbers 25, 50, 75, and 100.
You can choose how many 'small' and/or 'large' numbers you require.
Duplicates, if they occur, are allowed.
Then a 'target' number is randomised from 100 to 999

Contestants then have 30 seconds to calculate, with the 4 arithmetic operators +, -, * and /, the 'target' number using any or all of the 6 random numbers, ensuring that each number is only used once.

At no stage of the calculation can you have a negative or fractional number.

As an example:-
Let's say you have chosen 2 'large' and 4 'small' numbers.
When revealed they were 75, 100, 7, 10, 1, 10 and a 'target' number of 543

Here's one way to calculate the 'target' number:

```
(75+1)*7    = 532
(100+10)/10 =  11
532+11      = 543
```

To win the TRS8BIT IMPOSSiPUZZLE25, you need to write a program, running on a TRS-80 emulator, to take the 6 random numbers, **25,75,5,2,8,5** with the 'target' of **124** and show how to use them to calculate the 'target' number. Remember of course, it may be an impossible task. You can decide how long to cycle the program looking for an answer or, if one is not found, just declare how close you were!

The 'Trashtalkers' team will judge the entries, taking into account, running speed, programing style, language(s) used, entry date and the number of TRS-80 models your program will run on.
.
The winner will receive their choice from one of Mav's wonderful products.

Closing date for the competition is
22:00 hrs GMT on Sunday 2nd November 2025
Entries to be sent to George Phillips at george@48k.ca

MAV'S TANDY ASSEMBLY QUARTERLY COLUMN

TA25 NEWS

by Ian Mavric



Bringing you the latest news you need to know about the greatest Tandy TRS-80 show on the planet. This time I talk about plans for the 2025 show which, which is only 7 short months away.

TA25 IS ALREADY IN THE PLANNING PHASE

The Tandy Assembly team never rests, though we do try to take it easy in the weeks following the show. 2025 is here now and we are already starting to make plans for this years show. Substantially the same the proven format will continue of Tandy Tech Track on Friday with 5 tutorial/educational style presentations, followed by dinner provided by the Marriott for everyone who attends.

Saturday will be the normal 'busy' day with up to three main presenters including the key-note speaker. In the afternoon the TA25 auction will be held. As usual items will be donations and any

money raised goes toward organising next year's Tandy Assembly event. Saturday night will have the famous (or infamous) TA trivia game, which now includes an Expert Round for each of the three contestants as well as a Phone-A-Friend (in the audience) and the fun Audience Pub-Style 10-question trivia game. I aim to find the smartest audience member in the room so attendance is mandatory, and you need to bring a pen or pencil. This year TA24 trivia game winner, Brendan Bergie, will be along to defend his title. Can he make it two years back-to-back trivia wins?

Sunday will have the last of the presentations and mid-after disassembly of the show, usually followed by dinner at the Texas Roadhouse steak house in Springfield for a farewell dinner.

Unofficially, as always, the Thursday has been a day for the early arrivals (this includes me and many who come from far interstate and overseas) for a gathering. Previously we have gone to the USAF Museum in Dayton then to a buffet Chinese lunch not far from there. This gives us a good chance to shake any jetlag and chit-chat about what we've been doing in a non-computer capacity since we last caught up. Dinner is usually something small in the little restaurant at the Marriott.

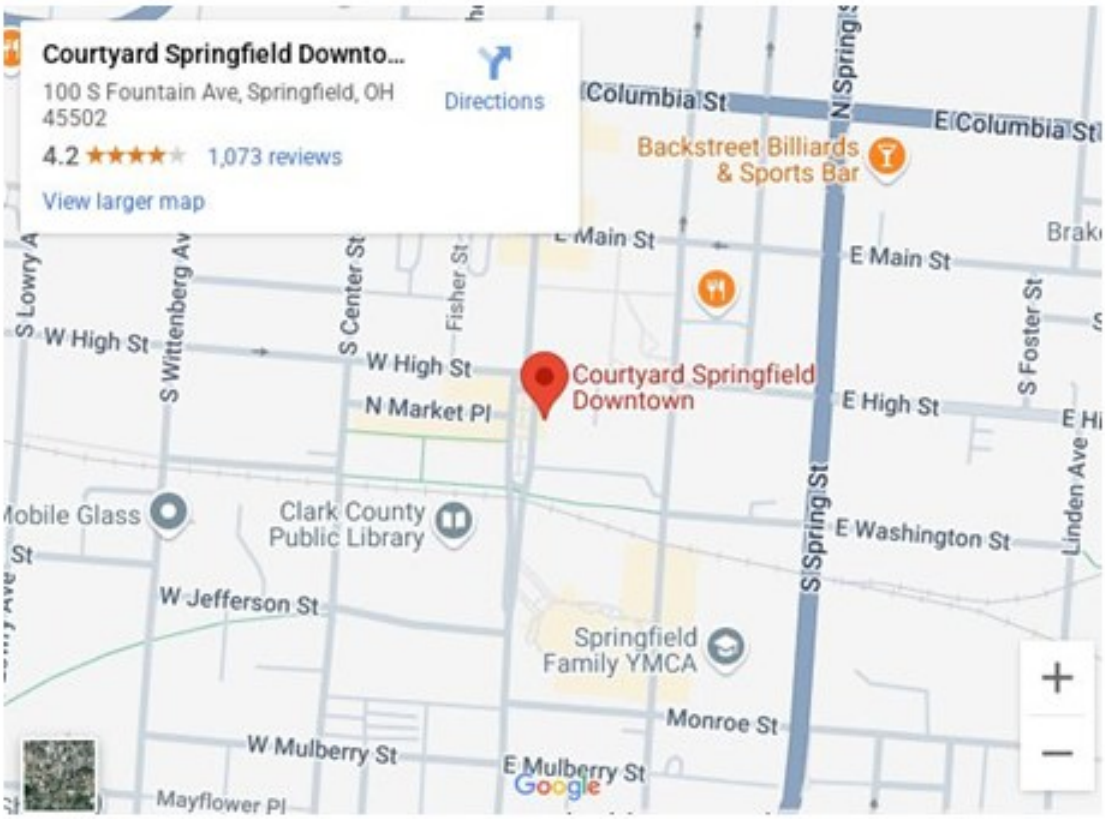
Speaking of the Marriott, as you can see from the masthead the dates have been set for the show, this year it's September 26th-28th, and is at our usual venue at the Marriott Courtyard in Springfield, Ohio. If you are planning on staying at the hotel, I suggest you call

them as soon as possible to get a room, as they are already selling fast.

VENUE

Tandy Assembly is held at the Courtyard by Marriott Springfield Downtown in Springfield, OH.

Please use this link or call the hotel directly at (937)-322-3600 to make your room reservations and be sure to tell them you are part of the Tandy Computer Convention to get a special reduced room rate.



It would be remiss of me not to tell you about the online link to book a room:

https://www.marriott.com/event-reservations/reservation-link.mi?guestreslink2=true&id=1736272541233&key=GRP

however, most people find it easier to just pick up the phone and call them.

SPEAKERS AND PRESENTERS MAKE THE SHOW

At the time of publication we are actively looking for presenters and a keynote speaker for the show. If you have always wanted to attend TA and have a story to tell or something you'd like to teach, we are all ears. Reach out and we'll make a space for you. Did you work for Radio Shack, write a popular piece of software, run a hardware or software company during the TRS-80 era? We'd love to hear your story.

You would be joining a long list of famous people who have spoken at TA like software tycoon Scott Adams, engineer and TRS-80 Coco 1B designer Paul T. Schreiber, programmer Lance Micklus, and most recently TRS-80 Model I designer Steve Leininger.

TRIVIA GAME ALSO NEEDS PARTICIPANTS

Mentioned earlier, the yearly trivia game needs contestants to go up against last year's winner. Do you know as much or more about Tandy and TRS-80 computers as Brendan? Send me a message and I'll be more than happy to sign you up for the game. Now with special Expert Rounds I will tailor 10 questions especially for each contestant. The more obscure your expert round topic the better your chance to win! Ladies, as always are most welcome. In fact if one signed up it would make it a TA first in the 8 shows we have run. Go on, I'm an equal opportunity trivia host. :-)

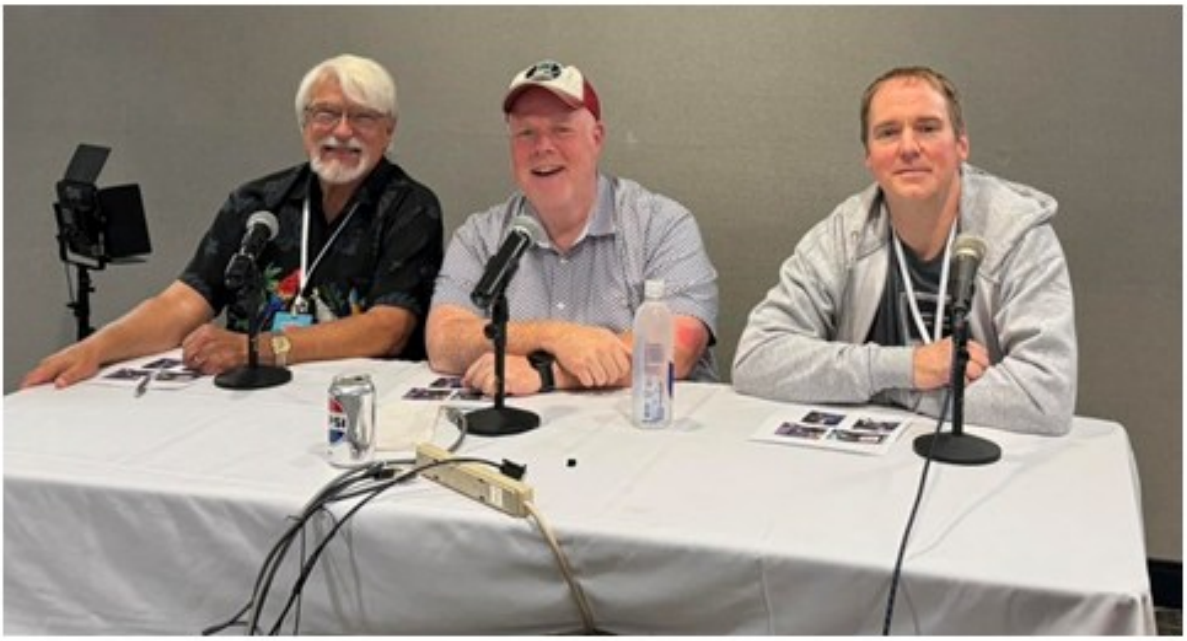


Photo: Brendan (middle) can you beat him at TA25?

If you have questions about the show or want to contact us directly out email address is: tandyassembly@gmail.com

We always appreciate hearing from people who wish to be part of the show. Next time I'll talk about some of the sponsors and exhibitors who have signed up, as well as news on speakers and other attractions. We may even have news for CocoFest attendees who want to see the biggest and best TRS-80 show IN THE WORLD!

Ian Mavric

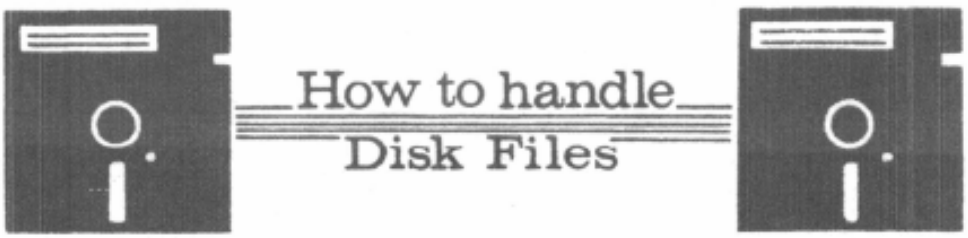
ianm@trs-80.com

TRSDOS NEWDOS LDOS DOSPLUS

DISK FILES

BY JIM NORRIS

"To field or not to field
that is the sequential"



NATIONAL TRS 80 USERS GROUP
MILTON KEYNES

TRSDOS NEWDOS LDOS DOSPLUS

TRS80 USER GROUP.

DISK FILE SYSTEMS FOR MODELS 1,2, and 3

By JIM NORRIS Part 1

INTRODUCTION

This series of articles is aimed at absolute beginners who want to know how to use the TRSDOS disk filing systems from BASIC. There are numerous practical examples, hints and tips. The articles cover all three models and differences between them are highlighted as they arise.

The examples are based on actual commercial applications where generally speaking the object has been to squeeze as much data as possible onto the disks and get as much mileage as possible out of the systems.

Whenever a technical word is used for the first time it will be explained.

These articles are intended to supplement and elaborate on the Tandy manuals and it is vital to have the appropriate manual to hand when implementing examples and overcoming errors.

1. DISKS

Here is a brief description of the disks themselves.

A disk in this context is a circular laminate made of plastic. It is coated on both sides with a ferrite surface. Tandy disks are only used on one side, they are thus single sided disks. The active side is not the side with the label.

Tandy disks are soft sectored. The disk position is known by the one small locating hole situated app .25" from the inside edge - rotate the disk in its cover to see the hole appear in the aperture. Hard sectored disks have holes all round the inside edge.

The read/write head is in contact with the disk in the area of the oval slot. It is worth emphasising that for reliable operation you should not get this area dirty in any way.

There is a write protect notch 1" from one corner. To make a disk write protected put a sticky label over this notch (5.25" disks), or don't put a label over this notch (8" disks) as required.

All master copies of disks should be write protected.

2. FORMATTING

Tand. disks are supplied un-formatted. Formatting consists of writing a magnetic pattern on the disk to delineate the tracks and sectors and to lay out the directory

track.

A track is one of the magnetically defined concentric circles on the disk. A track is made up of sectors. The directory track will contain details of the names and whereabouts of the files when you put data on the disk. A file is an organised set of data with a name and certain characteristics.

Follow the instructions in the manual for formatting and remember that a disk for formatting must not be write protected. If errors occur in formatting then check that the disk is in the drive the correct way (there is an 8:1 chance against getting it right).

If you re-format a disk with data on it then that data will be lost. The formatting program will warn you if there is data on a disk

Disks that keep producing errors should be re-formatted. If they still keep producing errors then throw them away.

3. SYSTEM DISK

There must be a system disk in Drive 0. A full system disk will contain all the machine code routines required to provide the disk operating system, enhancements to Basic, and other functions. A minimum system disk will contain only the absolutely necessary system routines, and therefore has more room for your files. A data disk has nothing on it except your files. Data disks can only be used in drives 1-3. A data disk is a newly formatted disk.

To make a system disk you must backup a formatted disk from your master disk. Refer to the manual for instructions on eliminating unwanted files from system disks. This is particularly important for single drive users.

4. BASIC

All of the TRSDOS file handling system is available from assembly level programming. However these articles concentrate on Disk Basic although in some of the examples there are descriptions of mixed basic and assembly level programs.

Before you insert a disk make sure everything that is connected to the system is turned on. Now insert a system disk in drive 0, press reset and eventually you will see TRSDOS READY.

DO NOT switch the power on or off with disks in the drives.

Type BASIC and <ENTER> (on model 2 type BASIC -F:1)

TRS80 USER GROUP.

DISK FILE SYSTEMS FOR MODELS 1,2, and 3
By JIM NORRIS Part 1

On models 1 and 3 you will be asked to specify the number of FILE BUFFERS, the default is 3 buffers. Buffers are described in detail in the next section. For the early examples one buffer is sufficient. On model 1 type in 1 and on model 3 type in 1V and <ENTER>.

You will be asked to specify MEMORY SIZE. This enables an area of high memory to be protected against BASIC so that you can preserve assembly level programs. For the early examples no protection is required, so just press <ENTER>

You should now see READY and you are in BASIC command mode.

5. FILE BUFFERS

Sections 1-4 were easy and you probably skimmed through. READ this section carefully and understand it thoroughly.

A disk Buffer is an area of memory used exclusively for disk transfers. Buffers are referred to by numbers (1-15). The buffer is used to hold data before it goes to and from the disk.

Disk transfers take place in minimum quantities. Sometimes a request for a set of data (known as a logical record) will actually need data from different physical places on the disk. This is all arranged by the disk transfer routines using the buffer as a collection point. This helps to explain why on models 2 and 3 the buffer is actually bigger than the largest transfer you can initiate.

The buffer also contains a Data Control Block (DCB). From Basic the DCB is transparent, but as a matter of interest it contains at one stage the name of the file and other parameters, and at another stage details of the current disk transfers on that file.

The next few paragraphs are for those who want to look into the memory and see the file buffer at work.

The file buffers are situated at the bottom of the user memory and on model 1 are 290 bytes each. This is 256 bytes for data, 32 bytes for the DCB (data control block) and 2 bytes of unknown use (to the author).

From Basic Command level do the following, pressing ENTER after each line:-

```
OPEN"R",1,"TEST"
FIELD 1,10 AS A$
LSET A$="0123456789"
A=VARPTR(A$)
CLOSE 1
```

DISK FILE SYSTEMS FOR MODELS 1,2, and 3
By JIM NORRIS Part 1

This will have put 10 characters at the beginning of file buffer No.1.
A is the address of a 3 byte string control block. Now do the the following pressing ENTER after each line:-

```
L=PEEK(A)
F=PEEK (A+1)+256*(PEEK(A+2))
```

If you print L then you will see the value 10 being the length of the string.
If you print F then you will see a decimal address being the first byte of the 256 byte buffer.
If you print PEEK (F) then you will see the ASCII code for "0", and PEEK (F+1) will be the code for "1" and so on, i.e. the 10 characters put into the buffer by the LSET instruction.
To see all the characters do the following:-

```
FOR X=0 TO 9:PRINTCHR$(PEEK(F+X))::NEXT X
```

NOTE: It is not necessary to understand the above in order to use the disk filing system.

A file is identified by a File specification.
This is well defined in the manual but it is worth mentioning that every set of data that you create as an entity is a file, and every file must have a unique name. The file spec consists of this name, an extension, a password, a drive number, and a record length.
You can only initiate data transfers between the disk and a buffer, therefore you have to start by connecting the required file to a buffer. When you have finished you have to disconnect the file from the buffer.
You can now use the buffer for other files. You must disconnect all buffers before leaving the program.
Here is the simplest possible sequence for doing this.

```
10 OPEN"R",1,"TEST"
20 GET 1,1
30 CLOSE 1
```

line 10. Connect the file named TEST with buffer 1 and specify random access "R".
line 20. Get the first record of file TEST into buffer 1.
line 20 Disconnect file TEST from buffer 1

This sequence achieves nothing since once you have CLOSEd a file you cannot read the data from the buffer (from BASIC).
In practice a file can remain OPEN all the time a program is running, BUT all files must be CLOSEd before you exit the

program.

6. RANDOM AND SEQUENTIAL ACCESS

There are two types of file access (i.e. the way you can get at the data). These are RANDOM (sometimes known as DIRECT) and SEQUENTIAL.

First of all it has to be stated that file data is organised into records. A record is a number of bytes. On model 1 this number has a fixed value (256), on models 2 and 3 the record length can be any number from 1 to 256.

The record is assembled by you in the buffer and then sent to the file on disk. But where in the file?

With sequential access the records are stored in the order they are sent. With Random access you decide where to send the record. The full power of a disk filing system only becomes apparent with Random access and this document concentrates on this type of file. However to start with here is a description of sequential access.

7. SEQUENTIAL ACCESS

If you are only familiar with tape then you will understand sequential access very quickly.

When you open a file for Sequential access and start sending data then ALL PREVIOUS DATA in that file will be LOST.

You send data as strings or numbers just as you would to tape, making sure there are separators so that you can make sense of them when you read them back.

Certain characters are understood to be separators or delimiters, these are commas, semi-colons and carriage returns.

They have the same effect in a disk file as they do on screen.

Here is a program to demonstrate sequential output and input using text and values

```
10 'TO TEST SEQUENTIAL FILING - SEQ/BAS
20 CLEAR 500:CLS:PRINT"SEQUENTIAL FILE TEST"
30 PRINT"WRITING DATA":OPEN"O",1,"TEST"
40 PRINT#1,"THIS IS A HEADING"
50      PRINT#1,"ITEM1                    ITEM2                    ITEM3
ITEM4"
60     DIM    V(10,4):FOR    X=1     TO     10:FOR    Y=1     TO
4:V(X,Y)=Y+4*(X-1):NEXT Y:NEXT X
70 FOR X=1 TO 10:FOR Y=1 TO 4:PRINT#1,V(X,Y):NEXT Y:NEXT X
80 CLOSE 1
90 END
100 CLEAR 500:PRINT"READING DATA":OPEN"I",1,"TEST"
110 INPUT#1,T1$,T2$
120 DIM V(10,4)
```

DISK FILE SYSTEMS FOR MODELS 1,2, and 3
By JIM NORRIS Part 1

```
130 FOR X=1 TO 10:FOR Y=1 TO 4:INPUT#1,V(X,Y):NEXT Y:NEXT X
140 PRINTT1$:PRINTT2$
150 FOR X=1 TO 10:FOR Y=1 TO 4:PRINT V(X,Y),:NEXT Y:NEXT X
160 CLOSE 1
170 END
```

RUN the program and two headings and 40 values will be sent to file TEST. Note that in lines 40, 50, and 70 the command PRINT#1 means put the following data in buffer 1.

When the buffer is full (256 characters) or you CLOSE the file, the data is written to disk exactly in the order you sent it.

Now RUN 100 and the data will be read from disk, the headings will be printed on screen, the numbers will be put back into the array and then printed on screen.

This example shows how easy it is to mix text and numerical values. But you must read back exactly the same way and with the same variable types as you sent the data in the first place. You can read the file as often as you like, but as soon as you send data again to the file all the current data will be lost.

8. RANDOM ACCESS - RECORDS AND FIELDS

The data you want to store in the file has to be packed into a record and then sent to a particular position in the file. When you read that record subsequently the data can be unpacked and interpreted correctly. With a random access file the layout of each record can be different but the length of the records must be the same.

How do you decide on the record length?. With model 1 this is easy since there is no choice, but with models 2 and 3 you decide the record length by the data that has to be packed into the record.

How do decide what data to pack into the record? This is where the science of computery becomes blurred with the art of system design, but hopefully you will develop a feel for this as you work through the examples.

How is data packed into a record? The record has to be split into sections each having a length and a name. These sections are called FIELDS. The names of the fields follow the same convention as string variables (A$, D3$, etc).

Laying out these fields can be simple and complex and the whole process is known as fielding buffers.

Having fielded the buffer the data has to transferred from your program variables to the fields of the buffer and there are special instructions to do this. But first a definition of the different types of fields that are possible.

9. FIELD TYPES

DISK FILE SYSTEMS FOR MODELS 1,2, and 3
By JIM NORRIS Part 1

a) Strings. If you have data in a string variable you can pack this into a field in a buffer directly. A string variable has a maximum length of 255 characters, and the buffer fields are the same, so to pack all 256 bytes of a model 1 buffer you will need two fields whose total length is 256 bytes.

b) Single byte variables. If you know that a program variable will never be less than 0 or greater then 255 then you can represent its value in a single byte.

c) Integer variables. If you have integer variables in your program (suffix %), or if you are confident that an ordinary variable will not have a value outside the range <-32768, >+32767, then you can store this in a two byte field.

d) Single precision variables. These require 4 bytes each.

e) Double precision variables. These require 8 bytes each.

10. EXAMPLES

Now we will leave the theory and facts behind and concentrate on examples.

Assuming you followed the sequence in section 5 and you are now in BASIC at command level, type AUTO and <ENTER> and then type in the following program:-

```
10 'FILE HANDLING TEST PROGRAM - FHT/BAS
20 CLEAR 1000
30 MS=1:'SET YOUR MODEL SWITCH (1,2 OR 3)
40 CLS:PRINTTAB(20-(MS=2)*10)"MODEL"MS" FILE HANDLING"
50 OPEN"R",1,"NAFILE"
60 R=1
70 FIELD 1,100 AS A1$,12 AS A2$
80 PRINT@5*(64-(MS=2)*16),CHR$(31+(MS=2)*7)"RECORD"R:
90 LINEINPUT"NAME AND ADDRESS, MAX 100 CHARACTERS:":A$
100 LINEINPUT"PHONE NUMBER, MAX 12 CHARACTERS:":B$
110 LSET A1$=A$:LSET A2$=B$
120 PUT 1,R
130 Y$="Y":INPUT"ANY MORE  /N":Y$:IF Y$<>"Y" THEN 150
140 R=R+1:GOTO 80
150 CLOSE 1
160 STOP
```

When the program is typed in press BREAK to get out of AUTO mode and save the program thus:

SAVE"FHT/BAS" <ENTER>

Thoughtful readers will now be asking themselves about the disk buffer that BASIC uses to save and load programs. The space for this buffer is taken by Disk basic when it loads and

the user does not have to make any provision for it.

Now RUN the program and see what happens. If you have a syntax error then step through the line with the space bar and correct as necessary. If the faulty line was between 50 and 150 then you will hear the disk run when you press <ENTER> at the end of the edit. This is because Basic had detected that the program has altered and that a new ball game has begun and so closes the file for you.

When the program works properly it will allow you to type in a name and address and then a phone number and it will pack these into a record and save the record to disk.

If your name and address is <100 characters then the remainder of the field will be filled with spaces. If your name and address is >100 characters then the balance will be lost. In other words LSET is similar to LEFT$, i.e. it packs data from the left. There is of course RSET which packs data from the right.

Our two fields only come to 112 characters so in a 256 byte buffer as this is, the remaining 144 characters are unused and wasted. On models 2 and 3 we could have specified a record length of 112 bytes in line 50 like this:

```
50 OPEN"R",1,"NAFILE",112
```

Even on model 1 we can make better use of the record by putting two sets of names and address in like this:

```
70 FIELD 1, 100 AS A1$,12 AS A2$,100 AS A3$,12 AS A4$
```

We would have to modify the LSET instruction so that for odd values of R we used A1$ and A2$, and for even values of R we used A3$ and A4$.

A more sophisticated technique is to split the record in half like this:

```
65 X=0:IF R/2=INT(R/2) THEN X=1
70 FIELD 1, 128*(X) AS DM$,100 AS A1$,12 AS A2$
140 R=R+1:GOTO 65
```

DM$ is a dummy field. When X=0, DM$ is 0 and A1$ starts at byte 1. When X=1, DM$=128 and A1$ starts at byte 129. Note that we must pass through lines 65/70 for every value of R since the field positions are to be re-computed each time.

The field instruction can have complex arithmetic in it, but do not make it so complex that you cannot debug the program.

TRS80 USER GROUP.

DISK FILE SYSTEMS FOR MODELS 1,2, and 3
By JIM NORRIS Part 1

USEFUL TIP ! The field instructions are more sensitive to ambiguities than other lines of basic, for example

FIELD 1,4 AS C1$ is ok, but
FIELD 1,4 ASC1$ will not work since ASC is a reserved word.
also
FIELD 1,4*(X) AS A$ is ok, but
FIELD 1,4*X AS A$ will not work. Variables even alone like this one must always be inside brackets.

Now we will add some more code to the program to read back the data. Type AUTO 150 and <ENTER>.

```
150*PRINT@5*(64-(MS=2)*16),CHR$(31+(MS=2)*7);:
160*R=0:INPUT"READ RECORD NUMBER";R
170 IF R=0 THEN 210 ELSE IF R<1 OR R>LOF(1) THEN 150
180 GET 1,R:PRINT"NAME AND ADDRESS:";PRINTA1$
190 PRINT"PHONE NUMBER:";PRINTA2$
200 INPUT"NEXT";X$:GOTO 150
210 CLOSE 1:STOP
```

The * after lines 150 and 160 indicate that these program lines exist already.
Save the revised program and then RUN. Type in some names and addresses and then answer N to the ANY MORE question and you will be able to enter record numbers and display the contents.
To stop the program just press <ENTER> to the READ RECORD NUMBER QUESTION.
Apart from arithmetic fields this program example contains all the essentials of random filing.

Disk filing apart, hands up anyone who can't understand lines 40, 80 or 150?

Model 2 has 80 characters per line and the code to clear to the end of screen is 24. Models 1 and 3 have 64 characters per line and the code to clear to end of screen is 31.
If you are on model 2 then you should have MS=2 in line 30.
The expression (MS=2) in lines 40, 80,150 equates to -1 if on model 2 and 0 if on models 1 or 3.
This puts the TAB at an appropriate position in line 40, and positions the cursor at line 5 and clears to the end of screen in lines 80 and 150.

USEFUL TIP ' You will have noticed that I used LINEINPUT in lines 90 and 100. In some name and address systems separate lines are fielded for the name, house and street, town, county etc. I find this restrictive so I prompt users to type

in the name and address continuously putting commas between sections. Normally a comma will delimit a string input command and all following data is lost. However, LINEINPUT will accept all characters into a string. When you come to print the name and address for a letter heading or label use INSTR to find the commas and print from the first character after each comma to the next comma.

11. SUMMARY OF DISK COMMANDS

OPEN"R",1,"NAFILE"....open NAFILE linked to buffer 1 with random access

FIELD 1, 100 AS A1$,12 AS A2$.....specify the first 100 bytes of the record as A1$ and the next 12 as A2$.

GET 1,R.....Get record R from NAFILE on disk and put in buffer 1.

PUT 1,R.....Put the contents of buffer 1 into record R on disk

LSET A1$=A$.....copy the contents of program variable A$ into field A1$ in the buffer.

A$=A1$.....copy the contents of field A1$ into program variable A$

L=LOF(1).....L now has a value being the number of records in file NAFILE.

CLOSE 1.....tidy up buffer 1, update the directory on disk for NAFILE and make buffer 1 free for another file.

Now for a few questions and answers:

Can field variables (A1$ etc above) be used as program variables?.....YES

Can a program variable have the same name as a field variable?.....YES

What will happen?.....you will get very confused.

Can a buffer be opened to more than one file at a time?.....NO

Can a file be opened to more than buffer at a time?..NO

Can different buffers use the same field names at the same time?.....YES

What will happen?.....everyone will get very confused.

USEFUL TIP ! With random access the length of the file is determined by the highest record number that has ever been PUT. If you OPEN a new file and accidently execute PUT 1,500 then your file will now be 500 records long i.e. will take up 128000 bytes on disk (for 256 byte record file). You can eliminate a file by executing KILL"NAFILE" in our example but to do this THE FILE MUST BE CLOSED FIRST.

If you want to keep some of the records before KILLING the file then PUT them to another file first. You can change the name of this other file once you have killed the first

one, but only from TRSDOS.

12. FILE COMMANDS FROM TRSDOS

You can execute some useful housekeeping commands from TRSDOS. DIR on model 2 or DIR (S,I,A) on models 1 and 3 will yield a lot of information about current files, their record length, file length etc

RENAME allows you to change the name of a file and LIST actually lists the content of a file on screen (optionally to printer). If the file contains ASCII text as in our example then you will see this. If the file contains coded numerical values then what you see will be meaningless unless you are very familiar with arithmetic representation.

13. EXERCISES

Starting with the demonstration program develop it to do the following:-

a) Instead of starting at record 1 let the user specify the record number, but don't let him specify a number greater than one more than the current length of file. This is to make him enter new names and addresses sequentially and not leave big gaps all over the file.

b) Let the user see the current record and then give him an option to edit it and automatically save back to disk.

c) Write a routine to sort through the names and addresses, find the commas and print the name and address in lined form as it would appear on a letter head. Compare the method whereby you examine every character (using MID$), with a method using INSTR.

d) Write a routine to read only the name part from every record and print them in alphabetic order on the lineprinter. If your file is only a few records long try and imagine how efficient your routine would be if there were 1000 records.

e) Write a routine to print the names only with phone numbers.

f) Write a routine that inputs a surname and finds and prints the full name and address with phone number. How long would this take if there were 10000 records?

g) How could you identify post codes or town names if there was a requirement to sort and select using these keys?

h) Incorporate an error trap routine with a standardised message printout and optional recovery where possible. How could you detect whether a file exists or not?

A Few tips to help with the above.

Consider creating a new file of the first 8 characters of the surname only. The record length would be 8, except on model 1 where you will need to develop a subroutine to pack 32

DISK FILE SYSTEMS FOR MODELS 1,2, and 3
By JIM NORRIS Part 1

of these 8 byte surnames to a 256 byte record and retreive them again.

Certain characters like * which are not normally used in names and addresses could preceed key fields for sorting. These characters would have to be eliminated when printing the name and address as letterhead or label.

If you have lower case then use this for prompts on screen. Do not let the screen scroll - this is very disconcerting for users. Keep the main heading at the top of the screen so that the user will know what function he is working with.

--== oØo ==--





Simple, Purpose Coded, Debug Routines for Troubleshooting

— ken brookner

Often, the best diagnostics are the simplest. We had really nice, professional diagnostic packages at Radio Shack, but you can do a lot with a TRSDOS debugger and some imagination. The caveat—and it is a big one, is that to use something like this successfully, you have to know how the machine works and how to read a schematic.

Case in point: trying to repair an Iomega SCSI interface board. I had the RS diagnostics and they got me part way, but they died because of no cartridge drive attached. All I needed was to trace bits through the SCSI data write channel. Alternating patterns work best for this—I can use a scope to check all the gates from start to finish. I just needed Debug.

If you don't know how to use TRSDOS Debug, shame on you. It's a great little debugger. Docs are in the TRSDOS or TRSDOS-II manuals.

Let's output a pattern to the SCSI data port 0x40... When you're done diagnosing, hit reset. Enter this starting at 0x3000. Jump to 0x3000 to execute.

See the quick Debug reference later on in this document. Once you're in Debug and have hit F1 to enter overtype mode, begin to enter the HEX bytes below—these are in the column after the address on the far left. Enter it all and don't use [ENTER], or spaces. 06 00 3E 55 D3 40… For the following port output routine you don't need to set any breakpoints unless you want to stop, holding the data lines in one state for you to probe.

F2 takes you out of overtype mode.

```
3000                            .ORG   3000h    ; Start at 3000h
3000   06 00        START:      LD   b,0        ; Counter for main loop
3002   3E 55        LOOP:       LD   a,55h      ; Output pattern 1 (alternating 1/0)
3004   D3 40                    OUT   (40h),a
3006   CD 22 30                 CALL   DELAY    ; SET BREAKPOINT
3009   3E AA                    LD   a,0AAh     ; Output pattern 2 (alternating 0/1)
300B   D3 40                    OUT   (40h),a
300D   CD 22 30                 CALL   DELAY    ; SET BREAKPOINT
3010   3E FF                    LD   a,0FFh     ; Output pattern 3 (all 1s)
3012   D3 40                    OUT   (40h),a
3014   CD 22 30                 CALL   DELAY    ; SET BREAKPOINT
3017   3E 00                    LD   a,00h      ; Output pattern 4 (all 0s)
3019   D3 40                    OUT   (40h),a
301B   CD 22 30                 CALL   DELAY    ; SET BREAKPOINT
301E   10 E2                    DJNZ   LOOP     ; Repeat until b=0
3020   18 DE                    JR   START      ; Loop forever
3022   C5           DELAY:      PUSH   bc       ; Save main loop counter
3023   06 60                    LD   b,60h      ; Delay loop counter (60h)
3025   10 FE        WAIT:       DJNZ   WAIT     ; Simple delay loop
3027   C1                       POP   bc        ; Restore main loop counter
3028   C9                       RET
```

```
START:              3000 DEFINED AT LINE 3
                    > USED AT LINE 17
LOOP:               3002 DEFINED AT LINE 4
                    > USED AT LINE 16
DELAY:              3022 DEFINED AT LINE 19
                    > USED AT LINE 6
                    > USED AT LINE 9
                    > USED AT LINE 12
                    > USED AT LINE 15
WAIT:               3025 DEFINED AT LINE 21
                    > USED AT LINE 21
```

Now two more routines. The first will read a port continuously with a delay, and put the port contents into register A. The second program will do the same except put the port contents into a memory location following a few bytes past the end of the program.

Enter them with Debug as before. Reset to exit.

Result in A Register:

```
3000                            .ORG   3000H
3000   DB 40        START:      IN   A,(40H)
3002   CD 08 30                 CALL   DELAY
3005   C3 00 30                 JP   START
3008   C5           DELAY:      PUSH   BC
3009   06 60                    LD   B,60H
300B   10 FE        WAIT:       DJNZ   WAIT
300D   C1                       POP   BC
300E   C9                       RET           ; Breakpoint
```

```
START:              3000 DEFINED AT LINE 2
                    > USED AT LINE 4
DELAY:              3008 DEFINED AT LINE 5
                    > USED AT LINE 3
WAIT:               300B DEFINED AT LINE 7
                    > USED AT LINE 7


Result at 0x3012:

3000                            .ORG   3000H
3000   DB 40        START:      IN    A,(40H)
3002   32 12 30                 LD    (DATA),A
3005   CD 0B 30                 CALL    DELAY
3008   C3 00 30                 JP    START
300B   C5           DELAY:      PUSH    BC
300C   06 60                    LD    B,60H
300E   10 FE        WAIT:       DJNZ    WAIT
3010   C1                       POP    BC
3011   C9                       RET           ; Breakpoint
3012   00           DATA:       DB    0       ; Data Read Stored at 0x3012


START:              3000 DEFINED AT LINE 2
                    > USED AT LINE 5
DELAY:              300B DEFINED AT LINE 6
                    > USED AT LINE 4
WAIT:               300E DEFINED AT LINE 8
                    > USED AT LINE 8
DATA:               3012 DEFINED AT LINE 11
                    > USED AT LINE 3
```

You will need to set breakpoints to stop these programs on each pass, otherwise you won't be able to read the contents of either the A register or of the memory location 0x3012.

Debug will allow you to set multiple breakpoints.



Normal Debug Display

Note that Debug will only take up the top half of the page, any other programs will show their output below the Debug window.

>**DEBUG ON** Turns Debug on from the ">" TRSDOS Command Prompt and returns you to the prompt.

>**DEBUG** The next DEBUG command after turning it on will take you into Debug. Executing a program will then also take you into Debug.

? is the Debug command prompt. You can type "**H**" to get a short help menu of commands.

B is the breakpoint command. Your program will run to the first breakpoint that you set and will stop. You can then look at the registers or memory and then press **C** to continue execution to the next breakpoint (if you've set more than one). Breakpoints must be set at the beginning of an opcode. You may set up to 8 breakpoints. When you press 1, for example, if the breakpoint is already set, Debug will display the original breakpoint address and the original instruction at that address (breakpoint has changed that instruction to 0xD7, which is what Debug uses to identify the breakpoint.
For example:

? **B #=1 A=2800**

This puts a breakpoint (#1) at address 0x2800. The memory display for 0x2800 now shows a D7.

To delete a breakpoint without affecting any others, press [ENTER] instead of providing a new address for the breakpoint.

LOAD is the opposite of **DUMP**. You can load an executable file into memory but not execute it with this command. The file loads and then command returns to the TRSDOS prompt—the file is not executed.

>LOAD FILENAME/OBJ [ENTER]

The TRSDOS-II Manual Commands & Utilities has several pages on DEBUG and DUMP. With DUMP you can save the routines to your disk. Debug lets you load one of these routines into ram, then exit back the system (before running it) with the "**S**" key. Once there you can then run the DUMP command to save the file onto your floppy or virtual floppy disk. Read the pages on the Dump command to read about how it works.

DUMP [filespec] START=3000 END=[address of last byte] TRA=3000 RORT=[?]

START is the address of the first byte of the program.
END is the address of the last byte of the program.
TRA sets the address at which the program begins execution—0x3000 in our case.
RORT=R loads the filespec but does not execute it.
RORT=T loads and executes the filename.

Example: **DUMP FILENAME START=3000 END=30C9 TRA=3000 RORT=T**
This works for the first routine.

This manual and others are available at the Model II Archive.

Happy debugging!
k.brookner ©2025 V4



Thirsting for adventure?
Brass lantern getting dusty?

UPDATED 2025!



Take your Model I/III/4/4p on exciting new adventures never available until now....

M4ZVM enables your TRS-80 Model I, III or 4 to play Infocom and post-Infocom adventures, taking full advantage of your hardware. Supports old favourites like Zork and Seastalker (including Sonarscope), as well as new-to-Model-4 games including Trinity, Border Zone and Sherlock, plus hundreds more available in the Interactive Fiction archive.



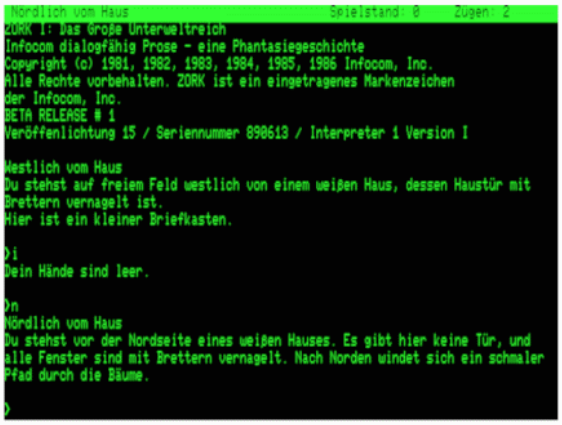
M4ZVM & M3ZVM with sample disks and game compatibility lists are at:
https://sijnstra.itch.io/m4zvm

Source and links to other versions at:
https://gitlab.com/sijnstra1/m4zvm

Actual Screenshots of games new to the model 4, running M4ZVM (on trs80gp) - check links for how-tos & disks



Beyond Zork



Zork 1 (German)

M4ZVM/M3ZVM/M1ZVM features include:
* z1-z8 game execution (z6 support is experimental)
* named game save/load
* accented characters where available
* Optionally set the "Tandy bit" for Tandy customization!

UPDATED 2025 version - faster, better, more bugfixes!

NEWS FROM FRANCE

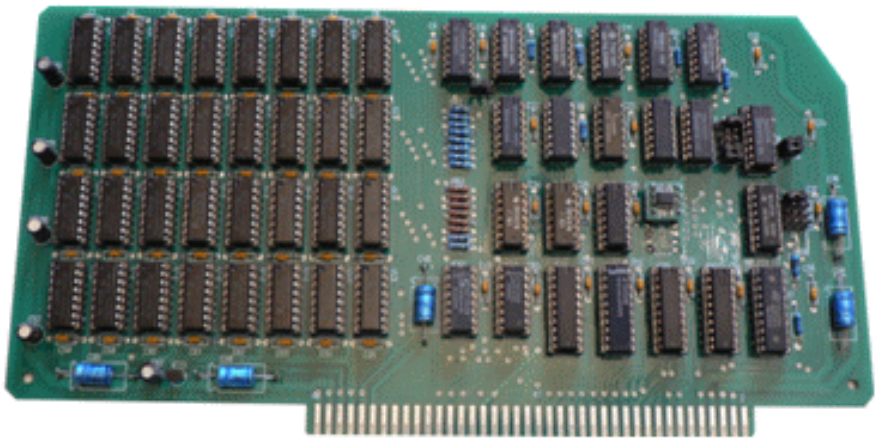
Liberté • Égalité • Fraternité

PASCAL HOLDRY

LAST BEST M12_M16B TRS-80 PSU



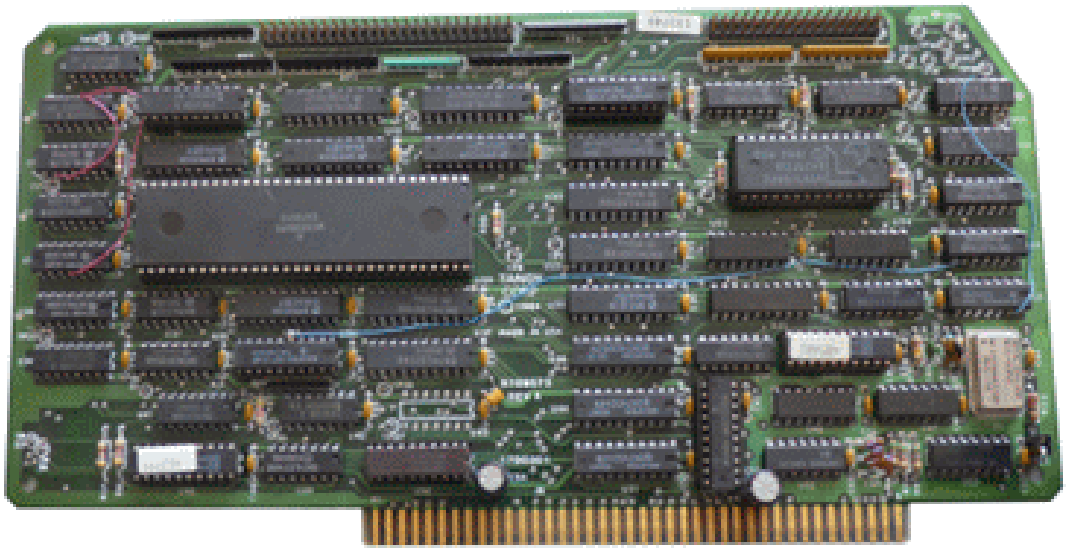
New map 32-64K



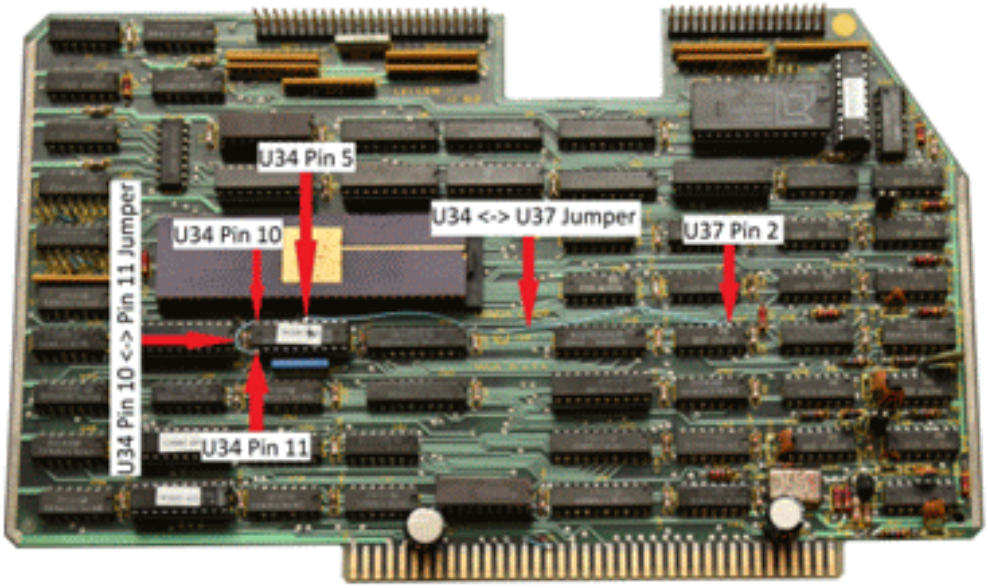
LAST BEST M2 M16A TRS-80 PSU



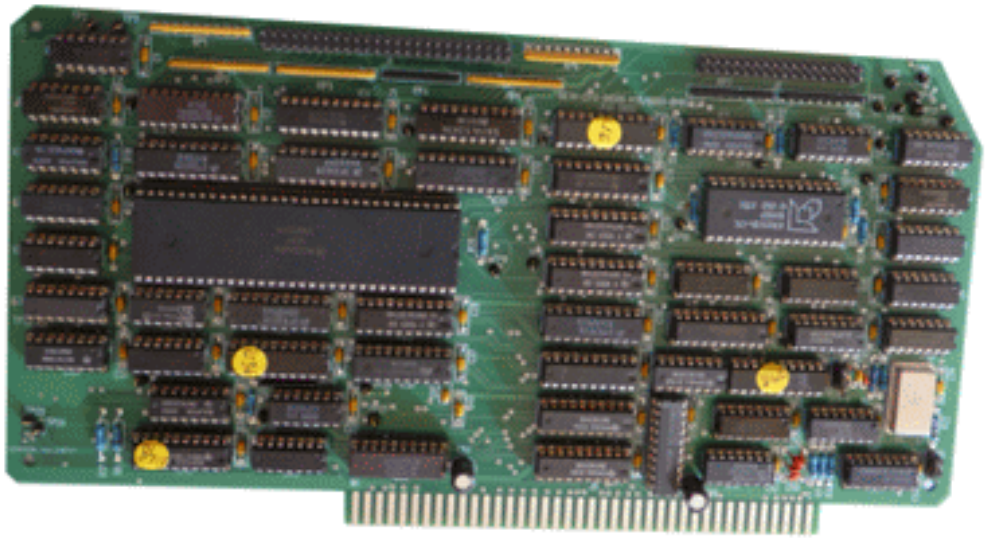
CPU card 68000 8Mhz N°8709573



CPU-68000 XENIX 3.x Cards Modifications



[New-CPU 68000
M16/6000]



LAST BEST M12_M16B TRS-80 PSU

Details

Written by Pascal HOLDRY
Category: Miscellaneous M2,M12,M16,M16B
Publication: December 16, 2024

This card replaces the power supply

TRS-80 M12/M16B ASTEC AA11082

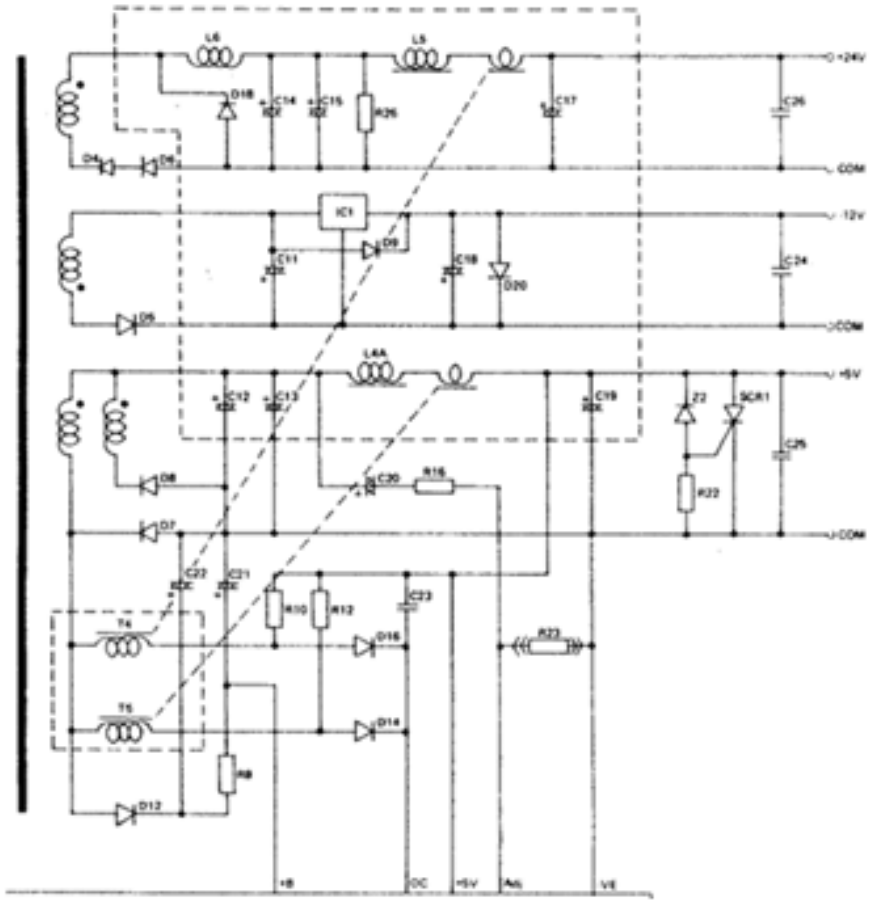
Please note that this is an exchange that I will call standard and is intended to be reversible.



Looking at the schematics I noticed that the TRS-80 M12, 16A, 16B and TANDY 60000 had an independent +12V supply regulated by a LM317 regulator

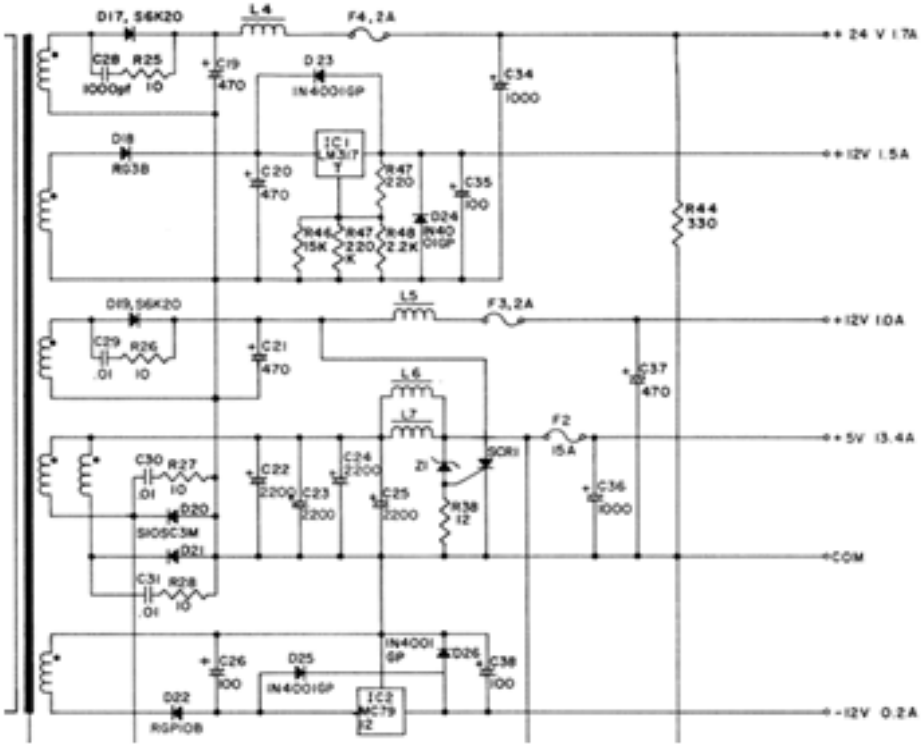
- P14 was reserved exclusively for powering the CRT tube
- P15 was reserved for powering the Card Basket for the M2, M16A and Motherboard + Basket for the M12, M16B and TANDY 6000

Partial diagram of ASTEC AA111080 power supply



Partial diagram of ASTEC AA111082 power supply

Partial diagram of ASTEC AA11132 power supply



Please note, there are two versions " **LAST BEST M12_M16BTRS-80 PSU** " A 200W version and a 320W version Following the numerous power supply problems in the technical bulletins, I recommend a +12V power supply reserved exclusively for the video power supply. This possibility is optional on this card with the S1 shunt and the installation of the small PS1 power supply unit. Please note that this power supply is limited at +12V 4A and +24V 3A output, to avoid melting the output connector and burning the tracks on the card cage. A small fan ensures the cooling of the power supply unit. The output connector is fully mechanically and electrically compatible with the original ASTEC power supply, which requires this power supply to be limited, except for the +5V which originally gives 13.5A and for this power supply 20A!!!

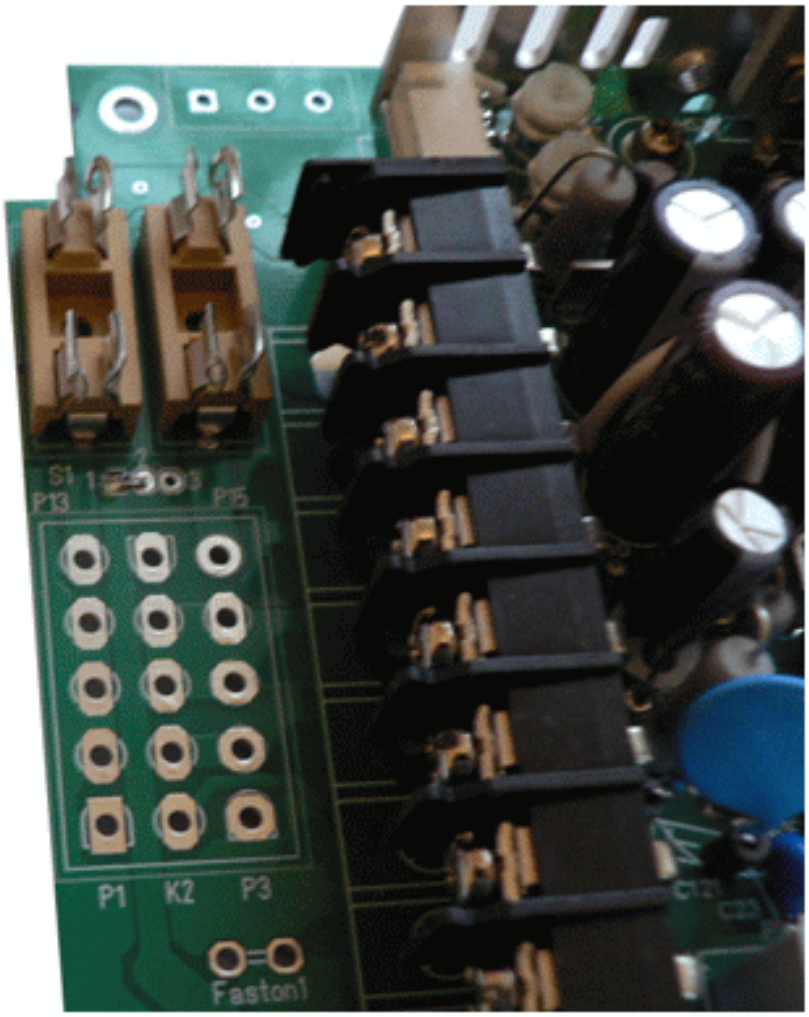
This power supply board is fixed exactly in place of the ASTEC "PCB", with the 5 "ASTEC" screws, plus 2 screws and nuts to reattach the original shielding" no drilling required !!!!! Absolutely nothing !!! But if you have an oxidized M16 like mine expect false contacts on the video card which is on the other side of the chassis !!!

View of the output connector

View of the output connector

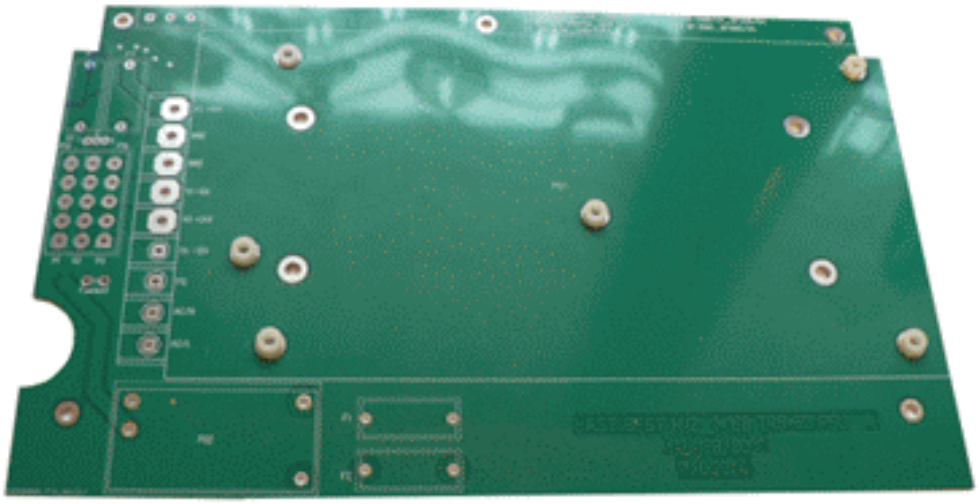


View of the S1 shunt

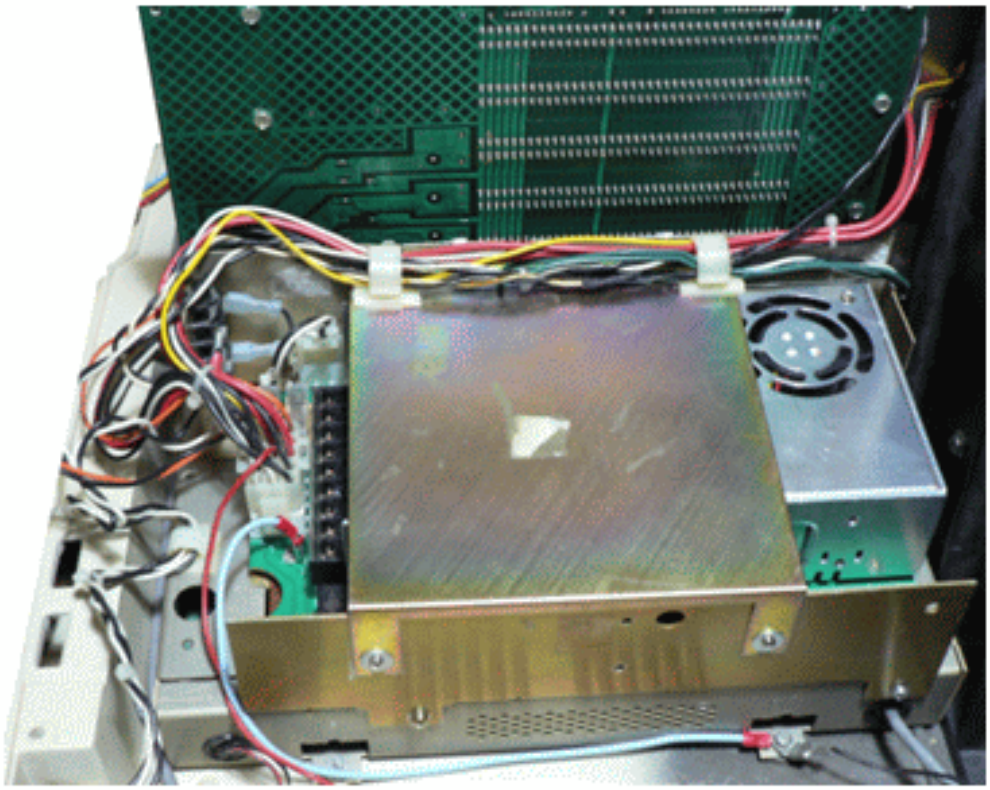


The power supply is isolated from the "PCB" by small plastic spacers to facilitate the flow of cooling air and to comply with the original electrical isolation with the ASTEC power supply, which was connected to the chassis of the M2 or M16 by a "FASTON" lug.

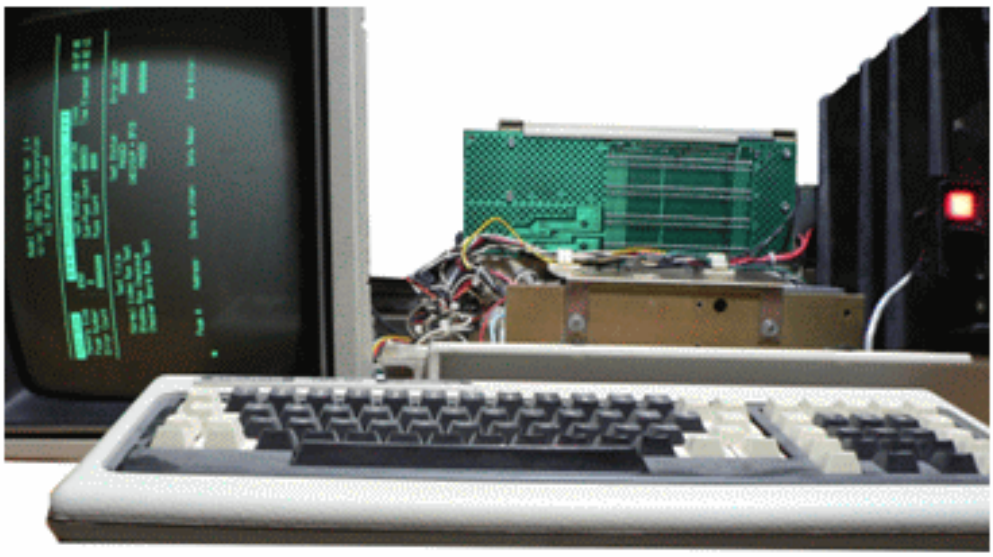
View of the "PCB" with its insulation spacers



Top view of the layout



Overall view



200W version

SPECIFICATION

MODEL		QP-200D			
OUTPUT	OUTPUT NUMBER	CH1	CH2	CH3	CH4
	DC VOLTAGE	5V	12V	24V	-12V
	RATED CURRENT	15A	4A	3A	0.7A
	CURRENT RANGE	3 ~ 20A	0 ~ 6A	0.4 ~ 5A	0 ~ 1A
	RATED POWER	203.4W			
	PEAK CURRENT Note.4	20A	7A	6A	1A
	RIPPLE & NOISE (max.) Note.2	100mVp-p	150mVp-p	150mVp-p	150mVp-p
	VOLTAGE ADJ. RANGE	CH1: 4.75 ~ 5.5V		CH2: 11.4 ~ 13.2V	
	VOLTAGE TOLERANCE Note.3	±3.0%	±3.0%	+10,-6%	±6.0%
	LINE REGULATION	±1.0%	±1.0%	±2.0%	±1.0%
	LOAD REGULATION	±2.0%	±2.0%	±6.0%	±2.0%
	SETUP, RISE TIME	800ms, 50ms at full load			
	HOLD UP TIME (Typ.)	24ms at full load			
INPUT	VOLTAGE RANGE Note.5	90 ~ 264VAC		127 ~ 370VDC	
	FREQUENCY RANGE	47 ~ 63Hz			
	POWER FACTOR (Typ.)	PF>0.95/230VAC		PF>0.98/115VAC at f	
	EFFICIENCY (Typ.)	75%			
	AC CURRENT (Typ.)	3.5A/115VAC		2A/230VAC	
	INRUSH CURRENT (Typ.)	COLD START 30A			
	LEAKAGE CURRENT	<2mA / 240VAC			

320W version

SPECIFICATION

MODEL		QP-320D			
OUTPUT	OUTPUT NUMBER	CH1	CH2	CH3	CH4
	DC VOLTAGE	5V	12V	24V	-12V
	RATED CURRENT	20A	10A	3A	2A
	CURRENT RANGE	2.5 ~ 20A	0 ~ 10A	0.2 ~ 5A	0.2 ~ 2A
	PEAK CURRENT	20A	10A	20A,≦1ms(Note5)	2A
	RATED POWER	316W			
	RIPPLE & NOISE (max.) Note.2	100mVp-p	150mVp-p	150mVp-p	150mVp-p
	VOLTAGE ADJ. RANGE	CH1,2:+10,-5%			
	VOLTAGE TOLERANCE Note.3	±3.0%	±3.0%	+10,-6%	±10%
	LINE REGULATION	±1.0%	±2.0%	±2.0%	±3.0%
	LOAD REGULATION	±2.0%	±3.0%	±6.0%	±8.0%
	SETUP, RISE TIME	800ms,50ms/230VAC 3000ms,50ms/115VAC at full load			
	HOLD UP TIME (Typ.)	16ms at full load			
INPUT	VOLTAGE RANGE	90 ~ 264VAC 127 ~ 370VDC			
	FREQUENCY RANGE	47 ~ 63Hz			
	POWER FACTOR (Typ.)	PF>0.95/230VAC PF>0.98/115VAC at full load			
	EFFICIENCY (Typ.)	83%			
	AC CURRENT (Typ.)	4A/115VAC 2A/230VAC			
	INRUSH CURRENT (Typ.)	25A/115VAC 45A/230VAC			
	LEAKAGE CURRENT	<2mA / 240VAC			

SPECIFICATION

MODEL		QP-200D			
OUTPUT	OUTPUT NUMBER	CH1	CH2	CH3	CH4
	DC VOLTAGE	5V	12V	24V	-12V
	RATED CURRENT	15A	4A	3A	0.7A
	CURRENT RANGE	3 ~ 20A	0 ~ 6A	0.4 ~ 5A	0 ~ 1A
	RATED POWER	203.4W			
	PEAK CURRENT Note.4	20A	7A	6A	1A
	RIPPLE & NOISE (max.) Note.2	100mVp-p	150mVp-p	150mVp-p	150mVp-p
	VOLTAGE ADJ. RANGE	CH1: 4.75 ~ 5.5V		CH2: 11.4 ~ 13.2V	
	VOLTAGE TOLERANCE Note.3	±3.0%	±3.0%	+10,-6%	±6.0%
	LINE REGULATION	±1.0%	±1.0%	±2.0%	±1.0%
	LOAD REGULATION	±2.0%	±2.0%	±6.0%	±2.0%
	SETUP, RISE TIME	800ms, 50ms at full load			
	HOLD UP TIME (Typ.)	24ms at full load			

Bonjour à tous

Attention for the M2, M12, M16A, M16B power supply cards I have restricted the +12V and +24V outputs to 4 and 5A !!!! It was not reasonable to output 10A in 12V or 120W on the PCB and the output connector !!!! same for the +24V !!!! The power supply does go into safety mode at around 130W for the +12V !!!
But people can add a fuse box and connect directly to the power supply output of the +5V, +12V and +24V to add auxiliary assemblies (fan ... etc.)

---

https://www.soigeneris.com/universal-backpack-drive



Universal Backpack Drive

The Backpack is a small SD card 'disk drive' for vintage computers that runs from a single AA battery

Choose Options

- ☐ Preloaded 8GB uSD card [+$5.00]
- ☐ Spare CR1025 RTC Battery [+$1.00]
- ☐ Tandy WP-2 DB9 to DB25 adapter [+$5.00]
- ☐ Sinclair Z88 DB9 t0 DB25 adapter [+$5.00]
- ☐ Epson PX-8 / HX-20 adapter [+$10.00]

$65.00



E.T. FONEHUME

HI DUSTY. HERE'S A ONE-LINER TO CHECK FOR A WORD SCORE IN SCRABBLE. I THOUGHT IT MIGHT COME IN HANDY FOR ANY KEEN SCRABBLE PLAYERS OUT THERE. IT'S VERY EASY TO USE, JUST TYPE IN THE WORD YOU'RE THINKING OF AND THE SCORE IS DISPLAYED. IT'S NOT NECESSARY TO ENTER A 'BLANK', BUT IF IT HELPS, ANY CHARACTER WILL WORK. (THE LETTER IS IGNORED AS THE SCORE IS ZERO IN THIS INSTANCE).

```
0 N$="0103030201040204010805010301010310010101010404080410":INPU
TW$:L=LEN(W$):FORX=1TOL:S=ASC(MID$(W$,X,1))-64:IFS<1ORS>26THENNE
XTXELSEPRINTMID$(W$,X,1);"=";:T=T+VAL(MID$(N$,(S*2)-1,2)):PRINTM
ID$(N$,(S*2)-1,2);" ";:NEXT:PRINT:PRINT"TOTAL";T
```

```
>RUN
? BASIC
B=03 A=01 S=01 I=01 C=03
TOTAL 9
READY
>RUN
? TRS8BIT
T=01 R=01 S=01 B=03 I=01 T=01
TOTAL 8
READY
>_
```


Joe Kay
Director
NORTH COMPUTER WEST MUSEUM
Create Experience Learn
joe@nwcomputermuseum.org.uk
01942 582826
Information Technology facilities & training
Workshops for programming, Electronics & VR
Community classes, clubs & retro games
IT Skills for the young & old
City & Guilds Creditation
Supporting education of local people
Creating opportunities & careers
Disable friendly
Volunteer- ed training & museum operations
NORTH WEST COMPUTER MUSEUM
Create Experience Learn
www.nwcomputermuseum.org.uk
Leigh Spinners Mills, Park Lane, Leigh, WN7 2LB

The Radio Shack Model 16 Rollout

A Roundtable Conversation with Sam Sawyer, John Elliott IV, and Ken Brookner

February 1, 2025



Recorded and transcribed: Ken Brookner

©Ken Brookner, Sam Sawyer, John Elliott IV 2025. All Rights Reserved.

By way introduction, the three of us were working at either Radio Shack or Tandy Systems Design, or Tandy R&D Software when the Model 16 rolled out to the public. Sam Sawyer was one of the design engineers on the Model 16—you will see his signature on the Model 16 schematics. John Elliott IV served as the 68000 kernel developer for XENIX, and as you'll read, prior to XENIX worked on the potential UNOS release. I was working in National Technical Support and Training, Radio Shack, as a technical instructor and was assigned the 68000 CPU and memory cards to teach on the rollout training *blitzes* across the country.

We wanted to keep the feel of the conversation, and to do that we've edited the transcription as little as possible. A few things were corrected, most of the "ums" and similar were edited out, but this is still pretty raw by intent. I've used [laugh] tags, as well, and anything needing some clarification appears within [ ].

We hope you find this roundtable discussion interesting.

"Tandy had the largest number of Unix systems on the planet at that point… it's pretty amazing what we had for a while and then let slip through our fingers." —John Elliott IV

[Ken] "Let's talk about Model 16 and where we were, and what we were doing when it came out. So let's introduce ourselves? Sam, why don't you go first?"

[Sam] "I was trying to think of what I was doing back when it was first introduced. I think by then I was probably working on maybe a Model 4 redesign or maybe 4P or something like that, and of course, finishing up probably loose ends on the Model 16. By introduced you mean how far back did I go?"

[Ken] "Just who you are and what you did at Tandy and maybe how you ended up up there."

[Sam] "Well, it may get boring before you tell me to shut up…"

[John] "Just imagine that the audience we're speaking to doesn't know who any of us are. So it's kind of like you're giving a talk in public, or whatever, so say, you know, hi I'm and I used to and all that kind of thing. That way anybody that runs across this, they'll understand if they're not familiar with us."

[Sam] "Well, I was lucky enough or unlucky in some respects to be hired by Tandy Systems Design back as the Model I was beginning to get into volume production. So, I was in on kind of the floor level, so that helped to be in early. Not as early as Steve Leininger, of course, but Mike Berger was hired along about the same time, and we thought we were design engineers when we first got there, the first few weeks."

"But then we got changed over to Tandy Electronics to train up technicians and troubleshoot really hard problems and basically make a nuisance of ourselves to Kenji, which we were good at—that was probably our best talent. It actually wasn't too hard to get under his skin." [Kenji was the plant Manager for the factory building the TRS80 Model I. He was a transplant from Japan and ran a very tight KanBan management style.]

"Most of the early problems were manufacturing-related, where parts were in backward and wrong parts stuffed and just general sloppiness because they had farmed it out, farmed out the initial board stuffing to some shoddy place. It got better over time. We installed inspectors that inspected the boards to make sure that it was stuffed correctly and just find stupid things that helped a lot."

"Eventually we got kind of ahead of where they thought they ought to be and let us go back to designing stuff and that's when we started the Model II. It was kind of vaguely modeled after the Intel development system in terms of the bus and general design concepts and stuff, and the Model II was the predecessor to the Model 16."

"We were pretty free to do kind of whatever we thought was right. We didn't get a lot of input from or push around." - Sam Sawyer on the hardware development process.

"It used the same system bus as the Model 16, except that the Model 16 also had a bus and memory and so forth mirroring much of what was in the original Model II, just a different processor, and of course, the 16 bit as opposed to an 8-bit, and faster clock speed as well."

"Berger and I kind of split up the various responsibilities in terms of dividing who designed what. Berger probably took most of the difficult design stuff associated with the 16. One of the things I did that was kind of critical was the interface between the two processors, and I got some help from —The technology I used was a programmable array device."

"First time I'd ever used it. I'd done some design, discrete design work, but I'd never done any programmable work, programmable logic work. So the company that was pioneering that work was MMI. I can't remember the total name of it, but Brad Mitchell was the rep for them and he was a very cooperative, helpful guy and got me up to speed on the stuff, and I explained the interface and he helped me with the equations to get it working, and he, of course, passed on a lot of his knowledge to the other engineers as well."

"I guess I could talk about how I got the job to begin with…"

[Ken] "If you want to…" [laughs]

[Sam] It goes way back... A friend of mine from college, we graduated the same year in the same major, and he had moved to the Dallas-Fort Worth area, and I was still stuck in Lubbock doing weird stuff or stuff that wasn't necessarily appropriate for my education, and he sent me an ad that Tandy Electronics had put out for design engineers, and he said, 'I think maybe you'd be interested in this.' So I followed up with it and interviewed with Jack Sellers and he hired me."

"The major experience I got prior to that with microprocessors was I was working for a company called Borns, the trim pot guys, but they had their fingers in a lot of other things. They had a card reader system that they had leased out to the Bell System telephone companies in the area, in the Lubbock area, basically in the southwest, and I had a mini computer that dialed up these remote card readers and read the cards and stored the data on tape. So I was in charge of an area to get all that stuff—keep all that stuff working."

"Penny and I drove thousands of miles in a car that they had provided, and finally got so sick of driving that I designed a system that I could run there at the house that would dial up a spare card reader that I had and I could test out the boards. There were five or six boards in there that would go crazy on a time or get intermittent. So I could get all the intermittent crap out by running it there, and I put together a nice paper about it."

"Then designed the control program with running a 6502 kit that I'd put together. It was a 6502 computer kit, and I hand assembled this program. It was about a 1K program by the time I finished it—hand assembled it without an assembler. So I learned how 6502 worked quite thoroughly, which is not a really difficult part."

[John] "That's pretty hardcore, Sam. I'm impressed."

[Sam] "[laughs] It was helpful, you know, you never know what you learn and how it's gonna help you later. I learned to work without an assembler, but I enjoy using an assembler a whole lot more."

"Most of the early problems were manufacturing-related, where parts were in backward and wrong parts stuffed and just general sloppiness because they had farmed it out" - Sam Sawyer on early production issues.

[Ken] “Well, yeah...”

[Sam] "I wrote a good paper on it, and it was suggested that Warren's produce some more of them, but they were getting kind of tired of that system and wanted to get rid of it, so they didn't jump to do it. I put together a good paper and I learned how to do that. Well, I did some of that in college as well. I could even show it to you if you're interested seeing it."

[Ken] "That'd be fun!"

[Sam] "I can't imagine you would, but anyway. [laughs]

[Ken] “What about you, John?"

[John] "Well, my name's John Elliott IV. I used to work at Tandy R&D Software. I got my start in computers in like junior high on a Teletype 33, talking to a HP system that did BASIC. While I was going to college, I got a part-time job programming a very fully loaded Model I with business stuff—accounts payable, accounts receivable, all that kind of stuff. The owner of the shop had gotten the software somewhere, but he didn't program, so he needed someone that could get in there and he'd tell me what he wanted and I'd go do it."

"One thing led to another and there were situations where I wanted to do things that BASIC just could not go fast enough. So I taught myself Z80 assembler so I could do USR routines from BASIC. And, added that feature. One of those was like say a character limit for a string that you want to save, let's say 40 characters or something. So I wrote a little assembler routine that BASIC could call that put a bar on the screen that showed you how many characters you had and stopped accepting characters when you hit the end, and then returned that to BASIC."

"That led to me fiddling around with TRSDOS supervisor calls on the Model I, and I ran into things where the doc was just wrong. I filed, I don’t know, two or three problem reports about them and one day I got a phone call from Tandy saying, 'Hey, we're starting up a QA department. You want to come work for us?' And you know, I was 19. I was like, 'Okay, sure.' [laughs]"

"So I got flown out to Fort Worth and got interviewed by a whole bunch of folks and ended up working in the newly formed QA department. I was testing software on a Model III, I want to say. I did that for about a year, did some OS QA work to check documentation, made sure the system calls worked the way they were supposed to and all that."

"Eventually—and I don't even remember who it was. I'm pretty sure I was working for Penny at the time—but eventually they said 'Yeah, we're throwing you up.' I was on ninth floor of tower two and they said 'Yeah, we're throwing you upstairs to 13' which is where all the Systems folks

were. So I basically got kicked upstairs and started learning all kinds of stuff."

"I didn't know C yet, so they sat me down at a terminal to the VAX, which at that point had moved off of DEC's OS and they installed Berkeley 4.1. So I started reading man pages and doing all the things, and at some point, the work for Model 16 stuff started happening. I was learning C, and they basically brought me a 68000, you know a Motorola programmer's reference or whatever the equivalent was back then and said 'Here, learn this' and I did some assembler to help test the boot code."

"Because at that point, at least someone in systems software had said, 'Yeah, we're going to go with Charles River Data's UNOS system, and we're writing the I/O subsystem for it on the Z80 side, and they're providing the kernel on the 68k side, and they would just send us drops, right?' And things wouldn't work, but so I started learning the 68000 and trying to write a little monitor program. When you do that you have to take all the hardware that has to be taken care of, set up all the vector handling routines, the exceptions, and you know, you're bringing up cold hardware."

"I did that for a while but eventually that pretty much got abandoned because they got their kernel working enough to where you could get a shell prompt. Then we started trying things out on UNOS. It was written in C that had a very different API. Like set-UID bits and set-GID bits were patented. So they couldn't use those and they had other mechanisms, system calls that you had to use that functionally did the same thing, but didn't function like set-UID or set-GID bits."

"They had some nifty utilities. I particularly liked their email utility and that was—the UNOS's mail utility was the genesis of me writing xmail for XENIX. I really liked their utility so I said, 'Oh, I'm gonna write one.' I put a lot of work into it."

[Ken] "xmail was nice!"

[John] "Still use it a little bit. It astonishes me somewhat that my fingers still remember all the buttons to push when I boot up my virtual XENIX in trs80gp [TRS-80 software emulator] and my fingers just seem to know what to do. [laughs] It's weird. I guess that's true for using vi, but anyway."

"We worked on UNOS for like nine months. Getting it ready, finding bugs, helping them get things fixed. I don't think they'd ever ported an OS to—no offense Sam—but to a processor board that had such a primitive MMU, oh the Model 16's MMU did the job, but it was about as low as you could go and still run a protected UNIX. So there were some issues."

[Sam] "No offense taken."

[laughter]

[John] "Oh, thank goodness, no guided missiles heading towards my house, okay. Well, it was, I mean, back in those days, the Tandy design mandate was inexpensive. Make it as cheap as possible, but still do the thing, and all of us had to do that. That was just how it came down from on high at Tandy."

[Sam] "It was physically related. Like if you look at even the smaller Model 16 CPU boards, that thing is pretty well packed. You're not going to jam too much more on there."

[Ken] "That's true. Yeah, there's a lot there."

[John] “Really was.”

[Sam] “Without making the package bigger and heavier and whatever."

[John] "Your idea with the GALs was awesome for that. Now we got rid of the ears on the dog-eared board. Yeah?”

[Sam] “Yeah, right.”

[John] “PALs. That was neat and it's weird later on in my career kind of after the Model 16. Somebody in TSD taught me how to program and burn PALs, and I can't remember if it was you [Sam] or not, but I was just delighted. I was like, 'Oh, this is so cool.'"

"The reason it even came up was there was a particularly stupid tape drive board that folks wanted to use for backups, and that was either on the 286 machine or the 386. I want to say 286, but I'm not 100% on that. The problem was that the board had been designed with DOS in mind, which meant that you basically had to poll the sucker to find out what it was doing and when it was done and when it had data ready, and doing that in XENIX was anathema, right? The whole system would just freeze while it's sitting there polling the hardware port."

"The funny thing was, though, I took a look at the schematic of the board and all the right signals were there to interrupt when it needed—had something ready or needed attention and it had a PAL on it. I was like, 'Oh, you know what? I bet we could just reprogram the PAL' and whoever I talked to over in TSD are like, 'Oh, yeah, huh? Okay,' and they just showed me how to lay out the new set of equations that you wanted and burn a PAL, and holy crap, it worked. I was just like, this is the coolest thing ever. Hardware that's kind of like you can program a little bit like software."

"I think it might've been you, Sam, but I can't remember, honestly.”

[Sam] “It could have been because it was a new thing and talent that not everybody had.”

[John] “It might've been Jerry [Ballard]. I honestly don't remember. I just went and said, 'Hey, I want to do this, and how do we do that?' They're like, 'Oh, okay. Here's the PalBurn utility.' I think I was given a couple of blank PALs. 'Here, go have fun.'" [laughs]

"I can't even remember… The card had a couple of the interrupt lines available. So I can't remember if we had to do a jump to get that to where the PAL would drive the interrupt line. But yeah, being able to set all that up in the PAL, that was just, to me, just amazing stuff. It was fun.”

"One of the things I did that was kind of critical was the interface between the two processors... First time I'd ever used it [PALs]. I'd done some design, discrete design work, but I'd never done any programmable work" - Sam Sawyer

[Sam] “It really is, and it was great groundbreaking at the time for sure."

[John] "Yeah… Anyway, rewinding back to the Model 16. I don't remember how many actual 16s we had. I vaguely want to say we only had a couple. And like our lead—I don't know if you

remember Erick Schmidt”

[Sam] “Oh yeah, Eric was a close friend.”

[John] “I think he had one, and I think we might have had one other one, but it was just basically an upgraded Model II, right? We didn't have any plastics or any of that kind of stuff. That was one of the last things that typically happened."

"Somebody just swapped out the power supply and gave us a memory board and a 68K CPU board. We worked on UNOS for nine months, but unbeknownst to us, Microsoft had found out somehow we were working on the Model 16. You know what though, I'm wondering if the 16 was already out at that point when we were working on UNOS. There was a whole bunch of negative press that Tandy was getting because all we had was like TRSDOS-16, which was just awful."

[Ken] “Well, we had said we were going to come out with a multi-user system—and we didn’t for a while”

[John] "So, yeah, we didn't, and there was no software for TRSDOS-16. So it was getting panned up and down. It's like, 'Well, why the hell should I buy your hardware if you don't have anything that I can do on it?'"

[Ken] "So, how close were you guys to having UNOS finished?"

[John] "We had made our first stab, let's call it an alpha, at a distribution disk. In other words, you take the floppies, you run the installation on a Model 16 and it sets up the hard drive and gets you going. We had the first version of that when we were informed that, 'Oh yeah, we're not doing that anymore. You know, we got a sizable enticement from Microsoft and we're going with theirs.'"

[Ken] "[laughs] So if you just stayed with UNOS, how soon, how much quicker would you have had it than XENIX?"

[John] "I'm really terrible with associating time and events. So I would have said we probably would have had UNOS out in another month. Given that Tandy would typically release things as soon as they could, despite us trying to fix bugs and whatever, I think a month or less we would have gone out the door, but because of the sizable Multiplan and BASIC enticement that was offered to merchandising, they said 'Yes, yeah, we want the free stuff. We'll go with XENIX.'"

"We were pretty upset. [laughs] Nine months of work. Yeah. Just throw that away. We're not doing that. But I'd say in the long run it turned out for the best, and man, I gotta give props to the Microsoft people that did… They basically ported XENIX in like three days. Bob Powell wrote the Z80 code and Jerry Dunietz, who I got to actually meet at Microsoft at one of his large number of company anniversaries—I think he was with them 30-35 years before he finally retired. Anyway, Jerry was their MMU specialist and that's one of the biggest things you gotta change when you port XENIX to new hardware."

"They just pulled off a miracle, I'm just… It was just amazing what they did for what little time they had and brought something demo-able. Now unfortunately for us in System Software, they obviously hadn't gotten all the bugs out. The hadn’t recompiled all the utilities to run. They just had the minimum to boot up, have a shell prompt and say 'watch this.'"

"We were informed that, 'Oh yeah, we're not doing that anymore. You know, we got a sizable enticement from Microsoft and we're going with theirs.'" - John Elliott IV on abandoning UNOS for XENIX.

"So we had to deal with the lack from that, later. They sent us like distributions on dump and restore, UNIX utilities that were unreadable, and Frank and I at the time were just making fun of them. He's like, 'Hey, what do you want to bet these disks are blank?' They just said they would ship it to us, [laughs] and they did. It was unproven, but we were kind of suspicious."

"So months dragged on while we were trying to get them to fix everything because at that point Frank and I and the rest of the software crew were really good at testing stuff. I remember when we did the final version for the Model 4, we actually got a letter from Bill Schroeder, who I think was the president of Logical Systems, complimenting Frank and I for our ability to totally destroy an operating system."

[laughter]

"TRSDOS was the better for it. 'Cause you know LS-DOS was basically adopted as the last OS for the Model 4, and because of all the really hard work that we put in, it ended up being way better, much higher quality. So we were grateful for that. But dang, we put the fear of Tandy into Logical Systems because they found out they couldn't get away with much. [laughs] Oh man, that stuff was fun."

"But anyway, we turned that same kind of attention onto XENIX and Microsoft and found all kinds of evil problems. I mean, we were pretty evil to those operating systems, but you need to do that kind of testing in order to make sure there isn't anything nasty lurking. I can't remember how much longer but that went on for several months before XENIX finally came out.

[Ken] "We were thrilled. It was great! We were really glad to see it and we had machines running it [in Technical Support] the day we got it."

"Gary Kueck, I think he ran the first system and he set me up with a login. We were both tech reps… Was I—what was I? Yeah, tech reps.... We had cubes across from each other.

[John] "I remember him. Cool!" [laughs]

[Ken] "That's where I learned XENIX and UNIX. I had no experience with it and it was like a brave new world. It was really cool."

[John] "It really was. I was having a blast learning all the things that I was learning and looking for things to apply what I was learning in the brave new UNIX world. I don't know if y'all remember this, but I also wrote T-Shell."

[Ken] "No, I didn't know that."

[John] "T-Shell was the 'let's make it a little more TRSDOS-like,' to help people kind of transition, and I mean, the back end of T-Shell was basically the system function call which just asked 'sh' to do the thing. [laughs] But T-Shell was kind of cool…"

[Sam] “It was a good bridge…”

[Ken] “It *was* a good bridge.”

[John] “It had aliases and it made things a little more TRSDOS-like for folks that needed that. Yeah, I think I actually did that one before I did xmail."

[Sam] "I actually used T-Shell a few months ago because I couldn't figure out any other way to do it..”

[John] “Probably to copy stuff from TRSDOS…”

[Sam] “Yeah, exactly…”

[John] “There was a utility that did that but I think T-Shell made it a little bit easier. I'd kind of forgotten about T-Shell and I stumbled across it again once we started playing with XENIX recently. I was like, 'Oh yeah, I remember this!' [laughs] It was a little bit weird to run it, and wow, okay, I'd forgotten I'd stuffed all this stuff in here."

"John [Prickett] came in my cube and dropped a couple of pieces of paper that described the 16 and a bunch of data books. He says, 'You're going to teach this on the training Blitzes in about three weeks.'" - Ken Brookner

"It's weird looking at something you did so long ago, hadn't thought about for decades, right? I imagine things were a little bit like that for you when you started tinkering around with our new hardware, Sam."

[Sam] "Yeah. Some learning curve, of course, but I picked up the PALs pretty quickly, and I don't think you ever forget Z-80 if you've learned it once."

[John] "I'd say that when I left Tandy, Z80 was probably the thing I was the most proficient in. I mean, I could disassemble by sight. I could just look at opcodes and go, 'Oh, okay. Yeah, I see what that's doing.' Wasn't quite there with 68K, it was 16-bit, but having the things that we did tool-wise in UNIX helped a lot to where I didn't have to do a whole lot to be looking at things at that level, whereas the Z80 was a little more oriented towards writing the lower-level stuff."

"Most of the XENIX kernel for example is in C. So it isn't until you start getting down to fiddling with hardware or setting up the basic structure that the OS needs to handle a processor, you don't really have to mess with assembler."

[Ken] "Well, I guess I'll jump in and do a brief intro and about what I was doing when the 16 came out. I was hired into computer customer services in 1980. They wanted a hardware guy who could help customers interface their serial printers or their light switches or whatever it is that they wanted to do. I had a Model I and I had some Fortran at TCU and knew BASIC really well and some Z80 and I'd interfaced my Model I to a Teletype 33 for a printer because I couldn't afford the two grand for the Quick Printer 2."

[John] “Sounds right...”

[Ken] "I'd also written code to interface it to a Baudot Model 15, which is a five-level code, instead of eight bits, like ASCII. So they figured I could do it, and so they, John Snodgrass, hired me and I had a great time. It was punch out at 5 and go home at 11. There was so much stuff to play with, and the customers were great."

[John] "[laughs] I was just agreeing with you. It was similar for us."

[Ken] "So, did you stay late too?"

[John] "Oh yeah.. I mean the problem was there were so many things to play with! I'd finish the day job, right? And then I'd be like—I need to work on xmail, or I need to fiddle with whatever, right? Or, I want to build this game and try it out later, or whatever would come down on USENET. Oh… Just so much..."

[Ken] "This was 1980 and about nine months later, Ronnie Franklin [manager of National Technical Support and Training] recruited me to work as an instructor, and I worked for John Prickett [Training Coordinator]. That's what I was… I was an instructor when the Model 16 came out, and John came in my cube and dropped a couple of pieces of paper that described the 16 and a bunch of data books. He says, 'You're going to teach this on the training Blitzes in about three weeks.'"

"So I had the idea that if I could make it fun, amusing, a little irreverent, because nothing coming out of Tandy was irreverent, so it would be different." - Ken Brookner on writing the *XENIX Information* documentation.

"Maybe it was four weeks or something. Basically we're going to all the regions. I think, I'm trying to remember where the regions were, the three or four regions and divisions in the regions. We're going to different parts of the country to do the intro [component level service training] for that and the pocket computer and the WP-50 Daisy Wheel was coming out and there was something else."

"I was teaching the 68000 boards and there wasn't anybody to teach me how they worked so I had to sit there with the schematic and the notes I had and [laughs] the data books and figure it out and it was actually a lot of fun, and it went well."

[Sam] "Kind of scary though, I would guess."

[Ken] "Oh, it was, it was. But I was used to teaching by then and you just kind of fill in the blanks. If you forget something, you just come back to it, and I had good notes, too. [laughs] We did Chicago… I flew into Chicago and a week later I flew out of Chicago. We did the training in the O'Hare Hilton. I was never outside. Got off the plane, walked through the tunnel to the Hilton and a week later I walked back to the tunnel to the plane. [laughs] So that was kind of weird."

"So then before the 16, before XENIX was released, I became a technical rep. So for anybody who's wondering, I was in service instead of R&D or writing software. Our vision of all this was a little bit different, and so then we started supporting it, you know, people fixing it and how to install XENIX and where everybody was learning. I had a couple of XENIX—a couple of UNIX

books."

[John] "Was that around the time where you came up with that just amazingly excellent guide to XENIX?"

[Ken] "Yeah, exactly. [XENIX Information Series] Because our techs were having trouble adjusting to it and the manuals were really dense, and, I mean, they were great manuals, but there was a lot there."

[John] "They weren't meant for beginners."

[Ken] "Yeah, and they were kind of boring. I mean, a guy that's worked all day and now he has to go home and study this stuff."

"So I had the idea that if I could make it fun, amusing, a little irreverent, because nothing coming out of Tandy was irreverent, so it would be different. One tech to another, that I could get the…"

[John] "...task enjoyable." Yeah, I would say you nailed that."

[Ken] "Thank you. Yeah, it worked. Franchise wanted it, and our customers wanted, it and I was amazed. Philip Hurrell helped me a lot on it. He was over in Customer Services, and later, Valerie See took it on and kept it going. Yeah, it was a lot of fun. That was my big contribution to the 16. [laughs]"

[John] "So you did your huge training blitz, I guess. How long did it take you to travel everywhere? I mean, there are tons of districts and regions, right?"

[Ken] "Yeah. We did it in four places. The first one was in California. It was Huntington Beach —I think."

[John] "Nice!"

[Ken] "So we would have everybody book into a large hotel and then we would have a really big conference room and we would set up some equipment, and then we just did it as a big lecture kind of thing. With schematics, we always taught to the schematics."

"But it went pretty fast, and it was about two days, two full days, each spot. So I did California, then I did Chicago, and I did some in Fort Worth. There was one that I didn't go to. I think Prickett and I switched off on that one. We did Tarrant County, we had one at Tarrant County Convention Center."

[John] "Wow…"

[Sam] "It's a big venue…"

[John] "Yeah!"

[Ken] "I think where we were was part of the Tarrant County Convention Center. Maybe not, but I want to say that it was there in one of their smaller areas. It was a big place and they had lots of little, big rooms squirreled away. So that was okay, and I had to learn the pocket computer because I subbed, I was a backup for Joe Manning, who was another technical instructor, and so he did the pocket computer and I cross-trained on that and he cross-trained on the 16, so if one of us got sick, we could pick it up."

[John] "Smart."

[Ken] “So that was Ronnie Franklin and John Prickett that set all that up. I worked for John and he was brilliant and a lot of fun to work for, and with. I learned a lot of stuff from him. Z80 programming on the fly, he taught me, and really a great guy. So, that was a lot of fun. That's probably pretty much all I remember about the 16 rollout other than people were really anxious to get the multi-user XENIX™. Upset when they couldn't get it.”

[John] “Yeah, especially I remember the customers that bought a 16 and then found out that they couldn't do much with it—yet. Frank made fun of that. I'm talking about Frank Durda IV, in case folks are wondering. Frank called the TRSDOS 16 'bowling ball DOS.' That wasn't even his. [laughter] Well, no, that wasn't even his. We had a business review that said the Model 16 was as useful as a bowling ball for your business right now, and that was where that came from, publication or whatever... The rest of us said, yeah, that fits. Yep. Bowling ball DOS."

[Ken] "I hadn't heard that.”

[John] XENIX changed all that which was wonderful. About the time where we started working on XENIX, we were familiar enough with running Berkeley on the VAX that we kind of got spoiled by it. Berkeley added all kinds of neat stuff to V7, and we slowly brought in a few of those things that XENIX didn't have."

"I had to cold code the new MMU bits into the kernel without ever trying them on hardware... and holy crap it worked." - John Elliott IV

"So there's a few Berkeley commands that showed up in the Tandy version of XENIX that weren't in the straight version of XENIX, if you will. Stuff like the ‘w’ command. I don't remember what all. We picked some things like, 'Man, I miss having this.' So we ported them, because with Berkeley you get full source, and we have UNIX licensing through Microsoft for XENIX, so there wasn't any issues there as far as I know."

"But yeah, we kind of gave XENIX little tiny upgrades like that where it was noticeably missed like, 'Hey man, I want to do this that I do on the VAX all the time.' So that was kind of how some of those little Berkeley upgrades ended up being in XENIX."

[Ken] "They were pretty cool to have, too…”

[John] “Oh yeah… I guess you could say it's just there were quality of life type additions. And anybody that knew UNIX would appreciate them being there. I don't know that the guys that ran accounts payable or receivable on their XENIX knew or cared about some of those little things, but anybody that knew or learned UNIX, they definitely appreciated them being there. Or, used them and didn't notice that they weren't supposed to be there." [laughs]

[Sam] “There’s probably plenty of that...”

[Ken] "So Sam, do you have any behind the scenes cool stuff about how you guys came up with the design for the boards?"

[Sam] "Well, we were pretty free to do kind of whatever we thought was right. We didn't get a lot of input from or push around. The management was kind of weak, I guess, in that respect. Bill

Wilson was from TI, and I don't think he had much design experience, if any. So, I'm not sure he had a lot to contribute, one way or the other. He would listen to what we said we're going to do as a manager, would follow up and stay on you to get it done. But not a lot of 'do this, do that.' You know, 'this is wrong' and stuff like that."

[Ken] "Did you guys get any help from Motorola?"

[Sam] "We got help from vendors, but I'm just saying our management was not very helpful, technically, let's say. We were the experts, and they didn't challenge us on that."

[John] "Did they just basically point you in a direction and say 'these are the goals—go make something happen?"

[Sam] "...If that much…" [Laughter]

[John] "I'm curious about the decision to include an MMU on the CPU board. How did you know that it would need that?"

[Sam] "That more than… well, if you think about it, it's kind of obvious, but probably the germ of it came from Robertson—what's his first name?"

[John] "George!!"

[Sam] "George Robertson. Yeah, George."

[John] "I was wondering if it was him. Okay, because he would have known."

[Sam] "Yeah, he certainly knew. Yeah. He had experience with the Tandem computers."

"Right. The ones in their Data Processing… Semi-mainframes, ran 24 hours a day… 100% backup.

[John] "...the guys that would talk to the stores and get daily receipts, and..."

[Ken] "Yeah, that's Carol Lou."

[John] "Oh… right. Yeah. Okay. Well, I was just curious because…"

[Sam] "He worked for Carol, and he was pretty much in charge of the Tandem.

[John] "It's a key component in order to make any kind of multi-user OS, including UNIX, be protected from user malfeasance."

"It's certainly possible to port things like UNIX and XENIX to stuff that doesn't have protection… I particularly remember a port of SCO UNIX to a 186. Now that has no MMU and so it just wasn't protected. As a user you could write a program that would look all through kernel memory and do whatever."

"I think it was close to 50,000, if I remember correctly." - Sam Sawyer on the number of Tandy Unix systems deployed.

[Sam] "Oh yeah, yeah, and you could with the 68000 as well, I believe."

[John] “No, the 68K, that was the MMU's job. So when the processor switches to user mode, what happened was the address range of zero, actually was the start of whatever memory had been allocated to that process. That was how the kernel was, and the other processes, were protected from each other."

"The kernel could see anywhere because it's in supervisor mode. When you switch to user mode, the MMU limited the amount of address that you could look at in the RAM to whatever the MMU was set for.”

[Sam] “Right, but it needs some external stuff to do that.”

[John] “Oh yeah, the 68K definitely, that's why I was kind of impressed that you guys knew enough to make sure that it had a basic MMU on the CPU board."

"Later versions of the 68K had it built in, but not the 68K.”

[Sam] “Yeah. I actually didn't know that the later versions had that, but…”

[John] “It wasn't until I want to say the 68040 that it was all built in. 68020 didn't have it. It just had more bigger stuff. Well… You know what? Actually it was the ‘30. The 68030 had a built-in MMU, and the ‘40, I believe, had both MMU and FPU. Man, it's been a while since I've thought about this stuff. I might have to go double-check on that. But anyway, yeah, the 68K, the ‘10, and the ‘20, none of those had internal MMUs."

[Sam] "Yeah, I never got that far with the 68000 family.”

[Ken] “It was a shame that the 16 was the only, or the 6000, that series, was the only thing that we did with the 68000."

[Sam] "Yeah, like everybody else went down the Intel path…”

[John] “Their PC, right? So, you know, everybody was copying them and Tandy followed suit like everybody else did. It was kind of a shame. I remember being so excited when you guys up at TSD got ahold of a 68-10 [68010], and so I went to look at XENIX, what we needed to do to change low level to make XENIX run on it. It was just, I was so happy... [laughs]”

[Sam] “Yeah, well, and that was it."

[John] "Whichever one of you that was talking about happened to look for things like parts installed backwards made me think of the infamous event—I got my very first 6810 [68010], I was gonna try it out in a CPU board and stupidly, I plugged it in backwards and fried it [laughs]…”

[Ken] “Oops!”

[John] “...and oh I was crushed. I had to go back to you guys and go, 'um, yeah, I…'" [laughter]

[Ken] “We've all done that…]

[John] “It might've been Jerry [Ballard], I don't remember, but whoever it was was very kind and said, 'Well, okay, don't do it again,' and gave me another…”

[Ken] “Patted you on the head…”

[John] “[laughs] Oh, I was crushed. I was almost in tears. I just couldn't believe it. I mean, you

know, I had the precious and I destroyed it."

[Sam] “It Happens...”

[John] "Needless to say that never happened again, but I was just too excited trying to get it plugged in and start playing with it.”

[Sam] “[laughs] It's a wonder you didn't drop it and step on it…”

[John] “I wouldn't have done that. Although, back then, I got to admit, I was kind of lax on my static protocol."

[Sam] "That was not meant to be personal either...”

[John] “Oh, no, I could be a klutz sometimes. It wasn't at all inaccurate. [laughs]”

[Sam] “We all can do that if we're excited.”

[Ken] “Yeah we’ve all done it.”

[John] “I'm sure everybody's got a story of, 'I can't believe I *whatever*...'"

[Ken] "Well, that's... You guys had a lot more fun with it than I did. No, it was pretty fun down there in service, too.”

[John] “oh yeah, you guys used the hell out of XENIX... We were pretty impressed with some of the stuff you guys came up with.”

[Ken] “We had two mux boxes running. [Multiplexor boxes had a backplane with more slots, so it could be populated with more cards—more expansion—than the standard RS XENIX systems]. Twelve people in one and twelve on another. And then there's [the XENIX system] techsup the shops would call into—techcomm..."

[John] "Yeah, it was really fun getting our little work USENET running, all the machines talking to each other.”

"It was punch out at 5 and go home at 11. There was so much stuff to play with, and the customers were great." - Ken Brookner on working at Radio Shack.

[Ken] “...ihnp4. Uucp was cool.”

[John] “Really was...”

[Sam] “Erick Schmidt and I delivered the first Muxbox to a company in Ohio."

[John] "Yeah, it had those little modem boards, the little serial—it was one each, right? You had one four-port serial board and the other was a four-port modem that they talked to. Those in the MUX box were pretty cool."

[Sam] "Yeah, they were definitely. I built, well, I say I bought all the parts, designed most of the hardware, and built the first five.”

[Ken] “You could get a hernia from that power supply. [laughs].”

[John] “It was linear, right? Yea, That thing was a monster”

[Sam] “I chose it for linear on purpose."

[Ken] "I have one.”

[John] “Oh, that's right. That's your lab machine, isn’t it.”

[Ken] “No, I don't use it. I was gonna use it, I just never did. I was gonna clean it up and that's why I did the video snoop thing because I wanted to do a console over secure shell."

[John] “Oh, Right!”

[Ken] "Somehow or another, I just didn't do it—I didn’t finish.”

[John] “Yes, so it goes with many of our hobby projects, right? This wasn't as cool as I thought it would be or I'm kind of done with it now. I think we all have projects like that."

[Ken] "I still plan on doing something with it. I don't have any of the original modem boards or any of that stuff, it's just, we used them for the backplane and power supply.”

[Sam] “It's pretty general purpose.”

[Ken] “We filled them up with cards [14 position backplane], and I still have all the cards that came out of this one. This was Qat. This is the one that ran the tech rep side."

"I was able to take it with me. Yeah, I didn't take the hard drives, which I should have, but I didn't. David Krebbs gave it to me—he said we're scrapping it, it's surplus, why don't you just take it home. So I did. [laughter].

[Sam] “Of course you'd spend a fortune shipping it somewhere. But…”

[Ken] “I didn't have to do that. But it's moved all over the country with me."

[John] "Let's talk about the MMU extender. The Tandy one.”

[Ken] “That was pretty cool, too…”

[Sam] “Yeah, it was…”

[John] “I didn't know how much you guys had to do with it. We came up with the basic 'we want this' because XENIX was limited to using the first meg of memory because the CPU board's MMU was only 8-bit. We did the math and figured out if you added three more bits you could go out to the full addressable memory on the Tandy 68k CPU board."

"We thought this would be cool to make. We should make one of these, and then customers that have added extra memory to their systems will be able to use them under XENIX. At that point, Tandy had already started shifting towards IBM, Intel-type boxes. First they're like 'Yeah we don't we're not spending money on that,' and we felt strongly about it enough. When I say we, Frank and I were involved and I think maybe Jerry helped us do the initial design."

[Sam] "I think it was Jerry's hardware design.”

[Ken] “Jerry Ballard”

[John] “Right, Jerry Ballard just came to me and, 'Okay, where should the bits be and what does that look like?' And so I sketched something out. We had the upper and lower segments of the MMU and I just said, 'Okay, put three bits here and three bits here at this address,' and just sort of wrote it out that way, and he implemented it—he designed it. He knew what was needed. He probably had maybe done the board layout."

"Microsoft [...] basically ported XENIX in like three days." - John Elliott IV

"But of course the folks in merchandising said, 'No way, man, that's gonna make people want more Model 16s and 6000s, we don't wanna do that, because that'll take away from Intel PC compatible sales.' I guess it makes sense if you put your marketing hat on, but we didn’t agree with that answer. [laughs] So we went back and said, 'Okay, we would like permission to build this ourselves and sell it.' And they were like 'Whoa, okay, if they're that serious, no. No way man—we’ll make it' [laughs] Then they made about 200 of them and marked the price up just insanely ridiculously."

"Jerry and Frank, I think, had done the full cost analysis—cost this much to make the board, this much for parts. They had it all lined up.”

[Ken] “I think I have all that—I have that memo…”

[John] “It was ready to go have boards made, but that whole process of us saying, 'we want to do this' and them saying, “no,” took so long that I had been working on the final wide release of XENIX, which was XENIX 3.2, and the release date came up before all this back and forth and will-they—won’t-they was done."

"I had to cold code the new MMU bits into the kernel without ever trying them on hardware.

[Sam] “No… [laughs]”

[John] “I just crossed my fingers and we shipped 3.2 and holy crap it worked. [laughs] We didn't get the hardware until XENIX was out the door and it was the final release, right? So, it wasn’t like we could do patch releases or anything for folks. I'm just very grateful that I took enough care with the 68k assembler code that I got it right. I’m still somewhat amazed..."

"The reason the later aftermarket MMUs, stuff like the XMMU or our own MMU-3 could work is because I was programming all three bits always, because, again, we didn’t have hardware. I had no idea that Jerry had dialed it back to two when he made the board. It made things a little simpler, I guess…”

[Sam] “Yeah…”

[John] “The other thing was, the maximum recommended number of memory cards was three, from Tandy. So he must've figured nobody's going to have four meg—four boards, so the extra two bits did just kicked us up to four megs—so it was like that's more than plenty, right?"

[Sam] “No one need that [4 meg]… [laughs]”

[John] “I mean you couldn’t install more than that… At least not at *that* time…”

[Sam] “Right, right...

[John] “I’m kinda grateful that I went ahead and programmed the extra bit in there cause it enabled later things—people wanting to tinker with their systems. Amazing that went out totally untested… And it worked! Still a little bit shocked.

[Sam] "That third bit blows up the number of parts required to implement that feature in hardware, at least it did back then.”

[John] “Well, I mean, we have GALs now.”

[Sam] “Yeah, it made a difference."

[Ken] “What year was that? The last release was ‘88? For 3.2?”

[John] "Something like that. We had put together a 3.3, but it never got released. It just kind of had the back-channel kind of thing after Tandy was like, 'Okay, we're totally done. We're not doing any more. Start working on you 1000’s, or what the heck.'”

“But yeah, that was sort of a backdoor, thrown off the truck release.” [laughter]

[Sam] “Did they make more than the first 200, or so? Do you know? Of the MMU?”

{IV] “They only ever made the one batch of about 200 and called it good. To be honest, there wasn't a whole lot of customers clamoring for it, but there was some. So they definitely sold some"

"And we had several [MMUs] floating around system software—test machines and some of our production work machines. I don't know if y'all remember we had 16s and 6000s named after the Seven Dwarves.”

[Ken] “Right, I remember that…”

[John] “...and some bastardized ones, like... I think Gordon named one ‘sneaky’.”

[Ken] “That’s Gordon Burdett…”

[John] “Yeah. I remember mine was named Doc. The seven dwarves—they had a few of the MMUs"

"I was just grateful when Tandy was done and started getting rid of all the Model 16s and Model IIs. We were able to go down to Walt’s and pick them up for whatever they were selling them for. That was nice because I definitely back then could not afford Tandy retail prices back then."

"What was the actual name of that place, Sam?"

[Ken] "Walt’s”

[John] “It was sort of Tandy's surplus place.”

[Ken] “Yeah, Tandy Outlet Center or something like that. My department surplussed a lot of stuff through Walt’s”

[John] “Something like that... We just called it Walt’s because that was the manager. Like, ‘we’re going to Walt’s—let’s go…’ I got my Bernoulli Drive over there and definitely got my 16B at Walt’s."

"We'd just go through there and start combing through stuff, looking for whatever we needed, because we knew enough about the hardware to know what we were looking at. Walt's was fun."

[John] "I was really surprised by some of the places the Model 16 ended up. I vividly remember getting flown out on the Tandy corporate jet to an Army base that was having problems with their 16. [laughs]

[Ken] "The Army had a bunch of those—they had problems with dirt and sand and stuff. The Navy had some too."

[John] "Back then, and as far as the early 90s, Tandy had the largest number of UNIX systems on the planet. It's pretty amazing what we had for a while and then kind of let slip through our fingers."

"But then we got changed over to Tandy Electronics to train up technicians and troubleshoot really hard problems and basically make a nuisance of ourselves to Kenji, which we were good at—that was probably our best talent." - Sam Sawyer on his and Mike Berger's early days.

[Sam] "I think it was close to 50,000, if I remember correctly."

[Ken] "I never saw any of those numbers… I'm not surprised..."

"I pulled the UUCP maps, the ones I could find from the mid '80s, and I haven't gotten to it yet, but I want to go through them to see if I can figure what percentage of XENIX, our XENIX boxes are in there compared to other UNIX systems. People generally put the type of system they were running…"

[John] "...what the USENET node was…"

[Ken] "...so I should be able to pick that out and count them up. And then count others up too and see what the predominant machines were. I have a feeling that we're probably going to be somewhere towards the top of that. We'll see."

[Ken] "Anything else before we close up?"

[Sam] "I gotta leave quick—pretty soon anyway."

[Ken] "OK.. This was fun… Yeah, I enjoyed this."

[John] "Thanks for getting this organized, Ken."

[Sam] "Yeah."

[Ken] "Well, I will type it up somehow and post it, and you guys can make any corrections you want to, and we'll go from there. But I had a great time and learned some stuff, so it was fun."

[John] "Are you going to edit and post the audio or just the transcript?"

[Ken] "Just the transcript. I mean, I guess we could do the audio. But if it's good enough, well,

we can decide if we need to do an edit pack on it."

[Sam] “See how it sounds…”

[John] “Yeah...”

[Sam] "Anyway, enjoyed it. Let's do it again.”

[Ken] “Let’s do it again. I'll see you guys later.”

[John] “Have a good one everybody."

[Everyone] "You too, bye bye."

John, Sam, and I have been busy. If you don’t know our work and are curious, you can find some information at http://www.tandy-emeritus.org/

THE 'COMPETITION-COMPETITION'



WOULD YOU LIKE TO WIN THIS RARE AND LIMITED EDITION, U.K. ROYAL MAIL, INFORMATION TECHNOLOGY, FIRST DAY COVER, FROM SEPTEMBER 1982? ALL YOU NEED TO DO IS TO SUGGEST THE THEME FOR THE 2026 COMPETITION.

JUST SEND IN YOUR ENTRIES TO THE 'TRASHTALKERS'

George@48k.ca

AND THE PERSON WHO COMES UP WITH THE BEST IDEA WILL RECEIVE, VIA SNAIL-MAIL, THE ABOVE PRIZE!


BRITAIN'S BEST-SELLING DIGITAL CULTURE MAGAZINE
PIXEL ADDICT
£6.50 ■ 2024 ■ ISSUE 19
Tandy
The Story of an Electronics Giant
PLUS
NEIL WEST OF SEGA POWER
40 YEARS OF KING'S QUEST
MULTITECH'S APPLE II CLONE
20 GOTO 10 BOOK REVIEW
20 GOTO 10

IN MAV'S WORKSHOP

A Model III Video Board

Repair anyone can Accomplish



by Ian Mavric

This is a short article that might help a large number of people who have a Model III or early Model 4 (which has the Model III video

board in it) which otherwise works but displays the streaky text shown in the picture above. This is how to fix it.

BACK IN 2004 I STARTED A NEW JOB...

My new job was not TRS-80-related however I did pick up a cheap Model 4 on eBay and had it delivered to work. Upon powering it up I could see the text however it was presented as streaky lines. I call it the streaky lines fault. So the next day I went in to work early, and figuring it was the CRT at fault, took in a known good CRT picture tube, and proceeded to change it, only to find the streaky lines persisted.

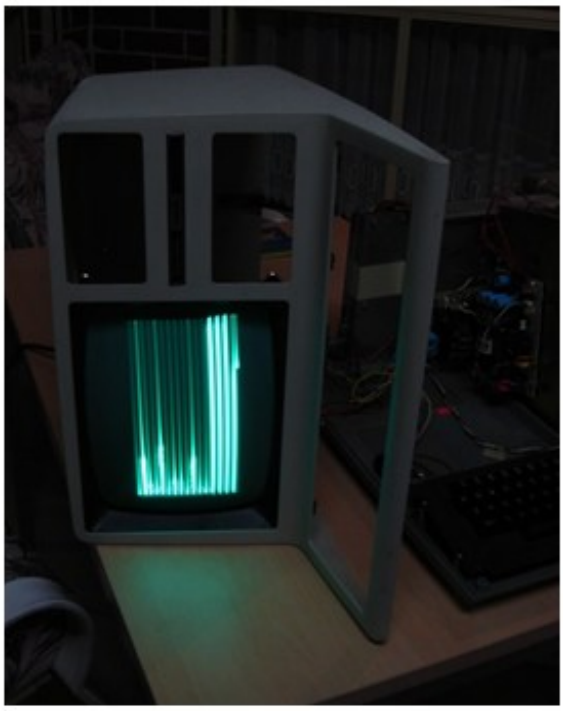


Photo: my bargain Model 4 with its streaky lines fault

I should point out that I'm more of a digital electronics guy not an analogue or TV tech so messing around with the video board, at the time, was very foreign to me. It's also easy to get zapped by the video board so bear that in mind if you plan to mess around with it. It holds a charge, as does the CRT, but if you leave it alone for half and hour after switching it off the dangerous high voltages have, for the most part, bled off and the board won't zap you.

Thinking I might have some scrambled output from the motherboard on the third day I came in with a Model III motherboard and again the streaky lines didn't go away. Drats, we have deduced the problem lies with the CRT video board. On the fourth day I brought in another video board that I knew was good and the Model 4 was fixed... all on work time ;-) I marked the bad CRT board with a tag which read, appropriately enough 'Faulty - Streaky Lines' and put it in the closet of all other faulty parts I would get to one day (I never throw anything away).

Keen spotters on eBay over the last 25 years will occasionally, and I mean really occasionally, like once every 2 or 3 years, see a Model III or early Model 4 advertised for sale as is for parts or repair, with the same streaky lines on the screen. Finally, in 2025, I have an answer.

FACEBOOK TO THE RESCUE

Massimo Pessino was recently restoring a Model 4, which are very hard to find in the EU. Looking through his pictures I notice his Model 4 had the streaky lines fault and he showed before and after pictures that he had fixed the problem. Being the gentleman he is I asked if he would tell me what went wrong and how to fix it. Simply an inductor on the video out line had failed and gone open, L301, and replacing it fixed the problem. I dug out my video board from 2004 and removed L301, it had indeed failed open. The Model III service manual states this is a 39uH inductor, so I ordered one online. While waiting I did a little research of my own on inductors and found they were just a coil of wire. Once the wire breaks the device no longer does its job properly, or at all.

I deduced that as a test you could just put a jumper wire in place of the removed inductor, and that should put the display right. I soldered in a wire and it worked. Once the correct 39uH inductor arrived I put it in but it's handy to know for testing purposes you can replace the inductor with a wire and see a marked improvement in the display



WHICH VIDEO BOARD DOES THIS AFFECT?

Notice I have been saying Model III CRT board but also mentioning Model 4s? The CRT board I am talking about is a cream-and-green board as fitted to the Model III, and it is completely analogue in operation.

Now RS, probably at the time the Model 4 was being put into production, must have found they had a large quantity of these analogue CRT boards, and those found their way into 16K diskless Model 4s (cat. no. 26-1067), and occasionally Model 4 disk systems. My guess is that the newer 'digital' CRT boards might run out at the TRS-80 production facility so the early board fitted to keep churning out Model 4s.

Also for field repairs it's possible that to RS the digital and analogue boards are "equivalent" to each other so substitute in a Model III CRT board when a Model 4 board fails and there are none on hand to replace it with. Indeed I have stated previously that the Model III analogue CRT board's picture quality is as good as a Model 4 digital CRT board if both are in excellent condition.

TIP: if you have a Model 4 with one of these CRT boards in it and can't find info in the Mod 4 Tech Ref manual (because it's not in there) pull out a Model III Tech Ref manual, it's in there :-)

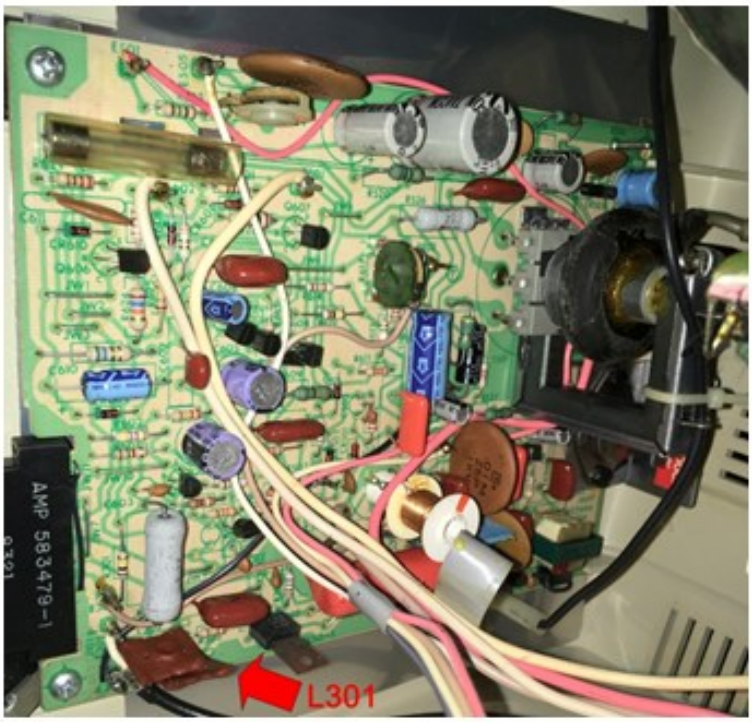


Photo: Model III CRT board with L301 indicated with a big red arrow

CONCLUSION

CRT board problems on a TRS-80 are among the most annoying part of computer restoration. Most of us who can get our heads around digital electronics and the low-voltage (5-12V) find TV type electronics anxiety-inducing. Not just due to the electronic concepts in use but the chance of being zapped by high voltage. At least with this article this cheap and easily replaced part may provide a simple repair to an otherwise good CRT board, to keep another TRS-80 Model III or 4 alive and kicking.

Ian Mavric

ianm@trs-80.com



Ian Mavric is an IT Specialist who also restores and collects TRS-80's and classic cars. He lives with his wife and kids in Melbourne, Australia.


ACTUAL REALITY
TRS-80 MODEL I/III/IV
DREAMLAND SOFTWARE


ROCK STAR!
TRS-80 MODEL I/III/IV
DREAMLAND SOFTWARE

POWER SUPPLIES

FOR THE TRS-80 MODEL 3 / MODEL 4/4D/4P



* Uses modern day MEAN WELL power supply units
* 50w for Model 3 version and 65w for the Model 4 version
* The Model 4 version uses a SEPARATE +12V POWER SUPPLY (prevents video "spasm")
* No more RIFA CAP explosions
* No more COLD solder joints
* Units are DIRECT REPLACEMENTS for the original power supplies in all models
* Extra +5V output connectors (1 on Model 3 and 2 on Model 4) for Gotek/freHD boards
* For more information and pricing contact

newSOFT

https://jaynewirth.wixsite.com/newsoft (website)
jaynewirth@verizon.net (email)



Servicing Your Tandy/Radio Shack System?
A Case for Containment…

ken brookner January 2025

Containment? Yeah, contain your enthusiasm—just a bit. How come? You have this new machine and you're itching to fix it. There are two things that can happen when you pull the covers and dig into that new machine unprepared. First, and most probable, is that you're going to make it worse. Second, you might end up taking that ambulance ride you always wanted.

Do I have your attention now?

Today, computers are mostly modularized and the highest voltage you might see in one is probably 12V. These systems are made for users to be into, swapping boards, upgrading power supplies, adding memory.. If you're wearing that flannel shirt, standing on that pile carpet, and drawing sparks—and you blow that new board. No problem, you can order another.

Do that with your 1980s Radio Shack machine—well, now you have a problem. Boards are getting scarce, and parts are too. We used a lot of the venerable 8T26A devices as bus transceivers in the Big Tandys (Models II/12/16/16B/6000) and they haven't been made since before 1990. They weren't all that reliable when new and now… even less so. You pop one of those babies and you can't just get another from your favorite parts house.

The Big Tandys, and the smaller Z80 based machines like the MI/III/IV, weren't intended to be user serviceable. Almost none of this stuff was user serviceable then and attention to customer safety inside the box was not even a consideration. The Model I is not really a problem, but the MIII/IV have AC Voltages exposed on the ON/OFF switches. Get across those and it'll hurt. They also have CRTs with voltages in the 10 kilovolt range—it's low current, but at a minimum it'll make you hurt yourself as

you flay your arm on that sharp sheet metal as you jerk it out of the machine, and at the maximum it can be fatal.

Similarly, the Big Tandys have a lot of exposed AC wiring, terminal blocks you can get your fingers on and the high voltage for the CRTs. The 12/16B/6000 are really the worst of this bunch because of where the AC ON/OFF switch is, it isn't covered, and the AC lines are right there for you to drag an elbow across.. Today you turn your PC on with a switch, and that switch is tied to a GPIO and indicates—hey! Turn me on! Older machines, and not just Radio Shack's did it by switching the live wall current.

There is a way to work on these venerable machines safely!

Let me ask you this: You're about to take that family vacation and drive the significant other and kids across the [*insert large geographic region here*] to see some new things. Are you going to take a map, or at least *something* with a GPS in it, or are you just going to pitch the luggage into the car, grab everyone and head out for the great unknown? Kick the tires and light the fires?

When you tear into your machine without doing any preparation, you're just like that family heading out unprepared—you'll probably end up lost, maybe wrecked somewhere.

If you're new to this, it's easy to think old equals simple, but old just equals old complexities you're not familiar with.

Fortunately, there is a ton of material available to help you service your machine—a lot of the service documentation released by Technical Support at Radio Shack, as well as Radio Shack Technical Publications is available today. All you have to do is download and become familiar with it before you reach for that screwdriver.

Publications

- *Model II Technical Reference Manual*
- *Model 16 Service Manual*
- *Model 16B Service Manual*
- *Model 6000 Service manual*
- *8 Megabyte Hard Drive Service Manual*
- *12/15 Meg Hard Drive Service Manual*
- and more...

These service manuals tell you how to take the machines apart, give you a theory of operation, pictures of all of the cards, the card order in the cage (and why card order is important), parts lists, alignments, and schematics current to the date of publication. Best of all they are free and available for immediate download.

- *Technical Bulletins*
- *Notes and Jumpers*
- *XENIX Information Series* (Version 7 and System III)

These were produced by Technical Support at Radio Shack. The *Bulletins* cover all of the fixes, running changes, and important service information you need to know before you grab that soldering

iron. They cover ALL of the machines and peripherals Tandy/Radio Shack produced in the computer line—start to finish (late 1970s into 1994). There is some crossover, so even if you just have the Model II, you should look at the Bulletins for the 12/16/16B/6000, too. Most cards are compatible between the machines, and today it's common to see the cards mixed.

Want to know what those components tacked onto the front of the board, or those wires on the back do? Think *Technical Bulletins*.

Notes and Jumpers is in two large volumes, one contains all of the jumper information and various notes on almost every board you may find in any of the systems—again start to finish. Volume II contains the graphic images that show the boards and where the jumpers are.

The *XENIX Information* series was written for our service technicians. It's a lighter look at XENIX care and feeding than the customer manuals, and more to the point. It contains the meat of what a technician needs to know when servicing a XENIX business system, from initialization, to backup, mounting hard drives… Check it out if you're playing with XENIX.

8" Diagnostics

8 Inch is how we distinguished the larger business systems from the smaller home systems—8 inch referring to the floppy drive size. This diskette contains diagnostics to test just about everything in those systems. *MEMII* is the authoritative RS memory diagnostic for these systems, and there are a host of other diagnostic programs on this disk. Be sure to get the manual when you get the software—you don't have to guess what it all means.

Schematics

The service manuals have schematics, but we were constantly improving the systems, so the schematics in the manual may not match what you have in your hands. Most of the schematics for all of the different revision levels of individual boards are available. If you're tracing bits with a scope, you should have the correct revision schematic—or you're going to get lost.

The Video Generator Card (VDG) in the Big Tandys are a case in point. One revision of this board changed all of the device designations. It was common when revving up a board to move devices, but they generally kept their designation (like U12, or U36). Not this VDG card. It looks pretty much the same but if you start troubleshooting it with an older schematic—you won't get very far. FYI.

You can find all of this and more at PSKI's Model II Archive and Ira Goldberg's TRS-80 Revived Site:

https://github.com/pski/model2archive
https://www.trs-80.com/wordpress/radio-shack-lore/

Don't forget the Internet Archive, too. If you don't find it at Ira's or PSKI's site, search for it in the archive. I've found a few gems there.

https://archive.com

Damage from Static

Final bit of advice here: Damage from static is real, and it's cumulative. Zap it once and maybe it keeps working—but whether you realize it or not, that card is degraded. Zap it again, maybe it degrades a little more—but you'll never know. Zap it another time and maybe that time it pops. You don't have to draw sparks to cause this damage—it's very subtle.

As you hold that board in your hands, consider how many other hands have held it in the last 4 decades and how they may have handled it.

How degraded is that board and how close is it to failing? You want to find out? Then, don't use anything like anti-static straps and mats. It's a very good idea to keep any boards not installed in an antistatic bag. These are inexpensive and can even be found at Amazon.

These are the ones I use for Big Tandy card cage boards:

Anti Static Bags, 25pcs Large Resealable ESD Bags 7.87x11.8in with Labels for GPU Hard Drive SSD HDD and Electronic Device.

Keep any static sensitive devices such as TTL 74 series, in anti static foam:

Anti Static Foam, High Density, Non-Corrosive, 305x305x6mm (2-Pieces).

Work on an anti static mat and use a wristband—all available at Amazon. None of this has to be expensive. If you ship someone a board, drop it into a static bag. I've received boards from Ebay vendors and friends wrapped in clear bubblewrap—this is the absolute worst packing material practically guaranteeing damage.

That's It

I've mostly discussed the larger business systems, but similar manuals, info, diagnostics, *Technical Bulletins*, and schematics are available for the other Z80 based computers, too, also at the Model II Archive.

---

80 - NET

ELECTRONIC NETWORK

* *FOUNDED NOVEMBER 1980* *

Chairman:- **LEON HELLER**
Secretary:- **BRIAN PAIN**

ADMINISTRATIVE OFFICES:-
40A HIGH STREET,
STONY STRATFORD,
MILTON KEYNES.

Tel: Milton Keynes 0908 - 566660

* THE NOTICEBOARD FOR ALL MICRO COMPUTER ENTHUSIASTS *

LOOKING FOR FAST, INEXPENSIVE, UNLIMITED MASS STORAGE FOR YOUR TRS-80 MODEL I/III/4/4P/4D?

The amazing

"FreHD"

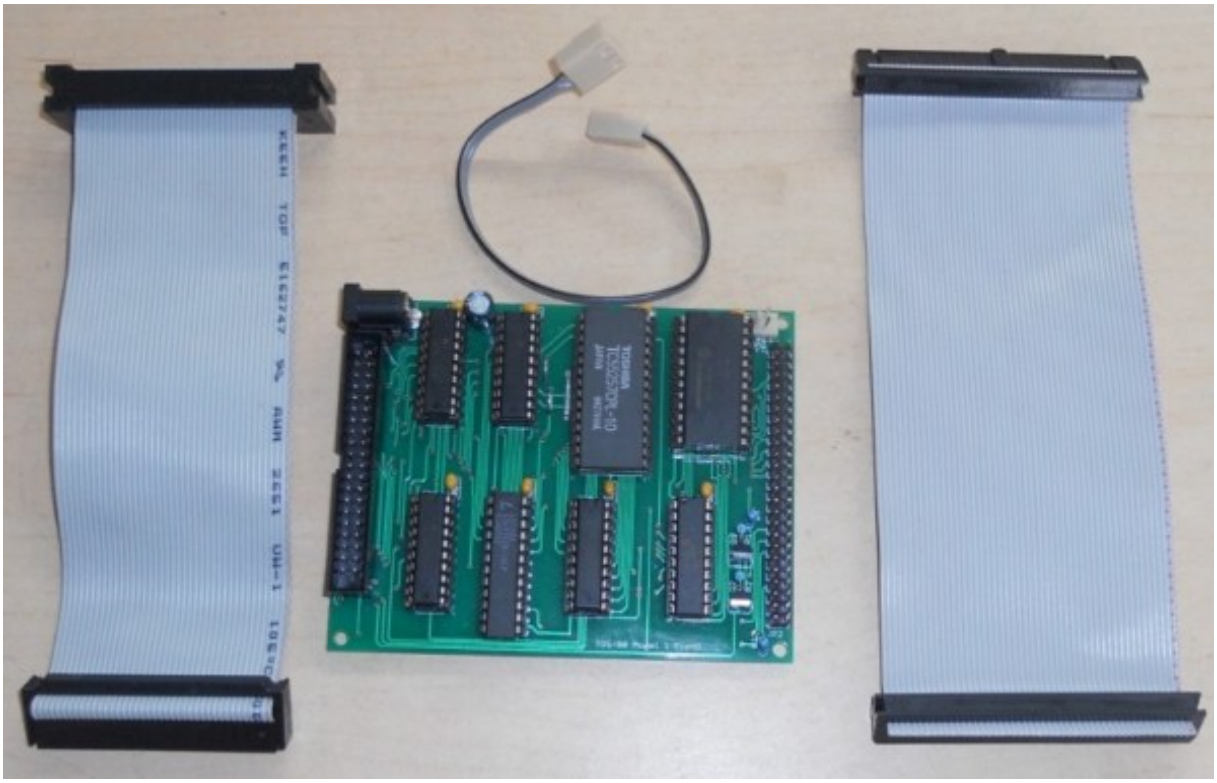


-Emulates a TRS-80 hard drive, but faster than any hard drive!
-Works with your favourite DOS (LS-DOS, LDOS, CP/M, Newdos/80 2.5)
-Uses SD card for storage medium
-Bonus free Real Time Clock function!
-Designed in Belgium and proudly built and shipped from Australia
-Kit form or fully assembled

Order yours today
http://trs-80.com.au/trs80/emulator.htm

“Quinnterface” Mini Expansion Interface for 16K Model 1 FreHD users.

- COMPLETLY ASSEMBLED AND TESTED -



This is THE perfect device for all Model 1 users who own a 16K Level II unit, (which is most of us!), but no Expansion Interface or disk drives, especially if you don't want to modify your M1 with upgraded boot ROM or memory upgrade.

The ‘Quinnterface’, developed by J. Andrew Quinn from New Zealand, adds 32K RAM and auto-boot functionality to you FreHD.

U.K. And Europe, contact Bas. at BetaGamma Computing.
U.S.A. And rest of the world, contact Mav. At “The Right Stuff”

EBAY.CO.UK 'BARGAINS'

HERE ARE JUST SOME OF THE MORE INTERESTING/UNUSUAL TANDY EBAY LISTINGS, SINCE OUR LAST ISSUE.



£102



£450
NS



£40



£282



£132



£25
NS



£90



£79



£195

EBAY.CO.UK 'BARGAINS'

HERE ARE JUST SOME OF THE MORE INTERESTING/UNUSUAL TANDY EBAY LISTINGS, SINCE OUR LAST ISSUE.
WERE YOU ONE OF LUCKY ONES?



£583!



£12



£80



£125



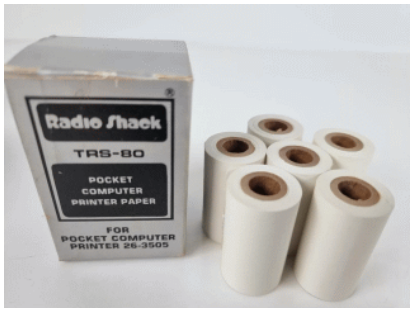
£35



£250



£190



£7



£14

BetaGamma Computing

Specialists in all things that compute





Based in Corfu Greece

- Qualified IT Engineers
- Repair of all PC's and Laptops
- Software and Hardware Upgrades and Installation
- Broadband and Network Installation
- Supply of Custom built to order PC solutions

RETRO RESTORATION SPECIALISTS



Call Corfu 26610 26358 Mobile 698 755 8427

Email: bgcompute@aol.com • www.betagammacomputing.com


THE WAY
WE WERE
1985
1976
1979
1983
1978
1980
1981
1977
1982
1984

NEW FOR THE TRS-80 AND VIDEO GENIE

FLOPPY DISC SYSTEMS EXPENSIVE?
CASSETTE SYSTEMS FRUSTRATING?

THEN YOU NEED THE

SOLO LOAD

Loads programs in less than one second. Saves programs in as little as two seconds.
NO MODIFICATIONS.
Just plug into expansion socket and run your programs instantly.

Completely automatic, contains its own operating system. Anybody can use it — it's that simple to operate. So banish those cassettes and forget about expensive Disc Systems.

LOOK AT THESE UNIQUE FEATURES

Directory of Files — No need to remember names! Load File — Automatic search and load of programs. Load Data — Load Data into any memory locations. Save File — Automatically produces a 'named' file with all information the computer needs for instant reload. Save Data — Effectively an automatic 128K bits PROM Programmer i.e. no need to sectionalize programs into PROM sizes. Multiple programs in single Rompack with overload prevention and auto verify. Does not use any user RAM space.

Perfect for engineers and programmers who want speed and reliability.

The **'SOLOLOAD'** is a new concept in Rompacks. It is unique in that it not only allows you to read pre-programmed Rompacks but create your own. The plug-in Rompacks are of course, **RE-USABLE.**

The function of the **SOLOLOAD** is to provide an alternative storage medium to that of tapes and discs which is very reliable and fast to operate.

It is based upon storing programs and data in semi-conductor memories which can be re-used whenever required but do not lose their information when un-plugged from the mains of host CPU. The memories used are EPROMS and CMOS RAMS.

N.B. Both basic and machine code programs can be saved and loaded.

USAGE OF EPROMS (16K ROMPACKS)

EPROMS provide permanent but re-usable program and data storage which can be loaded into the host CPU in less than one second. However they are not instantly alterable and would normally be used for long term storage, where maximum program security is required, it takes 50 milli-seconds per byte to store a program in EPROMS.

USAGE OF CMOS RAM (8K and 16K RAMPACKS)

These modules have an in built rechargeable battery and can retain data and programs for many months even if not on trickle charge. The advantage of CRAM modules is that they are instantly alterable and can save a 16K program in under two seconds. Their obvious use is for program development and rapidly changing data files.

TWO VERSIONS AVAILABLE For use with up to 16K systems. **EX-STOCK DELIVERY**

SOLOLOAD Read and write unit with printer interface option.
SOLOREAD A read only unit for pre-programmed Rompacks. Ideal for clubs.
Both units attractively boxed with own power supplies.

COMING SOON: Version for expanded systems. Version with Expansion Ram.

SOLO ELECTRONICS LTD

73 PRESTON ROAD, BRIGHTON, SUSSEX BN1 4QG (0273) 671873

The SOLOLOAD has its own internal operating system which is accessed by one simple keyboard command.

THE OPERATING SYSTEM GIVES ALL PROMPTS IN ENGLISH

The operating system has 8 general functions which sub-divide when required.

The options are basically as follows:

1. LIST:— Search through rompack and display the names of all files stored.
2. READ:— Load the host CPU with:
   (A) A named file (PROGRAM) in basic or machine code.
   (B) A prom or series of proms containing data i.e., Not a named file but say a mailing list or bank statement.
3. SAVE:— Save has several options:
   (A) The choice of EPROMS (for permanent storage) or CRAM for high speeds.
   (B) Saving a machine code or basic program.
   (C) Saving as a file or as data (effectively a 128K bits EPROM programmer in the save mode the SOLOLOAD also generates all the information the CPU needs for automatic re-load of the program.
4. BASIC:— A command which returns control to the most host CPU's basic monitor.
5. DATA:— This displays useful information to enable back-up copies of tape programs to be readily made in this format.
6. EXAMINE:— Since the SOLOLOAD is memory mapped but paged a special monitor has been provided to enable memory and rompacks to be examined.
7. FILL:— Enables sections of memory and CRAM to be filled with known data, again useful when making back-up copies of tapes.
8. JUMP:— Enables the operator to go to anywhere in memory e.g., Return to the editor assembler, etc.

INTRODUCTORY PRICE LIST

SOLOLOAD complete with free program in a ROMPACK ...	£125.00
SOLOLOAD printer interface option	£37.50
ROMPACKS (16K capability)	£10.00
8K CRAM RAMPACKS (with battery)	£62.50
16K CRAM RAMPACKS (complete)	£99.50
SOLO READ (read only unit)	£35.00
SOLO READ P.S.U.	£8.69
Spare EPROMs (2532's)	£4.50

Please add V.A.T. and P.&P. (£2.50/order).

Discounts for club and other bulk orders.

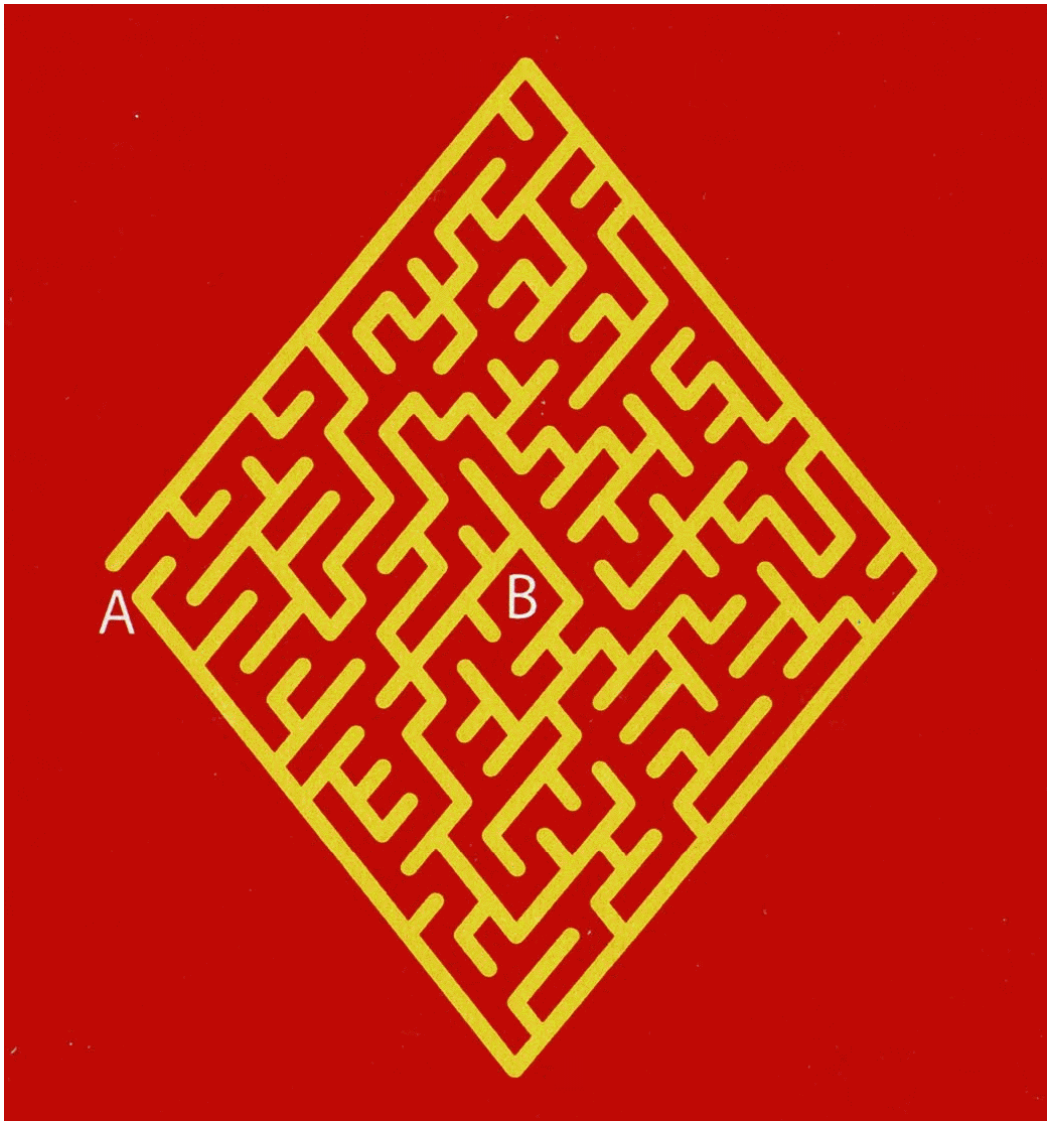
Full details available from:

SOLO ELECTRONICS LTD
73 PRESTON ROAD, BRIGHTON, SUSSEX BN1 4QG (0273) 671873



https://www.youtube.com/watch?v=ZE3nMxTLgjk

AMAZED



IF YOU HARDWARE-HACKERS ARE FINDING THOSE PCB BOARDS TRACKS ARE MERGING INTO ONE OR YOU ASSEMBLER CODERS HAVE LOST A PUSH OR A POP, GIVE YOURSELVES A BREAK.

SIT DOWN WITH A COFFEE, PUT A TIMER ON AND SEE HOW LONG IT TAKES TO GET FROM A TO B, FOLLOWING THE RED ROAD.
TRY FOR UNDER 10 SECONDS!



LEVEL 2.5 (TWOHAF) BASIC ALLOWS CASSETTE AND ACULAB FLOPPY TAPE USERS MANY OF THE TANDY DISK BASIC FEATURES, NAMELY -

"&", "DEF FN", "DEF USR", "MID$=", "INSTR", "MKD$", "MKI$", "MKS$", "CVD", "CVI", "CVS"

AVAILABLE, AS A **FREE DOWNLOAD**, FROM

WWW.TRS-80.ORG.UK

Surfin' The Net



ANOTHER CREATION FROM THE LAND OF TRS-8Ø. GOING IN A MORE POSITIVE DIRECTION AND A DIFFERENT TOUCH. GETTING A CLOSER IDEA OF WHAT THE TRS-SYNTH IS GOING TO SOUND LIKE WHEN IT IS FINALLY CREATED. I HOPE YOU ENJOY THIS, AS MUCH AS I DO! HOPING IT BRINGS SOME POSITIVE VIBES TO YOUR DAY.

https://www.youtube.com/watch?v=xeVfcLDC0UQ



John H has TBBS running on his Model I. To access, telnet (using Putty or similar telnet client; but not a browser) to `trs80bbs.tplinkdns.com`

For those interested, he is using WiModem232 Pro as his modem.

HELLO DUSTY.
THIS LINK CAUGHT MY EYE RECENTLY!

REGARDS - LEON

RadioShack Not Only Still Exists, It Sells Suspiciously Cheap SD Cards

JAN 27, 2025 JARON SCHNEIDER



RadioShack, once a ubiquitous American electronics store, filed for bankruptcy and shuttered all of its brick and mortar US locations way back in 2017. For most Americans, the brand appeared to die but that isn't entirely true: it shifted its efforts online where it now sells some suspiciously cheap SD cards emblazoned with the company's branding.

The electronics retailer has changed hands multiple times since its collapse in the 2010s which saw it delisted from the New York Stock Exchange in 2015 and filing for bankruptcy two years later. The company was acquired by Retail Ecommerce Ventures in 2020 and shifted to e-commerce. It was then acquired again by Unicomer Group, based in El Salvador, in 2023. Curiously, while RadioShack failed to hold on to a retail presence in North America, Unicomer maintains physical stores in Central America, South America, and the Caribbean.

That catches things up to 2025 and explains how the company is still operating and while it has expanded its offerings, many of the products that at one time could only be found at RadioShack are still listed: adapters, cables, connectors, and converters are all still front-and-center, which is sure to be a bit of a nostalgia hit for many.

RadioShack reached out to *PetaPixel* last week with a notice that it had a full line of "best-selling storage and memory products" that were available both at RadioShack.com and Amazon — two brands that 90s kids would never think would be mentioned in the same sentence.

https://petapixel.com/2025/01/27/radioshack-not-only-still-exists-it-sells-suspiciously-cheap-sd-cards/

Marcel Erz, through his [Retrostack github page] has put together a lot of modern replacements for TRS-80 Model I/III/4 parts, including motherboards, ROMs, character generators, and various plastic parts.

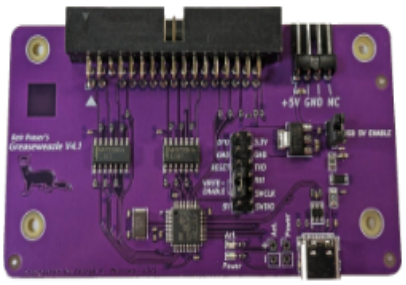
As of this posting, the site includes:

- Model I:
  - Motherboards: Revision A, D, E, G
  - ROMs: Broken out into 2×64, 2×128, 2×256, and 2×512
  - Keyboards: ALPS and MX
  - Power Supply with 3-D Printable Enclosure
  - Mods: XRX-III
  - Adapters: Keyboard, ROM, and Character Generator
  - 3-D Printable Parts
- Model III and 4:
  - Keyboard Adapter
  - 3-D Printable Parts

--== oØo ==--

Quentin Barnes has released an update to his Greaseweazle software, now at version 0.0.1.beta.1. Supported is Windows 32 and 64 bit varieties, and Linux Aarch64, Arm7L and x86/x64.

IMHO, he is being very conservative with his numbering – the program works perfectly.



Greaseweazel hardware

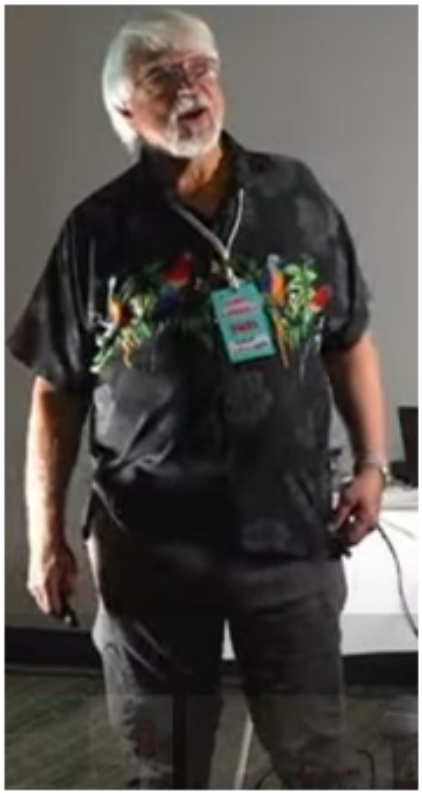
You can download it from Quentin's GITHUB at
https://github.com/qbarnes/gw2dmk/releases/tag/v0.0.1.beta.1

If you are dumping your own images, please consider sending them over to me for addition to the archive.

Randy Kindig interviews Don French and Steve Leininger at the 2024 Tandy Assembly

Randy Kindig: Hi, I'm here with Don French and Steve Leininger, who were the masterminds behind the TRS 80 Model 1. And actually, as I understand it, this is the first time they've been together since what, 1977? I think it was, Don. 77
Don French: yeah, 76, 77, 77





Randy Kindig: We're going to talk today with them a little bit more about why RadioShack got into the computer business to begin with, and how they met, and that whole interaction there.

Let's start out there. If you could, and talk about the history, and why RadioShack got into the computer business to start with.

Don French: Okay. Yeah, Don French, in case you don't know. I was a RadioShack employee starting at 16. I was the youngest salesman ever. I became a store manager at 18, the youngest store manager, and then I became the youngest executive ever at 23. And I was the product manager in the corporate office of parts and pieces, the semiconductors, resistors, all the good stuff. There were seven buyers, product managers, and the other 1, 200 items, the parts and 200 items, and the other 1, 200 that Radio Shack sold were the radios, antennas, CBs, speakers, all that good stuff. So one of the things that got me started in it is a magazine article for the Mark 8, 74. I saw that and I said, wow, this is interesting. And I called a gentleman that created it and talked to him and asked how many he sold. He sold 8, 000. And to me, that being a a product manager, that was a great deal because Oh, about six months, a

year earlier, I had created a new product in Radio Shack called Project Boards.

Project Boards were printed circuit boards designed to do something, but the key to Radio Shack was, you build it from every part that was in Radio Shack's inventory list. And that was the whole concept behind it, just build, help build the inventory. And when I saw that Mark 8, and I built it, and I was able to make it work, I was getting really interested.

And then the next year Altair came out, MITS came out with Altair and I bought one of those and I built that one up and actually used it in managing my 1200 items because we did what was called a one of a kind at least once a year and sometimes more. It was it took on a columnar pad.

Stock number, wholesale price, retail price, do them all, add them all, or take them up, add them up, and then you get what your one of a kind gross margin is. That was our standard to work from. And that was a P. I. A. to do. With 1, 200 items. Yeah. The 200 or 300 each. So it was nothing. And I was doing it that way, and then once I built the Altair, I had really built it up.

I've got an Icon disk drive, floppy disk drive, seven-inch floppies. I had a deck writer and an ADM three terminal. And I wrote a basic program to do my one of a kinds, keep my inventory basically. So I could just press a key and say, run the report and it would period out for me. And I taught my now ex how to do it.

And she could, if I need anything during the day, she knew how to look up an inventory item for me and that worked very well on The first time I turned in that report, it taught me a lot about acceptance of the computer. I turned it into I won't give her a real name, but she was nicknamed Dragon Lady.

Who was the re buyer of the computer. In other words, the person responsible for buying restock to keep what we put in as product marriage in the product line to put that out and re keep us in inventory. And I took it to her and handed it to her and she looked at it and said, what's this? And I said, it's my one of a kind.

She handed it back to me with a colander pad and said, do it right. In other words, hand copy it. I did. It was the worst writing of my life. positive. But it told me that getting anybody to accept computing was going to be a nightmare. And that was the first thing I experienced with it, but I didn't give up.

I started working on it and then in kept pushing and pushing, talking about it. Zero interest. Within Radio Shack? Within Radio Shack. Yeah. Nobody was interested in it. And then We, as product managers, we had to work five and a half days a week since the stores were open six. We worked Saturday mornings, also.

And we were in the this was in seventy six, because I had the alter. One Saturday, rumors started going around the office. Time isn't coming in on the Warpath. Oh, we dreaded those. He called all the product managers together in the conference room and he started it on the war path. Advent had just released a really high tech speaker system.

And he said, why can't RadioShack be innovative? And I listened because mine are parts and pieces. And one of the other product managers. Don's got a computer in his office. Gee, thanks. I did. I ended up with the Atium 3. I booped it in there. And so I could do it myself. Tanny just said, he didn't care about it.

And nobody did. Bernie Appel was my boss, Drek Boss, Vice President of Marketing. John Roach, Vice President of Manufacturing, no interest whatsoever, even though he was also head of IT, IT, the Computer Center. I keep forgetting that we didn't have IT back then. And no interest whatsoever. And there was a lot of interest in calculators watches, digital electronics, but there's nothing for a computer, no interest at all.

And I was trying to talk them into it. And so when the project board started working on doing it, because I knew, I controlled what was in that product line. I didn't have to get approval of Bernie Appel or anybody because they didn't care. Honestly, it was sales, it was working, and it was just running.

And but I am not, even though I'm a ham operator, amateur license, I know electronics, I can do stuff like that, no digital work. whatsoever in my life. But I was working with all the major vendors of semiconductors and processors and that AMD national were the primary ones I was working with and RadioShack didn't want it.

So I was free to talk about it. And I did, I talked to a gentleman named Chuck Fettel. You may know that name. Oh yeah. He was at AMD and he was. He was the creator of the Commodore and the PET. I had some long talk with him about it, what I felt the product was. I'm going to try to get him to build something for me that I could use, and it didn't happen that way.

But I described it to him exactly what it was, even to the extent of the keyboard I described was a microwave keyboard. Microwave keyboard. The touch. I can't get it. I can't make a mess of 'em because it's all plastic covered and they were cheap and it it didn't describe the boxy look of it. They didn't describe it at all because they didn't care.

But it also had the cassette player built into, 'cause that was the storage at that time. And that's what I described to him. And then he we talked about trying to getting, buying a MD chip and they didn't do it. And I talked to National, they were one of my primary suppliers and the National Semiconductor, sales rep, he was pushing it. He knew what I wanted to do. And he started getting the technical people at national to start making proposals to me about how to do it and do maybe to create a design of a project board using everything that we could. I had my line and their scamp initially chip and the ultimately looked at doing the pace also, which are like Microsoft excuse me.

National Semiconductor Processes, the two they had at the time. And still didn't get any interest from anybody that was working on it. Never got to turn in another a computerized one of a kind either. Still had to do them by paper next year too. It was crazy. And I was making trips out to California quite often because we bought a lot of surplus electronics.

And I planned a trip. around the computer fair. And so I got to go out there and there's one day in the 76, early, that was out there at the

show and Roach had come out to see, do a buying trip for the chips. I don't know if it was Fairchild and National, I don't know who else he was seeing. I didn't, I wasn't with him at all.

And I asked him, say, hey, you want to come to a computer show with me? And he looked at his schedule and said yeah, I'd love to. And it was McCormick Center in San Francisco. Huge complex. The line going in was around the complex totally. It took an hour standing in line to get in there, and I said, oh, this is going to destroy any chance. And we got in, and he was shocked about the number of people in there asking questions, looking at everything in there, the high interest level in computing.

Yeah. Yeah. And after he got back to Fort Worth and wrote a memo to Charles saying that there is something to this computing. He became interested in it and then I wasn't out in California on the next time it came up and he was out. National. And I knew that he was going to ask about it, because he asked my contacts there who was working with me.

And I didn't know what was going to happen. He asked about it at the end of their meeting about the watch and calculators. He asked, what do you know about the personal computers? And it was told, this is hearsay to me, because I wasn't there, but what he said is that they said, we have somebody that can talk to you about it intelligently.

Bring him in. And Steve came in.

Steve Leininger: Yeah, actually, they were going to have their we had a sales manager at National Semiconductor, Phil Roybal, and he was otherwise occupied that day I have no idea what he just got lucky that day. He wasn't there. Yeah. And they said Steve knows as much about the SCAMP microprocessors, and he, I couldn't talk pricing or anything.

Don French: It wasn't about pricing, it was about concepts.

Steve Leininger: Yeah. Yeah. And they had me come in and actually I was the right person for that because I had been going to the monthly Homebrew Computer Club meetings. The same place where Steve Jobs and Steve Wozniak also were going. And we were all just engineers then.

There aren't any superstars in the room. It was Lee Felsenstein who was actually the moderator for the Homebrew Computer Club. We had some star power there. The guys from mco Bob Marsh from processor BTech. Yeah. Processor technology, so it was the beginnings of things.

The guys who did I inside the ICI was like an Altair with a real power supply in it. Good description. Yeah. Yeah. And better switches. Yeah. And I was building a computer at home and I was also very familiar with the software for the. Low cost microprocessor that Radio Shack was selling. And I was doing a basic for that, it, it was.

Randy Kindig: Do you ever think what might have happened though if you hadn't actually been there that day?

Steve Leininger: And there was more than one part to that because I told them, some different things and all that and it was Jack Sellers and, john Roach.

Don French: And Jack Sellers was the head of development

Steve Leininger: They did the project kits. That group did the project kits for Don.

Don French: They didn't design it. They did anything that was assembled. They didn't do any of mine. They didn't even know what I was doing. I did this, had a board made. They made the board. Because we had our own board making plant. So they made the boards. But they were designed by somebody else. I did have a few of the engineers design a couple of them. Like the digital clock. Yeah. Designed by Radio Shack Tech, but they, that's all they did. They did not make anything other than they pressed the boards.

Steve Leininger: One of the things Mr. Roche asked about was are there any one of those newfangled computer stores in the area? And so happened that I was moonlighting at Byte Shop number two, the second of Paul Terrell's legendary computer stores.

Don French: He didn't tell them that.

Steve Leininger: And anyway but they asked where it was.

And I said yeah you go down here and go down here and you'll find it find the store. And not only did they, they went to dinner and then they went out on an excursion and they found me in the store. I was in there rolling up a paper tape when they came in, so it's like he's here, there, and everywhere.

And they looked around the store, checked things out. There wasn't anything that was like a traditional computer there yet, it was still the Altair kind of box. They had a couple different versions the original Apple the big cookie sheet. Yeah. No cases or anything.

Randy Kindig: Apple 1? Apple 1?

Steve Leininger: Apple 1, yeah. There were a few of those, and they, this cookie sheet printed circuit board sold for 666. 66, and you still had to add a keyboard to it, had to add a monitor to it it wasn't the same kind of thing. Even the the y sold the IMSAI, which were the high horsepower S-100 bus the Altair connector, and so anyway, they looked around and Jack Mr. Sellers, Jack Sellers said would you consider consulting on a project that we're working on? Because they knew enough about Don's project to to know that it had moved forward. Okay. Okay. And so I said, yeah I'd consider that. I didn't hear from him for six weeks and then I get a call Saturday morning saying we'd like you to come out and see our facilities.

And interesting thing about that is consultant doesn't really care what your facility looks like. Yeah. And so that was that was HR speak that they wanted to hire me, wife had been working the breakfast shift at McDonald's and she had a degree in marine geology.

Couldn't get a job in the Bay Area. She calls her cousin and who's a VP at Conoco's and says, what are the chances of finding a job in the Dallas Fort Worth area? He said, oh, they're way better for you than the Bay Area. And so I called back and said I need to bring my wife out to check things out also.

And basically unheard of at Radio Shack.

Don French: Totally.

Steve Leininger: Yeah. They had 3, 500 company stores. And, so when they bring people in, they weren't bringing people in and their wives, they were just bringing in the store manager.

Don French: We had an annual meeting of all the managers. And so that would be a lot of wives traveling.

Randy Kindig: That's true.

Don French: Very expensive.

Steve Leininger: Jack said I'll have to get back to you now. Yeah. And so he got back and said, okay, we'll do that. And I don't know what the decision process was behind that, but I think it's, I seem to know more about where the computers might be going than anyone else they'd ever talked to.

Okay, and so that's when I, so we get invited to come to Fort Worth.

Don French: That time, what happened in an intro is they told me about you. . And they wanted me to interview you because they couldn't interview you. . They didn't know what to do. That's what it was. They did not have any idea whether it was good or bad.

Huh. They knew what he was talking about or anything else. He knew way more than I did about the designing of it, which is fine. I didn't care. But I was talking about I didn't have to know what needs to be done. Huh. And when he came in I met with him and we we spent all day that day, I know together, I don't remember how many days you stayed but it, when we got done, one, we did become good friends. At that point. And I said he's the one to hire. They didn't give me any argument whatsoever. They just did it.

Steve Leininger: Yeah. They took us to lunch at a real nice place and Mr. Roach said we'd like to give you an offer. And it turns out the offer was like 20 percent more than I was making at National.

Randy Kindig: That's significant.

Don French: At a higher price area.

Steve Leininger: Yeah Bay area was even then hugely expensive. And Texas had. And it has no income tax. Ding! So it all worked out pretty well. And it met the possible requirements for my wife finding the job. Which she ended up with a 30 some odd year career in the petroleum business.

Which also pays money. And Yeah the interview was basically talking about computers and everything.

Randy Kindig: What kind of questions did you ask in that interview?

Don French: I can't remember now.

Steve Leininger: I think it was more talking about, if you were going to build a computer, what would you do?

Don French: Generic type.

Steve Leininger: And I think the hope was that I was going to be able to save this PACE project that they had gotten started, the 16 bit microprocessor. And it took me two months to figure out that, boy, where that was going just wasn't easy. And

Don French: it was still going to be a project board though.

Steve Leininger: Yeah.

Don French: That did not change yet.

Steve Leininger: Yeah. So it was going to be a project board, which was a kit, and because he would have put whatever microprocessor I'd ended up going in the line. And the problem is that you have to meet a certain level of work it, we decided it needed to have some kind of Liquid crystals weren't, available then and all that.

So it was going to hook to a TV somehow. And I knew some stuff based on work that I'd already been doing. I was making a glass teletype at home. And when I was in the Homebrew Computer Club, making a a terminal basically, built into the computer. And so in the middle of this one of these project boards came back

Randy Kindig: The clock.

Steve Leininger: Yeah. It was the clock.

Don French: It was the most expensive project board to build that we had, 40 bucks or something like that to build it. It was a good, great seller. It was a popular one too.

Steve Leininger: But you had to buy, six, seven segment displays, the LED displays, plus the drivers, plus the resistors, plus the timekeeping stuff and everything.

And we got one back. We were pretty generous about it. Using our engineering time to actually respond to customer problems on these things.

Don French: Technical problems, yeah.

Steve Leininger: Yeah. And they sent the thing in, and the board wasn't much bigger than this little notebook here. But, it the box was suspiciously heavy.

And, the, he pulled the engineer pulled it out, and he looked at it, and he turns the thing over, and there was a sheet of solder over the entire bottom of the board. I, someone figured out how to put an eighth of an inch of solder.

Don French: I don't know how they did it.

Randy Kindig: Wow. Eighth of an inch of solder.

Steve Leininger: And then they went back and read the instructions. It says, insert all the components, turn the thing over and solder everything to the bottom. And they did. And somehow, there was one big slab. And at that point Lew Kornfeld, who was president of Radio Shack said, it's not going to, we're not going to have any computer kits, but pretty well drove the wood to stake through the heart at that point, which was actually a super good idea.

Don French: It was a good idea, but that wasn't my ultimate goal though. That was just a way to get something in the company so I could move it into a real computer.

Randy Kindig: By first offering as a kit, you mean?

Don French: Yeah, the first a project board because I controlled the project boards. And I didn't have to get permission from anybody to do it.

Randy Kindig: I didn't realize that's why you originally went down that path was because you didn't have to have higher up permission in order to move forward.

Don French: Exactly.

Steve Leininger: But during this time Don and I would talk back and forth and I'd, I said, if you've got this board with a board in memory, I said, what are you going to do with it?

We said I've got an ADM-3, which was a thousand dollar, 999. It was a CRT monitor glass teletype basically. And we sold National Semiconductor sold boards that, would plug into that, the RS 232, and then you'd do debugging. You could do some crude programming and all that.

But you really couldn't do anything with it, and you still needed this thousand dollar thing. I said, we need to figure out how to put a display on this as part of it. And at the same time Don had included into the line a really nice keyboard that he'd gotten as surplus.

Don French: Most of, a lot of the stuff, majority of the parks, pieces, everything were surplus.

They may not have made full spec. Yeah. But they were their sub spec, but they were valid. They, they, I wouldn't show, put 'em on something going to the moon . But that wasn't for that though. It was for hobby.

Steve Leininger: Yeah. And so we already had a nice feeling keyboard that one of the other guys was making a keyboard interface for.

By the time you got done with it, you'd spent a hundred bucks at a Radio Shack store, making a keyboard. Which, back in that time frame, actually made a lot of sense. But, it wasn't a computer. And I said, this computer that you guys have been working with National on isn't the one you want.

I talked him off that ledge. And he said what do you think it should have? And I said it, I'd already been working on Tiny BASIC for using Lai Chen Wang's Tiny BASIC. I'd been working on one for National

Semiconductor. I said, what you probably need is BASIC built into the computer.

You need enough memory to run the thing. You need a display. And the popular thing at the time was a cassette interface to do loading things. But the Tarbell, which was the big one at the time. Cassette interface for the MITs computer. The Altair computer was a $295 item by itself.

So we're, every time you look at, if you start looking at all the numbers, you're over $1,200. Before you had something working. And I said, and I was told we, we had to sell this thing for $199.

No. The first part, it didn't have to have a monitor in that.. The computer itself had to be 99. It could have add ons to it. Sure. There was Like a cassette. A cassette. There was a cassette because there were no floppies. At that time. And so that's what we were building is a $99 computer. And that's why there's no uppercase.

And so it got to be increasingly difficult to do something like that. So I started refining some design work I'd done for my machine. I, let's use less memory. If you're doing 80 character lines and 24 lines You'd have to have a really good monitor to do that, right? You can't just use a TV, right? So it was, all right, what can we do and put it on a TV?

We already had a video game, right? We had a Pong equivalent game, right? And so we started looking at, all right what's the lowest common denominator computer you can do? And it would have been using a RF modulator, a radio frequency modulator to go to channel three. Retail for just an old video game and some version of Don's microwave oven keyboard but we had this other keyboard.

Don French: We had a better keyboard.

Steve Leininger: Yeah. So why don't you, why don't we use your nicer keyboard? And we were still, buying surplus ish kind of stuff. So it would have been a sold out when gone kind of thing which is a horrible thing. Yeah. But, when they pulled the kit kit rug out from under us because of that one clock all right, what's our fallback position?

What do we do? And I, what if we take, what if we take this really nice keyboard and you can clack around on it and it was a lot of, it felt good. And I said, and I know you can do basic cause I've done it before. You knew that and a certain amount of memory and memory was kinda complicated 'cause it was pretty expensive except that dynamic RAMs were much less expensive for the same amount of memory.

Don French: Yeah. The old static RAM which was common and dynamic was new and it was affordable.

Steve Leininger: Yeah. It was used in the mainframe it was less expensive. And it turned out that there was a fee there was a new chip on. The 8080 was used in the Altair and several other things.

The 8085 was coming out, which is a 5 volt only version. Big plus, because your power supply is less complicated. But the Z80 came out, and it was 5 volt only. It had Some extra registers, which you can make an argument for, could be useful. But it also had logic built into it that

would automatically update the it would refresh, as we call it the the dynamic RAMs.

Okay, and that became very important to the design. Otherwise, you'd have to have extra parts to do that.

Don French: It's more expensive doing it any other way.

Steve Leininger: Yeah, and this particular chip, they were pretty interested in working with us. It was designed by Zilog, but MOSTech which was in Northern Dallas ,Carrollton actually, was their foundry.

It was Zilog's foundry for the chip, and they could also co produce it. It, it, in the, back then it was important to be able to have multiple sources on things in case one company, lost the recipe. Even though it was just one company that had the recipe. So that chip allowed us to use larger RAM. We can put in 4K of RAM, which was a huge amount at the time.

Don French: That was a huge amount.

Steve Leininger: Yeah. The Altair only came with 256 bytes of memory and then had memory cards. A 4K memory card had four rows of memories and cost two or 300 just for memory. So we were able to get that crushed down.

And by the way those parts work with a couple jumpers. You can use either. 4K or 16K RAMs in there. So now we've got a path to doing something pretty interesting. And so the rest was, all right what else can you do? So we, I talked about moving the video around a little bit and we talked about cassette interfaces and the Tarbell stuff was pretty well described.

You don't have to do it all in hardware. You can do a lot of that in software, it turns out, which I did. Yeah. And we made a prototype and in fact they moved me out of the way of other engineering so that I wouldn't be given other duties and everything. It put me off in a hidden area.

Don French: They went out to the stockyards.

Steve Leininger: Yeah, to Tandy Audio. Where they built the speakers, I was in my own little room. They had an extra room up there.

Don French: Tandy was all over Ft. Worth.

Steve Leininger: Yeah, so it was I was a mysterious guy and ended up building A video card, a memory card a microprocessor card, and probably one other card.

And we had these cards, and they were built in Radio Shack format prototyping cards, hand wire wrapped. I did the wire wrapping on it from my schematics. And using Don's computer as a development system

Don French: the cost of this thing, besides personnel, to do this was really cheap.

They didn't buy any development software. Any development software.

Randy Kindig: I heard somewhere between 100, 000 to 150, 000 total.

Don French: Way too high.

Randy Kindig: Oh, really? Okay.

Steve Leininger: Yeah. That would have been an all in price, including the guys who's manufacturing it and all of it.

Randy Kindig: Salaries and stuff in there?

Don French: No, I said without that.

Steve Leininger: But. And so on, on Groundhog Day. Groundhog Day in 1977 I had started working there on in 76 the day after Independence Day. We got into town on the third. I started work on the fifth and didn't really do a whole lot for the first two months 'cause we were trying to figure, things out and everything got my marching orders somewhere must have been October ish is when we decided not a kit and oh my gosh, we need to do something here, and so the target was in February, we're going to demonstrate this stuff. And they contracted someone to actually write the software underlying the machine, but for the demo purposes, I had to do all the software.

Turns out, in the end, I had to do all the software anyway, because the guy we contracted ghosted us. He disappeared. And when we demonstrated it, it was these boards in a card rack with a wire wrapped backplane stuck under. We had the curtain on it, so yeah. We had a table with basically a curtain on it

Don French: But it had the keyboard in the final case

Steve Leininger: in a thing that looked like it. The plastic guys we used. They had a model maker that could make a looks like. Yeah, okay. It looked like it was. Yeah. And this is before 3D printing or anything crazy like that.

This was just flat pieces of plastic glued together and then painted Mercedes silver to match the TV.

Randy Kindig: Yep, because of the monitor on it.

Steve Leininger: Yeah. Yeah, that's what I heard. And so we had We had a mock up that looked just like the Model 1. And it, but it had integer basic in it. It only had 2k bytes, 2k bytes, that was scanning the keyboard, doing the cassette interface, and and a full 4k basic.

And the video. 2k bytes.

Randy Kindig: So it wasn't even the 4k ROM.

Steve Leininger: It didn't have floating point.

And that, that came up.

Don French: It almost killed it. It almost killed it. I wrote a little program, I call it H& R Shack. It was a 1040 simple, it asked for income, it calculated based on, it didn't take deductions or anything, it

took the calculation of how much it was, and it took the used the tax rate for that, and calculated for it.

Steve Leininger: And so Mr. Tandy comes up there and says, enter your income. So he put in 100, 000, which was low. But it was only good to, 32, 000.

Don French: What I told him was, Charles, take a pay cut. 32, 000 is the maximum amount right now. We know it needs to improve that, but this is just a proof of concept. And he did, put 32, 000 in, and it all worked.

Steve Leininger: And it worked. After he blew a big ball of smoke at it yeah, they, they said, okay, let's go ahead and do this. Yeah. We were going over to the conference room.

Don French: This was in the conference room. And we moved over to the conference table, Charles Tandy, Lew Kornfeld ,John Roach, Steve and me.

Randy Kindig: That was it?

Don French: That was it. That was it. A lot of people knew about it. And we moved it over to the conference table, and Steve and I hadn't talked about this number at all. Didn't even come up to it. And suddenly, it was Charles, I'm sure, asked, said, How many can we sell? We both said at the same time, 50, 000. And they just laughed.

Steve Leininger: To which John Roach said, horsesh**. And this is actually, they asked me about this and when Mr. Roach passed away, and it's in his obituary in the Wall Street Journal. They quoted it not everyone will quote that.

Randy Kindig: Do they say who quote, who who gave them the quote, did they have your name in there?

Steve Leininger: Oh yeah, and was, yeah, it was John, Mr. Roach that actually said that, but yeah, it was because we'd been asked, we were asked to give our numbers. Yeah. Yeah. And, and yeah. Anyway we were pessimistic as it turned out because

Don French: Yeah their number was 1, 000.

That was their number. That's what they wanted to to do.

When they rejected our number, they said they could 1, 000. I've got to remember, this was being done as a publicity stunt to prove that RadioShack could be creative.

Steve Leininger: So as soon as we were given the green light there they took, luckily I was pretty efficient and very complete on my design.

They were able to take that and start laying out the printed circuit board for it.

Randy Kindig: Right off the bat, with what you had?

Steve Leininger: Yeah. Yeah. And About a week or two later, after not being able to find the ghosting Bob Uterich, You're out there, Bob.

Randy Kindig: I'm not going to say who, but Bob.

Steve Leininger: Yeah. I ended up, basically, I was told by about three different people that, Obi Wan Kenobi, you're my only hope, I had to write the, I had to Add floating point. Yeah, I had to add floating point. I had to add Cassette data read and write. I think I had the line graphics already in there.

Randy Kindig: Basically you took it to 4k with all those extra.

Steve Leininger: Yeah, it ended up being 4k bytes. Yeah. But 4k bytes is still remarkably small for what the program did.

Don French: But it added a ROM though, because that's where we, that's what basic was. Everything was in that ROM. We still have 4k of memory, basically.

Approximately. . And the funny thing is, when he got done with it he, it didn't, I know it was a funny, but he said, we got four bytes left. And What do you want? An operating system? A disk operating system? Yeah. Said. Yeah, he suggested put our initials in it.

I don't know if he did or not. I don't know. It didn't matter to me. But we put, we stuck as much into that as possible. To make it a complete system. Because that was the goal. We had decided it had to be an absolutely stand alone system. Complete system. We missed the $99. The computer came out at $199.

And the monitor did not.

Randy Kindig: That's still remarkable.

Don French: It totally blew the market away. And John Shirley, That name should be familiar he became the president of Microsoft that he He was a vice president of Radio Shack in Europe, equivalent of Bernie Appel, my boss. One of the things that I had to do reporting, is I had to report to him what I was working on, just so he was aware, so he could plan for it in his product.

And I wrote him on Telex. Which we did every day for both Europe and our buyers in Asia. And I said, Computing is our next big area. And I'd like to keep you informed of what I'm doing and what we're doing. He wrote me back at Telex and said, RadioShack can't sell computers. Don't waste my time. And he was absolutely 100 percent correct.

RadioShack could not sell computers. They would buy 'em, people would buy 'em, but they could not sell 'em. They did not know how to sell 'em. And we announced that August 3rd, 1977 to the Warwick Hotel. They had the bombing of the tower.

Steve Leininger: The Puerto Rican Liberation

Don French: yeah, something, the bombing of it.

Steve Leininger: It was the Exxon building actually.

Randy Kindig: And were both of you there at that?

Don French: I was.

Steve Leininger: Yeah, I was at the Warwick.

Don French: And but then you were. Then I went to Boston after that. I took our total production run of computers.

Steve Leininger: All three of them.

Don French: Three. To the Boston Computer Fair.

And set it up in there. And I was totally swamped. There were five, six people deep trying to see it. I used to say I was interviewed by all the networks. All was three. At that time.

Randy Kindig: Pretty much all there was at the time,

Don French: yeah. And it hit the press, and when I got back to Fort Worth, the phones wouldn't work.

No phones in that office would work.

Steve Leininger: Because they'd overloaded the system.

Don French: Yeah. That's when every line, every call was on copper wire. They couldn't then multiplex it. And nothing. And so they, finally, they got things fixed, and We were swamped with calls. We didn't have any to sell, but we were swamped with calls.

I, they moved me out of my office. This is a nice office, 10 by 12, something like that. It was a nice sized office. Into the conference room, and the conference room became my office. And brought in an admin for me. And I took all the calls. Because I was the only other one that knew anything about the computer.

Until I knew enough about the marketing of it and all that, and what's out there. And so I told. customers that I'm sorry, RadioShack doesn't have on the we missed on the production schedule. I didn't tell them how much it was, but we missed on it. And you have three choices. If you want a computer right now, you can get one.

It's not from us. It's an Apple. It's $2,100. And but they do have some, I checked to make sure they did. The other one you can get is a Commodore PET. They have some also, and that it's $900. But by the way, that was also my concept. And they went, something went to 'em that the Apple, it saved 'em from bankruptcy, 'cause they were bankrupt at the time. They had not filed.

Randy Kindig: I was wondering just how much by you doing that, how much that increased their business.

Don French: Oh, totally. It saved them.

Randy Kindig: Because you made some awareness of personal computers.

Don French: Yeah. But they sold 7,900 the first year. Not a big number, but without the TRS-80...

And so that's what they sold. It kept us and we sold some, but there still, there was a six month wait in most of it. And we sold them. And Charles passed away before we really got into production, but we had sales. The sales were huge. CB radio was 40 percent of our business the previous year.

It went to 5 percent of our business in that next year. Computers were zero, and went to 40%.

Randy Kindig: Wow. Yeah, that fast.

Steve Leininger: It was the most expensive product that RadioShack had ever sold.

Don French: The most expensive product in RadioShack's history.

Steve Leininger: And we made Radio Shack's traditional high margin on it.

Don French: Yeah, we did.

Steve Leininger: And we learned a lot about manufacturing. We didn't, our manufacturing people do good work, but they didn't understand the level of, the complexity of it. We ended up making PCs later on for companies like Panasonic and DEC digital equipment.

Don French: Because we had a great manufacturing line. And they were all, these were all U. S. made. Nothing was sent onshore. It was made in Fort Worth.

Steve Leininger: Until the PC became a commodity, and at that point it was we could buy the whole populated motherboard offshore for less than we could have bought just the connectors.

At that, at that point you go, we can't do it.

Don French: But we assembled here though.

Steve Leininger: We assembled here. But we got the motherboards

Don French: Yeah, and it it saved Radio Shack because with a dropping CB business as it did, they would have had a major reduction in sales volume. And and so that had an impact.

I, I reminded Charles of that one Saturday.

Steve Leininger: But it was interesting how remarkably close, the Apple II, the Commodore PET, and the Radio Shack Model 1 was actually very close. Radio Shack TRS 80. It didn't become a Model I till later.

Don French: Same as it wasn't an Apple one. It was an Apple computer. It wasn't an Apple one. It was an Apple computer.

Steve Leininger: So anyway but they all came out within weeks of each other. And they all had their strengths and weaknesses. The case for the Apple was a superior design, but the whole thing cost a whole lot more.

We didn't have a monitor that matched it at the time ours was what it was, it was all in one with a monitor, modified TV, and

Don French: But we didn't use RF we never used RF

Steve Leininger: yeah, we never used RF, we never, it was hardwired. Yeah, and it's because we tried it both ways and when you have control over the monitor you get a better picture than that.

Randy Kindig: So you could do 64 characters.

Steve Leininger: Yeah. And right place, right time,

Randy Kindig: and the store itself was a huge benefit.

Steve Leininger: Having 3, 500 stores...

Don French: that's what made it. That's the reason there's a computer industry today. Because without that, nobody would have gotten the acceptance of it.

It took Radio Shack to do it. And because of that, Because it was put in the stores, it was force fed into them, and then did something else right after that. I became aware that Tandy became more interested in it, the corporation and Tandy himself. And so once I had hired another product manager to take the computer partially of it all the way back I opened up what was then called Tandy Computer Center.

That's what it opened as. It became RadioShack Computer Center. But it was a marketplace. A test facility, market test we put Altairs and everything in there, how to sell, what were people asking for?

Steve Leininger: Yeah. And at the same time we started working on the model two. . And the model two...

Randy Kindig: so that was the next project that you worked on?

Steve Leininger: Yeah. Yeah. And we had the color computer. That was also a project at that time. It was an upgrade to a thing that AgVision, we. We've worked with Motorola on providing AgVision was basically what was that? Teletext? It used they still do some of that stuff in Europe with a thing called the Fritz Box.

But, so it the idea was it was gonna replace phone, books and stuff like that in Europe, but the version in the US was being used for the farmers. A special channel for them to get the information they need. And the color computer was an upgrade to that. The model 2 was our first shot at a business machine, as we got a little more sophisticated on the software it put Wang out of business.

Wang was very popular word processing system at the time and we used the economy of scale to, yeah,

Don French: but the Model 2, I left before the Model 2 came out.

Randy Kindig: Okay, I was going to ask you, but you were involved in the project before?

Don French: Just the beginning of it. What had happened, Kandy had promised me, when I saw him at the time and told him that I saved Radio Shack.

C. B. died and the computer just took off. He promised me a large bonus and a vice presidency.

Steve Leininger: And then he died.

Don French: And then he died. And I went to see Kornfeld to get it. And his comment to me was, No young punk, his words, is gonna be a vice president of this company and make that much money. I excused myself and said, We'll talk later.

But I don't get mad like that and just fly off the handle. I went to my office, cleaned my office out. Nobody knew this was going on. I cleaned it out. Talked to Bill Gates, because we had hired him to do the BASIC, because the Model 1 BASIC needed, when we started doing business stuff, even on the Model 1, we needed that one called Level 2 BASIC.

Now it's Microsoft BASIC. It was written for, it was not written, it was highly modified for us. And so our software, our programs we wrote would run on it. And that's the reason IBM went to Microsoft to get the same exact BASIC.

Randy Kindig: It turned out to be a very powerful BASIC.

Don French: It was very powerful and they wanted it because we had so many computers out there.

They were finally allowed to come in because they wouldn't own the market. And that's why the Justice Department allowed them to do a computer. Justice Department allowed IBM to do a computer. And Radio Shack screwed up. I don't know how this happened but they screwed up and went to, the interrupts on Model 1 were a problem because you couldn't run CP/M easily. And so that was a problem. We had to use TRSDOS, the TRS DOS, and it was so one, but the MSDOS was good. When IBM came out, Radio Shack, I was told, this is hindsight. And my brother as a, was a store owner at that point in franchisee. And the information I got is that, yeah, it's the same but IBM software does not run on RadioShack.

RadioShack's version does not run on IBM. And so I don't know, I never tested it because I never bought a RadioShack IBM compatible computer because it wasn't compatible. And but that was the beginning of the downfall of RadioShack. That and they

they changed who they were trying to hire as our president. Kornfeld left, Roach took over of and then, I don't know the person's name, but he was a executive from CBS. In other words, they got away from people who knew what they were doing, and that was, it went on for years and it just kept dropping and dropping.

And that was the downfall of Radio Shack. They cut their own throat basically.

Steve Leininger: Actually, most of that stuff happened after I left too.

Don French: Yeah. Yeah. It was after way after I left.

Randy Kindig: So you. You left, you said in what year was it you left?

Don French: 78.

Randy Kindig: 78, you left. And you said you never even said goodbye to Steve, you left, right?

Don French: I didn't. Yeah. I talked to him afterwards.

Steve Leininger: What's the matter? Don't you love me anymore?

Don French: We talked afterwards because the S 100 bus for it. I had to do a S 100 plug in to the back of the TRS 80, and we're making an S 100 plug in. I have that board

Randy Kindig: so it was after the fact that he found out that he had left, and.

Steve Leininger: Yeah. News travels real quick. Did you hear about Don? No. Says, he's not there anymore. But there was no story that went with it. It's the, he's not there anymore. It's like any other, it's like a murder mystery. You don't know whether, whether he, went crazy, people don't go crazy anymore, of course. And we're talking from Springfield, Ohio, right now.

Don French: I'll clear the air of what happened. After Kornfeld told me, no young punk's going to be a vice president coming up. And I talked to Bill. He offered me a job. I turned him down. And I just thought I was going to do it myself.

I went back to see Kornfeld, and before I could even get in his office.

Steve Leininger: How did that work out?

Don French: What's that?

Steve Leininger: How did that work out?

Don French: He said, if you're here about the bonus and the promotion, the answer is still no. I'm going to keep it really clean from what I said.

Steve Leininger: There wouldn't have been nothing to say.

Don French: Yeah. I said, basically, I looked at him, turned to leave, looked back at him and said, Do something to yourself that's impossible. Oh, yeah. I did, and was walked out. Didn't turn back. I had, I knew what, I burned bridges on purpose. I was not going back, but I told myself if I, if they wanted me, they would have to pay through the nose.

And it would be a, not just a large bonus, it would be an extremely large bonus. And a contract, because I don't trust them. I didn't go back. I started my own business. Doing CP/M for the TRS-80. That's what I did. And Of course, that was a problem I had to change with Gary Kildall, all of the interrupts, because we couldn't use the low interrupts, because we were using them in the computer, TRS-80.

Don French: And so that wouldn't work. But I did that, and it sold very well. It's what I was able to do to start my business. And switch to healthcare. 1983 and been doing health care since then.

But it's been and there are a lot of things in the whole history that could have stopped it from being ever, or ever existing. I joined the Navy at 16, at 17. I wanted my college paid for and the only way to do that was to military. And so I I did it. My high school girlfriend's dad was the recruiter that got me in and I went to bootcamp. There was a special program they did. You could go to bootcamp between your junior and senior year when it was a radioman, because I went in as an E three. I didn't know enough about analyzing at that point I made could have been a deadly mistake. Marines don't have Radioman or they use Navy radiomen.

You travel on foot with the Marines on foot with the Marines, and with a radiomen with a sign over your head saying, shoot. The antennas back then were four or five feet tall. I could have been easily killed. But I got out the commander got me out because I screwed my knee up back again in high school.

And so he said, we don't want you. But it kept me out. That's one thing that could have ended it right there. Never happened. I wouldn't have been at Radio Shack. And then at Radio Shack, that's where it happened. I applied for a job after I was a disc jockey for a couple of years. It's gone to an AM station.

And I applied for a job temporarily at Radio Shack for Christmas. And I saw a store manager and he said, yeah, I'll be glad to hire you. Went in the next day, he said he'd have an application. And I told him I filled it out. And he saw I was 16. He said, oh, I can't hire you. You're older come back. I'm glad to.

I went the next day. He was gone. He was actually promoted to another store. The new manager there saw the representation. I'll hire you right now. So I got on there. The same thing happened going to Fort Worth. I was turned down by Bernie Appel. He said I needed five more years in the score. Four months later, I met him again at a regional meeting.

I had my five year pin on and they asked five years. I said no, seven. He asked if I would be interested in if I knew anything about electronics and I told him, the ham operator, and I've done some work and he asked me to come to Fort Worth. If I hadn't done that, yeah. Yeah. With a computer. Yeah. And if I hadn't been as committed to doing it, I was totally committed because of an issue that, through elementary school, that I was not taught properly, let's just put it that way that I wasn't dyslexic or anything, I just was bored and and so they did not teach me properly.

I didn't learn to read until the fourth grade and I they told me I'd never go to college. They were gonna put me in industrial arts in high school. I said, I'm going to college. And they because I was committed in my mind, I knew I was going and then I they, I had to signer. My mother had to sign also an agreement with the school saying that if I, when I didn't graduate from high school, not if I, when I didn't graduate from high school, I would I couldn't sue 'em.

And I accepted that. I knew I was going to graduate. I graduated three and a half years. But again, I was committed. And that's where it was

with this. I was committed totally to this. And after the 1, 000 projection that they made, we were working on it. We're going down that line. Kornfeld came to me and said, we've upped the production run 4, 500 since we have 4, 000 stores when the product fails, not if.

They did not believe in us yet. We want to use it for inventory control and I talked to our computer group about it. And they said, yes, we can. I wanted to tell them something again, profanity about you idiot. It's a computer. Of course you can. And but that's how we got to 4,500 as a production run. And they were making them is when once we got it announced that we're making, they weren't making any except the three that we have.

For the show, but they started making them quickly.

Randy Kindig: Yeah. Very. When the phone calls started coming in.

Don French: Very quickly.

Randy Kindig: When the interest exploded. Yeah, there's so many inflection points where things might have gone differently or

Don French: Easily. Easily.

Randy Kindig: It's amazing.

Don French: That's why, this thing, it's changed the world and I know, Steve and I did it.

I've never said I did it. It was my idea to get in the computer market, but it was his design, and so I said, we did it together, and people ask me questions about it, and they say, you had a major impact. I said, yeah, I know, and I put my pants on one leg at a time.

Steve Leininger: When I told friends and family that I was, going to Radio Shack because they had this computer project, it was Radio Shack, yeah, it was funny because, It was a geek friendly store, but but it wasn't, It was never top quality. No, it wasn't top quality. And in fact, the computer helped draw Radio Shack up for quite a while,

Randy Kindig: and probably kept them around a lot longer.

Steve Leininger: And it would have been, it would have been interesting if Charles Tandy would have stayed alive. Because Have been completely different. He had a rough way of dealing with things, but he kept a very shallow structure. You are only like two people away, your boss.

It wasn't a big hierarchy of people. It was it was your boss and Mr. Tandy in a lot of cases. And there might have been. And maybe two bosses between you and Mr. Tandy. But it was that shallow of a structure that and he if he didn't like something, there was no guessing about it.

Randy Kindig: You knew he didn't like it, right?

Steve Leininger: And Mr., I love John Roach to death, but he was more gentle than Mr. Tandy was. Probably to the detriment of the company, because people would make mistakes and quit twice for the same reason, the boss that was directly above me.

And the third time it was because Radio Shack marginalized the work I was working on. I was working on, I had a, there, there used to be a website within Radio Shack's website called Steve's Workbench. And better versions of it now at places like Adafruit and other companies, the stuff that's being done on GitHub and anything that Arduino does is pretty good.

A bunch of stuff that we missed opportunities on yeah, woulda, coulda, shoulda, it, I still have fun in the maker business and we my wife and I actually do STEM programs at the Dinosaur Resource Center in Woodland Park

Don French: the result of a simple thing, a concept, that was a, trying to get a foot in the door, has been amazing. Because that's all the Project was to get a foot in the door.

Steve Leininger: And it was very Wright Brotherish. Wright Brothers is two guys that, that, oh, they learned how to fly, right?

And the industry has grown up hugely. from their initial stuff, but, I don't believe they were ever as successful as, the people that came on beyond them in the airplane industry. And, so anyway, I'm glad to have been part of it.

Don French: Yeah, I'm glad that you were working that day.

Steve Leininger: Yeah. No, a lot of happy accidents.

Don French: There were a lot of happy accidents in the whole thing. Absolutely true. I don't know that When I met Steve immediately, I just knew we were going to hit it off. I didn't know how much until we started, I started talking with him and we had a lot of discussions and then when the the clock issue, we both were on the same page and trying to do something.

Steve Leininger: Yeah. I always thought we'd lost the project, because we couldn't do the kit anymore. Which was Actually the right answer, but at that point then, do you do a CPU and memory board and then have an add on cassette board and then have an add on keyboard?

Don French: If you're doing a hobbyist, yes, but if you're doing what we were doing, no.

Steve Leininger: Yeah the idea then was we got to do a finished product.

Randy Kindig: Just plug it in, turn it on, and it's ready to go.

Steve Leininger: An appliance, as you might call it.

Don French: The Radio Shack had writers for books and that. Forrest Mims, a well known writer. They didn't have anybody to write a manual. I remember my junior college business writing professor who was also a computer professor.

So I called him, David Lien.

Randy Kindig: Oh, so he's, that's how that happened. He did a great job.

Don French: He jumped all over it. I made a multi millionaire out of him. We've made a lot of multi millionaires. Even billionaires.

Steve Leininger: It's like at the show that we're at here, in Springfield, Ohio, the Tandy Assembly Show.

So many people come up and say because of you, I had a job. . Or it changed my life. The direction that stuff I get,

Don French: you get 'em all the time. And actually the first one happened for me before it was even, it was announced, but it was doing what we call our road shows with it.

You're in town and they set up an exhibit where people come in and see it. 'cause we didn't stock 'em, the stores at that point because they didn't have any . And, I was then one in Fort Worth. A gentleman walks up to me and says Are you Don? I had a tag on, and he said, Don French? And I said, yeah.

And he said, are you the one? And I said, I don't know. What did I do? I did the computer of that you're talking about. He said, yeah. He said, I want to thank you for it. And I said, oh, He said, you don't know what happened. My daughter is autistic. And she was struggling. I bought the computer and wrote some software, and she's doing so much better.

That made it all worthwhile for me. I love kids. I have five kids, and I love kids.

Randy Kindig: Yeah, that would that would make you feel good.

Don French: And I hear it all the time, though. I hear it over and over again here. of how they changed their career. They did everything that's Their life would be totally different if it hadn't been for this computer.

And that's the thing that that people ask me what I'm proud about. That's the thing I'm proud about. I helped people. We helped people. We helped people change their career and be better.

Randy Kindig: Excellent. That's a probably a great stopping point. I really appreciate the time from you guys today. Thank you.

It's been a lot of fun to hear how you guys met and then interact with each other. And all that and

Steve Leininger: most of it's true.

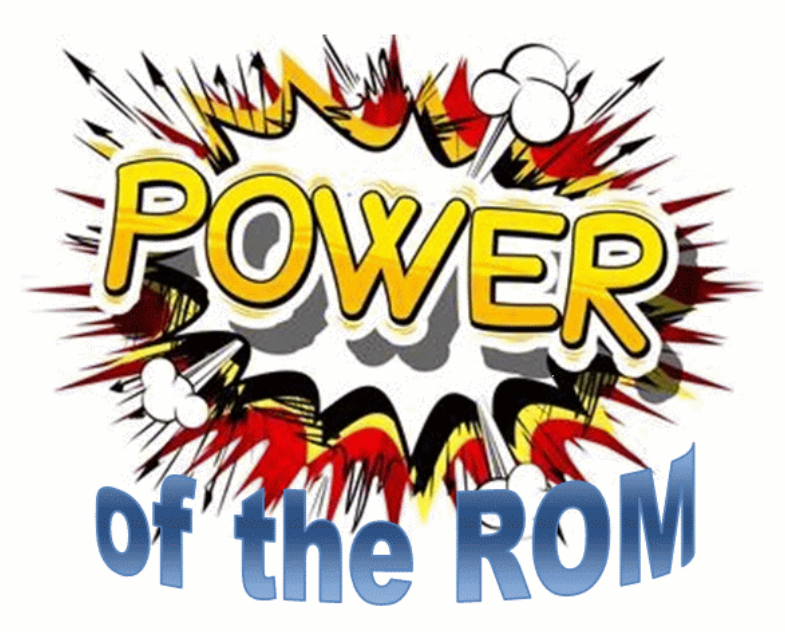
Randy Kindig: Yeah. Most of it. Most of it.

Don French: Yeah. I've signed more computers today than I have in my life.

Randy Kindig: All right. Thank you very much gentlemen.

Steve Leininger: Thank you.

Don French: Thank you.s

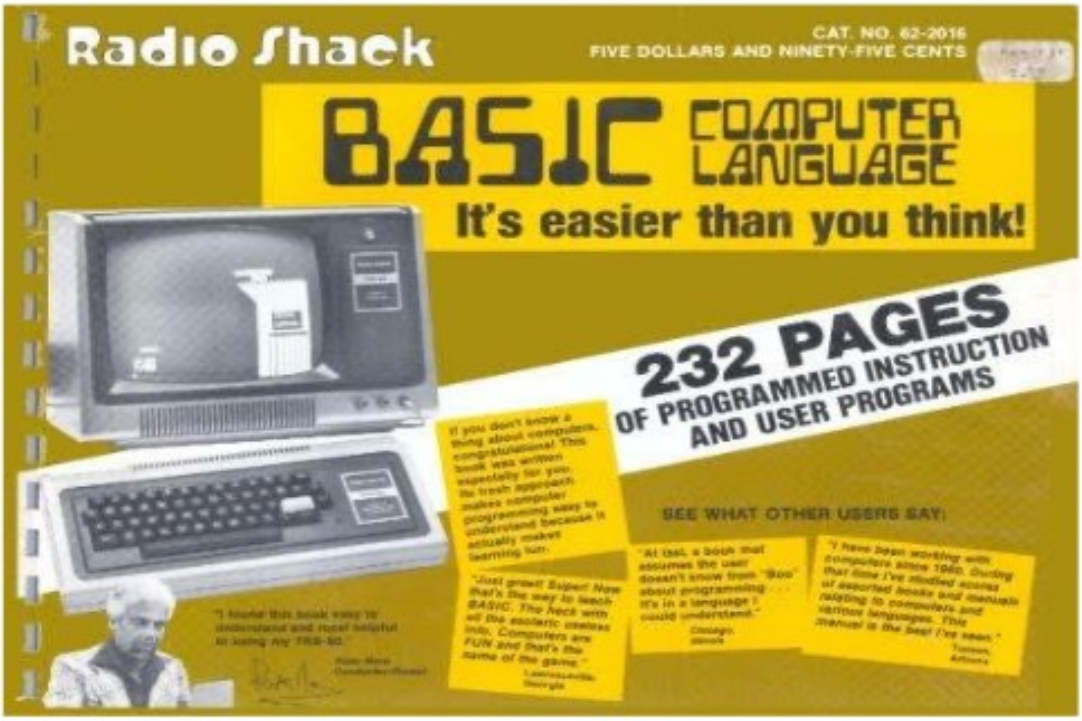


TRS-80 ASSEMBLY LANGUAGE

It's easier than you think!

Vince Otten

If you recognize the subtitle, you may have memories similar to mine. It's 1978, and I'm in the 2nd year of my Refrigeration and HVAC apprenticeship, working out of a small shop in Cambridge, ON. By now I'm pulling in 50% of a licensed mechanic's income, so I've been able to afford a TI-58 programmable calculator, but what's really got my attention are these "microcomputers" showing up in *Popular Electronics* and *Byte* magazines. It's challenging for a Canadian to try to mail-order an Altair from Albuquerque, so I'm thrilled to discover that the local Radio Shack is selling one of these micros!



On every lunch break, I run over to the store and investigate their floor display. The manager is kind enough to let me read this marvelous introduction to Level I BASIC, and I type in the sample programs. I've never forgotten the thrill of learning, due to Dr. David Lien's unique, friendly style.

Now, here's what I want to do. I've learned a bit about using ROM calls in assembly language on my Model I, and I'd like to introduce them by following the programs Dr. Lien taught in this book, like

```
10 PRINT "HELLO THERE! I AM YOUR NEW TRS-80 MICROCOMPUTER!"
```

(page 8), and

```
10 REM * CELCIUS TO FAHRENHEIT CONVERSION *
20 C = 25
30 F= (9/5)*C + 32
40 PRINT C; "DEGREES CELCIUS =" ;F;"DEGREES FAHRENHEIT."
```

(page 25), but converting them to Z-80 Assembly Language. I want my programs to produce the same result.

CLS

Okay. Dr Lien's first program is the "HELLO THERE!" above. But every BASIC program I wrote back in the day began with something like

```
10 CLS
```

because I felt that every production had to start with a blank slate. So I am going to introduce ASM (Assembly Language) ROM calls with this simple example.

```
; CLEAR THE SCREEN
CLS       CALL      01C9H
```

There. That was easy. Try it--it works. Just one teeny problem, which you might only notice if you're running it in my favorite prototyping environment: Microsoft's "Editor Assembler Plus". Here's a fancier BASIC program which demonstrates the difficulty.

```
10 CLS
20 A$= INKEY$: REM ONLY WORKS IN LEVEL II BASIC
30 IF A$ = "" GOTO 20
```

There. The screen gets cleared, and nothing appears until you press a key. Let's try the same with ASM:

```
; CLEAR THE SCREEN
LINE10      CALL    01C9H
LINE20      CALL    002BH       ; "INKEY$"—KEY PRESSED IN ACCUMULATOR A
            OR      A           ; IF NO KEY PRESSED, A CONTAINS A 0
            JP      Z,LINE20

            RET
```

Try this, but before you do, a few quick reminders:

1. In any assembler, you need to "finish off" your code with the
   END
   instruction.
2. If you're assembling to tape or to disk, the "END" instruction must be followed by the address where you want your code to start executing, like
   END LINE10
3. If you're not using Editor/Assembler Plus ("EA+") to assemble your code directly to memory, you'll also need to begin your code by using the "ORG" instruction to specify an address where execution will begin. For example, ORG 5200H.

So here's what the code looks like in its entirety, ready to assemble to disk or tape:

```
            ORG    5200H

; CLEAR THE SCREEN

LINE10      CALL   01C9H
LINE20      CALL   002BH     ; "INKEY$"--KEY PRESSED IN ACCUMULATOR A
            OR     A         ; IF NO KEY PRESSED, A CONTAINS A 0
            JP     Z,LINE20

            RET

            END    LINE10
```

Using EA+, I assembled this with the command line

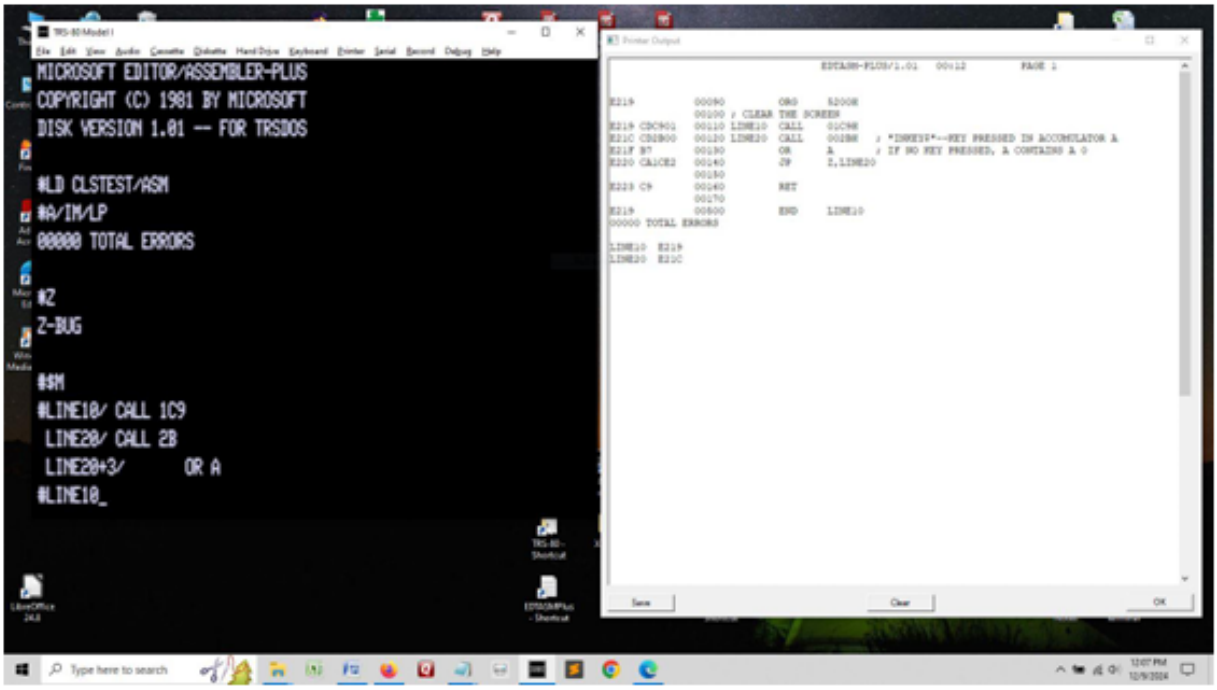
```
A CLSTEST:3
```

which created

```
CLSTEST/CMD
```

on the floppy in drive :3 . Then I tested it—great! It works.

Going forward, I'm going to assume that you're including the "ORG" and "END" instructions in your assembler source files, so I can concentrate on the working code segments. Not only does this make the code examples a bit cleaner, but it hides the fact that I typically write and test my code *in memory* using EA+'s symbolic debugger. When assembling into memory, EA+ decides where to place the program, so you don't typically use ORG or give END an address. This is what that looks like, using the trs80gp emulator:



The command "A/IM/LP" tells EA+ to "Assemble Into Memory, listing to the Line Printer". After assembly with 0 errors, I invoked the ZBUG Symbolic Debugger, and chose "Mnemonic Mode" with the $M ([Esc] M) command. Then I listed a few lines of the assembled code in memory—you can see from the printed listing that "LINE10" starts at memory location E219H. Finally, I was getting ready to run the code, using the command

```
LINE10$G
```

("LINE10 Go"). I've found EA+ and its in-memory symbolic debugger a delight in learning assembly and developing my library of subroutines. It's a lot like programming in BASIC: write a few lines of code, and then run them to see if they work. Examine some variables, go back and edit the program, and try again—all without leaving your programming environment. And this was not modern Visual Studio, but a single program from the early 1980s running on the original TRS-80!

Well. That was exciting. We cleared the screen.
I think I better stick with Dr. Lien's program. What's his first bit of code?

```
10 PRINT "HELLO THERE. I AM YOUR NEW TRS-80 MICROCOMPUTER!"
```

That should be pretty easy to write in Assembly language, right? There *is* a ROM subroutine we can CALL to print to the screen—but it only prints one character at a time! When the BASIC interpreter comes across a program line like this one,

```
10 PRINT "HELLO THERE."
```

it does some "parsing" and decides to send "H", "E', "L", "L", "O", etc. to the subroutine, one character after another.

Here's the subroutine CALL to print one character to the screen:

CALL 0033 Video Display

Displays the character in the A-register on the video. Control codes are permitted. All registers are used.

```
;
; DISPLAY MESSAGE ON VIDEO
;
      LD       HL,LIST      ; MESSAGE ADDRESS
LOOP  LD       A,(HL)       ; GET NEXT CHARACTER
      OR       A            ; TEST FOR END OF MESSAGE
      JR       Z,DONE       ; JMP IF END OF MESSAGE (DONE)
      PUSH     HL           ; NT END, PRESERVE HL
      CALL     33H          ; AND PRINT CHARACTER
      POP      HL           ; RESTORE HL
      INC      HL           ; BUMP TO NEXT CHARACTER
      JR       LOOP         ; LOOP TILL ALL PRINTED
DONE  .
      .
      .
LIST  DEFM     'THIS IS A TEST'
      DEFB     0DH          ; CARRIAGE RETURN
      DEFB     0            ; END OF MESSAGE INDICATOR
```

This is from James Farvour's "Microsoft BASIC Decoded & Other Mysteries" (1981). What's helpful is that he gives example code that actually prints out an entire string ('THIS IS A TEST'). It starts by loading HL with the address of the string called "LIST". Then it loads the accumulator A with the first character of that string: LD A,(HL) says, "Load the accumulator with whatever HL is 'pointing to'". Notice that we "TEST FOR END OF MESSAGE" by ORing A with itself:

```
OR      A
JR      Z,DONE
```

This bit of code is looking for a "zero byte". LIST is made up of the string 'THIS IS A TEST', a 0DH byte (which is a "carriage return"), and a 0 byte ("END OF MESSAGE INDICATOR"). BASIC considers a string to "end" when the interpreter encounters the second double quote (") in a literal like line 10 above. But many languages like C see the end of a string as a zero byte, and that's the

convention that Farvour uses in his example. I think it's a good tradition, because then a string can contain any ASCII value, including double quotes. So here his example code first tests for the ending zero byte with "OR A". If the accumulator contains a 0, "OR A" will set the Z flag, and then the Z80 will jump to "DONE".
Next, the sample code specifies

```
PUSH        HL
```

to preserve a copy of HL on the stack. Apparently, the ROM call will "clobber" some registers—use them for its own nefarious purposes—so we can't assume that HL will still contain the same address of LIST after we CALL 33H. Now at last we're ready to print whatever is in A to the screen with

```
CALL        33H
```

Then we POP HL to retrieve the address of LIST we preserved with a PUSH HL earlier. Next, we INC HL to point to the *next* character in LIST, and then circle back to LOOP to test-and-print all over again.
Notice that Farvour added a 0DH byte—a Carriage Return—after the message. His sample code is similar to BASIC's PRINT, but like this:

```
PRINT "THIS IS A TEST";
```

Notice the trailing semicolon. Remember that in BASIC, that indicates that the next PRINT statement will continue on the same line, so if you added this code:

```
PRINT "AND SO IS THIS"
```

what you'd see on the screen is

```
THIS IS A TESTAND SO IS THIS
```

but if your BASIC code was

```
PRINT "THIS IS A TEST"; CHR$( 13);
PRINT "AND SO IS THIS"
```

then you'd see

```
THIS IS A TEST
AND SO IS THIS
```

The 0DH byte following 'THIS IS A TEST' in our assembly-language sample performs the function of the CHR$(13) in the BASIC code. What I've discovered as I've been exploring assembly language and ROM calls is how I have to track *every tiny detail*.

So here's our version of the first BASIC program in Dr. Lien's book:

```
;
;     "LIENP08A/ASM"
;
;     PRINT EXAMPLE FROM LIEN P. 8
;
; 10 PRINT "HELLO THERE. I AM YOUR NEW TRS-80 MICROCOMPUTER!"
;
LINE10  LD      HL,HELLO
        CALL    PRINT
        JP      FINISH

HELLO   DEFM    'HELLO THERE. I AM YOUR NEW TRS-80 MICROCOMPUTER!'
        DEFB    0DH             ; RETURN / "NEWLINE"
        DEFB    0               ; END-OF-STRING MARKER

PRINT   LD      A,(HL)          ; WHILE (NEXT CHAR
        OR      A               ;     IS NOT A ZERO
        RET     Z               ;     BYTE):
        PUSH    HL                  ; SAVE ADDR OF STRING
        CALL    33H                 ; PRINT CHAR
        POP     HL                  ; RESTORE STRING ADDR
        INC     HL                  ; AND INCREMENT IT
        JR      PRINT           ; :WEND
```

I've turned Farvour's sample program into a subroutine called PRINT. To simulate his BASIC Line 10, I just load HL with the address of the string I want to print, HELLO, and then call my subroutine. Finally, I exit the program with JP FINISH (more about that in a minute).
Compared to assembly language, BASIC is remarkably capable—and complex. BASIC's PRINT statement is a prime example. PRINT can handle string and numeric "literals", like

```
PRINT "HELLO THERE"
```

and

```
PRINT 3
```

but it can also deal with variables

```
PRINT X
```

and expressions like

```
PRINT X / 3 + 9
PRINT LEFT$( "HELLO THERE", 5)
```

And formatting suffixes like trailing commas and semicolons, and even the amazing USING modifier (which itself can be specified with a literal, or a variable, or an expression):

```
PRINT A, 3.14, X
PRINT "THIS IS A NUMBER: "; X
PRINT USING "###.##"; A
PRINT USING F$; X
```

My assembly-language PRINT subroutine can only do *one* thing: print a zero-terminated string like HELLO. If I want my PRINT to handle a number, that will have to be converted to a zero-terminated string first. If I want it to print an expression like X / 3 + 9, that expression will have to be resolved to a number, and that number will have to be converted to a zero-terminated string and...you get it. The good news is that this series of articles will demonstrate a way of doing just that, using Level II BASIC's ROM calls.

A note about JP FINISH, which ends our program. I didn't list the code at FINISH, because that depends upon your "programming environment". You could plan to assemble the program and run "LIENP08A" as a cassette-based "SYSTEM" program. In that case, when your program ends, it will likely "dump" you back into Level II BASIC, so FINISH could look like this:

```
FINISH      JP      0000H           ; just reboot the TRS-80
```

Maybe you're a T-Bug expert, and have used that debugger to LOAD your cassette-based program. In that case, you might want to jump back into T-Bug with

```
FINISH      JP      43A0H           ; re-enter T-Bug
```

Or maybe you're writing this with a TRSDOS-based assembler, and will create a LIENP08A/CMD program. In that case, you'll want to exit back to TRSDOS, so

```
FINISH      JP      402DH           ; re-enter TRSDOS
```

In my case, I write and test my programs with Microsoft Editor Assembler Plus for disk. As I showed above, I assemble into memory, and then test it with Z-Bug. I can re-enter Z-bug with

```
FINISH      JP      5786H           ; resume Z-bug
```

Next: in Chapter 2, Dr. Lien shows Level I BASIC newbies "How To Expand A Program" by adding

```
20   PRINT "YOU CALLED, MASTER. DO YOU HAVE A COMMAND?"
```

Here's how that would look in assembly language, using our first program as a starting point.

```
LINE10      LD      HL,HELLO
            CALL    PRINT
LINE20      LD      HL,COMAND
            CALL    PRINT
            JP      FINISH

HELLO       DEFM    'HELLO THERE. I AM YOUR NEW TRS-80 MICROCOMPUTER!'
            DEFB    0DH         ; RETURN / "NEWLINE"
            DEFB    0           ; END-OF-STRING MARKER

COMAND      DEFM    'YOU CALLED, MASTER. DO YOU HAVE A COMMAND?'
            DEFB    0DH
            DEFB    0
```

(I'm not listing PRINT or FINISH as they remain identical.) Once we've set up the PRINT subroutine, it's pretty easy to add similar lines like LINE20. The process of identifying common procedures and creating subroutines that can do their work "out of sight" is called *factoring*. Factoring is a valuable process because it

a) makes the program more readable;
b) makes it more reliable because the factored code—the subroutine—gets tested and debugged *once* ;
c) reduces program memory usage.

Factoring is an essential element of languages like Forth and Lisp, and I recommend it highly. Here's another example of factoring: Dr. Lien then teaches how to separate lines of print with

```
10 PRINT "HELLO THERE. I AM YOUR NEW TRS-80 MICROCOMPUTER!"
15 PRINT
20 PRINT "YOU CALLED, MASTER. DO YOU HAVE A COMMAND?"
```

Of course, we can easily update LIENP12A/ASM like this:

```
LINE10      LD      HL,HELLO
            CALL    PRINT
LINE15      LD      HL,RETURN
            CALL    PRINT
LINE20      LD      HL,COMAND
            CALL    PRINT
            JP      FINISH

HELLO       DEFM    'HELLO THERE. I AM YOUR NEW TRS-80 MICROCOMPUTER!'
            DEFB    0DH
            DEFB    0

RETURN      DEFB    0DH             ; RETURN / "NEWLINE"
            DEFB    0

COMAND      DEFM    'YOU CALLED, MASTER. DO YOU HAVE A COMMAND?'
            DEFB    0DH
            DEFB    0
```

or we can factor the "data" a little like this:

```
HELLO       DEFM    'HELLO THERE. I AM YOUR NEW TRS-80 MICROCOMPUTER!'
RETURN      DEFB    0DH
            DEFB    0

COMAND      DEFM    'YOU CALLED, MASTER. DO YOU HAVE A COMMAND?'
            DEFB    0DH
            DEFB    0
```

Another way to factor this is to create a new subroutine that uses PRINT. Here's how that looks:

```
;
;     "LIENP12B/ASM"
;
;     PRINT EXAMPLE FROM LIEN P. 12
;
; 10 PRINT "HELLO THERE. I AM YOUR NEW TRS-80 MICROCOMPUTER!"
; 15 PRINT
; 20 PRINT "YOU CALLED, MASTER. DO YOU HAVE A COMMAND?"
;
LINE10      LD      HL,HELLO
            CALL    PRINT           ; PRINT "HELLO..."
LINE15      LD      HL,ENDSTR
            CALL    PRINL           ; PRINT
LINE20      LD      HL,COMAND
            CALL    PRINL           ; PRINT "YOU CALLED..."
            JP      FINISH

HELLO       DEFM    'HELLO THERE. I AM YOUR NEW TRS-80 MICROCOMPUTER!'
RETURN      DEFB    0DH             ; RETURN / "NEWLINE"
ENDSTR      DEFB    0               ; END-OF-STRING MARKER

COMAND      DEFM    'YOU CALLED, MASTER. DO YOU HAVE A COMMAND?'
            DEFB    0

PRINT       LD      A,(HL)          ; WHILE (NEXT CHAR
            OR      A               ;     IS NOT A ZERO
            RET     Z               ;     BYTE):
            PUSH    HL                  ; SAVE ADDR OF STRING
            CALL    33H                 ; PRINT CHAR
            POP     HL                  ; RESTORE STRING ADDR
            INC     HL                  ; AND INCREMENT IT
            JR      PRINT           ; :WEND

PRINL       CALL    PRINT           ; PRINT THE STRING
            LD      HL,RETURN       ; NEWLINE
            CALL    PRINT
            RET
```

We first "factored the data" by including labels for RETURN ("Return" key/CHR$(13)/"newline") and ENDSTR ("End of String") within the HELLO string data structure. That comes in handy as we also write a new subroutine that uses PRINT to first print whatever HL is pointing to, then PRINTing a "Return" or newline. A final bit of factoring is in LINE15, where we emulate BASIC's standalone PRINT statement in line 15 by telling PRINL to PRINT an empty string—loading HL with the address of ENDSTR. (Assembly language programmers do this kind of tricky stuff all the time.

It goes with the territory. It's shocking how often Bill Gates and Paul Allen jumped right into the middle of their own ROM BASIC code in order to re-use procedures and save some memory space.)

In the next article, I'll start diving into the third chapter of Dr. Lien's book, "But Can It Do Math?" The arithmetic subroutines in the ROM—especially for floating point math—were what drew me into investigating BASIC's machine code in the first place. I'll show you not only how to use the math functions, but how to set up your assembly language programs to avoid crashing when you do use the ROM calls.

TRS-80 & Video Genie Programs

DUPLITAPE £9.50

A copy program which can copy any length program or data-file on a machine with 4K of R.A.M. DUPLITAPE is not a simple copy program but is designed to alter and reposition the tape pulses so that difficult tapes will load easily. It requires two cassette recorders. It will copy most protected tapes and adventures.

CHAIN LINK £9.50

This program allows long BASIC programs ot be split up so that they will run in a small memory size. It allows variables including strings to be passed from one program to another. It has a built in MERGE for BASIC programs and allows two BASIC programs to be kept separate in memory if needed. 16 KL11 only.

WIZARDS CASTLE £9.50

A role playing adevnture with real maps. The objective is to find the treasures in a 500 room castle. The castle has monsters which have to be fought, together with other hazards. Every time the program is run, the castle will be different. Magic spells are also used. For dwarves, hobbits, elves and humans. 16K L11 only.

ACULAB DATA-BASE £9.50

A general purpose data base to run with an Aculab Floppy Tape under Extended BASIC. The use of the P.I.M.S. manual by Scelbi Publications is recommended. Functions supported include Help Command, Arithmetic, Searching, Sorting, Listing, Printer output, if required; Printing of 5 line address labels. Commas are allowed in the text lines. As this is intended for present owners of P.I.M.S., the instructions will only cover the commands fully, other commands being summarised.

AND NOW NIGHT DRIVER BY B. DANIELS

A fast 3-D car drive along a road at night. You see the twisting road through your windscreen. Can you control your car and avoid the police speed patrol? With sound.

Disc £11.50 Tape £9.50

From **ALBION SOFTWARE, LAMMAS ROAD, LONDON E10** *or SAE for more details*

Software Tools In BASIC - Episode 6
"The Shell Game"

Vince Otten

Can you imagine waiting for a train, only to discover that just the locomotive arrived, but no coaches? That's what typically running a BASIC program on the TRS-80 is like: you invoke "BASIC" from TRSDOS, then respond to the prompts for the Memory Size and number of Files, and finally

```
RUN "MYPROG/BAS"
```

There's no way to specify command-line options like we're accustomed to with MSDOS or Unix or Linux:

```
MYPROG/BAS –FLAGS <SOURCE/FIL >/DEV/PRINTER
```

So, for "Software Tools in BASIC", I created a "shell program", SH/BAS, which allows you to input a single command line, parses its options into "standard input", "standard output", and "standard error" files, preserves the rest of the options as a line that your program can interpret, passes that information to your program in a file called "CMDLINE/SH", and RUNs your program. If your program was written using SKELETON/BAS as its outline, it will read in that information from CMDLINE/SH and process it as you've developed it, and when it's finished, CLOSE any files and run SH/BAS again, waiting for another command line.

Here's what using SKELETON/BAS as the framework for a software tool and then invoking that tool from SH/BAS looks like:

Original RATFOR code from "Software Tools" by Kernighan & Plauger:

```
# copy – copy input characters to output
    integer  getc
    integer  c

    while (getc(c) != EOF)
        call  putc(c)

    stop
    end
```

Converted to BASIC code that can run in the SKELETON/BAS environment:

```
39999 '

    COPY -- COPY INPUT CHARACTERS TO OUTPUT (ST P. 8)

    WHILE (GETC() != EOF)

40000 GOSUB 2100: IF NOT VR%(0) THEN GOTO 40200'

        CALL PUTC( VA$(0))

40010       VA$(0)= VR$(0): GOSUB 2300:
            GOTO 40000'

    STOP

40200 CLOSE
```

1. In BASIC,
   LOAD "SKELETON/BAS"
2. Type in lines 39999..40010 above.
3. SAVE the program:
   SAVE "COPY/BAS"
4. Fire up the "shell" to test your program:
   RUN "SH/BAS"
5. Sample command line in the shell to try it out:
   $
   $ COPY <FILE1/ASC >/DEV/PRINTER
   $

As mentioned previously, SH/BAS does the following to our sample command line:

1. Memorizes the program name ("COPY"). If it doesn't have an extension, it appends "/BAS" before actually RUNning it. The program name will be saved in CMDLINE/SH *without* any amendments as the first part of line 1.
2. Parses out any standard files, and saves them as lines to be written out to the CMDLINE/SH file: stdin in line 2, stdout in line 3, and stderr in line 4.
3. Any remaining text will be saved as a string of argument(s) appended to line 1, after the program name.

Here's what the CMDLINE/SH file would look like, then:

```
COPY
FILE1/ASC
-3
```

In this case, line 4 will be empty, indicating to the program that stderr will go to its default destination (the screen). Line 3 contains a device indicator: programs built on the SKELETON/BAS framework will interpret it as "printer".

4. SH/BAS will then
RUN "COPY/BAS"

and that program, having been built on the SKELETON/BAS outline, will read CMDLINE/SH and open the appropriate files so that stdin, stdout, and stderr are attached to the files or devices we specified.

SH/BAS itself is written on the SKELETON/BAS framework so that it can use some of its useful subroutines. The main program (starting at line 40000, of course) is quite simple:

1. After some initialization,
2. Prompt the user for a command line (like "MYPROG –FLAGS <SOURCE/FIL >/DEV/PRINTER"),
3. "Parse" the command line, ie divide it up into—
    a. a program name ("MYPROG/BAS"—remember that SH/BAS will append a default "/BAS" if an extension is not supplied);
    b. any input redirection ("SOURCE/FIL");
    c. any output redirection (">/DEV/PRINTER");
    d. any error redirection (such as "3>ERRFILE/DAT:3", but none in this example);
    e. a command line stripped of the redirection ("MYPROG –FLAGS")
4. Write out CMDLINE/SH based on the parsing in step 3, and
5. Finally, RUN the program name (ie, 'RUN "MYPROG/BAS"').

Here's the BASIC code for that main program:

```
39999 '
    INITIALIZATION:

40000 DIM F$(3), SS%(3)
40010 FOR X9%= 0 TO 3: F$(X9%)= "": NEXT' INIT STD FILENAMES
40030 PT$= "$" ' PROMPT
40032 SN%= 0' "STREAM NUMBER" (1..3 STDIN..OUT..ERR)
40034 RP%= 0' "REDIRECT POSITION"
40040 DR$= ":1": GOSUB 30300' DEFAULT DRIVE FOR CMDLINE/SH. EDIT HERE IF SPECIFYING DRIVE FROM
ARGV$()
40050 EO%= 0' "EOF" FLAG (FOR FUTURE DEVELOPMENT)
40099 '
    WHILE (!EOF) --

40100 IF EO% THEN 49000
```

```
40105 CL$= "": CN$= "": FOR X9%= 0 TO 3: F$(X9%)= "": NEXT: CN%= 0' RE-INIT COMMAND LINE, "COMMAND
NAMED" FLAG
40110 GOSUB 30000' GET-SH-LINE( VA$(0))
40199 '
          WHILE TK$= NEXT-TOKEN():

40200 GOSUB 5600: IF NOT SY%(12,0) THEN 48000' NO MORE
40210 TK$= VR$(0)' TOKEN$
40220 GOSUB 30500' PARSE-TOKEN( TK$);
40230 GOTO 40200
47998 '
     DONE:

48000 '
48099 '
     OUTPUT CMDLINE/SH

48100 OPEN "O", 1, CS$
48110 FOR L%= 0 TO 3: PRINT #1, F$(L%): NEXT
48190 CLOSE'

     RUN THE PROGRAM

48200 RUN CN$
```

The initialization code (Step 1) is from lines 40000..40105. Although the comments imply a "while loop" between lines 40100 and the non-existent line 49000 ("WHILE (!EOF)"), they're just there for future plans where I hoped to implement multiline inputs and pipes. As it stands, SH/BAS just runs through the main program once and "dispatches" the command line with a RUN statement. Most of the lines are commented; the "GOSUB 30300" in line 40400 simply appends a drive spec of ":1" to the default data filename of "CMDLINE/SH". I had planned to make SH/BAS "configurable" by letting it read its own command line.
Next, Step 2, we prompt the user to input a command line (in 40110: "GOSUB 30000"). The code for "GetSHline()" at lines 30000..30090 looks like this:

```
29999 '
     GET-SH-LINE()

     FROM KEYBOARD IN THIS VERSION, 1 LINE ONLY

30000 PRINT PT$; " ";: LINE INPUT VA$(0): IF VA$(0) = "" THEN 30000
30010 GOSUB 5160' INIT-SCAN-STRING()
30090 RETURN
```

We repeatedly prompt (using PT$) for a line of input until we get some non-empty text. Once we have, we "InitScanString()" with GOSUB 5160. This subroutine sets up the "String Scan Vectors" stack so that we perform recursive string scans on the input.

Step 3 is the fun part, where we actually parse the command line and assign its parts to the appropriate categories:

a) A program name to RUN;
b) A standard input filename or device, if any;
c) A standard output filename or device, if any;
d) A standard error filename or device, if any; and
e) A command line with arguments (flags, options, etc).

That happens during a "while loop" in lines 40200..49230. As pseudocode, it looks like this:

```
while (Token$= NextToken())
    ParseToken( Token$);
```

That looks easy. Here it is in BASIC:

```
40199 '
        WHILE TK$= NEXT-TOKEN():

40200 GOSUB 5600: IF NOT SY%(12,0) THEN 48000' NO MORE
40210 TK$= VR$(0)' TOKEN$
40220 GOSUB 30500' PARSE-TOKEN( TK$);
40230 GOTO 40200
```

"NextToken()" translates to GOSUB 5600, and SY%(12,0) is a flag which returns "True" (-1) if the latest String Scan operation was successful. Here's NextToken() in pseudocode, along with SkipWS():

```
NextToken() {
/* Lines 5600..5680                                                    */
    SkipWS();
    /* "GOSUB 5500"      skip any whitespace                  5600a   */
    if (MoreToScan) {
    /* "IF NOT SY%(11,0) THEN 5680"                           5600b   */
        Label BasicToken():                          /* line  5610    */
        SkipWhile(Whitespace$);
        /* "GOSUB 5120"                                       5610a   */
        VR$(0)= GatherUntil();
        /* "GOSUB 5050"gather up all non-whitespace characters 5610b   */
        SaveStringScanArgsAndReturns();
        /* "GOSUB 5100"                                       5610c   */
        if (ScanSuccessful) {
```

```
        /* "IF SY%(12,0)" ie, if we "gathered" some useful text                5610d  */
            Return$= MID$( CommandLine$, Start, Length);
            /* "VR$(0)= MID$( VA$(0), SY%(13,0), SY%(15,0))"                  5610e  */
            }
    UpdateScanCursor();
    /* "GOSUB 5140"                                                           5680   */
    }

SkipWS() {
    PopStringToScan();
    /* "GOSUB 5120"                                                                  */
    SkipWhile( Whitespace$);
    /* "VA$(1)= SY$(13,0)...GOSUB 5020"                                              */
    SaveStringScanArgsAndReturns();
    /* "GOSUB 5100"                                                                  */
    }
```

and in BASIC:

```
5499 '
    SKIP-WS()

5500 VA$(1)= SY$(13,0): GOSUB 5120: GOSUB 5020: GOSUB 5100: RETURN'

    NEXT-TOKEN()

5600 GOSUB 5500: IF NOT SY%(11,0) THEN 5680'

    BASIC GET-TOKEN()

5610 VR$(0)= "": SY%(12,0)= 0: GOSUB 5120: GOSUB 5050: GOSUB 5100: IF SY%(12,0) THEN VR$(0)= MID$( VA$(0),
SY%(13,0), SY%(15,0))
5680 GOSUB 5140: RETURN
```

Those "GOSUB 5xxx"s are doing a lot of the heavy lifting. They're part of the "String Scanning" subroutine package I described on page 31 of the March 2023 TRS8bit.
Speaking of "heavy lifting", it's ParseToken(Token$) that does some of the hard work—of course, by delegating. It checks each "token", a whitespace-separated substring, for a "<" to see if it's input redirection, for a ">" to see if it's output redirection, or otherwise if it's the command name (like "MYPROG") or else arguments (like "-FLAGS").

```
ParseToken( Token) {
    if (RedirectPosition= INSTR( Token, "<")) {
    /* "RP%= INSTR( TK$, "<"): IF RP% <> 1 THEN 30600"
                                if there's a "<" in the token... do input redirection      30500   */
        If (InputRedirect())
        /* "GOSUB 20000"                                                                  30505   */
            return;
            /* "IF VR%(0) THEN RETURN ELSE 31000"                                         30510   */
        }
    else if (RedirectPosition= INSTR( Token, ">")) {
    /* if there's a ">" in the token..do output redirection                              30600a  */
        if (!RedirectPosition < 1 OR RedirectPosition > 2)) {
        /* ie, if RedirectPosition == 1 OR 2:                                                 */
        /*  "IF RP% < 1 OR RP% > 2 THEN 30700"                                           30600b  */
            If (OutputRedirect())
            /* GOSUB 21000"                                                               30605   */
                return;
                /* "IF VR%(0) THEN RETURN ELSE 31000:                                     30610   */
        }
    else if (!CommandNamed) {
    /* if we haven't named the command yet (ie, this is the 1st non-redirect token            */
    /* "IF CN% THEN 31000"                                                                30700   */
        PrettyUp();
        /* "GOSUB 25000"    ie, prepare the Command Name forCMDLINE/SH                    30705   */
        CommandNamed= TRUE;
        /* "CN% = -1"       we have now named a command, so everything else
                            is an "argument" (option, flag, etc)                          30710   */
        }
    BuildCommandLine();
    /* "GOSUB 26000"                                                                      31000   */
    }
```

Here it is in BASIC:

```
30499 '
    PARSE-TOKEN()

30500 RP%= INSTR( TK$, "<"): IF RP% <> 1 THEN 30600'
    INPUT REDIRECTION:
30505 GOSUB 20000' INPUT-REDIRECT()
30510 IF VR%(0) THEN RETURN ELSE 31000
30600 RP%= INSTR( TK$, ">"): IF RP% <1 OR RP% > 2 THEN 30700'
    OUTPUT REDIRECTION:
30605 GOSUB 21000' OUTPUT-REDIRECT()
30610 IF VR%(0) THEN RETURN ELSE 31000
```

```
30700 IF CN% THEN 31000'
    NAME COMMAND:
30705 GOSUB 25000' PRETTY-UP()
30710 CN%= -1
31000 GOSUB 26000' BUILD-COMMAND-LINE()
31090 RETURN
```

So: if the token begins with a "<", we use InputRedirect() to check if it's redirection from a device or from a disk file:

```
InputRedirect() {
    InitFile$( SlotNumber= 1);
    /* "SN%= 1: GOSUB 39900"     Set F$[SlotNumber]= ""                    20000   */
    if (INSTR( Token, "/DEV/") > 0) {
    /* "IF INSTR( TK$, "/DEV/") > 0"                                       20010a  */
        Device();
        /* "GOSUB 10000:"    Apply any appropriate device numbers (-1..-3)  20010b  */
        }
    else {
        File$[SlotNumber]= MID$( Token, 2);
        /* "F$(SN%)= MID$( TK$, 2)"    Input filename= everything after the ">"  20020   */
    }

Device() {
    if ((INSTR( Token, "CAS") > 0) OR (INSTR( Token, "TAP") > 0)
    /* "IF NOT (INSTR( TK$, "CAS") > 0 OR INSTR( TK$, "TAP") > 0) THEN 10100"
                Look for any substring that might indicate "CASSETTE" or "TAPE"  10010   */
        File$[SlotNumber]= GetCassetteNum());
    else if ((INSTR( Token, "PR") > 0 OR INSTR( Token, "LP") > 0) {
    /* "IF NOT (INSTR( TK$, "PR") > 0 OR INSTR( TK$, "LP") > 0) THEN 10200"
                Look for any substring that might indicate "PRINTER" or "LPR"  10100   */
        File$[SlotNumber]= Printer$;
        /* "F$(SN%)= STR$( FN PR%( 0))"                                    10110a  */
        return SUCCESS;
        /* "VR%(0)= -1"                                                    10110b  */
        }
    else if (INSTR( Token, "NUL") > 0) {
    /* "IF NOT INSTR( TK$, "NUL") > 0 THEN 10300"                          10200   */
        File$[SlotNumber]= "0";
        /* 'F$(SN%)= "0":'                                                 10210a  */
        Return SUCCESS;
        /* "VR%(0)= -1"                                                    10210b  */
        }
}
```

```
GetCassetteNum() {
    SaveStringScanVectors();
    /* "GOSUB 5200"      push the current String Scanning state onto Scan Stack    11000   */
    InitScanString( Token);
    /* "VA$(0)= TK$: GOSU 5160"
                         set up to scan Token$ (instead of Command Line)          11010   */
    SkipUntil( "-012");
    GatherWhile( "-012");
    /* "VA$(1)= "-012": GOSUB 5030: GOSUB 5040"
                         Skip everything until we match any of "-", "0", "1" or "2";
                         then gather a string as long as we match any of those    11020   */
    CassetteNo= ABS( VAL (ReturnGathered()));
    /* "GOSUB 5400: C%= ABS( VAL( VR$(0)))"
                         Assign the matched substring to VR$(0);
                             the absolute value of that to C% (1 or 2)     11027..11030a   */
    RestoreStringScanVectors();
    /* "GOSUB 5250"                                                                11030b  */
    if ( CassetteNo == 1..2) {
    /* "VR%(0)= C% > 0 AND C% < 3: IF NOT VR%(0) THEN 11090"                       11040   */
        report SUCCESS;
        return (-4 - CassetteNo + 1);
            /* "VR$(0)= STR$( FN C1%( 0) – C% + 1"
                CassetteNo        Return Value
                    1                 -4
                    2                 -5                                          11050   */
        }
    else
        report FAIL;
    }
```

InputRedirect() is pretty simple: if what follows the "<" is a "/DEV/", we call on Device() to figure out what device we're redirecting to or from—Device() is used for both Input and Output Redirection. Device() is fairly straightforward as well, although if the device happens to be a cassette, it has to figure out which one, so it calls on GetCassetteNum(). Notice that Device() is somewhat forgiving when we specify a cassette number: for example, we can refer to "CASSETTE1" or "TAPE-1" or "CAS01" or even "TAP-00001" and it will correctly return the device number, in this case -4.
Here is the source in BASIC:

```
19999 '
    INPUT-REDIRECT()

20000 SN%= 1: GOSUB 39900' INIT-F$()
20010 IF INSTR( TK$, "/DEV/") > 0 GOSUB 10000: RETURN' DEVICE()
```

```
20020 F$(SN%)= MID$( TK$, 2)
20090 VR%(0)= -1: RETURN

9999 '
     DEVICE()

10000 VR%(0)= 0
10010 IF NOT (INSTR( TK$, "CAS") > 0 OR INSTR( TK$, "TAP") > 0) THEN 10100'
     TAPE:
10020 GOSUB 11000: IF VR%(0) THEN F$(SN%)= VR$(0)' GET-CASSETTE-NUM()
10090 RETURN'
     PRINTER:
10100 IF NOT( INSTR( TK$, "PR") > 0 OR INSTR( TK$, "LP") > 0) THEN 10200
10110 F$(SN%)= STR$( FNPR%(0)): VR%(0)= -1
10190 RETURN'
     NULL (BIT BUCKET):
10200 IF NOT INSTR( TK$, "NUL") > 0 THEN 10300
10210 F$(SN%)= "0": VR%(0)= -1
10290 RETURN
10300 ' ANY FUTURE DEVICES:
10990 RETURN'

     GET-CASSETTE-NUM()

11000 GOSUB 5200' SAVE-SSVS()
11010 VA$(0)= TK$: GOSUB 5160' INIT-SCAN-STRING()
11020 VA$(1)= "-012": GOSUB 5030: GOSUB 5040' SKIP-UNTIL( "-012"); GATHER-WHILE( "-012")
11027 GOSUB 5400' RETURN-GATHERED()
11030 C%= ABS( VAL( VR$(0))): GOSUB 5250' RESTORE-SSVS()
11040 VR%(0)= C% > 0 AND C% < 3: IF NOT VR%(0) THEN 11090'
     TRANSLATE CASSETTE #S:
11050 VR$(0)= STR$( FNC1%( 0) - C% + 1)
11090 RETURN
```

OutputRedirect() is similar to InputRedirect(), but it has to do some more work. It's possible to *append* to an output file, by specifying something like "`>>GROWING/FIL`". Output standard fiile numbers can also be redirected, so that we can save error messages in a file or direct them to another device, as in "`3>ERROR/FIL`" or "`3>/DEV/LPR`" (stdin is 1, stdout is 2, stderr is 3). Finally, we can redirect standard file numbers to *other* standard file numbers, so that output to stderr can be mixed in with output to stdout, as in "`3>2`". Here's OutputRedirect() in pseudocode:

```
OutputRedirect() {
    StreamNumber= 2;
    /* "SN%= 2"            Default "stream number" ie, std file # (1..3)                21000   */
    Append$= "";
    /* 'AP$= "":'          If file to be OPENed in "Append" mode, " +" will be added   21010a  */
    if (INSTR( Token, ">>") > 0)
        Append$= " +";
    /* "IF INSTR( TK$, ">>") > 0 THEN AP$= " +"                                        21010b  */
    if (RedirectPosition > 1) {
    /* "IF RP% < 2 THEN 21100"   "RedirectPosition" is char position in Token          21020   */
        /* "Stream Redirect?" "stream" is std file # (1..3). Check for something like "3>2".   */
        Stream= VAL( LEFT$( Token, RedirectPosition – 1));
        /* "ST%= VAL( LEFT$( TK$, RP% - 1))"   try to get a stream #                   21050   */
        if (Stream is 2 or 3)
        /* "IF ST% > 1 AND ST% < 4 THEN"                                               21060a  */
            StreamNumber= Stream;
            /* "SN%= ST%"                                                              21060b  */
        else
            return;
            /* "ELSE RETURN"    not a valid std assignment                             21060c  */
        }
    /* Get Output Token  */
    SaveStringScanVectors();
    /* "GOSUB 5200"      push the current String Scanning state onto Scan Stack     21100   */
    TokenSaved= Token;
    /* "TS$= TK$:"                                                                 2110a   */
    InitScanString( Token);
    /* "VA$(0)= TK$: GOSUB 5160"     Prepare to scan Token                         2110b   */
    SkipUntil( ">"); SkipWhile( ">");
    /* 'VA$(1)= ">": GOSUB 5030: GOSUB 5020' find any occurrence of ">",
                                          then skip past all occurrences of ">"     21120   */
    GatherUntil( Whitespace);
    /* "VA$(1)= SY$(13,0): GOSUB 5050"                                             21130   */
    Token= ReturnGathered();
    /*   "GOSUB 5400"                                                               21137..
        "TK$= VR$(0): SC%= VR%(0)"   note: "SC%" saves Success/Fail report here    21140a  */
    RestoreStringScanVectors();
    /* "GOSUB 5250"                                                                21140b  */
    if (ReturnGathered() was not successful in returning a token) {
    /* "IF NOT SC% THEN"                                                           21150a  */
        Token= TokenSaved;
        Report FAIL;
        return;
        }
        /* "TK$= TS$: VR%(0)= 0: RETURN"     simply return the previous token      21150b  */
    RedirectStream= VAL( LEFT$( Token, 1));
```

```
    /* "RS%= VAL( LEFT$( TK$, 1))          the stream (1..3) to which we are
                                           "forcing" output, as in
                                               "3>ERRFILE/TXT" or
                                               "3>2"                          21160   */
    if (RedirectStream is 2 or 3) {
    /* "IF RS% > 1 AND RS% < 4 THEN"                                          21200a  */
        File$[StreamNumber]= STR$( RedirectStream * 100);
        /* "F$(SN%)= STR$( RS% * 100):"    "200" or "300"                     21200b  */
        Report SUCCESS;
        return;
        }
        /* "VR%(0)= -1: RETURN"                                               21200c  */
    if (INSTR( Token, "/DEV/")) {
    /* "IF INSTR( TK   $, "/DEV/") THEN"   is it a device?                    21300a  */
        Device();
        /* "GOSUB 10000:"                                                     21300b  */
        if (Device assignment successful))
            return;
        /* "IF VR%(0) THEN RETURN ELSE"                                       21300c  */
        else {
            Token= TokenSaved;
            return;
            /* "TK$= TS$: RETURN"                                             21300d  */
        {
    /* Otherwise, it's a file:     */
    File$[StreamNumber]= Token + Append$;
    /* "F$(SN%)= TK$ + AP$:"                                                  21400a  */
    }
```

and in BASIC:

```
20999 '
    OUTPUT-REDIRECT()

21000 SN%= 2' DEFAULT STREAM #

    1. APPEND?
21010 AP$= "": IF INSTR( TK$, ">>") > 0 THEN AP$= " +"
21020 IF RP% < 2 THE N21100'
    2. STREAM REDIRECT?
21050 ST%=VAL( LEFT$( TK$, RP% - 1))
21060 IF ST% > 1 AND ST% < 4 THEN SN%= ST% ELSE RETURN'
    3. GET OUTPUT TOKEN
21100 GOSUB 5200' SAVE-SSVS()
21110 TS$= TK$: VA$(0)= TK$: GOSUB 5160' INIT-SCAN-STRING()
```

```
21120 VA$(1)= ">": GOSUB 5030: GOSUB 5020' SKIP-UNTIL(), SKIP-WHILE( ">")
21130 VA$(1)= SY$(13,0): GOSUB 5050' GATHER-UNTIL( " ")
21137 GOSUB 5400' RETURN-GATHERED()
21140 TK$= VR$(0): SC%= VR%(0): GOSUB 5250' RESTORE-SSVS()
    4. ANY OUTPUT TOKEN?
21150 IF NOT SC% THEN TK$= TS$: VR%(0)= 0: RETURN
21160 RS%= VAL( LEFT$( TK$, 1))' "REDIRECT STREAM"
    5. OUTPUT TOKEN TYPE:
    5A. STREAM REDIRECT
21200 IF RS% > 1 AND RS% < 4 THEN F$(SN%)= STR$( RS% * 100): VR%(0)= -1: RETURN'
    5B. DEVICE
21300 IF INSTR( TK$, "/DEV/") THEN GOSUB 10000: IF VR%(0) THEN RETURN ELSE TK$= TS$: RETURN'
    5C. OUTPUT FILE
21400 F$(SN%)= TK$ + AP$: VR%(0)= -1: RETURN
```

Once SH/BAS has dealt with all the different kinds of input and output redirection, it can assume that all the other tokens are either the program name or "arguments"—options, flags, etc. Why did I have SH/BAS deal with redirection first? Based on what I read about the original Bourne shell in books like *The Unix Programming Environment*, it's acceptable for a command line to have I/O redirection *anywhere* on the command line, so not only can you write

```
COPY <FILE/SRC >FILE/DST
```

but also any alternate form like

```
<FILE/SRC COPY >FILE/DST
>FILE/DST <FILE/SRC COPY
>FILE/DST COPY <FILE/SRC
```

So if we "strip away" the redirection, the program ("COPY") becomes the *first* token we find on the command line. That's why ParseToken(), listed above, takes care of input and output redirection, and then does something like this:

```
else if (!CommandNamed) {
/* if we haven't named the command yet (ie, this is the 1st non-redirect token          */
    PrettyUp();
    CommandNamed= TRUE;
    }
```

The first time ParseToken() is called, CommandNamed is false/0, and we will use PrettyUp() to clean up the token, append "/BAS" if necessary, and save it as the name of the program that SH/BAS will eventually RUN.

```
PrettyUp() {
    CommandName= Token;
    /* "CN$= TK$"                          We'll eventually RUN CN$              25000  */
    if (INSTR( Token, "/") < 1) {
    /* "IF INSTR( TK$, "/") > 0 THEN RETURN"   Is it already formatted as a filespec?  25010  */
        SaveDriveSpec();
        /* "GOSUB 25100"                   ie if specified as "PROGNAME:3"       25020  */
        CommandName= CommandName + "/BAS" + DriveSpec;
        /* 'CN$= CN$ + "/BAS" + DS$'                                             25030  */
        }
    }

SaveDriveSpec() {
    DriveSpec= "";
    /* 'DS$= "" '                                                               25100  */
    DrivespecPosition= INSTR( Token, ":");
    /* 'DP%= INSTR( TK$, ":"):'            Try to find a ":" in Token            25110a */
    if (DrivespecPosition > 0) {
    /* "IF DP% < 1 RETURN"                                                      25110b */
        DriveSpec= MID$( Token, DrivespecPosition, 2);
        /* "DS$= MID$( TK$, DP%, 2):"                                           25120a */
        CommandName= LEFT$( Token, DrivespecPosition – 1);
        /* "CN$= LEFT$( TK    $, DP% - 1)"                                      25120b */
        }
    }
```

In BASIC:

```
24999 '
    PRETTY-UP()

25000 CN$= TK$
25010 IF INSTR( TK$,"/") > 0 RETURN'
    FIRST, PARSE OUT ANY DRIVESPEC:
25020 GOSUB 25100' SAVE-DRIVE-SPEC()
25030 CN$= CN$ + "/BAS" + DS$
25090 RETURN

'
    SAVE-DRIVE-SPEC()

25100 DS$= ""
25110 DP%= INSTR( TK$, ":"): IF DP% < 1 RETURN
25120 DS$= MID$( TK$, DP%, 2): CN$= LEFT$( TK$, DP% - 1)
25190 RETURN
```

That step was simply to save the program name as a RUNnable BASIC filespec. Regardless of whether it was a program name or just another token, we need to append it to the edited command line that will be saved as line 1 in CMDLINE/SH. (Remember that a command line entered as

```
MYPROG <INFILE >OUTFILE –FLAGS 3>ERRFILE
```

would be stored in CMDLINE/SH as

```
MYPROG –FLAGS
INFILE
OUTFILE
ERRFILE
```

From this point on in ParseToken(), having stripped out the redirection, we're only building line 1, using BuildCommandLine():

```
BuildCommandLine() {
    Spacer= "";
    /* 'SP$= "" '              start with NO leading spaces          26000a  */
    if (LEN( File$[0]) <> 0)
    /* "IF LEN( F$(0) <> 0) THEN"                                    26000b  */
        Spacer= " ";
        /* 'SP$= " " '                                               26000c  */
    File$[0]= File$[0] + Spacer + Token;
    /* "F$(0)= F$(0) + SP$ + TK$"                                    26010   */
    }
```

and in BASIC:

```
'
    BUILD-COMMAND-LINE()

26000 SP$= "": IF LEN( F$(0)) <> 0 THEN SP$= " "
26010 F$(0)= F$(0) + SP$ + TK$
26090 RETURN
```

We don't want line 1 of CMDLINE/SH ("File$[0]") to begin with a blank: it should start with the program name ("MYPROG"). So each time through BuildCommandLine(), Spacer/SP$ is reset to an empty string "", and then we test how long line 1/File$[0] is so far. If we're just beginning, File$[0]/F$(0) will be empty, so the Spacer/SP$ remains empty as well, and only Token/TK$ gets

added to line 1. When or if we call BuildCommandLine() again, File$[0] will *not* be an empty string, so we expand Spacer to be 1 space, separating each element in line 1 of CMDLINE/SH.

So ParseToken() keeps getting called (from the loop at 40200..40230 listed at the beginning of this article) until there are no more tokens. We're done scanning the command line that the user input. What now?

```
48100 OPEN"O", 1, CS$
48110 FOR L%= 0 TO 3: PRINT#1, F$(L%): NEXT
48190 CLOSE
```

A simple FOR..NEXT loop writes out to CMDLINE/SH each element of File$/F$ as a single line. Finally, we get to

```
'
    3. RUN THE PROGRAM

48200 RUN CN$
```

and SH/BAS transfers to the program named in CN$. If that program was built on the SKELETON/BAS framework, it will end with

```
    RUN "SH/BAS"
```

to return to this shell for further commands.
Here's a few sample command lines that SH/BAS can interpret (assuming we've written a COPY/BAS program as listed at the beginning of this article):

```
COPY
```
(Copy from the keyboard to the screen.)

```
>/DEV/LPR COPY
```
(Copy to the printer from the keyboard. Hey! A typewriter! Remember those?)

```
COPY >FILE/DAT:2
```
(Type input to a file.)

```
>>FILE/DAT COPY
```
(Append some more text to FILE/DAT.)

>/DEV/PRINTER COPY <FILE/DAT 3>ERRFILE:3
(Print the contents of FILE/DAT; if there are any error messages, log them in ERRFILE .)

<FILE/DAT >CPYANDER/FIL:2 COPY 3>2
(Copy FILE/DAT to CPYANDER/FIL; if there are any error messages, shove them in that file too.)

You can find copies of all the listings, support files, and .DMK diskette images mentioned in these articles (June 2023, Sept 2023, Mar 2024, June 2024, Dec 2024, and this final article, as well as the String Scanning routines incorporated into SKELETON/BAS in the Mar 2023 issue) using GitHub: look for the "Software_Tools_In_BASIC_Final" repository under VinnieMan1. The files include:

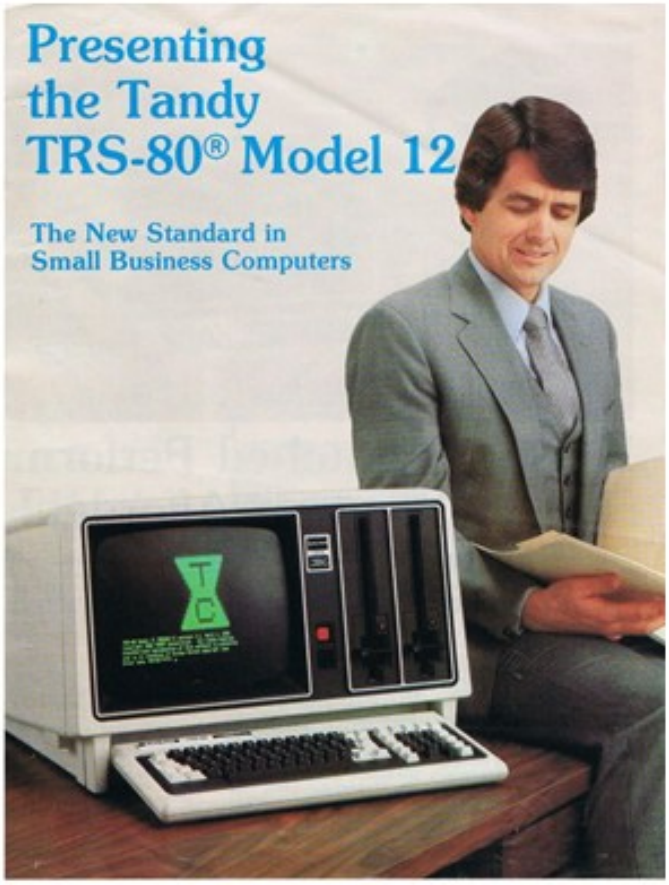
Skeleton in PseudoCode.txt
SKELETON Basic Listing.txt
Skeleton Usage Guide.txt
PseudoCode Aliases.txt
SKELETON complete with Append Mode.dmk
SH BASIC Listing.txt
Software Tools Ch 1.dmk

You can find me in all the TRS-80 related groups on social media, as well as the Tandy and trs80gp Discords. Just look for "Vince Otten" or "VinnieMan", or follow the link

https://github.com/VinnieMan1/Software_Tools_In_BASIC_Final

IN MAV'S HISTORY WORKSHOP

What *Was* The TRS-80 Model 12?



by Ian Mavric

An interesting mid-performance computer in Radio Shack's line of 8in 'business' systems, the Model 12 is sometimes overlooked and misunderstood. Occasionally when they come up for sale people ask what it is and what was it used for, this article looks as the TRS-80 Model 12 Microcomputer system, a small business Z80 power house. TL/DR: It replaced the TRS-80 Model II.

HISTORY: THE TRS-80 MODEL II STARTED IT ALL

By 1983 the TRS-80 Model II was getting stale. Released in 1979 with good performance and expandability, the Model II was popular with small and medium businesses due to it's low cost compared to a mini computer, reliability and good range of software. Being a 64K 80-column screen Z80 system it ran CP/M like a treat. The 400K 8in disk drive held a decent amount of data, three more of these drives could be added, or an 8 megabyte hard drive. In '79 or '80 if your business had one of these your company was at the forefront of datamation. You could even network them by the early 80s in an arrangement known as ARCnet.

In '82 the platform was expanded with the Model 16 multi-user system. Using a Motorola 68000 CPU on an expansion board, some extra memory, and connected via RS232-C serial to smart terminals a small office multi-user network could be set up based around the newly-released Xenix operating system. But if you looked closely, under it's stylish white paint-job, the Model 16 is basically a warmed-over Model II with better disk drives, 68K and Ram cards. It also did a great job and for a while was the most popular Xenix system on the market, but that's a story for another time.

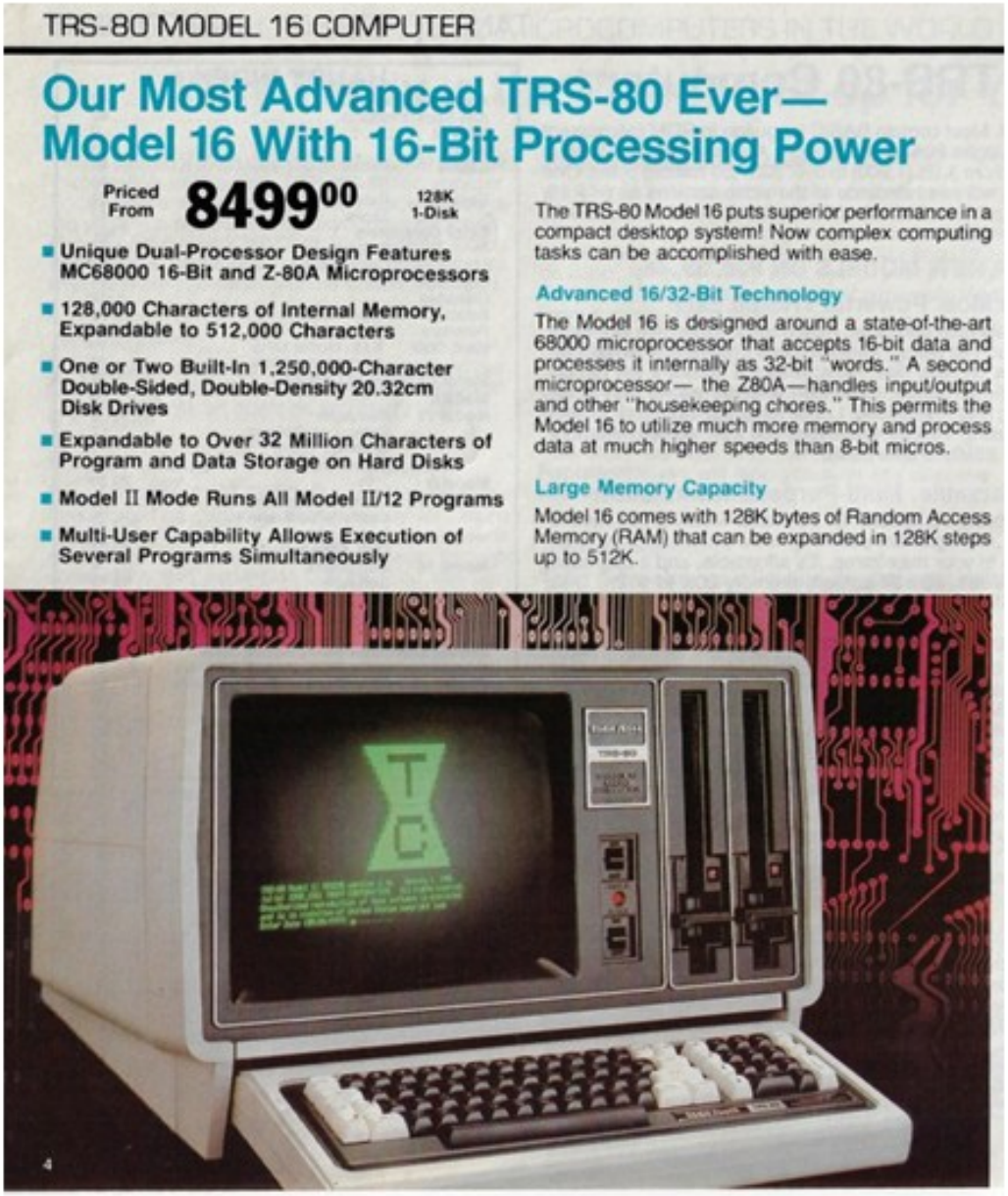


Photo: Model 16 is a hot-rodded Model II, was expensive to produce and expensive to buy

Lessons learned from the Model II/16 were reflected up on and implemented in it's replacement. Enthusiasts separate TRS-80 8in systems into two distinct 8in platforms: the 'A' series comprising of the Model II and 16 (sometimes called the 16A), and the 'B' series comprising the Model 12, 16B and 6000. With exception of the

video/keyboard controller card, and the disk drives in the case of the 16A, the B series is a complete ground-up re-design of the A series.

I speculate that Radio Shack did some surveying of Model II users and found to nobody's surprise that most people used their machines exactly as delivered, only a small percentage of users addition additional cards. Those who have looked inside a Model II (or 16A) will have noticed there is no motherboard, simply a slotted back-plane board with card connectors. Although this was not called the "card cage" at the time, we will call it that from now on, it becomes more important in the B-series.

A Model II comprises of 4 cards: Z80A CPU/SIO/PIO card, 64K Ram card, floppy controller card, and keyboard/video card. A Model 16A has the same but in addition a 68000 CPU card and one or more 128K Ram cards. Always looking for ways to make machines cheaper and easier to produce, and more reliable, the B series introduces a motherboard with the basics on it: Z80A CPU, 64-80K Ram, SIO/PIO interfaces, and floppy disk interface. A single slot on the motherboard connects to a card cage in the 16B and 6000, for it's the additional 68000 and memory cards. So what has this to do with the Model 12?

MODEL 12: THE MODEL II REPLACEMENT FOR Z80 USERS



Photo: Model 12 came in lower price than the Model II and much lower than the Model 16

If you leave the card cage out of the Model 16B and plug the keyboard/video card directly into the motherboard, you have a Model 12. Yes, the motherboard in the Model 12, 16B and 6000 is essentially the same motherboard. Smart thinking Radio Shack, design the board once, and use it in three systems. There is an important difference: on the Model 12 the 64K Ram bank and 16K ram bank are populated (total 80K Ram), but on the 16B and 6000 only the 64K bank is populated. People are sometimes disheartened when they open the back of a Model 12 and find no card cage, but this is how they are designed. It allowed Radio Shack to market the

Model 12 at a lower price point than the 16B, but would still run all the great Model II software which was very popular. If you didn't need to run anything that required the 68000 CPU of the 16B, or hard drives, then the Model 12, like the Model II before it, made a good choice for office datamation. For a lot of users during this period if a hard drive was prohibitively expensive, dual drives and a printer was a standard and capable office system.

The 80K Ram is interesting in that it was added so that the new TRSDOS 4.2.x operating system would run properly. 4.2.x takes up more Ram, so in order to leave the same amount of user Ram for programs like VisiCalc, Scripsit, and Accounts Receivable some additional Ram was added to the motherboard. This Ram is not present on the 16B and 6000; the sockets are there, they are just not populated. Lastly if you added a hard disk to a Model 12 you *must* run 4.2.x and the hard disk drivers are an additional overhead, they need to go somewhere, hence the Model 12 being an 80K system.

OK BUT NOW I WANT TO ADD A HARD DRIVE TO MY MODEL 12

Of course, Radio Shack was always there to sell solutions for those who needed them and the Model 12, like it's predecessor the Model II, could be outfitted with a hard disk drive. This is where things got a little expensive. On the 16B, if you owned the dual 8in drive version of the 16B you bought your 15Mb external disk drive from Radio Shack, and a hard disk adapter kit, which comprised a Hard Disk Adapter card, cables, and operating system. To do the same

thing on the Model 12 you needed to install a card-cage kit, which was an additional cost.



Photo: Card Cage is the key to expansion. Included with the 16B/6000 but missing in the 12.

Once the card cage was installed the keyboard/video card plugs into it, as well as the Hard Disk Adapter (HDA). Externally a hard drive is connected and the system runs under TRSDOS 4.2.x which booted from the hard drive. It was a neat setup and I have seen a few of these expanded hard-disk based Model 12s. Looking at the card cage with only two cards in it reminds me of a lot of work and expense to add a hard drive, however those who used those systems back in the day would have thought it was worth it.



Photo: External hard drive for TRS-80s. Notice cable kit (26-4157) for $495? That included the Type-II HDA which is why it was so expensive. The $45 Tandy 2000 cable kit, by contrast, was just a cable.

HOW FAR CAN THE MODEL 12 BE EXPANDED?

Since the cabinet, CRT, motherboard and disk drives are the same as the 16B, the Model 12 can be expanded to the full performance of the 16B once the card cage is fitted. This was accomplished with the Model II/12-to-16 upgrade kit (cat. no. 26-6010). It's an expensive and rather long-winded way of achieving the same result as a 16B, but Radio Shack would have rationalised end-users could buy the Model 12 as a cost-effective good business workhorse, then expand the system as business grew, into a more powerful and/or multi-user system.



Photo: from RSC-10 the Model II/12-to-16 upgrade kit (26-6010) was $1499 and that was even before installation.

CONCLUSION

The Model 12 built on the good reputation of the Model II, modernising the look of the machine, reducing production costs while increasing durability, and at the same time maintaining 100%

software compatibility. Ergonomics were improved with an office-oriented P31 green screen CRT, and the keyboard was re-designed for long hours of typing in the offices which were eventually destined to be installed in.

Finally the disk drives were improved from the old 400K single-sided 8in disk drive units in the Model II were upgraded to a modern 1.2Mb double-sided double-density drive.

These collectable machines come up on eBay with about the same regularity as the Model 16B and 6000, though there seems to be far more Model IIs available at any given time. The case on 12s seems to go yellow with age, and like all 8in TRS-80 people need to factor in re-foaming the keyboard and possible disk drive work to get the machine up and running.

Shameless self promotion plug: F48, the version of the FreHD hard drive emulator for 8in TRS-80s, works great on the Model 12, as long as you have a card cage and Type-II HDA.

Ian Mavric

ianm@trs-80.com



Ian Mavric is an IT Specialist who also restores and collects TRS-80's and classic cars. He lives with his wife and kids in Melbourne, Australia.



The *BEST* in TRS-80s
Call The Right Stuff

Ask for Ian
The number is +61 416 184 893

That's The Right Stuff
And he's in Melbourne

http://members.iinet.net.au/~ianmav/trs80/





TRS8BIT IS SUPPORTED BY YOU AND PRODUCED
AT WWW.TRS-80.ORG.UK

WHERE 8 BITS AND 16K STILL RULE OK

TRS8BIT

PRODUCED AT WWW.TRS-80.ORG.UK

JUNE 2025 - VOLUME 19 ISSUE 2

PAGE 03 2025 COMPETITION - IMPOSSIPUZZEL25
ONCE AGAIN, GREAT PRIZE SO GET YOUR ENTRIES IN QUICK

PAGE 05 TA25 NEWS - IAN MAVRIC
PLANNING THE 2025 TANDY ASSEMBLY

PAGE 11 DISK FILES (PART 2) - JIM NORRIS
A RARE FIND AMONGST A BATCH OF OLD TANDY PAPERS

PAGE 23 STAR TREK - UPGRADE
ALAN WALLACE

PAGE 25 APRIL BASQUE TOURNAMENT
EGOITZ CAMPO
RESULTS FROM THE LATEST TRS-80 BASQUE COMPETITION

PAGE 28 POWER OF THE ROM
VINCE OTTEN
PART 2 OF HIS SERIES

PAGE 49 IN MAV'S HISTORY WORKSHOP
IAN MAVRIC
WHAT WAS THE SECOND COLOR COMPUTER 1?

PAGE 60 I-CHING
E.T. FONEHUME
EVER WONDERED IF IT WORKS? NOW'S YOUR CHANCE!

PAGE 80 OUR 2ND COMPETITION
HAVE YOU ANY IDEAS, WHAT WOULD YOU LIKE TO SEE?
GET YOUR SUGGESTIONS OFF TO GEORGE ASAP

PAGE 85 THE WAY WE WERE
JUST A QUICK LOOK BACK TO HOW SOME OF THE THINGS WERE, 'IN THE DAY'.
DID ANYONE HAVE A GO WITH THE 'TINY PASCAL'?

PAGE 89 'AMAZED'
HAVE A BREAK AND HAVE A GO - TRY FOR 10 SECONDS!

PAGE 91 SURFIN' THE NET
AN UPDATED CP/M OPTION FOR A MODEL 1

LATEST UPGRADE NOW AVAILABLE
VERSION 2.5.5

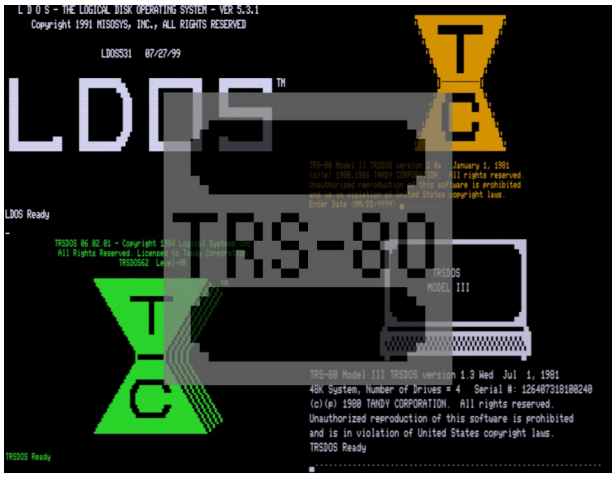
DON'T HAVE A TRS-80?

CRAZY PRICES ON EBAY?

TIRED OF RIFA CAPS BLOWING?

OF FLOPPIES SHEDDING OXIDE?

OR JUST PLAIN DEAD CHIPS?



THEN WHAT YOU NEED IS TRS80GP.
THE ULTIMATE TRS-80 EMULATOR

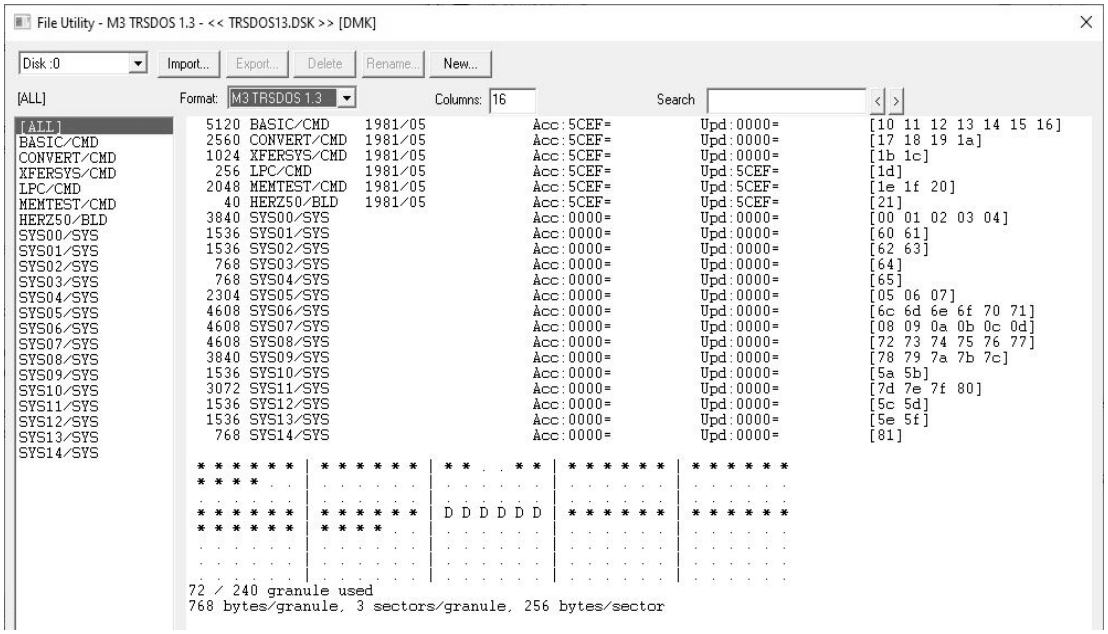
RUNS ON WINDOWS, MACOS, LINUX & RASPBERRY PI.
EMULATES MODELS 1, 2, 3, 4, 12, 16, 6000, MC-10,
COLOR COMPUTER, DT-1, VIDEOTEX AND NOW THE **TANDY 10.**

JUST LIKE A REAL TRS-80 WITH NONE OF THE HASSLE.
BUILT-IN DEBUGGER AND OTHER FEATURES, JUST GREAT FOR PROGRAMMERS.

GET IT NOW AT HTTP://48K.CA/TRS80GP.HTML

UPDATED VERSION, 2.5.5, ADDITIONALLY OFFERS :

FILE UTILITY TO EXAMINE DISK FILES; DEBUG - FILE UTILITY
CUSTOMIZABLE MENU VIA - MENU OPTION SCRIPTS
"TYPING" KEYBOARD LAYOUT BETTER FOR LEVEL 1 INPUT



TRS8BIT's IMPOSSiPUZZLE25

A U.K. quiz program called 'Countdown' has a fascinating numbers game.
You select 6 unseen numbers comprising of a group from 1 to 10 (small numbers) and the four large numbers 25, 50, 75, and 100.
You can choose how many 'small' and/or 'large' numbers you require.
Duplicates, if they occur, are allowed.
Then a 'target' number is randomised from 100 to 999

Contestants then have 30 seconds to calculate, with the 4 arithmetic operators +, -, * and /, the 'target' number using any or all of the 6 random numbers, ensuring that each number is only used once.

At no stage of the calculation can you have a negative or fractional number.

As an example:-
Let's say you have chosen 2 'large' and 4 'small' numbers.
When revealed they were 75, 100, 7, 10, 1, 10 and a 'target' number of 543

Here's one way to calculate the 'target' number:

```
(75+1)*7    = 532
(100+10)/10 =  11
532+11      = 543
```

To win the TRS8BIT IMPOSSiPUZZLE25, you need to write a program, running on a TRS-80 emulator, to take the 6 random numbers, **25,75,5,2,8,5** with the 'target' of **124** and show how to use them to calculate the 'target' number. Remember of course, it may be an impossible task. You can decide how long to cycle the program looking for an answer or, if one is not found, just declare how close you were!

The 'Trashtalkers' team will judge the entries, taking into account, running speed, programing style, language(s) used, entry date and the number of TRS-80 models your program will run on.
.
The winner will receive their choice from one of Mav's wonderful products.

Closing date for the competition is
22:00 hrs GMT on Sunday 2nd November 2025
Entries to be sent to George Phillips at george@48k.ca


ACTUAL REALITY
TRS-80 MODEL I/III/IV
DREAMLAND SOFTWARE


ROCK STAR!
TRS-80 MODEL I/III/IV
DREAMLAND SOFTWARE

MAV'S TANDY ASSEMBLY QUARTERLY COLUMN

TA25 NEWS

by Ian Mavric



Bringing you the latest news you need to know about the greatest Tandy TRS-80 show on the planet. This time I talk about plans for the 2025 show which, which is only 3 short months away.

TA25 FIRST PRESENTERS ANNOUNCED

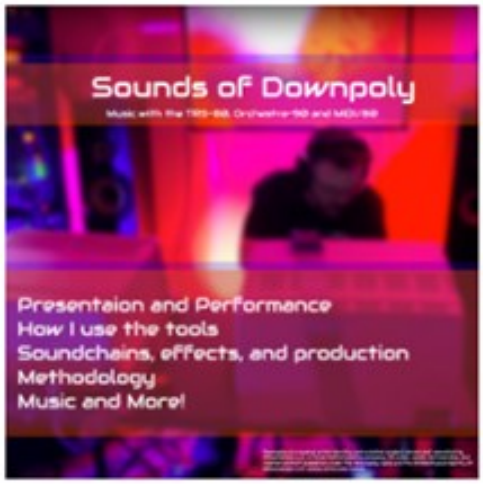
I am pleased to announce that presenters have already been signed up, so let's take a look at who:

This year we have a very special PAIR of presentations by the brilliant Joel Hilliard who has been doing a lot with music on the TRS-80 Model 4 and 4P. Firstly, Joel will be giving both a Friday Tandy Tech Track tutorial presentation about MIDI/80 and it's software:



"A Presentation covering the MIDI80, the core functions and hardware, how it works and demonstration of the software including the TRACKER, STORCH (Stereo Orchestra Organ) as well as examples of MIDI Control IN, OUT and THRU including controlling the Orchestra-90 and how you can fit the TRS-80 in the studio!"

On Saturday Joel will be back with a second presentation about Downpoly, his further exploration into music using TRS-80s. He will also give a demonstration of his musical work.



"Joel will talk about the music he created using the Orchestra-90, and MIDI/80, a fusion of Synth meets Electric Rock all created using a few TRS-80 Model IIIs and 4/4Ps. He will give a presentation on the methodology used to create the music, hardware and sound setups, CDs of the initial release, along with a Limited Release Color Vinyl and more!"

You won't want to miss it!

We also have the fantastic Marcel Erz who has done some amazing work on the reverse-engineering and reproduction work on the TRS-80. His Tech Track talk on Friday is called Rebuilding the TRS-80 Model I: A Reverse Engineering Journey



"In this talk, I'll walk through my journey of reverse engineering the TRS-80 Model 1, sharing some of the curiosities I found and highlighting key differences across various board revisions. I'll also discuss the development of custom parts I've created over the past year—including a replacement keyboard, case, and other hardware—offering a look at what it takes to recreate and preserve such an iconic machine from the inside out."

Back for another year to teach us about CP/M on the TRS-80 is Tim Halloran. He will deep-drive into CP/M implementations on TRS-80s, right back to the Model I.



"This technical presentation explores the methods and challenges of running the CP/M operating system on Tandy's popular TRS-80 microcomputers (including Business systems). While TRSDOS and its tree of derivative DOSs served TRS-80 users, accessing the vast library of CP/M software required adaptations. We will delve into how CP/M works and introduce popular third-party CP/M implementations (e.g., Pickles & Trout, FMG, Lifeboat). Discuss the BIOS-level customizations needed to bridge the gap between the CP/M standard and the TRS-80's unique architecture. Discover the technical ingenuity that transformed the TRS-80 into a capable CP/M workhorse."

FREE ENTRY TO THOSE WHO ATTENDED COCO FEST 2025

Like last year, we are extending an invitation to your fellow Coco fans who attend CocoFest, for free entry to TA25. In 2024 quite a few people took us up on our offer. To do so just show us your CocoFest pass (similar to the one pictured below) as a way of proving you attended CCF and you'll be admitted free to the Tandy Assembly, best Tandy TRS-80 show in the world!



Picture: the CCF 2025 Pass everyone who registered received.

This offer applies to the Saturday and Sunday main show, not the Friday Tandy Tech Track.

THE ICING ON THE CAKE

Keep an eye out at the show on Friday for the giant cake! Have your photo taken with it, even... Andrea Zaleski, who at previous Tandy Assemblies had the table with cakes, cookies and cotton candy, will this year be making a Tandy 6000 cake. Not only that it will be life-sized 1:1 with the T6K, so in cake terms, that's enormous! Think of that television series "Is It Cake?". How close to the real thing will it be? I can't wait to see :-)



Photo: Tandy 6000HD system, typical.

TRIVIA GAME ALSO NEEDS PARTICIPANTS

I know I mentioned it last issue but I'll mention it again, I still need trivia game contestants. Can you beat Brendan Bergie (last year's winner)?

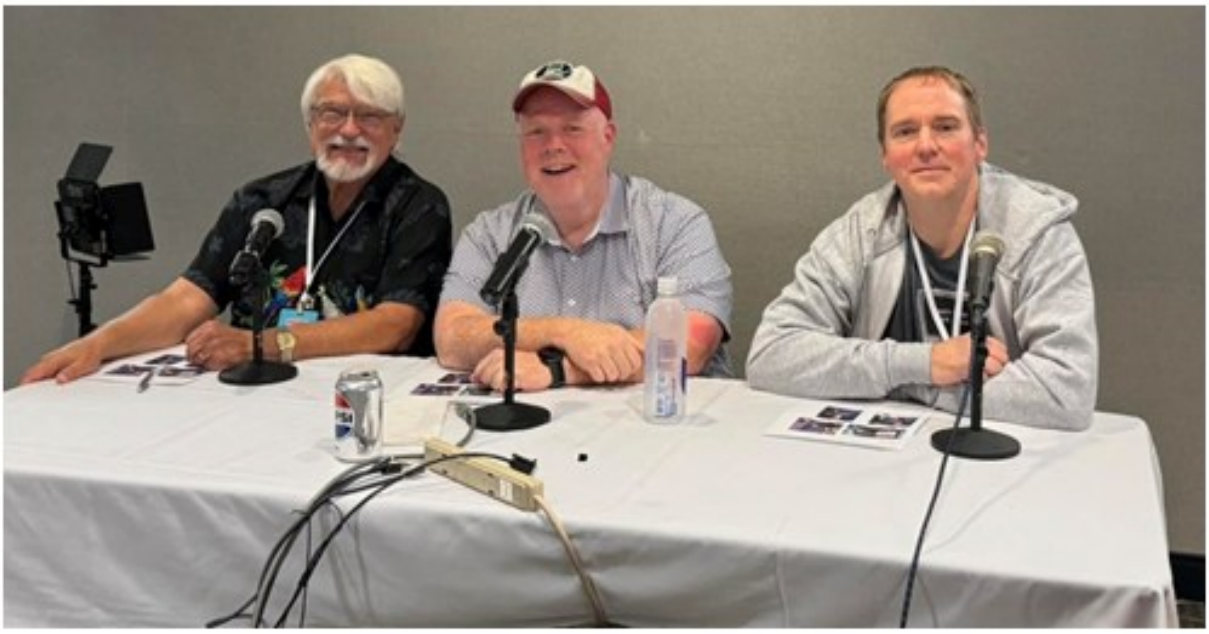


Photo: Brendan (middle) can you beat him at TA25?

If you have questions about the show or want to contact us directly out email address is: tandyassembly@gmail.com We always appreciate hearing from people who wish to be part of the show.

Next time will be less than a month before the show so I will be bringing you the rest of the presenters and other relevant information to make your visit to Tandy Assembly the best yet.

Ian Mavric

ianm@trs-80.com



Ian Marvic is an IT Specialist who also restores and collects TRS-80's and classic cars. He lives with his wife and kids in Melbourne, Australia.

BetaGamma Computing

Specialists in all things that compute



Based in Corfu Greece

- Qualified IT Engineers
- Repair of all PC's and Laptops
- Software and Hardware Upgrades and Installation
- Broadband and Network Installation
- Supply of Custom built to order PC solutions

RETRO RESTORATION SPECIALISTS

Call Corfu 26610 26358 Mobile 698 755 8427

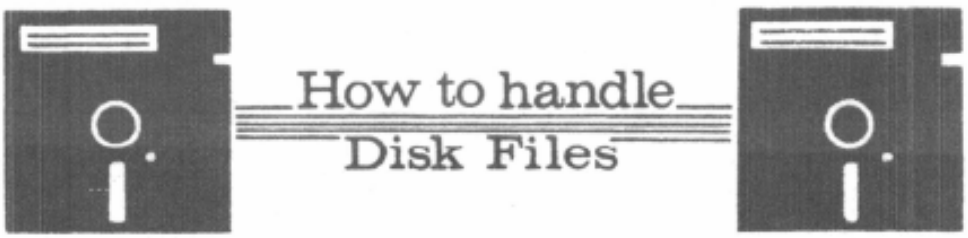
Email: bgcompute@aol.com • www.betagammacomputing.com

TRSDOS NEWDOS LDOS DOSPLUS

DISK FILES

BY JIM NORRIS

"To field or not to field
that is the sequential"



NATIONAL TRS 80 USERS GROUP
MILTON KEYNES

TRSDOS NEWDOS LDOS DOSPLUS

TRS80 USER GROUP.

DISK FILE SYSTEMS FOR MODELS 1,2, and 3
By JIM NORRIS Part 2

14. FIELDING NUMERICAL VALUES IN RANDOM ACCESS FILES

Most people are familiar with the the commands STR$ and VAL.

The problem with STR$ is that it produces different numbers of characters depending on the value of the variable.

When packing values in a random file record it would be very cumbersome for each value to have a different number of ASCII characters. However there is a much better way which enables you to pack values in a uniform way without any loss of precision.

The method of packing does NOT use ASCII characters.

The four options are described in Section (9) above and there now follow some examples:-

Note: All these examples assume a 256 byte record length.

a) It is proposed to build a file containing the heights of all the children in a school. The heights will be in decimal metres, they are to be grouped according to 6 age ranges and there is a maximum of 60 children per age range.

There will have to be an input program to put the values in an Array. Since the values have fractional parts but lie in a limited range then ordinary single precision arithmetic will be adequate. The input program could be like this:-

```
10 DIM H(6,60)
20 INPUT"AGE RANGE 1-6";X
30 INPUT"PUPIL NUMBER 1-60";Y
40 INPUT"HEIGHT";H(X,Y)
50 GOTO 20
```

In practice this would be rather more sophisticated, however eventually all the data would be entered in the array.

Each element of the array is represented in memory by 4 bytes. The first byte is an exponent and the other three the mantissa. Our task is to save this data on disk in an organised way. First open a random file:-

```
100 OPEN"R",1,"HFILE"
```

Now build a loop to pack the data into records and save them to disk. The data for year group 1 will be in record 1, for year group 2 in record 2 and so on

```
110 FOR R = 1 TO 6:'YEAR GROUP
120 FOR P = 1 TO 60:'PUPIL NUMBER
130 FIELD 1, 4*(P-1) AS DM$, 4 AS H$
```

DISK FILE SYSTEMS FOR MODELS 1,2, and 3
By JIM NORRIS Part 2

```
140 LSET H$ = MKS$(H(R,P))
150 NEXT P
160 PUT 1,R
170 NEXT R
180 CLOSE 1:STOP
```

When P=1 then the length of DM$ is 0 and H$ starts at byte 1 and finishes at byte 4. The value from the array which is 4 bytes long is set into bytes 1-4 of the record.

When P=2 then the length of DM$ is 4 and H$ starts at byte 5 and finishes at byte 8. The next value from the array is set into bytes 5-8.

The process continues until P > 60. The last 4 byte value goes into bytes 237 to 240 of the record leaving 16 bytes unused.

The whole record is then written to disk and the process is repeated for the next year group until all six records have been written.

It is now possible to read any individual pupils' height from disk as follows:-

```
200 OPEN"R",1,"HFILE"
210 INPUT"AGE GROUP";R
220 INPUT"PUPIL NUMBER";P
230 GET 1,R
240 FIELD 1, 4*(P-1) AS DM$,4 AS H$
250 PRINT"HEIGHT = ";CVS(H$)
260 GOTO 210
```

If R < 1 or R> 6 then an error will occur and similarly if P < 1 or P > 64 an error will occur, so in a practical program R and P must be validated before going to line 230.

MKS$ and CVS. These are complimentary commands. MKS$ (MaKe a Single precision variable into a String) simply copies the 4 byte representation of a single precision variable from memory into the record in the buffer. CVS (ConVert a Single precision string into a variable) takes the 4 bytes out of the buffer and puts them into the specified variable in memory. In line 250 there is no variable name specified and so BASIC treats the byte string H$ as if it were a variable.

If it was proposed to carry out a lot of manipulations on the data in the six records then it would be more efficient to read all the values back into the array H(6,60). However it is perfectly permissable to carry out all the required manipulations in the record buffer. This might well involve a large number of disk transfers and would be slow and place an unnecessary strain on the disk drive.

TRS80 USER GROUP.

DISK FILE SYSTEMS FOR MODELS 1,2, and 3
By JIM NORRIS Part 2

b) A shopkeeper carries 10000 stock items and he wants to accumulate over a period of time the units sold of each item. He numbers his stock from 1 to 10000, and he knows that the most he will ever sell of any item will be 25000.

Since the values are integers and since the numeric range is less than 32767 we can use integer variables which only use 2 bytes in memory. We can write a program which updates the stock items directly to disk without using an array, but first we must set all the values on disk to 0

```
10 'SET ALL VALUES TO ZERO
20 OPEN"R",1,"STOCKFILE"
30 FOR R = 1 TO 80
40 FIELD 1,250 AS S$
50 LSET S$ = STRING$(250,CHR$(0))
60 PUT 1,R
70 NEXT R
80 CLOSE 1
```

Now we can write the program that can be running all day and allow updating of stock values at any time.

```
100 OPEN"R",1,"STOCKFILE"
110 INPUT"STOCK REFERENCE";S
120 INPUT"QUANTITY SOLD";Q
130 R = 1 + INT((S-1)/125)
140 P = S - 125*(R-1)
150 GET 1,R
160 FIELD 1, 2*(P-1) AS DM$, 2 AS S$
170 LSET S$ = MKI$(CVI(S$)+Q)
180 PUT 1,R
190 GOTO 110
```

In this example we are fielding 125 stock items per record using 250 bytes and leaving 6 bytes unused. Therefore we need 80 records to represent all 10000 stock items.

Lines 130 and 140 find the record number and the position in the record of the stock item, and 160 identifies the item as S$.

In line 170 S$ is converted to a numerical value, added to Q and the result converted back to a 2 byte string and set back in the record.

MKI$ and CVI. This is a complementary pair. MKI$ (Make an Integer into a string) copies the value of ANY variable into a two byte destination in the record buffer. This only works if the variable has a value in the range >-32767, <+32768.

CVI (ConVert a two byte string into an Integer) copies the two byte integer value into ANY variable.

DISK FILE SYSTEMS FOR MODELS 1,2, and 3
By JIM NORRIS Part 2

Providing the value is in range then all these are allowed:-

```
LSET S$ = MKI$(S%)
LSET S$ = MKI$(S!)
LSET S$ = MKI$(S#)
```

Fractional parts of S! and S# will be lost of course.

c) A mail order company has 50,000 accounts and requires to keep a record of the last occasion that goods were ordered. They only need to know to the nearest month and the system has to work for 20 years. Can they get all the data on one Model 1 Data Disk ?

In a period of 20 years there are 240 months and so for each account we need to have a variable which is an integer and does not exceed a value of 240. We can use a single byte for this as follows:-

```
10 OPEN"R",1,"CUSTOMERS"
20 INPUT"CUSTOMER NUMBER";N
30 R = 1 + INT((N-1)/256)
40 P = N - 256*(R-1)
50 GET 1,R
60 FIELD 1,1*(P-1) AS DM$,1 AS M$
70 PRINT"LAST ACTIVE MONTH NUMBER";ASC(M$)
80 M = ASC(M$)
90 INPUT"LATEST ACTIVE MONTH NUMBER";M
100 LSET M$ = CHR$(M)
110 PUT 1,R
120 GOTO 20
```

Here we are using every byte in each record for a customer. This is 50 Kbyte in total which can be accommodated on single density 35 track model 1 DATA disk.

ASC converts a single byte string into is binary value and places it in a variable (M in this case).

CHR$ will convert any variable into a single byte value providing the variable is in the range 0-255.

The effect of line 80 is to preset M to the current value in case the operator presses ENTER with no value to the question in line 90. This ensures that at least the current value is kept in line 100.

This pre-setting of variables before a question can be very useful to allow operators to press ENTER with no value and still retain the original value.

(d) The only complementary pair of buffer commands not mentioned so far are MKD$ and CVD$. These operate on double precision variables (T#, U# etc) and they need 8 bytes, and otherwise can be used in the same way as described above.

TRS80 USER GROUP.

DISK FILE SYSTEMS FOR MODELS 1,2, and 3
By JIM NORRIS Part 2

15. REAL LIFE EXAMPLES.

The examples in section (14) were contrived to demonstrate the four arithmetic forms. The following examples are taken from actual applications. It must be recognised that the fixed 256 byte record length of Model 1 is a handicapp compared to the variable record lengths available in Models 2 and 3. If the following examples are to be implemented on model 1 then you will have to write a subroutine to pack and unpack sub-records into the standard 256 byte record.

(a) Stock control system for a Seeds Merchant using Model 3.

Stock File.
Name: SFILE Record Length: 50 File length: 500+
Record layout:

Bytes	Length	Type	Name	Description
1-30	30	$	S1$	Text Description
31-34	4	CVS	S2%	Quantity in stock
35	1	$	S3$	Units ("N","K","G")
36	1	ASC	S4%	Supplier number (1-255)
37-40	4	CVS	S5%	Buying price (sterling)
41	1	ASC	S6%	Supply Date (month number)
42-45	4	CVS	S7%	Quantity Supplied
46	1	ASC	S8%	VAT Code
47-50	4	CVS	S9%	Buying Price - Foreign Currency

This is designed for a specific application and is only one of 10 files in the system. In practice this file was always on a disk in drive 1 and always used buffer 2. Therefore a file open subroutine was placed in each program using this file as follow:-

```
300 OPEN"R",2,"SFILE:1",50
310 FIELD 2, 30 AS S1$,4 AS S2%,1 AS S3$,1 AS S4%,4 AS S5%,1
AS S6%,4 AS S7%,1 AS S8%,4 AS S9%
320 RETURN
```

Once a file is open and the buffer fielded as above it does not have to be fielded again however many disk transfers take place.

REMEMBER. Do not have string variables in your program with the same names as buffer variables.

Having opened the file you will need a program to enable you to enter and edit items into the records. This program module creates new records and allows entry and editing of the text.

DISK FILE SYSTEMS FOR MODELS 1,2, and 3
By JIM NORRIS Part 2

```
500 R=0:INPUT"STOCK REFERENCE";R: IF R=0 THEN 560
510 IF R<1 OR R>LOF(2)+1 THEN 500
520 IF R=LOF(2)+1 THEN LSET S1$="Not in use" ELSE GET 2,R
530 PRINT"STOCK DESCRIPTION:";S1$
540 A$=S1$:INPUT"CHANGE TO:";A$
550 LSET S1$=A$:PUT 2,R:GOTO 500
560 'NEXT MODULE
```

In line 500 R is pre-set to give a method of escape from the sequence.

In line 510 R is validated and only allows existing records or the next unused record for a new stock item.

In line 520 the description is pre-set for a new item or the record R is obtained for an existing item.

In lines 530-540 you can print straight from the buffer but if you input to the buffer directly then BASIC will create a new variable S1$ which will be other than in the buffer, so use an intermediate variable, A$. A$ is pre-set to S1$ in case the operator presses ENTER with no value.

It would be best at this stage to write these program modules and experiment with them. In a practical program it would also be necessary to position the PRINT and INPUT statements in the same place on screen for every new item so that a stable and acceptable picture was presented without scrolling.

(b) Indexed Filing. The above is a very simple example of random filing. This example is also for a stock system, this time for manufactured parts.

The parts are numbered from 1 to 2000, but many of the part numbers are not used and so to have a detailed stock record for every part number would be a waste of disk space.

The stock records are 50 bytes long and contained in a file called SFILE, but there is another file to cross reference the part numbers to the stock records, this is called CRFILE.

CRFILE is 2000 records long and each record contains either the value 0 indicating an unused part number, or a record number in SFILE.

If only half the part numbers are used then the disk space used with this technique is 1000*50 bytes for SFILE and 2000*2 bytes for CRFILE, a total of 54000 bytes.

If every part number had a record in SFILE then this would use 2000*50 bytes, i.e. 100000 bytes. So the cross reference technique saves 46000 bytes.

The file open modules are as follows:-

```
300 OPEN"R",1,"CRFILE:1",2
310 FIELD 1,2 AS C1$:RETURN
320 OPEN"R",2,"SFILE:1",50
```

DISK FILE SYSTEMS FOR MODELS 1,2, and 3
By JIM NORRIS Part 2

```
330 FIELD 2,30 AS S1$, etc etc as in (a)
340 RETURN
```

This module gets the stock file entry for any part number:-

```
500 N=0:INPUT"PART NUMBER";N:IF N=0 THEN 540
510 IF N<1 OR N>LOF(1) THEN 500 ELSE GET 1,N
520 S=CVI(C1$):IF S=0 THEN 500 ELSE GET 2,S
530 PRINT"STOCK DESCRIPTION:";S1$:GOTO 500
540 'NEXT MODULE
```

In line 500, N is preset to 0 to provide an escape.
In line 510 N is validated to make sure it exists.
In line 520 the stock record number S is calculated and if zero then return is to 500 for another part number.

Of course all this pre-supposes a previous program module to build the stock file entries and the cross reference file. Try writing a program to do this, such that part numbers can be entered at random, and arrange it so that part numbers greater than 2000 can be entered in the future without program modification.

Note: It will be noticed that the stock references in both cases are simple integers starting at 1. In many practical situations stock referencing is often by complex combinations of letters and numbers, frequently with no logical progression. This presents a serious problem on micros with limited memory as will now be demonstrated:-

(c) Stock system for Jewellery Manufacture.

In this example there are 6000 stock items and an ad-hoc system of identification has built up over a period of 50 years. Some items are known by simple numbers and others by combinations of numbers and letters and there is no obvious logical progression.

It is not possible to write a program to analyse these stock references and relate them to a record number. The references need 8 bytes each and therefore it will be necessary to build a disk file of 6000*8 byte records which can be searched for any stock reference that is typed in and then use the same record number in a stock file.

This searching process however will take a long time, in fact anything up to several minutes which is quite unacceptable to the operator.

If all the references could be held in memory then a machine code routine could find the reference in a second or so, but to keep all 6000 references in memory would require at least 50000 characters, leaving no room for BASIC or any

DISK FILE SYSTEMS FOR MODELS 1,2, and 3
By JIM NORRIS Part 2

programs.

An answer to this problem is to persuade the user to refer to his stock by simple numbers which can be the record numbers of the stock items. This gives him instant access to any item. The text description in each stock record can include the 8 byte reference used before, and this is displayed on screen for confirmation. If the operator has typed in the wrong number then this is obvious and he can type in another number. Experience has shown that after a relatively short period of time the operators learn the new simple numbering system and rely less and less on seeing the old reference appear on screen for confirmation. In general if a file is OPEN then a record can be read and displayed on screen within .2 second.

The new generation of 16 bit micros with memories of 500Kbyte or more will of course be able to hold long and complex cross reference tables and selection and sorting using machine code routines will produce quite startling results.

(d) Purchase Orders - Engineering

This application required that detailed purchase orders should be held in memory and recalled and updated as deliveries occured - possibly on several occasions before the purchase order was complete. Each record is one order and contains fixed and variable information as follows:-

Name: PBFILE Record Length: 248 File Length: 50+

Bytes	Length	Type	Name	Description
1-4	4	CVS	B1$	Order Number (0=not in use)
5-7	3 * 1	ASC	B2$	Day (1-7))
			B3$	Month (1-12)) Order Date
			B4$	Year (-1982))
8-9	2	CVI	B5$	Pointer to another record
		Item 1		
10-11	2	CVI	I1$	Part number (0=not used)
12-13	2	CVI	I2$	Stock file record number
13-15	2	CVI	I3$	Number of parts ordered
16-17	2	CVI	I4$	Parts supplied so far
18-20	3 * 1	ASC	I5$	Day (1-7))
			I6$	Month (1-12)) Delivery date
			I7$	Year (-1982))
21	1	ASC	I8$	Completion Flag this item
22-33		As above for item 2		
34-45		As above for item 3		
etc				
etc				

237-248 As above for item 20

This more complex layout has a number of features. Each order has a maximum of twenty items - an agreed specification. If any order would have required much more than 20 items then a completely different approach would be required.

All the suppliers are in a supplier file and a two byte field in a supplier record points to one of these orders in PBIFLE. B5$ contains a pointer to another record in PBFILE, and in this way any supplier can have any number of orders at a time.

Each order is identified by an order number (B1$).

The method of representing the date (B2$ - B4$) is efficient without using unnecessary space and at the same time providing the relevant figures for printout without too much re-processing.

This program module demonstrates obtaining and displaying the details of any item on the order:-

```
500 'assume an order record already in buffer 2
510 'and that buffer 1 is open to the stock file
520 I=0:INPUT"ORDER ITEM NUMBER";I:IF I=0 THEN 660
530 IF I<1 OR I>20 THEN 520
540 FIELD 2,9+12*(I-1) AS DM$,2 AS I1$,2 AS I2$,2 AS I3$,2 AS
I4$,1 AS I5$,1 AS I6$,1 AS I7$,1 AS I8$
550 IF CVI(I1$)=0 THEN 520 ELSE GET 1,CVI(I2$)
560 PRINT"PART NUMBER"CVI(I1$)
570 PRINT"DESCRIPTION"S1$
580 PRINT"NUMBER OF PARTS ORDERED"CVI(I3$)
590 PRINT"NUMBER SUPPLIED"CVI(I4$)
600 PRINT"LAST SUPPLY DATE";
610 PRINTUSING"##";ASC(I5$);:PRINT"/";
620 PRINTUSING"##";ASC(I6$);:PRINT"/";
630 PRINTUSING"####";ASC(I7$)+1982
640 IF ASC(I8$)=1 THEN PRINT"THIS ITEM COMPLETED" ELSE
PRINT"THIS ITEM NOT YET COMPLETED"
650 GOTO 520
660 'NEXT MODULE
```

The practical program laid the details out horizontally so that most of the items could be displayed on screen at the same time under column headings. Also of course there would be other program modules to build the order record, edit it, notify deliveries, and print the details as required.

16. FILE AND PROGRAM DESIGN

To succesfully write a suite of programs using file handling in a practical application you should plan and execute as follows:-

TRS80 USER GROUP.

DISK FILE SYSTEMS FOR MODELS 1,2, and 3
By JIM NORRIS Part 2

(a) Examine the requirements of the application in detail, and write a specification for the application in non-programming terms. e.g. for a business application, how many customers, suppliers, stock items, ledger types, etc (allowing for future expansion)! the precise details of the information required including length of text, dynamic range of variables etc! special features involving analysis of data, printout, operator response etc!

(b) Now try and plan the filing system to cater for all the data. Check to see if there is sufficient disk capacity (don't forget cross reference tables and programs). If there is limited disk space and disk swapping is required then carefully plan the file layout to keep this to a minimum.

(c) Try and assess if the response of the system to operator commands will be fast enough to be acceptable. If not, then re-plan the filing system and consider incorporating machine code routines for faster selection and sorting.

(d) Try not to overexercise the disk drive - if you plan long sorts or searches on disk then anticipate disk errors and plan your program to be able to recover without loss of system integrity.

(e) Plan the programs so that related functions are grouped together. Make sure there is enough work space for arrays. Try and avoid unnecessary string manipulation. In a crowded memory the system will keep locking up while it carries out string housekeeping and is most annoying to operators.

(f) Now write the programs. Always display stable pictures and never scroll off screen - this is most disconcerting to operators. Tandy Basic is well provided with screen control codes and there are techniques to change the scroll position. If a file is going to be used in many parts of a program then leave it OPEN all the time. Make sure you have enough buffers available to avoid constantly opening and closing files. Write the program in easily debugged modules. Put all the open commands and fielding commands in one place as sub-routines to avoid accidental and incorrect re-fielding. Make the program menu-driven and when in actual use disable the Break key. Don't forget to make proper provision to CLOSE all files before leaving the program. Document EVERYTHING you do.

Thirsting for adventure?
Brass lantern getting dusty?

UPDATED 2025!



Take your Model I/III/4/4p on exciting new adventures never available until now....

M4ZVM enables your TRS-80 Model I, III or 4 to play Infocom and post-Infocom adventures, taking full advantage of your hardware. Supports old favourites like Zork and Seastalker (including Sonarscope), as well as new-to-Model-4 games including Trinity, Border Zone and Sherlock, plus hundreds more available in the Interactive Fiction archive.



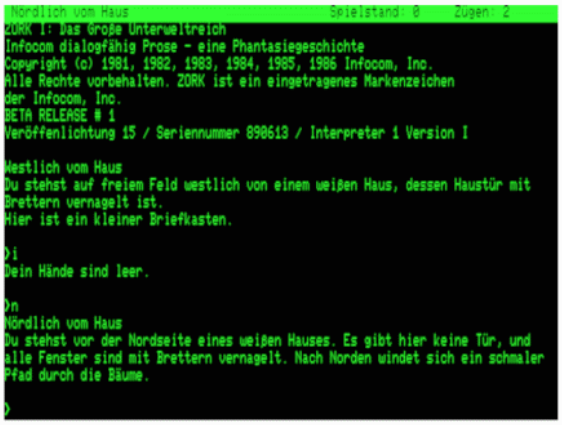
M4ZVM & M3ZVM with sample disks and game compatibility lists are at:
https://sijnstra.itch.io/m4zvm

Source and links to other versions at:
https://gitlab.com/sijnstra1/m4zvm

Actual Screenshots of games new to the model 4, running M4ZVM (on trs80gp) - check links for how-tos & disks



Beyond Zork



Zork 1 (German)

M4ZVM/M3ZVM/M1ZVM features include:
* z1-z8 game execution (z6 support is experimental)
* named game save/load
* accented characters where available
* Optionally set the "Tandy bit" for Tandy customization!

UPDATED 2025 version - faster, better, more bugfixes!



Alan Wallace has just released version 3.99922 of his much enhanced *STAR TREK* game



https://www.youtube.com/watch?v=ZE3nMxTLgjk

IX. Basque Tournament of TRS-80

The last 22th of april, thursday, the Euskal Retro organized the IX. Basque Tournament of TRS-80. We met 4 players in Bilbao, in Cafe Berria (a bar located in Santutxu), and we played TRS-80 in the second floor of the bar, as always we do.

It was a great afternoon, funny as always that we gather to play consoles. We plugged in the TRS-80 when we arrived and started playing. It was the first console that we played. Then later we played other classic consoles such as the Commodore Amiga 500 or Sega Megadrive. We played to these 4 games: Monster Maze, Canyon Climber, Downland and Clowns & Ballons.

Egoitz got more points than the rest of players after playing 4 games, but the other players played well and all of them passed a great time playing TRS-80. It should be remembered, however, that Egoitz practices at home, and for that reason he does have a slight advantage. For the next tournament, possibly in july of 2025, we hope to have new games like Dragon Fire or Robot Battle, I have to buy new ones.

Here you have the results:

Canyon Climber

1-Egoitz Campo 4 points (13.200)
2-Joseba Zubiaurre 3 points (3800)
3-Daniel Barrientos 2 points (3.200)
4-Lander Campo 1 point (2.800)

Monster Maze

1-Egoitz Campo 4 points (1.190)
2-Joseba Zubiaurre 3 points (770)
3-Daniel Barrientos 2 points (410)
4-Lander Campo 1 point (200)

Downland

1-Egoitz Campo 3 points (1.337)
2-Daniel Barrientos 2 points (800)
3-Joseba Zubiaurre 1 point (432)

Clowns & Ballons

1-Egoitz Campo 4 points (2.260)
2-Daniel Barrientos 3 points (990)
3-Joseba Zubiaurre 2 points (600)
4-Lander Campo 1 point (240)

Classification

1-Egoitz Campo 15 points (Basque Country)

2-Joseba Zubiaurre 9 points (Basque Country)

3-Daniel Barrientos 9 points (Basque Country)

4-Lander Campo 3 points (Basque Country)













SCORE
000010
MEN
001
HIGH
SCORE
001190





Part 2, Maths UG!!

I'm an old dog learning new tricks, like doing quick calculations using *Google Home*:

> "Hey Google! If it takes me an hour-and-a-half to get to London, Ontario at 60 miles per hour, how far away is it?"

What I found out is that *Google Home* can't make sense of this query. No matter how I rephrased it, *GH* replied, "I can't answer that question," and would offer some web search results. It was only able to give me a correct answer when I asked:

> "Hey Google! What's 60 times 1.5?"
> *"60 times 1.5 equals 90."*

Gee, thanks. I had that figured out in my head before I'd even phrased the first question. But that's the first equation that Dr. Lien uses to begin to teach Level I BASIC math with the question, "How far is it from Boston to San Diego if a jet plane traveling an average speed of 500 miles per hour makes the trip in 6 hours?":

```
10 REM * DISTANCE, RATE, TIME PROBLEM *
20 R = 500
30 T = 6
40 D = R * T
50 PRINT "THE DISTANCE (IN MILES) IS", D
```

As an introduction to Single Precision math operations using the Level II ROM in a Model I TRS-80, converting this program to Z80 assembly can demonstrate:

- Assigning a number to a variable
- Accessing the number in a variable
- Performing an arithmetic operation with Single Precision numbers
- Outputting a numeric value

Why do I say "Single Precision"? Aren't `500` and `6` (and the calculated value of `D`) *integers*? They are, but

- The Level I BASIC taught in Dr. Lien's book has only one number type: Single Precision; and
- I'm going to demonstrate how to use the Single Precision ROM math routines in these articles.

Let's get right to it. First off—if we just call a ROM Math routine (to input a value, or add it to another, for example), the TRS-80 will crash and usually reboot. There's a fair bit of setup code that must be executed before any ROM Math subroutines are called. As I demonstrated in the first installment of this series, in my working environment I use Microsoft Editor Assembler Plus, which provides a limited macro capability. So I usually include a couple of lines in my assembly source that look like this:

```
START   EQU     $
        INIT                    ; SETS UP SEVERAL SUPPORT ROUTINES
        ;
        ; MY NON-MATH CODE HERE…
        ;


        ;
        ; NOW I WANT TO USE MATH ROUTINES:
        ;
        USES    MATH            ; NOW I CAN USE MATH SUBS
        ;
        ; MY MATH WORK HERE…
        ;
        ; NOW TO RESTORE THE ORIGINAL VECTORS
        ;    BEFORE EXITING MY PROGRAM:
        ;
        FORGET MATH             ; ORIGINAL VECTORS RESTORED
        ;
        JP  EXIT
        ;
```

The macro INIT "expands" to several lines of code that initialize some necessary values used by the ROM routines. USES MATH is a macro which calls all the subroutines necessary to initialize the vectors which the BASIC ROM Math routines expect to use. FORGET MATH is the third macro which resets those vectors to the state they were in before we hijacked them for math purposes. (All of this is explained in greater detail in the article "*ROM Calls from TRSDOS 2.3 With Error Trapping*" on page 92 of the September 2024 edition of TRS8Bit.) So: if we want to use ROM math routines, we have to

- "INIT": Run the code in INIT *before* we run our user program;
- "USE MATH": run the code in USES MATH to Initialize all the Math vectors before we use the ROM math routines; and
- "FORGET MATH": run the code in FORGET MATH to reset the vectors after we're done using the math subroutines.

What I'll do here in this article to keep it simple is to simply provide the subroutines for INIT, USE MATH, and FORGET MATH without a lot of explanation.

Here's the relevant code in INIT:

```
;
;     INIT CODE
;
BERROR      EQU     409AH           ; BASIC "ERR" HERE
                                    ; CHECK THIS AFTER USING
                                    ; ROM MATH CALL
                                    ; 0 IF NO MATH ERROR
INIT        EQU     $
            XOR     A               ; ZERO VALUE
            LD      (BERROR),A      ; BASIC ERROR CODE 0
            LD      (411BH),A       ; TURN "TROFF" OFF
            RET
```

(Why do I start subroutines like INIT and those following with an EQU $ pseudo-op instead of the actual instructions? The Microsoft Editor Assembler Plus (and the earlier Radio Shack TRS-80 Editor/Assembler upon which it was based) uses "plain" labels, whereas assemblers like Macro-80 append a colon to them, like

```
INIT:
            XOR     A
            LD      (BERROR),A
            ...
```

Hopefully, the EQU $ makes my code a little more readable for users of other Z80 assemblers.)

USES MATH is a simple subroutine call, but that subroutine factors out to 5 separate tasks:

```
;
;    "USES MATH" CODE
;
I00CRM      EQU     $
;
;     1. PATCH "RET" FOR DISK BASIC ROUTINES
            CALL    PCHRET
;
;     2. "DIVISION SUPPORT ROUTINE"
            CALL    DIVSUP
;
```

```
;     3. PATCH 28 COPIES OF "JP 012DH" INTO
;          4152H..41A5H (DISK BASIC VECTORS)
;          TO PRODUCE "SN ERROR L3" IF
;          THEY ARE CALLED (FROM L2 BASIC)
               CALL     PTCH28
;
;     4. SET BASIC "ON ERROR" TRAP TO "ERTRAP"
               CALL     INITER
;
;     5. SET "DOS EXITS" AT 4000H..4015H
;          TO THOSE USED BY LEVEL II BASIC
               CALL     L2DOSX
               RET
```

and here are each of those subroutines:

```
;
;          PCHRET -- PATCH "RET" FOR DISK BASIC ROUTINES
;
PCHRET    EQU      $
               LD       HL,41A6H          ; PATCH 41A6H..41E2H
               LD       B,15H             ; 21X WITH
I0LOOP    EQU      $
               LD       (HL),0C9H         ; "RET"
               INC      HL                ; ORIGINAL CODE @
               INC      HL                ; 00A1H
               INC      HL
               DJNZ     I0LOOP
               RET

;
;          DIVSUP -- "DIVISION SUPPORT ROUTINE" INIT
;
DIVSUP    EQU      $
               LD       HL,18F7H          ; COPY OF CODE @ 18F7..190DH
               LD       DE,4080H          ; PATCH 4080H..4096H
               LD       BC,22
               LDIR
               RET

;
;     PTCH28 -- PATCH 28 COPIES OF "JP 012DH" INTO
;          4152H..41A5H (DISK BASIC VECTORS)
;          TO PRODUCE "SN ERROR L3" IF
;          THEY ARE CALLED (FROM L2 BASIC)
;
```

```
PTCH28      EQU     $
            LD      DE,012DH        ; "012D"
            LD      B,1CH           ; 28 COPIES
            LD      HL,4152H        ; START @ 4152H
L00R96      EQU     $
            LD      (HL),0C3H       ; "JP"
            INC     HL
            LD      (HL),E          ; "2D"
            INC     HL
            LD      (HL),D          ; "01"
            INC     HL
            DJNZ    L00R96
            RET
;
;     INITER -- INITIALIZE ROM ERROR TRAP VECTOR
;
INITER      EQU     $
            LD      HL,41A6H        ; REPLACE EXISTING ERROR
            LD      (HL),0C3H       ; VECTOR WITH JUMP TO
            LD      DE,ERTRAP       ; "ERTRAP" ERROR-TRAP
            INC     HL
            LD      (HL),E          ; ROUTINE
            INC     HL
            LD      (HL),D
            RET
```

If the BASIC ROM math routine encounters an error (like "divide by 0"), it will jump to 41A6H, the "Error Vector". INITER replaces the code there with a JP ERTRAP, directing the Z80 to our replacement error-handling routine, ERTRAP (which in this case performs the equivalent of BASIC's "RESUME NEXT" by resuming with the next instruction after the CALL to the ROM Math routine we just invoked. Here's ERTRAP:

```
;
;         ERTRAP -- "ERROR TRAP" SUBROUTINE
;
ERTRAP      EQU     $
            LD      HL,(SVSP00)     ; RESTORE SP
            LD      SP,HL
            LD      HL,(SVNX00)     ; RETURN TO NEXT
            JP      (HL)            ; INSTRUCTION AFTER
                                    ; CALL TO ROM CODE
;
SVSP00      DEFW    0               ; SET BY "GETSPC"
SVNX00      DEFW    0               ; ADDR OF NEXT INSTRUCTION
```

Note: whenever we call a BASIC ROM Math routine in our user code, it's imperative that we precede that call with a CALL GETSPC first. For example, we want to convert a 0-terminated string like "1234567" to a single-precision number using the ROM call 0E6CH. Here's how we might code that safely:

```
;
;       CONVERT STRING POINTED TO BY HL TO NUMBER
;
CONVRT  EQU     $
        LD      HL,NUMSTR       ; STRING TO CONVERT
        CALL    GETSPC          ; "ON ERROR RESUME NEXT"
        CALL    0E6CH           ; CONVERT IT TO NUMBER
NEXT    EQU     $               ; EVEN IF CONVERSION HAS
                                ; AN ERROR, CONTINUE HERE
;
NUMSTR  DEFM    '1234567'
        DEFB    0               ; 0-TERMINATED STRING
;
```

So whether the CALL to 0E6CH succeeds or fails, the program will continue at NEXT. (How do we know if there was an error? By inspecting BERROR after the CALL to the ROM.) Here's the code for GETSPC:

```
;
;   GETSPC – "GET SP & PC" SUBROUTINE
;       SAVES CURRENT SP AND PC OF
;       INSTRUCTION FOLLOWING NEXT "CALL XXXX"
;       USED IN ERROR-TRAPPING FOR
;       BASIC ROM ROUTINE CALLS
;
GETSPC  EQU     $
        PUSH    DE          ; DON'T CLOBBER S.P. ARG
        PUSH    HL
        PUSH    AF
        XOR     A           ; CLEAR "BERROR"
        LD      (BERROR),A
        DEC     A
        LD      (40F2H),A   ; RESET "ON ERROR" FLAG
        LD      HL,8        ; GET SP BEFORE THIS CALL
        ADD     HL,SP
        LD      (SVSP00),HL ; SVSP00= SP "BEFORE"
        DEC     HL
        DEC     HL
        LD      E,(HL)      ; DE= PC OF CALLER
```

```
        INC     HL
        LD      D,(HL)
        INC     DE
        INC     DE
        INC     DE              ; DE= PC FOR NEXT INSTR
        LD      (SVNX00),DE ; SVNX00= PC + 3
        POP     AF
        POP     HL
        POP     DE
        RET
```

#5: it's necessary to replace the "DOS Exits" at 4000H. Whenever the Z80 encounters a RST instruction (RST 08H, RST 10H, RST 18H..RST 38H), the ROM code immediately directs it to JP instructions stored at 4000H, 4003H, 4006H..4015H. In a cassette-based machine like my original 4K Level II TRS-80, the JP instructions just point back into the ROM. Most of the literature I've read refers to these JP instructions as "DOS Exits", presumably because Disk BASIC will replace them with jumps into its extended code. *However,* if you're running a TRS-80 under TRSDOS, these DOS Exits will not have been initialized to any BASIC code, and so once again when you try to use ROM BASIC subroutines you'll end up in never-never land. It's necessary to patch them with JPs to the BASIC code, but before we do, we must save the existing vectors so that they can be restored after we're done with using Math routines.

```
VS          EQU   7*3               ; 7 INSTANCES OF 3-BYTE INSTRUCTIONS
                                    ; RADIO SHACK EDITOR/ASSEMBLER USERS:
;
;     L2DOSX – SAVE DOS EXITS AND INSERT LEVEL II VECTORS
;
L2DOSX      EQU   $
            CALL  V4000S            ; A. SAVE EXISTING
                                    ; DOS EXITS
                                    ; B. INSERT L2 EXITS
            LD    HL,V4000M         ; LIST OF "JP"S FOR MATH
            LD    DE,4000H
            LD    BC,VS
            LDIR
            RET
;
;     V4000M -- DISK BASIC DOS EXITS AT
;           4000H..4012H
;
V4000M      EQU   $
            JP    1C96H       ; RST 08 / 4000H
            JP    1D78H       ; RST 10 / 4003H
```

```
        JP      1C90H       ; RST 18 / 4006H
        JP      25D9H       ; RST 20 / 4009H
        JP      4BA2H       ; RST 28 / 400CH
        JP      44B4H       ; RST 30 / 400FH
        JP      4518H       ; RST 38 / 4012H

;
;   V4000S -- SAVE VECTOR(S) AT 4000H+
;
V4000S  EQU     $
        LD      HL,4000H
        LD      DE,V4000D
        LD      BC,VS
        LDIR
        RET
;
V4000D  DEFS    3*7         ; SAVE DOS VECTORS HERE
```

So L2DOSX ("Level II DOS Exits") CALLs V4000S to "save the Vectors at 4000" in V4000D before patching in the list of JPs at V4000M that Disk BASIC uses.

A couple of notes about early TRS-80 assemblers:

1. Radio Shack's Editor/Assembler doesn't support
    a. Multiplication ("*") in its expressions;
    b. Expressions as an argument to the DEFS pseudo op: it's necessary to replace every instance of "VS" with a literal "21".
2. Microsoft Editor Assembler Plus: if a DEFS contains a label as in its argument, that label must be defined in the source code *before* it's used.

Later, when we are done with the Math routines and want to FORGET MATH, the original vectors saved in V4000D will be patched back into RAM at 4000H..4015H. We can do that with a call to V400R:

```
;
;   V40000R -- RESTORE SAVED VECTORS TO 4000H+
;
V4000R  EQU     $
        LD      HL,V4000D
        LD      DE,4000H
        LD      BC,VS
        LDIR
        RET
```

So, in our main program, we start with the initializing code that will prepare for when we want to use the BASIC ROM routines.

```
MAIN    EQU     $               ; START OF USER PROGRAM
CALL    INIT
CALL    I00CRM                  ; "USES MATH"
```

There. We're finally ready to actually use some ROM code! What was that BASIC program that we were going to translate into assembly language again?

```
10 REM * DISTANCE, RATE, TIME PROBLEM *
20 R = 500
30 T = 6
40 D = R * T
50 PRINT "THE DISTANCE (IN MILES) IS", D
```

Now assembly-language experts know that R and T get set to *integer* values and even D will be calculated as an integer "small enough" to fit into the Z80's 16-bit registers, so we could do all this math quite quickly without ever using ROM floating-point subroutines. But of course, that's not the point of this series of articles: we *want* to use the ROM, because we want to learn how to do floating-point math without re-inventing the wheel. So what we can do is to *start* with R and T as integer values:

```
RINT    DEFW    500     ; RATE AS AN INTEGER
TINT    DEFW    6       ; TIME AS AN INTEGER
```

What we'll do next is to *convert* these integers to Single Precision floating-point numbers. But Z80 assemblers only recognize 3 "types" of variables:

- Bytes
  - 8-bit values from 0..255 or -127..128
  - Stored in single registers like A or E
  - Stored in variables defined with the assembler's DEFB pseudo-op
- Words
  - 16-bit values from 0-65535 or -32767..32768
  - Stored in register pairs like HL or BC
  - Stored in variables defined with the DEFW pseudo-op
- "Storage"
  - Any size of memory from 1 byte up to all available memory
  - Can only be accessed by the processor using
    - "indirect" addressing like "(HL)" or "(address)"; or
    - "indexed" addressing like "(IX+d)"

What this means is that although the Z80 can work with numbers like 8-bit bytes or 16-bit words, it has no native method of dealing with floating-point numbers, unlike the math coprocessor built into modern X86 machines. So Microsoft decided on a 4-byte "storage" solution for Single Precision variables.

```
RSP     DEFS    4       ; RATE AS A S.P. VARIABLE
TSP     DEFS    4       ; TIME AS A S.P. VARIABLE
```

Notice that although we've provide space for these two single-precision variables, we're unable to initialize them to a value like we were able to do with RINT and TINT above. We're going to have to convert each of those integer values to S.P. using a ROM call.

It's probably helpful to think of Microsoft having created a kind of "floating point microprocessor" in ROM. The Z80 has registers like A—the accumulator; F—the flags; and other 8-bit registers like B, C, D, E, H, and L; and 16-bit pairs like BC, DE, HL, IX, and IY. Similarly, the "floating point processor" has at least two Floating Point "registers", and several "flag" registers, one of which we've seen already: the memory location 409AH, which I've named BERROR, and which contains the BASIC Error code (returned by PRINT ERR in BASIC). Let's talk about at least one of the Floating Point Registers. I believe that some of Microsoft's source code calls it ACC1, which is good mnemonic, since it's one of two or three Floating Point accumulators. In *Microsoft BASIC Decoded*, James Farvour calls it WRA1, or "Working Register Area 1". In my code, I'll call it that as well, since I've based my examples on Farvour's book.

Address	Integer	Single Precision	Double Precision
411D			LSB
411E			NMSB
411F			NMSB
4120			NMSB
4121	LSB	LSB	NMSB
4122	MSB	NMSB	NMSB
4123		MSB	MSB
4124		Exponent	Exponent

Where:
LSB = Least significant byte
NMSB = Next most significant byte
MSB = Most significant byte

WRA2 has an identical format.

Figure 1.7: Working Register Area layout.

Farvour refers to a second Floating Point Accumulator, "WRA2". That's mostly used for Double Precision math. In these articles we're going to stay with Single Precision. Here's how the ROM routines access two operands:

Integer

Destination Register	Operation	Source Registers
HL	Addition	HL + DE
HL	Subtraction	HL - DE
HL	Multiplication	HL * DE
WRA1	Division	DE / HL

Single Precision

Destination Register	Operation	Source Registers
WRA1	Addition	WRA1 + (BCDE)
WRA1	Subtraction	WRA1 - (BCDE)
WRA1	Multiplication	WRA1 * (BCDE)
WRA1	Division	WRA1 / (BcDE)

Double Precision

Destination Register	Operation	Source Registers
WRA1	Addition	WRA1 + WRA2
WRA1	Subtraction	WRA1 - WRA2
WRA1	Multiplication	WRA1 * WRA2
WRA1	Division	WRA1 / WRA2

Figure 1.8: Register arrangements used by arithmetic routines.

If, for example, we want to multiply the Single Precision value in "R" by the S.P. value in "T" (from Dr. Lien's code here), the value of R would need to be in WRA1 and that of T in register pairs BC and DE. Of course, we first have to get the original values into those locations. Does the ROM provide a subroutine that lets us convert the numbers in RINT and TINT into Single Precision values?

CALL 0AB1 Integer To Single

The contents of WRA1 are converted from integer or double precision to single precision. All registers are used.

```
;
; CONVERT INTEGER VALUE TO SINGLE PRECISION AND MOVE TO
; LOCAL AREA
;
       LD      A,59H
       LD      (4121H),A       ; LSB OF INTEGER 26457 (10)
       LD      A,67H
       LD      (4122H),A       ; MEN OF INTEGER 26457 (10)
       LD      A,2             ; TYPE CODE FOR INTEGER
       LD      (40AFH),A       ; SET TYPE TO INTEGER
       CALL    0ADBH           ; CONVERT INTEGER TO SP
       LD      HL,VALUE        ; ADDR. OF AREA FOR SP EQUIVALENT
       CALL    O9CBH           ; MOVE SP VALUE FROM WRA1 TO VALUE
       .
       .
       .
VALUE  DEFS    4               ; WILL HOLD 26457 IN SP FORMAT
       .
       .
```

Farvour's example code here converts 26457or 6759H to Single Precision in WRA1, and then saves that Single Precision result in a variable called VALUE. Interestingly, he uses a slightly different address for the conversion routine than shown in his title, but I've tested and used the title address of 0AB1H successfully. The sample code does raise another important point when dealing with ROM Math routines: the "Number Type Flag", or NTF.

Notice that in the 5th and 6th lines of code, we store a value of 2 at 40AFH with the comment "SET TYPE TO INTEGER". Most ROM Math routines have an NTF associated with each Working Register Area. WRA1 has NTF1 at 40AFH, and WRA2 has NTF2 at 40B0H. For Math, these NTFs should contain one of these values:

Integer	2
Single Precision	4
Double Precision	8

The value corresponds to the "size" of each numeric type: an Integer occupies 2 bytes, a Single Precision number takes 4, etc. Using this information, we now can write some user code to convert RINT ("R as an integer") to RSP ("R as Single Precision").

```
;
; CONVERT RINT TO RSP
;
; ADDRESSES:
NTF1        EQU     40AFH
WRA1        EQU     4121H           ; ACC1 FOR INT OR S.P.
;
; CODE:
CONVR       EQU     $
            LD      HL,(RINT)
            LD      (WRA1),HL       ; RINT -> WRA1
            LD      A,2
            LOAD    (NTF1),A        ; INTEGER TYPE -> NTF1
            CALL    GETSPC          ; "ON ERROR RESUME NEXT"
            CALL    0AB1H           ; CONVERT INT TO S.P.
;
                                    ; IF WE WANTED TO TRAP AN ERROR,
                                    ; WE WOULD INSPECT "BERROR" HERE
            LD      HL,WRA1         ; NOW COPY
            LD      DE,RSP          ;     WRA1 -> RSP
            LD      BC,4            ;     AS A S.P. VALUE
            LDIR
            RET
```

There! It only took us 14 pages to set up enough to actually be able to use a ROM Math subroutine. It gets easier from here on.

CONVR only converts 1 value from Integer to S.P., though. Surely we can convert the code to a more general-purpose subroutine so we can use it for both R and T in our example program. How do we tell our new general-purpose CONVRT subroutine what integer argument to use (like TINT) and where to store the result (like TSP)? What if we "pre-load" HL with the integer similarly to how we do it in the line after "CONVR" above, and DE with the address of the S.P. variable we want to store the result in? Our new subroutine could look like this:

```
CONVRT      EQU     $
            LD      (WRA1),HL       ; INT -> WRA1
            LD      A,2
            LD      (NTF1),A        ; INTEGER TYPE -> NTF1
            PUSH    DE              ; PRESERVE S.P. VARIABLE ADDRESS
            CALL    GETSPC          ; "ON ERROR RESUME NEXT"
            CALL    0AB1H           ; CONVERT INT TO S.P.
            LD      HL,WRA1         ; NOW COPY
            POP     DE              ;     WRA1 -> S.P. VARIABLE
            LD      BC,4            ;     AS A S.P. VALUE
            LDIR
            RET
```

Now we can simply convert both R and T from integer to S.P. values with 2 subroutine calls:

```
;
; LINE 20 "R= 500"
;
; CONVERT R FROM INTEGER TO S.P.
;
            LD      HL,(RINT)           ; R AS INTEGER
            LD      DE,RSP
            CALL    CONVRT              ; CONVERT IT TO S.P.
;
; LINE 30 "T= 6"
;
; CONVERT T FROM INTEGER TO S.P.
;
            LD      HL,(TINT)           ; T AS INTEGER
            LD      DE,TSP
            CALL    CONVRT              ; CONVERT IT TO S.P.
```

Done! What to do next?

```
10 REM * DISTANCE, RATE, TIME PROBLEM *
20 R = 500
30 T = 6
40 D = R * T
50 PRINT "THE DISTANCE (IN MILES) IS", D
```

Oh: we'd better set up the Single Precision variable to store D, the result of our calculation.

```
DSP         DEFS    4               ; SINGLE-PRECISION "D"
```

Line 40: do the Math! We need to multiply the S.P. version of R by the S.P. version of T. So we need one of those values in WRA1, and the other in BC and DE. Well, we just converted T from Integer to S.P., so its Single Precision value is still in WRA1. We just have to get S.P. value of D into BC (the higher 2 bytes) and DE (the lower 2 bytes). Then we can call the ROM Multiply routine.

```
;
; LINE 40 PART 1
;
; MOVE S.P. VALUE OF D FROM RSP -> BC/DE
;
            LD      DE,(RSP)            ; LOWER 2 BYTES
            LD      BC,(RSP+2)          ; HIGHER 2 BYTES
```

Let's make sure that this is a Single-Precision multiply by setting the Number Type Flag to 4. Then we can use GETSPC again to "RESUME NEXT" if there's an error, and actually do the multiplication. Finally, let's move the result to the S.P. version of D.

```
;
;    LINE 40 PART 2: SINGLE-PRECISION MATH OPERATION
;
             LD      A,4             ; S.P. NUMERIC TYPE
             LD      (NTF1),A        ;     -> NTF1
;
;    LINE 40 PART 3: DO THE MULTIPLY
;
             CALL    GETSPC          ; "ON ERROR RESUME NEXT"
             CALL    0847H           ; WRA1= RSP * TSP
;
;    LINE 40 PART 4: SAVE THE RESULT IN S.P. VERSION OF "D"
;
             LD      HL,WRA1
             LD      DE,DSP
             LD      BC,4
             LDIR
```

Now we can print out the result.

```
;
;    LINE 50: THE STRING LITERAL
DIST         DEFM    'THE DISTANCE (IN MILES) IS'
             DEFB    0
;
;    PRINT IT
             LD      HL,DIST         ; "THE DISTANCE..."
             CALL    PRINT           ; SEE TRS8BIT MARCH 2025 P. 118
             CALL    VPTCOM          ;  COMMA: MOVE CURSOR TO
                                     ; NEXT TAB STOP
;
```

Wait. What's that CALL VPTCOM? Well, Dr. Lien's code PRINTs "THE DISTANCE IN MILES IS" with a "trailing comma":

```
50 PRINT "THE DISTANCE (IN MILES) IS", D
```

As seasoned BASIC programmers, we remember that this comma will move the cursor to the next "tab stop" before PRINTing the next item. Of course, assembly language has no such provision. Here's my alternative:

```
;
;          VPTCOM – "VIDEO PRINT COMMA"
;               EMULATE BASIC'S TRAILING COMMA
;
VPTCOM      EQU     $
            LD      A,(4020H)       ; CURSOR POSITION LOW BYTE
            AND     63              ; MOD 64 = COLUMN
            CP      63-8            ; IF (COLUMN <= (63 - 8)):
            JR      NC,VPTCO1        ; CALCULATE NEXT TAB STOP
            AND     7                ; MOD 8 = POSITION IN COLUMN
            LD      C,A              ; PRESERVE POSITION
            LD      A,8
            SUB     C                ; A= # OF COLUMNS TO MOVE OVER
            LD      B,A              ; PRESERVE # COLS
            LD      A,' '            ; PRINT SPACES
VPTCO2      EQU     $                ; WHILE (B--):
            CALL    032AH             ; PRINT A SPACE
            DJNZ    VPTCO2            ; :WEND
            RET
VPTCO1      EQU     $               ; ELSE:
            LD      A,0DH            ; PRINT A
            CALL    032AH            ;    NEWLINE
            RET                     ; :ENDIF
```

Now before we can print the value of D, we need to convert it from a S.P. number to a string, and then prepare to print that. There's a very nice ROM routine that will print whatever floating-point value is in WRA1, complete with PRINT USING-like formatting if desired.

CALL 0FBE — Floating to ASCII

Converts the single or double precision number in WRA1 to its ASCII equivalent. The ASCII value is stored at the buffer pointed to by the HL register pair. As the value is converted from binary to ASCII, it is formatted as it would be if a PRINT USING statement had been invoked. The format modes that can be specified are selected by loading the following values into the A, B ,and C registers.

```
REGISTER   A = 0 ... Do not edit. Strictly binary to ASCII.
REGISTER   A = X ... Where x is interpreted as:

     7 6 5 4 3 2 1 0  = BIT
     x x x x x x x x
     | | | | | | | |
     | | | | | | | +---- EXPONENTIAL NOTATION
     | | | | | | +------ RESERVED
     | | | | | +-------- SIGN FOLLOWS VALUE
     | | | | +---------- INCLUDE SIGN
     | | | +------------ PRINT LEADING $ SIGN
     | | +-------------- INCLUDE LEADING ASTERISKS
     | +---------------- PRINT COMMAS EVERY 3RD DIGIT
     +------------------ 0 - DO NOT PERFORM EDIT FUNCTIONS
                         1 - EDIT VALUE ACCORDING TO OPTIONS

REGISTER   B = The number of digits to the left of the
               decimal point.
REGISTER   C = The number of digits after the decimal point

;
;
;
       LD       HL,AVAL1        ; ASCII VALUE TO CONVERT
       CALL     0E6CH           ; CONVERT ASCII TO BINARY
       LD       HL,AVAL2        ; BUFFER ADDR. FOR CONVERTED VALUE
       LD       A,0             ; SIGNAL NO EDITING
       CALL     0FBEH           ; CONVERT SP VALUE BACK TO ASCII
       .
       .
       .
AVAL1  DEFM     '1103.25'       ; ORIGINAL ASCII VALUE
       DEFB     0               ; NON-NUMERIC STOP BYTE
AVAL2  DEFS     7               ; WILL HOLD RECONVERTED VALUE
       .
```

By examining Ira Gold's amazing disassembly of the ROM at https://www.trs-80.com/wordpress/roms/mod-1-rom-disassembled-page-1/ I was able to determine that calling address 0FBDH (instead of 0FBEH) would simply print a floating-point number with "no editing" (ie, A contains 0). Here's how I use that to convert D to a string ready to print out:

```
        CALL    GETSPC          ; "ON ERROR RESUME NEXT"
        CALL    0FBDH           ; CONVERT WITH NO FORMATTING
```

At this point, HL holds the address of the string result, so we can just print it out.

```
        CALL    PRINT
;       ALSO PRINT NEWLINE
        LD      HL,RETURN       ; SEE TRS8BIT MARCH 2025 P. 122
        CALL    PRINT
```

That's it! We have a complete assembly-language version of Lien's BASIC program. With a bit of re-arranging, the main program code looks like this:

```
;
; CONVERT "R" & "T" FROM INTEGER TO S.P.
;
        LD      HL,RINT         ; R AS INTEGER
        LD      DE,RSP
        CALL    CONVRT          ; CONVERT IT TO S.P.
        LD      HL,TINT         ; T AS INTEGER
        LD      DE,TSP
        CALL    CONVRT          ; CONVERT IT TO S.P.
;
;   MOVE S.P. VALUE OF "D" FROM DSP -> BC/DE
;
        LD      DE,(DSP)        ; LOWER 2 BYTES
        LD      BC,(DSP+2)      ; HIGHER 2 BYTES
;
;   SINGLE-PRECISION MATH OPERATION
;
        LD      A,4             ; S.P. NUMERIC TYPE
        LD      (NTF1),A        ;     -> NTF1
;
;   DO THE MULTIPLY
;
        CALL    GETSPC          ; "ON ERROR RESUME NEXT"
        CALL    0847H           ; WRA1= RSP * TSP
;
;   SAVE THE RESULT IN S.P. VERSION OF "D"
```

```
;
            LD      HL,WRA1
            LD      DE,DSP
            LD      BC,4
            LDIR
;
;   PRINT IT
;
            LD      HL,DIST         ; "THE DISTANCE..."
            CALL    PRINT           ; SEE TRS8BIT MARCH 2025 P. 118
            CALL    VPTCOM          ;  COMMA: MOVE CURSOR TO
                                    ; NEXT TAB STOP
;
            CALL    GETSPC          ; "ON ERROR RESUME NEXT"
            CALL    0FBDH           ; CONVERT WITH NO FORMATTING
                                    ; ("D" STILL IN WRA1)
            ;
            CALL    PRINT
;
;   ALSO PRINT NEWLINE
;
            LD      HL,RETURN       ; SEE TRS8BIT MARCH 2025 P. 122
            CALL    PRINT
;
;   ** DATA **
;
RINT        DEFW    500             ; RATE AS AN INTEGER
TINT        DEFW    6               ; TIME AS AN INTEGER
RSP         DEFS    4               ; RATE AS A S.P. VARIABLE
TSP         DEFS    4               ; TIME AS A S.P. VARIABLE
DSP         DEFS    4               ; SINGLE-PRECISION "D"
;
DIST        DEFM    'THE DISTANCE (IN MILES) IS'
            DEFB    0
```

Next time: how to convert *unsigned* integers up to 24 bits long to and from Single Precision (nice for printing out memory addresses, for example). Also My Dear Aunt Sally (the Scientist, apparently), and a Math Test.

Floppy Days
Vintage Computing Podcast
with Randy Kindig

Pull out a vintage computer and compute as if it were yesterday.

Subscribe Today!

Tandy-specific Episodes:

- TRS-80 Model I
- TRS-80 Model II
- Tandy Pocket PC
- Steve Leininger and Don French interviews
- The MISE with Peter Bartlett
- CoCo
- Boisy Pitre and Bill Loguidice book interview
- CoCoFest and Tandy Assembly coverage
- and much more...

https://www.floppydays.com

https://www.soigeneris.com/universal-backpack-drive



Universal Backpack Drive

The Backpack is a small SD card 'disk drive' for vintage computers that runs from a single AA battery

Choose Options

- ☐ Preloaded 8GB uSD card [+$5.00]
- ☐ Spare CR1025 RTC Battery [+$1.00]
- ☐ Tandy WP-2 DB9 to DB25 adapter [+$5.00]
- ☐ Sinclair Z88 DB9 t0 DB25 adapter [+$5.00]
- ☐ Epson PX-8 / HX-20 adapter [+$10.00]

$65.00

IN MAV'S HISTORY WORKSHOP

What *Was* The *Second* Color Computer *1*?



by Ian Mavric

"That's a strange sounding title" I'm sure a lot of you are thinking. We all know the original Color Computer was silvery-grey, a paint scheme carried over from the TRS-80 Model I, II and III but did you know there were two distinct versions? I am going to look at the second version, designed by Paul T. Schreiber from Tandy R&D in 1982, which was sneaked onto the market in late 1982 with little fanfare.

HISTORY: THE FIRST COCO 1 - DALE CHATHAM

Well read people know the Coco 1 was released in 1980 and was an evolution of what started out as Project Green Thumb terminal, which was released eventually as the AgVision terminal, and Radio Shack released the Vide-O-Tex terminal. Add some more computery features like joysticks, serial and cassette connections, RAM and a cartridge slot and you have a home computer[1]. It was designed by Dale Chatham at Tandy R&D and was released as a 4K system with Color Basic 1.0 (catalog no. 26-3001) and as a 16K system with Extended Basic 1.0 (catalog no. 26-3002). Some time later a 32K Extended Basic version was released (catalog no. 26-3003). It was a big hit and it's motherboard went through versions A, C, D, E. 'A' versions became modified (upgraded) with a PAL encoder board and were sold in countries like Australia and Europe. Very few 'C' versions were released and the most common are 'D' and 'E'. Externally these have a Ram badge on the top, the nameplate is off to the left, and there is a bigger black border around the keyboard.

The motherboard from this era had some interesting jumpers to achieve 32K Ram marked HI/LO. It was decided the most economical way to produce a 32K computer was with half-bad 64K RAM chips which Motorola had in abundance. If you had 8 chips where the top 32K was bad you would set the motherboard jumper to LO, which locked out the 32K in the top of the Ram chips. Interestingly the 32K Coco turned out to be so popular Motorola quickly ran out of half-bad 64K Ram chips so many computers manufactured from 1982

onwards have perfectly good 64K chips. Around this time Frank Hogg Laboratories worked out a way to modify the system to Rom-Write-Enable which enabled the second 32K Ram bank, turning it into a 64K computer capable of running FLEX and OS/9. Although modifying these early Cocos for 64K was complicated with lots of trace cuts on the motherboard and bodge wires added, thousands of these were converted.

PAUL T. SCHREIBER ENTERS THE COCO WORLD

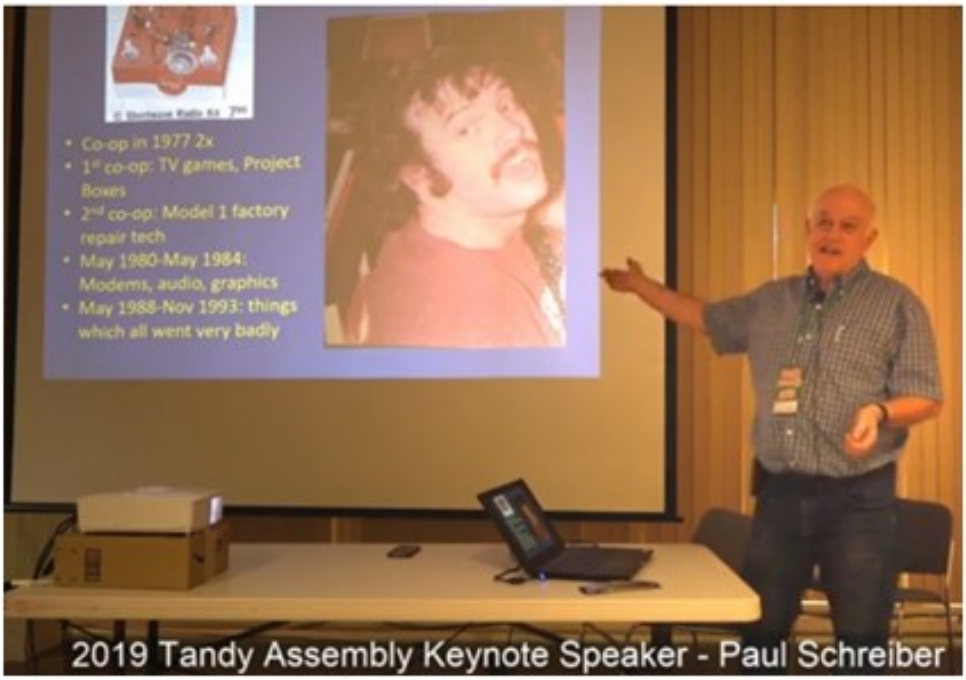


Photo: Courtesy of Tandy Assembly

Paul had been working at Tandy since 1976 and in 1982 at the age of 26 pitched to the sales department that he could re-design the Color Computer to be much cheaper to produce. Companies love when a product they sell which is already very successful can be

manufactured for less cost. It allows them to reduce the cost to buyers while maintaining profit margin. Also as electronics techniques improved rapidly at this time, allowing an overall reduction in the number of parts while maintaining performance and compatibility. Paul set to work and the first thing he did was make the motherboard much smaller, so more motherboards could be made out of a single sheet of circuit board copper clad. Even if nothing else was accomplished this would make production already significantly cheaper. Thus the 2nd Color Computer 1 came into existence.

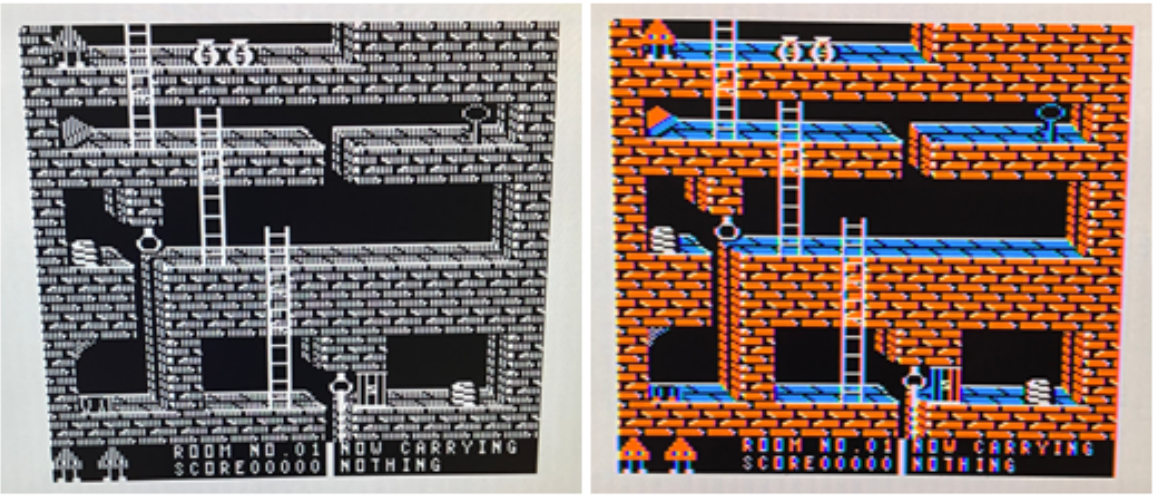
MEMORY WAS SIMPLIFIED NEXT: GOODBYE 4K COCO

One of the major improvements was to simplify the memory so that only 16K and 64K were the two options, and design the thing so that upgrading from 16K to 64K only needed jumpers moved clipping out some capacitors. The Rom-Write-Enable was also incorporated into the motherboard. 4K was dropped from the lineup. It might have been a popular entry-level computer when it first came out but by 1982 it was the least popular seller. The new entry-level Coco was to be a 16K with standard Color Basic. Since it's spec was different from the 4K Coco a new catalog number was assigned, this time 26-3004. It was sold alongside the 16K Extended Basic (now called 26-3002A), and the 32K Extended Basic (now called 26-3003A). But wait a minute? Didn't you just say it was a 64K? Yes I did, and the motherboard on these only has jumpers for 16K/64K, but Radio Shack continued to sell it as the 32K Coco. Why they did this is not exactly explained but I think possibly RS didn't want their home computer sporting more RAM than their popular Model III 48K

computer or the same amount of RAM is their extremely expensive Model II business computer. By 1983 this became less of an issue, and I'll talk more about it later.

VIDEO IMPROVEMENT GAVE A SUPERIOR TV PICTURE

The other area where the new Coco excelled was the picture on the TV screen was much sharper. Paul was rightly pleased with his effort to get a very clear signal from the MC6847 VDG out to the RF modulator to the TV set that was very sharp and extremely low on RF interference lines. This came back to bite Paul when he was asked to explain why a popular game didn't show colours in the highest resolution (PMODE4). Dale's Coco with its older design video circuits artefacted orange and blue on an NTSC screen when certain patterns were displayed, and programmers used this to add colour to what would be otherwise a black and white display. Paul's superior design fixed that "feature" but he was requested to rectify it back to artefact colours. A resistor and inductor addition was all that was needed to add a little noise into the signal, and the colours returned.



Pictures: PMODE4 game, left has no artefact colour, right has it

MOTHERBOARD REVISION NAMING CONVENTIONS:

As mentioned previously Dale's motherboards went from Revision A-E, each update being easy to identify by the motherboard ID number near the cartridge slot, just look at the last letter. Paul's board has been dubbed the 'F' board even though it doesn't say F on it. It just that is came after the 'E' so logically it's the F. It does have motherboard number which ends in 285, and some people call it the '285' board. Lastly some people call it the 'NC' board, which stands for 'No Code'. Back before digital cameras and the internet describing boards took more thoughtful explanation, however these days people who are unsure of a board revision can just take a photo and post it to the FB Coco group and someone will be able to identify it in minutes.

WHILE AT IT, LET'S CLEAN UP THE OUTER APPEARANCE

I always thought Dale's Coco 1 was the prettier machine but RS higher-ups (not Paul) decided a change to the styling was in order to a more understated look. The Ram badge was gone and the nameplate centered. The black border around the keyboard was also gone. What many people don't realise was that underneath the silvery-grey paint on Dale's Coco 1 was black plastic, and once you used the computer for a while the paint on the palm rests wore away to reveal the black plastic underneath. It didn't look great and for intense users could be worn away in as little as 6 months. On Paul's machine the case was made of the same grey plastic as the Model III so once the paint wore away it was less obvious.



Picture: a Dale Chatham Coco 1 with worn paint along the front

The 2nd Coco 1 did retain the chicklet keyboard which while maligned and rather dead to type on it was long lasting and reasonable once you got used to it. The design of the chicklet keyboard changed around this time to one with an internal membrane instead of brass springs. It was cheaper to produce and the keyboard connector on the motherboard changed from a pin header to a receptacle for a thin membrane connection.

I asked Paul if he ever brought it up with TCA to make a better keyboard for the Coco and he said "They just laughed. Other cheap computers were harder to type on (the Atari 400) and were selling in record numbers." Some time later, as we all know, eventually Radio Shack had to lift it's game in this area and so the Coco 2 and 3 got better and better keyboards as time went on.

ONE LAST TRICK UP IT'S SLEVE - THE 64K COLOR COMPUTER 1

By the middle of 1983 a few unrelated things happened which came together to change Radio Shack's mind and release an official 64K version of the Color Computer 1:

1. 64K Ram became the de-facto memory size for home computers thanks to the popularity of the Commodore 64

2. The TRS-80 Model III was discontinued so there was no need to keep the memory size at 32K for the Coco 1

3. OS/9 Level I was about to be released for the Coco and it required 64K Ram

4. The Coco 2 was on the drawing boards but it was clear that while the 16K version was ready for production, the 64K version was not going to be ready for a few more months, and a machine was needed to fill that gap for a little while.

Radio Shack was also in the phase of changing their computers from silvery-grey to white, so a white version of the Coco 1 case was manufactured, Paul T. Schreiber's motherboard was used, and the keyboard from the soon-to-be released Coco 2 was used. The resulting machine was the 64K Color Computer (catalog no. 26-3003B) and was sold for around 3 months at the end of 1983.



Photo: 64K Color Computer (catalog no. 26-3003B)

I have written about this extensively in the past so if you want to know more about this computer it's in a previous issue of TRS8Bit[2].

CONCLUSION

The cost reduction of the Coco 1 by Paul ensured the success of the computer through the heady years of 1982-3. By lowering the cost of the Color Computer to keep it relevant against stiff competition like the Commodore 64 and Atari 800XL the computer remained popular through what was soon to be the 1983 Video Game Crash[3], which affected not only video games consoles but home computers in general. Many computers with similar spec sheets went to the wall at this time, which included popular (for a time) machines like the TI 99/4A, SpectraVideo SV-318, and Acorn Electron.

Interestingly the 64K Color Computer 1 could have been left out of production altogether and the silvery-grey and chicklet keyboard of the "32K" (in name only) machine officially renamed the 64K Color Computer - as it would only require an updated nameplate. However Radio Shack took the time to refresh the look of the computer into a white case and include a decent keyboard when it didn't really need to, as the machine was only on the market for some 3 months until the 64K Coco 2 was ready.

Ian Mavric / ianm@trs-80.com

Further reading:

1. CoCo: The Colorful History of Tandy's Underdog Computer, B.G.Pitre, B.Loguidice, CRC Press 2014. ISBN: 978-1-4665-9247-6

2. **http://www.fabsitesuk.com/tandy/trs8bit_year10.pdf** Pages 179-191

3. **https://en.wikipedia.org/wiki/Video_game_crash_of_1983**



Ian Mavric is an IT Specialist who also restores and collects TRS-80's and classic cars. He lives with his wife and kids in Melbourne, Australia.


BRITAIN'S BEST-SELLING DIGITAL CULTURE MAGAZINE
PIXEL ADDICT
£6.50 ■ 2024 ■ ISSUE 19
Tandy
The Story of an Electronics Giant
PLUS
NEIL WEST OF SEGA POWER
40 YEARS OF KING'S QUEST
MULTITECH'S APPLE II CLONE
20 GOTO 10 BOOK REVIEW
20 GOTO 10



E.T. FONEHUME

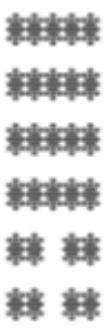
HI DUSTY. MANY THANKS FOR YOUR ASSISTANCE WITH THE I-CHING PROGRAMS! IT IS, OR RATHER CAN BE, VERY COMPLICATED. ANYWAY, HERE'S WHAT THE 'INTERWEB', NAMELY 'DEEPSEEK', HAD TO SAY.

“The I Ching, also known as the Book of Changes, is an ancient Chinese text and one of the oldest classics in Chinese literature. It is a divination system and philosophical guide that uses a set of 64 hexagrams—symbols made up of six stacked horizontal lines, each line either broken (yin) or unbroken (yang). These hexagrams represent various states and transformations of the universe.

The I Ching is used for guidance, offering insights into the dynamic balance of opposites, the flow of change, and the interconnectedness of all things. It has influenced Chinese philosophy, religion, and culture for millennia, and its principles are often applied in areas such as ethics, decision-making, and understanding natural phenomena.”

I DECIDED THAT AN ELEMENTARY ONE-LINER WOULD WORK QUITE WELL, WITH ALL THE 'FANCY BITS' SUCH AS CHANGES AND DESCRIPTIONS ETC BEING EXCLUDED.

```
10 CLEAR500:CLS:RANDOM:A$(0)="** **":A$(1)="*****":FORZ=6TO1STEP
-1:X=RND(16):H=-(X)=8)*-((X)=9)):C(Z)=H:PRINTA$(H):N=N+(C(Z)*(2↑
(Z-1))):NEXT:PRINT"HEXAGRAM (BINARY) NUMBER";N
```



```
HEXAGRAM (BINARY) NUMBER 60
READY
>_
```

AS YOU CAN SEE, IT DOES WORK, BUT IN A RELATIVELY SIMPLE WAY.

AS YOU KNOW, I THEN STARTED TO THINK ABOUT A MORE DETAILED PROGRAM, WHICH WOULD INCLUDE NOT JUST THE HEXAGRAM NUMBER BUT FURTHER DETAILS.

IT SEEMS THERE ARE MANY VERSIONS AND UNLIMITED VARIATIONS OF THE I-CHING, EACH ONE USING THE SAME 64 HEXAGRAMS BUT WITH DIFFERENT NUMBERS, DESCRIPTIONS AND NAMES, SO I DECIDED UPON USING THE 'BINARY' HEXAGRAMS IN THE PROGRAM.

I ALSO WANTED TO IMPROVE THE 'RANDOM' SELECTION IN THE ONE-LINER SO AS TO MIMIC THE MORE TRADITIONAL METHOD OF USING 'YARROW STALKS' AND, HOPEFULLY, PROPERLY IMPLEMENT THE I CHING PROBABILITIES (7:7:2 RATIO FOR OLD YIN, OLD YANG, AND CHANGING LINES).

THE PROGRAM IS, OBVIOUSLY, SOMEWHAT LARGER, BUT AS YOU CAN SEE, IT PRODUCES MUCH MORE INFORMATION. AT THIS POINT, YOU WILL NEED TO CONSULT 'THE BOOK OF CHANGES' (PAGES FURTHER ON) TO DECIPHER JUST WHAT IT ALL MEANS. EVEN THEN, I'M NOT SO SURE I CAN UNDERSTAND IT ALL.

I CAN'T PROMISE THAT THE REAL 'TRUTH' WILL BE REVEALED AS THE PROGRAM IS INTENDED TO BE USED JUST FOR YOUR OWN AMUSEMENT. YOU ARE INSTRUCTED TO THINK OF THE PROBLEM FOR WHICH ANSWERS ARE REQUIRED AS YOU THROW YOUR STALKS (OR IN SOME CASES COINS). PERHAPS THINKING, WHEN YOU PRESS THE <ENTER> KEY, WOULD TURN OUT TO BE JUST AS SUITABLE?

THANKS FOR THAT E.T. I KNOW YOU SPENT QUITE SOME TIME ON IT ALL. I WOULD GUESS IT'S ABOUT AS ACCURATE AS MY GREAT AUNT DOROTHY WOULD ACHIEVE READING THE TEA-LEAVES FROM YOUR TEA-CUP, BACK IN THE 1950'S (WHOM I WAS LEAD TO BELIEVE AT A VERY EARLY AGE, WAS A "TRUE MYSTIC WITH REAL PSYCHIC ABILITY").

IF ANYONE WOULD LIKE A COPY OF THE FULL PROGRAM, PLEASE EMAIL ME AT THE USUAL ADDRESS AND I'LL EMAIL YOU A .TXT FILE WHICH WILL JUST PASTE INTO TRS80GP. LIFE'S SO MUCH EASIER WITH AN EMULATOR, THANKS TO GEORGE AND PETER.!!

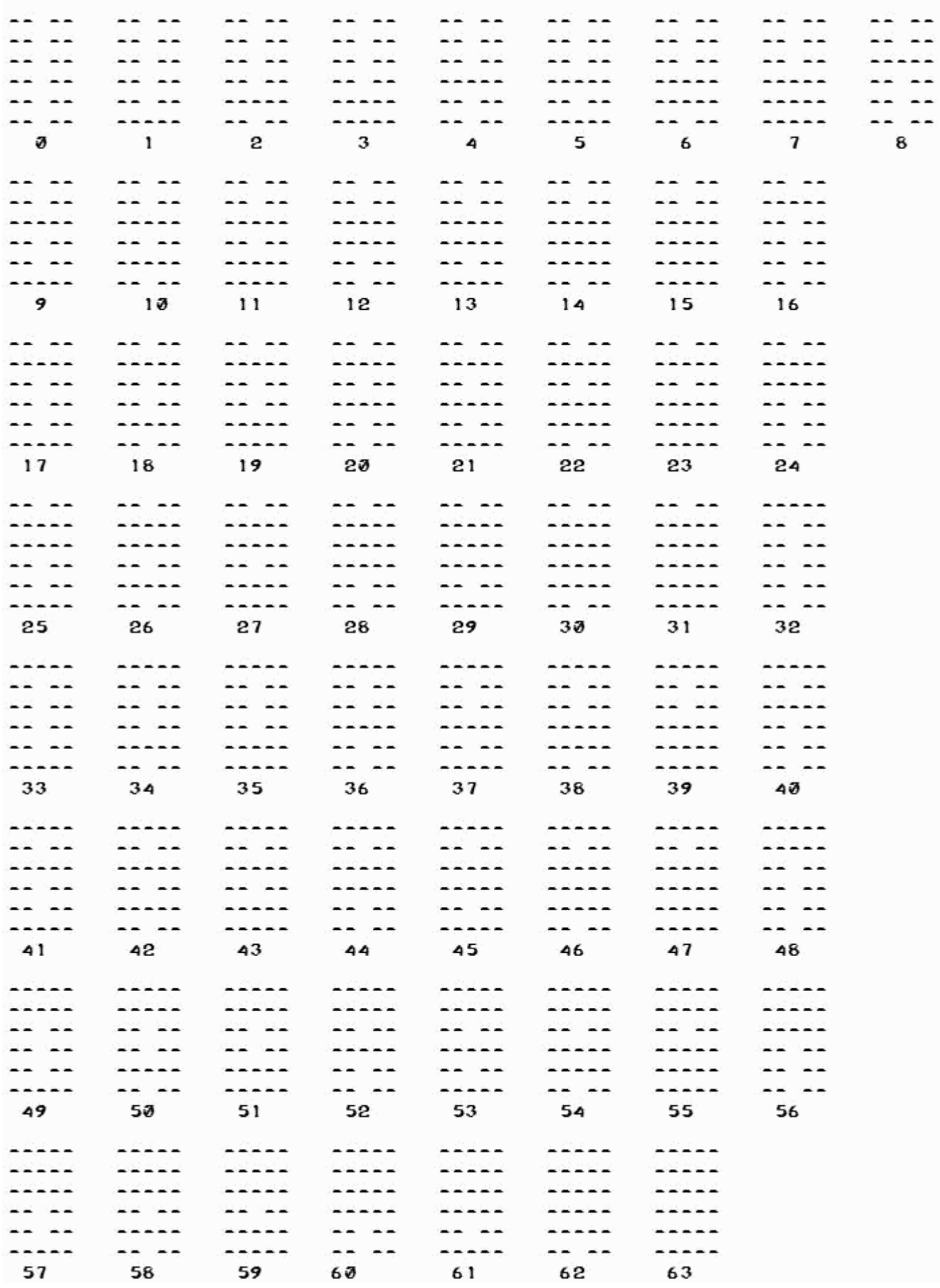
```
#####          #####
## ##          ## ##
# 0 #          ## ##
## ##          ## ##
#####          #####
#####          #####
```

PRIMARY HEXAGRAM NUMBER - 43
TREADING

CHANGED HEXAGRAM NUMBER - 35
REVOLUTION
READY
>_



I-CHING BINARY HEXAGRAMS IN SEQUENCE



```
100 REM  ### I CHING HEXAGRAM GENERATOR (BINARY SEQUENCE WITH NAMES, (PINYIN)
110 REM  ### NUMBER AND ENGLISH MEANING)

120 CLS:CLEAR500:RANDOM:DIMA$(64),N(64)
130 FORX=0TO63:READN,A$:N(X)=N:A$(X)=A$:NEXT

200 REM ### GENERATE PRIMARY HEXAGRAM
210 FORL=6TO1STEP-1
220 R=RND(16):G=-(R>=7)*-((R>=8)+(R>=15)):G=G+1
230 ON G GOSUB600,700,800
240 NEXT
250 GOSUB 300
260 REM ### CHANGE REQUIRED?
270 IF DD=1THEN GOSUB900ELSEPRINT"(NO CHANGES)"
280 END

300 REM ### CALCULATE HEXAGRAM NUMBER
310 FORZ=6TO1STEP-1:V=V+(H(Z)*(2[(Z-1))):NEXT
320 PRINT`448,"PRIMARY HEXAGRAM NUMBER -";V:PRINT@512,A$(V)
330 RETURN

600 REM ### PRINT YIN LINE
610 PRINT"** **":H(L)=0:H2(L)=0
620 RETURN

700 REM ### PRINT YANG LINE
710 PRINT"*****":H(L)=1:H2(L)=1
720 RETURN

800 REM ### CALCULATING CHANGE(S)
805 DD=1
810 H(L)=RND(2)-1:IFH(L)=0THENPRINT"* X *":H2(L)=1ELSEPRINT"* 0 *":H2(L)=0
820 RETURN

900 REM ### PRINT 2ND HEXAGRAM WITH CHANGE LINE(S)
905 PP=16
910 FORZ=6TO1STEP-1:IFH2(Z)=0THENPRINT@PP,"** **";:PP=PP+64:NEXTELSEIFH2(Z)
=1THENPRINT@PP,"*****";:PP=PP+64:NEXT:PRINT`576,""
920 REM ### CALCULATE CHANGED HEXAGRAM NUMBER
930 FORZ=6TO1STEP-1:V2=V2+(H2(Z)*(2[(Z-1))):NEXT
940 PRINT@640,"CHANGED HEXAGRAM NUMBER -";V2:PRINT@704,A$(V2)
990 RETURN

2000 REM ### BINARY I CHING SEQUENCE (NUMBER,(PINYIN),ENGLISH DESCRIPTION
2010 DATA 0,"(KUN) the receptive",1,"(FU) pushing upward",2,"(SHI) return"
2020 DATA 3,"(LIN) the army",4,"(QIAN) gathering together"
2030 DATA 5,"(MING YI) the clinging fire",6,"(SHENG) progress"
2040 DATA 7,"(TAI) pervading",8,"(YU) stagnation",9,"(ZHAN) modesty"
2050 DATA 10,"(XIE) enthusiasm",11,"(GUI MEI) the arousing thunder"
2055 DATA 12,"(XIAE GUO) deliverance",13,"(FENG) before completion"
2060 DATA 14,"(HENG) the wanderer",15,"(DA ZHUANG) duration"
2070 DATA 16,"(BI) holding together",17,"(ZHUN) difficulty at beginning"
2075 DATA 18,"(KAN) exhaustion",19,"(JIE) nourishment"
2080 DATA 20,"(JIAN) contemplation",21,"(JI JI) dispersion"
2090 DATA 22,"(JING) inner truth",23,"(XU) development",24,"(CUI) obstruction"
2100 DATA 25,"(SUI) keeping still mountain",26,"(KUN) work on the decayed"
2105 DATA 27,"(DUI) the well",28,"(XIAN) the gentle wind"
2110 DATA 29,"(GE) limitation",30,"(DA GUO) the abysmal water
2120 DATA 31,"(GUAI) youthful folly",32,"(BO) dreams",33,"(YI) retreat"
2130 DATA 34,"(MENG) infulence",35,"(SUN) Revolution",36,"(GEN) coming to meet"
2140 DATA 37,"(BI) preponderance of the great",38,"(GU) the cauldron"
2145 DATA 39,"(DA XU) possession in great measure",40,"(JIN) opposition"
2150 DATA 41,"(SHI KE) the marrying maiden",42,"(WEI JI) the joyous lake"
2160 DATA 43,"(KUI) treading carefully",44,"(LU) conflict",45,"(LI) innocence"
2170 DATA 46,"(DING) following",47,"(DA YOU) oppresion",48,"(GUAN) conflict"
2180 DATA 49,"(YI) preponderance of the small",50,"(HUAN) influence"
2190 DATA 51,"(ZHONG FU) obstruction",52,"(JIAN) waiting",53,"(JIA REN) after completion"
2200 DATA 54,"(XUN) grace",55,"(XIAO XU) darkening of the light",56,"(PI) progress"
2210 DATA 57,"(WU WANG) enthusiasm",58,"(SONG) contemplation"
2220 DATA 59,"(LU) holding together",60,"(DUN) splitting apart"
2230 DATA 61,"(TONG REN) the receptive",62,"(GOU) stagnation",63,"(QIAN) the creative"
```

Please note that the '[' in lines 310 and 930 represent the up-arrow in Level 2 BASIC - Ed.

Binary I Ching – Book of Changes – the 64 Hexagrams

Hexagram 0: The receptive

KUN is the image of a bird with its wings about to fly up. However, it is not the right time to begin. It is wise to fly down to the mountain to the safety of the nest.
For business, it implies that it's not a good time to push forward for expansion or to start a new venture. Aiming too high yields no reward. Stick to your daily routine, tend to details, and avoid risks. Keep a low profile and concentrate on small challenges. By doing so, your shortcomings will be viewed as small mistakes that can be easily forgiven.

Hexagram 1: Pushing upwards

FU means great savings, accumulation, and abundance. It also means power, knowledge, successful, and a bright future.
You have patiently stored up great creative energy and you are ready to embark upon a major project. Marshall your resources and begin.
Move ahead without hesitation. Much can be achieved now. Great success lies just ahead generating your wealth to a great height.

Hexagram 2: Return

SHI means to return or restart. It the time of transition; a time for the old ways to be discarded, so a new way can come into existence.
It is also a sign of growth and a turning point. New relationships with like-minded people will be formed.
This period is strengthened by rest. Your luck will start to improve and things will get better. Put aside pride and learn from those who know more than you. Your energy will become stronger to face any new challenges ahead.

Hexagram 3: The Army

LIN An army runs best on strong and wise leadership. To achieve your goals, you must marshal your resources and the people around you. You have to assume leadership of this particular situation. Most importantly, you must know exactly what it is you wish to achieve. Without clear goals and strategy, you will fail.
You are strong and wise enough to overcome difficulties. Share the good and bad with your followers at the start to gain trust. A dangerous situation requires the most experienced leader to maintain discipline. Employ only the best of them or your effort will be wasted.

Hexagram 4: Gathering together

QIAN means to assemble, gather, and come together. People gather and rejoice with delight. The advantage is you must come through firm corrections. They communicate and respond sincerely to each other with their heart.
It is time to form strong bonds by sharing a common vision with a common goal. Great strength is achieved when people are united with a single purpose.
A firm leader is needed to guide this increased power to unite factions. Good fortune is indicated, but always prepare for the unforeseen.

Those who have ulterior motives shall never last long in the group. As a leader, you gain support not from your position, but from your commitment to the group.

Hexagram 5: Clinging fire

MING YI implies intelligence, brightness, and beauty. The double fire signifies bright sunlight, which indicates a moment of happiness, success, and recognition.
Fire needs fuel, otherwise it will die off. But if there is too much fire, the outcome can be dangerous.
Things will go smoothly for the honest and upright person, but for those with ill intentions, failure will prevail. Serious focus is required before any action.

Hexagram 6: Progress

SHENG means to advance or proceed. It is a time of new activity and new relationships. It's also a new day and beginning of action. The prospects for the future are very bright. You will achieve a high level of success in business and your career.
For this good fortune to continue, one must possess behaviour that is uncorrupted, for even the greatest fortune can dissipate immediately by negative motivations.

Hexagram 7 Pervading

TAI A time of peace, harmony and spreading abundance. Offer what you have to your highest ideals. Beware of greed, malice and mean-spirited actions. Follow the natural cycles with gratitude and the blessings will flow.

Hexagram 8: Stagnation

YU It is not a good time to start a new venture. People obstruct, negative influences abound, and communications break down. Withdraw into yourself to escape further problems.
If you feel that you have the power and the authority to instigate change, you will be successful. The standstill is ready to be overcome, but it won't end on its own. Some effort on your part is required.
This is a turning point for those who have risen to power. Exercise great caution during this period of transition. If you cannot make your influence count, withdraw to avoid humiliation.

Hexagram 9: Modesty

ZHAN Prosperity and good fortune will come if you stay humble and sincere. Do not undervalue yourself or over-promote yourself.
Your natural self will attract support and your words and actions speak for themselves. Modesty is not a weakness. Act firmly if necessary.
Stay objective and you will see the way forward even in difficult circumstances. You can ultimately achieve something good.

Hexagram 10: Enthusiasm

XIE The times are in tune with your undertakings. The confidence you display attracts supporters. Make sure your words and actions stay appropriate to their concerns in order to avoid resentment.

It also suggests a time for new ventures, change of jobs, or travelling. But one must make careful preparations in advance and proceed with caution. Good fortune can be maintained.

Hexagram 11: The arousing thunder

GUI MEI is the image of two thunder creating shock and fear. It implies natural disaster, loss of wealth, conflicts, and violent acts. A sudden unexpected shock may come into your life. It causes people to shiver in fear and take a stand.
Do not panic and use this moment to revitalize your situation. It is an opportunity to see what is no longer sustainable in your life. Do a deep self-cleansing and clear the way for a better future.

Hexagram 12: Deliverance

XIAE GUO represents supreme achievement, prosperity, and abundance. Share your abundance with others as they will be uplifted by your personality. This cannot last forever, but enjoy the moment rather than be anxious about the future.
Any problems in your life can be settled now, but beware: Everything declines after reaching the prosperity peak. Therefore, it is necessary to guard against being over-confident. Success comes fast, but it can also vanish quickly.

Hexagram 13: Before Completion

FENG represents a time when order has become disorder. The Yin and the Yang lines are not at their proper position. It means both Water and Fire cannot work together.
Your groundwork is not yet finished and rushing to completion now is dangerous. Take extreme care by taking things one step at a time. Being impulsive and careless in conducting affairs will result in failure. With caution, progress and success arrives. Recognition also comes your way.

Hexagram 14: The Wanderer

HENG signifies travelling. You are in a new territory dealing with the unknown. A traveller who left home and lives as a stranger in an unfamiliar place, trying to adapt to the new environment, the language, and the customs of the locals.
With no circle of protective friends, you have to stay watchful, ready to defend yourself against humiliation. You must also remain reserved in order to achieve any progress.
Avoid drawn out conflicts and resolve them quickly. In an unfamiliar situation, you must be confident to play the role of a humble learner and observer.

Hexagram 15: duration

DA ZHUANG means advancement and moving upward. When you plant your seeds, the tree will start to push to break out from the earth, gradually growing into tall tree.
This indicates that you need to be patient and slowly, this will lead to fulfilment. Progress any further and it will be fruitless. When attempting to ascend further, it is beneficial to guard against over-confidence. You will eventually push upward from below to supreme

success. Your relationships are beneficial and those who are above you will assist you.

Hexagram 16: Holding Together

BI This is a time of unity and leadership that promises success. The leadership requires wisdom, consistency, strength, and honest intentions.
Ask the I Ching again if you have the qualities necessary to be at the centre of this group. If you feel you do not possess those qualities, you should seek another group to join.
The formation of a strong relationship requires absolute sincerity. This will bring unexpected good fortune.
Convene only with those who join you of their own accord. This will lead to fuller commitment. If you have missed the opportunity to unite with the right people, you will feel deep regret.

Hexagram 17: Difficulty in the Beginning

ZHUN A new situation is arising and the unknown lies ahead. The initial confusion and discomfort you face can be overcome by persistent efforts.
With the advice of someone more experienced in the field, you can walk through the difficult moment with ease. Do not hold back from asking for help due to your pride. Be patient when seeking mentorship.
Advance with caution initially. Do not press the matter and avoid moving forward recklessly. This will bring you ultimate success.

Hexagram 18: Exhaustion

KAN means being trapped, difficulty, distress, and suffering. This hexagram shows the lake draining dry of water; an image of exhaustion. It indicates that there are obstacles everywhere and progress is not possible.
One is trapped in a dangerous situation. Your energy is weak and your words will be wasted. There is obstruction and corruption. No one trusts each other due to the fear of being betrayed.
Proceed with extreme caution or retreat and wait for your luck to change for the better. You are encountering difficulties, but some opportunities may still exist.

Hexagram 19: Nourishment

JIE Imagine a mouth chewing food, providing nourishment to the body. Good food will nourish the body, but spoiled food will bring harm. Sickness may come due to lack of proper care of your health. The mouth gives voice to our inner thoughts, so it is necessary to be discreet in your speech as it often can be the cause of many problems.
be careful with one’s words. In this way, you can discover who is wise and who is not. Seek out wise advice. Do not attempt a difficult task now.

Hexagram 20: Contemplation

JIAN You have gained an understanding of how things really are. The work of establishing your position still remains to be done. If you act as a detached observer with a broad view, you are less likely to be misunderstood.

Guan signifies self-examination and introspection. Take this period of time to reflect upon yourself and observe the situation carefully before taking any action. Put yourselves in another's position and you will understand and gain more.

Hexagram 21: Dispersion

JI JI represents dispersion or dissipation. It has the image of wind and water. These two forces are not mutually integrated. A stiring dissolves obstacles in your path.
Misunderstanding can now be resolved. Your personal strength and rapport with others enable you to dissolve conflict quickly.
Working with others is beneficial. It indicates that progress and success will be attained by remaining true to what is right and good.

Hexagram 22: Inner truth

JING hexagram depicts that you have a sincere and true friend; someone whom you can be trust and depend on. It addresses the relationship of a superior treating his subordinates with truth and sincerity.
If your heart is free from prejudice, you are certain to succeed in whatever you do. Good fortune lies ahead. However, if your inner mind harbours bad intentions, beware of the consequences which are revealed through your own discretions.
Understand the basic truth of the matter and do not rush to judgement when dealing with other people's mistakes or shortcomings. Without sincerity, friendship will not last.

Hexagram 23: Development

YU represents addition or increase. This is a good time to undertake new works. One has to be familiar with the wisdom of giving in order to benefit others.
Your luck will improve if your success prompts a sense of generosity towards others. Imitate good practices and correct the bad ones.
In business, it's a time for expansion. This period does not last forever, so make the most of it.

Hexagram 24: Obstruction

CUI This hexagram often refers to a legal problem and a conflicting issue. There is a possibility that you will be entangled in a legal lawsuit with someone or experience something that is creating obstacles for you.
Be firm and remain considerate. You can reach your objective if you are able to remove or overcome obstruction.
Compromise with the conflicting parties for solutions to resolve the issues. Stay positive and determined and deal with the difficulties one by one until you have removed the problem for good.

Hexagram 25: Keeping still mountain

SUI means gradual advancement or progress. This hexagram is a picture of trees growing on a mountain. In order to withstand its exposure to the elements, the tree must remain strong and firmly rooted. Such stability can only be achieved through slow and steady progress.
Hasty action is unwise. Resist the impulse to force issues. It is important to be patient and move forward one step at a time. Also

establish good relations along the way and that will lead you to success.

Hexagram 26: Work on the Decayed

KUN You are faced with a situation of disorder and decay. It reflects to a point where you have reached rock bottom; a time of ruin, corruption, and bankruptcy.
It may also refer to an older woman getting involved with a younger man, resulting in relationship grief and distress.
Dealing with change will be painful and met with resistance. However, if your course is corrected by casting out the old, it can bring you a new beginning. This leads you to success by reversing your fortunes.

Hexagram 27: The Well

DUI You have everything you need within you already. No matter what the situation is, you can achieve anything. But you need to pursue this course of action and dig much deeper to find out more in order to get things done. If you proceed carelessly, you will experience difficulties.
Jing is a well that stores water. If the well is clean and well maintained, the water that you draw from it will be nourishing. But if it is dirty and neglected, the water will bring illness. The value of it depends on how well you maintain a constant supply of clean water.

Hexagram 28: Gentle Wind

XIAN Double layer of wind means gentle and penetrating. From the gentleness of wind, one must learn how to give and take. From the force of the wind, one can learn how to balance and control oneself. It is an energy which acts by invisible ways yet achieves visible results.
In nature, it is the wind that disperses the gathered clouds, leaving the sky clear. It is suggestive of news travelling far and wide.
To get things done, you have to follow the right orders. Your ultimate goal must be clear in your mind before your start. It will take time, but you can achieve enduring results.

Hexagram 29: Limitation

GE This is a time of transition when stress has become too great to bear. There is a serious imbalance here. Adjustments have to be made quickly but not forcefully to relieve the situation.
It implies that your ambitions surpass your capabilities; you have over-expanded your business or you are living beyond your means and are facing constant setbacks. To overcome this, you need to be brave and resolute.

Hexagram 30: The abysmal water

DA GUO represents danger; a hazard that you must quickly escape from. Water is a very powerful force which can cause floods and people to drown.
It is a sign of bad luck with unexpected pitfalls. You are in a dreadful situation where you cannot advance or retreat. You need to have a calm mind to effectively overcome the difficulty. The danger can be surmounted with your confidence and skill.

Do not delay and do what has to be done. Water brings wealth but if it is too much, water can be a destructive force.

Hexagram 31: Youthful Folly

GUAI There are two meanings here. The first one indicates that you have asked the same question too many times to deserve any further answers. You are not sincere in asking and the oracle can only answer correctly the first time only.
The second meaning suggests that you have much to learn from the experienced if you want success. Be wary of being overly ambitious and be more realistic. Seeking instruction like a curious child will bring you good fortune.

Hexagram 32: Dreams

BO refers to a series of images with unusual vividness, clarity, order, and significance, sometimes seen in sleep or a trance. The pathway forward can be seen but beware of false images that create anxiety or offer easy gains. Be warned, great experience is often required to interpret.

Hexagram 33: Retreat

YI Yield to an opposing force by small adjustments is often the best way of achieving your future goal. If things are not in your favour, it is better to retreat rather than to advance.
For those in business, it is advisable to cut losses and not proceed any further. Also, it is not a good time to start new ventures or to expand. As a tactical move, retreat and conserve your energy for a new undertaking. This way, less people will suffer. Retreating is not regarded as defeat.

Hexagram 34: Influence

MENG This signifies the start of new growth and power. Opportunities arrive and success is certain. But you must make the most of it as it does not last forever. You have great influence now and must make sure you take care of those around you.
Lin means to expand, increase, and advance. The image of this hexagram is of “mother” and “daughter,” whereby the daughter is being well taken care of with guidance from the mother.
You should share your knowledge and experience generously as this will benefit everyone and bring great progress.

Hexagram 35: Revolution

SUN means to reform, change, or transform. Water and Fire in nature act against each other. It can be compared to two sisters of total different characters and ambitions that stay together even in disagreement.
It describes a group of people having different opinions. The conflict can be resolved in a new situation with drastic change. This requires the right leader and right timing, and must be carried out in the right way for success.
Be free of selfish aims and discuss the issues. This will win the group over. Great progress and supreme success will arise from the change.

Hexagram 36: Coming to meet

GEN Two or more people come together with open and mutual aims. The bond between them creates a new power if there is a clear direction. Avoid being selfish towards others, for fellowship comes only through common goals. The fellowship of people will give you the combined strength and cooperation to achieve all great things in life.
Hexagram 39: Possessions in great measure
DA XU This is a period of outstanding success, achievement, wealth, and prosperity. You are able to combine clarity of mind and inner strength to achieve your goals.
Being kind to others will reap bigger harvest. Being humble to others will draw heaven's blessings.
Your actions are in accordance with the present moment. Everything will go your way as people trust you even when you make unexpected decisions.

Hexagram 37: Preponderance of the great

BI means strength and power. You have reached a position of the power and are in excellent position to get things done. Ensure that you channel this power in the right way and you exercise it both impartially and unselfishly.
Excessive force and over-confidence may be offensive to others. Stay open to advice and insist on doing what is right.

Hexagram 38: The Cauldron

GU You have the opportunity to shine. Your abilities and your inner strength come to light. Be yourself and favourable results will flow. Be aware that the mystery of the universe is hidden within everyday experiences.
Ding is the symbol of the Cauldron, which dispenses spiritual wisdom and knowledge. The image is a boiling pot and wood fuel below with Fire on top.
In ancient times, it was used for a loyal feast to offer to heaven and ancestors. It was then shared by loyal warriors and talent officers as a token of respect from their emperor.
There's a need to follow the right path so that perfection can be achieved. Supreme success is indicated.

Hexagram 39: Possession in great measure

DA XU represents the growth of peace between opposing factors. Good fortune arrives. Heaven and Earth united, brings you peace and harmony. All things bloom and prosper. This hexagram is very auspicious as it indicates everything will go well as expected. The future looks good and business will flourish.
Success will be easy to come by. Small investments can result in big profits. Small effort will bring big rewards. Enjoy your good fortune while it lasts, all while being aware that all prosperity is followed by decline.
Do not boast of your good fortune and instead offer genuine assistance to others. By doing this, others will support your plans and the outcome will be successful.

Hexagram 40: Opposition

JIN Opposition to your plans arises within your group. Be aware that opposing views can be useful. Forcing the matter brings further opposition, but agreement is possible.
Fire burns upward and water from the lake flows downward. The theme is conflicts, contradictions, and oppositions.
It also means that although two daughters live together, they can be different in their character and views which can result in disagreement. Misunderstanding can be solved through rational thinking and wisdom.

Hexagram 41: Marrying Maiden

SHI KE hexagram depicts a young concubine being brought into a traditional household where her fate is uncertain. She finds herself in a subordinate role and somewhat taken for granted because her rank is well beneath that of a principled wife.
One has to prepare at all times and to accept the new fate with a subdued mind. Do your best to adapt to new surroundings and dispel any unrealistic dreams.
Your success is entirely dependent upon how you conduct yourself. Recognize that any assertiveness on your part will be unsuccessful and may even bring misfortune.

Hexagram 42: The Joyous lake

WEI JI symbolizes joy, pleasure, and satisfaction. Joy comes from within while pleasure comes from the outside world. Carried to extremes, it can cause your harm. The time is favourable for your undertaking. Your good mood and perseverance is infectious and attracts others.
Dui also represents a young girl bringing joy and happiness to the household. It is also associated with speech. One has to speak with wisdom, share their goodwill and good humour, and not gossip or criticize.

Hexagram 43: Treading Carefully

KUI You tread on the tail of the tiger, but it does not bite you. You are in a tricky situation that calls for care and caution. Stay with the old and tested ways, as you are dealing with powerful people. Treat them with respect to keep them on your side. Success comes if you act tactfully and patiently.
A good amount of progress is made when you advance without making demands on others. Avoid a high profile and be independent. Always keep a sharp awareness of your position. If you are aware of all the dangers around you, a successful outcome is assured.

Hexagram 44: Conflict

LU You feel you are right but face conflict. Make yourself clear at the very beginning of the relationship, for the roots of conflict lie here. Consider what you want to achieve and avoid confrontation.
Seek to compromise instead of to win. Because even if you win, you will not gain anything out of it. Your opponent has superior strength, so

surrender and retreat now. You can only find lasting peace by turning back.
If you insist on conflict to the end, the victory is not worth it. Others will have no respect for you and the conflict will continue. It is easy to avoid a dispute now in order to gain a favourable outcome later.

Hexagram 45: Innocence

LI has the meaning of innocence and simplicity. Carelessness will lead to harm. You must ride the wave to make progress, but also to benefit others.
Adapt to circumstances and avoid opposing the natural laws of right and wrong. Children should listen to parents and followers should listen to their leader.
With proper action and behaviour, you will be led to success.
Concentrate on the quality of your work now, not the eventual goal.

Hexagram 46: Following

DING means to follow, to obey, and to learn from others.
Keep an open mind and be adaptive to suite the situation by adjusting your approach. Avoid the use of force or deceit when obtaining followers.
People will follow only those whom they trust willingly. Respect other views and do not insist on your own opinion or else this will result in lost opportunities. You must make room for new things in order to succeed.

Hexagram 47: Oppression

DA YOU represents restriction, control, or restraint. The image is of destruction created by a flooded river. When too much rain is received, the lake will overflow. If there is insufficient rain, the lake will dry up.
Therefore, this is a hexagram that stresses the need to live within your limits. Jie addresses proper control and self-restraint to be properly executed in order to benefit all parties. Set your limits and live a balanced life.

Hexagram 48: Conflict

GUAN represents the way of dealing with those people who do wrong. This hexagram describes a situation when suppressed anger and frustration finally break out. Since confrontation is unavoidable, we have to decide how to deal with the situation before it reaches its tipping point.
In human affairs, one should not use excessive force. You also shouldn’t allow anger to take centre stage. Use reason and good will to secure your progress.

Hexagram 49: Preponderance of the small

YI The prospect of success is signalled, but no major moves can be made now. You want to advance, but this is not in harmony with the moment. Obstacles remain in your way.
Remain patient and prepare the foundation for the changes that are on their way. Your patience will be rewarded, so do not force any issues

now. Hold back, so you can avoid needless conflict. Return to a position where you are free to advance or retreat. Good fortune will follow.
Success comes from the accumulation of small effects. Be content with what you have achieved. Any further advancement will bring misfortune.

hexagram 50: Influence

HUAN means to feel and respond; feeling what you have to do with the heart. To accept a partner now brings good fortune. It is the law that compatible elements are drawn together.
It also suggests a bonding like a marriage between a couple or a merge of two companies. You attract people by being approachable, not by pushing for a connection.
Your sincerity and suitability will become obvious to them without any work on your part. Be yourself and those that belong in your life will accept you.

Hexagram 51: Obstruction

ZHONG FU is one of the dangerous hexagrams with the image of a destructive waterfall. This indicates danger, obstruction, and uncertainty. It also implies that many obstacles are blocking your path right now and there's no way out.
In divination, it means that moving either forward or backward is a challenge. At this time, it would be wise to look for someone who can assist you. This period of hardship must be faced with caution.

Hexagram 52: Waiting

JIAN Some outcomes simply take time. They cannot be hurried. Attempting to force a result, leads to unfavourable results now.
Take an objective view of any situation and try to understand why waiting is the best option. Exercise cheerful patience in order to achieve success when the time arrives. Stay calm and stay within your normal routine. If you rush ahead, this will only worsen the situation. Disagreement and gossip will arise and you will become entangled in a troublesome position. Look for a way out and act with care. If help comes unexpectedly, kindly accept it. This will save you from further complications.

Hexagram 53: After Completion

JIA REN represents completion and successful accomplishment. All the Yin and Yang lines are at their proper placement. This suggests success and fame.
Keep a close eye on details. Stay tuned to any possible problems arising and take nothing for granted as you consolidate your gains. Life has reached a stage when all things seem too perfect. It warns us to take precaution for when fortune reaches its peak, signs of decline will appear.

Hexagram 54: Grace

XUN signifies grace and beauty. It refers to the rising sun: luminous at dawn but slowly fading away in the evening.

Allowing your light to shine is helpful, but it is not enough to achieve great success. Your personality is a great asset most of the time, but it is not the only thing that counts.
Inner qualities are more crucial. A true relationship is much more important.
Do not just perceive things or people at a surface level. Be cautious and avoid being cheated by others based on external appearances. Only small things can be achieved now. Major issues have to wait.

Hexagram 55: Darkening of the light

XIAO XU depicts the extinguishing of light. The sun has descended below the horizon and now there's darkness. There is evil and danger everywhere.
Do not engage in conflict as this will lead to loss. Conceal your abilities for now and be patient for the impending renewal. The time will come and you will emerge into broad daylight once again.

Hexagram 56: Progress

PI means removal of obstacles, difficulty, and burdens. A period of tension and complication is over. Progress has been blocked but the conditions now favour progression in your projects.
However, any unresolved issue needs to be addressed, for that's the only true way that progress can be made. It is time to settle any existing conflicts and disputes.
If changes are required, do not hesitate to move quickly. The period of hardship is finally over now.

Hexagram (57): Enthusiasm

WU WANG symbolizes perseverance and consistency. Enduring success is based on keeping abreast of time and changing it without abandoning your goal. It's based upon standing firm yet remaining flexible like a tree bending in the wind but not breaking. This applies equally to personal and professional relationships.
A blissful and enduring marriage and the active and the passive work harmoniously together leads to both progress and success.

Hexagram 58: Contemplation

SONG Relationships depend on the understanding of different qualities of those involved and the recognition of their complimentary roles. This hexagram exhibits the function of husband and wife—both adhering to their roles and performing their duties by following a set pattern and rules.
Great respect is shown to those whom are familiar with creating a strong foundation. Remember that words without matching deeds have no effect.

Hexagram 59: Holding together

LU means decreasing, suggesting financial loss, or experiencing a setback in business. This oracle is here to help you plan ahead by giving you warning in advance: A rough time lies ahead.
A temporary setback makes you into a stronger and better person. Gradually, this opposing tide will turn back and success will slowly

come your way. If you have to take a lesser role now, so be it. This difficult time will pass. From this point forward, the only way is up.

Hexagram 60: Splitting apart

DUN means an unexpected encounter or a casual meeting. An apparent harmless element comes forward. However, it is stronger than it seems. Danger presents itself if you allow this element to have power over you.
This Hexagram indicates a situation whereby a single lady is surrounded by many suitors. It can also indicate a woman who appears to be gentle, weak, and harmless at the surface, but ends up possessing a lot of power.
For divination, it may be applied to both genders. Yin to Yang or Yang to Yin. Those who desire and fall for his or her charms could encounter heartbreak. Maintain your integrity and do not allow yourself to be seduced by such influences.

Hexagram 61: The Receptive

TONG REN You are in the early stages of a new condition that needs to be carefully nurtured in order to flourish. It is not a time to push forward without consultation. If you can develop a clear idea of your position, you have the potential to achieve great success.

Hexagram 62: Stagnation

GOU Both the upper and lower trigram represents the Mountain. It implies that your path is being blocked and obstructed by a mountain, which is hindering your progress.
It is time to stop, rest, and be calm. There is no point in taking any action as the time is not right for movement. Forget about the past and don't fret about the future. You need to focus and be on guard. Otherwise, it will lead to misfortune.

Hexagram 63: The Creative

QIAN It's a very favourable time because everything goes your way. You are proceeding in the right direction at the right time. Others will look to you for confident leadership and you will benefit from them in return. Success is assured if your motives are earnest.


Joe Kay
Director
NORTH COMPUTER WEST MUSEUM
Create Experience Learn
joe@nwcomputermuseum.org.uk
01942 582826
Information Technology facilities & training
Workshops for programming, Electronics & VR
Community classes, clubs & retro games
IT Skills for the young & old
City & Guilds Creditation
Supporting education of local people
Creating opportunities & careers
Disable friendly
Volunteer- ed training & museum operations
NORTH WEST COMPUTER MUSEUM
Create Experience Learn
www.nwcomputermuseum.org.uk
Leigh Spinners Mills, Park Lane, Leigh, WN7 2LB

Boost for Kemi as video game tycoon gives Tories £2million

EXCLUSIVE



Andrew Pierce reporting

A BRITISH video game tycoon has given a £2million donation to the Tory party.

The money from Jeremy Elliott, who is better known as Jez San, will give a huge boost to Kemi Badenoch's leadership, just days before challenging local elections.

I can disclose that Mr San, who founded Argonaut Software in his bedroom as a teenager during the early 1980s computer boom, gave £1million in February.

He then gave a second surprise £1million earlier this month. He is the biggest donor to Tory funds since Mrs Badenoch became party leader in November.

Mr San used to be a Labour supporter, but turned against Keir Starmer after the record £40billion tax-raising Autumn Budget.

His Argonaut company produced the Harry Potter, Alien Resurrection and Star Fox hit console games. It was also instrumental in developing early 3D graphics technologies, including the SuperFX chip used by Nintendo.



Delight: Kemi Badenoch

Mr San, 59, and his wife Natasha had dinner with Mrs Badenoch and her husband Hamish, who works in the City. They bonded over the Tory leader's pledge to cut regulation, taxes and to try to ensure Britain develops maximum potential from AI.

Computer games have been kind to Mr San. In 2022 he bought Palazzo di Vista, a £40million mansion in Los Angeles where his neighbours include Jennifer Aniston, Beyonce and Elon Musk. He's also a major donor to Great Ormond Street children's hospital.

Like most of his competitors, Mr San was an early computer user. He was 12 when his father brought back from the US a TRS-80, a forerunner of today's PCs. But his parents and teachers were far from encouraging. In a rare interview, he said: 'Everyone thought it was a terrible influence. I was staying up all night, every night.

'It's a while since I've seen the sunrise after working all night, but at school it happened all the time. I was hacking before it was made illegal, but never since.' He was only 16 when the money started to roll in. 'Some friends and I wrote a game called Skyline Attack,' he says. 'My share of the profits bought a second-hand Volkswagen Scirocco GTi. Computers have given me a great new car about every 18 months since.'

Starting from a Saturday job at a computer shop in Edgware, north-west London, less than a mile from Argonaut's offices, he now employs hundreds globally.

Mrs Badenoch told me: 'I was delighted to meet Jez San, who is one of our great entrepreneurs and innovators. His donations are an important statement of support in the Conservative Party to lead Britain out of the current economic malaise.' A source close to Mr San told me: 'Jez does not believe you can tax your way to growth. He has huge concerns over Labour's tax-heavy approach, which is driving top talent and investors abroad.

'He will offer policy advice to the Tories to help champion high-growth sectors and reward home-grown innovation. He believes you have to reward risk takers.'

The revelation of the donations comes at the end of a strong week for Mrs Badenoch who had her best performance at the despatch box at Prime Minister's Questions this week when she mocked Sir Keir's U-turn on the definition of woman in the aftermath of the landmark ruling by the Supreme Court.

Reform on track to have two mayors

By David Wilcock

NIGEL Farage's Reform UK is on course for two thumping wins in Red Wall mayoral elections next week, a poll suggests.

Dame Andrea Jenkyns, former Tory minister, is set to become Reform mayor of Greater Lincolnshire. As is former boxer Luke Campbell in Hull and East Yorkshire. Reform continues to perform well in the Midlands – gaining an MP for Boston and Skegness last year – and increasingly former Conservative councillors are defecting to the party.

However, the Tories hold eight of the 12 parliamentary constituencies and held half of the council seats across Greater Lincolnshire in last May's elections.

While Dame Andrea is on course to win – standing 15 points clear of her Tory rival in the YouGov poll – she could yet be stripped of the post. She is facing an inquiry by election chiefs after complaints she does not live in the area.

Robert Hardman PAGES 30 & 31

SORRY THIS IS THE 'WRONG' WAY UP BUT IT HAS A STRONG TRS-80 CONNECTION WHICH I THOUGHT WAS INTERESTING! IT DOESN'T TELL US WHICH MODEL TRS-80 WAS PURCHASED THOUGH. EVEN SO, IT'S WORTH A QUICK READ AS TRS-80 RARELY GET A MENTION IN THE BRITISH PRESS

TRS-80

Repairs & Upgrades

HNC qualified electronics engineer with 40 years TRS-80 experience.
Enquiries to: trs80@shanefoster.co.uk

THE 'COMPETITION-COMPETITION'



WOULD YOU LIKE TO WIN THIS RARE AND LIMITED EDITION, U.K. ROYAL MAIL, INFORMATION TECHNOLOGY, FIRST DAY COVER, FROM SEPTEMBER 1982? ALL YOU NEED TO DO IS TO SUGGEST THE THEME FOR THE 2026 COMPETITION.

JUST SEND IN YOUR ENTRIES TO THE 'TRASHTALKERS'

George@48k.ca

AND THE PERSON WHO COMES UP WITH THE BEST IDEA WILL RECEIVE, VIA SNAIL-MAIL, THE ABOVE PRIZE!

POWER SUPPLIES

FOR THE TRS-80 MODEL 3 / MODEL 4/4D/4P



* Uses modern day MEAN WELL power supply units
* 50w for Model 3 version and 65w for the Model 4 version
* The Model 4 version uses a SEPARATE +12V POWER SUPPLY (prevents video "spasm")
* No more RIFA CAP explosions
* No more COLD solder joints
* Units are DIRECT REPLACEMENTS for the original power supplies in all models
* Extra +5V output connectors (1 on Model 3 and 2 on Model 4) for Gotek/freHD boards
* For more information and pricing contact

newSOFT

https://jaynewirth.wixsite.com/newsoft (website)
jaynewirth@verizon.net (email)

LOOKING FOR FAST, INEXPENSIVE, UNLIMITED MASS STORAGE FOR YOUR TRS-80 MODEL I/III/4/4P/4D?

The amazing

"FreHD"

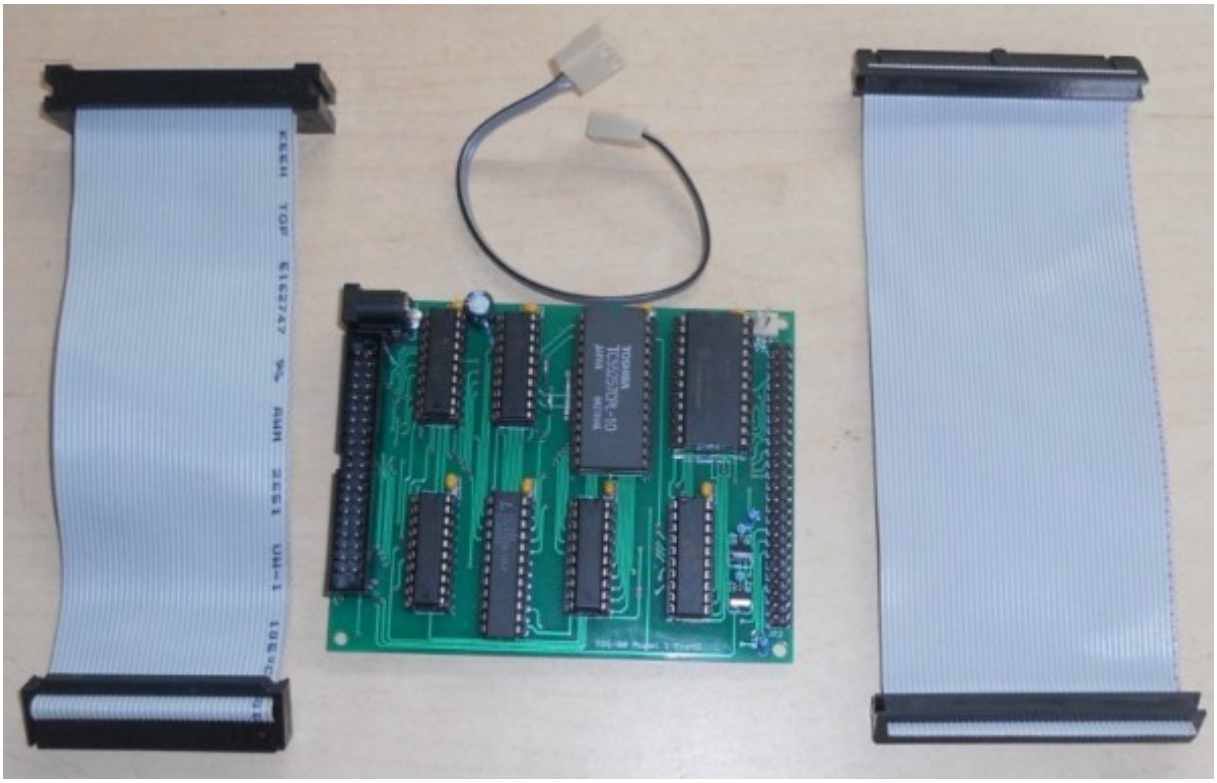


-Emulates a TRS-80 hard drive, but faster than any hard drive!
-Works with your favourite DOS (LS-DOS, LDOS, CP/M, Newdos/80 2.5)
-Uses SD card for storage medium
-Bonus free Real Time Clock function!
-Designed in Belgium and proudly built and shipped from Australia
-Kit form or fully assembled

Order yours today
http://trs-80.com.au/trs80/emulator.htm

“Quinnterface” Mini Expansion Interface for 16K Model 1 FreHD users.

- COMPLETLY ASSEMBLED AND TESTED -



This is THE perfect device for all Model 1 users who own a 16K Level II unit, (which is most of us!), but no Expansion Interface or disk drives, especially if you don't want to modify your M1 with upgraded boot ROM or memory upgrade.

The ‘Quinnterface’, developed by J. Andrew Quinn from New Zealand, adds 32K RAM and auto-boot functionality to you FreHD.

U.K. And Europe, contact Bas. at BetaGamma Computing.
U.S.A. And rest of the world, contact Mav. At “The Right Stuff”

EBAY.CO.UK 'BARGAINS'

HERE ARE JUST SOME OF THE MORE INTERESTING/UNUSUAL TANDY EBAY LISTINGS, SINCE OUR LAST ISSUE.
WERE YOU ONE OF LUCKY ONES?



£157



£7



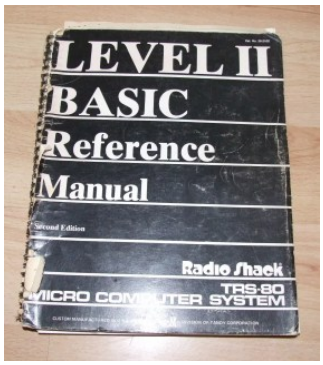
£260



£35



£268



£12



£376



£500
NS



£38



SOFTWARE

TRS-80 MODEL I
LII AND GENIE

CLUSTER CONFLICT is a one player space strategy wargame. A large fleet of Alien Ships has invaded Human Space, can you survive long enough to defeat them and time itself? Very tough wargame nedding thought and planning. At the higher levels each game can provide many hours of battle. Gamesave is included to allow return to family life. A snip at "£11.80" needs 16k.

SYSTORE is a m/Code package that will save you money by producing copies of your expensive software. Will copy m/Code and basic programs (with variables) and write any Ram block to tape. SYSTORE programs can be loaded in different memory locations to avoid clashing with resident programs. Check your system with the RAM DIAGNOSTIC provided. No knowledge of M/Code needed. full instructions supplied. An essential addition to your library at £12.90. 16k and 4k versions.

GAMESAVER is a m/code program that loads below basic text for use with programs that need all 16k of memory, ie DUNJONQUEST series. Full instructions to allow GAMESAVER S to return as if they never left. Hellfire warriors can now return retaining Dunjon position, magical items, character, treasures etc, one continuous quest. Get the best from your complex games for only £4.95.

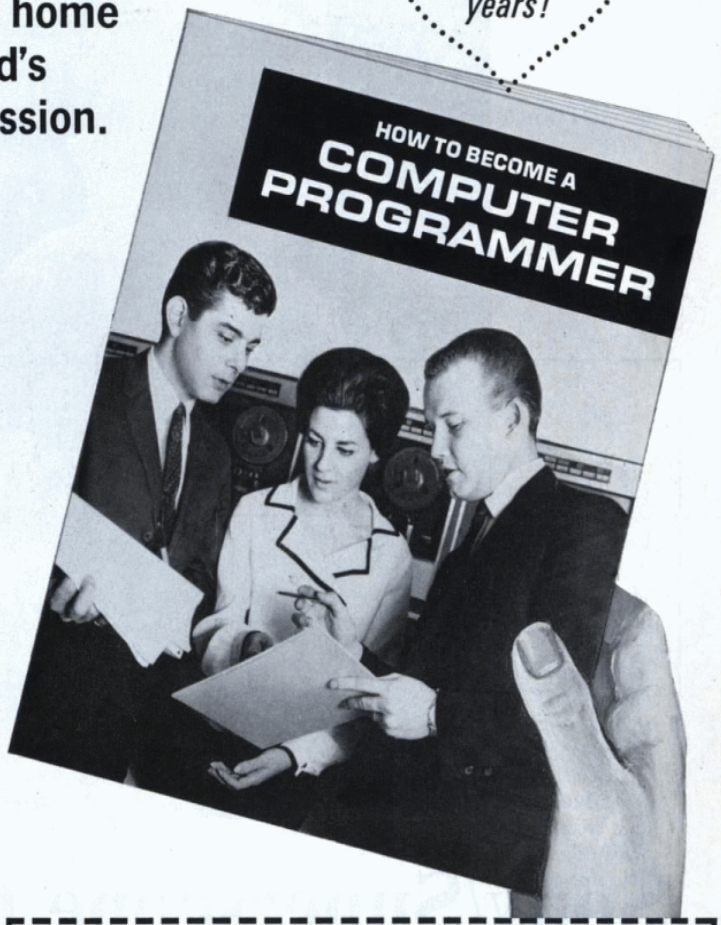
All items are double recorded on high quality cassettes and are guaranteed. All prices include VAT & postage, please state computer system. Send cheque/PO to

J.K. Gosden (Software — W),
13, Ashtead Common,
Ashtead, Surrey KT21 2ED.

This free booklet will tell you how you can become a Computer Programmer

50,000 more programmers needed now! 500,000 more will be needed in a few years!

...and how you can train at home for big earnings in the world's newest, most exciting profession.



IF YOU'RE dissatisfied with your present job, why don't you become a programmer? So great is the demand for programmers, you'll have your choice of openings, with a growing future ahead.

For the vast majority of good positions, a high school education, a logical mind and the right preparation are all that are required. LaSalle, world leader in home education, will train you as a programmer in your spare time.

Why not plan for bigger success in the field where employers are looking for *you? Authorities agree that for many years to come there will be more openings in computer programming than people to fill them!* Even beginners are paid well; and with experience, programmers can fill major supervisory positions at salaries of $12,000 to $15,000 a year – often as much as $20,000.

Without leaving your present position, you can start your training now – at very low cost. You can study at home, in hours of your own choosing. Experienced LaSalle instructors will guide you by mail every step of the way. Industry experts say this is the most thorough course of its kind that they have seen. The ease and rapidity with which you learn will surprise you.

LaSalle has just published a fascinating booklet "How To Become a Computer Programmer." In simple, non-technical language this booklet tells you what computer programming is; what the programmer does; how you can train by mail through the school that has enrolled more than 1,500,000 people in its many success courses. Mail the coupon for your free copy or write to LaSalle, 417 S. Dearborn Street, Chicago, Illinois 60605.

© 1967 LaSalle Extension University

LA SALLE EXTENSION UNIVERSITY

A Correspondence Institution

417 S. Dearborn Street, Dept. 86-070, Chicago, Illinois 60605

Please mail to me the free illustrated booklet "How To Become a Computer Programmer."

Print Name.....Age....

Address.....Apt. No.....

City.....

State.....Zip No.....

551

PCW SUBSET

6502/Z80 check digit

To reduce the chances of miskeying important numbers (say a product number when asking for its price or availability or an account number when asking for its balance) it is common, in business applications, for such numbers to have a check digit added on the end. The check digit is calculated by multiplying each digit of the basic number by a weighting digit from a fixed sequence, adding all the results together, dividing the final result by 10 and taking the one-digit remainder as the check digit.

If, for example, the weighting digits are 1, 3 and 7, a basic number of 2784 becomes 27843 with the check digit, 3, calculated:

```
2 x 1 =  2
7 x 3 = 21
8 x 7 = 56
4 x 1 =  4
        --
    10) 83

     8 remainder 3
```

If any digit in the number 27843 is entered incorrectly, it is probable that the number will fail the check digit test and the user can be asked to re-enter the number correctly.

At the end of Chapter 8 of his book *Z80 Assembly Language Programming*, Lance A Leventhal sets an exercise to write a routine to calculate a check digit from a string of BCD digits with this weighting 1, 3, 7 mod 10. Our next Datasheet, CADIDZ, from Dave Barrow of Hemsworth, gives the routine to do this in both Z80 and 6502 code.

Datasheet

```
;*CADID-BCD Checksum

;/CLASS: 1
;/TIME CRITICAL?: no
;/DESCRIPTION: calculates checksum (weighted 1,3,7,
;/             mod 10) on packed BCD string
;/ACTION: initialise checksum & multiplication factor
;/        (m) to zero. Iterate for each byte.
;/        Iterate for each digit
;/        Get digit in low-nibble storage
;/        m<- m*2+1
;/        m.l.c.<- m
;/       if m=7 then m<- 0
;/        iterate for m.l.c.
;/        checksum<- checksum+digit
;/        checksum<- checksum mod 10
;/SUBr DEPENDANCE: none
;/INTERFACES: none
;/INPUT: Z80:HL points 1st byte of string, B=no of bytes
;/       6502: M1,0 points 1st byte, M2= no of bytes
;/OUTPUT: Z80: HL points at string+1, B=checksum
;/        6502: M1,0 points at string+1, M2=checksum
;/REGs USED: Z80: B,HL   6502:M0,1,2
;/STACK USE: Z80: 8   6502: 5
;/LENGTH: Z80: 41   6502: 77
;/TIME STATES: Z80: 87+Av 180 per digit
;/             6502: 134+Av 83 per digit (ignoring page
;/             boundaries)
;/PROCESSOR: Z80   6502

;*CADIDZ- Z80 version
CADIDZ PUSH AF      ;                          F5
       PUSH BC      ;                          C5
       PUSH DE      ;                          D5
       XOR  A       ;initialise checksum and   AF
       LD   D,A     ;mult factor to 0          57
NXTBZ: LD   E,(HL)  ;save byte                 5E
       LD   C,+2    ;set digit counter         0E 02
NXTDZ: PUSH BC      ;save byte & digit counts  C5
       LD   B,A     ;get next digit in         47
       RLD          ;low-nibble C              ED 6F
       LD   C,A     ;and                       4F
       LD   A,B     ;checksum in A             78
       SLI  D       ;m<- m*2+1                 CB 32
       LD   B,D     ;additive loop count       42
MADDZ: ADD  A,C     ;checksum<- checksum+      81
       DAA          ;(digit*m)                 27
       DJNZ MADDZ   ;                          10 FC
       BIT  2,D     ;if m=7 then m<- 0         CB 52
       JR   Z,TDIG  ;                          28 01
       LD   D,B     ;                          50
TDIG:  POP  BC      ;restore byte&digit counts C1
       DEC  C       ;                          0D
       JR   NZ,NXTDZ;                          20 EA
       LD   (HL),E  ;restore byte and          73
       INC  HL      ;point to next             23
       DJNZ NXTBZ   ;                          10 E3
       AND  0FH     ;mod 10                    E6 0F
       POP  DE      ;                          D1
       POP  BC      ;                          C1

       LD   B,A     ;return checksum in B      47
       POP  AF      ;                          F1
       RET          ;                          C9


;*CADIDZ-6502 version
CADIDZ:PHP          ;                          08
       LDX  #4      ;save accumulator          A2 04
PHLP:  PHA          ;and                       48
       LDA  M1,X    ;M5,4,3                    B5 zz
       DEX          ;                          CA
       BNE  PHLP    ;                          D0 FA
       STA  M3      ;byte count                85 zz
       TXA          ;initialise checksum and   8A
       TAY          ;pointer index and         A8
       STA  M4      ;mult factor to 0          85 zz
       SED          ;decimal mode              F8
NXTBS: LDX  #2      ;set digit counter         A2 02
NXTDS: STA  M2      ;save checksum             85 zz
       LDA  (M0),Y  ;get byte                  B1 zz
       CPX  #2      ;and if                    E0 02
       BNE  NEWDIG  ;high nibble digit         D0 04
       LSR  A       ;shift it                  4A
       LSR  A       ;down                      4A
       LSR  A       ;into                      4A
       LSR  A       ;low-nibble A              4A
NEWDIG:STA  M5      ;                          85 zz
       LDA  M4      ;m                         A5 zz
       CMP  #7      ;                          C9 07
       BNE  NEWFAC  ;if m=7 then m<- 0         D0 01
       TYA          ;                          98
NEWFAC:SEC          ;m<- m*2+1                 38
       ROL  A       ;                          2A
       STA  M4      ;                          85 zz
       TAY          ;m.l.c.                    A8
       LDA  M2      ;                          A5 zz
MADDS: CLC          ;checksum<- checksum +     18
       ADC  M5      ;        (digit*m.l.c.)    65 zz
       DEY          ;                          88
       BNE  MADDS   ;                          D0 FA
       DEX          ;                          CA
       BNE  NXTDS   ;                          D0 DB
       INC  M0      ;point to next byte        E6 zz
       BNE  TEND    ;                          D0 02
       INC  M1      ;                          E6 zz
TEND:  DEC  M3      ;                          C6 zz
       BNE  NXTBS   ;                          D0 CF
       AND  #0F     ;mod 10                    29 0F
PLLP:  STA  M2,X    ;return checksum in M2     95 zz
       PLA          ;and restore M3,4,5        68
       INX          ;and accumulator           E8
       CPX  #4      ;                          E0 04
       BNE  PLLP    ;                          D0 F8
       PLP          ;                          28
       RTS          ;                          60
```

I hardly dare draw your attention to the sly use in CADIDZ of the non-existent SLI (shift left and increment) instruction. But what more perfect application of it could there be than to implement m ← m * 2 + 1 in calculating check digits with a 1, 3, 7 weighting specified by Levanthal?

Nobody reported non-execution of the mystery op codes CB 30 to CB 37, when we asked in Feb 81. But if any reader has a Z80 processor which doesn't execute SLI D, let us know and we will apologise.

Extra instructions-Z80

The code printed in April for LD HL,(HL) is improved by Roger Hargrave of Crawley to:

```
PUSH DE      ;D5
LD   E,(HL)  ;5E
INC  HL      ;23
LD   H,(HL)  ;66
LD   L,E     ;6B
POP  DE      ;D1
RET          ;C9
```

Roger asks why not a LD DE,(DE) and gives:

```
PUSH AF      ;F5
EX   DE,HL   ;EB
LD   A,(HL)  ;7E
INC  HL      ;23
LD   H,(HL)  ;66
LD   L,A     ;6F
EX   DE,HL   ;EB
POP  AF      ;F1
RET          ;C9
```

or a LD BC,(BC) not so conveniently coded as follows:

```
PUSH AF      ;F5
LD   A,(BC)  ;0A
PUSH AF      ;F5
INC  BC      ;03
LD   A,(BC)  ;0A
LD   B,A     ;47
POP  AF      ;F1
LD   C,A     ;4F
POP  AF      ;F1
RET          ;C9
```

THE LEAST EXPENSIVE PROGRAMS YOU CAN BUY.

Up to 77 high-quality programs for TRS-80, only $8!

THE $15.50 COMPILER:

TINY PASCAL FOR TRS-80

At last your TRS-80 can run Pascal too! The Chung/Yuen "tiny" Pascal is fully implemented for Level II TRS-80, 16K and up. You no longer need be left out of the growing group of Pascal users, because People's Pascal gives you everything you need to write structured Pascal programs:

- tiny Pascal compiler
- complete text editor for writing your programs
- complete tiny Pascal monitor
- sample Pascal programs
- user's manual

(TRS-80 Computing issue 1:4)

People's Pascal is both a powerful, structured language and "CPU expeditor". People's Pascal programs execute at least four times faster than Basic, and often eight-times faster! Special functions open up the complete graphic capability of TRS-80. You now have the means to write those dazzling, impressive, high-speed graphics programs that are great for games, plotting, statistics, etc.

For the serious computerist, side two of People's Pascal II (tape 6) contains a larger compiler and complete source to the compiler, written in Pascal! This means you can re-compile the compiler, making changes, adding features, etc. (but this will take at least 36K RAM and a solid knowledge of programming).

With the complete People's Pascal operating system, you can save and load both source (Pascal) programs, and compiled programs, to or from cassette tape. This means that once you have de-bugged a program, you can save the P-code (compiled program) and thereafter, to run the program, you need only load the super-fast P-code.

Here is a partial list of People's Pascal features: recursive procedure/functions • for (loop) • case if/then/else • one-dimensional arrays • write • read constant • repeat/until (loop) • "peek & poke" • plot (graphics for TRS-80)

DEALER INQUERIES INVITED

People's Pascal 1 (tape 3) is written in Basic, implemented for TRS-80 by John Alexander of Berwick Australia. It compiles P codes more slowly and is harder to use than Pascal 2, but its P codes can be translated into Z80 native code and saved as System tapes. Pascal 2 requires that Pascal be resident at run time—Pascal 1 does not. Other People's Software tapes $8.

TAPE 1 LEVEL 2

Mortgage calculations, Dow Jones Industrial, cash flow, inventory-change, California income tax, journal ledger (8K), loan amortization, perpetual calendar, bio rhythm, payroll, diet planning, speed reading, touch typing, sales receipt tally, decision maker, mail addressing, straight depreciation, double-declining depreciation, and revolving charge account.

Also, math problems, queen, Star Trek I, number guessing, wheel of fortune, World War II bomber, rock-scissors-paper, seek, Star Trek II, Red Baron, mini-Trek, strategy, pilot, battleship, "On A Snowy Evening", mastermind, tic-tac-toe, grand prix auto race, capitals, etch sketch, hangman. Total programs: 34; Level 1 version available: 24 programs. $8. $8.

TAPE 2 Some Common Basic Programs (lev. 2)

Fully documented in **Some Common Basic Programs** by Lon Poole & Mary Borchers (Osborne & Associates, 630 Bancroft way, Berkeley CA 94710—or from CIE—$12.50 postpaid from CIE, via UPS, CA residents add tax (to $13.25)):

Investment, future value regular deposits,; regular deposits; regular withdrawals, initial, minim (for withdrawals); nominal interest, effective & earned-interest; depreciation rate, amount depreciation; salvage value; discount com'l paper; loan principal, regular & last payment, remaining balance, term-loan; mortgage amortization; greatest common denom.- integer prime factors; polygon area; triangle parts; analysis, operations two vectors; radian-degree., degree-radian conversion; coordinate, polar equation, functions plot; linear, curvilinear interpolation; Simpson's & trapezoidal rules, Gaussian quadrature integration; derivative.

Side 2—quadratic equation, polynomial (Newton) & half-interval-search roots; trig polynomial; simultaneous equations; linear programming; matrix addition, subtraction; scalar multiplication, inversion; permutations & combinations; Mann-Whitney U test; mean, variance, standard deviation; geometric mean & deviation; binomial, Poisson, normal, Chi-square distribution; Chi-sq., student's T-distribution test; F-distribution; linear correlation coefficient; linear, multiple-linear, Nth order, geometric, exponental regression; system reliability; future projections; Federal withholding taxes; tax depreciation schedule; check writer; recipe cost; map check; day of week; days between two dates; anglo to metric; alphebitize. $8

TAPE 3 People's Pascal Development System 1

Pascal 1 compiler-program development system. $15.50.

TAPE 4 LEVEL 1

Election returns, business percentage, ups and downs of business, index, inventory control, sales receipt tally, gas mileage, driving distance, mixed monthly sales report, payroll, annual earnings, speech recording aid, and double-declining depreciation.

Also, math problems, cash register, chase, snoopy, commander-in-chief, Christmas graphic, air raid, balance scale, stock market, tic-tac-toe and **On A Snowy Evening**. $8.

TAPE 5 LEVEL 2

Memory test, mortgage payments, tension breaker, lineprinter-screen & vice-versa utilities, Federal income tax, election returns, business percentage, vacation planner, car pool(disk), diet planning 2, mailing list(disk) and first aid.

Also spelling bee, Star Trek 3, mind bender, tachistoscope, chase, common factor, klingon capture, spelling practice, Hamurabi, animals, Snoopy, cryptogram, starship, ants, **Yesterday**, and Pilot(disk). Pilot is the language of computer-aided instruction (CAI)..

TAPE 6 People's Pascal 2

Pascal 2 compiler-program developemnt system. $23.50.

TAPE 7 LEVEL 2

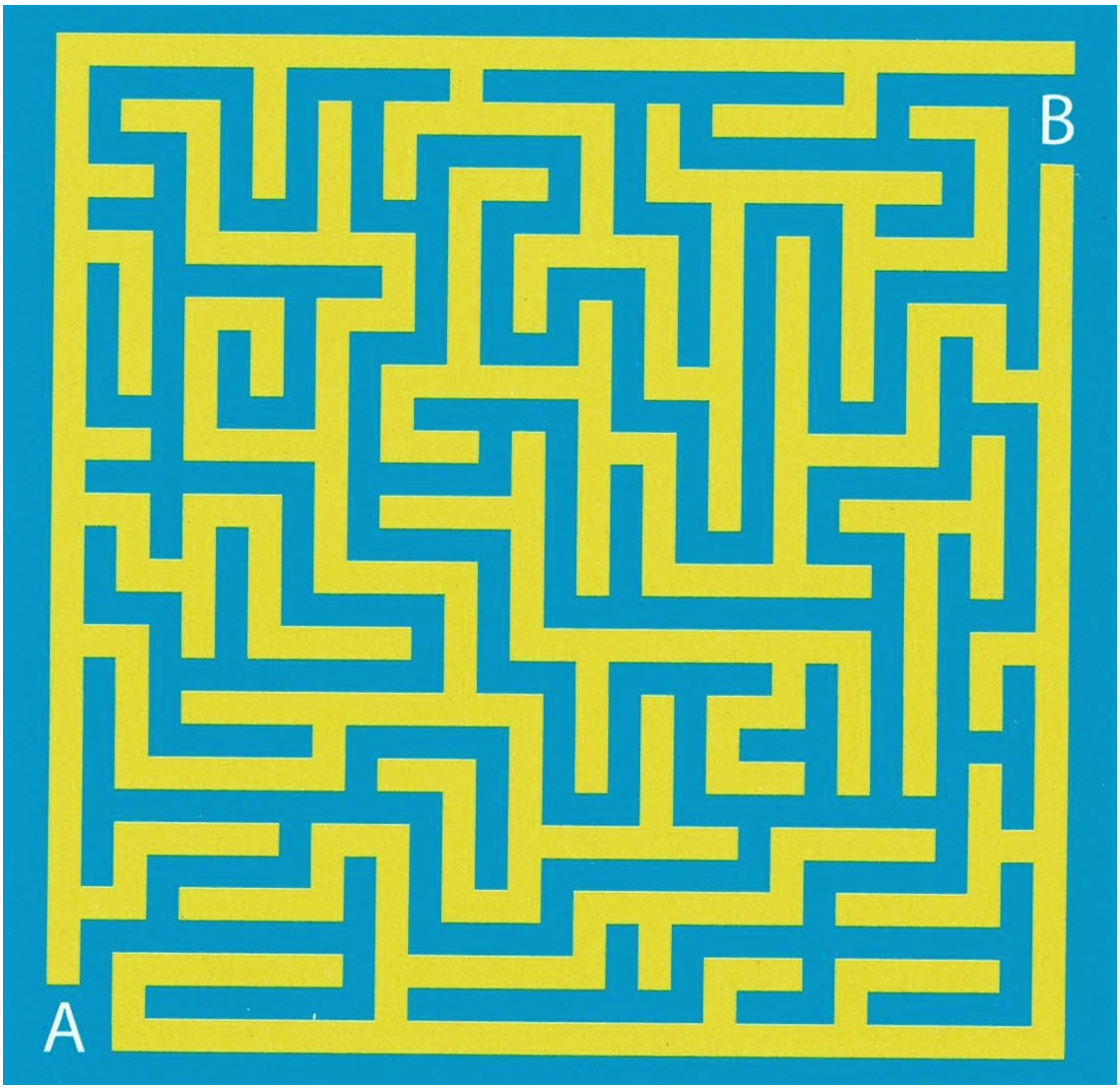
Disassembler, Pilot, roster, dropout, memory loader, memory sort, inventory control, graph, land surveying, mixed monthly sales report, shopping list, diet planning 3, loan progress chart, hex-decimal conversion.

Also Star Trek 4, states and capitals, battleships 2, spelling practice 2, number guessing, hangman 2, snark, slot machine, cipher, target, surround, adder, termites, lunar lander, multiplication exercise, five-in-a-row, Bastem, and write. A number after a program indicates there are other similar People's Software programs. Pilot is the same as the disk pilot on tape 5, except runs on 16K tape systems. $8.

—CA Residents Add 6 per cent TAX ($8.45, $16.40 & 24.88). Quoted Prices INCLUDE $.50 Shipping—

COMPUTER INFORMATION EXCHANGE
Box 159, San Luis Rey CA 92068 (714) 757-4849

AMAZED



IF YOU HARDWARE-HACKERS ARE FINDING THOSE PCB BOARDS TRACKS ARE MERGING INTO ONE OR YOU ASSEMBLER CODERS HAVE LOST A PUSH OR A POP, GIVE YOURSELVES A BREAK.

SIT DOWN WITH A COFFEE, PUT A TIMER ON AND SEE HOW LONG IT TAKES TO GET FROM A TO B, FOLLOWING THE BLUE ROAD.
TRY FOR COMPLETING IN UNDER 10 SECONDS!



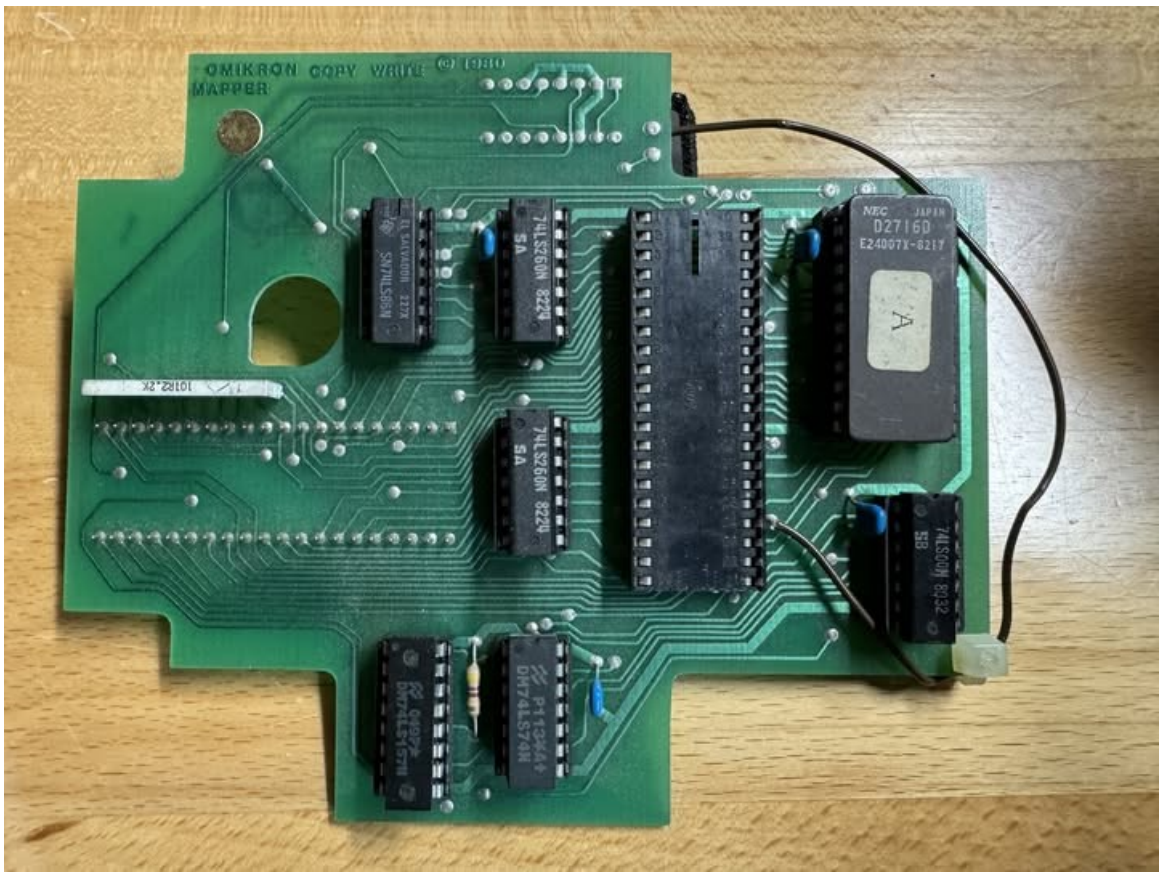
LEVEL 2.5 (TWOHAF) BASIC ALLOWS CASSETTE AND ACULAB FLOPPY TAPE USERS MANY OF THE TANDY DISK BASIC FEATURES, NAMELY -

"&", "DEF FN", "DEF USR", "MID$=", "INSTR", "MKD$", "MKI$", "MKS$", "CVD", "CVI", "CVS"

AVAILABLE, AS A **FREE DOWNLOAD**, FROM

WWW.TRS-80.ORG.UK

Surfin' The Net



THE OMIKRON MAPPER FOR THE TRS-80 MODEL I. THIS BOARD ALLOWED YOU TO USE CP/M ON THE TRS-80. NORMALLY THIS WAS NOT POSSIBLE SINCE THE TRS-80 ROM RESIDES IN THE BOTTOM ADDRESS SPACE STARTING AT MEMORY LOCATION 0000 WHICH IS WHERE CP/M NEEDS TO LIVE.

INSTALLATION WAS FAIRLY EASY. YOU WOULD REMOVE THE Z80 CPU FROM YOUR TRS-80, INSERT IT INTO THE 40PIN SOCKET ON THE MAPPER AND THEN INSERT THE ENTIRE MAPPER BOARD INTO THE COMPUTER'S Z80 SOCKET. THERE WERE OTHER CP/M HARDWARE SOLUTIONS FOR THE TRS-80 BUT THE OMIKRON MAPPER WAS BY FAR THE MOST POPULAR. YOU CAN GET MODERN REPRODUCTIONS OF THIS BOARD FROM IAN MAVRIC.



The *BEST* in TRS-80s

Call The Right Stuff

Ask for Ian
The number is +61 416 184 893

That's The Right Stuff
And he's in Melbourne







TRS8BIT IS SUPPORTED BY YOU AND PRODUCED
AT WWW.TRS-80.ORG.UK

TRS8BIT

PRODUCED AT WWW.TRS-80.ORG.UK

SEPTEMBER 2025 - VOLUME 19 ISSUE 3

PAGE 03 — 2025 COMPETITION - IMPOSSIPUZZEL25
ONCE AGAIN, GREAT PRIZE SO GET YOUR ENTRIES IN AS TIME IS RUNNING OUT.

PAGE 05 — TA25 NEWS - IAN MAVRIC
PLANNING THE 2025 TANDY ASSEMBLY

PAGE 15 — POWER OF THE ROM
VINCE OTTEN
VINCE CONTINUES HIS SERIES WITH PART 3, USING DR. DAVID LIEN'S BOOK "IT'S EASIER THAN YOU THINK"

PAGE 39 — OBIT. MATTHEW REED

PAGE 40 — IN MAV'S WORKSHOP
IAN MAVRIC
"YOU BOUGHT A MODEL 2000". WHAT NOW!

PAGE 53 — STAR TREK NEWS

PAGE 55 — CREATING PASSWORDS
E.T. FONEHUME
E.T. HAS A ONE-LINER BASIC PROGRAM TO CREATE 'MODERN DAY' PASSWORDS WITH A RATHER SNEAKY WAY TO ALLOW LOWER CASE LETTERS ON A STANDARD M1 L2!

PAGE 57 — THE COMPETITION, COMPETITION
WHAT ARE YOUR VIEWS? PLEASE LET US KNOW.

PAGE 60 — U.K.'S EBAY BARGAINS
SINCE OUR LAST ISSUE. DID YOU MANAGE TO 'WIN' ANYTHING EXCITING?

PAGE 62 — THE WAY WE WERE
A LOOK BACK TO VARIOUS PUBLICATIONS WITH SOME INTERESTING ARTICLES FROM 'BACK IN THE DAY'
DID ANYONE OUT THERE USE A MICROWRITER?

PAGE 70 — APL
VINCE OTTEN
VINCE REVIEWS THE PROGRAMING LANGUAGE WHICH WAS IMPLEMENTED FOR THE MODEL 1

PAGE 83 — NEWS FROM FRANCE
PASCAL HOLDRY

PAGE 85 — BASQUE TOURNAMENT RESULTS - JULY 2025
EGOITZ CAMPO

PAGE 88 — 'AMAZED'
JUST FOR FUN! HAVE A GO, TRYING TO BEAT 8 SECONDS

PAGE 89 — LINUX BIRTHDAY!

PAGE 91 — IN MAV'S TRAVELS
IAN MAVRIC
MAV REPORTS ON THE GREAT TIME HE HAD AT VCF-SW

PAGE 103 — SURFIN' THE NET
WATCH THE VIDEO FOR THIS M1 UPGRADE!

LATEST UPGRADE AVAILABLE SHORTLY
VERSION 2.5.6

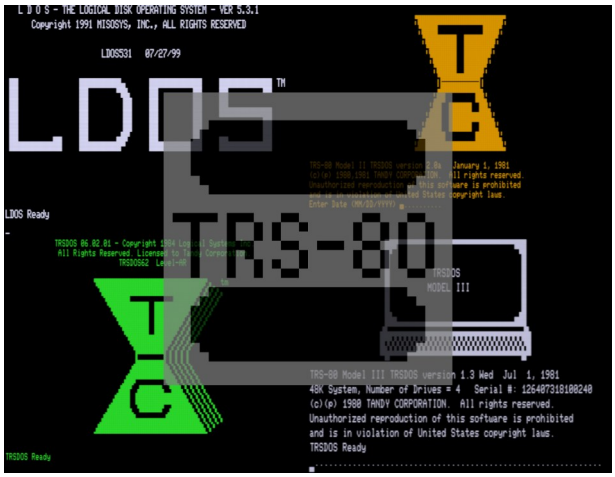
DON'T HAVE A TRS-80?

CRAZY PRICES ON EBAY?

TIRED OF RIFA CAPS BLOWING?

OF FLOPPIES SHEDDING OXIDE?

OR JUST PLAIN DEAD CHIPS?





THEN WHAT YOU NEED IS TRS80GP.
THE ULTIMATE TRS-80 EMULATOR



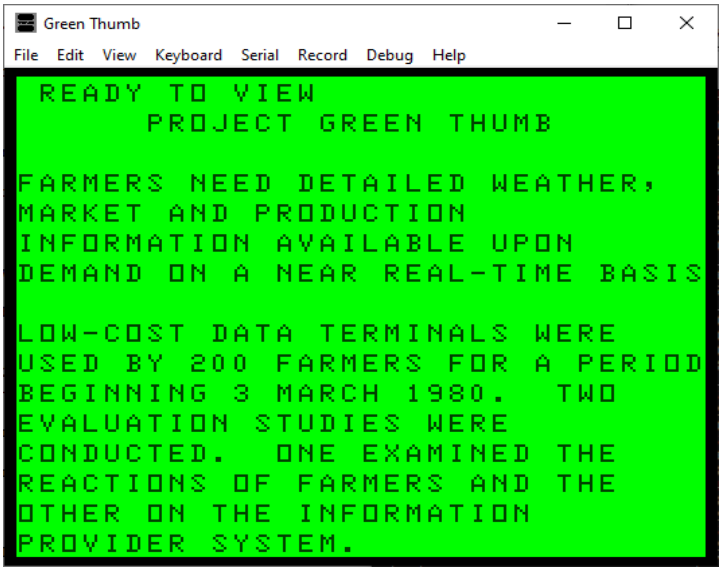
RUNS ON WINDOWS, MACOS, LINUX & RASPBERRY PI.
EMULATES MODELS 1, 2, 3, 4, 12, 16, 6000, MC-10,
COLOR COMPUTER, DT-1, VIDEOTEX, TANDY 10 AND NOW 'GREEN THUMB'.

JUST LIKE A REAL TRS-80 WITH NONE OF THE HASSLE.
BUILT-IN DEBUGGER AND OTHER FEATURES, JUST GREAT FOR PROGRAMMERS.

GET IT NOW AT HTTP://48K.CA/TRS80GP.HTML

UPDATED VERSION, 2.5.6, ADDITIONALLY OFFERS :

'GREEN THUMB' EMULATION
NEW SCRIPTING COMMANDS



TRS8BIT's IMPOSSiPUZZLE25

A U.K. quiz program called 'Countdown' has a fascinating numbers game.
You select 6 unseen numbers comprising of a group from 1 to 10 (small numbers) and the four large numbers 25, 50, 75, and 100.
You can choose how many 'small' and/or 'large' numbers you require.
Duplicates, if they occur, are allowed.
Then a 'target' number is randomised from 100 to 999

Contestants then have 30 seconds to calculate, with the 4 arithmetic operators +, -, * and /, the 'target' number using any or all of the 6 random numbers, ensuring that each number is only used once.

At no stage of the calculation can you have a negative or fractional number.

As an example:-
Let's say you have chosen 2 'large' and 4 'small' numbers.
When revealed they were 75, 100, 7, 10, 1, 10 and a 'target' number of 543

Here's one way to calculate the 'target' number:

```
(75+1)*7    = 532
(100+10)/10 =  11
532+11      = 543
```

To win the TRS8BIT IMPOSSiPUZZLE25, you need to write a program, running on a TRS-80 emulator, to take the 6 random numbers, **25,75,5,2,8,5** with the 'target' of **124** and show how to use them to calculate the 'target' number. Remember of course, it may be an impossible task. You can decide how long to cycle the program looking for an answer or, if one is not found, just declare how close you were!

The 'Trashtalkers' team will judge the entries, taking into account, running speed, programing style, language(s) used, entry date and the number of TRS-80 models your program will run on.
.

The winner will receive their choice from one of Mav's wonderful products.

Closing date for the competition is
22:00 hrs GMT on Sunday 2nd November 2025
Entries to be sent to George Phillips at george@48k.ca


ACTUAL REALITY
TRS-80 MODEL I/III/IV
DREAMLAND SOFTWARE


ROCK STAR!
TRS-80 MODEL I/III/IV
DREAMLAND SOFTWARE

MAV'S TANDY ASSEMBLY QUARTERLY COLUMN

TA25 NEWS

by Ian Mavric



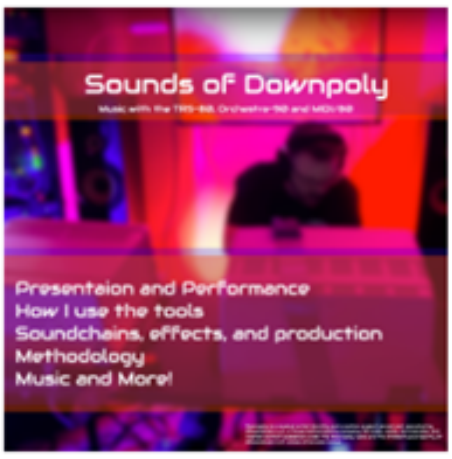
Bringing you the latest news you need to know about the greatest Tandy TRS-80 show on the planet. This time I talk about the rest of the presenters, with the show being only 3 short weeks away now.

TA25 ALL PRESENTERS ANNOUNCED

Last time I mentioned the following people were already signed up:

Joel Hilliard:





"A Presentation covering the MIDI80, the core functions and hardware, how it works and demonstration of the software including the TRACKER, STORCH (Stereo Orchestra Organ) as well as examples of MIDI Control IN, OUT and THRU including controlling the Orchestra-90 and how you can fit the TRS-80 in the studio! Joel will talk about the music he created using the Orchestra-90, and MIDI/80, a fusion of Synth meets Electric Rock all created using a few TRS-80 Model IIIs and 4/4Ps. He will give a presentation on the methodology used to create the music, hardware and sound setups, CDs of the initial release, along with a Limited Release Color Vinyl and more!"

Marcel Erz:



"In this talk, I'll walk through my journey of reverse engineering the TRS-80 Model 1, sharing some of the curiosities I found and highlighting key differences across various board revisions. I'll also discuss the development of custom parts I've created over the past year—including a replacement keyboard, case, and other hardware—offering a look at what it takes to recreate and preserve such an iconic machine from the inside out."

Tim Halloran:



"This technical presentation explores the methods and challenges of running the CP/M operating system on Tandy's popular TRS-80 microcomputers (including Business systems). While TRSDOS and its tree of derivative DOSs served TRS-80 users, accessing the vast library of CP/M software required adaptations. We will delve into how CP/M works and introduce popular third-party CP/M implementations (e.g., Pickles & Trout, FMG, Lifeboat). Discuss the BIOS-level customizations needed to bridge the gap between the CP/M standard and the TRS-80's unique architecture. Discover the technical ingenuity that transformed the TRS-80 into a capable CP/M workhorse."

WITHOUT FURTHER ADO, THE REST OF THE PRESENTERS...

TTT104 (on Friday) Jeff Birt "Hey Birt":



"When working with our vintage computers we all get ideas about some new modern accessory that would be nifty. We might grab an Arduino, etc. and whip up a prototype. Flush with excitement that our gadget works we wonder if other folks would like one too. What is really involved in building and marketing modern accessories for vintage computers? What are the challenges? What are the rewards? Will you make a few $$ or will you lose $$$. This is the story of my journey..."

TTT105 (on Friday): Christopher Hyzer



"The VIS (Video Information System) may not have been successful in the market, but it represents a unique implementation of a PC-ish x86 architecture. It represents a subset of PC BIOS, DOS, and Windows with a slew of specialty calls and interfaces intended for set-top implementations with easy application portability from the PC. In this presentation, I'll dive into a brief overview of VIS, its future and fate, followed by a look at some of the technical aspects of the platform. I'll look at the available app. environments, the licensing and security system, similarities and differences to the PC architecture, and some of the unique features and system calls. Finally, we'll look at how you can write software for the system using "classic" (legacy) tooling."

(On Saturday): Jim Brain - "Beyond 512kB: Enhancing the CoCo Memory Management Unit"



"With the introduction of the Tandy Color Computer 3 came a versatile memory management unit allowing up to 512kB of physical RAM usage and support for multiple MMU mappings. Let's look at how the CoCo3 squeezes 512kB of RAM into the 6809's 64kB address space and how we can expand that support. We also will show how we can retrofit an MMU into the non MMU CoCo1 and 2 and Dragon systems."

For those who like to plan ahead to make sure you fit in everything you wish to see at Tandy Assembly, here is the schedule for the show:

Friday, September 26

Tandy Tech Track

1:00 pm - TTT101 - Tim Halloran - "CP/M on the TRS-80: An Introduction"

2:15 pm - TTT102 - Joel Hilliard - "The MIDI/80 and Software"

3:30 pm - TTT103 - Marcel Erz - "Rebuilding the TRS-80 Model I: A Reverse Engineering Journey"

4:45 pm - TTT104 - Jeff Birt - "Building Modern Accessories for Vintage Computers for Fun ~~and Profit~~"

6:00 pm - TTT105 - Chris Hyzer - "Exploring 513: A Technical Look at VIS"

Saturday, September 27

9:00 am - Show Opens - Light Breakfast Provided

10:00 am - Speaker - Jim Brain - "Beyond 512kB: Enhancing the CoCo Memory Management Unit"

11:00 am - Exhibits

1:00 pm - Speaker - Joel Hilliard - "Downpoly"

2:30 pm - Auction

8:00 pm - Evening Events

Trivia Contest hosted by Mav (bring a pen to compete and possibly win a prize)

Evening Socializing

Sunday, September 28

9:00 am - Show Opens - Light Breakfast Provided

10:00 am - Speaker - TBD

11:00 am - Exhibits

3:00 pm - Show Closes - Exhibit Teardown

TAUT - TANDY ASSEMBLY UNOFFICIAL THURSDAY

Like previous years there is a gathering of people who arrive on Wednesday evening or really early on Thursday. The function is a non-Tandy computer related get together to help get over some jetlag and catch up on what has been going down for the last 12 months.

This year we are doing three things:

Before lunch: Midwest Surplus Electronics in Fairborn, Ohio. Basically an electronics surplus store has lots of interesting and obsolete electronics parts. Good place if you are building or repairing vintage computers. Bring you BOM!

Lunchtime: Lunch (obviously) somewhere. Previously we went to a Chinese Buffet in Dayton. A new place has not been decided as yet, so the buffet might get the nod again. The food is great with much variety, the drinks are huge, and prices extremely reasonable.

Afternoon: VOA Museum "National Voice Of America Museum Of Broadcasting" we go on a tour for the princely sum of $10.00



For more information: **https://voamuseum.org/exhibits/**

We typically leave the Marriot hotel around 9:30am-10am. Don't have a car? No worries, just hitch a ride with one of the many people attending who have one :-) Come and find us in the breakfast room before 9am and introduce yourself, we will make space for all.

TRIVIA GAME STILL NEEDS PARTICIPANTS...

At this risk of sounding like a broken record, I still need trivia game contestants. Suitably qualified (or unqualified) women and men are encouraged to apply. Email me: ianm@trs-80.com



Photo: TA25 Trivia contestants Steve Leininger, Brendan Bergie, Neil Blanchard

Plot twist! Brendan is unable to attend TA this year to defend his title of trivia champion so Neil, due the being last year's runner-up, will return in his place. Do you have what it takes to beat Neil? Sign up for the trivia game and find out!

If you have questions about the show or want to contact us directly out email address is: tandyassembly@gmail.com We always appreciate hearing from people who wish to be part of the show.

Ian Mavric

ianm@trs-80.com



Ian Mavric is an IT Specialist who also restores and collects TRS-80's and classic cars. He lives with his wife and kids in Melbourne, Australia.

Thirsting for adventure?
Brass lantern getting dusty?

Take your Model I/III/4/4p on exciting new adventures never available until now....

UPDATED 2025!



M4ZVM enables your TRS-80 Model I, III or 4 to play Infocom and post-Infocom adventures, taking full advantage of your hardware. Supports old favourites like Zork and Seastalker (including Sonarscope), as well as new-to-Model-4 games including Trinity, Border Zone and Sherlock, plus hundreds more available in the Interactive Fiction archive.



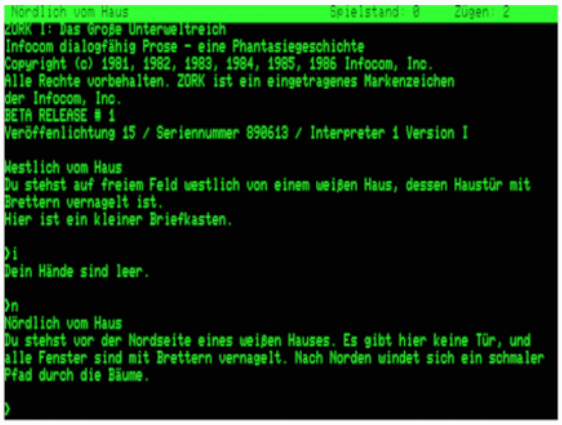
M4ZVM & M3ZVM with sample disks and game compatibility lists are at:
https://sijnstra.itch.io/m4zvm

Source and links to other versions at:
https://gitlab.com/sijnstra1/m4zvm

Actual Screenshots of games new to the model 4, running M4ZVM (on trs80gp) - check links for how-tos & disks



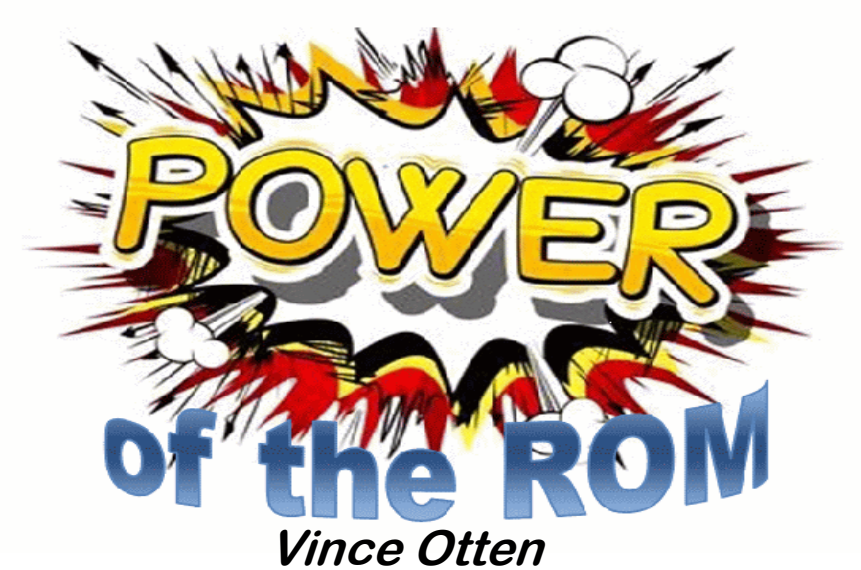
Beyond Zork



Zork 1 (German)

M4ZVM/M3ZVM/M1ZVM features include:
* z1-z8 game execution (z6 support is experimental)
* named game save/load
* accented characters where available
* Optionally set the "Tandy bit" for Tandy customization!

UPDATED 2025 version - faster, better, more bugfixes!



Vince Otten

Mr Dear Aunt Sally

If you've been following this series on using TRS-80 Model I Level II BASIC ROM calls, you know that we've covered:

- CLS
- How to PRINT a string
- Beginning Math Calls:
  - Mandatory Setup Routines
  - The Floating Point Accumulator and "Number Type Flag"
  - Converting Integers to Single Precision
  - Single Precision Multiplication
  - Converting Single Precision to a String
  - (PRINT with BASIC's "Trailing Comma")

This voyage of discovery follows the map of Dr. David Lien's sample programs in his Level I BASIC text.



What's next? In Chapter 5, we find this example:

```
10  REM * CELSIUS TO FAHRENHEIT CONVERSION *
20  C = 25
30  F = ( 9 / 5 ) * C + 32
40  PRINT C; "DEGREES CELSIUS =" ; F; "DEGREES FAHRENHEIT."
```

Sleeves rolled up, let's translate this into Assembly Language for the TRS-80. First, don't forget to set up:

```
MAIN    EQU   $          ; PROGRAM STARTS HERE
        CALL  INIT       ; INITIALIZE SOME BASIC "PROPERTIES"
```

You can find the code for INIT in Part 2 of this series (TRS8Bit June 2025). Also, whenever we want to "USE MATH", we need to

```
        CALL  I00CRM     ; "USES MATH"

;  "10 REM * CELSIUS TO FAHRENHEIT CONVERSION *"
```

Now we're ready to interpret line 20 of Dr. Lien's BASIC code.

```
20  C = 25
```

Since all Level I BASIC variables used in the text are Single Precision, we want to have a S.P. variable in our Assembly Language program as well. We're learning about BASIC ROM calls after all. One way that we did this in Part 2 of the series was to convert an Integer representation to Single Precision:

```
CINT    DEFW  25         ; C AS AN INTEGER
CSP     DEFS  4          ; SPACE TO STORE
                         ; C AS SINGLE PRECISION

        LD    HL,(CINT)
        LD    DE,CSP
        CALL  CONVRT     ; FROM TRS8BIT JUNE 2025
```

Is there any *faster* code? After all, we're programming in Assembly Language so we can speed up TRS-80 programs. What if we could store "C" as a Single-Precision value rather than as an integer which has to be converted? Of course you can do that with modern x86 processors: they have a floating-point coprocessor built in. But our primitive Z80 has nothing like that. In fact, I can remember an "floating point coprocessor" being marketed in the late 1970s which was just a calculator interfaced to the S-100 bus!

Having said that, we *can* store "c"as a Single Precision value from within the assembler. Here's how:

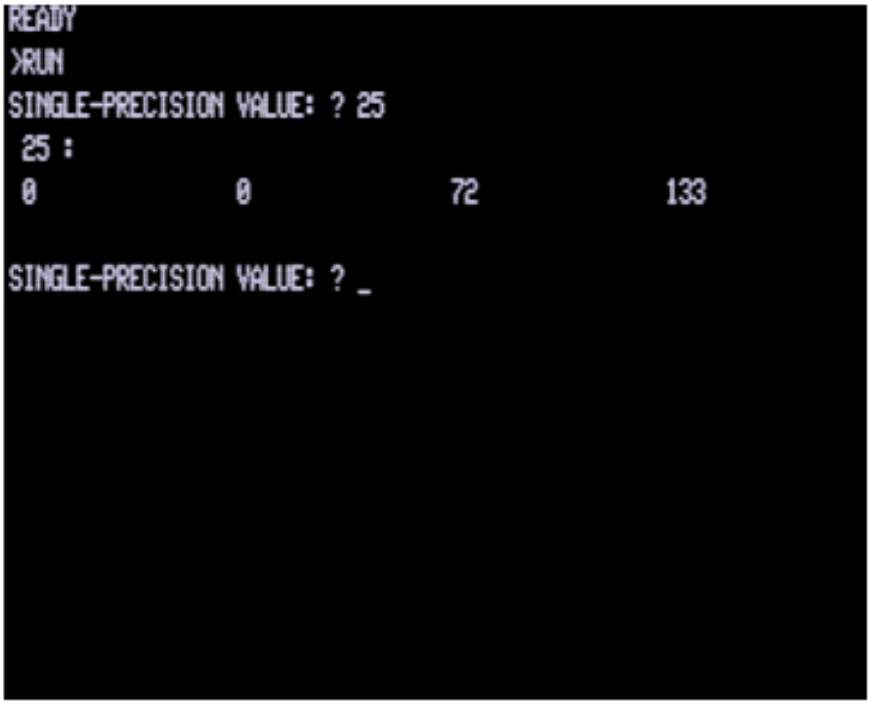
```
10 '
     ** DISPLAY BYTES FOR SINGLE-PRECISION VALUES **

100 INPUT "SINGLE-PRECISION VALUE: "; SP!
220 '
     CONVERT SP! TO A SINGLE-PRECISION NUMBER

230 SP!= CSNG( SP#)
290 '
     PRINT THE VALUE, THEN THE 4-BYTE REPRESENTATION

300 PRINT SP!;":",
310 SA%= VARPTR( SP!)
320 FOR AD%= SA% TO SA% + 3
330     PRINT PEEK( AD%),
340     NEXT
400 PRINT
410 GOTO 100
```

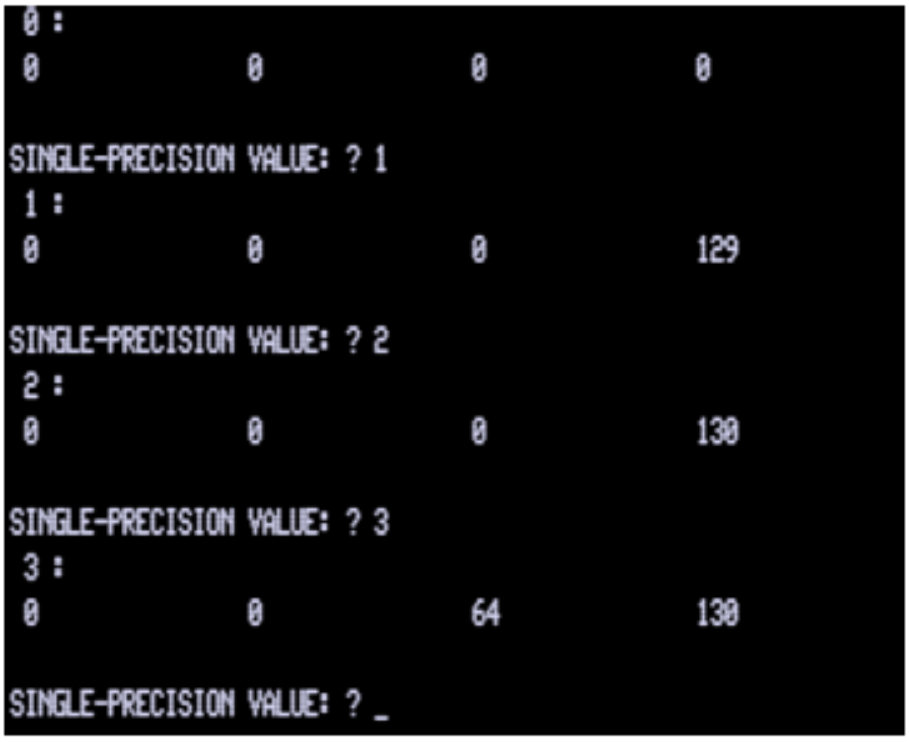
This BASIC program prompts you for a Single Precision value (line 100), makes sure it's an S.P. number (line 230), then prints the number again (line 300), and prints out the 4 bytes that make up the S.P. representation in memory (lines 310..400). Here's the result for a value of "25":



This means we could store "C" as a Single Precision variable in Assembler this way:

```
CSP     EQU     $           ; "C" AS SINGLE PRECISION
        DEFB    0           ; LOWEST BYTE
        DEFB    0
        DEFB    72          ; HIGHEST BYTE OF "25"
        DEFB    133         ; EXPONENT
```

When we're ready, we can simply move these 4 bytes into the Single Precision Accumulator "WRA1" and do some math with this value. (Just remember: be sure to load NTF1 (40AFH) with a 4 to indicate you're working with a Single Precision value in WRA1 before you call the ROM Math routines.) If you're curious, you can see the mysterious way BASIC interprets Single Precision numbers:



If you're more interested in the *readability* of your Assembly-language code than in its speed—what a concept—then you could store your floating-point constant as a *string*, later converting it to a Single Precision value, perhaps like

```
CASTR   DEFM    '25.0'          ; C AS A STRING
        DEFB    0               ; 0-TERMINATION

PIASTR  DEFM    '3.1415926'     ; PI AS A STRING
        DEFB    0
```

(There was a C compiler for CP/M back in the 1980s which did all its non-integer math using strings.) We'll look at converting strings to numbers later on in this article.
On to the next line in the BASIC program:

```
30  F = ( 9 / 5 ) * C + 32
```

Now that we have to convert it to Assembly Language, BASIC looks so appealing. You can just type in whole formulas! That was the impetus for the first programming languages like Fortran: the ability to put floating point variables directly into expressions, add parentheses, and even apply a hierarchy of operations like "My Dear Aunt Sally"—MDAS, or "Multiply—Divide—Add—Subtract". The ROM subroutines know nothing of that. You have to put one value in the WRA1 accumulator and another in the BC/DE register pair group, and then call the DIVIDE subroutine, for example. Here's one way we can calculate the expression in line 30:

1. Put a Single Precision "5" in WRA1, the Floating Point Accumulator.
2. Put an S.P. "9" in the BC/DE pair grouping.
3. Call the ROM DIVIDE subroutine. The result will be in WRA1.
4. Put the S.P. value of "C", that is "25", in BC/DE.
5. Call the ROM MULTIPLY subroutine. The result will again be in WRA1.
6. Put an S.P. "32" in BC/DE.
7. Call the ROM Add subroutine.
8. Move the S.P. result in WRA1 to the "F" S.P. variable.

More complicated, but manageable: every intermediate result was nicely left in WRA1. We just worked our way from left to right. Now what if the expression we'd wanted to evaluate was in this equivalent form:

```
F = 32 + (C * ( 9 / 5 ))
```

Attempt #1:

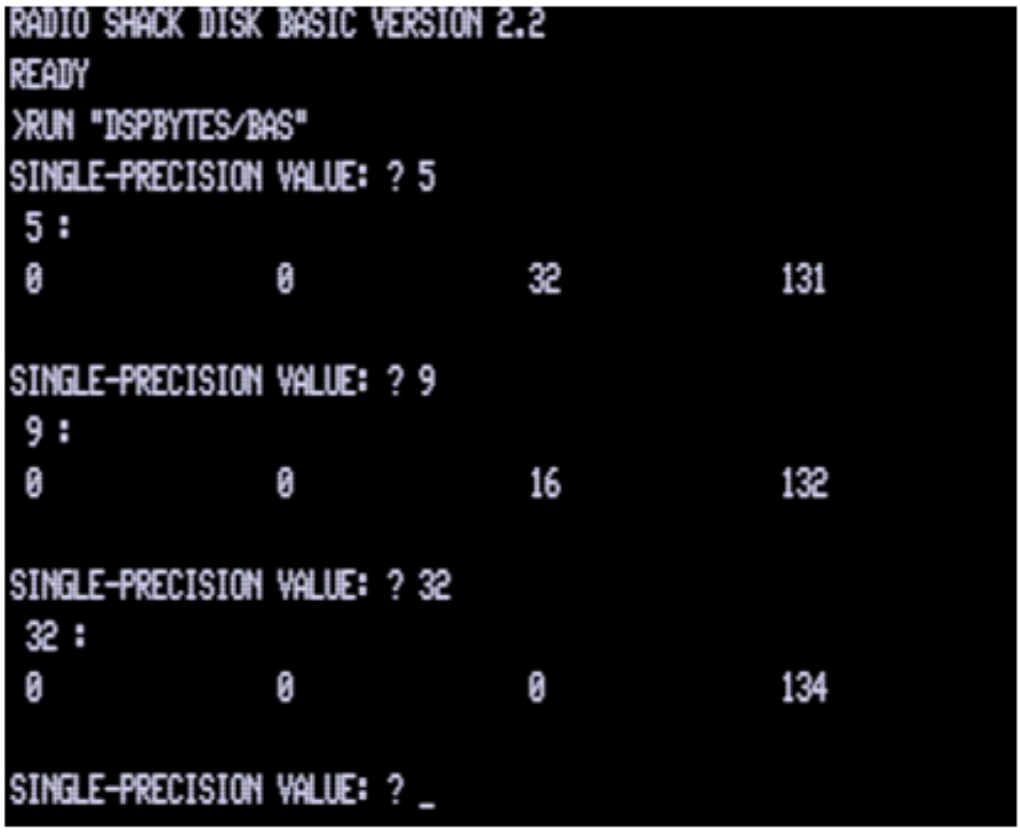
1. Put a Single Precision "32" in WRA1.
2. Put the S.P. value of "C" in BC/DE… wait. We have to multiply C by 9/5 first.

I'm old enough to remember not only primary school, but high school and college *without* calculators. We had to figure out everything "by hand", and learned to work our way from the innermost set of parentheses outward, meanwhile always remembering "My Dear Aunt Sally"—doing the multiplication and division before the addition and subtraction. That's what Assembly Language programmers have to do with arithmetic expressions. In this example, we'd have to start with the "9/5", then multiply that result by "C", then add "32". Just like in our first

set of 8 steps. We'll also need to "hand optimize" our order of execution so that we use as few "temporary variables" as possible. Temporary variables are necessary whenever WRA1 has to be loaded with some new value, and this saving-and-reloading operation will slow our code down.
Enough theory. Let's put our 8 steps into practical Z80 code.

1. Put a Single Precision "5" in WRA1, the Floating Point Accumulator.

The fastest way to do this is to use our BASIC program to see what set of bytes to put into a variable we'll call "FIVE". While we're at it, let's get the bytes for "NINE" and "THIRT2" ("32") as well:




```
FIVE    DEFB    0       ; "5" AS S.P.
        DEFB    0
        DEFB    32
        DEFB    131

NINE    DEFB    0       ; "9" AS S.P.
        DEFB    0
        DEFB    16
        DEFB    132
THIRT2  DEFB    0       ; "32" AS S.P.
```

```
        DEFB    0
        DEFB    0
        DEFB    134
```

Now we can move a Single Precision "5" into the Accumulator WRA1. Why not the "9"? The BASIC ROM "Divide" subroutine expects the Divisor (the number on the bottom) to be in WRA1, and the Dividend (the number on the top) to be in the DE/HL pair.

```
LINE30  EQU     $
;
;  LINE 30 PART 1: S.P. "5" -> WRA1
;
        LD      HL,FIVE
        LD      DE,WRA1
        LD      BC,4            ; MOVE 4 S.P. BYTES
        LDIR
```

Next, we load the BC/DE register pair grouping with the S.P. "9".

```
;
; LINE 30 PART 2: S.P. "9" -> BC/DE
;
        LD      DE,(NINE)
        LD      BC,(NINE+2)
```

We tell the ROM that we're working with Single Precision:

```
;
; LINE 30 PART 2B: INDICATE S.P. OPERATION
;
        LD      A,4
        LD      (NTF1),A        ; NUMBER TYPE FLAG= 4
```

Now we call call the ROM "Divide" routine. But first, make sure we "trap" any error (like "divide by 0") with a CALL to GETSPC first. Then CALL the Divide subroutine.

```
;
; LINE 30 PART 3A: CALL "DIVIDE"
;
        CALL    GETSPC          ; "ON ERROR RESUME NEXT"
        CALL    08A2H           ; WRA1= WRA1 / BCDE
```

We really should check for errors before assuming everything's fine. How can we tell? By testing BERROR for a non-zero value. Remember that BERROR / 409AH is set to 0 ("No error") by our INIT subroutine.

```
;
; LINE 30 PART 3B: CHECK FOR ERRORS
;
        LD      A,(BERROR)      ; 409AH
        OR      A               ; IS IT 0?
        JP      NZ,ERROR        ; IF NOT, REPORT ERROR & QUIT
;
;       ...CONTINUE: NO ERROR
;

;
;  ERROR REPORTING HERE
;
ERROR   EQU     $
        LD      HL,ERMSG        ; PRINT ERROR MESSAGE
        CALL    PRINT
        JP      FINISH          ; EXIT PROGRAM DUE TO ERROR
;
ERMSG   DEFM    'MATH ERROR: CAN'T CONTINUE'
        DEFB    0DH             ; NEWLINE
        DEFB    0               ; 0-TERMINATION
```

What's next? "4. Put the S.P. value of "C", that is "25", in BC/DE."

```
;
; LINE 30 PART 4: PUT S.P. VALUE OF "C" IN BC/DE
;
        LD      DE,(CSP)
        LD      BC,(CSP+2)
```

"5. Call the ROM MULTIPLY subroutine. The result will again be in WRA1."

```
;
; LINE 30 PART 5: MULTIPLY WRA1 * BCDE
;
        CALL    GETSPC          ; "ON ERROR RESUME NEXT"
        CALL    0847H           ; WRA1= WRA1 * BCDE
;
; LINE 30 PART 5B: CHECK FOR ERRORS
;
        LD      A,(BERROR)
        OR      A               ; IS IT 0?
        JP      NZ,ERROR        ; IF NOT, REPORT ERROR & QUIT
```

"6. Put an S.P. "32" in BC/DE."

```
;
; LINE 30 PART 6: S.P. "32" -> BC/DE
;
        LD      DE,(THIRT2)
        LD      BC,(THIRT2+2)
```

"7. Call the ROM Add subroutine."

```
;
; LINE 30 PART 7: ADD WRA1 + BCDE
;
        CALL    GETSPC          ; "ON ERROR RESUME NEXT"
        CALL    0716H           ; WRA1= WRA1 + BCDE
;
; LINE 30 PART 7B: CHECK FOR ERRORS
;
        LD      A,(BERROR)
        OR      A               ; IS IT 0?
        JP      NZ,ERROR        ; IF NOT, REPORT ERROR & QUIT
```

"8. Move the S.P. result in WRA1 to the "F" S.P. variable." First, we better make space for the Single Precision "F" variable.

```
FSP     DEFS        4           ; F AS S.P. VARIABLE
;
; LINE 30 PART 8: MOVE WRA1 -> FSP
;
        LD          HL,WRA1
        LD          DE,FSP
        LD          BC,4
        LDIR
```

There! We're done our calculations on Line 30. What's next?

```
40  PRINT C; "DEGREES CELSIUS =" ; F; "DEGREES FAHRENHEIT."
```

We'll have to split this line up into several procedures as well.

1. Convert "C" from Single Precision to a string, since our PRINT subroutine only understands 0-terminated strings.
2. PRINT the string version of "C".
3. Immediately after that on the same line, PRINT the "DEGREES CELSIUS =" message.
4. Convert "F" to a string.
5. PRINT the string version of "F".
6. Immediately after that on the same line, PRINT the "DEGREES FAHRENHEIT." message. Be sure to include the final newline character.

Here we go. '1. Convert "C" from Single Precision to a string, since our PRINT subroutine only understands 0-terminated strings.'

```
LINE40  EQU     $
;
; LINE 40 PART 1: CONVERT S.P. "C" TO A STRING
;
;       PART 1A: MOVE "C" -> WRA1
;
        LD      HL,CSP
        LD      DE,WRA1
        LD      BC,4
        LDIR
```

```
;
; LINE 40 PART 1B: CONVERT IT
;
        CALL    GETSPC          ; "ON ERROR RESUME NEXT"
        CALL    0FBDH           ; CONVERT WITH NO FORMATTING
;
; LINE 40 PART 1C: CHECK FOR ERRORS
;
        LD      A,(BERROR)
        OR      A               ; IS IT 0?
        JP      NZ,ERROR        ; IF NOT, REPORT ERROR & QUIT
```

RINT the string version of "C".' At this point, HL holds the address of the string result, so we ust print it out.

```
;
; LINE 40 PART 2: PRINT STRING VERSION OF "C"
;
        CALL    PRINT
```

mmediately after that on the same line, PRINT the "DEGREES CELSIUS =" message."

```
DEGCEL  DEFM    ' DEGREES CELSIUS = '
        DEFB    0
;
; LINE 40 PART 3: PRINT "DEGREES CELSIUS = "
;
        LD      HL,DEGCEL
        CALL    PRINT
```

might have noticed that we added a space before the "D" in ' DEGREES CELSIUS = '. BASIC matically places a space after a number when PRINTing it. Our ROM call does not do that, so ave to add the space.

onvert "F" to a string." We've done this to "C" in step 1. Here's the code for "F":

```
;
; LINE 40 PART 4: CONVERT "F" TO STRING
;
        LD      HL,FSP
        LD      DE,WRA1
        LD      BC,4
        LD      A,C
        LD      (NTF1),A        ; JUST TO BE SURE...
```

```
        LDIR                    ; "F" NOW IN WRA1
        CALL    GETSPC          ; "ON ERROR RESUME NEXT"
        CALL    0FBDH           ; CONVERT TO STRING
```

'5. PRINT the string version of "F".'

```
;
; LINE 40 PART 5: PRINT STRING VERSION OF "F"
;               HL ALREADY HOLDS ADDRESS OF STRING
;
        CALL    PRINT
```

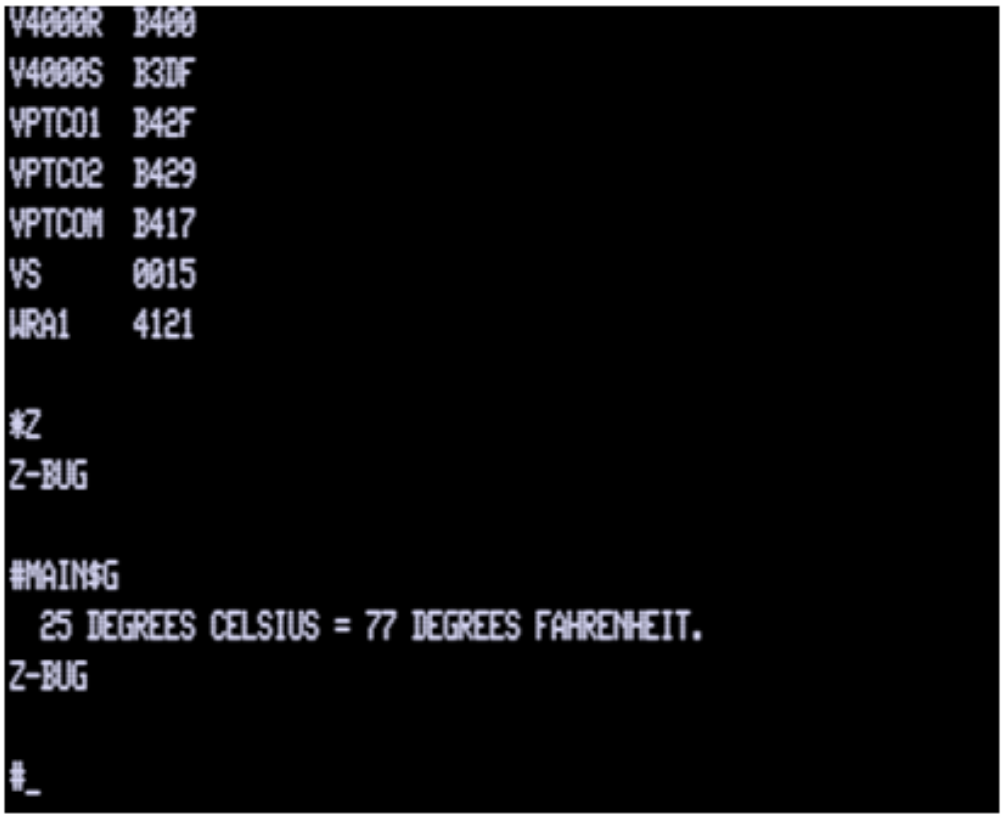
'6. Immediately after that on the same line, PRINT the "DEGREES FAHRENHEIT." message. Be sure to include the final newline character.'

```
DEGFHR DEFM     ' DEGREES FAHRENHEIT.'
        DEFB    0DH             ; NEWLINE
        DEFB    0               ; 0-TERMINATION
;
; LINE 40 PART 6: PRINT " DEGREES FAHRENHEIT."
;                       AND NEWLINE
;
        LD      HL,DEGFHR
        CALL    PRINT
```

Provided we wrote this in the framework of the code we've developed in the last 2 articles, we're just about done.

```
EXIT    EQU     $
        CALL    V4000R          ; RESTORE DOS EXITS,
                                ; IE, "FORGET MATH"
        JP      FINISH          ; EXIT THE PROGRAM
```

To quote the "Brochet" guy from TikTok, "It works! It worrrrkkkks!!" I test each line of code in Microsoft Editor Assembler Plus (and it's not surprising how often I miss something and have to edit). Here you see the tail end of the symbol table as I assemble into memory. The "Z" command takes me from the editor into the resident Z-Bug debugger. At that point, I set the program counter to MAIN—the label we placed at the beginning of our program—and use the "G" command to "Go". (About the extra spaces before "25 DEGREES CELSIUS...": The first is inserted by Z-Bug. I haven't figured out why, although I have a subroutine called "BAXUP" which moves the cursor back to the beginning of the video line. The second is provided by the ROM routine to print a number: it places a space instead of a minus sign for positive numbers.)



Can we speed this up any more? One thing about the formula

```
F = ( 9 / 5 ) * C + 32
```

is that it not only contains constant *values* like 9, 5, and 32, but it also contains a constant *expression*: "9 / 5". If we alter our BASIC "Display S.P. Bytes" program above with these line replacements:

```
100 SP!= 9 / 5     and
410 STOP
```

we'll get the byte values

```
102     102     102     129
```

We can save this constant result in

```
NINES   DEFB    102         ; 9 / 5 AS S.P. CONSTANT
        DEFB    102
        DEFB    102
        DEFB    129
```

and then alter our Assembly Language program this way. Delete from the "LINE 30 PART 1" comment right up to just before the "LINE 30 PART 4" section of code that will 'PUT S.P. VALUE OF "C" IN BC/DE', and replace it with:

```
;
; LINE 30 PART 1: CONSTANT "9/5" -> WRA1
;
LD      HL,NINE5
LD      DE,WRA1
LD      BC,4
LD      A,C
LD      (NTF1),A            ; 4: SINGLE-PRECISION
LDIR
```

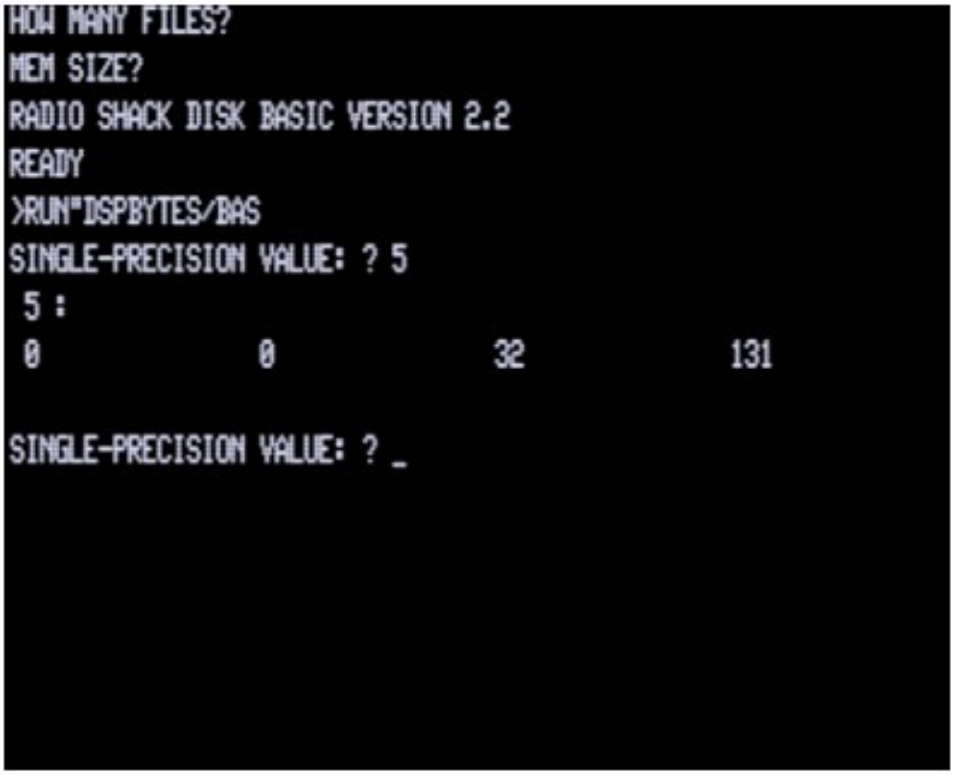
We'll come back to the Celsius – Fahrenheit program later, when we can input new values and make it more useful. Meanwhile in Chapter 6 of his book, Dr. Lien is teaching us about IF and THEN.

```
10 A = 5
20 IF A = 5 THEN 50
 30 PRINT "A DOES NOT EQUAL 5."
40 END
50 PRINT "A EQUALS 5."
```

In order to emulate this BASIC program in Z80 Assembly, we need code to compare two floating point numbers. Yes—I know—5 is an integer, but the Level I BASIC that Dr. Lien is teaching only supports Single Precision, and we're using his code to learn how to access the (Level II) BASIC ROM subroutines. So here goes:

```
10 A = 5
```

Let's store A as the 4-byte S.P. equivalent of 0005. We can use our BASIC program to discover each byte's value:




```
ASP     DEFB    0           ; "A" AS S.P.
        DEFB    0
        DEFB    32
        DEFB    131
```

Next:

```
20 IF A = 5 THEN 50
```

I suppose we need to create a constant named "FIVE". (We already know what its byte values will be, but for the sake of demonstrating a Single Precision Compare subroutine, we'll write it out in full:

```
FIVE    DEFB    0           ; "5" AS S.P. CONSTANT
        DEFB    0
        DEFB    32
        DEFB    131
```

Now we're ready to compare two Single Precision values, "A" and "5".

```
;
;       SPCMP – COMPARE VAL1, POINTED TO BY HL, WITH
;                       VAL2, POINTED TO BY DE
;       CONDITION       A       FLAGS
;       VAL1 < VAL2     -1      S / C
;       VAL1 = VAL2     0       Z
;       VAL1 > VAL2     1
;
SPCMP   EQU     $
        LD      A,4             ; TYPE: SINGLE PRECISION
        LD      (NTF1),A        ; NUMBER TYPE FLAG
        PUSH    DE              ; PRESERVE ADDR OF VAL2
        LD      DE,WRA1         ; PREPARE TO COPY
        LD      BC,4            ; VAL1 TO WRA1
        LDIR
        POP     HL              ; NOW ADDR OF VAL2 IN HL
        LD      E,(HL)          ; COPY VAL2
        INC     HL              ;     TO
        LD      D,(HL)          ;     BC/DE
        INC     HL
        LD      C,(HL)
        INC     HL
        LD      B,(HL)
        CALL    GETSPC          ; "ON ERROR RESUME NEXT"
        CALL    0A0CH           ; S.P. COMPARE
        RET
```

(I've implemented this as a subroutine just to keep the main program's source code a bit more readable. It's actually a copy of a pre-tested "macro" that I regularly use: that means you can use it as inline code if you wish by just deleting the RET instruction.)

As described in the source code above, if VAL1, the 4-byte Single-Precision value pointed to by HL, is *less than* VAL2, the S.P. value pointed to by DE, then A will have a value of -1 (0FFH), and the S and C flags will be set (C, "carry", is the important one here). If VAL1 is *equal to* VAL2, then the important flag is Z, which will be set to indicate that there is "zero difference" between the two. If you need to check that VAL1 is *greater than* VAL2, you'll need to confirm that neither C nor Z are set.

Here's how we use SPCMP in our example.

```
LINE20  EQU     $
;
;    "20 IF A = 5 THEN 50"
;
; LINE 20 PART 1: LOAD HL WITH ADDR OF "VAL1", A
;                 LOAD DE WITH ADDR OF "VAL2", 5
;
        LD      HL,ASP          ; A
        LD      DE,FIVE         ; 5
;
; LINE 20 PART 2: COMPARE THEM
;
        CALL    SPCMP
;
; LINE 20 PART 3: GO TO LINE 50 IF EQUAL
;
        JP      Z,LINE50
```

We'll get to LINE 50 down the road. Next is:

```
30 PRINT "A DOES NOT EQUAL 5."
```

Based on what we've learned so far, that's easy to implement.

```
NOTEQ   DEFM    'A DOES NOT EQUAL 5.'
        DEFB    0DH             ; NEWLINE
        DEFB    0               ; 0-TERMINATION

LINE30  EQU     $
;
; '30 PRINT "A DOES NOT EQUAL 5."'
;
        LD      HL,NOTEQ
        CALL    PRINT
```

Dr. Lien's next statement is

```
40 END
```

but in the case of our TRS-80 Assembly Language programs, we need to be sure to "clean up", ie, FORGET MATH and jump to the appropriate exit point. We'll do this:

```
LINE40  EQU     $
;
; "40 END"
;
        JP      EXIT
```

where EXIT is the same code as the previous program, above. Now, for LINE 50.

```
ISEQ    DEFM    'A EQUALS 5.'
        DEFB    0DH
        DEFB    0

LINE50  EQU     $
;
; '50 PRINT "A EQUALS 5."'
;
        LD      HL,ISEQ
        CALL    PRINT
;
        JP      EXIT            ; "END"
```

Next on Dr. Lien's agenda is to teach us about the INPUT statement, adapting his Celsius-to-Fahrenheit program to demonstrate user interaction.

```
10 REM * IMPROVED CELSIUS TO FAHRENHEIT CONVERSION PROGRAM *
20 INPUT "WHAT IS THE TEMPERATURE IN DEGREES CELSIUS" ; C
30 F = 9 / 5 * C + 32
40 PRINT C; "DEGREES CELSIUS = "; F; "DEGREES FAHRENHEIT."
50 GOTO 20
```

His two changes are the INPUT on line 20 and the GOTO 20 on line 50. This definitely makes the program more user-friendly: no more re-writing line 20 every time you want to calculate degrees F from a new value of C! (Remember, the BASIC code originally looked like this:)

```
20  C = 25
```

Now, before we get to INPUTing in Assembly Language, a brief rant. Even back in the late 1970s when all I had was a 16K Level II Model I, I loathed BASIC's INPUT statement, and coveted Disk BASIC's LINE INPUT instead. In fact, when Microsoft brought out their "Level III BASIC" upgrade for cassette-based machines, I immediately sent them my hard-earned $80. That supplement not only added line renumbering and fast drawing statements to BASIC, but an actual working

LINE INPUT! Why did I want LINE INPUT so badly? Simply—because it let you enter a line of text without editing it in the least, without burping over commas and misunderstanding other punctuation marks. You could then interpret the text however you wanted, without the dreaded

?REDO

error message. What's the reason for this rant? I don't have a ROM entry for INPUT, so I'm going to provide LINE INPUT instead.

CALL 05D9 Wait For Next Line

Accepts keyboard input and stores each character in a buffer supplied by caller. Input continues until either a carriage return or a BREAK is typed, or until the buffer is full. All edit control codes are recognized, e.g. TAB, BACKSPACE, etc. The calling sequence is: On exit the registers contain:

```
;
; GET NEXT LINE FROM KEYBOARD. EXIT IF BREAK STRUCK.
; LINE CANNOT EXCEED 25 CHARACTERS
;
SIZE   EQU     25          ; MAX LINE SIZE ALLOWED
       LD      HL,BUFF     ; BUFFER ADDRESS
       LD      B,SIZE      ; BUFFER SIZE
       CALL    5D9H        ; READ NEXT LINE FROM KEYBOARD
       JR      C,BREAK     ; JMP IF BREAK TYPED
       .
       .
BUFF   DEFS    SIZE        ; LINE BUFFER
       .
       .
```

HL	Buffer address
B	Number of characters transmitted excluding last.
C	Original buffer size
A	Last character received if a carriage return or BREAK is typed.

Carry Set if break key was terminator, reset otherwise.

If the buffer is full, the A register will contain the buffer size.

This ROM routine is quite useful, but there is one caveat: you have to provide an external "buffer", a region of memory that can hold enough characters for your user to type in

everything you need. On the other hand, the buffer can be up to 255 bytes long (the maximum positive value you can store in the B register, and it will report

- If the user pressed "ENTER" (ie, Return / Newline); and
- If the user pressed "BREAK"

Here's my "LINPUT" subroutine which takes advantage of this ROM call.

```
;
;       LINPUT -- INPUT A STRING TO BUFFER
;           POINTED TO BY HL, WITH MAX
;           SIZE OF 255 BYTES. 0-TERMINATE
;           STRING.
;           BASED ON MBD P. 16
;           RETURNS:
;           HL:     BUFFER  ADDR
;           BC:     # OF CHARS TYPED
;           A:      0DH IF TERM'D BY <ENTER>;
;                   01H IF TERM'D BY <BREAK>
;           C FLAG: SET IF <BREAK>
LINPUT  EQU     $
        LD      B,255           ; MAX $ SIZE
        CALL    05D9H
        PUSH    HL              ; SAVE BUFFER START ADDRESS
        PUSH    BC              ; SAVE BC RETURN VALS
        PUSH    AF              ; SAVE FLAGS
        LD      C,B             ; PREPARE TO INCREMENT HL
        LD      B,0             ; BY LENGTH OF STRING INPUT
        ADD     HL,BC           ; HL NOW AT STRING END
        LD      (HL),0          ; 0-TERMINATE $
        POP     AF              ; RESTORE FLAGS (ESP. "C")
        POP     BC              ; AND RETURNS FROM 05D9H
        POP     HL
        RET
```

My general-purpose LINPUT assumes a buffer size of 255 bytes, so if you're going to use it, make sure your buffer is at least 255 bytes long! After it CALLs the ROM routine, it will place a terminating 0-byte at the end of what was typed (including *after* an ENTER/Return/newline, but *not* including the Break key if that was typed). It's easy to catch if the user hit BREAK, ie to escape from the running program, by testing the Carry flag, as we'll see in our latest version of the Celsius-to-Fahrenheit program.

Pull up your copy of the C-to-F Assembly Language program. We're going to insert this code ahead of the the section of code that looks like

```
LINE30      EQU     $
;
;  LINE 30 PART 1...
;
```

with code to

1. Prompt the user for “C”;
2. Input a value for “C”;
3. Test if the user pressed BREAK to exit the program and jump to EXIT if they did;
4. Otherwise, convert the text they entered to a Single Precision representation of “C”;

which corresponds to LINE 20.

```
20 INPUT “WHAT IS THE TEMPERATURE IN DEGREES CELSIUS” ; C
```

Here we go.

```
PROMPT      DEFM    'WHAT IS THE TEMPERATURE IN DEGREES CELSIUS? '
            DEFB    0
```

(Note that we had to add a question mark and a space, which INPUT supplies.)

```
BUFFER      DEFS    255             ; BUFFER FOR LINE INPUT

LINE20      EQU     $
;
; '20 INPUT “WHAT IS THE TEMPERATURE IN DEGREES CELSIUS” ; C'
;
; LINE 20 PART 1: PROMPT USER FOR “C”
;
            LD      HL,PROMPT
            CALL    PRINT
;
; LINE 20 PART 2: INPUT A VALUE FOR “C”
;
            LD      HL,BUFFER
            CALL    LINPUT
;
; LINE 20 PART 3: TEST FOR <BREAK> & EXIT IF SO
;
            JP      C,EXIT
;
; LINE 20 PART 4: CONVERT STRING TO S.P.
;
```

Oh! We haven't written that conversion routine yet! Here it is:

```
;
;    STR2SP – CONVERT STRING TO S.P.
;         ARGUMENTS:
;         HL:      ADDR OF STRING
;         DE:      ADDR OF S.P. VARIABLE
;
STR2SP  EQU     $
        PUSH    DE          ; PRESERVE ADDR OF S.P.
        CALL    GETSPC      ; "ON ERROR RESUME NEXT"
        CALL    0E6CH       ; CONVERT
        POP     DE
        LD      A,(BERROR)
        OR      A
        RET     NZ          ; IF (NO ERROR):
        PUSH    DE           ; PRESERVE ADDR OF S.P.
        LD      A,(NTF1)
        CP      4
        JR      Z,ISSNGL        ; IF (NTF <> S.P.):
        CALL    GETSPC              ; "ON ERROR RESUME NEXT"
        CALL    0AB1H               ; CONVERT # TO S.P.
ISSNGL  EQU     $                   ; :ENDIF
        POP     DE
        LD      A,(BERROR)
        OR      A
        RET     NZ              ; IF (NO ERROR):
        LD      HL,WRA1             ; COPY WRA1 TO S.P.
        LD      BC,4                ; POINTED TO BY
        LDIR                        ; DE
        RET                         ; :ENDIF
                                ; :ENDIF
```

This subroutine is a bit more complex, because we might have to do either 1 or 2 conversions, depending on what was typed in. If the user typed a "25", for example, the initial string-to-number conversion at 0E6CH will happily translate it as an Integer, but in our case we want the final result to be Single Precision. So we check the Number Type Flag, and if it doesn't indicate "4", S.P., we have to call the "Convert Number To Single Precision" at 0AB1H. Also, there's error-checking after each conversion: no point in CALLing ROM code if our value is unrecognizable to it. It's also a good idea to check BERROR after calling STR2SP to ensure that there was a proper translation of the user input. Here's how we use it:

```
;
; LINE 20 PART 4: CONVERT STRING TO S.P.
;
        LD      DE,CSP          ; CSP =
        CALL    STR2SP          ;     VAL( BUFFER$)
```

Don't forget to test for errors after the conversion!

```
;
; LINE 20 PART 5: EXIT IF ERRORS
;
        LD      A,(BERROR)
        OR      A
        JP      NZ,ERROR        ; IF ERROR, REPORT & EXIT
```

The only other alteration is to copy this:

```
50 GOTO 20
```

by locating the end of our program:

```
;
; LINE 40 PART 6: PRINT "DEGREES FAHRENHEIT."
;               AND NEWLINE
;
        LD      HL,DEGFHR
        CALL    PRINT
```

and appending:

```
LINE50  EQU     $
;
; "50 GOTO 20"
;
        JP      LINE20
```

Now we know why we added all those "LINExx EQU $" labels. Handy!
I know…I promised code to convert 24-bit unsigned integers to floating point numbers. Hopefully, next article. (I should know better by now, don't you think?)

BetaGamma Computing

Specialists in all things that compute



Based in Corfu Greece

- Qualified IT Engineers
- Repair of all PC's and Laptops
- Software and Hardware Upgrades and Installation
- Broadband and Network Installation
- Supply of Custom built to order PC solutions

RETRO RESTORATION SPECIALISTS

Call Corfu 26610 26358 Mobile 698 755 8427

Email: bgcompute@aol.com • www.betagammacomputing.com

IRA HAS JUST POSTED THIS ITEM.

I AM SORRY TO ANNOUNCE THAT MATTHEW REED HAS PASSED AWAY. MATTHEW HAD DONE SO MUCH FOR THE TRS-80 COMMUNITY, AND FOR ME PERSONALLY, THAT THIS IS A HUGE LOSS.

MATTHEW HAD PREVIOUSLY MADE ALL OF HIS "REGISTERED" SOFTWARE FREE AND ENSURED THAT 100% OF HIS SOURCE CODE WAS PASSED INTO GOOD HANDS, SO NOTHING THERE IS LOST.

HIS TWO TRS-80 RELATED DOMAINS, WWW.TRS-80.ORG AND WWW.TRS-80EMULATORS.COM HAVE NOW BEEN REGISTERED TO ME, AND I AM CURRENTLY HOSTING THEIR PRIOR CONTENT, WHICH HE HAD CAREFULLY CONVERTED HIS 'WORDPRESS' TO HTML SO THAT I COULD JUST UPLOAD IT ONCE HE PASSED AWAY.



IN MAV'S HISTORY WORKSHOP

You Bought a Tandy Model 2000! What Now?



Part 1, by Ian Mavric

Some time ago[1] I wrote an article about how the Tandy TRS-80 Model 2000 Personal Computer(to be called T2K for the rest of this article) was the finest and most powerful TRS-80 ever made. It could have rightly been called the Model 5, a successor to the Model 4, as it had so many of the improvements TRS-80 enthusiasts had been

requesting already incorporated into the new design. Even Tandy hoped it would appeal to long time Model I,III,4 owners looking to update their computing workstations for the 2nd half of the 1980s. In this article I will talk about the history of the T2K marketing, it's pros and cons, and what to look for when buying one for your collection.

WHAT WAS THE TANDY 2000 AND WHY IS IT SO RARE?

The short story is that by 1983 every major computer manufacturer could not ignore that the world was moving towards MS-DOS systems, following the success of the IBM PC of 1981 and it's enhanced version, the IBM PC/XT of 1983. Most manufacturers hurried to release PC-compatible systems. It was the early days of MS-DOS, and compatibility was a grey area. Some were quite compatible, running some 90% or more of MS-DOS software without problems, and others were sold as 'improved' designs which had better performance than the PC/XT at the expense of compatibility. Tandy bet the house on this latter design, producing a system which far exceeded the PC/XT in all areas except for compatibility. It failed in the market place even though it was available for purchase until 1987. This was mainly due to so many being over produced in 1984 that it took until 1987 for the warehouse to sell out of them.

Technically the T2K is called a high-performance MS-DOS system, and based around the 80186 16-bit CPU running at 8MHz. It's a proper 16-bit CPU for both data bus and internally, compared to the PC/XT 8-bit external bus and 16-bit internal processing 8088

4.77MHz CPU. Maximum memory for the T2K was 768K vs. 640K for the PC/XT. The T2K disk drives held twice the amount of information as that of the PC/XT, and the graphics options, when fitted, offered 640x400 in 8 colours or monochrome, whereas the PC/XT's 'CGA' was 320x200 in 4 colours, and non graphics in 'MDA' monochrome. The T2K keyboard had 90 keys vs. the PC/XT Model F keyboard's 84 keys. What this all added up to was a very fast computer with great storage and ergonomics which was a pleasure to use, if the software you wanted to use was available for it. More on that later.

Price Comparison†	Tandy 2000	IBM Personal Computer
Base Unit	$2750	$2104
2nd Drive	Included	$529 (320K)
Monochrome Monitor	$249	$345
Display/Printer Adapter	Included	$335
128K RAM (Total)	Included	$165
RS-232	Included	$120
MS-DOS 2.0	Included	$60
Total Cost*	$2999	$3658
Feature Description		
Internal Memory	128K Standard	64K Standard
Disk Capacity Per Drive	720K	160 or (320K optional)
Clock Speed	8 MHz	4.7 MHz
True 16-Bit Microprocessor	Yes (80186) 16 bit/16 bit data path	No (8088) 16 bit/8 bit data path
User-Available Expansion Slots*	4	2
Graphics Options		
Color Resolution	640 x 400	320 x 200
Number of Colors	8	4
Monochrome Resolution	640 x 400	640 x 200
Graphics on Monochrome Monitor	Yes	No

*Comparable IBM configuration with monochrome adapter and display, communications adapter, two 320K disk drives and 128K RAM. †Manufacturer's pricing as of 9/1/83.

Photo: T2K comparison with the IBM PC 5150 (source: RSC-11)

WHAT TO LOOK FOR WHEN YOU FIND ONE ON eBay?

Despite being a failure in the marketplace the T2K is not hard to find because there was some 50,000 of them sold, and their high prices meant many have been kept and stored long after they were replaced by newer machines. They come out of the attic regularly to

the tune of one every 1-2 months on eBay. You just need to be patient and decide what you want your T2K 'experience' to be.

Two versions of the T2K were released: a dual floppy-drive 128K "low end" version which carries the catalog number 26-5103, and an "upmarket" version called the Tandy 2000 HD (T2K HD) which had an internal 10Mb MFM hard drive, 256K Ram, and it's catalog number was 26-5104. There were also two choices for monitor, a high-resolution monochrome monitor called the VM-1 and a high-resolution colour monitor called, appropriately, the CM-1. Unlike a Model I, III, or 4 which included the monitor with the computer, with the T2K it was an additional cost item, as was everything else with the T2K.

PRICING AND COMPARISON WITH WHAT CAME BEFORE IT

Microcomputers, as we knew them from approximately 1977 with the trinity of home computers were released, typically had an 8-bit CPU. IBM, when they released the 5150, named it the 'Personal Computer'. PCs typically had 16-bit CPUs based around Intel's '86 architecture. By the time the T2K was released the title 'Personal Computer' was synonymous with any machine based around an Intel CPU, so the T2K was naturally called one.



Since the T2K was seen at Radio Shack to be a stepping-up point from the Z80 8-bit micros, RS figured a lot of people who used a Model 4 would earmark the T2K as their next computer.

Contrasting the Model 4 and the T2K is not really a fair comparison, but let's try it anyway. For my chart I am comparing a 64K dual-drive Model 4 used in a home or small business with applications to run WP, Spreadsheet, and DBM. To try to keep some semblance of similarity I will take a plain dual-drive 128K T2K, add a monochrome monitor, and three popular WP, SS and DBM apps of the time. Software is interesting because unless someone is after some ultra complex spreadsheet with thousands of cells and numerous re-calculations, VisiCalc and Lotus 1-2-3 basically do the same job as well as each other. If you have thousands of cells then the more powerful machine would have a big advantage. Likewise with DBM if you are only working on smallish databases the speed of Profile 4+ compares favourably with dBASE II, but if you have an enormous database the power of dBASE II shines through. Finally WP in the text era (pre-dating true-type fonts and GUI era) mainly needs to cover justification and proportional printing, bold, underline and italic text, as well as simple macros etc., so for this I have chosen

SuperScripsit on the Model 4 up against an early text version of MS Word on the T2K.

One thing you quickly see is that the price for software in the early days of the IBM PC/XT and AT was a lot more expensive than similar programs for an 8-bit TRS-80. This was actually one of the main contributing factors for people as late as the early 90s hanging onto the Model 4s instead of buying a PC and all the expensive new software needed.

	Model 4 1984	T2K 1984	Model 4 1985	T2K 1985
Computer	$1,999	$2,750	$1,299	$2,750
Monitor	Built in	$249	Built in	$249
Word Processor	SuperScripsit $199	MS Word $375	$199	$375
Spreadsheet	Visicalc $199	Lotus 123 $495	$99.95	$495
Database	Profile 4+ $249	dBASE II $595	$249	$495
	$2,646	**$4,464**	**$1,846.95**	**$4,364**
	Model 4D 1987	T2K 1987		
Computer	$1,199	$1,599		
Monitor	Built in	$199.95		
Word Processor	SuperScripsit $199	MS Word $375		
Spreadsheet	Visicalc $99.95	Lotus 123 $495		
Database	Profile 4+ $249	dBASE II $495		
	$1,746.95	**$3,163.95**		

(I have deliberately left out a printer because there was a wide variety of dot matrix and daisy wheel printer options which worked equally well on both computers. Buyers would determine their own printing requirements and buy accordingly.)

So let's put ourself in the mind of a Model III (or Model I) owner in early 1984 and they are unsure whether to buy a Model 4 or a T2K system. They already use Scripsit, Profile, and VisiCalc. They also want to re-use their printer. A shiny new Model 4 with software will cost $US2,646.00 and a T2K will set them back an eye-watering $US4,464.00 If the Model III owner waits a year things get even better as the Model 4 with software has dropped to $US1,846.95, nicely under the psychological $US2,000 mark. You could even upgrade the Ram on the Model 4 to 128K for $US79.95 without breaking $US2,000. By contrast the T2K system has not budged as much, it has come down a little but is still $US4,364.00

It could be argued that the T2K and PCs in general offer much more sophisticated software than on the TRS-80, and while this may be true, as eluded to earlier, for most home users and even small business applications the requirements and results of the software on both systems could be considered roughly equivalent. It's also worth mentioning that in both the Model 4 and the T2K we are talking about dual-floppy machines. Once your databases (in particular) get past a certain size a hard drive becomes mandatory for efficient use of the computer. This was exemplified on the Model III with Profile III+ HD, which was necessary once databases got too large to be practically run off floppy disks.

1987 was the last year the T2K featured in the RSCC, and it's high performance was still well and truly able to outrun the Tandy 1000 series. For the software which ran on it, it often ran better on the T2K than an IBM PC/AT or Tandy 3000. However being a 4 year old computer Radio Shack wisely chose to discount the rest of them and clear the space in the warehouse. The last part of my price chart skips 1986 and looks at what a buyer could get for their money in 1987. The Model 4 has been updated to the 4D by now, and the disk

drives hold more data each. The cost of the 4D and software has come down to $US1,746.95, and the T2K has also had it's biggest price drop to $US1,599.00 for a dual-drive (now) 256K T2K. Add software and a monochrome monitor brings that up to $US3,163.95 which is much better than 1984/5 but still a considerable jump on the Model 4D. One of the most convincing reasons to have a PC over a TRS-80 was software compatibility with work IBM PCs and ATs, however because this was an issue with the T2K that was less of a selling point than you would think. The disk format incompatibility was particularly annoying because if you used Lotus 1-2-3 at work on an IBM PC/XT you couldn't just bring home a floppy disk and stick it in your T2K. Though there were work-arounds for the different disk formats, the process is tedious.

My chart shows why the Model 4 continued to be a strong seller during that 1984-87 period. On a side note the Tandy 1000 was released to remedy most of the incompatibility issues of the T2K but it too was a system which $US1,199.00 bought you a base unit with keyboard, and you had to add a monitor, additional disk drive and hardware whereas a Model 4 included everything needed to start computing.

HOW ABOUT HARD DISK OPERATORS?

As mentioned earlier there was a hard-disk based T2K for professional operators and those who could afford to re-mortgage their house to get one. The T2K HD included a 10Mb MFM hard drive, and took away one of the 720K drives. Ram was increased to

256K and the machine came with MS-DOS 2.10 Price was increased too to a staggering $US4,250.00



Picture: T2K featured in RSC-11, for the 1984 model year. Monitor and mouse are not included btw.

For those who like to make comparisons, that price is up there with the TRS-80 Model 16B. A single-drive 16B is $US3,999.00 and a dual-drive 16B is $US4,699.00 Remember you have to add an extra cost monitor to your T2K. Also, anyone who has used a hard-disk based IBM PC or compatible with 256K Ram knows that is a bare minimum of Ram, and it is recommended to upgrade to 512K or preferably 640K, and most people did. In 1984 a 256K Ram card for a T2K cost $US499.00 (cat. no. 26-5161) though that did come down somewhat to $US149.95 by '87 and two could be added so that along with 256K on the motherboard you could max-out the T2K to 768K of Ram.

Hard disk computer	Model 4 1-dr 1984	T2K HD 1984	Model 4 1-dr 1985	T2K HD 1985
Computer	$1,699	$4,250	$1,099	$4,250
Hard drive	5Mb $1,999	Built in	15Mb $2,495	Built in
Monitor	Built in	$249	Built in	$249
Word Processor	SuperScripsit $199	MS Word $375	$199	$375
Spreadsheet	Visicalc $199	Lotus 123 $495	$99.95	$495
Database	Profile 4+ $249	dBASE II $595	$249	$495
	$4,345	**$5,964**	**$4,141.95**	**$5,864**
	Model 4D 1987	T2K HD 1987		
Computer	$1,199	$2,499		
Hard drive	10Mb $699,$449	Built in		
Monitor	Built in	$199.95		
Word Processor	SuperScripsit $199	MS Word $375		
Spreadsheet	Visicalc $99.95	Lotus 123 $495		
Database	Profile 4+ $249	dBASE II $495		
	$2,894.95	**$4,063.95**		

Ah yes, now I remember why so few people had hard drives in the 80s: the obscene cost just to have 5-15Mb of storage. It's hard to make fair calculations but since the T2K HD only had a single floppy drive I compared it to a single-drive Model 4 (yes, there was such a thing) with an external hard drive. In 1984/5 it cost you nearly 6 grand to have a T2K HD and a still significant $US4,000+ for a Model 4 with hard disk. This got better by 1987 when a Model 4D hard disk system comes in at under $US3,000.00 and the T2K HD drops to just over $US4,000.00

DESKMATE

It could be argued that all the expensive software could be left out and just run DeskMate on both the T2K and Model 4, which would save a lot of money. Whether the functionality of DeskMate could

be compared to that of the likes of MS Word, Lotus-123 or Profile 4+ is debatable, it is generally acknowledged that DeskMate for the T2K was far superior to DeskMate for the Model 4.

GRAPHICS WAS WHERE THE T2K REALLY SHONE

Until VGA graphics (640x480) came along half-way through 1987 the T2K has the best graphics on any PC, and that was from 4 years before VGA. Even EGA (640x350) and PCjr/Tandy 1000 (640x200) had fewer pixels than the T2Ks 640x400. In fact when the T2K was released the closest thing was the IBM CGA adapter which had a 320x200 pixel matrix. It's worth mentioning the Hercules graphics card which added 720x350 but was monochrome only. The T2K could display 8 colours out of a palette of 16. To display such high resolution in colour a suitably hi-res monitor was needed and this came in the form of the CM-1, which had a very sharp picture but was also seriously expensive.

Upgrading the T2K for graphics work was not for the faint hearted. Just like everything else related to the T2K, it was expensive. Once you had decided to go colour and recovered from the advertised price of the CM-1 at $US799.00 you read the fine print and learn you need to add a graphics option card to the computer. This card (cat. no. 26-5140) added what was called B&W graphics and was priced at $US449.00, and then if you wanted to take advantage of the colour graphics you needed a graphics board memory kit (cat. no. 26-5141) for $US199.00 In case you don't have a calculator handy these three items added $US1,447.00 to the T2K. It's no wonder finding this

monitor and graphics card set up for colour graphics is truly difficult and rare.

A note for would-be CM-1 buyers when you find one for sale on eBay or other online vendors: you need to make sure the special video cable is included. It's completely different to any other video cable for other monitors, and you can't use the CM-1 without it. Technically it's an EIAJ 8-pin connection which is a common video monitor connection used in Japan in the early 1980s. It is therefore no surprise either that the CM-1 is Made in Japan. It appears to be a Mitsubishi monitor, made with any changes requested by Radio Shack and labelled as such as a Tandy TRS-80 CM-1 High-Resolution Color RGB Monitor, and later the TRS-80 was dropped to become simply the Tandy CM-1 High Resolution RGB Color Monitor.

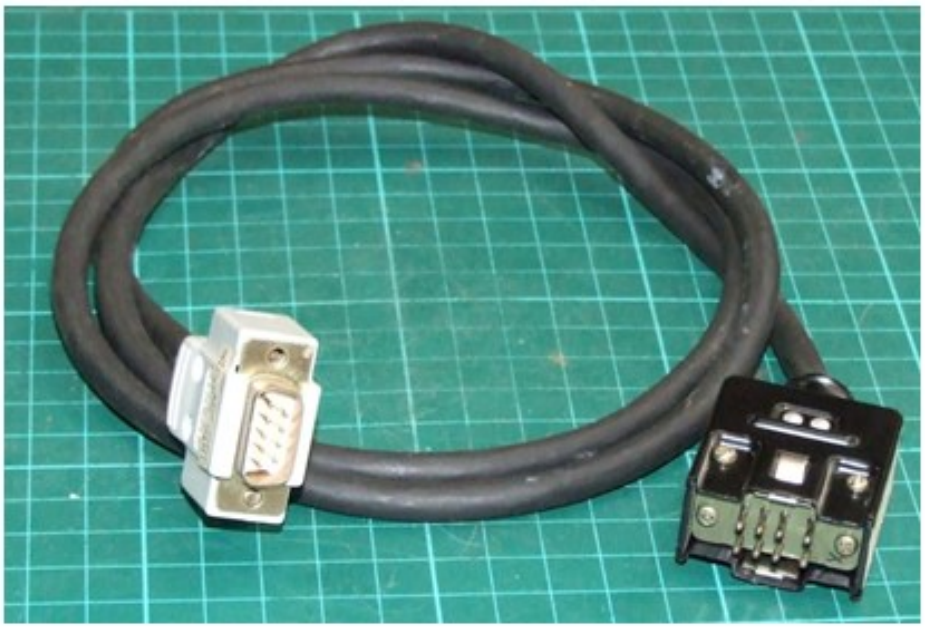


Photo: the special CM-1 monitor cable needed for the T2K

If you find a CM-1 that is broken the Service Manual is easy to find online[2]. Also a video by Adrian Black's Digital Basement shows how to repair one of these monitors[3].

WAS THIS A COMPUTER FOR RICH PEOPLE ONLY?

My throw-away statement (above) had a small amount of truth to it. The T2K was not the most expensive computer sold by Radio Shack, but it was up there. By the time it was equipped with a hard drive it was only a little less expensive than the TRS-80 Model 16B. But the 16B is a straight medium- and large-business computer with little appeal to home users. The T2K did have business credentials but was still a 'Personal Computer' that was released hoping to appeal to home users as much as anything else. The simple fact that if you outfitted a T2K with hard drive and CM-1 monitor and it's necessary cards the cost could easily hit $US6,500.00 and in 1984 that was the price of a decent small imported car like a Mazda GLC (323).

NEXT TIME

In the next instalment of this series we'll take a close look inside the computer and how it expands. I'll also examine some of the software which was available that really made the machine so awesome, and talk about why the BIOS was so incompatible which lead to so much software not working on it.

Ian Mavric / ianm@trs-80.com

Further reading:



Ian Mavric is an IT Specialist who also restores and collects TRS-80's and classic cars. He lives with his wife and kids in Melbourne, Australia.

1. http://www.fabsitesuk.com/tandy/trs8bit_years01to07.pdf pp.427-431, 2013.

2. https://archive.org/details/cm-1-service-manual-26-5112_202405

3. https://www.youtube.com/watch?v=Ulkpu9gu5DI



Alan Wallace has just released version 3.99922 of his much enhanced *STAR TREK* game



https://www.youtube.com/watch?v=ZE3nMxTLgjk



E.T. FONEHUME

HI DUSTY. I WAS CHATTING TO THE GRANDKIDS THE OTHER WEEKEND AND THE PROBLEM OF CHOOSING AN 'ACCEPTABLE' PASSWORD CAME UP. ONCE YOU'VE EXHAUSTED KNOWN NAMES OF PEOPLE, PETS, AND ADDRESSES ETC. WHAT DID I DO (BACK IN THE DAY). DO YOU REMEMBER THAT DAVE HAD A SYSTEM WHICH WAS QUITE RANDOM BUT STILL RELIABLE, OF USING HEADLINES FROM DAILY PAPERS, FOR EXAMPLE "POLICE CLASH WITH DRUNKEN REVELLERS' WOULD BE 'PCWDR'

NOT LONG OR VARIABLE ENOUGH BY TODAY'S STANDARDS SO HERE'S A ONE-LINER TO CREATE A PASSWORD USING ANY REQUIRED LENGTH FROM A POSSIBLE 73 CHARACTERS!

I'VE INCLUDED AN EXAMPLE OF 5, 13 CHARACTER PASSWORDS. AS THIS IS MADE TO RUN ON AN UNMODIFIED M1, WITHOUT LOWER CASE (EVEN ON TRS80GP!), THE LAST 26 LETTERS IN THE VARIABLE A$ ARE LOWERCASE. EASY ENOUGH TO DO USING TRS80GP BY LLISTING THE PROGRAM, EDITING THE LETTERS TO LOWERCASE, COPYING THE 'NEW' PROGRAM AND JUST PASTE IT BACK INTO BASIC

```
0 CLEAR500:RANDOM:CLS:INPUT"LENGTH OF P/W";L:A$="!$%&*()+-@#?012
3456789ABCDEFGHIJKLMNPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZ":FORX
=1TOL:B$=MID$(A$,RND(73),1):PRINTB$;:LPRINTB$;:NEXT
```

```
!ELkKaYFAV?gy
EYfAPuN)pLuko
*wz%+5GxL%o4Z
Y4tp02AH24Itz
fdlX3Qh5(&Uj(
```

Floppy Days
Vintage Computing Podcast
with Randy Kindig

Pull out a vintage computer and compute as if it were yesterday.

Subscribe Today!

Tandy-specific Episodes:

- TRS-80 Model I
- TRS-80 Model II
- Tandy Pocket PC
- Steve Leininger and Don French interviews
- The MISE with Peter Bartlett
- CoCo
- Boisy Pitre and Bill Loguidice book interview
- CoCoFest and Tandy Assembly coverage
- and much more...

https://www.floppydays.com

https://www.soigeneris.com/universal-backpack-drive



Universal Backpack Drive

The Backpack is a small SD card 'disk drive' for vintage computers that runs from a single AA battery

Choose Options

- ☐ Preloaded 8GB uSD card [+$5.00]
- ☐ Spare CR1025 RTC Battery [+$1.00]
- ☐ Tandy WP-2 DB9 to DB25 adapter [+$5.00]
- ☐ Sinclair Z88 DB9 t0 DB25 adapter [+$5.00]
- ☐ Epson PX-8 / HX-20 adapter [+$10.00]

$65.00

TRS-80

Repairs & Upgrades

HNC qualified electronics engineer with 40 years TRS-80 experience.
Enquiries to: trs80@shanefoster.co.uk

THE 'COMPETITION-COMPETITION'



WOULD YOU LIKE TO WIN THIS RARE AND LIMITED EDITION, U.K. ROYAL MAIL, INFORMATION TECHNOLOGY, FIRST DAY COVER, FROM SEPTEMBER 1982? ALL YOU NEED TO DO IS TO SUGGEST THE THEME FOR THE 2026 COMPETITION.

JUST SEND IN YOUR ENTRIES TO THE 'TRASHTALKERS'

George@48k.ca

AND THE PERSON WHO COMES UP WITH THE BEST IDEA WILL RECEIVE, VIA SNAIL-MAIL, THE ABOVE PRIZE!

LOOKING FOR FAST, INEXPENSIVE, UNLIMITED MASS STORAGE FOR YOUR TRS-80 MODEL I/III/4/4P/4D?

The amazing

"FreHD"

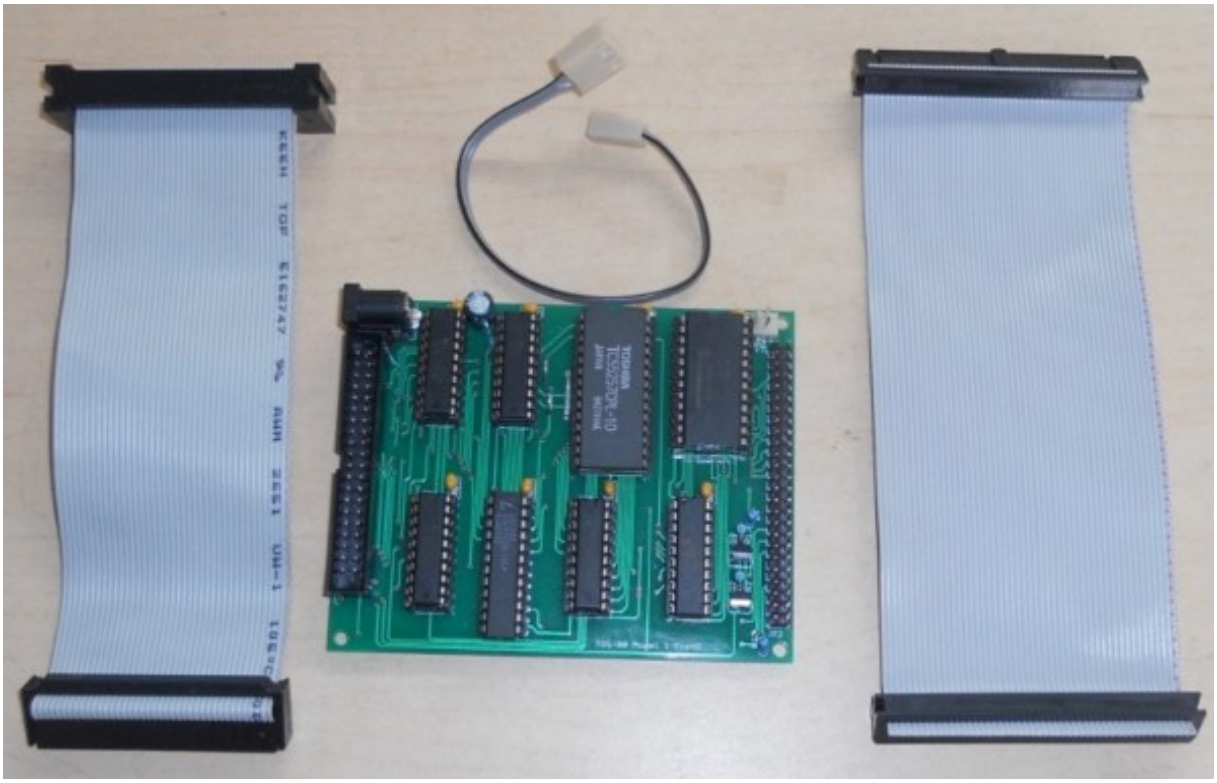


-Emulates a TRS-80 hard drive, but faster than any hard drive!
-Works with your favourite DOS (LS-DOS, LDOS, CP/M, Newdos/80 2.5)
-Uses SD card for storage medium
-Bonus free Real Time Clock function!
-Designed in Belgium and proudly built and shipped from Australia
-Kit form or fully assembled

Order yours today
http://trs-80.com.au/trs80/emulator.htm

“Quinnterface” Mini Expansion Interface for 16K Model 1 FreHD users.

- COMPLETLY ASSEMBLED AND TESTED -



This is THE perfect device for all Model 1 users who own a 16K Level II unit, (which is most of us!), but no Expansion Interface or disk drives, especially if you don't want to modify your M1 with upgraded boot ROM or memory upgrade.

The ‘Quinnterface’, developed by J. Andrew Quinn from New Zealand, adds 32K RAM and auto-boot functionality to you FreHD.

U.K. And Europe, contact Bas. at BetaGamma Computing.
U.S.A. And rest of the world, contact Mav. At “The Right Stuff”

EBAY.CO.UK 'BARGAINS'

HERE ARE JUST SOME OF THE MORE INTERESTING/UNUSUAL TANDY EBAY LISTINGS, SINCE OUR LAST ISSUE.
WERE YOU ONE OF LUCKY ONES?



£15



£35



£57



£10



£300



£86
NS



£60



£86
NS



£100
NS

POWER SUPPLIES

FOR THE TRS-80 MODEL 3 / MODEL 4/4D/4P



* Uses modern day MEAN WELL power supply units
* 50w for Model 3 version and 65w for the Model 4 version
* The Model 4 version uses a SEPARATE +12V POWER SUPPLY (prevents video "spasm")
* No more RIFA CAP explosions
* No more COLD solder joints
* Units are DIRECT REPLACEMENTS for the original power supplies in all models
* Extra +5V output connectors (1 on Model 3 and 2 on Model 4) for Gotek/freHD boards
* For more information and pricing contact

newSOFT

https://jaynewirth.wixsite.com/newsoft (website)
jaynewirth@verizon.net (email)



MICRO-80 UK Subscription Dept.
24 Woodhill Park Pembury Tunbridge Wells Kent TN2 4NW

LOOK what you get when you subscribe to MICRO-80 the monthly magazine for TRS-80 and Video Genie. Now in the 3rd year of publication!
All new subscribers (and existing ones when they renew) receive free of charge a Software Cassette complete with 62 page user manual featuring
★ **Level I in Level II** – Convert your Level II TRS-80 to operate as a Level I machine.
★ **Copier** – Copies Level II System tapes, irrespective of where they load in memory.
★ **Z80 MON** – A low memory, machine language monitor.
★ **Improved Household Accounts** – Powerful enough to be used by a small business.
★ **80 Composer** – A music generating program.
★ **Plus Two Games** – Poker and Cube (a version of the Rubiks cube for Disk users).
Order just the magazine or take it each month complete with the published programs ready-to-load on cassette or disk.

Please enrol me for an annual subscription and send me my FREE cassette program. I enclose £16.00 ☐ (magazine only) or £43.60 ☐ (magazine and cassette edition) or £75.00 ☐ (magazine and disk edition).
(Enclose your cheque/P.O. made payable to MICRO-80 and send to the above address.)
Software offer, and prices apply to U.K. residents only. Overseas subscription rates on application.

Name .....
BLOCK CAPITALS PLEASE
Address .....
.....
PCW/9/82

TJ's WORKSHOP

NUMBER FORMATTING

I was interested in D Gayler's program in the June TJ's workshop to print numbers, including values less than 0.01, to three places of decimals. If you input some numbers greater than 1, the decimal points no longer fall in line. If you input numbers between 0.0095 and less than 0.01 the system fails altogether and prints .00:

The effect sought by D Gayler, with the additional benefits of showing a leading zero for quantities less than 1 and keeping the decimal points in line when some values are greater than 1, can be achieved using PRINT USING as shown in Program 1.

In some circumstances it could be desirable for the actual number entered to be printed without rounding. Program 2 prints out all the entered figures without rounding and with all trailing zeros supressed to avoid giving a false idea of accuracy.

These programs were run on a TRS80 Level II.

Peter Davy

```
10 LPRINT"NORMAL";TAB(15)"IMPROVED"
20 INPUT N
30 LPRINT N;
40 LPRINT TAB(15) USING"##.###";N
50 GOTO 20
```

```
NORMAL          IMPROVED
 12             12.000
 12.2375        12.238
 9.5E-03         0.010
 9.2E-03         0.009
 .543            0.543
 4.67            4.670
 5.786           5.786
 .2039           0.204
 9.87654         9.877
 8.87654         8.877
```

```
10 LPRINT"NORMAL";TAB(15)"IMPROVED"
20 INPUT N
30 IF N=INT(N) THEN A$="##":GOTO 60
40 IF INT(N)=0 THEN M=N+1 ELSE M=N
50 A$="##."+STRING$(LEN(STR$(M))
     -LEN(STR$(INT(M)))-1,"
60 LPRINT N;
70 LPRINT TAB(15) USING A$;N
80 GOTO 20
```

```
NORMAL          IMPROVED
 12             12
 12.2375        12.2375
 9.5E-03         0.0095
 9.2E-03         0.0092
 .543            0.543
 4.67            4.67
 5.786           5.786
 .2039           0.2039
 9.87654         9.87654
 8.87654         8.87654
```

ROUNDING

With reference to D Gayler's note about printing floating point numbers to three decimal places (*PCW* June) perhaps some of your readers do not know that B=INT (A*10 ↑ D+0.5)/10 ↑D gives B the value of A rounded to D decimal places.

Peter Howard

TRS-80 EXIT

Here's a routine to deal with situations where your main program calls subroutine A, which then calls subroutine B, then subroutine B wishes to return straight back to the main program without going through subroutine A.

Some Extended Basics allow this, using a POP or EXIT command which deletes from the stack all information concerning the last GOSUB statement executed. So in our example a POP followed by a RETURN in subroutine B would go straight back to the main program.

Program listing 1 contains a short 12-line program to implement such a command on a 16k LII TRS-80.

If the program is typed in and executed it will reserve memory at the top of memory for a short machine code routine; it will also POKE the routine into memory and link the routine to the LINE command, which is only used if you have disks.

The routine poked into memory is only 10 bytes long, and all that it does it to clear all information concerning the last GOSUB from the stack before returning to the program.

Program listing 2 illustrates how the statement is used. If the LINE command in line 50 were not present then the RETURN in line 60 would return execution of the program to line 30, but the LINE command in line 50 clears all information about the GOSUB 50 in line 30 from the stack.

Once program listing 1 has been run the machine code routine will stay in memory until the computer is switched off or crashes — a NEW command will not delete the routine.

The program is short enough to load every time you have a session with your computer and you do not need to set the memory size as the program does it itself. It was written on a Model I, but it should work on a Model III or a Video Genie without any alterations.

Tim Pile

```
10 PRINT"LINE 10":GOSUB 30
20 PRINT"LINE 20":END
30 PRINT"LINE 30":GOSUB 50
40 PRINT"LINE 40":RETURN
50 PRINT"LINE 50":LINE
60 PRINT"LINE 60":RETURN

RUN
LINE 10
LINE 30
LINE 50
LINE 60
LINE 20
```

Listing 2

```
0  POKE 16562, 127: POKE 16561, 245  'SET TOP OF MEMORY
1  POKE 16545, 127: POKE 16544, 195  'SET STRING SPACE
2  CLEAR                             'REINITIALISE
3  FOR I = 32758 TO 32764            'THIS LOOP POKES
4  POKE I,51                         'SEVEN "INC SP"'S
5  NEXT I                            'INTO HIGH MEMORY
6  POKE 32765, 195                   'THIS POKES THE
7  POKE 32766, 30                    'CODE FOR
8  POKE 32767, 29                    ' JP 1D1EH
9  POKE 16803, 195                   'THIS LINKS THE
10 POKE 16804, 246                   'LINE STATEMENT
11 POKE 16805, 127                   'TO BASIC
```

Listing 1

MICROTAN IMPROVED USR

This routine greatly extends the USR(I) instruction of Microtan Basic, whilst using only six bytes of user RAM. It enables more than one machine-code subroutine to be called from Basic without repeatedly POKEing zero page.

To use, enter the following code using Tanbug's M command:

IFFA 20 F5 DF JSR$D5F5
1FFD 6C 33 00 JMP ($33)

Enter Basic, answering 8186 to the memory size prompt. To initialise the subroutine, POKE 34,31: POKE 35,250. This can be done as part of a program or directly. Convert the start address of the machine code into decimal, and use it as the argument of the USR instruction. For example, to call the keyboard scan routine of Tanbug, use J=USR(65018).

If this routine is to be incorporated into a home brew toolkit, then the only modification necessary is the start address poked into memory locations 34 and 35.

Clive Jones

The OFFICE INFORMATION CENTRE

- Word Processing
- Accounting Systems
- Financial Planning

Model II is British Telecom Approved for Datel 200,600 and 2412 Services



Tandy COMPUTER CENTER

Availability - TRS-80™ Computers and accessories available at most of our 300 UK outlets. Some listed below.

Software - Tandy because TRS-80 has the biggest computer library of ready-to-run programs. No experience needed.

Expandability - Buy your ideal sized computer and never replace it. Simply expand it as your needs grow.

Systems From £2699
All Prices Include V.A.T.

Training • Classrooms • On-Site Servicing • Leasing

YOUR NEAREST TANDY COMPUTER CENTRE

BIRMINGHAM 021-643-5368/5362
Edgbaston 021-455-9175
BRADFORD 0274-28431/2
BRISTOL 0272-214721/2
CROYDON 01-680-1852
EDINBURGH 031-225-8337/8854
LIVERPOOL 051-708-0133
LONDON
Centre Point 01-836-1327
Buckingham Palace Road 01-828-1423
Kingsway 01-242-9425
Bishopsgate 01-626-6888/9
Seacoal Lane 01-248-5313/4
MANCHESTER 061-236-2122
Deansgate 061-832-2242/9314
ST. HELENS 0744-30774
SOUTHAMPTON 0703-36492

TANDY COMPUTER DEALERS

ABERGELE 0745-826020
ASHFORD 07842-53024
BRIXHAM 0845-6565
CAMBRIDGE 0223-65334
CORK 22954
DUBLIN 0001-730698
0001-776270
LONDON 01-388-5271
RADSTOCK (Nr. Bath)
0761-32570
WINDSOR 075-35-58072

Send For Your FREE 1982 TRS-80™ Computer Catalogue



Tandy Corporation (UK Branch),
Tameway Tower, Bridge Street,
Walsall, West Midlands. WS1 1LA

Name _____

Address _____

_____

Postcode _____ Tel. No. _____ PCW1

MICROWRITER Revisited

Peter Rodwell re-assesses a unique text processing device.

Nearly two years have passed since I first reviewed the Microwriter, the device which enables you to 'type' single-handedly, to store your text either in the machine's own memory or onto tape and to print it out onto paper.

Since I first encountered it, the Microwriter has undergone considerable development, and the company which makes and markets it — Microwriter Ltd — has revised its marketing strategy and is now ready to hit the world with what is now not only a unique text-handling device but a very sophisticated one at that.

But first, for those of you who may never have heard of the Microwriter before, or who missed my original review, here's a brief outline of what the Microwriter is, what it does and how it is used.

The Microwriter was invented by expatriate American Cy Endfield. He hit on the idea when, after buying one of the early pocket calculators, he was trying to figure out a way of incorporating a word game into a hand — held electronic device.



The problem, he felt, was that using one key for each letter would require a very fiddly keyboard. The alternative, a chord keyboard with one key for each finger, had the drawback that learning the key combinations appeared impossible. Cy Endfield's big breakthrough was in devising a particularly clever and easy system for learning the key combinations by using the shapes of the letters themselves as mnemonics — each letter is formed by pressing the keys which correspond to the main features of its outline.

Exactly how this works can best be seen from Figure 1, taken from one of the handy reference cards supplied with the

Microwriter. When I first tried my hand at Microwriting, it took me about half an hour to learn the basic alphabet and about three weeks to learn all the other symbols and to build up speed to the point at which it was quicker and more convenient to Microwrite than to use a pen or a typewriter. This appears to be an average learning curve; the record for learning the key combinations is 1½ minutes!

Probably the most difficult part of learning to use the Microwriter is persuading yourself to try it in the first place, for the sight of a skilled user tapping away can be quite awe-inspiring: it just looks impossible. Once you get over that hurdle and actually try it out for a few minutes, it quickly becomes obvious that there's really nothing to it. The whole system is so well thought out that there's really no excuse for failing to learn!

Given that it's easy to learn, what do you do with it once you've mastered it? Well, the Microwriter is very nearly a portable word processor. 'Very nearly' because, although it allows you to enter, sort, retrieve and print out text it lacks facilities such as searching and replacing which are found on real word processors. But in its relaunched form, the Microwriter is an extremely powerful and versatile device; as you'll see, describing it as 'very nearly' a word processor hardly does it justice. Its great advantage over conventional word processors is, of course, its portability: I'm Microwriting this review while sitting outside in the sun, for example!

Hardware

The most immediately obvious physical change to the Microwriter is its case, which has changed from white to two-tone brown and cream and appears to have been flattened slightly and generally smartened up. The old LED display has been replaced with a 14-character LCD dot matrix unit which displays both upper and lower case characters and is much nicer.

On the front edge of the device is a massive socket which allows maintenance engineers access to the system's internal signals. Next to this is the socket for the battery charger. On the back edge are sockets for the RS232, TV and cassette interfaces and the on/off switch. There are in fact three ways of turning the Microwriter off: you can use the on/off switch, in which case when you turn on again you'll find yourself at the beginning of whatever's in memory; you can execute a series of key combinations; or you can simply leave the machine alone and it will turn itself off after eight minutes. In the last two cases, turning it on again puts you straight back to the point in memory where you were when you turned off — a handy feature.

Inside the machine some considerable development work seems to have taken place. There's still an RCA 1802 CMOS processor at the heart of things and there's still 8k of CMOS RAM to hold the text — enough for about 1200 words. The RAM is permanently powered by the rechargeable battery pack, incidentally, so its contents are retained even when the unit's switched off. The batteries power the unit for about 30 hours and it can be recharged overnight or while you're still using it.

The original Microwriter had two PCBs inside; the new model has only one, yet the cassette and RS232 interfaces, previously separate units, have been fitted inside the main unit. This eliminates one of my earlier gripes — the untidy tangle of wires and the constant plug-swapping which resulting during a heavy editing, taping and printing session.

The TV interface remains a separate, external unit which you purchase as an optional extra. It has, however, become a little smaller and now uses the battery charger as its power supply instead of running from the mains.

New facilities

A whole host of new facilities has been added to the Microwriter, of which the two most exciting are on-screen writing for the video interface and the ability to communicate with other equipment such as word processors and computers.

On the earlier model it was only possible to dump text a screenful at a time to the video interface. The facility was useful for checking whether you had formatted the text correctly





Fig 1 Part of the alphabet learning card (copyright Microwriter Ltd)



The video interface has outputs for TV and monitor.

MICROWRITER Revisited

for printing but, if you hadn't, changes had to be made within the Microwriter and the text re-dumped to the screen for rechecking. Now, though, the display is fully interactive and you not only see what you write as you write it but can make on-screen formatting changes as well as editing your text. If you were using the Microwriter mainly in one location , the video interface would be well worth getting. Good though the Microwriter's display is, I for one find it far easier to write when I can see a whole block of text in one go rather than just a 14-character window.

As before, the Microwriter has insert and delete commands and you can read through your text either forwards or backwards — the window has two scrolling speeds, the higher of which jumps along a word at a time fairly quickly but still readably. Text in memory can be split into separate documents and there are commands to jump to the start or end of each doument or to the start or end of memory. Likewise, the cassette handling works on documents rather than on total memory contents, so it's possible to do crude block moves by placing document markers around the block, taping it, erasing it, moving to the point in text to which you want to move the block, entering insert mode and playing in the tape. I found the tape interface easy to use and had absolutely no trouble at all with it. A very neat feature when reading tapes is an indicator on the display to show when the tape recorder's volume is at the optimum setting — with my Pearlcorder I found that tapes loaded perfectly over a very wide volume range, from full volume right down almost to the minimum setting.

Once you've written your text, you'll want to print it out somehow. The Microwriter has a standard RS232 interface with programmable baud rates from 110 to 4800 and adjustable framing (stop bits, parity, etc) so it will interface to any standard printer. Among the many new facilities added to the machine are what are called 'menu items' for setting and altering various system parameters, including all those for the RS232 interface. Once you've set the machine up to work with your printer, the settings are retained even when you switch off so you can then forget about them.

When the Microwriter prints text, it automatically prints a line feed and carriage return at the last space before the 60th character on each line; unfortunately it doesn't right-justify text. The line length can be altered with a menu setting or the facility can even be suppressed entirely. Other menu items allow you to preprogram certain key combinations to produce a string of up to three ASCII characters in a text stream — useful for sending instructions to a printer for turning bold printing or underlining on or off.

The menu settings allow you to configure the Microwriter as a terminal for use with a word processor or computer. I had no trouble at all in hooking up to my Cromemco for use with Cromemco's so-called 'word processing' package and in fact I now tend to use the Microwriter as a dumb input terminal in preference to the normal, typewriter-style VDU, although the latter is still used to display the computer's output, a 14-character display being pretty useless for this purpose.

There are two points worth noting when using the Microwriter as a terminal. Firstly, it's necessary to suppress the auto CR-LF facility as this plays havoc with a computer; and secondly, while you're using it as an input terminal, everything you type not only gets sent to the computer but is also put into the Microwriter's own memory so that, in a long session, you have to stop occasionally to clear it out. A memory full early warning is given on the display when there's only space left for 256 more characters.

Documentation

As well as upgrading the machine, Microwriter Ltd has revamped its documentation. Each Microwriter now comes with two booklets, a beginner's guide and a more advanced book, plus a series of very useful reference cards which you can keep in the machine's carrying case.

The beginner's guide firstly teaches you the alphabet key combinations and then takes you step by step through the remaining widely-used functions, including using the video interface, printing out text and using the cassette interface. Generally, the documentation is well-written, concise and clear and manages to teach without condescending — not always the case in the micro world. The beginner's book is enlivened with small cartoons featuring a cube-shaped entity reminiscent of the Tate and Lyle symbol.

Things are rather more serious in the second book, which provides full details of all the remaining facilities, including detailed descriptions of all the menu settings, what they do, their default values and how to change them, with plenty of information on interfacing the device to other machines. As the Microwriter can transmit the complete ASCII character set, including control codes, an ASCII table is thoughtfully provided.

The reference cards include the recommended learning sequence for alphanumerics plus a complete list of all the menu settings, with spaces for you to write down the values of any you change.

Prices

In my first review of the Microwriter, I incurred Cy Endfield's displeasure by stating that I thought the machine was overpriced. What I neglected to say in the review was that this was not solely my opinion but the result of showing the machine to some 30 or 40 people. All were very impressed until they heard that it cost £500, at which point they tended to wince or laugh and change the subject.

The basic Microwriter, including carrying case, cassette lead, charger and documentation, and with six months' warranty, is now £485 plus VAT. In fact the price has dropped in real terms, for not only has inflation taken its toll in the last two years but the RS232 and cassette interfaces are included both in the machine and in its price (they were optional extras before). And, of course, a very large number of extra features have been added to the machine. Still, though, I have to report winces from those to whom I have shown the device, a reasonable cross-section of potential Microwriter users.

Whether you would be prepared to pay this money for such a device is, obviously, a decision only you can make. Personally, I find it invaluable — not only because it allows me to write outside in the sun but because I find it quicker and easier to use than a conventional qwerty keyboard. And I suspect that this is the case with a great many Microwriter users: the qwerty keyboard can be an intimidating thing, especially for the businessman who has neither the time nor the inclination to learn touch-typing. The Microwriter has certainly gained favour not only in all sorts of business environments but in government departments, too: the Home Office has long used them, particularly for the preparation of documents too hush-hush to be given to a typist.

The Microwriter's six-month guarantee can be extended by a further 12 months for £28.75 and a similar scheme, costing £19.55, applies to the TV interface. Additionally, for £28.75 a year, Microwriter offers a loan scheme: if your Microwriter breaks down, you can take it to the nearest Microwriter centre (a network of which is being established across the country) and borrow another one while yours is being repaired.

Conclusion

The new Microwriter is a very attractive and versatile device which in a single package provides an easy-to-learn-and-use alternative to the qwerty keyboard plus a versatile text processor and a communicating terminal which can easily be interfaced to a computer or word processor.

I feel that the £485 + VAT purchase price may still act as a deterrent to many people — you've got to be able to justify that sort of expenditure in most businesses and the Microwriter isn't something which will immediately impress the accountant, even though I'm certain many people would find it a distinct productivity aid. Likewise, it would be expensive to equip a whole office with them, particularly if they were being used purely as input terminals; there's probably a market for a cheaper, 'cut-down' version which can act only as a dumb input device.

I must admit, though, that my main reason for wanting the device's price dropped is to encourage its wider use. I think it's a super device, embodying some really inspired thinking, which deserves to succeed.

Ian Sinclair



After a month or two of playing with his Trusty 80 Ian Sinclair has a few more comments to make.

After several months of intense 'playing around' with my TRS-80 I have managed to untangle some of the problems that were encountered when I was writing my review, see November's Computing Today. In an attempt to assist anyone who has bought themselves one here are a few hints.

Cassette Handling

In the review I said one or two unfair things about the cassette file handling capabilities of the machine, or so it appears on further scrutiny. I've learned that it pays to test out any program which uses cassette files, there is a very simple scheme for doing this.

In place of every PRINT#–1 statement put PRINT : STOP, and similarly for each INPUT#–1 you substitute STOP : In this way you will see on the screen exactly what is going to be recorded and you can jump to the step after INPUT#–1 to see what happened on replay. This tends to save a lot of trouble with cassette testing and you can see at a glance if you are using too many bytes, or if the replay procedure is wrong.

Television Or Monitor

After Mr Heller's comments I played around with the TV circuits and concluded that I didn't really need a monitor after all. If you find that the lettering on your screen looks a bit disjointed, in particular double ee's, it's a fair bet that there's too much bias on the modulator. Open up the modulator box and you find a standard ASTEC device, see Fig.1, with the video input taken through a 100 uF capacitor and biassed by two resistors. Try connecting a 10 K pot between the video input and the earthed case, or better still connect a 5 K pot in place of the two resistors. Starting with the voltage at around 2V5 gently twiddle the pot until you have the lettering as you like it (to coin a cliche). The improvement can be quite dramatic and is well worth a couple of minutes of your time. It should also be possible to improve the graphics by doing a DC restore at this point.

Problem Loads

Because BASIC is so much simpler to operate than machine code it occurred to me that system tapes could be entered as part of a BASIC program, the KBFIX being a prime example. Keyboard bounce has always been a problem on the TRS-80, not of major proportions but if you get LLIST instead of LIST and like me you haven't got a lineprinter yet (at £1200 a time who has?) the whole thing hangs up until you use RESET. The manual, and other sources, tell you to remove the offending keytop and clean the contacts but on my TRS-80 they are NOT removeable so where do we go from here.

The answer is a machine code subroutine which slows down the rate at which the keys are scanned, this means that the key isn't read until after the bounce. There is a routine supplied with the machine but it simply wouldn't load, not at any setting, and a quick listen convinced me that it wasn't even made for the same machine, the version on the other side did load fortunately. As I didn't feel like going through all this bother each time I listed the machine code and wrote a short BASIC program which POKE's the values into the correct addresses at the top of memory. Now, whenever I want to enter a long BASIC program I start with this tape which lies in lines 1 to 5 along with entry procedures. I run this and then start entering the new program from lines 10 onwards. Once the new program is entered, I can then delete

TRS8BIT - VOLUME 19 ISSUE 03 - SEPTEMBER 2025 - PAGE 68 - TRS8BIT

lines 1 to 5, or if the new program is one which requires a lot of keyboard entries I can keep the first five lines in place to make sure that there is no bouncing on new entries. It is much more satisfactory than using a SYSTEM tape. At the moment, I'm developing a method (which has worked in its first trials) of placing bytes from a system tape directly into a BASIC program without having to note down the values and enter them.

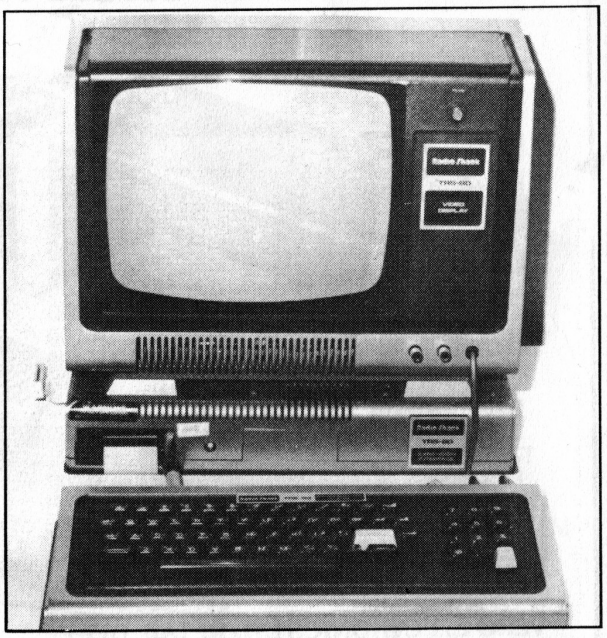
Printer Or Disc

After the first few weeks with the machine, I was convinced that the first major addition would be a disc system. A bit more experience has changed my mind. Useful as disc operation might be, a printer now seems a much more useful addition, because any program needs referring to and unless you're going to spend hours at the keyboard, the referring *has* to be done on a printout. It is decidedly infuriating to spend a long time sorting out a program and then having to hand-copy it from the screen.

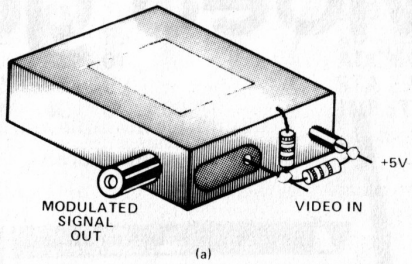
The snag at the moment is the silly prices of printers. A printer is a box which is mainly empty and certainly doesn't contain so much expensive equipment as a £150 Japanese electric typewriter. It looks very much like the pocket calculator story again, and production levels must surely be getting to the stage where prices will drop. I'm just not going to be tempted by printers which give only 40 columns on 'peculiar' paper, what's needed is at least 80 columns on plain paper – *not* sprocketed, since the price of these holes is just ridiculous. Any genius who can convert a £100 electric to use bog roll and interface it to the TRS-80 should earn a fortune!

Self Instruction

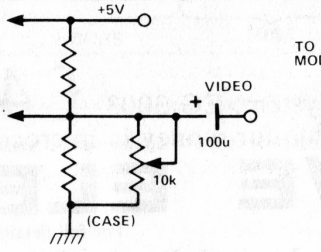
All-in-all, then, I've had an instructive time – and I'm still learning fast. I've seen some good software (from A.J. Harding) and I now have most of the programs I need for keeping track of my books and my accounts. There's one thing I think should be stressed to all prospective computer owners – time. It takes a long time to enter a program from the keyboard, it takes longer to get it running the way you want it, and longer still to tidy up the printing. The factor which, more than anything else, distinguishes a 'professional' program from an amateur one is foolproof operation. If each act on your part is prompted by clear instructions on the screen, if each mistake results in a rescue operation, rather than a blank screen, if answers are printed legibly with explanations of what they represent, then you have a reasonably professional program. Don't kid yourself that you can get by with much less. The acid test is to go back to a program you last used over a month ago. Can you run it right away without reading a listing to see what it's all about? If you can't or if it's not immediately obvious what you should do, then it needs a lot of work – and it all takes time. This time factor is one which must be explained to any prospective user of a microcomputer. A business user expects to be able to switch on, load a program and start operations. We may find it more interesting to develop our own programs, but there's no need to settle for less once we have them running. From that point of view, the excelling editing facilities of the TRS-80 have been worth their weight in gold. From the adverts I read, it appears that the unfortunate buyers of a system costing twice as much have to spend £80 odd on a software package which lets their machine have some of the features which are completely standard on the TRS-80. Now that more than one supplier is selling TRS-80 at £399 + VAT, it's a better buy than ever.



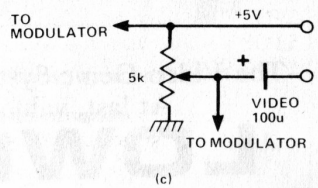
The debated TRS 80 monitor. If you modify the modulator circuit as shown below you can use your telly instead and save yourself a few pennies.



(a)



(b)



(c)

TRS8BIT - VOLUME 19 ISSUE 03 - SEPTEMBER 2025 - PAGE 69 - TRS8BIT

Elegance and power in a mathematical language.

APL

by Phelps Gates



RATTYWARE

Software you can rely on.

Now, a high-level, scientific programming language that doesn't cost $200 or $300 for the home computer. This language is perfect for the mathematician, scientist, engineer, or anyone who just wants to learn a new language. The power of this language is in its strong mathematical operations, especially with regard to matrices and vectors. Programs requiring matrix multiplication or other matrix problem solving that would require hours of programming time in BASIC are solved quickly and with minimal effort in APL. Not only is math made easy, but upon gaining proficiency in APL programming various string manipulations become child's play.

To aid in learning APL, lessons are included on the disk. Starting from the basics, you are brought step by step through the various programming techniques involved with APL. These lessons act as a tutor in a "learning by doing" atmosphere which will have you "talking APL" in no time. Also available is the book, **APL: An Interactive Approach,** which reinforces many of the examples given in the lessons. The book also provides additional insight into APL programming.

LIMITATIONS

Due to the absence of the special APL character set on the TRS—80, APL-80 uses shifted letters to represent the various APL characters. These shifted letters are identified on the screen by a graphics block before each shifted letter. If you have a modified TRS-80 (Electric Pencil Modification), a lower case driver is included to display the shifted letters on the screen.

In addition to the keyboard limitations, there are several other limitations. Lamination, domino, and matrix inverse are not implemented but can be derived with user-defined functions.

Multiple specifications must be split into two statements unless the left-hand assignment is to a quad. This also applies to implied multiple specifications.

Reduction and reshape (p) are not permitted for empty arguments; the argument of add/drop may not be scalar; empty indices are not permitted.

A quad (q) can't be typed in response to a quad (nor can the name of a function which itself gets input from a quad). Quote-quad (m) is permitted.

No more than 32 user functions can be defined in a single workspace and a function may not contain more than 255 lines.

A comment (c) must occupy a separate line: a comment can't follow a function statement on the same line.

In the tape version, arrays are limited to five (5) dimensions.

FEATURES

APL-80 on disk contains the following features:)SAVE and)LOAD workspace on disk;)COPY other workspaces into current ones; Return to DOS for directory or commands without losing your workspace; Send output to lineprinter; Five workspaces of lessons included; Sequential and random files; 15 digit precision; Monadic and dyadic transposition; Easy editing within FUNCTION lines; Latent expression (FUNCTION can "come up running" when loaded); Tracing of function execution; Real-time clock; User-control of random link; Workspace is 25587 bytes (in 48K machine); Arrays may have up to 63 dimensions.

COMMANDS I APL-80

APL-80 supports the following commands: Absolute value, add, and, assign, branch, catenate, ceiling, chr$/asc, circular, combinatorial, comment, compress, deal, decode, divide, drop, encode, equal, expand, exponential, factorial, floor, format, grade down, grade up, greater, greater/equal, index generator, indexing, index of, inner product, label, less, less/equal, logarithm, maximum, member, minimum, multiply, nand, negate, nor, not, not equal, or, outer product, peek, poke, quad, quote quad, random, ravel, reciprocal, reduction, reshape, residue, reverse, rotate, scan, shape, sign, system, subtract, take, transposition.

SPECIFICATIONS

Minimum system requirements: 32K disk system (48K recommended) Includes APL-80, Five workshapes of lessons, instruction manual.

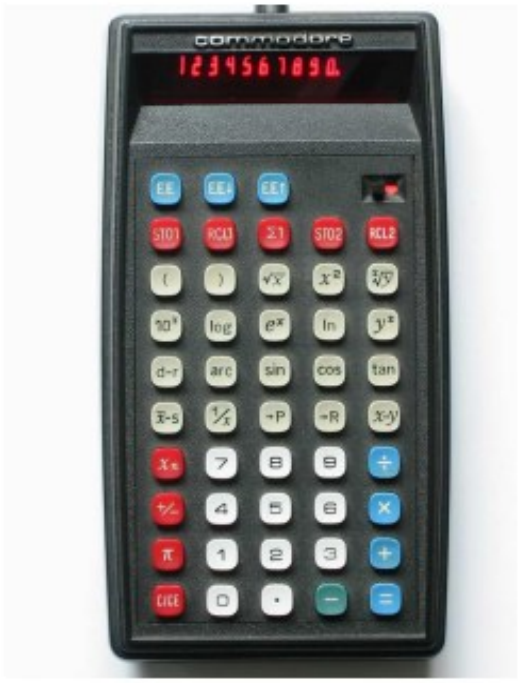
Reduced feature: 16K Level II tape version, no lessons.
Transpositions, format, and inner product not implemented. Reduced domain for some functions. 6 digit accuracy.

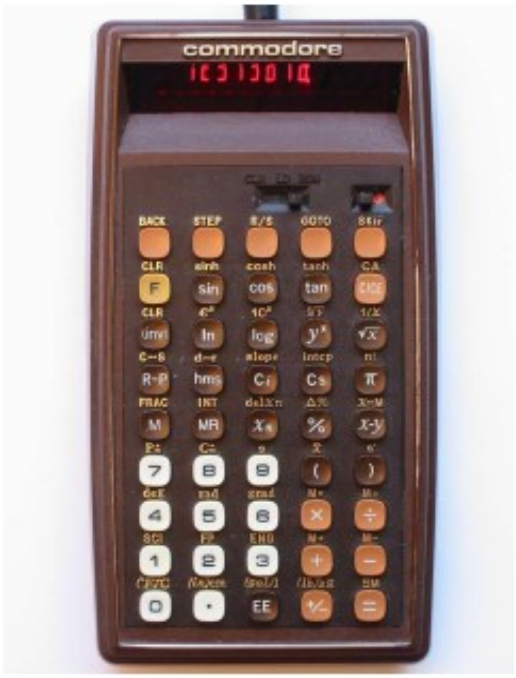
MICROCOMPUTER
APPLICATIONS
42A CHURCH STREET
CAVERSHAM
READING RG4 8AU
ENGLAND



Programming Languages for the Model 1
Vince Otten

In the mid-1970s, I could finally afford to retire my slide rule when Commodore began producing inexpensive calculators with dozens of fancy functions. Soon after I got a device like the SR-4148, I saw an ad for something even more exotic in *Popular Science* magazine: the PR100, which had enough storage to let you save and run up to 72 calculator steps!





The programming looked something like this:

```
M
1
F
FRAC
X
1
EE
4
X
1
F
INT
=
```

One line per key press: this was known as "keystroke programming". I tortured this calculator, trying to squeeze every ounce of programming that I could out of it. Soon I discovered *Byte* magazine, and learned about things like "subroutines" and "alphanumeric displays"—and so I wanted more. By 1978, my father and I saved up enough to buy a 4K TRS-80 with Level II BASIC. What a giant leap forward! Now, one line of code could do so much more. BASIC was amazing—it could not only work with individual numbers, but with character strings and graphics. I'm still learning about BASIC's skills today—my recent TRS8bit articles on "Software Tools in BASIC" are an example.

But *Byte* wouldn't leave me alone. In August of 1977, the magazine began a 10-year tradition of introducing a new programming language, and the premier choice was APL. This language was so strange, so unreadable, but apparently so powerful that it could calculate the average of all the elements in an array of 1000 numbers, for example, with this single-line function:

```
      ∇R←AVG X
[1]   R←(+/X)÷⍴X∇
```

whereas in BASIC, I had to write a subroutine something like

```
1000 S=0
1010 FOR I=1 TO 1000
1020       S=S+X(I)
1030 NEXT
1040 R=S/1000
```

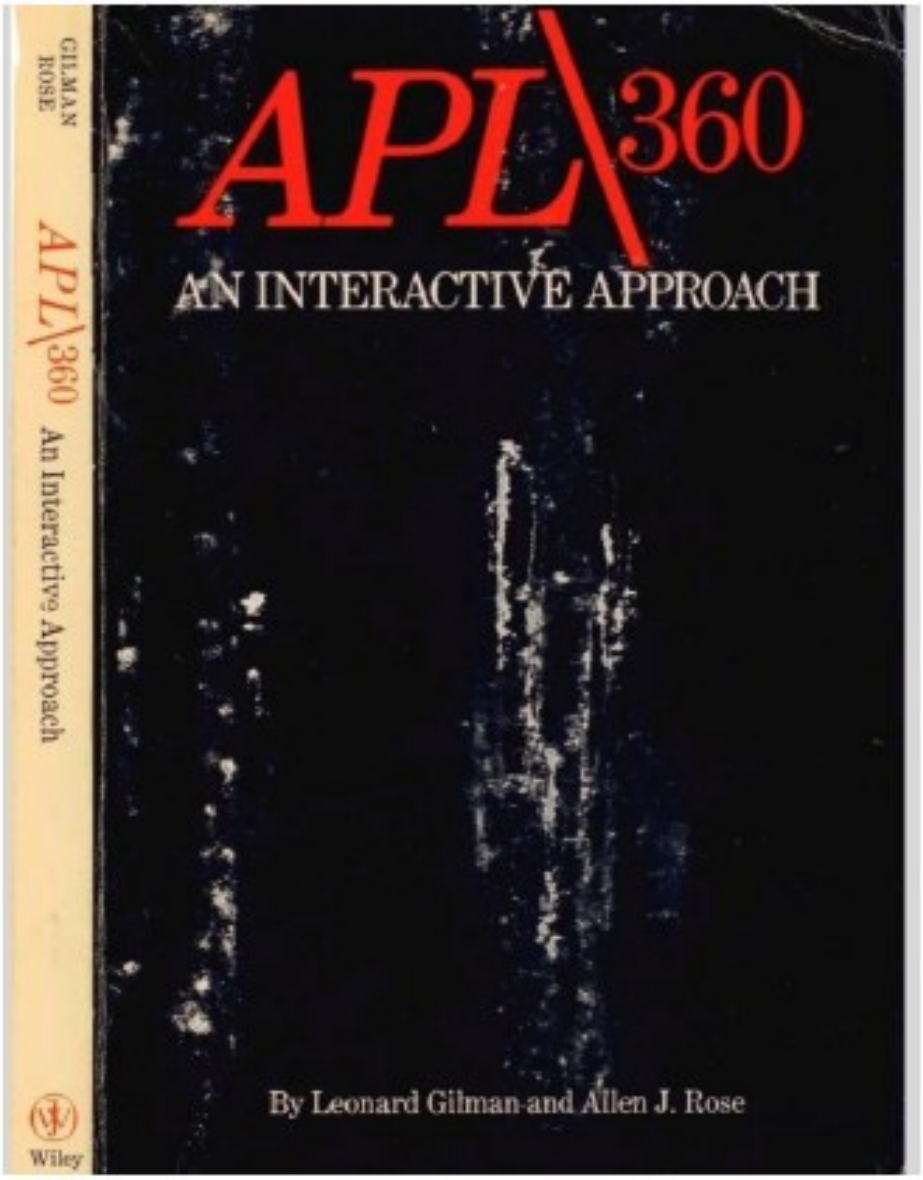
Of course, the BASIC code was hard-wired for a size of 1000 elements, and it left the global variables S, I, and R "lying around", whereas the APL function would work on any size of its array argument "*X*", so if we assigned 5 numbers to the APL variable NUMS, we could get the average this way:

```
NUMS←  1   3.1415  17  1009   5
AVG NUMS
  207.0283
```

or we could just use it with a list of numbers:

```
AVG 7 22 109 -20
  29.5
```

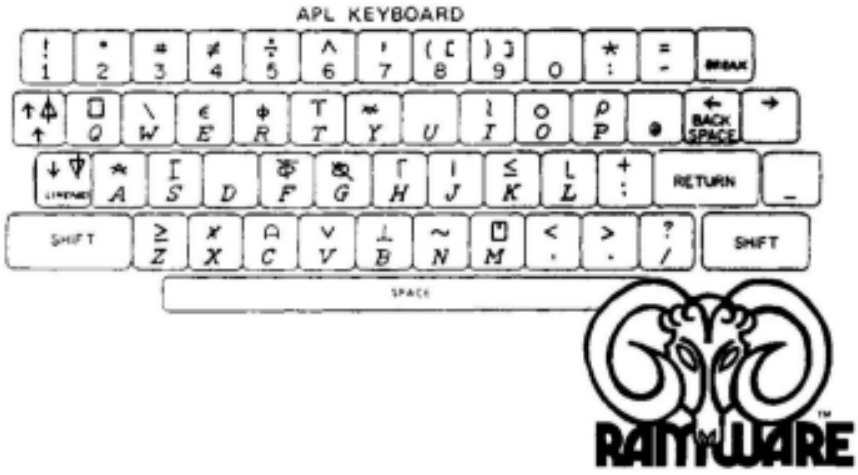
and all this was done interactively, without a "RUN" command! Soon, I'd pilfered a used copy of the original APL textbook from the local university-town used bookstore (probably for less than $5), and tried to understand its graduated examples. This was the first textbook on APL, and I still recommend it for any newcomer to the language.



What a demonstration of the TRS-80's amazing market share within the first few years of its release: by the end of 1980, there was an actual implementation of APL for our lowly Z80 machine!

APL80

by Phelps Gates



I know very little about Phelps Gates, other than this:

- He also wrote a remarkable Pascal compiler for the Model I
- He implemented the *APL*Plus* interpreter for the well-known STSC APL company, where he was Senior Systems Programmer from 1981 to 1994

And he created this remarkably complete and fast version of a truly high-level language for a rather slow microcomputer, 45 years ago! Today once again I'm working my way through the *APL\360* textbook, and I never feel like I'm twiddling my thumbs waiting for APL80 to respond to my entries—even while it's performing all these operations with 15 digit (Double Precision) accuracy.

Okay, but here's the big question for those who know anything about APL: "What about the keyboard and all the special symbols?" You see a standard APL keyboard on the APL80 title page above. How—especially with such low-resolution graphics as the Model I supported—could you possibly depict this?

```
     ∇R←AVG X
[1]  R←(+/X)÷⍴X∇
```

Here's what it looks like on the TRS-80's screen:

```
A P L - 8 0
VERSION 3.0
BY PHELPS GATES

(C) 1980, THE SOFTWARE EXCHANGE
MILFORD NH 03055

WS CLEAR

)DEF R← AVG X
 1:    R← (+/X) % ▯P X
 2:
NUMS← 1 3.1415 17 1009 5
AVG NUMS
 207.0283
```

The ÷ "divided by" symbol in APL is replaced by the TRS-80's "%", and the ρ "length" symbol is replaced by the graphics square and a capital "P" (which kind of looks like a Greek "rho" ρ). The APL80 manual has a whole list of replacements.







That screen shot shows off another uncommon feature of APL: it's a "REPL". Only a few programming languages are implemented as a "Read-Evaluate-Print-Loop" (REPL): APL, Lisp, and the UNIX/Linux "shell" are the only three I can think of, as well as one other that I suspect

every reader has made use of: the good old *calculator*. You type something in like "2 x 5" ("Read"), the processor works with it ("Evaluate"), it prints out the result "10" ("Print"), and "Loops" back for more input. This is very useful, because you can mess around with data and your function programming experiments, and the interpreter gives you immediate feedback.
"But isn't BASIC like that?" Not really. It's more like a "REL": you have to explicitly tell it to PRINT something, and in fact it doesn't even automatically "return a result" like APL does.
Here's what I mean. Suppose we have an arbitrary list of numbers, like

```
1  17  -3 25.976  -99
```

and we want to multiply each number by 3.35. In APL, we just write

```
3.35  x  1  17  -3 25.976  -99
```

and it immediately spits out

```
3.35  56.95  -10.05  87.0196  -331.65
```

whereas in BASIC we'd have to write

```
PRINT 3.35 * 1; 3.35 * 17; 3.35 * -3; 3.35 * 25.976; 3.35 * -99
```

But APL also lets us assign the *result* of our expression to a variable:

```
TIMES. THREE ← 3.35  x  1  17  -3 25.976  -99
```

In BASIC, you pretty much have to write a program and then RUN it:

```
10  SZ%= 5: REM SIZE OF ARRAY
20  DIM T3#(SZ%): REM "TIMES.THREE" DOUBLE-PRECISION ARRAY
30 FOR I%= 1 TO SZ%
40      READ DP#
50      T3#= 3.35 * DP#
60 NEXT I%
70 DATA 1, 17, -3, 25.976, -99
```

Of course, the BASIC code is hardwired to a particular array size. Not so in APL.

```
TIMES. THREE ← 3.35  x  1  2  3  4  99  100  1000  5555  3.14  -22.357  0  111
TIMES. THREE ← 3.35  x  17
TIMES. THREE ← 3  3.14  3.0009 x  1  2  3
```

"TIMES.THREE" doesn't complain about what's assigned to it. In the last example above, it becomes a 3-element array consisting of

```
3 x 1
3.14 x 2
3.0009 x 3
```

so if we ask for the value of TIMES.THREE, we'll see

```
3  6.28  9.0027
```

APL's functions, like "x", are consistent, whether they're applied to a single element (APL calls this a "scalar")

```
3 x 1
```

a list of elements (called a "vector"), as in any of these:

```
3 x 1 2 3
3 4 5 6 x 7 8 9 10
11 12 13 14 x 15
```

or even an APL "array" (what we're used to calling a "multidimensional array"). In this example, we "build" an array "A" of 3 rows by 4 columns, filled with the first 12 numbers. We then type its name "A" to view its contents. We multiply it by 2 and assign that result to array "B", and finally view the contents of "B".

```
A ← 3 4 ⍴ 1 2 3 4 5 6 7 8 9 10 11 12
A
      1 2 3 4
      5 6 7 8
      9 10 11 12
B ← A x 2
B
      2 4 6 8
      10 12 14 16
      18 20 22 24
```

What blew my mind back in the 1980s was the remarkable *consistency* of APL. You didn't need a separate procedure or operator to work with different-sized objects like scalars, vectors, or

arrays, like you would in Forth or C. You didn't need to allocate memory every time you worked with a new object. If you don't need "A" any more, you can

```
)ERASE A
```

and if you're experimenting with numbers like

```
10 100 1000 ÷ 3 4 5
```

the two vectors

```
10 100 1000
3 4 5
```

automatically "disappear" after use. Another feature that APL shares with the other "REPL" language, Lisp, is that functions typically "return" values—scalars, vectors, or arrays—that can be used by other functions or assigned to variables. So above, when we assign B the value of A × 2, the "×" function returns the array value computed by "A × 2", and then that's assigned to B. If there's no assignment, APL (like Lisp) simply *prints* the return value of the last function.

All this extra work, though, in a language that's interpreted—it must be pretty *slow*, right? Especially on a TRS-80? Let's look at an interesting example of APL code, and compare it with Level II BASIC. Here's one way to approximate the value of π—you know, "3.1415926..."

$$PI=4\times\left(\frac{1}{1}-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\frac{1}{9}-\ldots+\ldots\right)$$

This is how we build up a function to calculate this in APL. First, we need to be able to create that sequence of "1/3", "1/5", "1/9", etc. APL has a function called the "index generator", ⍳, which is used to generate a list of numbers.

```
⍳10
    1 2 3 4 5 6 7 8 9 10
```

Through a bit of experimentation, I've discovered that APL80 can generate a vector of up to 3190 elements (remember, these are Double-Precision 8-byte numbers). So we could generate

all those 1..3..5..7..9.. divisors by multiplying ⍳ 3190 by 2 and then subtracting 1. Here's one way to do that in APL:

```
¯1 + 2 x ⍳ 3190
      1 3 5 7 9 ...
```

(APL uses a separate character for "negation"—that is, "negative 1" is "¯1".) Here's a bit of a "quirk" in APL: unlike BASIC and many other languages, it recognizes no "operator precedence". It doesn't multiply before it adds, for example. All operations are performed from *right to left*—not left to right! So in our line of code above, we first generate the first 3190 numbers, then multiply that by 2, then "subtract 1" by adding -1.
Now we want the "inverse", that is, "1 divided by" each of those odd numbers. For that, we use the APL ÷ function, which when used on its own, gives us the inverse.

```
÷ ¯1 + 2 x ⍳ 3190
      1 .333333333333333 .2 .1428571428571429 .111111111111111 ...
```

What now? Let's look at the original formula:

$$PI = 4 \times \left(\frac{1}{1} - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \ldots + \ldots \right)$$

We have to *subtract* the .33333... from the 1, then *add* the .2, then *subtract* the .142857..., then *add* the .111111..., and so on. APL has a function which I believe was unique for its time: the "reduction" operator. Here's how that works. If we had a vector of 1 2 3 4 5 6 7 8 9 10, and we wanted to add up all its elements, we'd place the "+" operator between each.

```
1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10
      55
```

But when you have 3190 numbers, that becomes a chore. The "/" reduction operator to the rescue!

```
+/1 2 3 4 5 6 7 8 9 10
      55
```

That "/" tells APL to place the function on its left between every element on its right. Now here's how "/" can work for our approximation formula. Try this:

```
-/1 2 3 4
      -2
```

Huh? Remember: APL evaluates from right to left, so "-/1 2 3 4" is the equivalent of

```
1 – 2 – 3 – 4
```

or

```
1 – (2 – ( 3 – 4 ))
```

3-4 is -1. 2 - -1 is "2 + 1" or 3. 1 – 3 is -2! In other words, "-/" is like placing a "-" between the first two numbers, then "+" between the 2nd and the 3rd, then "-" between the 3rd and the 4th, and so on, so our example is equivalent to

```
1 – (2 + ( 3  – 4 ))
```

This means that we can use "-/" to sum up all those inverse numbers just like the whole -..+..-..+ sequence in the formula.

```
-/ ÷ ¯1 + 2 x ⍳ 3190
      .785319793494176
```

What's left to do? Oh yeah—multiply that by 4.

```
4 x -/ ÷ ¯1 + 2 x ⍳ 3190
      3.141279397367
```

This takes about 3:11 to calculate in APL on the trs80gp emulator running as a 48K Model I (ROM version 2). How long for an equivalent BASIC program? To be fair, our BASIC code should do everything that APL did: create an Double-Precision array A() filled with the first 3190 numbers, multiply each by 2, subtract 1 from each element, then fill each one with its inverse. Then sum a Double-Precision variable T by alternatively subtracting and adding each array element to it. The program below shows a run time of 5:47.

```
00/00/00 00:00:43             00/00/00 00:05:30
 3.141279173973749
5 PRINTTIME$,
10 DEFDBLA-Z:S%=3190:DIMA(S%):FORI%=1TOS%:A(I%)=CDBL(I%):NEXT
20 FORI%=1TOS%:A(I%)=2*A(I%):NEXT
30 FORI%=1TOS%:A(I%)=A(I%)-1:NEXT
40 FORI%=1TOS%:A(I%)=1/A(I%):NEXT
50 F%=0:T=A(1)
100 FORI%=2TOS%
110 IFF%THENT=T+A(I%)ELSET=T-A(I%)
115 F%=NOTF%
120 NEXT
200 T=T*4:PRINTTIME$,T:LIST-300
300 END
READY
>_
```

Does this mean BASIC is slower than APL at calculating π? Of course not. BASIC doesn't *have* to fill a 3000+ Double Precision element array and keep looping over it like our example code here in order to approximate pi, but what this program does demonstrate is how fast APL80 is at doing these tasks! (Interestingly, I adapted the BASIC code for Microsoft Fortran-80, and *it* ran in approximately 3:10—no faster than APL80!)

Speaking of demonstrations, APL80 comes with 5 tutorial files, LESSON1/APL..LESSON5/APL. They are APL "workspaces", that is, an APL environment complete with variables and functions. In each file, one function has been tagged as "AUTO", so that it executes as soon as you enter

```
)LOAD LESSON1/APL
```

It looks like this:

```
     W E L C O M E   T O   A P L - 8 0 !

THIS WORKSPACE INTRODUCES YOU TO SOME ELEMENTARY FEATURES
OF APL-80.  IT INCLUDES THE FOLLOWING MODULES (TO RUN ONE OF
THEM, JUST TYPE ITS NAME):

DIRECT   VECTORS   MINUS   FUNCTIONS   VARIABLES
CHAR   COMMANDS   DYADIC   MONADIC   SI   GOODBYE

FOR A LIST OF MODULES, TYPE   MENU
TO RESTART THE WORKSPACE, TYPE   GO
TO START THE FIRST MODULE, TYPE   DIRECT
```

If you type "VARIABLES", for example, the function VARIABLES will print out a new "man" screen about APL variables, along with some sample code you can try out just by typing it in. These lessons are quite thorough, and will teach you a lot about APL.
You can)SAVE,)LOAD, and even merge in, or ")COPY", the workspace, which contains all your functions and variables as a sort of "work in progress". Speaking of variables, any function can also contain local variables, something like C's "auto" variables. There's also a complete set of Random-Access File handling functions. APL has something similar to BASIC's LINE INPUT statement, called the "quad" ⎕, entered as a shift-Q in APL80. What's remarkable about the quad function is that you can enter not only numbers or strings, but even expressions like 2+2 or variable names, or a function... It's like a REPL within a REPL! You can sort vectors or arrays "up" or "down". You can "rotate" them. You can transform a multi-dimensional array into a vector...or a vector into an array. You can check if each element in an object is in a set using the ∈ operator, entered as shift-E. For example,

```
'HELLO' ∈ 'AEIOU'
      0 1 0 0 1
```

This tells us that 'H' is *not* in the set 'AEIOU' (the first 0), 'E' *is* in the set (the second element, 1), 'L' is not (0), the next 'L' is not (0), and the final 'O' is in the set (1). Similar code in Disk BASIC might look like this:

```
10  H$= "HELLO": ST$= "AEIOU"
20  FOR I= 1 TO LEN( H$)
30      IF INSTR( ST$, H$(I)) > 0 THEN
            PRINT "0 ";
        ELSE
            PRINT "1 ";
40  NEXT
```

It's just such a different way of thinking, and that's what I love about learning new programming languages. APL80 was one of the most remarkable language implementations I ever saw for my lowly TRS-80. While numerous other languages were presented as "toys", often missing features like floating-point math or entire elements of the language (like Tiny C or Tiny Pascal), APL80 was a nearly complete implementation of the environment that most contemporaries could only find on an IBM mainframe.
(A quick note about ROM versions: hats off to George Phillips for pointing this out. I couldn't get the "down arrow" (which is used for sorting or as the "drop" operator in APL) to respond properly. He identified the problem as originating with the ROM version of Model I BASIC in my machine, the trs80gp emulator running with the default latest version 3. As soon as I "downgraded" by selecting ROM version 2, APL80 worked perfectly.)

NEWS FROM FRANCE

Liberté • Égalité • Fraternité

PASCAL HOLDRY

NEW_80-GRAFIX

Details

Written by Pascal HOLDRY

Category: Graphics Cards

Publication: July 15, 2025

Views: 14

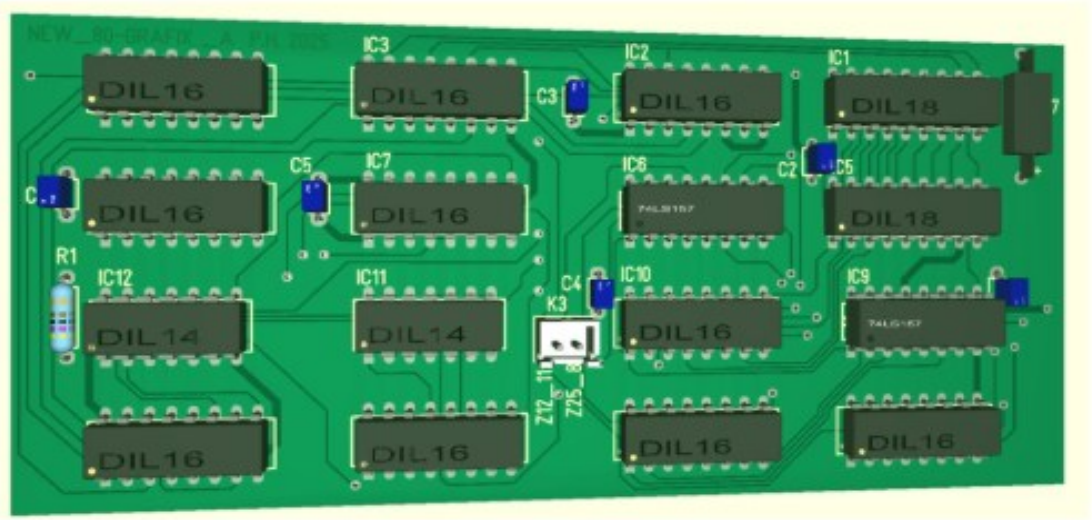
NEW_80-GRAFIX

Graphics Card for the TRS-80 Model I

Aim:

1. Provide the ability to work in graphics mode on TRS-80 Model I microcomputers
2. Being able to recover existing software
3. Keeping track of our digital heritage
4. Provide a diagram for troubleshooting and understanding how the card works.
    1. This card does not add a Graphics Processor.

Face components




Joe Kay
Director
NORTH COMPUTER WEST MUSEUM
Create Experience Learn
joe@nwcomputermuseum.org.uk
01942 582826
Information Technology facilities & training
Workshops for programming, Electronics & VR
Community classes, clubs & retro games
IT Skills for the young & old
City & Guilds Creditation
Supporting education of local people
Creating opportunities & careers
Disable friendly
Volunteer- ed training & museum operations
NORTH WEST COMPUTER MUSEUM
Create Experience Learn
www.nwcomputermuseum.org.uk
Leigh Spinners Mills, Park Lane, Leigh, WN7 2LB

X. Basque Tournament of TRS-80

The last 8th of july, tuesday, the Euskal Retro organized the X. Basque Tournament of TRS-80. We met 8 players in Bilbao, in Cafe Berria (a bar located in Santutxu), and we played TRS-80 in the second floor of the bar, as always we do.

It was an amazing afternoon, funny as always we meet and gather to play consoles. We plugged in the TRS-80 after playing ZX Spectrum and we enjoyed playing this american machine. This time we played these 3 games: Monster Maze, Canyon Climber and Demolition Derbi

Egoitz was the player who got more points after playing the 3 games mentioned but the rest of the players played well and enjoyed a lot. Egoitz practices at home, and for that reason he does have a slight advantage. For the next tournament, possibly in april of 2026 we hope to have new games like Dragon Fire or Robot Battle, I have to buy new ones and this time I must buy them because we want tom play different games.

Here you have the results:

Canyon Climber

1-Egoitz Campo 7 points (16.800)
2-Axel Pernett 6 points (7.600)
3-Daniel Barrientos 5 points (7.000)
4-Jon Proaño 4 points (4.200)
5-Joseba Zubiaurre 3 points (3.200)
6-Jesus Carballeira 2 points (3.000)
7-Mikel Torde 1 point (2.200)

Monster Maze

1-Egoitz Campo 4 points (1.180)
2-Joseba Zubiaurre 3 points (120)
3-Daniel Barrientos 2 points (30)
4-Mikel Torde 1 point (0)

Demolition Derbi

1-Egoitz Campo 4 points (1.700)
2-Jon Proaño 3 points (1.200)
3-Alvaro García 2 points (700)
4-Daniel Barrientos 1 point (600)

Classification

1-Egoitz Campo 15 points (Basque Country)

2-Daniel Barrientos 8 points (Basque Country)

3-Jon Proaño 7 points (Basque Country)

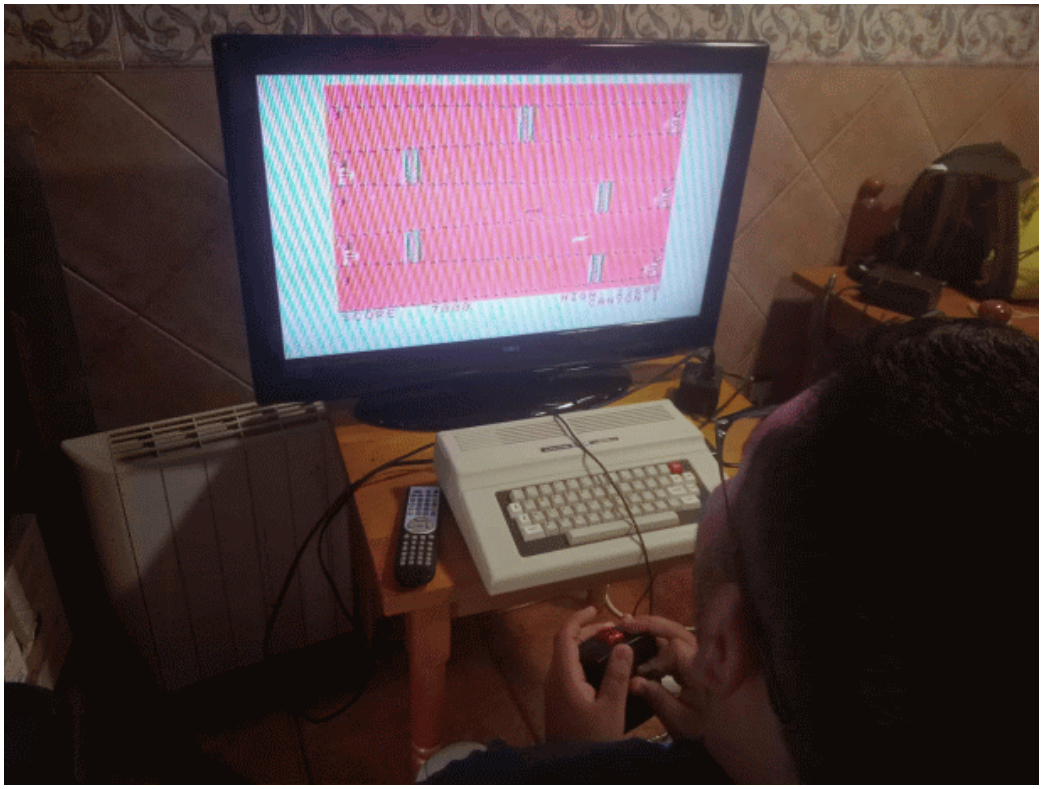
4-Joseba Zubiaurre 6 points (Basque Country)

5-Axel Pernett 6 points (Colombia)

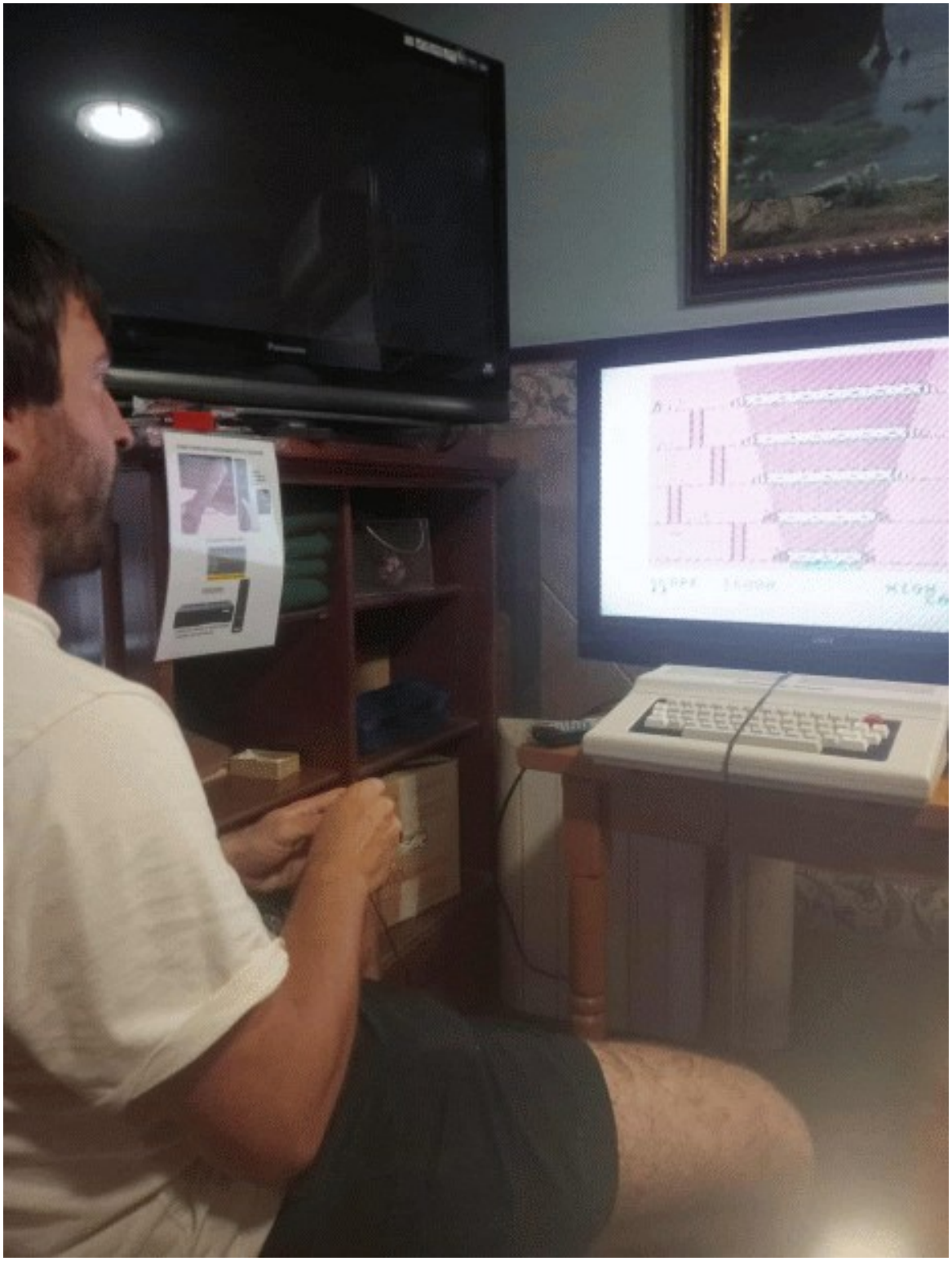
6-Mikel Torde 2 points (Basque Country)

7-Jesus Carballeira 2 points (Galicia)

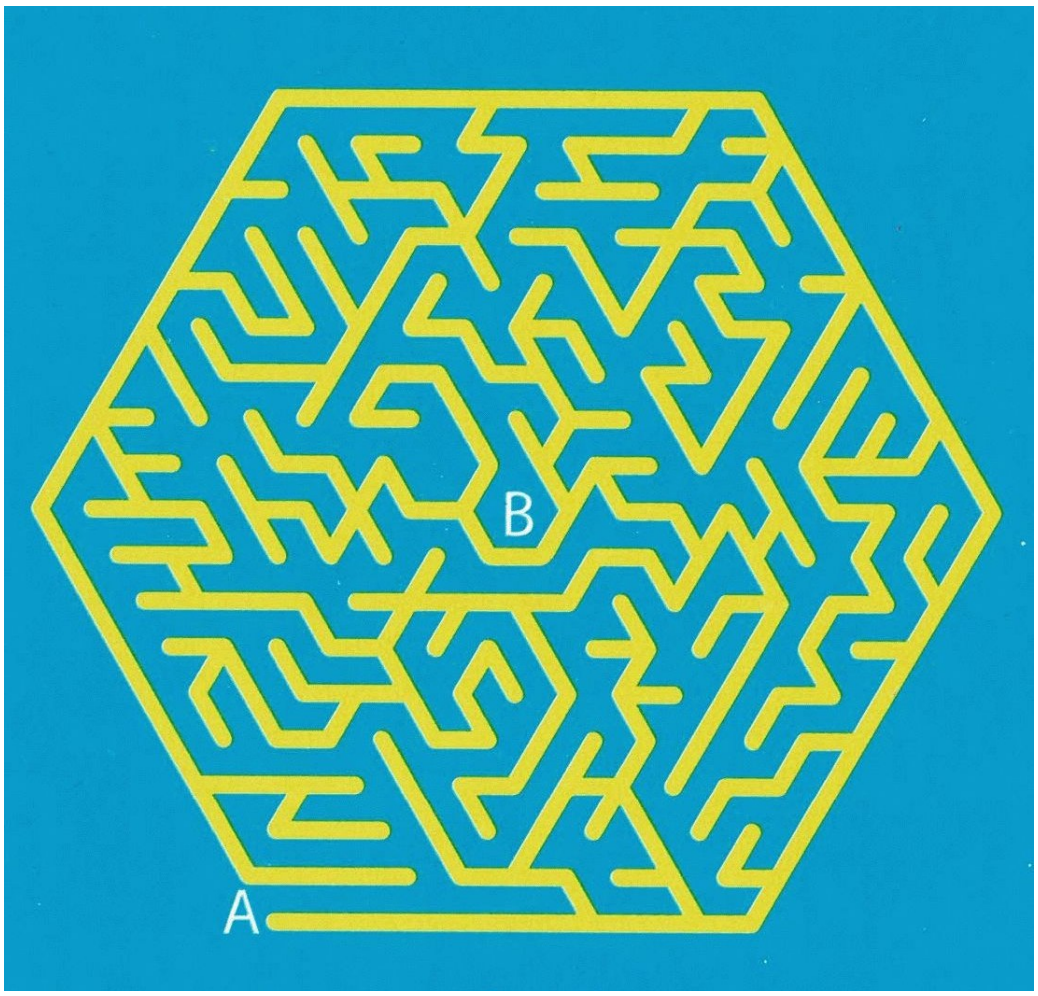
7-Alvaro García 2 points (Basque Country)







AMAZED



IF YOU HARDWARE-HACKERS ARE FINDING THOSE PCB BOARDS TRACKS ARE MERGING INTO ONE OR YOU ASSEMBLER CODERS HAVE LOST A PUSH OR A POP, GIVE YOURSELVES A BREAK.

SIT DOWN WITH A COFFEE, PUT A TIMER ON AND SEE HOW LONG IT TAKES TO GET FROM A TO B, FOLLOWING THE BLUE ROAD.
TRY FOR COMPLETING IN UNDER 8 SECONDS!

HAPPY 34TH BIRTHDAY TO THE LINUX KERNEL!

ON THIS DAY, AUGUST 25TH, WE CELEBRATE 34 YEARS SINCE LINUS TORVALDS ANNOUNCED HIS NEW TOY PROJECT. WHAT STARTED AS A SMALL, PERSONAL PROJECT HAS GROWN INTO THE POWERFUL LINUX KERNEL, THE HEART OF COUNTLESS DEVICES AROUND THE GLOBE.

ITS SUCCESS IS A GOOD REMINDER OF THE POWER OF OPEN-SOURCE COLLABORATION AND THE GLOBAL COMMUNITY OF DEVELOPERS WHO HAVE CONTRIBUTED TO ITS GROWTH. THANK YOU, LINUS, AND THANK YOU TO EVERYONE WHO HAS MADE LINUX WHAT IT IS TODAY.

From: torvalds@klaava.Helsinki.FI (Linus Benedict Torvalds)
Newsgroups: comp.os.minix
Subject: What would you like to see most in minix?
Summary: small poll for my new operating system
Message-ID:
Date: 25 Aug 91 20:57:08 GMT
Organization: University of Helsinki

Hello everybody out there using minix -

I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386(486) AT clones. This has been brewing since april, and is starting to get ready. I'd like any feedback on things people like/dislike in minix, as my OS resembles it somewhat (same physical layout of the file-system (due to practical reasons) among other things).

I've currently ported bash(1.08) and gcc(1.40), and things seem to work. This implies that I'll get something practical within a few months, and I'd like to know what features most people would want. Any suggestions are welcome, but I won't promise I'll implement them :-)

Linus (torvalds@kruuna.helsinki.fi)

PS. Yes - it's free of any minix code, and it has a multi-threaded fs. It is NOT protable (uses 386 task switching etc), and it probably never will support anything other than AT-harddisks, as that's all I have :-(.

Judging from the post, 0.01 wasn't actually out yet, but it's close. I'd guess the first version went out in the middle of September -91. I got some responses to this (most by mail, which I haven't saved), and I even got a few mails asking to be beta-testers for linux. After that just a few general answers to quesions on the net:



LEVEL 2.5 (TWOHAF) BASIC ALLOWS CASSETTE AND ACULAB FLOPPY TAPE USERS MANY OF THE TANDY DISK BASIC FEATURES, NAMELY -

"&", "DEF FN", "DEF USR", "MID$=", "INSTR", "MKD$", "MKI$", "MKS$", "CVD", "CVI", "CVS"

AVAILABLE, AS A **FREE DOWNLOAD**, FROM

WWW.TRS-80.ORG.UK

IN MAV'S TRAVELS

I ATTENDED VCF-SW

- The best VCF for TRS-80 enthusiasts -

by Ian Mavric



This year I decided to visit a VCF for the first time. As many know I am attended many Tandy Assembly shows, but people have been telling me how big the VCFs are. I chose the Southwest show (to be called VCF-SW or just SW from now on) which was held in Dallas, Texas, from 20-22nd of June. It is the closest VCF to where Tandy/Radio Shack was based and the show did not disappoint. Lots of former Tandy engineers and people who worked in the headquarters of Radio Shack drop into VCF-SW.

GETTING THERE - NOT AS BAD AS I EXPECTED

Normally when I go to TA the route taken is three legs, MEL->LAX, LAX->ORD, ORD->DAY, and going home is the reverse. Melbourne to

Texas is only 2 legs, MEL->SYD and SYD->DFW. That second leg is very long but once you are there the long flight is quickly forgotten.



Photo: Qantas A380 as used on the SYD->DFW route

Travelling from Australia is never cheap, and when I decided to go to SW it was late March, so I started looking for flights. The airfares were stuck resolutely high for more than 2 months, so high in fact that if they didn't come down I would most likely change my mind about attending SW this year. In late May Qantas had a 24 hour sale and dropped the round trip flights from $A3550.00 down to $A1780.00 I jumped at it and a day later the price went back up again. Lesson here: be patient, check the airline websites twice daily, and have your payment ready for when a price drop occurs.

ACCOMODATION - ONLY THE BEST FOR MAV

The organisers of SW have special room rates with two local hotels. One was at the Double-Tree by Hilton in Richardson TX, and the

special room rate was $US125.00 per night. I considered this very reasonable for a place with the Hilton's good reputation and thoroughly enjoyed my stay. The important things for me was that the bed was comfortable and the A/C effective. Speaking of A/C this is a good time to remind everyone that the temperature in Texas in June ranges between 34C and 38C each day, and nights never seem to drop below 28C (at least while I was there). Packing appropriately means lightweight clothing like short-sleeve shirts and shorts. I didn't mind the high temperatures, I wasn't out in the heat for any length of time, and every place indoors is well air-conditioned.



Photo: The Double Tree by Hilton Hotel was only 2 miles from VCF-SW

After breakfast in the buffet room I headed to VCF-SW for the Friday morning start which was around 10am.

WHEN YOU GET THERE - DAY 1 - FRIDAY JUNE 20

The first thing you see when you walk in the door is a bunch of 'freebie' tables, where people bring in things they just wish to give away. This area is always extremely busy, and popular items not even lasting a few minutes at the table before being collected and taken away. People are adding things all day for all three days so it makes good sense to check back regularly, like every few hours. And if you are uncertain about taking an item and want to walk away to think about it, don't be surprised if someone else takes it. Due to luggage weight limitations the only item I took from the freebie table was an IBM PCjr monitor cable:



A large amount of books (too heavy to bring home) and hardware (too heavy) was left behind by me, but the amount of free stuff was staggering.

Once you pass the freebie table you sign in and pick up your lanyard, which you need to wear all the time at the show.



See those little ribbons? They denote that I was both an exhibitor and also had a speaking presentation to do. I was met by Eric Dittman and we manned the Tandy Assembly table. On either side of us were Tandy/TRS-80 related exhibits by Eddie Malphrus, Joel Hilliard and Brendan Donahe.



Photo: Eric and I at the Tandy Assembly table (Brendan D.'s exhibit is to the far right of the picture, and Joel Hilliard's exhibit 'Downpoly' is on the left).

We had a Model 4 with a FreHD connected and a Model I with a FreHD/Quinnterface combo box set up. Our main task was the spread the word about Tandy Assembly and while we were at it we met some former Tandy/TRS-80 engineers and hit them up with the good word that we would like them to come and keynote speak at a future TA show.

We ran into Mark Segal who designed the Coco 3, and had a hand in the Tandy 1000 and 2500, as well as the Model 100. He was both involved as a designer and product team leader. Jerry Heep was also there, he had been with Tandy from the 1975 to 2012, and worked with Motorola developing chips for the Green Thumb project which eventually became the TRS-80 Color Computer.



Photo: Eric and I talk to Mark Segal, designer of the Coco 3 among other Tandy computer products.



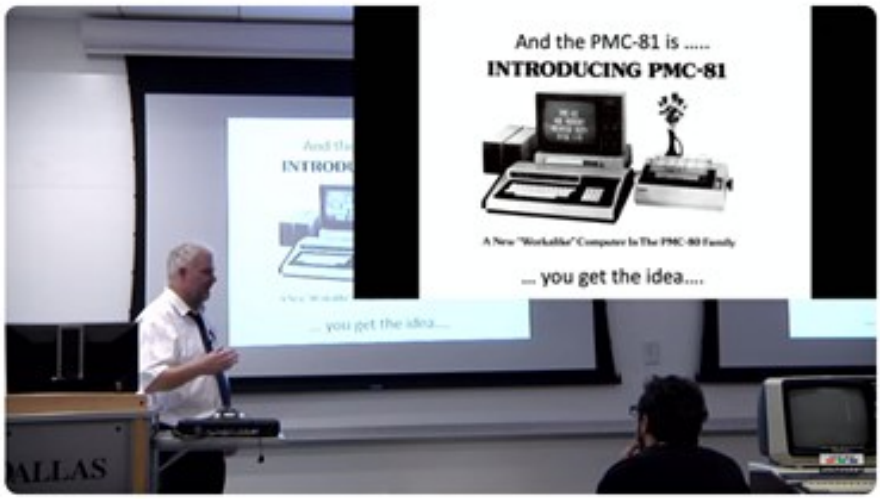
Photo: Eddie Malfrus' Tandy Showcase



Photo: Jerry Heep with a patent for some of his Coco work

DAY 2 - SATURDAY JUNE 21

On this day I did a talk about the Dick Smith System 80 computer. Known as the Genie-I in Europe and PMC-80 in the USA, it's a rare and interesting Model I compatible. Presentations are a major part of VCF shows and there were some 20 of them over 3 days to attend. Mine was on Saturday morning and I had a full room of people.



You can watch the full video here:
https://www.youtube.com/watch?v=f2Rs8SKp8mo

Saturday evenings at VCFs are devoted to the entertainment games and the auction. This year I was asked to help during the Hollywood Squares game:



DAY 3 - SUNDAY JUNE 22

Sunday is the shortest day, as people generally start packing up around 2:30pm and Eric and I were pretty much finished by 3:30pm. Earlier in the day we both attended a practical session to learn soldering of SMD devices. To attend this you were required to buy a kit of equipment needed to learn the skill.



Photo: One of the many classes at VCF. They were always packed with people.

I also took the opportunity on Sunday to buy as many goodies as I thought I could get in my luggage to bring home. The main Tandy item I bought was a non-working Color Computer 3, which once I got it home on my test bench was a quick fix (a bad Ram chip). PCI and ISA cards made up the bulk of the rest of my purchases. It was interesting to note that while a VCF is not specifically a yard-sale, flea market for vintage and retro computer items, I would say that some

50% of the exhibits had things for sale. So you will definitely buy something there.

On Sunday I also took time to catch up with old-mate Adrian Black from YouTube channel Adrian's Digital Basement, we had a great chat. I also caught up with many other YouTube content creators, as VCFs are a magnet for the tech YT channels.



Photo: myself and Adrian Black

SO IS IT WORTH IT?

Conversations I have had with fellow-minded vintage computer enthusiasts seem to have one common topic, attending a VCF, or VCFs, *is on the bucket-list* of many individuals. Cost is the main limiting factor for would-be VCF visitors living outside the USA. My trip came out to under $A4,000.00, of which some $A2,300.00 was for airfares and travel insurance. Passports are also expensive in Australia (nearly $A500 now for a ten-year passport) but I already

had one. The cost can be made more manageable by putting away savings monthly in the range of $A350-$A400 over 12 months, and keeping a close eye on airline websites for snap sales on the route you wish to travel. VCFs typically announce their dates for the next show around a year in advance so it's easier to work out a savings plan that way and organise the time off work to attend the event.

The dates for next year's VCF-SW show have already been announced as May 29-31, 2026 and I am planning to go. It would be great to meet up with more international fly-ins.



Ian Mavric

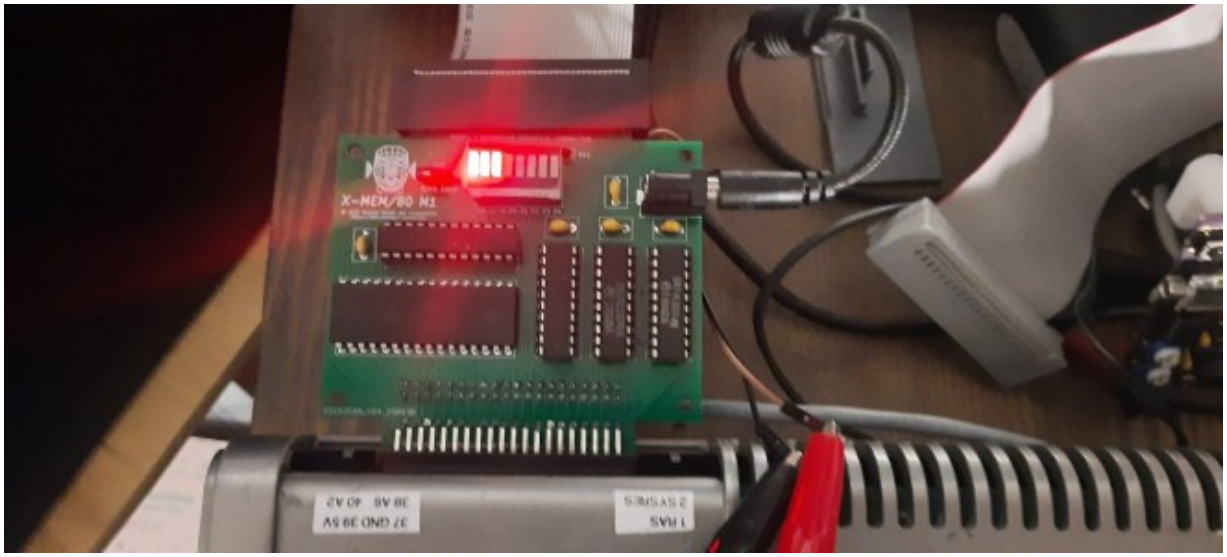
ianm@trs-80.com

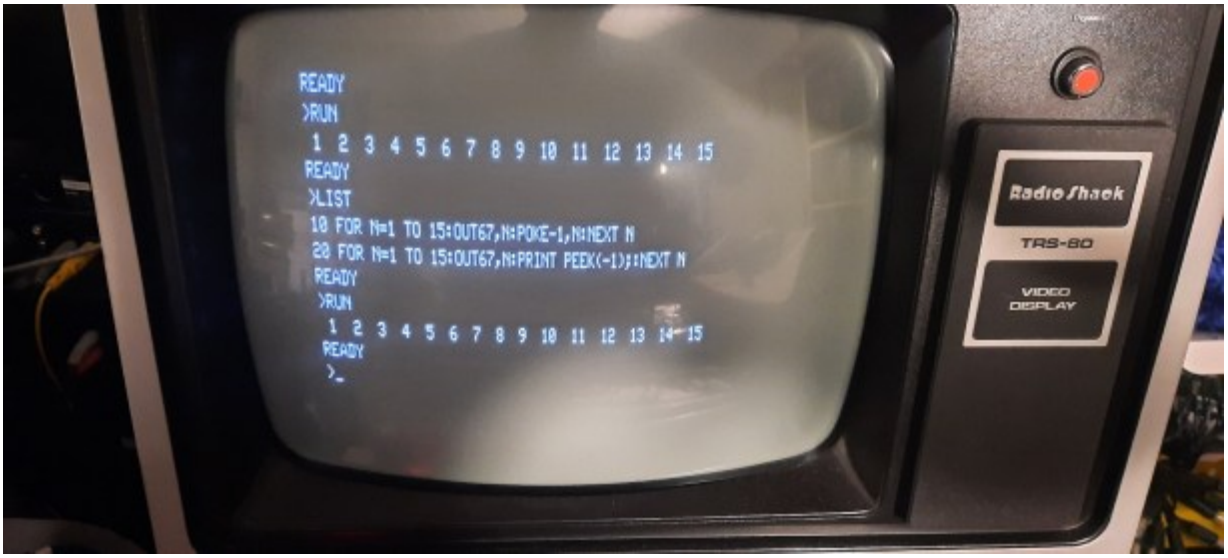


Ian Mavric is an IT Specialist who also restores and collects TRS-80's and classic cars. He lives with his wife and kids in Melbourne, Australia.

Surfin' The Net

Developed with colleagues on the TRS-80 Discord - it's compatible with the original 'Supermem' (16 32 KB pages from 0x8000 - 0xFFFF), but also offers 32x upper and lower 16 KB pages (0x8000 - 0xBFFF and 0xC000 - 0xFFFF).





WATCH THE VIDEO AT :-

https://www.youtube.com/watch?v=DtBPcdUbfEY



The *BEST* in TRS-80s

Call The Right Stuff

Ask for Ian
The number is +61 416 184 893

That's The Right Stuff
And he's in Melbourne







TRS8BIT IS SUPPORTED BY YOU AND PRODUCED
AT WWW.TRS-80.ORG.UK

TRS8BIT

PRODUCED AT WWW.TRS-80.ORG.UK

DECEMBER 2025 - VOLUME 19 ISSUE 4

PAGE 03 — DETAILS OF THIS YEARS **COMPETITION WINNER**
CONGRATULATIONS TO **BART JEROEN HEMSTRA**

PAGE 09 — IN MAV'S WORKSHOP
IAN MAVRIC
ON THE DOWNGRADE TO LEVEL 1
FANCY RUNNING UNDER LEVEL 1? MAV SHOWS US HOW!

PAGE 16 — POWER OF THE ROM
VINCE OTTEN
VINCE CONTINUES HIS SERIES WITH PART 4, USING
DR. DAVID LIEN'S BOOK "IT'S EASIER THAN YOU
THINK" WITH 'THANKS FOR THE MEMORIES'

PAGE 35 — SNOWFLAKE SCREEN SAVER
E.T. FONEHUME
A BASIC ONE-LINER, XMAS SCREEN SAVER

PAGE 36 — PROGRAMING LANGUAGES FOR THE MODEL 1
VINCE OTTEN
VINCE LOOKS AT EARLY PASCAL COMPILERS

PAGE 47 — NEW FROM FRANCE
PASCAL HOLDRY
PASCALS LATEST HARDWARE RECREATIONS.
FULL DETAILS ON HIS WEBSITE AT HTTPS://PROF-80.FR/

PAGE 53 — MAV'S TANDY ASSEMBLY NEWS
IAN MAVRIC
MAV'S REPORT OF THE 2025 T.A. AND LOOKS FORWARD TO 2026

PAGE 67 — A ROUNDUP OF SOME U.K. EBAY BARGAINS
WERE YOU ONE OF THE LUCKY ONES?

PAGE 69 — THE WAY WE WERE
A SHORT LOOK BACK TO THE EARLY 1980'S

PAGE 71 — AMAZED!
JUST FOR FUN! HAVE A GO AND SEE HOW QUICK YOUR ARE.

PAGE 72 — SURFIN' THE NET
'KIWISINCEBIRTH'
SHOWS OFF HIS NEW MODEL 1 BOARD. FANTASTIC!

PAGE 73 — IN MAV'S WORKSHOP
YOU BOUGHT A MODEL 2000 WHAT NOW! PART 2
IAN MAVRIC
MAV LOOKS AT HARDWARE ADD-ONS.
A VERY RARE MACHINE IN THE U.K. I CAN'T RECALL SEEING ONE!

MERRY CHRISTMAS EVERYONE

FROM DUSTY & DEE AT TRS8BIT

LATEST UPGRADE AVAILABLE SHORTLY
VERSION 2.5.6

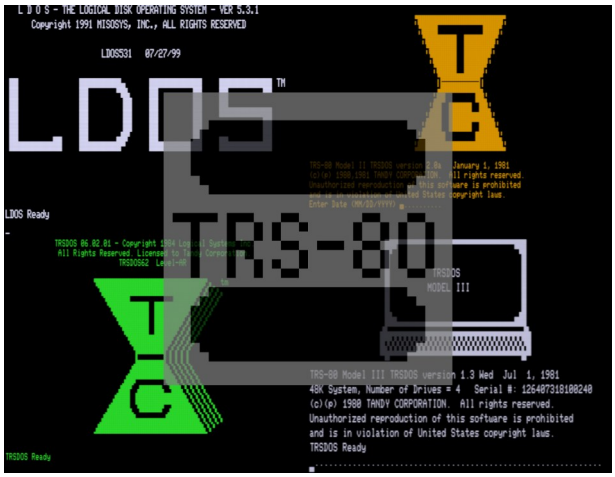
DON'T HAVE A TRS-80?

CRAZY PRICES ON EBAY?

TIRED OF RIFA CAPS BLOWING?

OF FLOPPIES SHEDDING OXIDE?

OR JUST PLAIN DEAD CHIPS?



THEN WHAT YOU NEED IS TRS80GP.
THE ULTIMATE TRS-80 EMULATOR

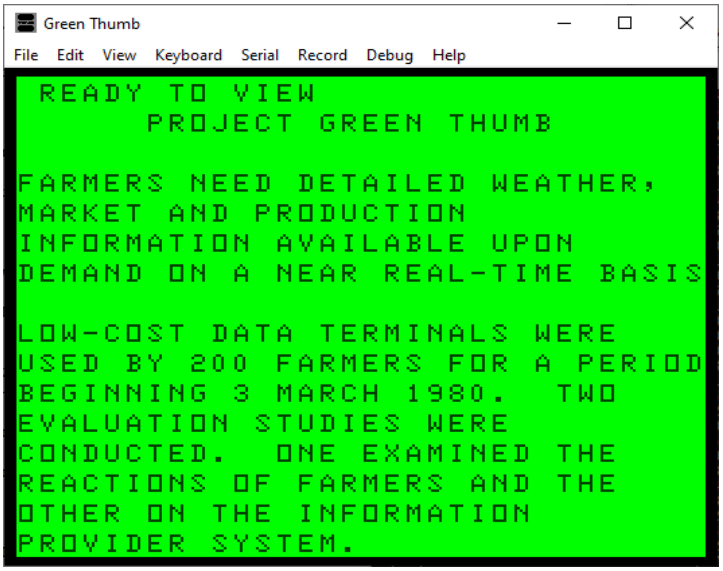
RUNS ON WINDOWS, MACOS, LINUX & RASPBERRY PI.
EMULATES MODELS 1, 2, 3, 4, 12, 16, 6000, MC-10,
COLOR COMPUTER, DT-1, VIDEOTEX, TANDY 10 AND NOW 'GREEN THUMB'.

JUST LIKE A REAL TRS-80 WITH NONE OF THE HASSLE.
BUILT-IN DEBUGGER AND OTHER FEATURES, JUST GREAT FOR PROGRAMMERS.

GET IT NOW AT HTTP://48K.CA/TRS80GP.HTML

UPDATED VERSION, 2.5.6, ADDITIONALLY OFFERS :

'GREEN THUMB' EMULATION
NEW SCRIPTING COMMANDS



IMPOSSiPUZZLE25 Competition Results

The 2025 competition involved writing a TRS-80 program to solve the number puzzle as seen on the U.K. Quiz program 'Countdown'. The object is to take 6 randomly selected numbers and use add, subtract, multiply and divide to mathematically combine them to reach a target number. The randomly selected numbers come from two sets – the large numbers 25, 50, 75 and 100 and the small numbers which are from 1 to 10 repeated twice. The player gets to choose how many large numbers to include, anywhere from 0 to 4 and the remaining six are picked from the small numbers. The target is a random number from 100 to 999. Each number can be used only once and in no stage of the calculation can you have a negative or fractional number.

Normally there is a limit of 30 seconds to come up with an exact solution or as close to the target number as you can. This year's winner wrote a clever little BASIC program that takes a bit longer than that but can solve an impressive range of possible puzzles. Congratulations to:

Bart Jeroen Hemstra

Here's how Bart went about developing the program:

My first idea is (as always) to brute-force my way through it. On my modern Mac it is no problem, and with modern software like Wolfram Mathematica it is quite easy to do as well. The program I made gave 50 different solutions for the puzzle with target 124.

```
  n = {75, 25, 8, 5, 5, 2};
  target = 124;
  g = Groupings[Array[Slot, 6], 2];
  f = Tuples[{Plus, Times, Subtract, Divide, #1 &}, 5];
  calcs = Function /@
    DeleteDuplicates@
     Flatten[Outer[ReplacePart[#, Thread[Position[#, List] ->
#2]] &,
          g, f, 1], 1];
  p = Permutations[n];
  all = Quiet@Outer[Apply, calcs, p, 1];
  Column@DeleteDuplicates[
    Position[all, target] /. {c_, v_} :>
      calcs[[c]] @@ (HoldForm /@ p[[v]])]
```

The first few of the 50 solutions:

```
5+2 (5-8+25)+75
(1+2) 8+25+75
(-2+5) 8+25+75
2+5-8+5 25
5-25+2 (5-8+75)…
```

I tried your other example with target 543, as well as many other ones I created randomly, or which I found on YouTube, including the 'Most Extraordinary Numbers Game Ever': with the numbers 25, 50, 75, 100, 3 and 6, with target 952.

While looking at all the solutions it occurred to me that many solutions have the same format. Most solutions use all 6 numbers, and most use primarily addition and multiplication, and less subtraction and division. Giving Mathematica another try I found out that the most common forms are:

```
((A+B)*C-D)/E+F
((A-B)*C-D)/E-F
(((A+B)*C*D)-E)/F
```

Since brute-forcing is very difficult on a small computer like the TRS-80 (especially speed and stack size are very limited) I decided to simplify my program in a very serious way. In such a way that not all solutions for all situations can be found but at least the more common solutions.

I decided to check only the most common solution forms, only the three I mentioned above! It will solve about 60% of all my tests (according to Mathematica) so I was a good start. I created a quick-and-dirty BASIC-program on my Model 4 and to my surprise it solved the problem in less than 90 seconds, only 3 times the time the participants in the quiz had!

What is more, the other example you gave can be solved this way, and even the Extraordinary game from YouTube is solved in no time.

What does the program do?

- Initialize all variables. The random figures in an array, target, best-till-now and counters are all integers
- Loop through all combinations of indexes, avoiding the use of the same index twice or more
- Calculate the solutions and set the best-till-now value

if best == target jump out of the loop and print the 'winning' solution

It runs on most of the TRS-80's without (much) modification. As I prefer the Model 4 with LDOS this is the machine it has been real life tested. In trs80gp it runs on Models 1, 2, 3 and 4. Most likely on the coco as well! On my Model 4 it takes 2:15 min to calculate the solution, on the Model 2 1:30 min.

While is it very feasible for a modern computer to search all combinations of the numbers to find the best answer, I wonder if this is in reach of the TRS-80? Coming up with an extract number of equations to try was beyond my capabilities. Suppose you have only two numbers, A and B. Then there are only 4 equations to try:

```
A + B, A – B, A * B, A / B
```

It might seem you need to also try B – A and B / A but it isn't necessary since our subtract and multiply are special. Because negative intermediate results are not allows, A – B is effectively equal to the absolute value of A – B. Or, put another way, only A – B or B – A need be tried. Either A = B and the answer is 0 or only one of A – B and B – A can be a positive number.

Similarly, either A / B = B / A or only one of A / B or B / A is a whole number.

Extending this reasoning up to even 3 numbers gets tricky. It's fairly easy to come up with over-estimates but exact numbers eluded me. Naturally, I turned to the internet to find the answer. However, the people working on solvers took the practical stance that even an overestimate of the combinations (say, 2 million or so possibilities) was no problem for a modern machine and rightfully saw to reason to do better.

I couldn't even find the answer in the ever wonderful On-Line Encyclopedia of Integer Sequences. But there was one that was close:

https://oeis.org/A140606

That lists the number of unique expressions of N variables using the conventional add, subtract, multiply and divide operations. For 2 variables it is 6 (as noted above) and for 6 variables it is a whopping 793,002.

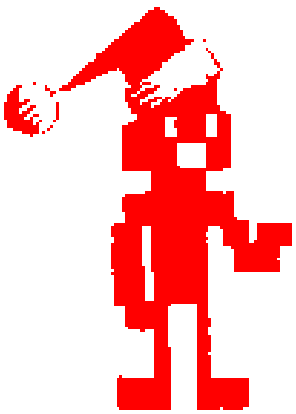
Given that our subtract and divide take away possibilities I feel confident the actual number to search will be smaller but dare not hazard a guess as to how much smaller. Perhaps some intrepid reader can solve this for us!

Exact numbers notwithstanding, I do think a TRS-80 could do a brute-force search. A model 4 can manage about 20 million instructions in 30 seconds. That works out to about 25 instructions for each combination. That isn't enough to evaluate every expression from scratch, but it does feel enough to operate on two numbers and I think that'll be fine. The trick is to check the expressions in an order where previous results can be re-used. For instance, if (A+B)+C, (A+B)-C, (A+B)*C and (A+B)/C are to be checked then you only need to add A and B once instead of 4 times. With more care in the ordering, clever programming tricks and an expected reduction in the actual number of expressions, it all looks achievable.

Program Listing:

```
1 REM COUNTDOWN IMPOSSIPUZZLE - MODEL 4
5 PRINT TIME$
10 DIM N(6)
20 N(1)=25: N(2)=75: N(3)=5: N(4)=2: N(5)=8: N(6)=5
30 TARGET = 124
40 BEST=0
50 FOR A=1 TO 6
60 FOR B=1 TO 6
70 IF B=A THEN GOTO 460
80 FOR C=1 TO 6
90 IF C=A OR C=B THEN GOTO 450
100 FOR D=1 TO 6
110 IF D=A OR D=B OR D=C THEN GOTO 440
120 FOR E=1 TO 6
130 IF E=A OR E=B OR E=C OR E=D THEN GOTO 430
140 FOR F=1 TO 6
150 IF F=A OR F=B OR F=C OR F=D OR F=E THEN GOTO 420
155 REM most common: ALGORITHM 1 ((A+B)*C-D)/E+F
160 X1=N(A)+N(B)
170 X2=X1*N(C)
180 X3=X2-N(D)
190 IF N(E)=0 THEN 420
200 IF X3/N(E)=INT(X3/N(E)) THEN X4=X3/N(E) ELSE 240
210 X5=X4+N(F)
220 IF ABS(X5-TARGET)<ABS(BEST-TARGET) THEN BEST=X5: PRINT
"BEST TILL NOW ALGORITHM 1: ";X5
230 IF BEST = TARGET THEN GOTO 510
240 REM other computation: ALGORITHM 2 ((A-B)*C-D)/E-F
250 X1=N(A)-N(B)
260 X2=X1*N(C)
270 X3=X2-N(D)
280 IF N(E)=0 THEN 420
290 IF X3/N(E)=INT(X3/N(E)) THEN X4=X3/N(E) ELSE 330
300 X5=X4-N(F)
310 IF ABS(X5-TARGET)<ABS(BEST-TARGET) THEN BEST=X5: PRINT
"BEST TILL NOW ALGORITHM 2: ";X5
320 IF BEST = TARGET THEN GOTO 540
330 REM yet another computation: ALGORITHM 3 (((A+B)*C*D)-E)/
F
340 X1=N(A)+N(B)
350 X2=X1*N(C)
360 X3=X2*N(D)
370 X4=X3-N(E)
380 IF N(F)=0 THEN 420
390 IF X4/N(F)=INT(X4/N(F)) THEN X5=X4/N(F) ELSE 420
400 IF ABS(X5-TARGET)<ABS(BEST-TARGET) THEN BEST=X5: PRINT
"BEST TILL NOW ALGORITHM 3: ";X5
410 IF BEST = TARGET THEN GOTO 570
```

```
420 NEXT F
430 NEXT E
440 NEXT D
450 NEXT C
460 NEXT B
470 NEXT A
480 PRINT "CLOSEST SOLUTION: ";BEST; "  TARGET IS: ";TARGET
500 GOTO 600
510 REM ALGORITHM 1
515 PRINT "EXACT SOLUTION: ";BEST
520 PRINT "((";N(A);"+";N(B);") *";N(C);"-";N(D);") /";N
(E);"+";N(F)
530 GOTO 600
540 REM ALGORITHM 2
545 PRINT "EXACT SOLUTION: ";BEST
550 PRINT "((";N(A);"-";N(B);") *";N(C);"-";N(D);") /";N
(E);"-";N(F)
560 GOTO 600
570 REM ALGORITHM 3
575 PRINT "EXACT SOLUTION: ";BEST
580 PRINT "(((";N(A);"+";N(B);") *";N(C);"*";N(D);") /";N
(E);")/";N(F)
590 GOTO 600
600 PRINT TIME$
```



WE DONE BART, A REALLY FANTASTIC EFFORT MAKING YOU A WELL DESERVED WINNER.

PLEASE CONTACT MAV, DIRECT, AND CHOOSE YOUR PRIZE

ED.

Floppy Days
Vintage Computing Podcast
with Randy Kindig

Pull out a vintage computer and compute as if it were yesterday.

Subscribe Today!

Tandy-specific Episodes:

- TRS-80 Model I
- TRS-80 Model II
- Tandy Pocket PC
- Steve Leininger and Don French interviews
- The MISE with Peter Bartlett
- CoCo
- Boisy Pitre and Bill Loguidice book interview
- CoCoFest and Tandy Assembly coverage
- and much more...

https://www.floppydays.com

https://www.soigeneris.com/universal-backpack-drive



Universal Backpack Drive

The Backpack is a small SD card 'disk drive' for vintage computers that runs from a single AA battery

Choose Options

- ☐ Preloaded 8GB uSD card [+$5.00]
- ☐ Spare CR1025 RTC Battery [+$1.00]
- ☐ Tandy WP-2 DB9 to DB25 adapter [+$5.00]
- ☐ Sinclair Z88 DB9 t0 DB25 adapter [+$5.00]
- ☐ Epson PX-8 / HX-20 adapter [+$10.00]

$65.00

IN MAV'S WORKSHOP

On the Downgrade... ... to Level I



by Ian Mavric

Over the years people have asked me for a Level I Rom set for their TRS-80 Model I. I used to ask why would you want them, Level II is so much better, but the repeated answer I received was they had so many Model Is, including ones that were originally Level I, and wanted to restore them to originality. This article explains how to accomplish this if you are missing the Level I Rom chips.

TWO LEVEL I MACHINES: 4K AND 16K RAM

When people think of the Model I they originally purchased and used back in the late 1970s they usually fall into one of two categories, those who had the Level I version with 4K, and those who had the Level II with 16K.
It may surprise people that Radio Shack also sold a Level II version with 4K and a Level I version with 16K. These were not particularly popular because 4K on a Level II machine did not really have enough memory to do much with the sophisticated programming environment provided by Level II. 16K on a Level I was not popular for the opposite reason, the lack of sophistication in Level I didn't lend itself to writing enormous complex programs. Also, at a load speed of 250 baud a full 16K long program could take as long as 20 minutes to load, souring the user experience of the machine.

The 4K Level I carried the catalogue number 26-1001 and the 16K Level I was catalogue number 26-1003. When you opted for the 16K Level I you paid quite a price premium over the 4K so Radio Shack added a numerical keypad to the 26-1003 as well. Look under your trusty TRS-80 Model Is and if the catalogue numbers are 1001 or 1003 then that was originally a Level I even though it was almost certainly upgraded to Level II before you acquired it.



Picture: Factory 16K Level I is catalogue number 26-1003.

QUICK 'N' DIRTY DOWNGRADE USING 1 EPROM

Level I originally came on a pair of 2K ROM chips, and then later in the production of the Model I in 1979 Level I came on a single 4K ROM chip.

Locating these can be difficult, but also due to the age of the chips being 45+ years old they are susceptible to byte-rot, making them unusable.

The method I explain below uses a new 2732 4K Eprom which should give a very good long service life. Even if you have original Level I ROM chip(s) I would recommend this method.

1. Program the 2732 with the file LEVEL1.ROM
2. Bend out pin 18 and solder a wire to pin 12 (ground) near the top of the pin so pin 12 still goes into the IC socket.
3. Bend out pin 21 and solder a wire about 10cm long to it.

At this point you need to remove the Level II Eproms out of Z33 and Z34 on the motherboard, or if you have an earlier Level II upgrade connected via a ribbon cable remove that, flip over the board and remove the Level II ROM board and the 4 (or 6) wires which run to the motherboard.
One of them, which is usually Yellow, is address line 11 (A11), take note of that.

4. Insert the 2732 into Z33 and solder the wire from pin 21 to where the Yellow A11 wire used to be, this time on the component side.

Once complete it should look like this:

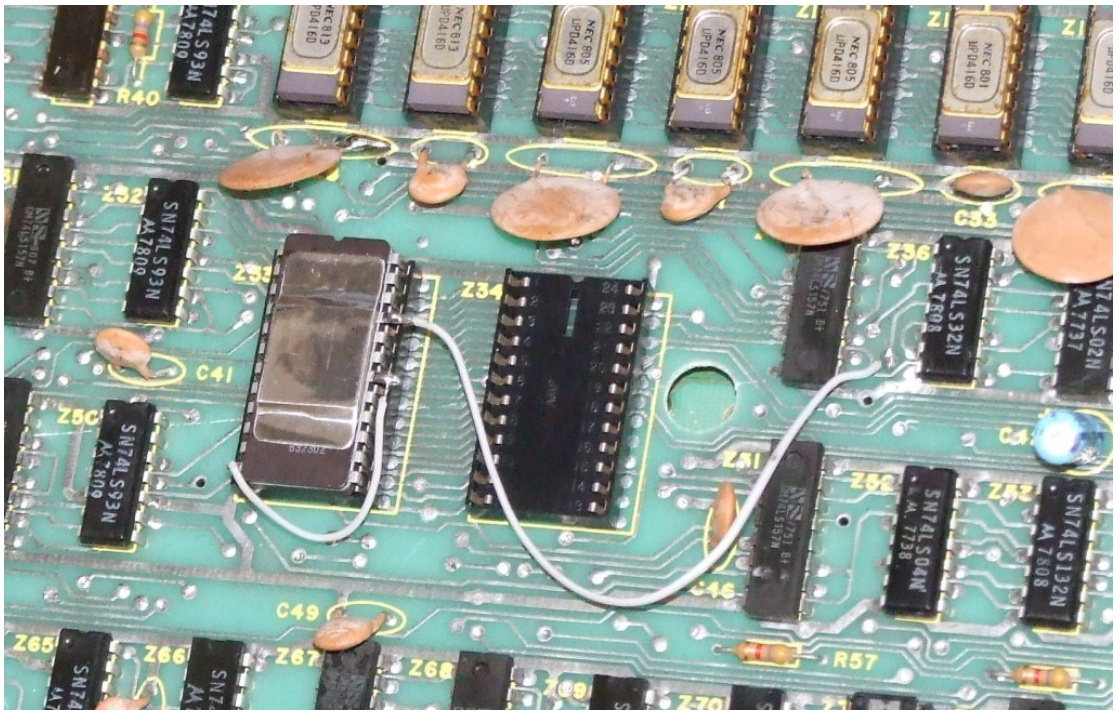
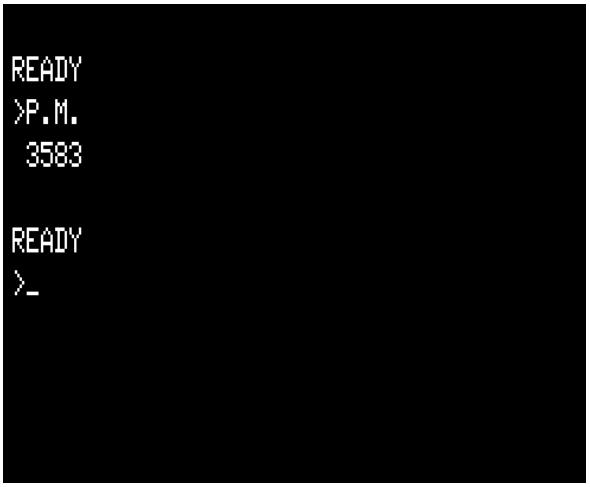
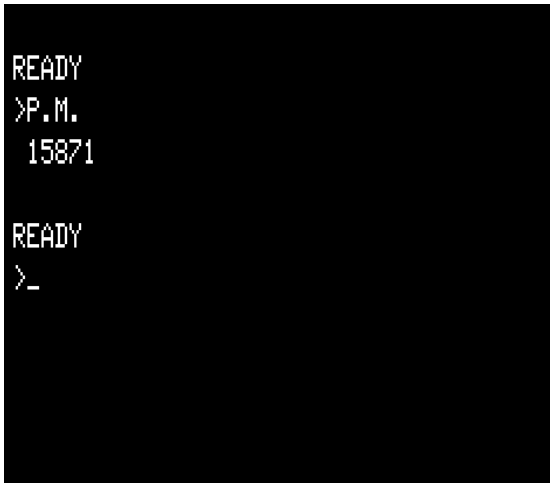


Photo: 2732 Eprom with Level I on it installed into Z33

Power up the machine and this time you should be greeted with READY>_ prompt instead of MEMORY SIZE of Level II. Type P.M.<Enter> which is the Level I version of Level II's ? MEM<Enter> and it should return the number of bytes free, which in a 4K machine is 3583 and on a 16K machine 15871.

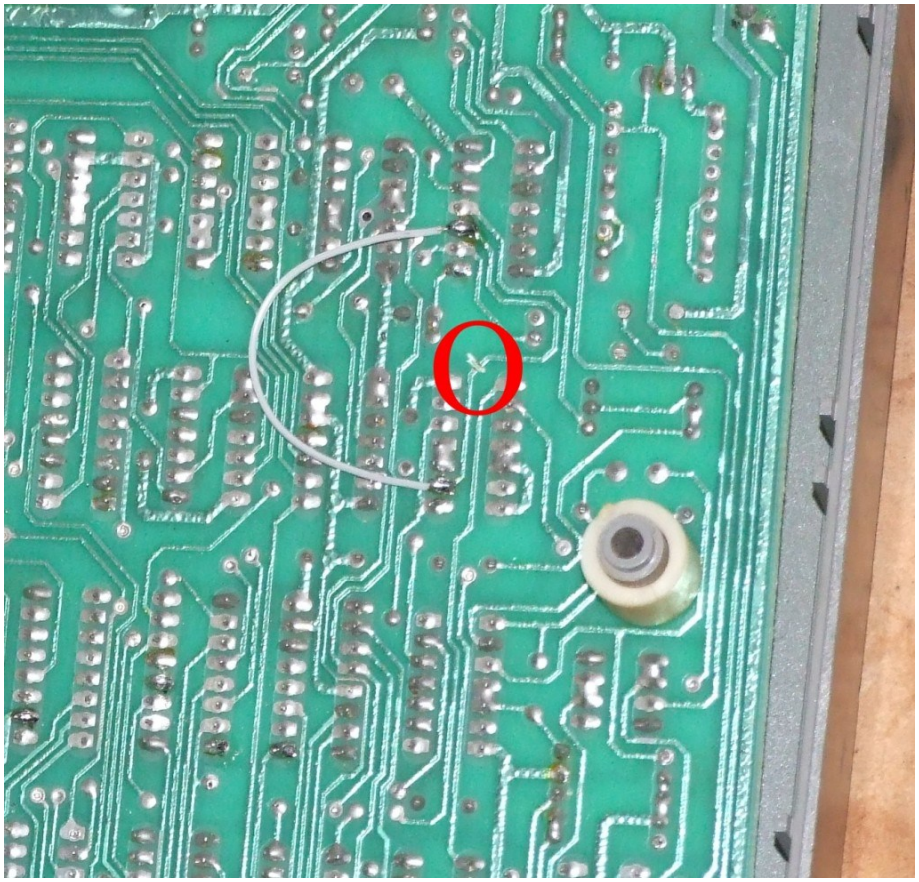




ABOUT THE CASSETTE LOADING

Many early Level II machines had a cassette modification called XRX-III which improved loading of 500 baud tapes. This modification doesn't work with Level I so if fitted, needs to be removed. 6-wires need to be unsoldered and one trace cut needs to be restored and the cassette functionality fixed.

In the diagram below the red circle shows where the trace cut normally is made with adding the XRX-III modification. You can bridge the cut with a small piece of wire like the lead from a resistor, or run a wire from the two points show by the grey wire in the picture. The result is the same.



Picture: restoring cassette function when removing XRX-III.

CONCLUDING

Little defence has been made of Level I Basic over the years. Level II Basic, at 12K in size is three times the size of Level I Basic but fifty times more useful. However, collectors are beginning to recognise the rarity of the Level I computer and for completeness in their collections a running Level I should always be part of their menagerie of TRS-80s. If you have questions on how to complete this downgrade feel free to reach out to me.

Ian Mavric
ianm@trs-80.com



Ian Mavric is an IT Specialist who also restores and collects TRS-80's and classic cars. He lives with his wife and kids in Melbourne, Australia.

BetaGamma Computing

Specialists in all things that compute





Based in Corfu Greece

- Qualified IT Engineers
- Repair of all PC's and Laptops
- Software and Hardware Upgrades and Installation
- Broadband and Network Installation
- Supply of Custom built to order PC solutions

RETRO RESTORATION SPECIALISTS



Call Corfu 26610 26358 Mobile 698 755 8427

Email: bgcompute@aol.com • www.betagammacomputing.com



Thanks for the MEMories - Vince Otten

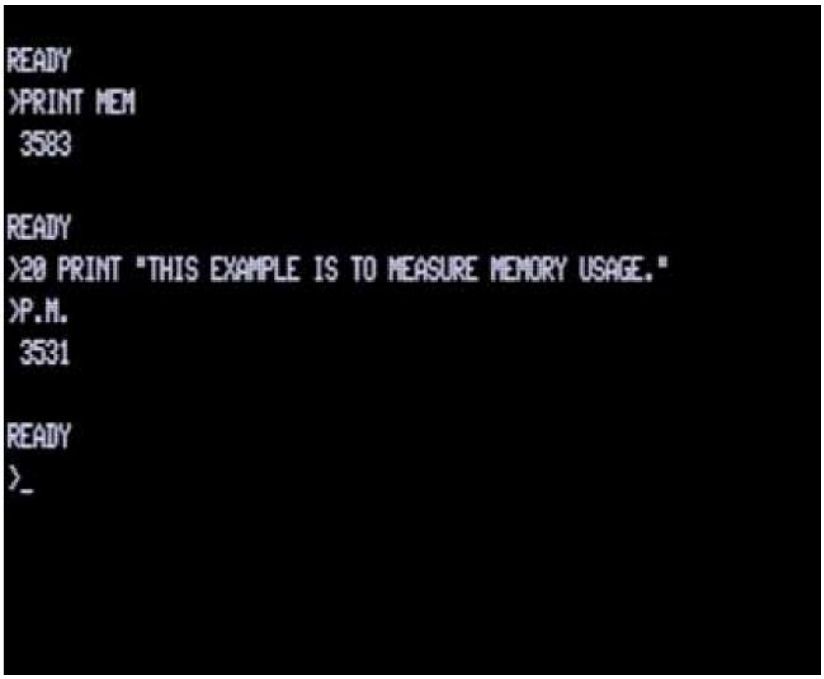
If you've been following this series on using TRS-80 Model I Level II BASIC ROM calls, you know that we've covered:

- CLS
- How to PRINT a string
- Beginning Math Calls:
    - Mandatory Setup Routines
    - The Floating Point Accumulator and "Number Type Flag"
    - Converting Integers to Single Precision
    - Single Precision Multiplication
    - Converting Single Precision to a String
    - (PRINT with BASIC's "Trailing Comma")
- "Pre-loading" Single Precision Constants
- Hand-parsing complex arithmetic
- More Math Calls:
    - Single Precision Division
    - Single Precision Addition
    - Single Precision Comparisons
    - Converting a String to a Single Precision Value
- Other ROM Calls:
    - LINE INPUT

This series follows the map of Dr. David Lien's sample programs in his text, "BASIC Computer Language: It's Easier Than You Think" that was shipped with every Level I TRS-80 back in the late 1970s. We've been able to translate those BASIC programs like

- a kind of "Hello World"
- a Distance / Speed / Time printout
- Celcius-to-Fahrenheit conversion, and
- how to compare Single Precision values and use them in IF...THEN...ELSE

procedures This time around, we're continuing from chapter 8, where Dr. Lien discusses what he calls "The Memory Command". He has us "PRINT MEM" on the early 4K Model I with Level I BASIC, then enter a program line, and determine remaining memory. In my screenshot, I use the old Level I abbreviations for PRINT and MEM, P. and M., in the second iteration.



A few quick reminders about this series of articles:

- Dr. Lien's excellent introduction to programming is about the initial *Level I* BASIC, which
  TRS-80 creator Steve Leininger adapted from "Tiny BASIC" with remarkable results.
- We're using it as an outline to discover *Level II* BASIC ROM routines.
- We're using *Z80 Assembly Language* rather than BASIC for our code.

Now: is there any reason to emulate BASIC's **PRINT MEM** (or the "P.M." abbreviation in Level I or "? **MEM"** in Level II) in Assembly Language? I can see a few uses:

- to determine how much free space we have left as we're developing an A/L program
- to confirm that there's enough memory for a completed, running program (does the user's machine have 4K as in the above screenshot? 16K? 48K?
- to measure how much memory we can allocate for strings or a heap in an application like an editor or a language compiler we're writing
- to report to the user of a running program how much memory they've got left (for text in a Scripsit-like word processor, or cells in a Visicalc emulation)

To calculate how much memory is free, we need two values:

1. The first memory address that our program doesn't use; and
2. The actual "memory size" of the TRS-80 our Assembly Language program is

running on. We will have to provide "value 1" ourselves. An easy way to do that for simple Assembly Language programs is to save the address following the last bit of code in our program, something like

```
;
; our program code and data ;
; finally:
;
LAST    EQU    &
        END
```

More complex A/L programs, such as those using libraries (with assemblers like Microsoft's Macro-80 and linkers like Link-80) may need more refined methods to determine the memory usage of your program. I leave this as an "exercise for the reader".
Next, we need to determine the highest RAM address. Model I TRSDOS stores that for us as a 16-bit value at 4049H. That means that, for our simple example above, we can determine "MEM" with code like

```
FREEM   EQU     $
        LD      DE,LAST         ; END OF PROGRAM
        LD      HL,(4049H)      ; END OF MEMORY
        OR      A               ; CLEAR CARRY FLAG
                                ; (FROM GEORGE PHILLIPS
                                ; "ADVANCED Z80 PROGRAMMING")
        SBC     HL,DE           ; SIZE OF FREE MEMORY
        LD      (MEM),HL
        ;
MEM     DEFW    0               ; "MEM" STORED HERE
        ;
```

Now to PRINT the MEM. It's an integer, which we can convert to a string, and then PRINT, right? Well, class, I don't intend to make this semester easy. Since we're looking at Single Precision ROM calls in this course, here's what we'll do:

1. Convert the Integer to a Single Precision value.
2. Convert the Single Precision value to a String.
3. Print the string.

We've already learned how to do each of these steps in previous articles: steps 1 & 2 in Article 2 (TRS8Bit June 2025) and step 3 in Article 1 (March 2025). Don't forget to set up the INIT and the CALL I00CRM "USES MATH" code from Article 2 before you do *any* math; CALL V4000R ("FORGET MATH") when you're done with the math ROM calls; and to JP to your FINISH exit point at the end of your code (ie, 402DH if you're writing a TRSDOS /CMD program). Here's what that code can look like:

```
;
;     *** MAIN() ***
;
MAIN        EQU $
              CALL     INIT
              CALL     I00CRM          ; "USES MATH"

;     1. GET END-OF-CODE VALUE -> DE
              LD       DE,MYLAST       ; "MY LAST" BIT OF CODE
;
;     2. GET FREE MEMORY SIZE "MEM" -> HL
              CALL GETMEM
```

In this case, my end-of-code is marked by

```
MYLAST      EQU $                     ; END OF MY CODE

              END
```

and I've turned the code which calculates the free memory into a subroutine:

```
;
;     GETMEM -- CALCULATE FREE MEMORY BASED
;          ON TRSDOS END-OF-MEMORY AND
;          "END OF CODE" VALUE IN DE
;          RETURN FREE MEM SIZE IN HL
;
G E T M E M  E Q U  $
              LD       HL,(4049H)      ; TRSDOS E-O-M
              OR       A               ; CLEAR CARRY
              SBC      HL,DE           ; SIZE OF FREE MEMORY
              RET
```

Carrying on with the main code:

```
;     3. CONVERT SIGNED INT IN HL -> S.P. VAR "MYMEM"
              LD       DE,MYMEM
              CALL CONVRT
```

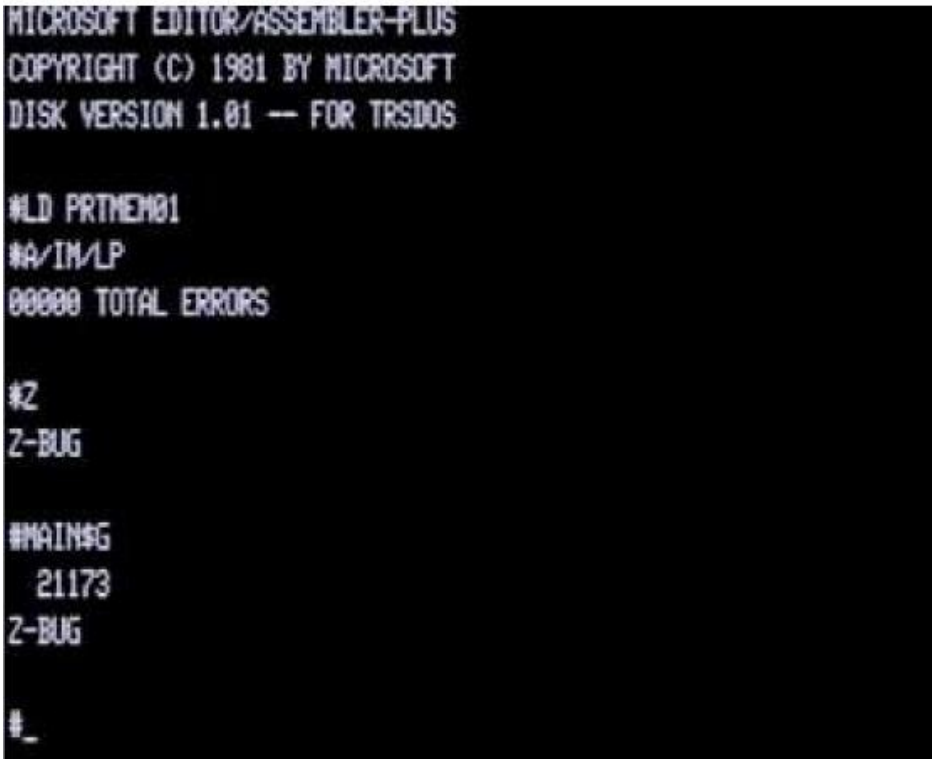
If you recall from Article 2, "CONVRT" is a subroutine which converts a signed integer to a Single Precision value which is then stored in a variable pointed to by DE. In this case, the variable is MYMEM.

```
;
;    4. CHECK FOR ERRORS
             LD      A,(BERROR)
             OR A
             JP      NZ,ERRORS
```

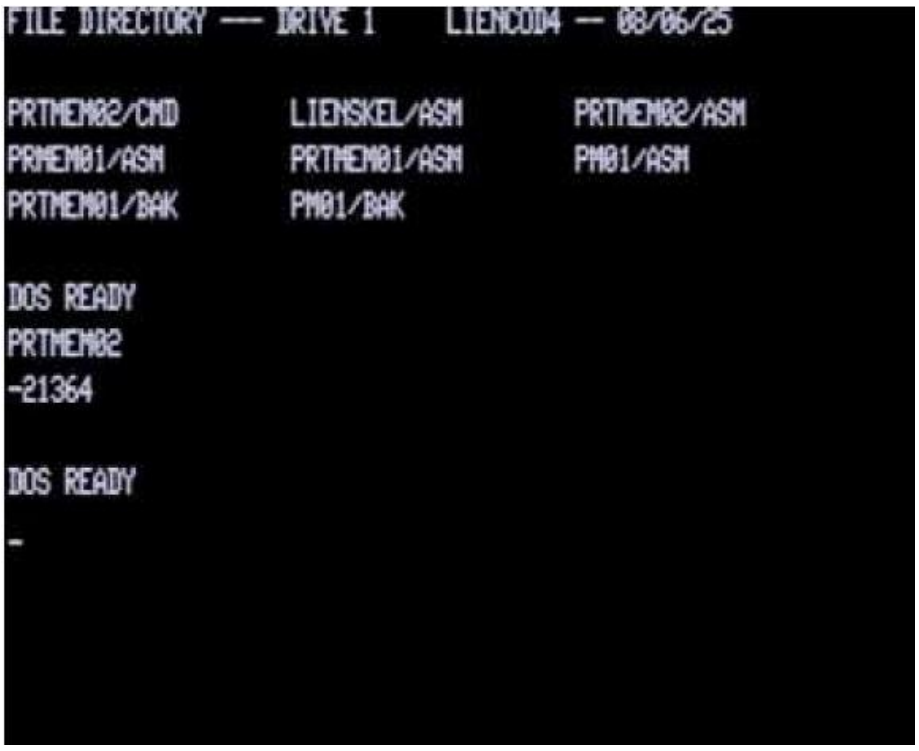
Remember that we identified BERROR with 409AH, the location where the ROM stores any BASIC errors. If there are no errors, BERROR will contain a zero. In my code, I have a separate routine ERRORS which will print out a generic "MATH ERROR(S)" message and jump to FINISH to exit. It's usually a good idea to jump to or call an error-handling routine if BERROR is non-zero, although there may be times where you can just ignore it. This method of checking BERROR after a ROM call seems to me to be the simplest way to trap errors. By the way, if you check back in Article 2 you'll notice that the INIT subroutine zeroes BERROR before your main code starts running. On we go:

```
;    5. CONVERT S.P. VALUE IN WRA1 TO STRING
;         AND RETURN ADDR OF STRING IN HL
             CALL GETSPC                ; "ON ERROR RESUME NEXT"
             CALL 0FBDH                 ; CONVERT WITH NO FORMATTING
;
;    6. CHECK FOR ERRORS
             LD      A,(BERROR)
             OR A
             JP      NZ,ERRORS
;
;    7. PRINT "MEM" (STRING POINTED TO BY HL)
             CALL PRINT
;
;    8. ALSO PRINT A NEWLINE
             LD      HL,RETURN
             CALL PRINT
```

I assembled and tested this code in Microsoft Editor Assembler Plus:

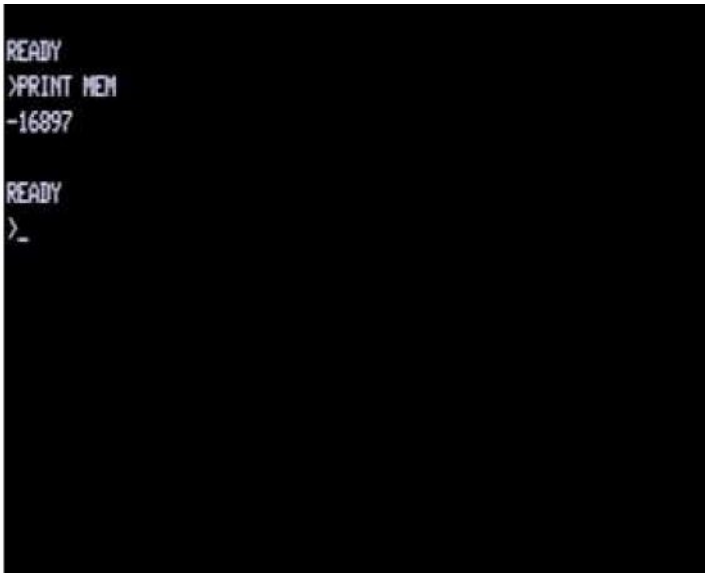


21173 bytes remaining—with its source code, assembler, and Z-BUG debugger all in memory as well, that's not too surprising. Next, I converted it to a /CMD file and ran it from the TRSDOS prompt:

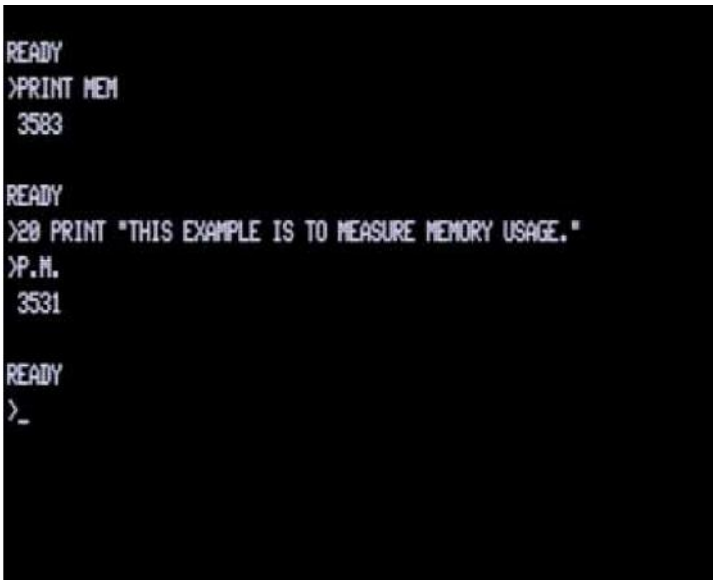


-21364 HUH!

Well, that's similar to what you might see in a 48K Level I BASIC TRS-80:



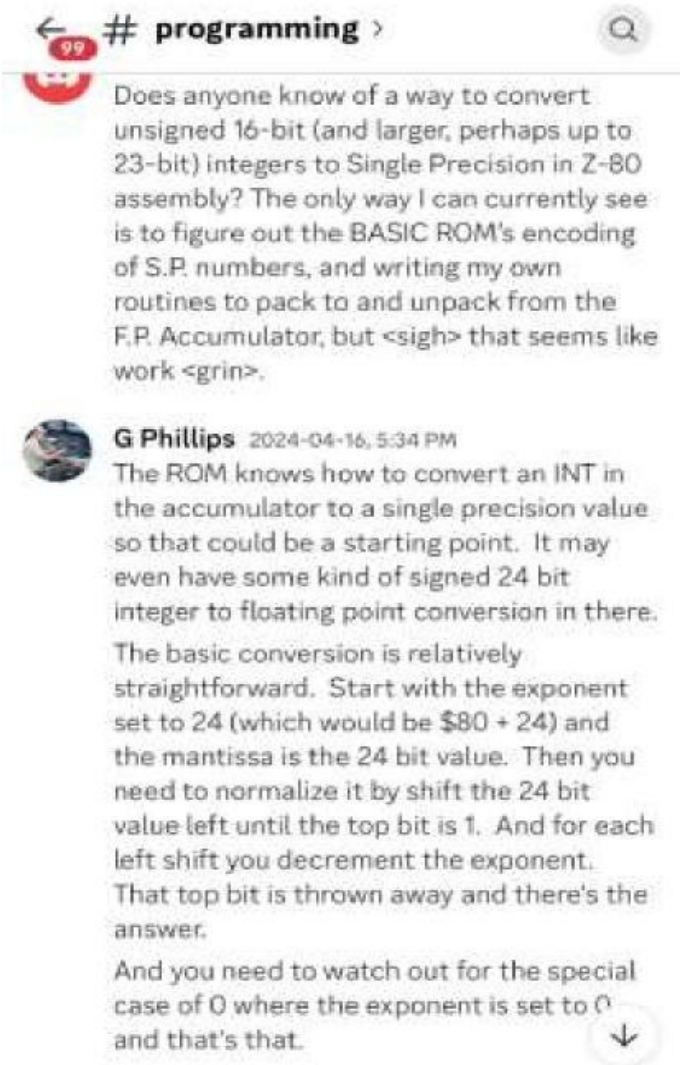
Contrast that with the more common 4K Level I:



Of course—these are *signed* integer values. Remember that an unsigned FFFFH corresponds to the signed value -32767 in 16-bit integers. Our code, as it stands, reports the memory size just like BASIC does—as a signed integer. In my series "Software Tools in BASIC", I presented a defined function which would convert a regular signed integer to an unsigned Single Precision value:

```
DEF FN IS!( I%)= (I% < 0) * -(65536 + I%) + (NOT I% < 0) * -I%
```

If the signed integer IS% was less than zero (like some of our MEM values here), the test "(I% < 0)" would return a -1, otherwise it would return a 0. That gets multiplied by the negated value "65536 + I%", so if I% is -16897 for example, the resulting value is -1 x (65536 + -16897) which is 48639. On the other hand, if I% is 3531, then the test I% < 0 returns a 0, which gets multiplied by whatever 65536 + 3531 is, in other words 0. So much for the first half of my DEF FN IS!(). Now we go on to add that result, in our examples here either 48639 or 0, to the second half of the expression (NOT I% < 0) * -I%. That's a simpler equation. If I% is -16897, for example, then "NOT I% < 0)" is NOT -1, or 0, which gets multipled by –I%, and returns 0. 0 gets added to our previous value of 48639 supplied by the first half of our function, and doesn't alter it in any way. On the other hand, if I% is 3531, then the test "NOT I% < 0)" is NOT 0, or -1, which gets multiplied by -I% or -3531, which results in +3531. That gets added to the previous value of 0 supplied by the first half of our function, and returns an unaltered 3531. Voila.
Do we really have to do all that work in Assembler to convert our actually-unsigned-but-apparently-signed MEM value to a Single Precision value? I posted that question in the Tandy Discord server. In one reply, an algorithm was suggested which converts up to 32 bits of unsigned integer numbers to Single Precision values.



As we've seen before, BASIC's Floating Point numbers have an unusual internal representation. Here's part of a spreadsheet in which I track how well my attempts match BASIC.

Floating Point Representation of Integers

Val Int	Hex	Var MSB	...	LSB	EXP	CONVRT	UI2SP Macro	SUI2SP Subroutine
0	00	0	0	0	0	x	x	x
1	01	0	0	0	81	x	x	x
2	02	0	0	0	82	x	x	x
3	03	0	0	40	82	x	x	X
4	04	0	0	0	83	x		X
5	05	0	0	20	83	x		
6	06	0	0	40	83	x		
7	07	0	0	60	83	x		X
8	08	0	0	0	84	x		X
9	09	0	0	10	84	x		
10	0A	0	0	20	84	x		
11	0B	0	0	30	84	x		
12	0C	0	0	40	84	x		
13	0D	0	0	50	84	x		
14	0E	0	0	60	84	x		
15	0F	0	0	70	84	x		X
16	10	0	0	0	85	x		X
17	11	0	0	8	85	x		
18	12	0	0	10	85	x		

It's a bit ragged, but here's some observations:

1. A zero value is identified by an Exponent of 0.
2. Exponents are not "powers of 10", but rather powers of 2, in this sense: subtract 81 from the exponent, and that will indicate the highest bit number set. For example,the exponent of 1 is 81; 81 – 81 is 0; a "1" in binary is two bytes of 0 and one byte with bit 0 set. A "3" in binary is two bytes of zero and one byte whose "highest" bit is set is bit Exponent:83 – 81 = bit 2. Don't ask me what the MSB..LSB encoding is.
3. The 3-byte mantissa is stored with the most significant byte at the *lowest* memory location, and the least significant byte at the *highest.* (This keeps getting me screwed up, because of course the Z80 understands what's called "little endian", where the low byte of a 16-bit value is store in the low byte, and then we store the high byte.)

So—I adapted George's suggested algorithm. Here's the code:

```
;
; SUI2SP – "UNSIGNED INT TO S.P" SUBROUTINE
;         CONVERTS 24-BIT UNSIGNED INT IN E/HL
;         TO S.P. VALUE IN WRA1
;
S U I 2 S P  E Q U  $
             XOR     A              ; ASSUME 0 VALUE
             OR      L              ; TEST FOR 0 VALUE
             OR      H
             OR      E
             JP      Z,JL0003       ; IF (UINT != 0):
             LD      A,80H-i-23      ; INITIALIZE EXPONENT
JL0001       EQU     $               ; REPEAT:
             ADD     HL,HL            ; SHIFT L UINT LO
             EX      DE,HL            ; DO SAME IN DE—
             ADC     HL,HL            ; SHIFT L UINT HI
             EX      DE,HL            ; RESTORE DE/HL ORDER
             BIT     7,E              ; 24TH BIT SET?
             JP      NZ,JL0002        ; IF (24TH BIT NOT SET):
             DEC     A                 ; ADJUST EXPONENT
             JP      JL0001           ; :WHILE (! 24TH SET)
JL0002 EQU $
             RES     7,E              ; RESET 24TH BIT
JL0003       EQU     $               ; :ENDIF
             LD      D,A             ; HIGHEST BYTE= EXPONENT
             LD      (WRA1),HL       ; STORE CONVERTED VALUE
             LD      (WRA1-i-2),DE   ; IN ACCUMULATOR
             LD      A,4             ; AND "S.P." TYPE
             LD      NTF1,A          ; IN NUMBER TYPE FLAG
             RET
```

To use this, load your 24-bit unsigned integer into E/HL, with the highest byte in E and the lowest in L, then CALL SUI2SP. The resulting Single Precision representation can be found in WRA1. *Only—there's a problem.* I tested the subroutine out using the spreadsheet, and as soon as I got to a value greater than 7FFFH or 32767, it started producing erroneous results. There was something missing.

So then I thought, "What if I traced through the ROM for BASIC's CVSNG() function? Ira Gold's trs-80.com website provides commented disassemblies of the Model I and Model III BASIC ROMs, like this:

0ACCH-0ADAH -LEVEL II BASIC NUMBER CONVERSION ROUTINE – "CONSI"

Convert Integer to Single Precision. Every register is affected.

Note: If you wanted to convert integer to single precision via a ROM call, you would store the integer input variable in 4121H-4122H and then call to 0ACCH. The result (as a single precision number) will be in 4121H-4124H.

0ACC-0ACE L CONSI	LD HL,(4121H) LD HL,(FACLO)	Load Register Pair HL with the integer value from the ACCumulator
0ACF-0AD1 L CONSIH	CALL 0AEFH CALL VALSNG	Go set the current number type flag to single precision
0AD2	LD A,H	Now we need to prepare the registers for the FLOATR routine. First, load Register A with the MSB of the integer value in Register H
0AD3	LD D,L	Load Register D with the LSB of the integer value in Register L
0AD4-0AD5	LD E,00H	Zero Register E
0AD6-0AD7	LD B,90H	Load Register B with the initial maximum exponent
0AD8-0ADA	JP 0969H JP FLOATR	Jump to 0969H to float the integer into single precision

Folks: the work Ira has done is incredibly useful. Used side-by-side with the Debugger in the trs80gp emulator, it's possible to understand just what Bill Gates, Paul Allen, and Monte Davidson did when they created their 8-bit 8080 BASIC. Even the labels like FACLO, VALSNG, and FLOATR have been copied over from their source code.
But in the case of "CONSI" here, it soon became too complex for me to understand and track. There had to be a simpler solution. Then it occurred to me: "Wait a minute. I'm supposed to be using BASIC ROM calls. What if I do my 24-bit unsigned Integer to Single Precision conversion this way?"

1. Convert the high byte to S.P. and multiply it by 65536. Save it as "HighByteSP".
2. Get ready to add the low 16-bit value to that.
3. But first: if the low 16-bit value is negative, what do we do with it? For example, -32767 is actually an unsigned value of FFFFH or 65535.

a. In assembly language: test if it's negative. If not, just go to step 4, below. If it is,
Check if it's an odd number (and add a 1 later if so: see step below).
Divide it by 2 to make it a positive number.

4. Convert the low 16-bit value to S.P.
5. If it was a negative number, multiply it by 2 to restore its original unsigned value.
    a. Also, if it was an odd negative number, add the 1 that dropped off when we divided it by 2 in step 3.a.i, above.
6. Now add the high byte value "HighByteSP" to our converted low-16-bit value.

Here's what that looks like in pseudocode:

```
float SubUnsignedIntToSinglePrecision( EHL: 24 bit unsigned int) {
/* 1. High 8 bits: */
HiZeroFlag= (E == 0);
if (!HiZeroFlag) {

        /* save low 16 bits          */
        /* "PUSH HL"                 */
        D= 0;
        WRA1= DE;
        NTF1= 2;
        ConvertToSP( WRA1);
        WRA1 *= 65536; TempSP= WRA1;
        /* restore low 16 bits       */
        /* "POP HL"                  */
        }
    /* Test Sign Bit                 */
    SignFlag= ("BIT 7,H"); if (SignFlag) {
        /* Test "Odd" Bit            */
        OddFlag= ("BIT 0,L");
        /* Shift HL right            */
        /* ie,      "SRL H"
                    "RR L"           */
        HL= HL / 2;
        }
    /* Convert to S.P.               */
    WRA1= HL; NTF1= 2;
    ConvertToSP( WRA1); if (SignFlag) {
        /* we divided it by 2,
             so restore it * 2 */
        WRA1 *= 2;
        if (OddFlag) {
             /* low bit was set (odd),
                  so add 1 */
             WRA1 += 1;
             }
    if (!HiZeroFlag) {
        /* add value of high (E) byte */
        WRA1 += TempSP;
        }
```

That's getting more complex! I've noticed that pseudocode translates well into BASIC code (like it did in my "Software Tools in BASIC" articles), but less so for assembly language. Here, everything is so fine-grained. You notice that I had to set and take notice of 3 flags: HiZeroFlag, SignFlag, and OddFlag. Higher-level languages like BASIC just "automatically" remember things like this, but in assembly language you're constantly shuffling results around. Here's the resulting SUI2SP ("Subroutine: Unsigned Int To Single Precision"):

```
;
; SUI2SP -- "UNSIGNED INT TO S.P." SUBROUTINE
;     CONVERTS 24-BIT UNSIGNED INT IN E/HL
;     TO S.P. VALUE IN WRA1
;
; S.P. CONSTANT: 65536
;
TWO16 DEFW      0000        ; S.P. 65536.0, LOW 2 BYTES
      DEFB      00          ;       3RD BYTE
      DEFB      145         ;       EXPONENT
ISFLGS DEFB     0           ; FLAGS BYTE
                                ; BIT   0: "HI !ZERO"
                                ;       1: "SIGN"
                                ;       2: "ODD"
;
; TEMPORARY STORAGE OF HI BYTE * 65536 AS S.P.
;
TEMP24 DEFS4
;
SUI2SP EQU      $
;
; A. INITIALIZE FLAGS
;
      LD        IX,ISFLGS
      LD        (IX+0),0
;
; 1. HIGH 8 BITS
;
TEST FOR 0
      LD        A,E
      OR        A
;
SAVE "HI !ZERO" FLAG
      JR        Z,LOW16     ; IF (!0):
      PUSH      HL              ; SAVE LOW 16 BITS
      SET       0,(IX+0)        ; SET "HI !ZERO" FLAG
```

```
; 1C. IF !ZERO--
;     1D. CONVERT TO S.P.
          LD        D,0
          LD        (WRA1),DE          ; SAVE INT IN WRA1
          LD        A,2                ; NTF= 2
          LD        (NTF1),A           ; IE, "INTEGER"
          CALL      GETSPC             ; "ON ERROR RESUME NEXT"
          CALL      0ADBH              ; CONVERT TO S.P.
;
;     1E. MULTIPLY BY 65536.0
          LD        DE,(TWO16)         ; S.P. 65536.0 LOW WORD
          LD        BC,(TWO16+2)       ; HI WORD + EXP.
          LD        A,4                ; NTF1=
          LD        (NTF1),A           ; 4 / S.P.
          CALL      GETSPC             ; ON ERROR RESUME NEXT
          CALL      0847H              ; WRA1= WRA1 * 65536.0
;
; 1F. SAVE RESULT IN TEMPORARY STORAGE
          LD        HL,(WRA1)          ; RESULT LOW 16 BITS
          LD        (TEMP24),HL
          LD        HL,(WRA1+2)        ; RESULT HIGH BYTE + EXP
          LD        (TEMP24+2),HL
          POP       HL                 ; RESTORE LOW 16 BITS
LOW16 EQU           $                  ; :ENDIF
;
; 2. LOW 16 BITS:
;
;2A TEST BIT 15 (SIGN BIT)
          BIT       7,H                ; Z RESET IF BIT 15 (SIGN) SET
          JP        Z,NOSHF            ; IF (SIGN):
          SET       1,(IX+0)               ; SET "SIGN" FLAG
;
TEST BIT 0 (ODD/EVEN)
          BIT       0,L                ; Z RESET IF ODD VALUE
          JR        Z,I2SHF            ; IF (ODD):
          SET       2,(IX+0)           ; SET "ODD" FLAG


;
; 2C  SHIFT HL RIGHT
12SHF     EQU       $
          SRL       H
          RR        L                   ;FROM "CHILLIANT.COM"
NOSHF     EQU       $                   ;:ENDIF
;
` 3. CONVERT TO S.P.
;
;3A MOVE HL -> WRA1 AND SET AS INTEGER
```

```
        LD      (WRA1),HL
        LD      A,2
        LD      (NTF1),A
;
; 3B. CONVERT TO S.P.
        CALL    GETSPC          ; ON ERROR RESUME NEXT
        CALL    0ADBH           ; CONVERT INT -> S.P.
;
; 4. IF "SIGN" FLAG SET--
;
        BIT     1,(IX+0)        ; GET "SIGN" FLAG
        JP      Z,NOSHF2        ; IF (SIGN):
;
; 4B. MULTIPLY BY 2.0
        LD      DE,0000H            ; BC/DE=
        LD      B,130               ; S.P.
        LD      C,00                ; 2.0
        LD      A,4                 ; S.P.
        LD      (NTF1),A            ; OPERATION
        CALL    GETSPC              ; ON ERROR RESUME NEXT
        CALL    0847H               ; WRA1= WRA1 * 2.0



;
; 4C IF "ODD" FLAG SET
      BIT       2,(IX+0)            ; IF (ODD)
      JP        Z,NOSHF2
;
; 4D. ADD 1.0
        LD      DE,0000H            ; BC/DE=
        LD      B,129               ; S.P.
        LD      C,0                 ; 1.0
        CALL    GETSPC              ; ON ERROR RESUME NEXT
        CALL    0716H               ; WRA1= WRA1 + 1.0
                                    ; :ENDIF
;
; 5. "LOW" 16 BIT VALUE NOW IN WRA1 AS S.P. VALUE
;
NOSHF2 EQU      $               ; :ENDIF
;
; 5A TEST "HIGH !ZERO" FLAG
        BIT     0,(IX+0)
        JP      Z,NOADD         ; IF (HIGH BYTE != 0):
;
; 5B IF HIGH BYTE WAS NOT 0
        LD      DE,(TEMP24)         ; BC/DE=
        LD      BC,(TEMP24+2)       ; HI BYTE * 65536.0
;
```

```
; 5C. ADD IT (* 65536.0) TO WRA1
        CALL    GETSPC          ; ON ERROR RESUME NEXT
        CALL    0716H           ; WRA1= WRA1+ HI * 65536.0
NOADD   EQU     $               ; :ENDIF
        RET                     ; WRA1= S.P. 24-BIT UINT
```

This subroutine comes with 3 items of data:

- TWO16, a Single Precision version of the constant 65536.0
- TEMP24, a 4-byte temporary storage for the converted High Byte
- ISFLGS, "Integer to Single Flags", a byte in which we maintain the "High !Zero", "Sign", and "Odd" flags

I'm sure there's a more efficient way to convert a 24-bit unsigned Integer to Single Precision. In my defense, I'll point out that

- This uses ROM routines, which is kind of the point
- It's a relatively rarely used conversion: mostly for printing out unsigned integer values, so it doesn't have to be all that "fast" (it's still much faster than BASIC)

Here's an example of "PRINT MEM" written in Z80 Assembler for TRSDOS 2.3. It assumes that the source includes code for

Subroutine	*TRS8Bit Issue Volume & Number*	*Date*
INIT	19 #2	June 2025
I00CRM		
GETSPC		
PRINT	19 #1	March 2025

A sample run follows.

```
;           PRTMEM/ASM -- PRINT MEMORY DEMO
;
;                       SEP 05, 2025
;                       TESTED: 9/5/25

            TITLE       'PRTMEM/ASM'

            ORG         5200H
;
;   *** MAIN() ***
;
MAIN        EQU         $
            CALL        INIT
            CALL        I00CRM      ; USES MATH
```

```
; 1. GET END-OF-CODE VALUE -> DE
            LD              DE,MYLAST ; "MY LAST" BIT OF CODE
;
; 2. GET FREE MEMORY SIZE "MEM" -> HL
            CALL            GETMEM
;
; 3. CONVERT SIGNED INT IN HL -> S.P. VALUE IN WRA1
            LD              DE,0        ; DE: 0000 HL: INT VALUE
            CALL            SUI2SP
;
; 4. CONVERT S.P. VALUE IN WRA1 TO STRING
;    AND RETURN ADDR OF STRING IN HL
            CALL            GETSPC      ; ON ERROR RESUME NEXT
            CALL            0FBDH       ; CONVERT W/ NO FORMATTING
;
; 5. HECK FOR ERRORS
            LD              A,(BERROR)
            OR              A
            JP              NZ,ERRORS
;
; 6. PRINT MEM$ (POINTED TO BY HL)
            CALL            PRINT
;
; ALSO PRINT A NEWLINE
            LD              HL,RETURN
            CALL            PRINT

            JP              FINISH

ERRORS EQU                  $
            LD              HL,ERRMSG
            CALL            PRINT
            LD              HL,RETURN
            CALL            PRINT
            JP              FINISH
;
;    FINISH -- EXIT TO TRSDOS
;
FINISH      EQU             $
            CALL            V4000R      ; RESTORE DOS EXITS

FIN01       JP              402DH       ; EXIT TO TRSDOS
;
;    ** DATA DIVISION **
;
;    * SYSTEM DATA *
```

```
;
;
BERROR      EQU   409AH        ; BASIC 'ERR' HERE
                               ; CHECK THIS AFTER USING
                               ; ROM  MATH CALL
                               ; 0 IF NO MATH ERROR
WRA1        EQU   4121h        ; WORKING REGISTER 1
                               ; (INT & S.P. BASE ADDR)
NTF1        EQU   40AFH        ; "NUMBER TYPE FLAG" 1

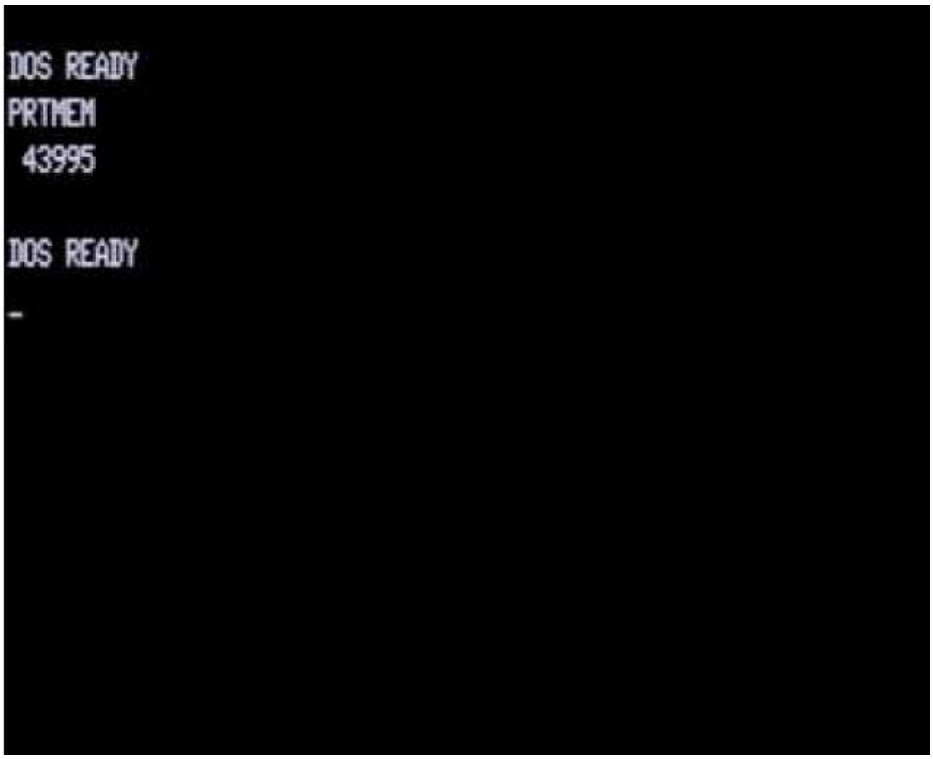
RETURN      DEFW  0000H        ; NEWLINE
;
;       * USER DATE *
;

MYMEM       DEFS  4            ; "MY 'MEM' FREE MEMORY AS S.P.

ERRMSG      DEFM  'MATH ERROR(S)'
            DEFB  0

MYLAST      EQU   $       ;END OF MY CODE

END         MAIN
```



Next time: the Line Printer and FOR,NEXT loops.


ACTUAL REALITY
TRS-80 MODEL I/III/IV
DREAMLAND SOFTWARE


ROCK STAR!
TRS-80 MODEL I/III/IV
DREAMLAND SOFTWARE



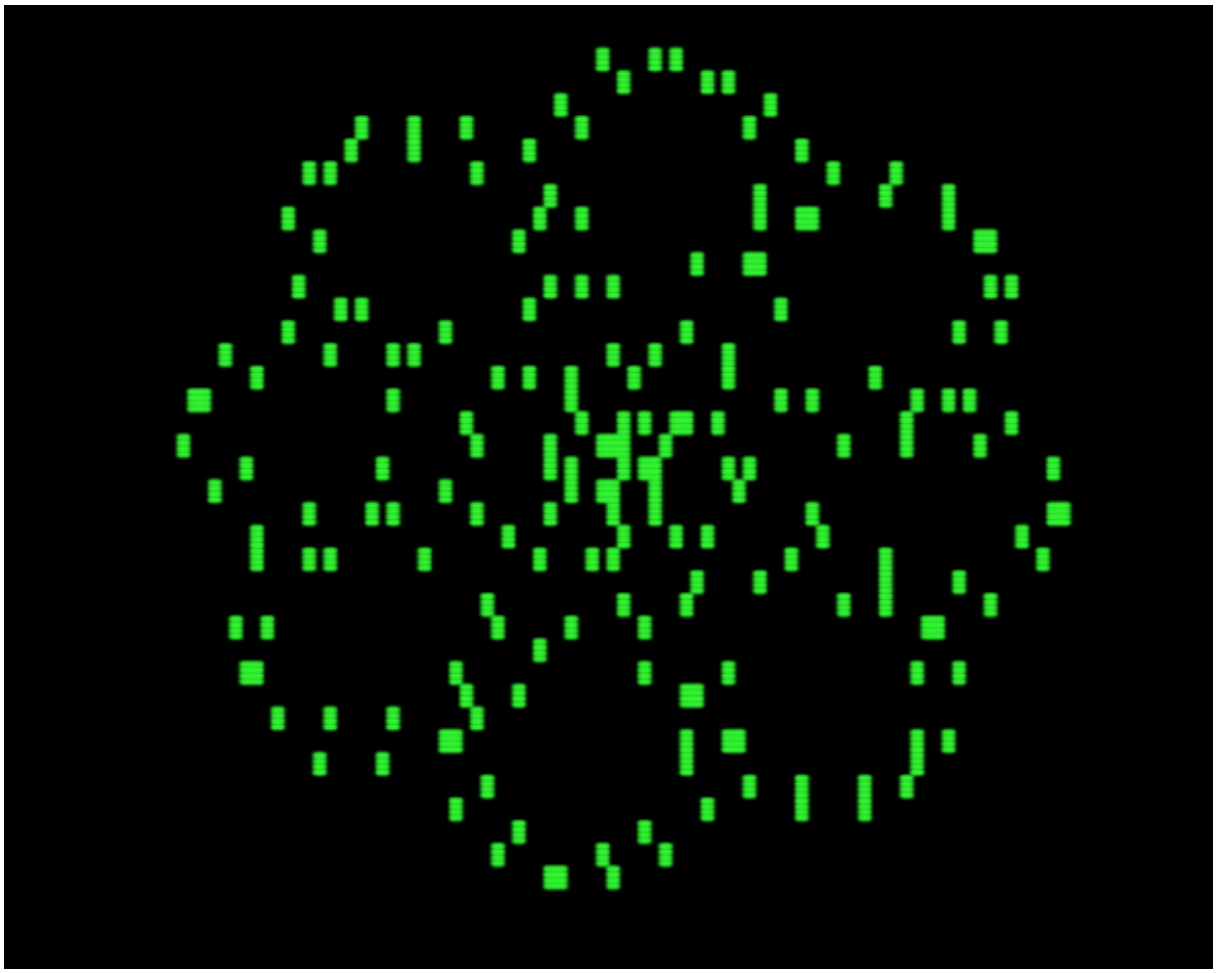
SNOWFLAKE SCREEN SAVER

E.T. FONEHUME

HI DUSTY.
WISHING YOU AND DEE A VERY MERRY CHRISTMAS AND A HAPPY NEW YEAR. JUST TO GET EVERYONE IN A TRUE FESTIVE MOOD, THIS ONE-LINER IS A SNOWFLAKE (LIKE!) SCREEN SAVER. I KNOW, MODEL 1 GRAPHICS CAN'T QUITE EMULATE HI-RES FRACTAL IMAGES, BUT IT'S THE NEXT BEST THING! IT HAS A REASONABLE SPEED, BUT IF YOU'RE USING TRS80GP, SWITCH ON THE 'TURBO' OPTION.

THANKS FOR ALL YOUR HELP OVER THE LAST YEAR. I'M LOOKING FORWARD TO THE START OF OUR 20TH YEAR!

```
0 RANDOM:CLS:C=RND(300):FORA=0TO6.28STEP.03:R=6*SIN(C*A):X=R*7*C
OS(A)+64.5:Y=R*3*SIN(A)+24.5:SET(X,Y):NEXT:FORZ=1TO2000:NEXT:RUN
```



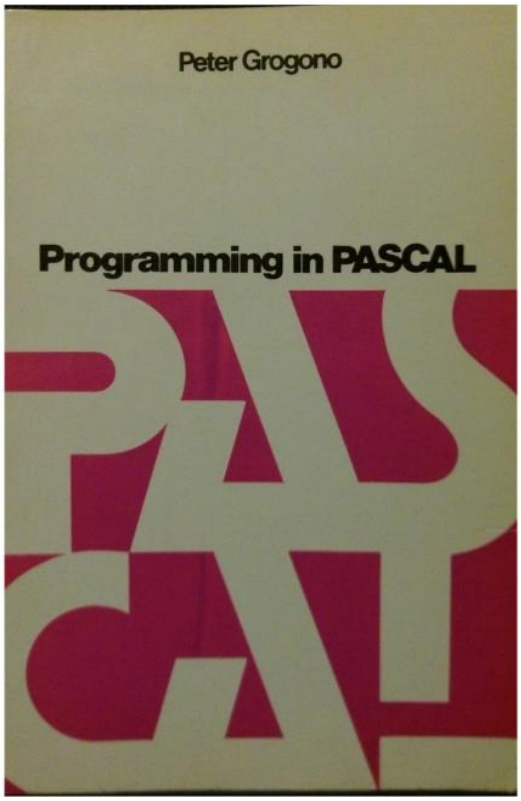
Programming Languages for the Model I

Early Pascal Compilers

Vince Otten

It's 1980, I've upgraded my TRS-80 to 16K bytes, and I've written and sold not only demo programs but a store inventory application (using cassettes!) to local Radio Shack dealers. Now I'm encouraging them to refer potential buyers to me for custom programming. I want to turn my hobby into a *business.*

The dealers aren't flooding my inbox with requests, so I turn to a name I respect more than any other for quality machines: Hewlett-Packard. There's a small branch office in London, Ontario— just there to support their medical instrumentation—and I have the nerve to just drop in and ask if they're interested in taking me on as a contract or support programmer. When one kindly executive takes the time to ask what language I code in, I have to admit that it's BASIC. He pulls out a book by Peter Grogono and boldly says, "*This* is the language everyone's going to." He's actually good enough to lend me his own copy.



Back in those days we didn't have any easy, inexpensive way to copy pages of interest: if I remember correctly, copies were $0.15 a sheet at the local library. So I took handwritten notes of *everything.* It took hours, but I sure learned what this amazing new way of programming was all about.

“Now, if only I could get a Pascal compiler for my TRS-80,” I thought. In those days you couldn’t just Google “Pascal compiler for TRS-80” and do a quick download. The World Wide Web wouldn’t even come around for another 9 years! But to my delight, I found a product from *Supersoft* in a computer store in Toronto. This compiler would become a Radio Shack product, only available in Canadian stores a year or two later:



This wasn’t “real” Pascal—there were some significant restrictions.

- Only integers and 1-dimensional integer arrays
- The only I/O devices are the keyboard and the screen (no printer, no cassette, no disk files)

So this was never going to be the tool I could use to develop business applications. On the other hand, you could do TRS-80 graphics, and the package promised that they would be *fast!* If I recall correctly, they claimed a 3-5x increase in speed over BASIC. Was that true? When I got the cassette home and loaded in the sample programs, they were indeed quicker than anything I’d been able to write.

Recently, I found a disk version of the Tandy product, and decided to “race” it against TRS-80 BASIC. I chose as my demo an implementation of “Bresenham’s Algorithm”, a method of drawing angled lines. My example simply draws a line from the top left to the bottom right of the screen. Experienced Pascal (or Delphi) programmers will quickly notice some differences between this code and the standard, such as no “Program *Name*(input, output)” statement. But this is what my Tiny Pascal source looks like:

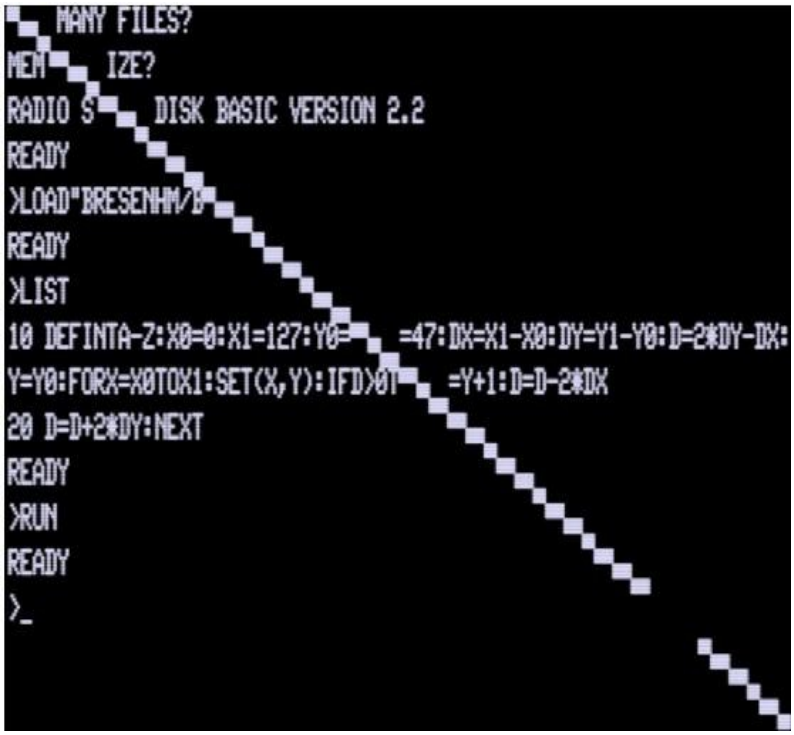
```
(* LINE DRAWING SAMPLE USING BRESENHAM'S ALGORITHM *)
```

```
VAR X, X0, X1, Y, Y0, Y1, D, DX, DY: INTEGER;
BEGIN
X0:= 0;
X1:= 127;
Y0:= 0;
Y1:= 47;
DX:= X1 – X0;
DY:= Y1 – Y0;
Y:= Y0;
FOR X:= X0 TO X1 DO BEGIN
    PLOT( X, Y, 1);
    IF D > 0 THEN BEGIN
        Y:= Y + 1;
        D:= D – 2 * DX;
        END;
    D:= D + 2 * DY;
    END;
END.
```

The same algorithm can be expressed this way in Level II BASIC.

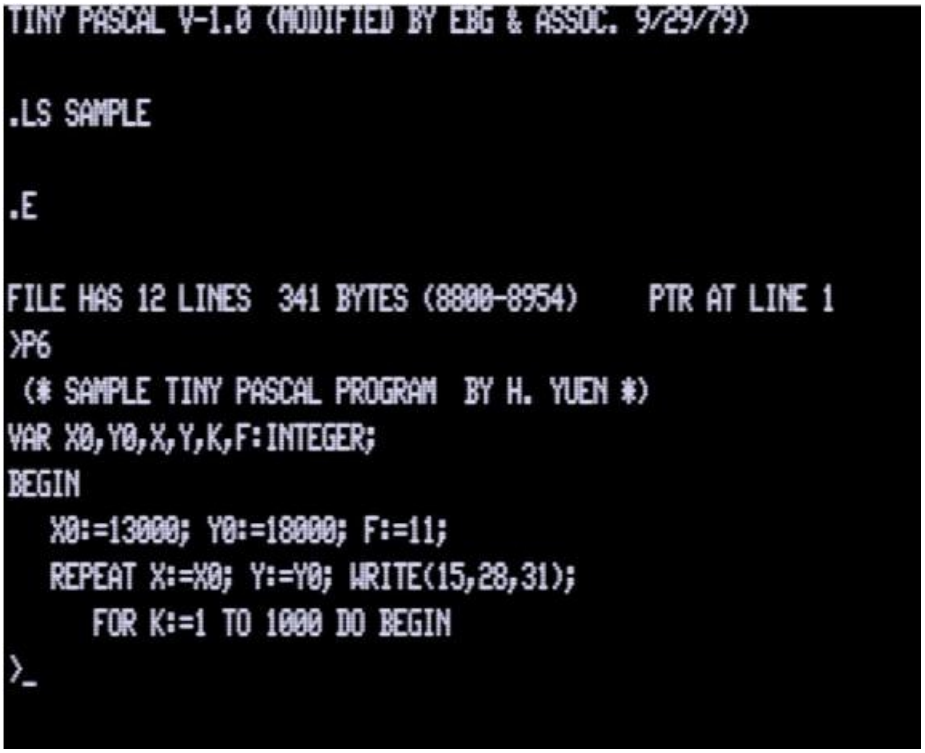
```
10 DEFINT A-Z: X0= 0: X1= 127: Y0= 0: Y1= 47: DX= X1 - X0: DY= Y1 - Y0: D= 2 * DY - DX: Y= Y0: FOR X= X0 TO
X1: SET( X, Y): IF D > 0 THEN Y= Y + 1: D= D - 2 * DX
20 D= D + 2 * DY: NEXT
```

I actually wrote the BASIC code without any spaces to speed it up a little, but it did fit into these two lines, again for purposes of speed. Here's what it looks like running:



How long did that take to draw in BASIC? Approximately 3 seconds. The Tiny Pascal interpreter ran its "P-code" version in under 1 second (see below for just *how* fast it was).

So—what's it like programming with Tiny Pascal? Actually, it's kind of fun, at least for me. I seem to enjoy the restrictions of some of these old-time programming environments, like Microsoft Editor Assembler Plus, Level II BASIC, and TP (but not all of them, as you'll see later). Admittedly, I'm constantly jumping out to Notepad to document my work, so it's not exactly working in the coal mines. When you invoke PASCAL, you enter the TP "Monitor", where you can load and save files, enter the Editor, Compile your source, or Run your compiled P-code. Here's a sample screen:



At the Monitor's "dot" prompt, I asked it to "Load Source" SAMPLE, which caused it to actually look for a file named SAMPLE/TPS. Then I entered the Editor, which provides a ">" prompt, and asked it to print the first 6 lines.

I can return from the Editor to the Monitor by pressing <BREAK>. At that point, entering the "C" command will compile this source code. It's still fun after all these years watching the compiler cursor through the code. After it's compiled, entering the "R"un command will execute the compiled P-code. (P-code is a mini-language of single-byte instructions that a "P-code interpreter" then executes; so Tiny Pascal does *not* produce actually Z80 machine code.) You'll notice that this is one of several versions of TP adapted for TRSDOS, in this case by "EBG & ASSOC". Some trivia about the modification: this TP automatically appends the "/TPS" extension when you ask to to "Load Source" or "Write Source". It won't allow you to add a drive number. How do I save my code to a drive other than :0 ? I first copy something like

```
COPY SAMPLE/TPS MYCODE/TPS:2
```

Then I fire up PASCAL, "Load Source" LS MYCODE, edit it to my liking, and then "Write Source" WS MYCODE. A bit of a roundabout, but it works. By the way, this version doesn't seem to let me write totally new source code, only edit a file that I've loaded in.

So—what's Tiny Pascal good for? It's *not* good for taking a text like Grogono's or the popular *Oh, Pascal* that came out a bit later, following along, and learning all about the language, like I did with APL (see the "APL-80" article in the previous issue of TRS8Bit). Tiny Pascal is just a "kind of Pascal". It doesn't support any of the data structures that you can build with *real* Pascal, like

```
type FlightData= record
                    FlightNumber: integer;
                    DepartureTime, ArrivalTime: real;
                    Terminal: char;
                    GateNumber: integer;
                    OnTime, Cancelled, SoldOut: boolean
                end;
var Phoenix, StLouis, Denver, Miami, Buffalo: FlightData;
```

This was the kind of new and useful stuff that excited me, coming from BASIC, and I couldn't even use it! So again, what's TP good for anyway?

Graphics. Fun. Accessing machine-level code. TP gives you a byte array called MEM() that allows you to PEEK and POKE the whole of memory:

```
A:= MEM(24467) + 3;        (* READ FROM MEMORY *)
MEM(T):= 0;                (* WRITE TO MEMORY *)
```

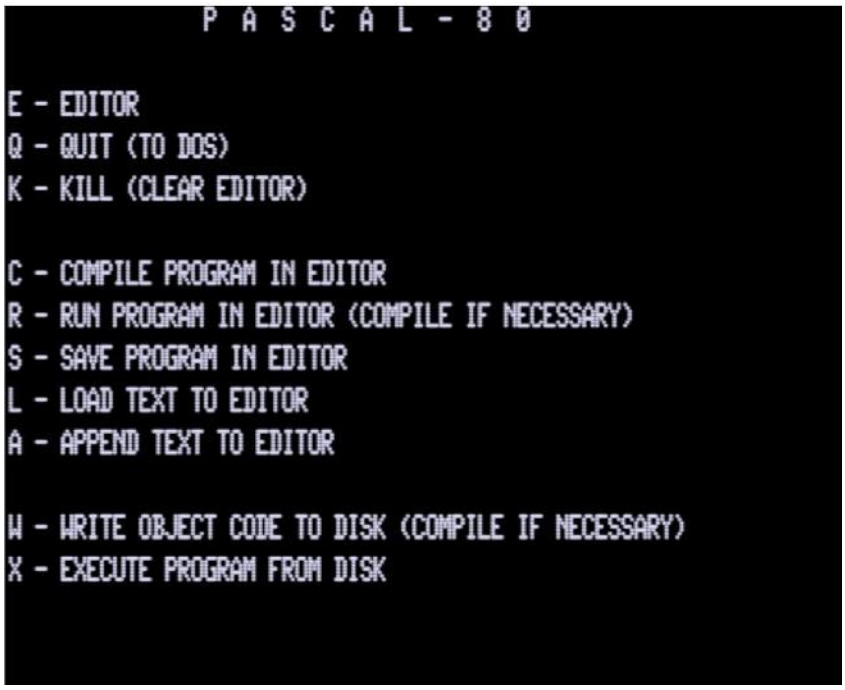
as well as a complimentary 16-bit array "MEMW()". You call also CALL a machine language subroutine.

Here's what really excited me about TP back then: it came with the entire source code for its compiler, written in Tiny Pascal! I knew nothing about compilers back then (and precious little now), so this fascinated me, even if I couldn't understand the actual source. Here's what a bit of the program looked like:

```
PROC INIT;
    VAR PADDR: INTEGER;
    BEGIN
        RDPTR:= MEMW(SYCB) - 1;
        MADDR:= MEMW(SYCB + 4);
        KADDR:= MADDR;
        GENP:= MEM(SYCB + 5) <> 0; (* SET GEN CODE FLAG *)
        RWDTBL:= MEMW(SYCB + 10) - %0120; (* ADDR OF COMPC – 18 REC *)
```

You get the point. A lot of data structure stuff is implemented in those MEMW() and MEM() arrays. It's not like I'll quickly understand a recursive-descent compiler from this code! Where did Tiny Pascal come from? According to the late lamented Matthew Reed's thorough history on the trs-80.org website, it was first introduced in *Byte* magazine in September of 1978 (following *Byte*'s regular August "programming language" issue—this time covering Pascal, of course. That original compiler was written in North Star BASIC! This concept of a "Tiny" Pascal followed some of the earliest language work for microcomputers, beginning with "Tiny BASIC". (Level I BASIC for the TRS-80 was adapted from "Tiny BASIC" by Steve Leininger, the man who designed the entire Model I.) A couple of years later we have another "Tiny" language, "Small C", introduced in the May 1980 issue *Dr. Dobb's Journal.*

So if Tiny Pascal was more of a "toy" language, what would be a good compiler to learn "true" Pascal with? As well, I wanted something that could produce machine-language "/CMD" files so business users could have the simplest experience possible. Could I provide a central "menu" program that would run as the AUTO command in TRSDOS, so the user could just insert the main floppy and press the Reset button, automatically entering their <BREAK>-key immune application?



Once again, Matthew Reed provided us a detailed history of "PASCAL-80" at trs-80.org. This compiler went through several revisions. I can't recall which version I used back in the early 1980s, but for this review I downloaded the "New Classics Software" disk image from classic-computers.org.nz, a website by Terry Stewart dedicated to the Dick Smith System-80. This package was introduced in Spring of 1982.



NEW CLASSICS SOFTWARE

239 FOX HILL ROAD · DENVILLE NEW JERSEY 07834 · TELEPHONE 201-625-8838

PASCAL 80 by Phelps Gates

"If anyone could devise a good PASCAL system for the TRS-80® it was Phelps Gates. I am happy to report he has done just that."
BYTE, Dec. 1981, pg. 304

"After trying out...(the competition) I found myself using PASCAL 80 exclusively." Creative Computing, Nov. 1981, pg. 96

PASCAL 80 is the friendliest PASCAL available anywhere! Monitor, editor, and compiler are in memory at the same time, avoiding time consuming and annoying disk access and disk switching. Yet, it uses standard PASCAL syntax and leaves 23K of work space in 48K (32K at run time).

PASCAL 80 has 14 digit accuracy and requires a 48K TRS-80® with one disk drive. It comes in a binder with easy and complete user instructions for $99. Add $2 shipping and handling per order and specify Model I or Model III.

Credit card orders: (201) 625-8838

NEW CLASSICS SOFTWARE
239 Fox Hill Road
Denville, NJ 07834

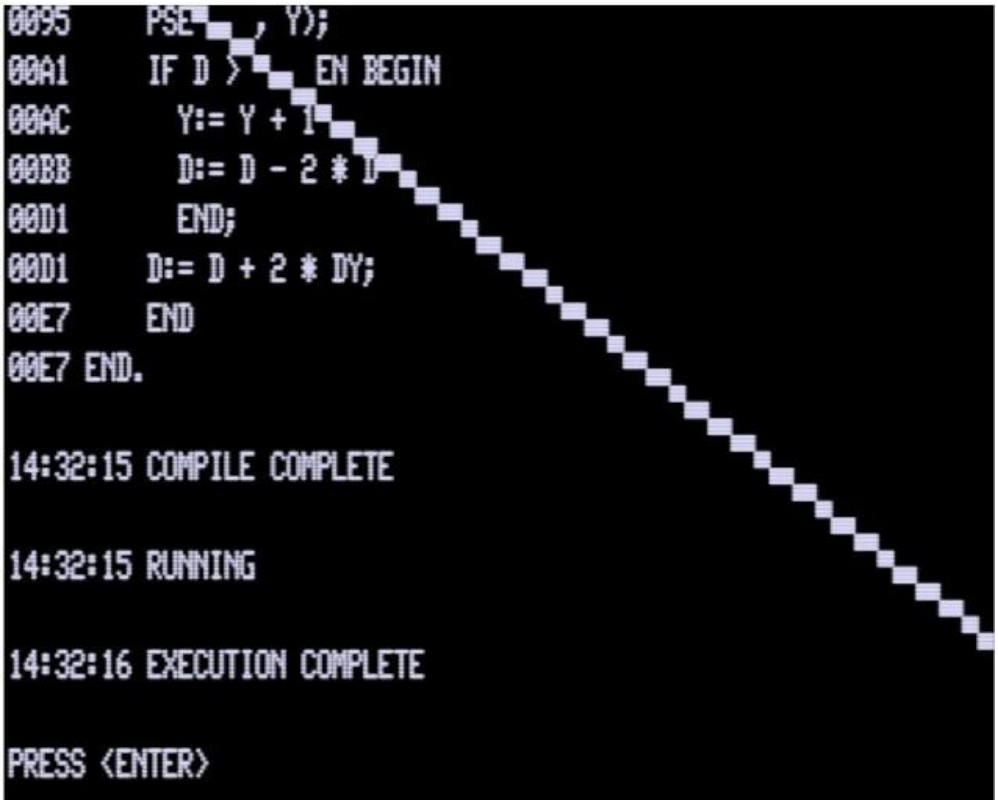
(PASCAL 80 does not implement variant records, pointer and window variables, or functions and procedures used as parameters.)

Yes—that Phelps Gates. The man who wrote the remarkable APL-80 language package for the TRS-80 Model I that I reviewed in the last issue of TRS8BIT. Was this compiler for Pascal as complete as his APL implementation? Here' what Matthew Reed said:

'...Pascal 80 was described by the manual as "basically Standard Pascal, with some restrictions and extensions." Those restrictions meant that "pointer Variables, Variant Records, NEW, DISPOSE, WITH, GET, and PUT are not implemented in PASCAL 80."' (These restrictions essentially meant "no Heap (dynamic) storage".) Could Pascal 80 be a good platform to learn the language? Absolutely!

Extensions included graphics; Random Disk Access; Peek(), Poke(), and Call(); and the ability to create /CMD programs and to "chain" from one Pascal program to another. This sounded like something I could use to create fast turnkey business applications!

How fast was it? Here' the "Bresenham" algorithm I implemented for Tiny Pascal above.



That's quick, too. I ended up modifying the Tiny Pascal and Pascal-80 versions to loop 100 times so I could more accurately measure how long each run of the algorithm took. Here's my results:

Language	*Run Time / 1 iteration*	*Improvement over BASIC*
BASIC	—3.00 seconds	
Pascal 80	0.76 seconds	—3.9x faster than BASIC
Tiny Pascal	0.65 seconds	—4.6x faster than BASIC

Pascal 80's graphics speed is probably quick enough for some games, just like Tiny Pascal. But what about the "business end of things"? How fast was its floating-point math? If you've read my review of Phelps Gates' unique APL-80 in the previous issue of TRS8Bit, you'll know that he liked to implement it as double-precision 8-byte math. In Pascal 80, Double Precision is the default for *real* variables, and though he provides *real6* as a memory-saving 4-byte alternative, all math operations are still performed at the Double Precision level.

You may remember how I tested APL-80's speed by filling a 3190-element array and performing arithmetic on that array to calculate an approximation of "pi". I then also created a BASIC program that duplicated the APL code and test-ran that. As well, I tested a Fortran-80 version. I discovered that even though APL is an *interpreted* language, it was the fastest! Here, I've again implemented a version in Pascal 80.

```
(* PI/PAS -- CALCULATE PI FOR SPEED DEMONSTRATIONS *)
PROGRAM PI( INPUT, OUTPUT);

CONST SIZE= 3190;
TYPE INVERS= ARRAY(.1..SIZE.) OF REAL;
VAR I: INTEGER;
A: INVERS;
TOTAL: REAL;
FLAG: BOOLEAN;

BEGIN
(* INITIALIZE INVERS() WITH #S 1..SIZE *)
FOR I:= 1 TO SIZE DO
A(.I.):= I;
(* MAKE A() EVERY EVEN # *)
FOR I:= 1 TO SIZE DO
A(.I.):= A(.I.) * 2;
(* MAKE A() EVERY ODD # *)
FOR I:= 1 TO SIZE DO
A(.I.):= A(.I.) - 1;
(* MAKE A() '1/A()' INVERSE *)
FOR I:= 1 TO SIZE DO
A(.I.):= 1 / A(.I.);
(* GET READY TO TOTAL UP ALL OF A() *)
TOTAL:= A(.1.);
FLAG:= FALSE;
FOR I:= 2 TO SIZE DO BEGIN
IF FLAG THEN
   TOTAL:= TOTAL + A(.I.)
ELSE
   TOTAL:= TOTAL - A(.I.);
FLAG:= NOT( FLAG)
END;
TOTAL:= TOTAL * 4;
WRITELN( TOTAL)
END.
```

Here are the test results.

Language	*Run Time*
APL-80	3:11
Fortran-80	3:11
Pascal 80	4:55
Model I Disk BASIC	5:47

(The array was large enough—3190 elements x 8 bytes each—that I couldn't just enter the source code in the Editor, Compile it, and Run it: I had to (W)rite Object Code to Disk first, then (X)ecute it From Disk, a procedure which deletes the Editor and Compiler from the IDE in order to make room for the code and data.)

If my Pascal program is processing a lot of math—creating a Mortgage Amortization Table, for example—it's not going to run all that much faster than BASIC. Happily, most business programs are I/O intensive rather than math-intensive. It's probably still useful for creating applications, right?

Well.. I haven't been able to create standalone /CMD programs with Pascal 80, to start with. There's a detailed set of instructions in the manual supplemented with on-screen prompts, and although the process does work, the final /CMD program seems to just load and then abort back to TRSDOS without actually running. I've tried various versions of Pascal 80 with the same sad results.

Then there's "program chaining", the concept of running one Pascal program from another. The manual describes a procedure called *Chain()*, claiming that it exists in a file on the Pascal 80 disk called CHAIN/SRC. I haven't been able to find a copy of CHAIN/SRC in any of the sources for Pascal 80, but the manual does provide a listing, so I tried entering that and Compiling it. The compiler doesn't seem to like the source code, so after several retries I finally gave up. I expect that, given enough research, I could make this work, but given the problems with creating /CMD files that would actually run, I have to conclude that Pascal 80 wouldn't be a useful "production compiler".

Oh, and another gripe: the Editor. It looks nice: it's one of the earliest "full screen" editors available. Nice—except for the fact that it's always in "overwrite" mode. In other words, unlike Notepad or Microsoft Word or even Scripsit, you can't just *insert* text. You either have to write all over the line that you've entered, or use a [Ctrl][O] editor function to "open up" a *single* character space for your new text. Yes: you'll have to repeat [Ctrl][O] as many times as there are characters in the text you want to insert. And [Ctrl][O] is actually [Shift][DownArrow][O], all pressed simultaneously. I got used to this, somewhat, but I can't see writing a complete application with it.

On the other hand, Pascal 80 serves very well as a tool to learn the language. Its implementation is close to complete. It's a fast way to explore the example code in a book like *Oh, Pascal*. And for the time, it was yet again a remarkable language implementation by the one and only Phelps Gates.

We've been following the Programming Language series that was presented in each August by *Byte* magazine: APL in 1977, and Pascal in 1978. What did they cover in 1979? Lisp! Were there any implementations of that language for the Model I? Yes, there were, and I hope to cover that next time.

Thirsting for adventure?
Brass lantern getting dusty?

UPDATED 2025!



Take your Model I/III/4/4p on exciting new adventures never available until now....

M4ZVM enables your TRS-80 Model I, III or 4 to play Infocom and post-Infocom adventures, taking full advantage of your hardware. Supports old favourites like Zork and Seastalker (including Sonarscope), as well as new-to-Model-4 games including Trinity, Border Zone and Sherlock, plus hundreds more available in the Interactive Fiction archive.



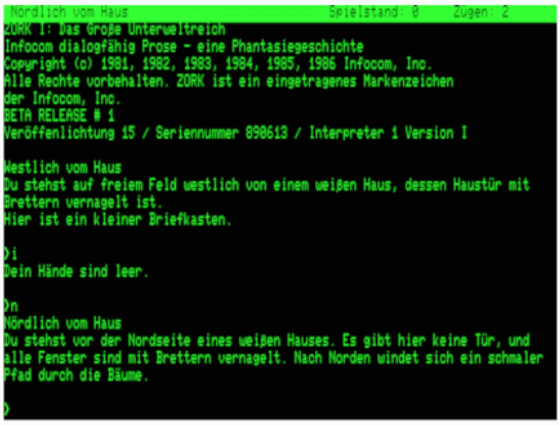
M4ZVM & M3ZVM with sample disks and game compatibility lists are at:
https://sijnstra.itch.io/m4zvm

Source and links to other versions at:
https://gitlab.com/sijnstra1/m4zvm

Actual Screenshots of games new to the model 4, running M4ZVM (on trs80gp) - check links for how-tos & disks



Beyond Zork



Zork 1 (German)

M4ZVM/M3ZVM/M1ZVM features include:
* z1-z8 game execution (z6 support is experimental)
* named game save/load
* accented characters where available
* Optionally set the "Tandy bit" for Tandy customization!

UPDATED 2025 version - faster, better, more bugfixes!

NEWS FROM

Liberté • Égalité • Fraternité

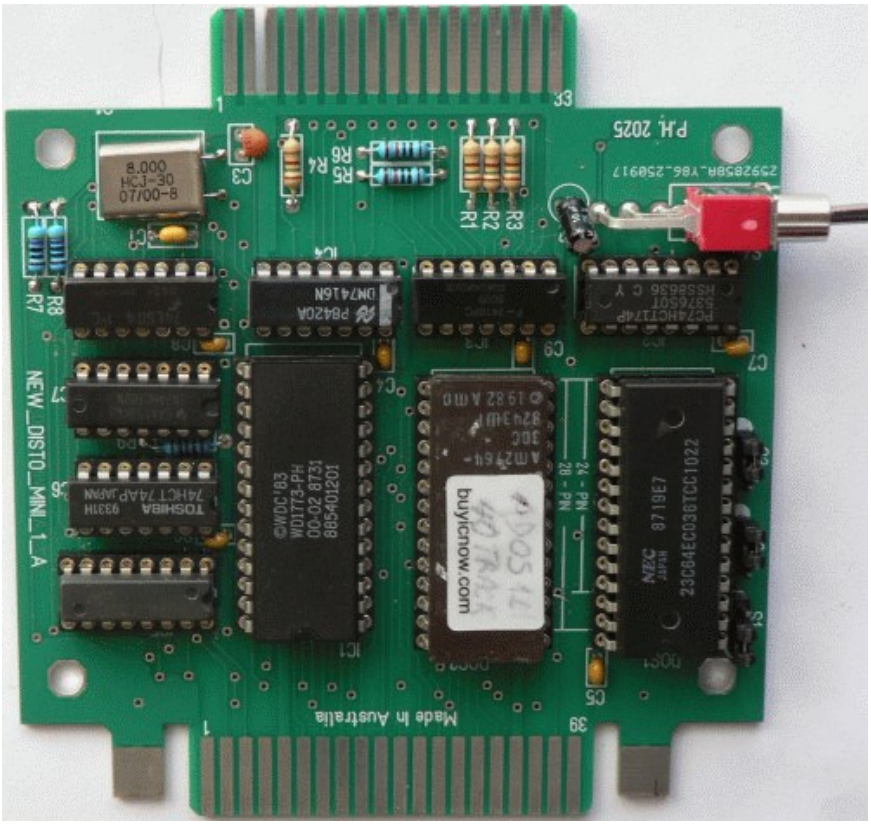
PASCAL HOLDRY

NEW_DISTO_MINI_1

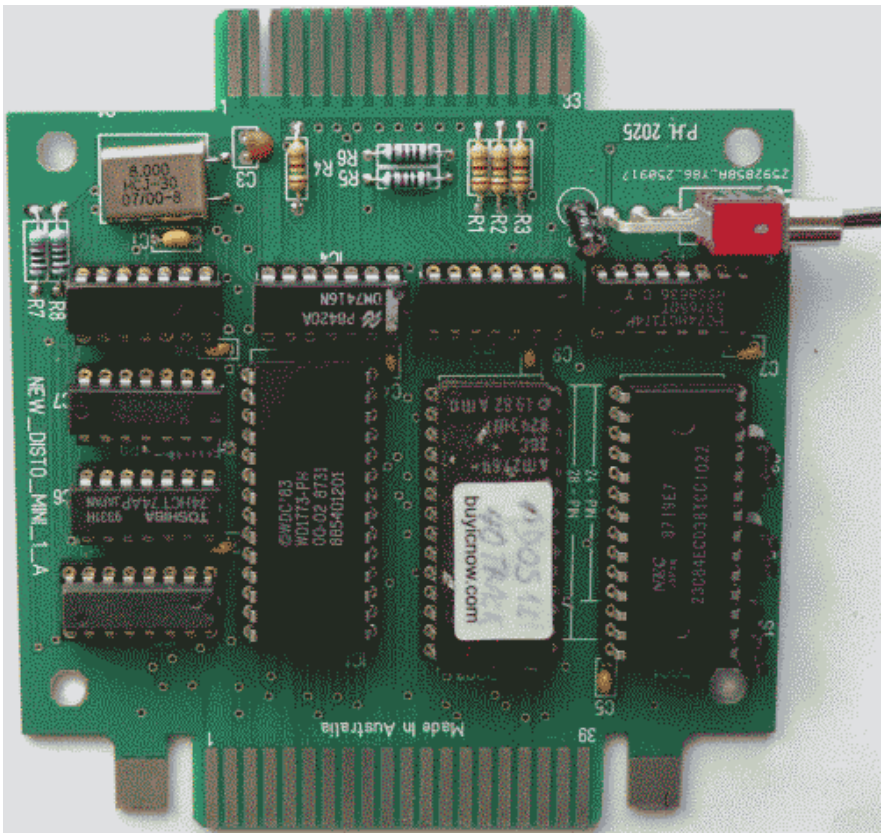
Aim:

1. Offering the possibility of working with two different "DOS" on a TRS-80 Color
2. To be able to recover existing software
3. Keeping a record of our digital heritage
4. Provide a diagram for troubleshooting and understanding how the card works.
5. The "A" version card allows a standard exchange with the original "DISTO" card and fits into the "DISTO" case with a slight cut in the case for the passage of the switch.
6. The "B" version card allows for the exchange of the "TANDY" card and fits into an FD502 type case and/or a 3D printed case (See on the Internet)

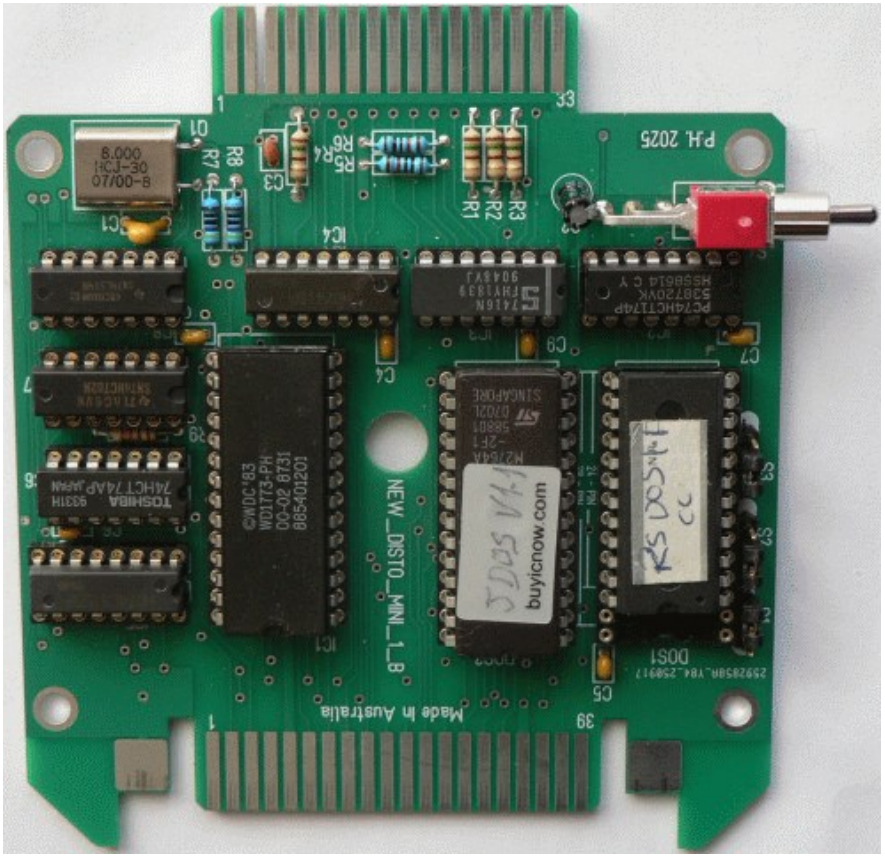
Top view of version "A" with a 28-pin PROM



Top view of version "B" with a 28-pin EPROM



Top view of version "B" with a 24-pin "TANDY" PROM



DISTO Super Controller

Written by Pascal HOLDRY

Aim:

1. To be able to recover existing software
2. This type of card is sensitive to oxidation due to the casing; my prototype fits in place of the faulty card.
3. Keeping a record of our digital heritage
4. Provide a diagram for troubleshooting and understanding how the card works.

FEATURES:

I. *4 x 28-pin sockets for software scalability.*
II. *Uses 2764 or 27128 EPROMs.*
III. *EPROMs are software selectable.*
IV. *Internal mini-extension interface for*

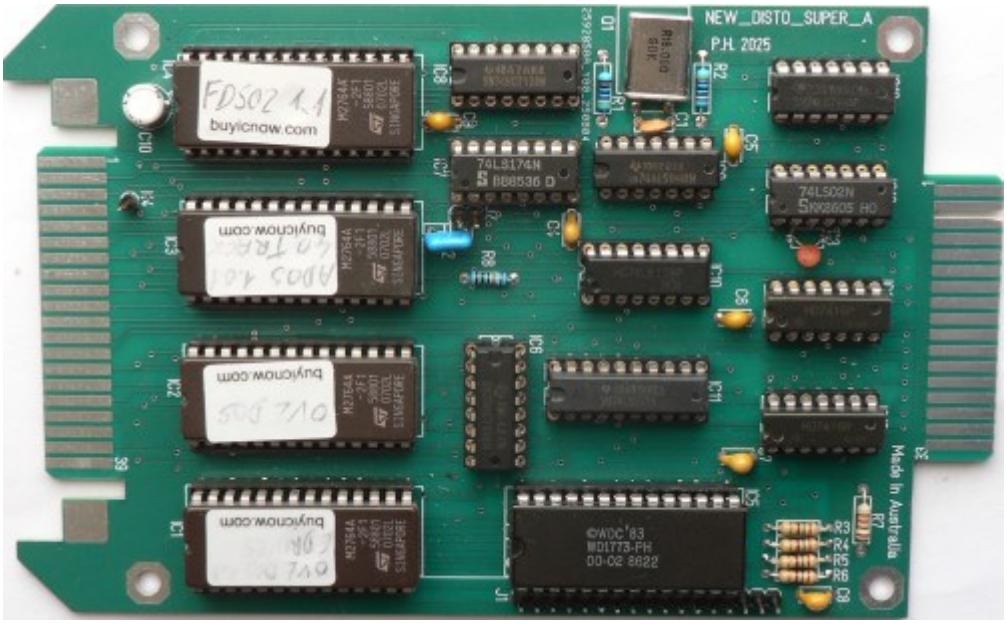
1. *Real-time clock and/or*
2. *Parallel printer or EPROM programmer or*
3. *Display of 80 * 24 characters or*
4. *User projects . .*
5. *Fully compatible with Radio Shack.*
6. *New technology, no adjustments needed.*
7. *High-speed 16 MHz master clock with high precision.*

Requires only +5 Volts, works on all COCO or COCO II models. (This board has not been tested on a COCO III by me)

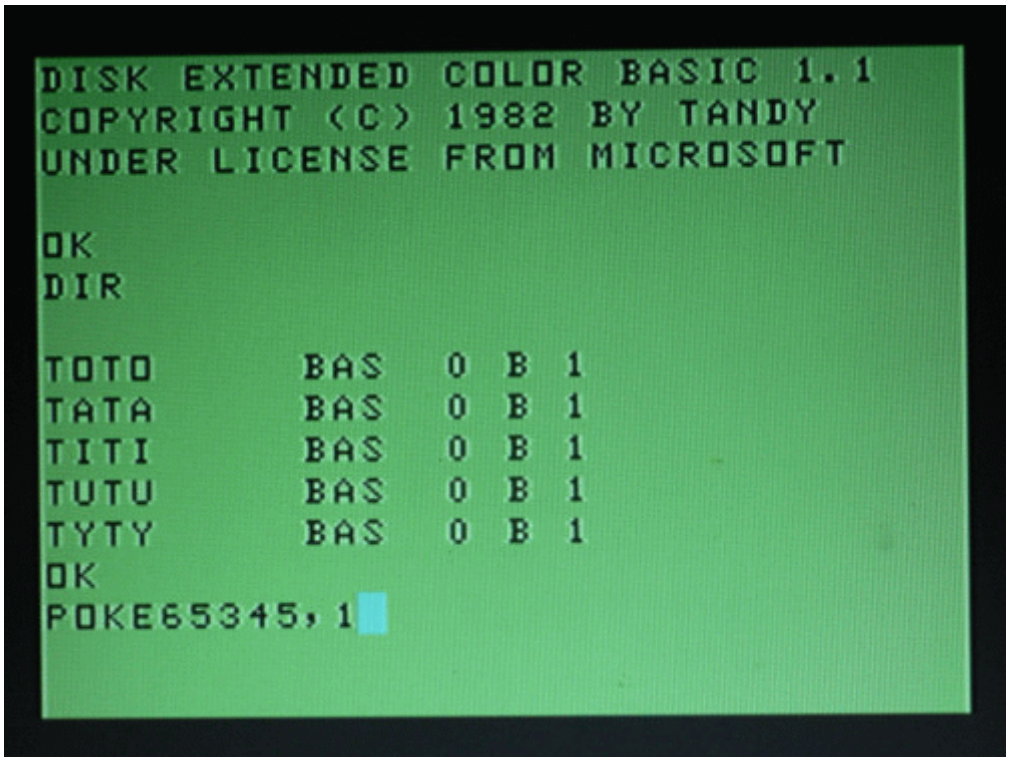
DESCRIPTION:

The main features (aside from the fact that it's a floppy disk controller) of this controller are the additional IC sockets included inside for software expansion and the mini expansion bus (MEB for short), used to add optional peripherals. First, this controller has four—yes, four 28-pin IC sockets—allowing you, the user, to install any DOS (disk operating system) or other software and utilities you want.

This gives you a total of 64KB of user EPROM memory at your disposal. You can have one to four DOS systems present inside your controller and switch between them with a simple POKE. Second, the "MEB" is an internal 17-pin expansion bus. Its primary use is for our optional real-time clock and/or the Centronics-compatible parallel printer adapter. Other options are possible.



Top view



When the COCO is powered on, you will automatically be taken to the S1 component support (here, DOS RS V1.1).



LEVEL 2.5 (TWOHAF) BASIC ALLOWS CASSETTE AND ACULAB FLOPPY TAPE USERS MANY OF THE TANDY DISK BASIC FEATURES, NAMELY -

"&", "DEF FN", "DEF USR", "MID$=", "INSTR", "MKD$", "MKI$", "MKS$", "CVD", "CVI", "CVS"

AVAILABLE, AS A **FREE DOWNLOAD**, FROM

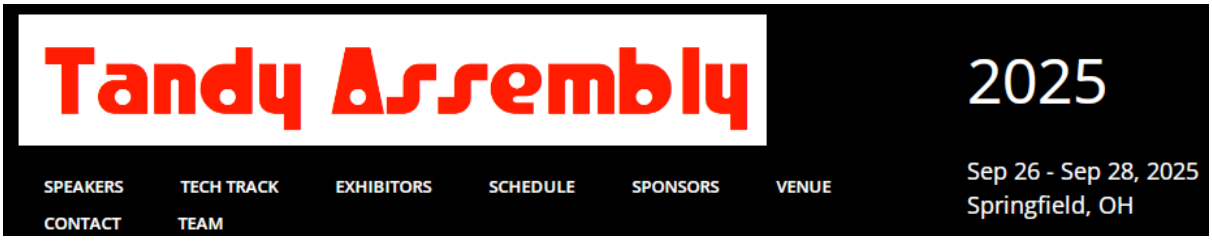
WWW.TRS-80.ORG.UK


TANDY LAND
The sweet side of serious computing.
Ian Maurie
MOLASSES SWAMP
LOLLYPOP WOODS
93 MILES
70 MILES
GINGERBREAD PLUM TREE
PEPPERMINT STICK FOREST
START

MAV'S TANDY ASSEMBLY QUARTERLY COLUMN

TA25 NEWS

by Ian Mavric



Bringing you the latest news you need to know about the greatest Tandy TRS-80 show on the planet. This time I talk about the 2025 show which has just finished.

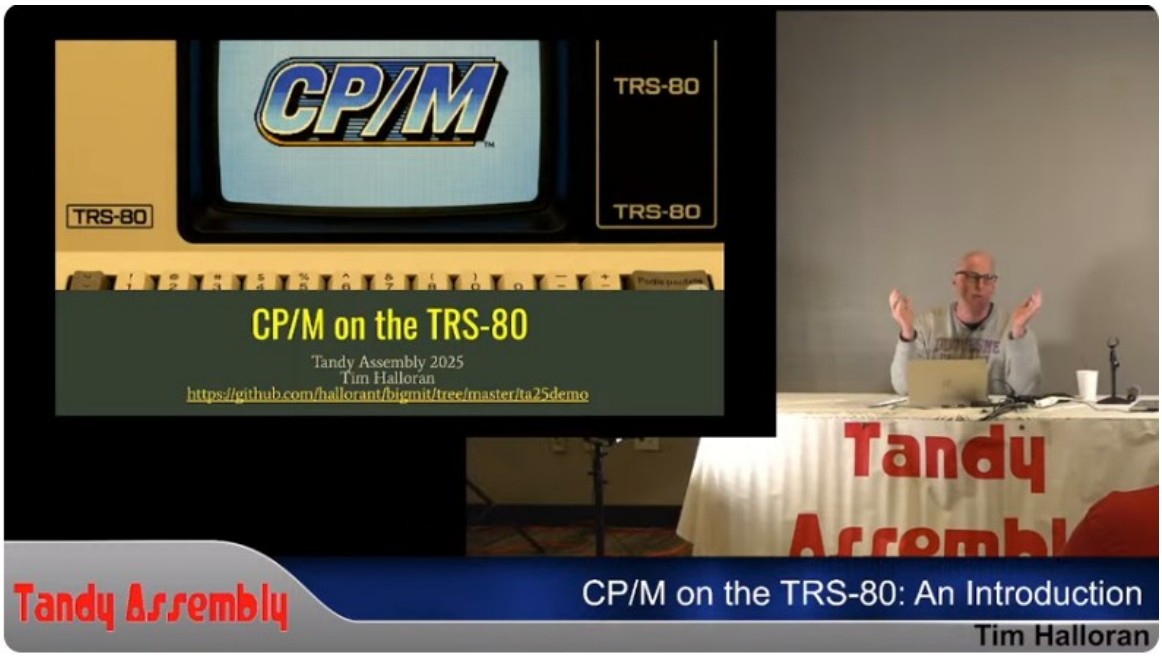
THE 2025 SHOW HAS BEEN AND GONE - LET'S LOOK AT THE HIGHLIGHTS



The year of the cake! Last issue I mentioned that Andrea Grimwood-Zaleski would be making a Tandy 6000 cake, 1:1 size with the original to commemorate 40 years since the Tandy 6000 was released. The cake, pictured above, was enormous and tasted very nice! Next year, we commemorate 40 years of the Tandy Color Computer 3, which is make producing a cake much more compact. Can't wait to see it!

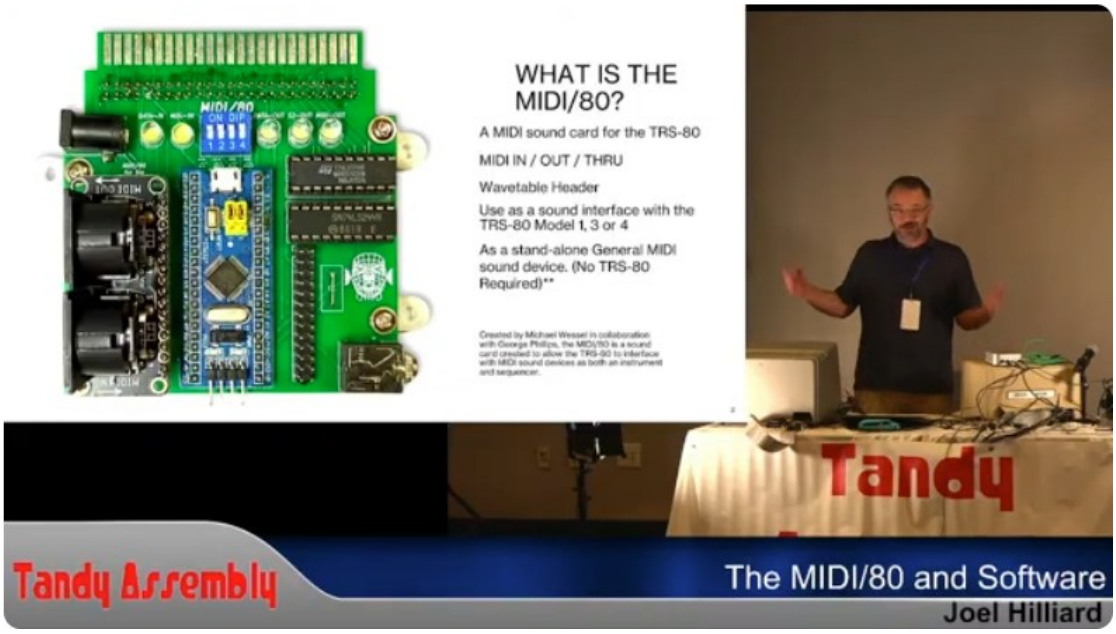
Friday started with the usual Tandy Tech Track sessions and there was much to be learned during these popular tutorial presentations. If you missed them I have included links to each of the talks.

TTT01: Tim Halloran: CP/M on the TRS-80: An Introduction



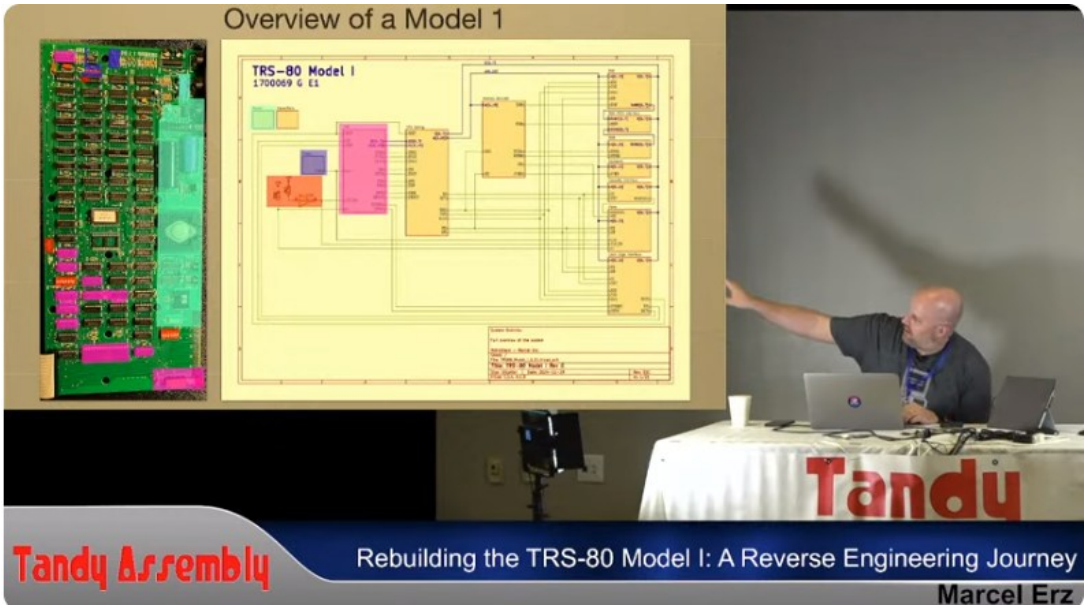
https://www.youtube.com/watch?v=ZAl5JfZP_HA

TTT02: Joel Hilliard: The MIDI/80 and Software



https://www.youtube.com/watch?v=VoPTrOpMQ2E

TTT03: Marcel Erz: Rebuilding the TRS-80 Model I: A Reverse Engineering Journey



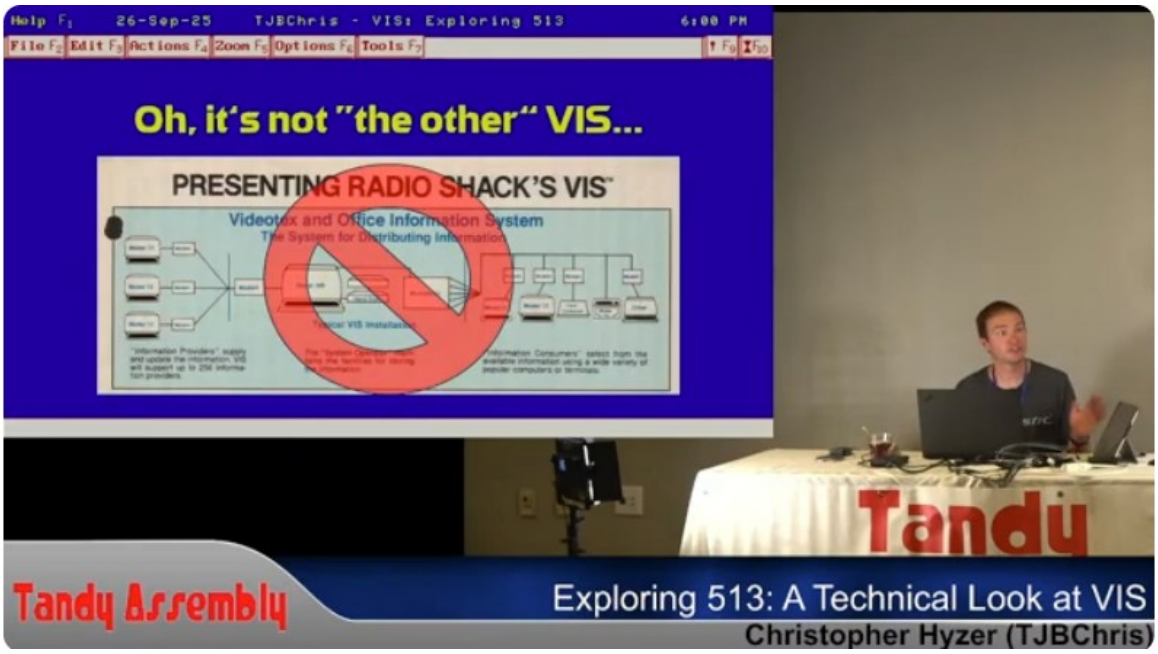
https://www.youtube.com/watch?v=cmATnqcOjRw

TTT04: Jeff Birt: Building Modern Accessories for Vintage Computers for Fun and Profit.



https://www.youtube.com/watch?v=cWf6vp9nqkE

TTT05: Chris Hyzer: Exploring 513: A Technical Look at VIS



https://www.youtube.com/watch?v=3yNZ0pIqWrs

SATURDAY: TA25 BEGINS IN EARNEST WITH KEYNOTE SPEAKERS

A full audience was in attendance for the main keynote speaker, Joel Hilliard who gave a live performance of his musical project called Downpoly which utilises TRS-80s with Orchestra-90s as sound producers.
These sounds are then mixed and modified with a selection of effects pedals and MIDI interface to produce a full bodied sound which comes over as musically pleasing, instead of the mechanical synth sound typically made by the Orch-90.



Photo: Joel getting ready to perform songs from his debut LP 'Choice Cuts'

KEYNOTE SPEAKER: Joel Hilliard: Downpoly



https://www.youtube.com/watch?v=QNZUELyjySw

2nd SPEAKER: Jim Brain: Beyond 512kB: Enhancing the CoCo Memory Management Unit



At the time of writing the video is not currently available on YouTube. If it comes available in the future I'll post an update with a link. It's always good to have Coco content at TA and we were happy to have Jim, the president of Glenside Color Computer Club.

We then moved onto the entertainment part of the show. The auction of vintage TRS-80 items which generates funds for next years Tandy Assembly event was a success and fun to watch. It was followed by the TA trivia game which, like previous years, involves the audience playing a pub-trivia style simplified TRS-80 game.

SATURDAY AFTERNOON: AUCTION



https://www.youtube.com/watch?v=owCo0bVYpn4

SATURDAY NIGHT: TRIVIA GAME

This year I expanded the trivia game to accomplish something I have been meaning to achieve for years, which is getting some women involved.

I lined up Morgan Donahe whos expert topic was the MC-10, Wendy Hilliard who's expert topic was Downpoly, and Sandy Weimar who's expert topic was Cloud-9 Tech. A short 4 round game was organised and all three ladies told me they had a great time. You'll need to watch it to see who won :-)



Photo: L-R Sandy, Wendy and Morgan, in the TA25 ladies trivia challenge

After the ladies we went onto the main trivia game which was eight rounds and the experts were Marcel Erz, Jeff Birt, and Joel Hilliard. their expert topics being the TRS-80 Model I, Pocket computers, and TRS-80 music respectively.



https://www.youtube.com/watch?v=LKh3plUOR5w

As usual there was also an audience trivia game played by everyone who attended, and a great time was had by all.

SUNDAY: FINAL SPEAKER AND THE SHOW WINDS DOWN

RUUD BROERS: TRS-Time: A Modern Real-time Clock for the TRS-80 Model II



https://www.youtube.com/watch?v=iZjp41Xvnik

NOTES ABOUT TA25

1. The experimental nature of the TA trivia game meant that it ran longer than planned. The YouTube video shows it's run time as being 1hour23minutes.

Bear in mind this compares favourably with Pub trivia games I often attend which regularly start around 6:30pm and run up until 10pm. However when the TA runs too long it makes it difficult to plan what to do after the game is over, other than head to bed!

For TA26 I am going to aim for a 1hr trivia game comprising of 4 rounds for the ladies and 6 rounds for the gentlemen. Naturally I will be looking for experts to play the trivia games next year.

2. 2025 was the 40th anniversary of the release of the Tandy 6000 business computer. While it is not as well known as some of the other TRS-80 models, it is significant that it holds the record for being the last 8in system in the line which started with the Model II in 1979. It was also for a time the best selling Xenix multi-user micro computer system, and also the last computer of any brand to be sold with an 8in disk drive. The Tandy 6000 was released in 1985 and was available up until early 1988.



Picture: Tandy 6000 as advertised in RSC-19 in 1988

3. We have already decided that 2027 will be commemorate 50 years of the TRS-80 Model I, released in 1977. For 2026 we will be celebrating 40 years of the Color Computer 3, the last and most capable of the Coco line.
We always like there to be presenters giving talks on Coco topics, and next year would be perfect. Please contact us if you have some history with the Coco 3 and a story to tell, or a tech talk that everyone would appreciate.

ANOTHER TANDY ASSEMBLY DONE AND DUSTED

By mid afternoon on Sunday the show was over and we all slowly packed our tables and talked about how great the show was and who would be back for TA26 (next year's show).
The committee for TA which I am a part of have already started outlining next years show, but if you have any ideas, no matter how crazy you think they are, drop us a line at andyassembly@gmail.com and tell us all about it.

Better still if you know of veteran TRS-80 personalities from back in the day and think they would be interested in telling us their story we'd be all ears!

NEXT TIME: It will be the new year so let's start talking about TA26!

Ian Mavric
ianm@trs-80.com



Ian Mavric is an IT Specialist who also restores and collects TRS-80's and classic cars. He lives with his wife and kids in Melbourne, Australia.

LOOKING FOR FAST, INEXPENSIVE, UNLIMITED MASS STORAGE FOR YOUR TRS-80 MODEL I/III/4/4P/4D?

The amazing

"FreHD"

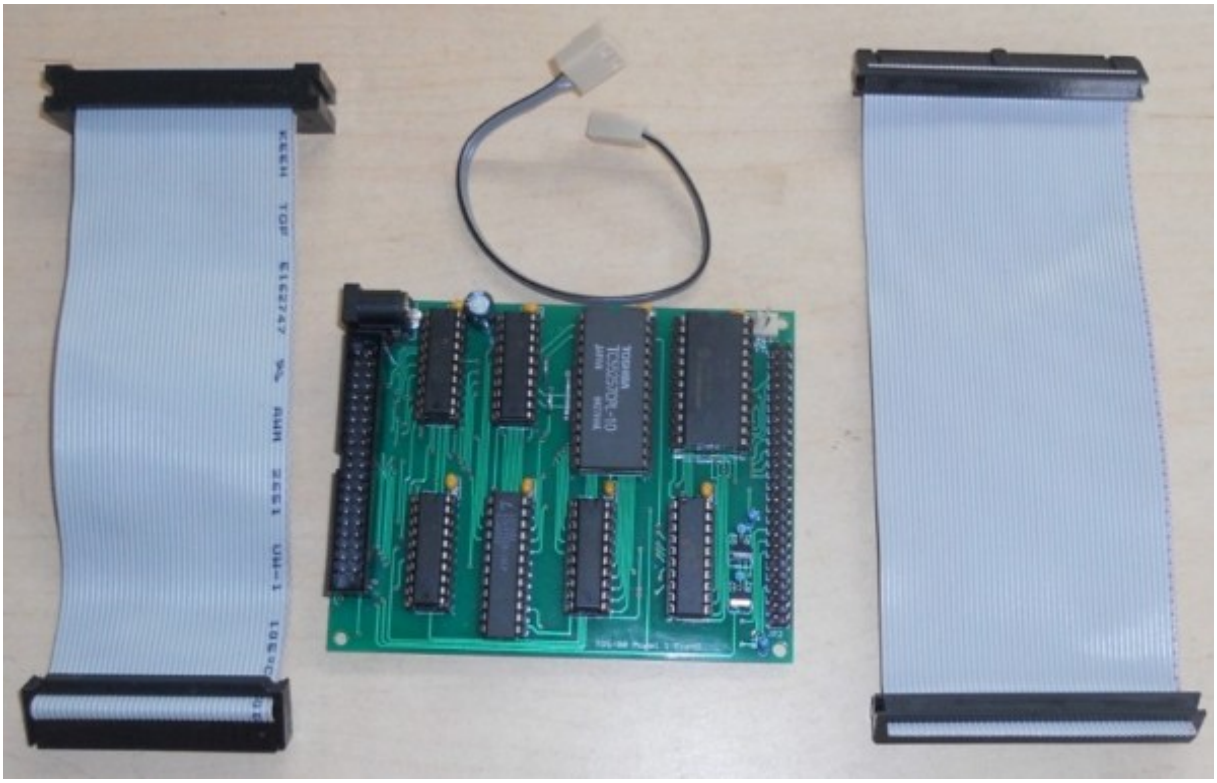


-Emulates a TRS-80 hard drive, but faster than any hard drive!
-Works with your favourite DOS (LS-DOS, LDOS, CP/M, Newdos/80 2.5)
-Uses SD card for storage medium
-Bonus free Real Time Clock function!
-Designed in Belgium and proudly built and shipped from Australia
-Kit form or fully assembled

Order yours today
http://trs-80.com.au/trs80/emulator.htm

“Quinnterface” Mini Expansion Interface for 16K Model 1 FreHD users.

- COMPLETLY ASSEMBLED AND TESTED -



This is THE perfect device for all Model 1 users who own a 16K Level II unit, (which is most of us!), but no Expansion Interface or disk drives, especially if you don't want to modify your M1 with upgraded boot ROM or memory upgrade.

The ‘Quinnterface’, developed by J. Andrew Quinn from New Zealand, adds 32K RAM and auto-boot functionality to you FreHD.

U.K. And Europe, contact Bas. at BetaGamma Computing.
U.S.A. And rest of the world, contact Mav. At “The Right Stuff”



https://www.youtube.com/watch?v=ZE3nMxTLgjk

EBAY.CO.UK 'BARGAINS'

HERE ARE JUST SOME OF THE MORE INTERESTING/UNUSUAL TANDY EBAY LISTINGS, SINCE OUR LAST ISSUE.
WERE YOU ONE OF LUCKY ONES?



£1



£225



£225



£53
NS



£312
NS



£210
NS



£157
NS



£20



£256

POWER SUPPLIES

FOR THE TRS-80 MODEL 3 / MODEL 4/4D/4P



* Uses modern day MEAN WELL power supply units
* 50w for Model 3 version and 65w for the Model 4 version
* The Model 4 version uses a SEPARATE +12V POWER SUPPLY (prevents video "spasm")
* No more RIFA CAP explosions
* No more COLD solder joints
* Units are DIRECT REPLACEMENTS for the original power supplies in all models
* Extra +5V output connectors (1 on Model 3 and 2 on Model 4) for Gotek/freHD boards
* For more information and pricing contact

newSOFT

https://jaynewirth.wixsite.com/newsoft (website)
jaynewirth@verizon.net (email)



TRS-80 System

All items stocked, Barclaycard, Access & American Express are welcome, or apply for your own RADIO SHACK Charge Card.
U.K. Delivery by Securicor.
Direct and Personal Exports.

RADIO SHACK LTD.
188 Broadhurst Gardens,
London NW6 3AY.

Tel: 01-624 7174 Telex 23718

Functional Business Software on Cassette for 'TRS 80' LEVEL II 16K

British Software written for British companies and now is in daily business use:

1) "BANK A/C" PROGRAMME!
21 column analysis, self totalling on all columns keeps full alpha and numeric records at command show's monthly and yearly totals to date, partners drawings, total o'heads to date, etc. £21.95.

2) "SALES LEDGER"
Full record for a month, 17 entries for each invoice, plus totals 8 columns, searches and totals individual accounts, weeks sales, months individual heading totals, displays entire files £35.

3) "ORDERS FILE" PROGRAMME!
Searches by name, order no., customer order no., agent. Running totals of orders still in hand, totals agent sales, etc., displays entire file £14.95.

4) "IMPORTERS, COSTING AND SELLING PRICE" PROGRAMME! £9.95.

All programmes include practical instructions! Tailored Software available. Packing & V.A.T. INCLUDED.

ACCESS COMPUTERS 2 Rose Yard, Maidstone, Kent ME14 1HN Tel: Maid. (0622) 58356

TRS-80 I & II

DISK SOFTWARE

THE TRS-80 SPOOLER gives your Model I/II TRS-80 the same performance edge bigger computers have had for years. With the SPOOLER installed, CPU time ordinarily spent waiting for disk and printer operations to complete is made available for useful work. Typical commercial applications can run up to TWICE as fast with the SPOOLER installed. Installation is simple and fast and requires no changes to your programs! — £45.00.
For Model I state NEW DOS/TRS DOS or NEW DOS/80.
For Model II state serial or parallel printer.

MODEL II EXTENDED BASIC adds command abbreviations and additional commands for variable/line number cross referencing, re-numbering with block re-location, dynamic dump of variable contents and search for embedded strings or commands in program text. Requires no user memory or user disk space — a must for every programmer! — £65.00.

MODEL II DOS PACK adds DOS commands to compress Basic programs, echo screen to printer, re-route printer output to screen, print screen from Basic or keyboard and save inadvertantly 'lost' Basic programs — £60.00.

FRIEND is an extremely powerful set of Model II DOS utilities which allow 'group' or 'generic' processing of disk files. Copy, kill, print, display — a real time saver — £37.50.

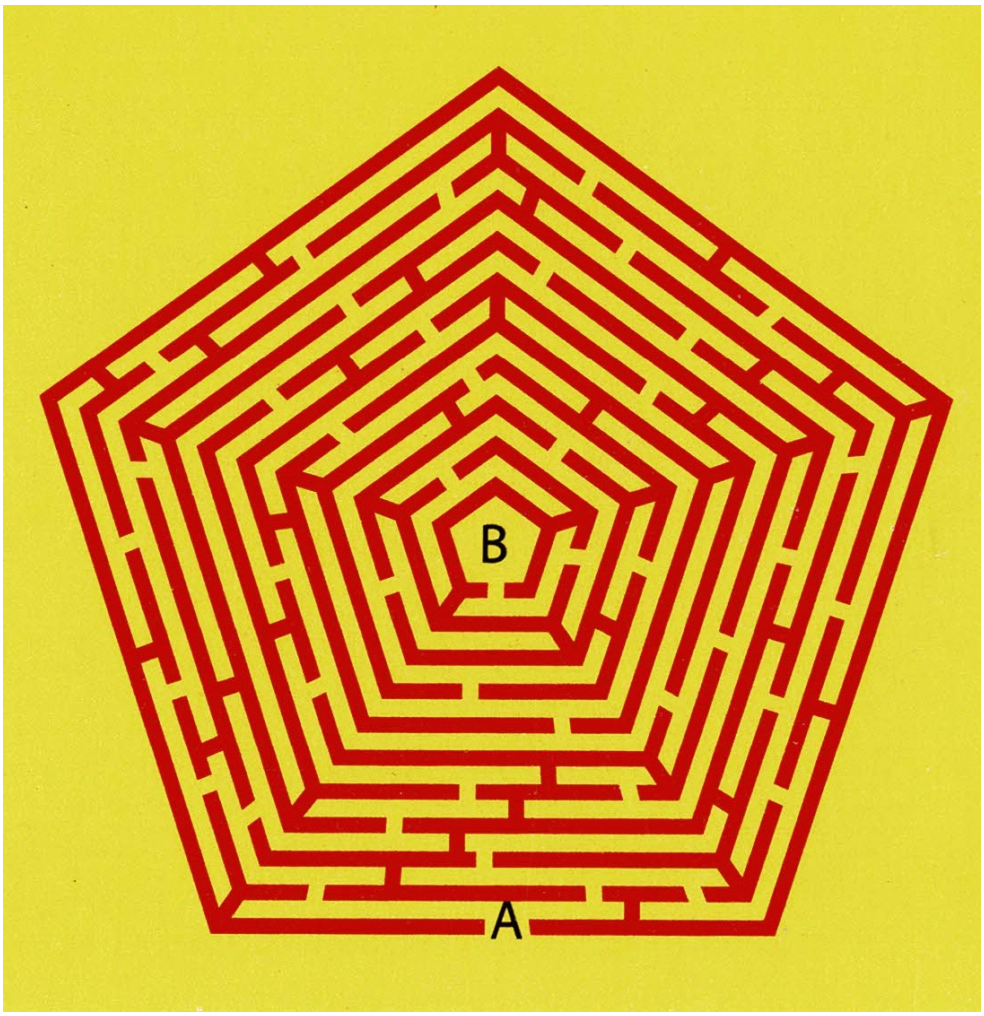
Other Model II software includes remote date entry, inter-computer communications and spoolers with dynamic disk buffers.

All Model II disks include a free DOS program which makes over 35 corrections and enhancements to TRSDOS, including moving 'break' to 'ctrl-6'.

All prices are inclusive. Write or call for further details.

SYSTEM SOFT
49 Dunvegan Drive, Rise Park, Nottingham NG5 5DX
Tel: (0602) 275559

AMAZED



IF YOU HARDWARE-HACKERS ARE FINDING THOSE PCB BOARDS TRACKS ARE MERGING INTO ONE OR YOU ASSEMBLER CODERS HAVE LOST A PUSH OR A POP, GIVE YOURSELVES A BREAK.

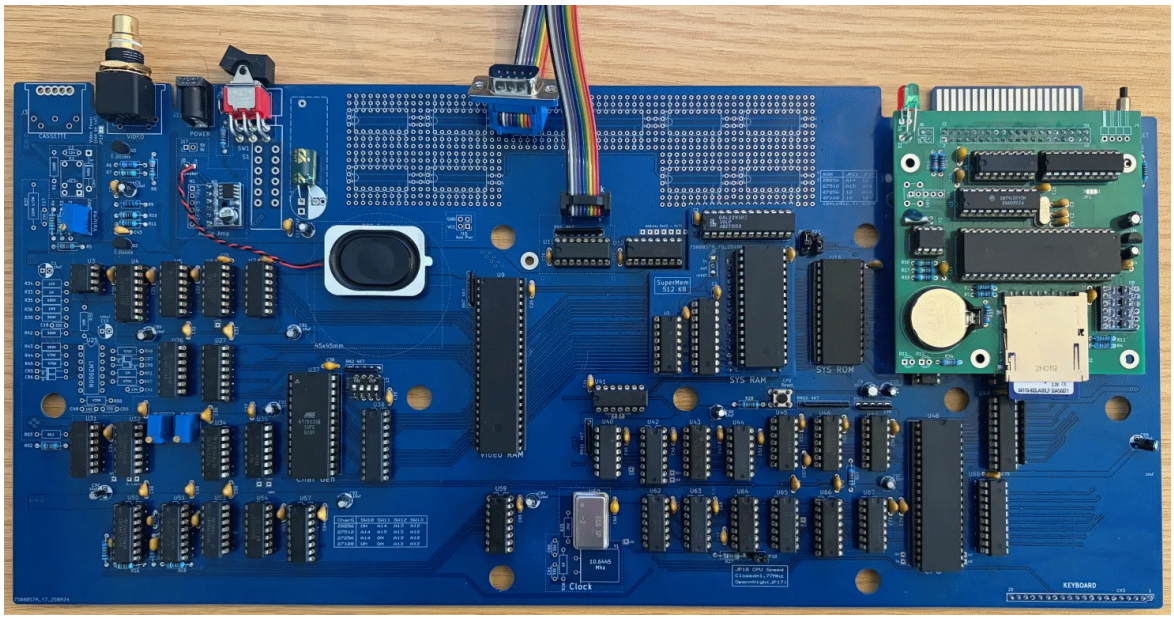
SIT DOWN WITH A COFFEE, PUT A TIMER ON AND SEE HOW LONG IT TAKES TO GET FROM A TO B, FOLLOWING THE YELLOW ROAD.
TRY FOR COMPLETING IN UNDER 8 SECONDS!

Surfin' The Net

KiwiSinceBirth has released v2.0a of his redesigned Model I motherboard.
He reports the following changes:

This project is an upgraded / evolved TRS-80 Model 1 main board replacement. It was designed to provide a reliable, compact, and modern replacement, removing some limitations of the original 1970's product, while still remaining faithful to the original technology (i.e. no emulation)

v2.0a provides significant improvements in video, not only does it use dual port RAM for snow free video, but it changes the video sync generation circuit bringing back the picture adjustment controls not present in V1. Also supported is software switchable fonts, for alternate languages or other graphics. Other major features added to the board include an audio amplifier for a small speaker (which can be mounted on the board), Alpha Joystick port compatible with most software, and ability to mount a modern clock oscillator. Have added about 10 new config jumpers/settings in support of new features. V1 is still fully supported including a separate build guide. Not forgetting Supermem bank RAM and internal FreHD



You can find all the information, as well as downloads for the BOM and circuit board, at

github.com/kiwisincebirth/TRS-80-MODEL-1-K/tree/main

IN MAV'S HISTORY WORKSHOP
You Bought a Tandy Model 2000! What Now?



Part 2, by Ian Mavric

This time I talk about how the Tandy 2000 can be upgraded and customised for various duties. I also talk about why the major compatibility issues caused such a headache for users, especially if they hoped to buy software from places other than Radio Shack.

HOW THE SYSTEM EXPANDS

The Tandy 2000 departed from a lot of PC compatibles of the time by putting things like the parallel port, serial port and monochrome text display on the motherboard. This was in contrast to an IBM PC/XT or AT on which you needed to install cards on the motherboard for serial, parallel printer, and display card into one of the 8 slots on the motherboard. This made sense at the time the IBM PC was designed... why include a serial port if the end user had no serial devices. Likewise if they didn't need to print then a parallel printer port is also wasted. However by 1983 when the T2K was being designed adding a printer was pretty standard for even the most low-end computerist, and a serial port was becoming a necessity as most people usually ended up adding at least one serial device (typically a dial-up modem). In the near future two serial ports were going to become a necessity as people needed to be able to connect a modem and a mouse to their PCs. Radio Shack designers rationalised that with more of the common items added to the motherboard, fewer expansion slots were needed, so they settled on 4 slots for the T2K.

Ever since the Apple][, and even the Altair S-100 systems cards were added to computers by opening them up to install a card. The IBM PC continued this tradition and so it also carried over to the clones and compatibles which followed in IBM's wake. Radio Shack has never really liked owners opening up and messing around inside their computers so when designing the T2K a way was devised to install expansion cards *without* opening the case: the cards slide into place after removing a small metal cover.



Photo: Installing expansion boards into the Tandy 2000 (source: RSC-11)

THOSE FOUR EXPANSION SLOTS

4 expansion slots sounds like plenty but it really becomes a shortcoming if you want to fully expand the system. Let me explain starting with how the Ram is configured on the T2K.

The base model T2K which comes with 2 floppy drives has 128K Ram installed and all 4 expansion slots are empty. This sounds like a good start. The T2K can be expanded to a maximum of 768K Ram, and the first step is to bring the system up to 256K. This is done by removing the motherboard and installing a second 128K Ram daughterboard onto the motherboard. This is as far as motherboard Ram can be expanded and is in fact how the T2K HD (which comes with 256K Ram as well as a hard disk drive) is set up. I should point at this moment that adding a hard drive to a T2K takes up one of the 4 expansion slots - the top one.

Ram can now be added in increments of 128K via expansion cards, which hold, initially, 128K each with room for an additional 128K Ram making each expansion card a 256K jump. Up to two cards can be added which, fully populated with Ram chips, brings the total to 768K Ram. Doing so takes away 2 more expansion slots. On a T2K HD we are now down to 1 empty expansion slot and 2 empty slots on a dual-floppy plain T2K.

You may recall last time I mentioned adding a colour monitor to the T2K required an additional expansion board for high-resolution graphics to connect to the CM-1 monitor. This, obviously, takes the last expansion slot on a T2K HD, and only leaves one empty on the plain T2K. Earlier this year I did pick up just such a T2K HD however I have yet to get it running well enough to find out what was on the hard drive. One thing I am pretty sure of is that it didn't need a mouse...

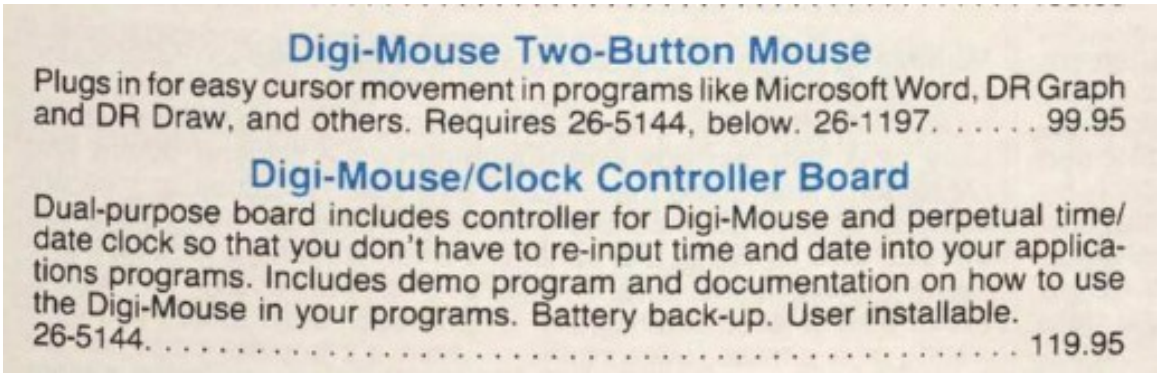
TO MOUSE OR NOT TO MOUSE

It can be hard to imagine it now, but there was a time that we *didn't* use one on an MS-DOS PC. Basically the first popular mouse appeared on the Apple Lisa, and then shortly after on the original 128K Apple Macintosh. When MicroSoft released Windows 1.0 in 1985 use of the mouse became essential, however Microsoft had released their first mouse in 1983. Called the 'green eyes' mouse it connected to it's own PC expansion card 'InPort', though within a few years the serial mouse was released, and the rest, they say, is history.

Serial connection allows a mouse to be inexpensively produced so that anyone who needed one could afford it. By this time virtually all PC-compatibles came with a serial port (or two), so you just bought a serial mouse, from either MicroSoft or any other company which made them, plugged it into a serial port, added a driver to AUTOEXEC.BAT and you were set. Prior to this time people operated their MS-DOS computers the same way we operate TRS-80s, which was with a command line with typed commands.

Eventually Radio Shack saw the wisdom in adding a mouse to their computer line, originally a black one-button mouse for the Color Computer. When their new MS-DOS masterpiece the T2K was nearing completion, RS took inspiration from Microsoft's 'InPort' and made an expansion card for the T2K and released a snazzy new 2-button mouse they called 'Digi Mouse'. For those who haven't already realised the issue, the expansion board takes another slot and if you have a hi-res colour T2K HD with 768K Ram you have no slots left. Removal of one memory expansion card brings the Ram down to 512K which was acceptable for the time, though some programs like Lotus Symphony, dBASE III, MultiMate, and SuperCalc[3] were much happier with 768K Ram.

To sweeten the deal with the Digi Mouse controller expansion card RS also incorporated a battery backed real time clock, another accessory which was gaining popularity on the PC by the mid 80s.



Digi-Mouse Two-Button Mouse

Plugs in for easy cursor movement in programs like Microsoft Word, DR Graph and DR Draw, and others. Requires 26-5144, below. 26-1197. . . . . 99.95

Digi-Mouse/Clock Controller Board

Dual-purpose board includes controller for Digi-Mouse and perpetual time/date clock so that you don't have to re-input time and date into your applications programs. Includes demo program and documentation on how to use the Digi-Mouse in your programs. Battery back-up. User installable. 26-5144. . . . . 119.95

Photo: Equipping the T2K with a mouse required these two items (RSC-11)

BEYOND THE BASICS - EXPANDING THE T2K FOR SPECIALIST TASKS

By 1986 Radio Shack had changed the marketing of the T2K into what it called the 'Advanced Technology Series' and the T2K was the proud leader of that series. By this time the Tandy 1000 had taken over as the RS MS-DOS computer of choice for home, small business and schools use, so the T2K was aimed more at high-end users which required it's brilliant performance in specialised areas. At this time four new expansions were released:
1. 8087 Math Co-Processor upgrade. Specialist mathematically intensive software, like VersaCAD, was becoming popular and was ported to the T2K. Math Co-Processor upgrades were popular for a while on PCs, they increased floating-point math calculation speed sometimes by orders of magnitude, though software had to be written in such a way to take advantage of it.

2. 3270 Co-axial adapter kit. Connecting PCs to IBM mini and mainframe computers was popular in the 1980s while there were many System/360 and AS/400 installations running in the corporate world. Connectivity to those machines was via a method called

3270, and IBM would prefer people bought IBM PCs, XTs, and ATs to connect to their mainframes and minis, but RS shows you could accomplish this with a T2K for a lot less.

3. 4-port serial expansion board. As the name suggests, it adds 4 RS-232C serial ports to the T2K (COM2, COM3, COM4, COM5). The purpose of this was that RS had been working on a version of Xenix III as a multi-user operating system. It was not popular and the Xenix installation on the Tandy 6000 was bug free and dependable.

4. Graphics Tablet. VersaCAD and Lumena graphic design applications used the GT-2000 Graphics Tablet for inputting graphic information onto the screen in what was the beginning of the popular computer-aided design occupation.



Photo: RSC-15 advert for GT-2000

(photo courtesy radioshackcatalogs.com)

WHY IS IT SO HARD TO FIND NON-RS SOFTWARE THAT WORKS?

From Wikipedia: "The Tandy 2000 was nominally BIOS-compatible with the IBM XT, so well-behaved DOS software ran on both platforms. However, most DOS software of the time bypassed the operating system and BIOS and directly accessed the hardware (especially video and external ports) to achieve higher performance. This rendered such software incompatible with the Tandy 2000."

When IBM designed the PC (and XT and AT) they released programmer guidelines on how to use system calls and the BIOS to access hardware. This system is clever because it makes writing software quicker and it will work on the largest number of MS-DOS systems if the programmer uses the proper system calls. The downside was that using this method was slow compared to directly accessing hardware. Believe it or not the hardware of the IBM PC and XT is not very fast, it's really on par with a TRS-80 Model 4 most of the time. Assembly language programmers are a clever bunch so once one started using direct hardware calls in their software pretty soon all of them did and the table of BIOS system calls went out the window. If you had an IBM-compatible that was pretty close to the XT this was not a problem. Your supercharged Advanced Technology Series hi-end computer (a.k.a T2K) had no chance of working with 90% of the software out there.

AM I RESTRICTED THEREFORE TO ONLY SOFTWARE SOLD AT RS?

Over the life of the Tandy 2000 things turned out to be not all bad. Talented programmers with the right information at hand could patch popular software surprisingly quickly. A legendary example of this was with Lotus 1-2-3, the killer-app spreadsheet which did for the PC what VisiCalc did for the Apple][. According to Lotus Corp. it only took 6 hours to patch the program for the T2K and the result, due to the ultra-high performance nature of the T2K, was the best computer at the time to run 1-2-3 on. The next few years saw most of the popular applications patched or ported over to the T2K, usually with the same eye-popping performance improvement over generic PCs, fully justifying the work it took to get the program running on the T2K.

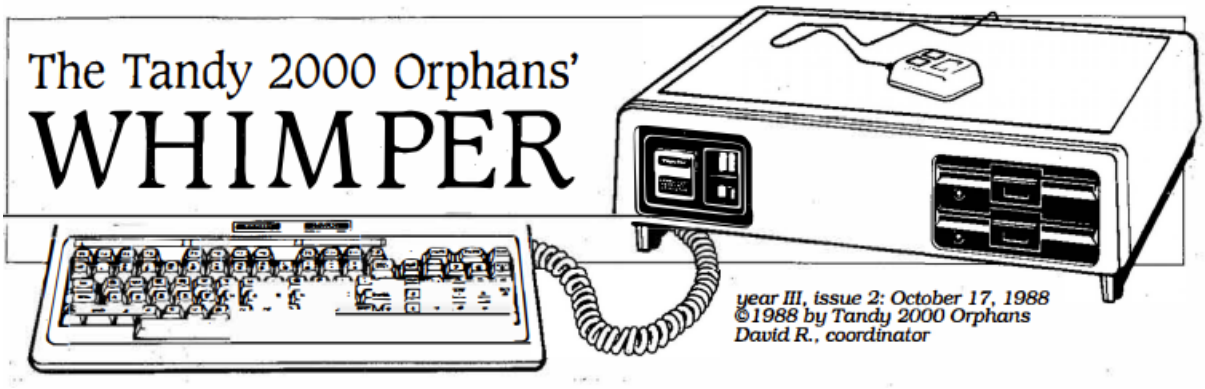
Google and old Radio Shack computer catalogues are your friend here, to find software which runs on the T2K. As mentioned before much of what was PC popular was available for the T2K eventually. While the computer started out as a bit of a let-down as far as popularity went, RS must be commended for sticking with the machine and constantly adding software to the library for the machine. The photo below shows that by 1985 the software library for the T2K was indeed impressive. In this picture you see software which was not even available to purchase at Radio Shack computer centres, but RS felt it important enough to point out that even if you bought the software elsewhere, the T2K was the best machine to run it on, hands-down.



Photo: T2K software library c. 1985
(photo courtesy radioshackcatalogs.com)

OTHER SUPPORT - NEWSLETTER

While there was never a major widely available monthly periodical for the T2K, a group of enthusiasts called the Tandy 2000 Orphans released a newsletter called "Whimper". The name of the newsletter describes how the computer came into the world with a bang of hype and interest, but by 1987 it was on close-out and support and interest in the machine petering out, the T2K's era whimpered to a close. I can understand this as at the time there was still much interest by Radio Shack in lower performing machines like the Tandy Model 4D and Coco 3.



Whimper magazine debuted in March 1987 and continued until at least January 1990. It's a great resource for information on programs which work, or don't work, or partially work. A lot of really smart people wrote great articles for this forgotten newsletter. If you are serious about messing around with the T2K these days you should download and read them. They are stored on a Git Hub server:

https://github.com/Tandy2K/Tandy2000/tree/master/Documentation/Magazines-Newsletters/Whimper

You soon notice that not all issues are present which is a shame, but it also means that if you find the issues which are missing in this collection when you find someone selling their T2K systems you have something rare and should consider scanning it and adding it to the resources Git Hub for Tandy 2000.

NEXT TIME

In the next instalment of this series I'll talk about some of the software which really shows off the Tandy 2000.

Ian Mavric

ianm@trs-80.com



Ian Mavric is an IT Specialist who also restores and collects TRS-80's and classic cars. He lives with his wife and kids in Melbourne, Australia.



The *BEST* in TRS-80s

Call The Right Stuff

Ask for Ian
The number is +61 416 184 893

That's The Right Stuff
And he's in Melbourne







TRS8BIT IS SUPPORTED BY YOU AND PRODUCED
AT WWW.TRS-80.ORG.UK