Orange County TRS-80™ Users Group

November, 1980

Ed Faulk, Editor, 2531 E. Commonwealth, Fullerton, CA 92631, (714) 738-0789

October

The October meeting of OCTUG, Inc. took place on Sunday, October 12, 1980 at the Fountain Valley Recreation Center in Fountain Valley. The meeting was well attended with around 165 people showing up to ogle the wares of the 15 or so vendors (which included the Orange County branch of Bob Thorpe's store!). We had the good fortune to have Bob bring a couple of Model III's for us to look at and play with (darn, I never get a chance to play with the new stuff!).

Our featured speaker was Alan Wisenberger who discussed applications using hard disks and the TRS-80. Based on the number of people that sat around, the presentation went over very well.

Board Meeting

After the regualr meeting we held another excellent board meeting. We continued to discuss the new club library and the way in which it will be run. It was decided that there would be a legal form filled out by each person submitting programs to the library stating that the program is his original work and that the club has non-exclusive rights to the program to use in any fashion deemed reasonable by the club.

The distribution media will probably be disk (without an operating system). We are considering building a mini-system that will belong to the club and may be used to load machine language programs. More on this as it becomes available.

Christmas

Our Christmas issue will be larger than usual since there are a few articles that we have been holding for some time. Look for yours if you have sent it to us and it hasn't appeared yet. We will print an issue that is around 32 pages (or so) long.

November

The November meeting of OCTUG, Inc. will be held on the second Sunday of November, that is, on November 9th at the Fountain Valley Recreation Center. The meeting will start at 1:30 PM (but the doors open at noon) and will run until around 4:30 PM.

The board meeting will follow the regular meeting and all members are welcome to stay and participate.

Newsletter

The OCTUG Newsletter is published monthly by OCTUG, Inc., a California non-profit corporation. Circulation is to 700 members and over 50 clubs.

Membership

Our membership has decreased quite a bit this month. We had 150 members whose membership expired this month and only 18 of them renewed. Let's get out and try and get more members. We need to have greater participation. Those of you who are out-of-town, this is a great opportunity for you to get more involved. Remember, membership is still only $10.

Table of Contents

Editor: Ed Faulk Cover Art: Jeannine Likins

Classified Ads

FOR SALE
Two MPI B-52 double-sided, double density drives. Never used. $295 each.
Burt Sigal. (213) 887-2979 (days) or (213) 789-5592 (eves).

FOR SALE
TRENDCOM 100 printer with interface & cable. For use with keyboard or expansion interface. $275. Call Darrell Flenniken at (213) 991-3984 / (213) 829-8865 (after 5:00)

FOR SALE
TRS-80, Model I, Level II, 16K Still in box. $650. Call Tim (714) 551-4011.

FOR SALE
Centronics 701. 132 columns, 60 characters per second bi-directional head. Upper and lower case. Call Barney at (714) 893-8100.

FOR SALE
Black Box printer, P interface, impact keys, 10 CPS, excellent condition. $300. Al Marino, Chatsworth, (213) 349-9434.

BUSINESS SIG MEETING
Meets 4th Sunday of every month at Fullerton Savings and Loan, FV. Brookhurst & Talbert. Approx 1 mi so. of club meeting place. Times: 1:00 to 3:00. For more info, Call Joe Deutsch, 540-2772.

Notes

The Deadline for the next issue of the newsletter is:
November 26, 1980

Mods for R.S. Business Mailing List

by
Dave Stambaugh

In the process of using Radio Shack's Business Mailing List System, a couple of shortcomings have sort of limited the usefulness of the program for my application. I have here some simple modifications which make the program a little more versatile, and may help other users of this otherwise good package.

The most serious limitation of the program for me is in the way selective listings are performed. As I outlined in a previous review of this program, each individual entry can be classified under 1 or more of a possible 8 sub-categories which are chosen at the time a file is initialized. When making selective printouts of a file, the program will only print out the entry if it matches in ALL of the sub-categories selected for printout. For example, if you were to tell the program to print out entries under sub-categories 1, 6, and 8, the entry would have to match all three of these in order to show up on the printout, i.e., an entry classified 1, 2, 6, and 8 would be printed, but one coded 1, 5, and 8 would not. What I needed was to be able to get the printout under any one match, i.e., any entry under category 1, or 6, or 8. Of course, the entry may also be coded under other categories; I just needed to be able to get all the entries with at least one match to my selected code(s).

If the ability to print out files this way interests you (and if I have not totally confused you yet), make the following change to the MLS program:

```
LINE 2380 -- CHANGE TO => 2380 IF(ES AND E7)=0THEN2400
                  WAS => 2380 IF(ES AND E7)<>0ESTHEN2400
```

The next couple of changes are just enhancements to make the program a little more convenient. One thing I wanted was a tally of the actual number of entries printed out each time. To accomplish this, make the following changes:

```
LINE 250 -- CHANGE TO => 250 CT=0:PRINT:PRINT"SELECT...
LINE 1360 -- CHANGE TO => 1360 LPRINT" ":LPRINT"TOTAL
ENTRIES PRINTED: ";CT:IFCK$...
LINE 2570 -- CHANGE TO => 2570 ...TAB(35);:CT=CT+1
```

Finally, it's nice to have the computer supply the date of the printout for you, rather than having to remember to write it on the listing by hand (of course you still must remember to set the correct date in DOS command mode using the DATE function!!).

LINE 1190 -- CHANGE TO => ...LPRINTCSZ$:LPRINT"DATE: ";LEFT$(TIME$,8)

These enhancements have made the Radio Shack Business Mailing List program more useful in my application, and hopefully will be of some value to other users of this software.

Model I Disk Controller

By
Dave Stambaugh

What I am going to try to do in this article is shed a little light on how the Model I actually performs disk operations at the hardware level. Many TRS-80 users are not aware of how the system works and that, armed with a little technical know-how, it is possible to go beyond the limitations of our disk operating systems. In fact, one could write his or her own disk operating system, complete with non-standard disk formats, and make it next to impossible for others to access these diskettes without the proper software. I'm not sure how interesting this is going to be for most readers; perhaps if there is some positive reaction to this first try, I'll elaborate on other hardware-related subjects.

As you may know, the Model I computer is a "memory-mapped" system. Just what does this mean? Well, the Z-80 microprocessor has the capability of directly addressing a total of 64K of memory, from address X'0000' to X'FFFF'. In the Model I (hereinafter and forevermore referred to as the M1), the Level II ROMs occupy address X'0000' to X'2FFF'; X'4000' to X'FFFF' (in a 48K system) is normal read/write memory. Did I hear someone ask "But what happened to that missing chunk of address space from X'3000' to X'3FFF' ??" Gee, I'm glad somebody asked.

Memory space from X'3000' to X'3FFF' is reserved for use in memory-mapped I/O (input/output) operations. That's right, to the Z-80 processor, the keyboard, video, disk drives, printer, and RS-232 board are all just like any other memory location! Most any memory-reference instruction can be used in talking to these devices, although the programmer must of course know how to use these addresses to accomplish anything useful. In this article, I am only going to discuss the addresses relating to the disk controller system.

The first address related to the disk system is X'37E0'. An output of the proper data to this address will select one of the four possible disk drives to be accessed in a subsequent operation; it will also start the drive motors and load the heads of all connected drives. The lower 4 bits of the data bus determine which drive is to be selected; bit 0 on selects drive 0, bit 1 selects drive 1, bit 2 selects 2, and 3 selects 3. For example, the following sequence would select drive 1:

```
LD      A,02
LD      (37E0),A
```

Once a drive has been selected, it will remain selected for about 3 seconds. The write to address X'37E0' triggers a special IC which acts as a sort of timer; when the timer expires, all drives will be de-selected. It is very important that during lengthy disk operations, this procedure of selecting the drive be repeated often enough to prevent the timer from "timing out" in the middle of a data transfer. This may sound familiar to some of you veterans of TRSDOS 2.1 and its occasional tendency to engage in "silent death".

Now I'll get to the real heart of the M1 disk system, the Western Digital 1771 disk controller chip, that big 40-pin job located in the expansion interface. In my humble opinion, this is one remarkable IC. It is actually another microprocessor, one specially programmed and designed to make interfacing with floppy disk drives as painless as possible. Floppy drives are very stupid machines; unlike their bigger brothers (hard disk systems such as the CDC Hawk) which have some "intelligence" of their own and can figure out how to get to the desired track without explicitly being told, floppy drives can only tell you when they are on track 0, and when the index hole in the diskette passes by, and that's about it. You cannot tell the floppy "Go to track 19" because the drive doesn't know how to get there. You must know where the head is now, where you want to go, figure out which direction (in or out) that you need to go to get there, and how many times to step the head in that direction. The beauty of the WD 1771 chip is that it can handle all this housekeeping for you. Once the chip has been initialized, you tell it "Go to track 19"; it figures out on its own how to make the drive get there and signals you when it is finished.

There are 4 addresses associated with the disk controller, which can be either read from or written to; these actions are summarized below:

ADDRESS	READ ACTION	WRITE ACTION
37EC	Read 1771 Status	Give 1771 a Command
37ED	Read 1771 Track Reg.	Write to 1771 Tk. Reg.
37EE	Read 1771 Sector Reg.	Write to 1771 Sect. Reg.
37EF	Read a 1771 data byte	Write a 1771 data byte

I'll start with the 1771 command register. There are four classes of commands which can be given to the controller, identified as Type I thru Type IV. I am only going to give a brief description of each command, rather than trying to explain them in detail. This should be sufficient to give you a good idea of how the controller functions in general terms. If enough interest is shown, I may go into more detail in a future article!

Type I commands are primarily used in initializing the controller, and in performing head positioning on the disk drive. These commands include Restore, Seek, Step, Step-In, and Step-Out. There are several variations to each of these commands, allowing the programmer to vary stepping rates, as well as making the controller chip verify its positioning by reading track information directly from the disk and comparing it to the track we intended to go to; a miscompare would cause a No-ID-Compare or Record Not Found condition.

Perhaps this is a good time to digress a little and talk about disk formatting. Many users are not aware of the fact that a large portion of every track on a diskette is "wasted" in being used for "overhead" functions. I am not talking about data areas reserved for DOS, which cannot be used for storing user data. As you know, each track is divided into 10 sectors, each of which contain 256 bytes of data. What you may not know is that these sectors are not actually contiguous; there are leading and trailing fields of "control" information before and after each sector which the disk controller chip needs to keep track of what's going on. Some of this excess is just plain "filler" while some of it is used to physically identify tracks and sectors for the controller. As a general rule, in a soft-sectored format such as that in the M1, there is about 20% "wasted" space on each track. The "filler" is necessary to compensate for such variances as diskettes being used in drives running at slightly different speeds. Incidentally, "soft sectored" means that sectors are identified by information actually written on the diskette; "hard sectored" means there are individual holes punched in the diskette to identify each sector -- in that configuration, it would be up to the controller to count these holes in order to know what sector the head is over. In a soft-sectored application such as the M1, it is possible to format the diskette in a non-standard manner using sectors of lengths other than 256 bytes; this is probably its main advantage over hard-sectoring. Conversely, hard sectoring requires less overhead on the diskette, and theoretically leaves more room for user data.

Now back to the Type I commands. After giving the controller chip a command (write to address 37EC), we need to know when and whether the process we told it to do is complete and we also need to know whether there were any errors. This is done by reading the Status Register (again, address 37EC). The conditions which can be reported during a Type I command are DRIVE NOT READY, DISKETTE WRITE PROTECTED, HEAD ENGAGED (Loaded), SEEK ERROR (Controller could not find the track we were looking for), CRC ERROR (Usually means a data transfer error -- more on this later), TRACK 0 SENSED, INDEX PULSE SENSED, and CONTROLLER BUSY (Command still in progress). It is interesting (sort of anyway) to note that you cannot immediately poll the status of the 1771 after giving it a command; you must wait at least 24 microseconds. In the M1 this is usually done with a sequence of time-wasting instructions such as:

```
PUSH    BC
POP     BC
```

```
PUSH    BC
POP     BC
```

This is the reason that the DOS software must be modified in a system using one of the available speed mods; increasing the clock speed of the processor causes it to execute the above time-waster in less time than is allowed by the 1771, so it becomes necessary to substitute some other instruction which takes longer to execute, to waste more time. Incidentally, the bootstrap loader in ROM actually violates this spec, which probably explains why my machine will occasionally have to try more that once to boot DOS when doing a reset.

As long as I'm once again off the subject of Type I commands, I'll explain the term "bootstrap". The Level II ROMs contain a short program which loads the data on drive 0, track 0, sector 0 into memory from address X'4200' to X'42FF' and then executes this as a program, jumping to address X'4200'. Thus this area of the system disk always contains the program which actually gets the system going by loading DOS software into the proper memory locations and executing it. This is what technique such programs as RSM-2D use to load themselves; track 0 sector 0 contains a program which loads one of the 3 versions of RSM2-D, without using any DOS at all. If you want to see what this ROM bootstrap routine looks like, disassemble ROM from X'0696' to X'06CB'.

Since this article is getting longer than I originally anticipated and the AL playoff game is coming on TV, I'll continue next month. And maybe I'll actually get around to those Type I commands I keep referring to! Please, if anyone is interested in articles like this, let Ed or me know. Also, for those of you who can't wait, I highly recommend Bill Barden's TRS-80 Disc Interfacing Guide. It is the only detailed writeup of the M1 disk system I have come across, and is invaluable for those who want to learn disk programming.

B17 Again

by
C. W. Evans

B17 high-speed software loader has been around for a while but now there is a much improved Version 2. It will load BASIC or machine language programs at four times the normal speed. This is a great help on long and frequently used programs such as SCRIPSIT which loads in one minute. Strangely, B17 programs load more reliably than their slower speed counterparts. Hardware high-speed devices may load faster but may require modifications to your CPU or make the use of a printer awkward by having to plug and unplug.

Version 1 did this too but its use on machine language programs required you to know the starting, ending, and execution addresses of the program. The new version is automatic in

this respect. It is self-locating. It creates a load-and-go version of the program.

Version 2 has another new feature. It works on data tapes and loads and saves up to 420 times as fast. This is because it eliminates the leaders needed for every data item. B17 writes only one leader followed by the entire data array. Data tapes have been one of the most disconcerting things about the TRS-80.

The B17 module loads first, then the program. The combination is then saved and thereafter the two are loaded together. A BASIC program may be NEWed out, leaving B17 available for use with another program.

CPUs equipped with the easy-loading modification require the installation of a simple switch to bypass the modification during loading. This voids your 90-day warranty but this should no longer be a problem because the current production does not have the modification and older models are past their 90-day limit. The current production of Model I accomplishes the EZ loading feature in ROM. A "-1" after the serial number of your CPU indicates that you have the modification.

There are some machine language programs that are not compatible with B17, those with non-standard or non-contiguous data arrangements. SARGON is one of these.

The program comes with a comprehensive 22-page manual. At $25, it is one of the best software buys around. Even if you have Version I, the new one is a good buy for its data handling ability.

If your dealer doesn't yet have B17, Version 2, write to ABS Suppliers, P.O. Box 8297, Ann Arbor, MI 48107.

C. W. Evans, 9806 Amber Trail, Sun City, AZ 85351.

Protect Your TRS-80

by
TED CREW

It seems that the thieves of the world can ruin your day,... or week,... or month, by stealing your TRS-80 and all of your programs.

I experienced that feeling about two weeks before Christmas.

As I drove up to my office I noticed the door was slightly open. I immediately thought "I couldn't possibly have left the door open." I was right; I hadn't! Even though I had a dead bolt lock on the door, it had been kicked in. When I entered my

office I was really upset to see my TRS-80 gone.....

As I stood there in disbelief I started thinking:"Where are all of my cassettes and disks with my custom programs that I had worked nine months on?" I frantically searched the office finding that everything had been taken.

Soon the sickening feeling gave way to utter outrage at the thought that my valued possessions were in the hands of a thief who would sell them for a couple of hundred dollars to another thief.

Since I had coverage under my business policy I wasn't too concerned about the cost of the equipment or the cost of the programs I had bought.

Even though I had taken care to always make a back up of my programs, they were of no value to me since I kept them with the computer and they were also in the hands of the thieves. I now take care to keep my backup copy at a location other than my home or office. (I have a friend that owns a TRS-80 that keeps a copy for me; I do the same for him.)

The criminal element is becoming more aware that the TRS-80 is a very attractive target due to it's cost and popularity. Because of this, it is most important for you to make sure it is insured.

The question, "When you cover your TRS-80, is it covered?", doesn't have a simple answer as you will see.

If you have a homeowners policy, and you don't use it for anything relating to business, the answer is probably yes. However, your definition of business use may not be the same as your insurance company's.

For example: If you are an engineer working for a company as an employee, but occasionally do some work as a consultant, and you make use of your TRS-80, this would constitute business use and would exclude your TRS-80 from coverage. If you deduct your system from your income tax this also constitutes business use.

Another situation that coverage would not be afforded pertains to business policies. Nearly all business policies exclude coverage for property when removed from the business location. This could be disasterous for you if you took your TRS-80 home for the weekend to work on a program and it was stolen or destroyed while at your home. Your business policy would exclude it because it was away from your business location, and your homeowners policy would exclude it because it's used for business.

It sure sounds like the insurance companies have us between a rock and a hard place!

Is there an inexpensive answer? ..... YES.

For example: If you have an office type business (professional, real estate, accounting, photo studio, tutoring, etc.) it is possible to cover this property under your homeowners policy while in your home. Your insurance company can probably give you coverage on your TRS-80 by giving you a special endorsement to your homeowners policy including it in coverage for an approximate cost of $10 to $15 per year.

I recommend that you read your policy or, if you want to save about a week of reading, contact your agent and ask him if it's covered. The time to do this is before you have a loss and you hear your agent say "I'm sorry. It's just not covered". Once a loss has occurred it is too late.

If your present company can't give you coverage for your TRS-80, I would be happy to quote a rate that would include coverage for your TRS-80. Please contact me, Ted Crew, 2780 Sepulveda Blvd., Torrance, Ca. 90505, (213) 325-8588. Please ask for Ted, because most agents aren't aware that this coverage even exists.

OS-80 Review

BY
LOUIS SHAPIRO

In the great American tradition of "Bigger Makes Better," the TRS-80 world keeps looking for new disk operating systems. The current battle of the Titans is between NEWDOS-80 and VTOS 4.0, two formidable opponents. In a fit of perversity and economy I went looking in the opposite direction and found something interesting: PERCOMS'S OS-80 (formerly MICRODOS).

OS-80 is an operating system for BASIC programmers. It is cheap, fast, and gives you more room on a disk than any other system. Now that I've gotten your attention, let me tell you its drawbacks. OS-80 contains no automatic directory capability or provision for naming files. Machine language files are not easily handled. The programmer (you) must keep track of where everything is on the disk, and be careful not to write over anything. There, that wasn't so bad, was it? In fact, for disks dedicated to specific applications, such as a collection of business programs, or all your versions of Star Trek, for example, it's not bad at all!

OS-80 normally organizes the disk into 40 tracks. Each track contains 10 sectors, giving a grand total of 400 sectors, numbered 0 to 399. OS-80, itself, lives in sectors 0 to 19. That's around 6KB. Once the system is booted, OS-80 is totally resident in the computer (!) for one drive owners, this means disks may be fully packed with programs and files, with no operating system required. With OS-80, you're always in Disk BASIC. You never see anything like 'DOS READY.' It just

doesn't exist. Not to worry; like in NEWDOS, you can execute 'DOS' commands from BASIC. The commands are so few, yet so powerful, that I might as well describe them all:

```
CMD"F",d   - This formats (and verifies) a blank disk
             in drive d.
CMD"H",a$  - Prints message a$ when system boots.
             For example, a title or instructions.
CMD"K",a$  - Executes function a$ when system boots.
             For example, reply to "MEMORY SIZE ?" and
             load and run first program (or menu).
CMD"I",d   - Formats the disk in drive d, and writes
             MICRODOS on it.
CMD"M",d   - Writes the resident version of MICRODOS
             onto the disk in drive d. Must be done
             after CMD"H" or CMD"K" to make them
             "permanent."
```

There are only 3 commands to know to save or load programs:

```
LOAD dssss  - Loads the program on disk d, sector
              ssss. "LOAD 10033,R" will load and run
              the program starting on drive 1,
              sector 33. dssss may be a variable.
SAVE dssss  - Saves the resident program at dssss.
              Responds with last sector used.
MERGE dssss - Same as "LOAD," but without wiping out
              the resident program.
```

All disk I|O is random access, a la TRSDOS. The difference is mostly using the dssss format instead of a FILESPEC, since there are no "OPEN" or "CLOSE" commands. In addition, there are some neat, information manipulating commands -- "GET A$,dssss" and "PUT A$,dssss." The field statement takes the form:

```
DEF FIELD N, n1 AS F1$,..., nI AS FI$
```

This is used in conjunction with "PUT (OR GET) N,dssss". LSET, RSET, MKI$, MKS$, MKD$, CVI, CVS, and CVD are all used in the same old ways.

There is one catch-all function that does not appear in TRSDOS or its brothers, LOC(n). This returns different information depending on the value of "n":

```
n=   RETURNED VALUE:
0    Last drive/sector accessed (dssss form)
1    Largest sector that may be accessed by system
2    Disk controller status code after error
3    Number of unverifiable sectors after formatting
```

Maybe there was a page missing from my manual, but LOC(2) makes no sense to me at all! The manual is pretty good, otherwise.

All the Disk BASIC enhancements that you know and love from TRSDOS etc. are here in OS-80. These include things like

H for Hex, MID$ on the left of the equal sign, 10 USR subroutines, and the INSTR function.

The OS-80 system disk also contains several BASIC programs, callable from a BASIC menu that autoloads on booting. These programs not only simulate some of the utilities missing from the system, but since they are in BASIC, they may be modified, or used as examples in developing your own programs. Of course, you can delete whatever you don't need on your copied disks. Here's what these programs are:

A disk file manager program lets you keep your own directory. Remember, you have the responsibility for keeping the directory accurate -- it ain't automatic!

A disk utility program that contains formating, backup, copying sectors, determining free disk space, erasing sectors, verifying disks (checking for bad sectors), and dumping a sector in printable ASCII. All these utilities are written using the standard OS-80 BASIC command set. Some, like "FREE", are slow and awkward. I felt that they could be improved, but not appreciably speeded up.

The PERCOM 5 1/4 inch notebook is a BASIC text saver, organized into "pages." It comes to you filled with a short OS-80 manual (you get a paper one too), as well as some bad gags. There's no reason why you couldn't use this program to keep any kind of text. It's a neat idea.

A disk diagnostic test that writes on, and reads an entire disk. This checks the disk for bad sectors (while destroying everything on it) as well as giving your drive a workout.

A patch program that allows you to make changes to OS-80, when and if PERCOM issues them.

Well, I really like OS-80, even with its limitations. For one thing, it appeals to my vanity, to be able to quickly, and totally (almost) understand and manipulate an operating system. For certain types of applications, OS-80's limitations vanish, and in those situations, it can't be beaten. Lastly, at $29.95, it's dirt cheap, at least in comparison to the rest of the field. Not being able to save and load machine language is a pain, but of course, there are solutions: you can write your own BASIC utilities to do it, as I did. PEEKing, POKEing and LSETing did the job like a dream (a slow dream). Jesse Bob and his bit kickers at the Circle J Software Ranch have written a utility that will do the same thing, but differently (and faster, I hope). Jim Stutsman, the author of OS-80, told me, via MICRONET, that the next version of OS-80 (as yet unwritten) will have this function built in.

PERCOM has just introduced a Double-density disk controller, that they call the Doubler ($219.95). It plugs into the expansion interface as does their data seperator. While the doubler comes with a TRSDOS-like system called DBLDOS, PERCOM

is selling a Double-density version of OS-80, called OS-80D ($49.95). One 5 inch drive, armed with the doubler, and OS-80D, provides 183 KBytes! I believe that Circle J will soon be selling Double-denisty patches for NEWDOS and VTOS.

I should mention that I've been happy with every PERCOM product, both hardware and software, that I've used.

OS-80 was written by Jim Stutsman, and may be purchased from:

PERCOM DA, INC.
211 N. KIRBY, GARLAND, TEXAS 75042
(214) 272-3421

LOUIS SHAPIRO, 415 WEST 24 STREET, NEW YORK, N.Y. 10011
MICRONET 70210,306 -- SOURCE TCB792

Buffer Recovery

by
Joseph X. Brennan

This article is really in two parts, the first part is about trying to detect difficulties and errors during input and output operations with my system and MICRONET. The second part is about recovering 'lost' buffer files in Scripsit.

A problem developed when I detected that output from MicroNet directed to my printer and a buffer at the same time differed. I noticed this when I printed the disk copy and compared it with the original printed copy. The comparision showed that the printed copy was perfect for 18 granules and discrepancies appeared after about 2 granules with the disk. I assumed of course that the line was erratic. This was not true, since it was too random for that. Next I tried the RS-232-C and the modem combination. Using echo, I eliminated the modem. The RS-232-C was something else. I tried cable connection corrosion, always a possibility in the TRS-80 Mod I, nothing there. So I tried looking at the buffer itself to see if the problem was localized to one function or another.

I use the ST80III program for my telecomputing, and I found that in my 48K system the buffer begins above 7D00H and runs to top of memory. With it, I have written 36+ granules to disk. I examined the buffer using Small Systems Software RSM2D loaded at the highest point in memory. If I overwrite it, I reload it. So the scheme was to print the source (Micronet, Orange County Data Exchange, COMM-80 and others) to paper and also load it into the buffer and write it to disk. Then go to DOS READY, call up RSM at EC00H and start looking above 7000H with the 'A' Command. This displays the buffer in ASCII. Comparing the printout and the buffer and the disk, I found that any line noise showed up in all three. Ignoring line noise

(very little to ignore), I found that the printout was good and the disk/buffer matched it below 7FFFH. The disk matched the buffer correctly but had errors when compared with a good printout above 7FFFH. I looked for memory problems in the interface. I had it only once before, and that was due to corrosion at a memory chip contact. Further checking gave the memory a clean bill of health.

Well, I looked back at the RS-232. I checked switch status with ST80 and with Radio Shack's STATCOM from the Communications Package. I tried that dastardly connector between the interface and the RS-232 board. This is always suspect, and it proved to be as unstable as usual. I wiggled the Baud Rate Generator - U10. It proved to be the culprit. I carefully removed it and replaced it after cleaning. Observe the same precautions here as in dealing with memory or any other CMOS device. I work in my bare feet, etc. The printout|disk|buffer are now all in agreement. In doing this my printer went into repair and I had to resort to neglected remedies in diagnosis.

This led to the second part. When you have a program that writes to a buffer and then the program permits dumping that buffer to disk, any inadvertent 'booting' or the like destroys the pointers and the buffer usually has all the data there and you can't get to it.

Not so, I first found that I could read the buffer with a monitor (RSM, MON-3, etc.). I could print it out in hex or ASCII (at least with RSM), I could pucnh it to tape with T-Bug or RSM, but best of all I could write it to disk.

For ease of handling try writing to disk. To write to disk there is TAPEDISK/CMD and the DUMP/CMD. We also have another method. These techniques require that we know the address for the beginning and the end of the data in the buffer. You may not be able to recover all of a long piece because you have overwritten it, but any usable amount is better than nothing. Find the beginning and ending addresses, save them and convert to decimal if needed. This is not needed in using TAPEDISK or DUMP. Use anything you wish for a TRA or entry address, since you are only trying to preserve the buffer. The results can be listed on the screen using LIST filespec, and examined with SCRIPSIT.

Another method designed by George Blank goes through the same procedures with more caution and better explanation. He writes to disk using a disk BASIC program. This method, from the August 1980 SOFTSIDE magazine, page 64, is excellent but for three errors in the program listing.

The corrected listing is as follows:

```
10 OPEN"O",1,"RECOVER/TXT:0"
20 FOR M= 32610 TO 32767
30 PRINT #1,CHR$(PEEK(M));
40 NEXT M
```

```
50 FOR M= -32768 TO -28447
60 PRINT #1,CHR$(PEEK(M));
70 NEXT M
80 PRINT #1,CHR$(0)
90 CLOSE
```

The above listing has corrections to the OPEN facility in line 10. The original had a comma after OPEN. The quoted "O" in line 10 was a numeral in original, it should be a capital O. The same problem also existed in line 80. It must have a capital O to write a file to Scripsit. If you write a number 'zero' instead, Scripsit won't read it.

In Mr. Blank's article he recommends using an RSM disk and using the distribution disk loader. For my purposes I have been able to use a system disk with BASIC, this RECOVER/BAS program and an RSM program somewhere in my disk system. As written, the procedure is as follows.

1. After 'booting' or whatever causes you to lose control of the buffer, load RSM into high memory.

2. Find the buffer start and end using the ASCII display (the 'A' command in RSM) and record it. After looking at the contents, you may not care to recover it. If you do, convert the start and end addresses to decimal peek addresses.

3. Place your disk with the RECOVER/BAS program on it in drive 0 and go to DISK BASIC. Set Memory size to just below the beginning of the buffer. If you use ?&HXXXX for each of the addresses, your TRS-80 will give you a correct peek address which will be correct for locations 8000H and above. Load RECOVER/BAS. Edit line 20 and replace the first decimal number with the decimal address of the start of the buffer, or where ever you want to start. It can be the first letter if needed. Edit line 50 and replace the second decimal with the ending address.

4. Check to see that you have the recovery disk with sufficient space for your recovered program in drive '0'. Then RUN the program. It should write the buffer between those two addresses to disk as filespec=RECOVER/TXT:0. Check your directory while in Disk BASIC. The operation is slow, but very handy. The first address in line 20 and the second address in line 50 were for one of my programs. Now reload SCRIPSIT/LC or whatever you prefer and reload your recovered buffer. Somewhere along the line, RENAME the recovered program or kill it from your Recovery Disk.

This method works for SCRIPSIT, and the ST80 series of programs. It should be useful for any buffer recovery and later examination.

For memory testing, I use the TEST1/CMD program supplied with TRSDOS 2.3 because it gives the Z number of the errant

chip. Many times I use another memory test to start with, but then use TEST1/CMD to find the location on the CPU board or in the interface. This recovery method works as well in TRSDOS as it does in NEWDOS 80.

Joseph X. Brennan, P.O. Box 302, Upland, CA 91786
(714) 982-5178

Another Version of Life

by
David Cornell

The attached program is yet another version of LIFE. While there is some question in my mind as to whether the world is waiting for yet another version of LIFE, I've had this program for some time and it may be of some interest.

The program is written in a combination of BASIC and machine language. The program requires a MEM SIZE of 43000. For systems with 64K (or 48K of user memory -- however you wish to describe it) the machine language routine will fit between the stack and the BASIC program and if you forget to protect high memory, there should be no problem.

```
1 '  - YET ANOTHER VERSION  OF 'LIFE'                    THIS ONE
BY DAVID CORNELL
100 ' THIS VERSION IS FOR A DISK SYSTEM
200 '    AND REQUIRES A MEM-SIZE OF 43000
300 ' IF YOU HAVE 64K IN YOUR TRS-80, AS A PRACTICAL MATTER
400 '  YOU CAN FORGET ABOUT MEM-SIZE AS THE MACHINE LANGUAGE
500 ' CODE WILL FIT IN BETWEEN THE STACK AND THE BASIC PROGRAM
600 '
700 CLS
800 DEFINT A-Z
900 SP=5
1000  CLS:PRINT"THIS  PROGRAM  DEMONSTRATES 'LIFE' ON A  16 x 16
FIELD"
1100 PRINT"'LIFE' IS NOT A GAME, BUT IS A SERIES OF PATTERNS."
1200 PRINT"EACH  GENERATION  BEING  DERIVED  FROM  THE  PREVIOUS
GENERATION
1300 PRINT"ACCORDING TO A NUMBER OF RULES"
1400  PRINT"PATTERNS  MAY  HAVE  INTRINSIC  PROPERTIES  SUCH  AS
STABILITY,"
1500 PRINT"OR OSCILLATION  OR  THE FACT  THAT THEY  JUST LEAD TO
OTHER"
1600  PRINT"PRETTY  PATTERNS.     a   FEW  INITIAL  PATTERNS  ARE
INCLUDED                          IN THIS PROGRAM:"
1700  PRINT"THE  GLIDER  MOVES  ON  A  DIAGONAL.  THE  SPACESHIP
HOROZONTALLY"
1800 PRINT"THE SNOWFLAKE  STABILIZES AFTER  15 GENERATIONS,  AND
THE"
1900 PRINT" SEED PRODUCES A REPEATING OR PULSATING PATTERN"
```

```
2000 PRINT"    FOR  A  COMPLETE DISCUSSION OF  'LIFE'  SEE  BYTE
MAGAZINE"
2100 PRINT"                    DECEMBER 1978"
2200 PRINT:PRINT,,"PRESS ANY KEY TO CONTINUE";
2300 IF INKEY$=""THEN2300
2400 CLS
2500  PRINT"ON THE LEFT OF THE SCREEN IS  THE GENERATION COUNTER
   ON THE RIGHT IS THE POPULATION COUNTER"
2600 PRINT"'C' CANCELS * UNDER CURSOR"
2700 PRINT"'ENTER' ENDS INPUT & STARTS GAME"
2800 PRINT"ARROWS MOVE CURSOR"
2900 PRINT"'S' TO CHANGE SPEED"
3000 PRINT"'P' TO ENTER PROGRAMMED PATTERNS"
3100 PRINT"ALL OTHER KEYS PRINT A '*'"
3200 PRINT"PRESS ANY KEY TO END GAME"
3300 GOSUB 6200
3400 GOSUB 12100
3500 PRINT:PRINT,"PRESS ANY KEY TO BEGIN";
3600 IF INKEY$="" THEN 3600
3700 'PRINT:GOSUB 270
3800 CLS:PRINTCHR$(23);:POKE15836,46
3900 A$=INKEY$:PRINTCHR$(14);
4000 IFA$=""THEN3900
4100 X=ASC(A$)
4200 IFX=91THENPRINTCHR$(27);:GOTO3900
4300 IFX=8PRINTCHR$(24);:PRINTCHR$(24);:GOTO3900
4400 IFX=9THENFORI=1TO4:PRINTCHR$(25);:NEXT:GOTO3900
4500 IFX=10PRINTCHR$(26);:GOTO3900
4600 IF X=83 GOSUB 5100: GOTO 3800
4700  IF X=13 PRINT CHR$(15);:GOTO 5500          : REM TURN OFF
CURSOR & BEGIN
4800 IFX=67PRINTCHR$(32);:PRINTCHR$(24);:GOTO3900
4900 IF A$="P" GOSUB 6800: GOTO 3900
5000 PRINT"*";:PRINT" ";:GOTO3900
5100 INPUT"SPEED  (1 IS FASTEST)";SP
5200 SP=SP-1: IF SP<0 THEN SP=0
5300 CLS
5400 RETURN
5500 'INITIALIZE, READ SCREEN
5600 X=USR1(0)
5700 'CALL GAME CONTROL AT 43000
5800 X=USR2(0)
5900 FOR I=1TO SP*50:NEXT
6000 X$=INKEY$:IFX$<>""THENCLS:GOTO 3800:ELSE 5800
6100 REM INIT PATTERNS
6200 PP=0
6300 READ X$
6400 IF X$="END" RETURN
6500 PP=PP+1:PT$(PP)=X$
6600 GOTO 6300
6700 REM PATTERNS
6800 GOSUB 7700
6900 CLS: PRINT CHR$(23);
7000 IF PT$(PT)="GLIDER" GOSUB 8500
7100 IF PT$(PT)="SPACESHIP"GOSUB 9100
7200 IF PT$(PT)="SNOWFLAKE" GOSUB 9800
```

```
7300 IF PT$(PT)="SEMAPHORES" GOSUB 10500
7400 IF PT$(PT)="SEED" GOSUB 11400
7500 RETURN
7600 REM LIST PATTERNS
7700 CLS
7800 FOR J=1 TO PP
7900 PRINT USING "###.   %              %"; J, PT$(J)
8000 NEXT
8100 PRINT,: INPUT "PATTERN NUMBER ";PT
8200 RETURN
8300 DATA  "GLIDER","SPACESHIP","SNOWFLAKE","SEMAPHORES","SEED",
"END"
8400 REM GLIDER
8500 PRINT@13*64,;
8600 PRINT"* * *"
8700 PRINT"    *"
8800 PRINT"  *  "
8900 RETURN
9000 REM SPACESHIP
9100 PRINT@5*64,;
9200 PRINT"  * * * *"
9300 PRINT"*       *
9400 PRINT"         *
9500 PRINT"*     *
9600 RETURN
9700 REM SNOWFLAKE
9800 K=476-4-64
9900 PRINT@K,"  *":K=K+64
10000 PRINT@ K,"*   *":K=K+64
10100 PRINT@K,"*   *":K=K+64
10200 PRINT@K,"* * *"
10300 RETURN
10400 REMSEMAPHORES
10500 PRINT@ 256,"* * *":PRINT
10600 PRINTTAB(22)"* *"
10700 PRINT TAB(22)"* *"
10800 PRINTTAB(22)"    * *"
10900 PRINTTAB(22)"    * *"
11000 PRINT:PRINTTAB(6)"* * *"
11100 PRINTTAB(6)"  * * *";
11200 RETURN
11300 REM SEED
11400 PRINT @ 7*64,;
11500 K=8
11600 PRINTTAB(K)"  * * * * *"
11700 PRINTTAB(K)"*     *     *"
11800 PRINTTAB(K)"  * * * * *
11900 RETURN
12000 REM INITIALIZE
12100 DATA  205, 78, 169, 205, 63, 169, 205, 95, 168, 205
12200 DATA  84, 168, 205, 62, 168, 205, 161, 168, 205, 210
12300 DATA  168, 205, 90, 169, 32, 245, 205, 236, 168, 205
12400 DATA  114, 168, 62, 2, 50, 175, 64, 201, 205, 55
12500 DATA  169, 205, 63, 169, 205, 75, 168, 1, 1, 0
12600 DATA  33, 0, 60, 17, 4, 0, 62, 42, 190, 204
12700 DATA  11, 169, 205, 90, 169, 200, 25, 3, 24, 242
```

```
12800 DATA  1, 255, 255, 175, 95, 221, 33, 104, 169, 205
12900 DATA  161, 168, 201, 229, 33, 0, 0, 34, 101, 169
13000 DATA  225, 201, 229, 42, 101, 169, 35, 34, 101, 169
13100 DATA  225, 201, 0, 229, 33, 0, 0, 34, 99, 169
13200 DATA  225, 201, 229, 42, 99, 169, 35, 34, 99, 169
13300 DATA  225, 201, 205, 125, 168, 17, 46, 60, 42, 99
13400 DATA  169, 24, 6, 17, 2, 60, 42, 101, 169, 237
13500 DATA  83, 32, 64, 205, 175, 15, 201, 62, 32, 40
13600 DATA  5, 62, 42, 205, 104, 168, 197, 217, 225, 43
13700 DATA  41, 41, 17, 0, 60, 25, 119, 217, 201, 84
13800 DATA  99, 175, 111, 95, 221, 203, 0, 70, 40, 1
13900 DATA  44, 3, 205, 44, 169, 32, 2, 221, 35, 246
14000 DATA  70, 50, 200, 168, 50, 169, 168, 197, 6, 3
14100 DATA  62, 254, 50, 199, 168, 221, 203, 0, 6, 40
14200 DATA  1, 28, 198, 2, 16, 242, 193, 201, 11, 122
14300 DATA  132, 87, 62, 15, 161, 122, 32, 4, 38, 0
14400 DATA  24, 1, 131, 149, 181, 254, 3, 32, 3, 205
14500 DATA  11, 169, 3, 201, 33, 134, 170, 1, 0, 0
14600 DATA  3, 205, 44, 169, 246, 70, 50, 1, 169, 230
14700 DATA  56, 32, 1, 35, 203, 70, 205, 139, 168, 205
14800 DATA  90, 169, 200, 24, 231, 205, 44, 169, 197, 11
14900 DATA  203, 56, 203, 25, 203, 56, 203, 25, 203, 56
15000 DATA  203, 25, 229, 33, 135, 170, 9, 246, 198, 50
15100 DATA  39, 169, 203, 198, 225, 193, 201, 0, 121, 61
15200 DATA  230, 7, 7, 7, 7, 50, 43, 169, 201, 1
15300 DATA  4, 1, 33, 103, 169, 24, 6, 1, 0, 1
15400 DATA  33, 135, 170, 11, 229, 209, 19, 54, 0, 237
15500 DATA  176, 201, 1, 0, 1, 33, 135, 170, 17, 105
15600 DATA  169, 237, 176, 201, 175, 229, 33, 0, 1, 237
15700 DATA  66, 225, 201, 0, 0, 0, 0
15800   FOR I=-22536 TO-22170:READ X: POKE I,X:NEXT I
15900 DEFUSR2=-22536
16000 DEFUSR1=38 -22536
16100 RETURN
```

Daisy Wheel II Review

by
Barney Dewey

Product: Daisy Wheel Printer II
Catalog Number 26-1158

Vendor: Radio Shack

Price: $1,960.00

GENERAL

The Radio Shack Daisy Wheel Printer II (DW II) is a letter quality printer. The DW II is only about the size of an office electronic typewriter. Like the other Radio Shack computer products it is colored silver and black.

TEC manufactures the DW II in Japan (the old DW I was made by Data Products). A Centronics parallel port interfaces the DW II with the computer. The DW II printer electronics use an 8085 microprocesor for control logic.

FEATURES

* Easy change "daisy" wheels for printing
* Prints at 43 characters per second
* 124 different printable characters
* switch (or software) selectable print densities of 10 pitch, 12 pitch or proportional spacing
* Spaces of various sizes down to 1/60"
* Accepts paper to 15" wide
* Optional tractor feed for forms
* Prints up to 163 characters per line
* Automatic carriage return/line feed if the line is longer than platen
* Ribbon out sensor
* Ribbon life of 270,000 characters
* Easy loading ribbon cartrige
* Switch selectable print impression and paper thickness adjustment (up to 6 sheets)
* Half line feed
* Backspace one character width (allows BOLDFACE characters and other special character printing)
* Self test
* Comes with Courier daisy wheel and one ribbon
* Speed optimizer (see below)
* Reverse line feed (see below)
* Automatic underlining (see below)

Optimizer. If the printer "sees" multiple space or control characters it buffers them and then executes them when the next character is received or 10 milliseconds of no data. This speeds up the "printing" of spaces. This is very noticeable when printing tabular columns (like assembler listings).

Reverse line feed. This provides great flexability in controlling the printer. A more descriptive term would be "cancel line feed".

"Automatic Line Feed" is one of Radio Shack's strange design philosophies. Radio Shack print driver routines send only a carriage return at the end of a print line. This requires the printer to do a line feed automatically whenever it receives a carriage return. Of course this means most printers can't make a double pass on one line to underline, etc.

The reverse line feed cancels the normal automatic line feed when a carriage return is encountered. This allows many passes on one line. Unfortunately, one can not turn the auto line feed off continually. A reverse line feed control character pair must be sent before each carriage return.

Of course, this feature wouldn't be needed if automatic line feed wasn't used in the first place!

Automatic underlining. To make underlining easy, automatic underline logic allows a single control character to be sent to set up an underlining field. Until a stop underlining control character is received all characters (ecxcept spaces) will be underlined. This allows underlining to be done simply and without two passes on the line.

AREAS FOR IMPROVEMENT

I have found a few areas where improvements could be made in the DW II printer.

1. The printer had no top of form feed. Therefore, one must keep track of where the printer is on the paper. Most word processing software takes care of this; however, many other types of programs don't. Radio shack must have realized this, but decided not to include form feed in the DW II printer. Instead, Radio Shack provides a "free" special driver program for Model I computers. I assume this special driver will keep track of the printer position in software.

2. The printer motor runs continuously. I prefer an automatic motor control so that it shuts off when not needed.

3. I have found no audible alarm. If one exists, it can't be accessed from software.

THE GREAT SOFTWARE GAP

Features such as automatic underlining, sub/super scripts, BOLDFACE, and 124 printable characters (see Figure 1 - Self Test print out) are necessary for "real" word processing applications. However, Radio Shack's SCRIPSIT (Model I) does not support any special characters or underlining. Additionally, SCRIPSIT (Model I) doesn't support proportional spacing with right justification.

The underlining feature of the ELECTRIC PENCIL can't be used with the DW II printer because the DW II does an automatic carriage return with each line feed. The ELECTRIC PENCIL takes two passes on one line to do underlining. This requires a printer without automatic line feed.

I feel that many of the features of the DW II (underlining, extended character set, and BOLDFACE printing) are so important to "real" word processing that I wrote a simple patch to access these features. Of course, all features of the DW II printer can be accessed using "CHR$(xx)" in BASIC.

DAISY WHEEL II AND OTHER COMPUTERS

I would beware of interfacing this printer to non-Radio Shack systems unless you can and will write the printer driver.

The DW II defines the functions of the ASCII control codes for its own use. Many of the characters in the expanded character set are printed with ASCII codes above 128. ASCII is only a seven (7) level code (maximum codes of 128). Therefore, if the eighth bit (bit number 7) is output by the printer driver (usually the eight bit is parity) undesirable characters may be printed. Radio Shack does refer to the ASCII codes used by the DW II printer as "modified."

Of course, your printer driver must account for the automatic line feed feature of the DW II. This along with the "modified" ASCII code may present a significant problem for non-Radio Shack systems.

SUMMARY

Some of the above information may unintentionally sound negative. Sincerely, I do not intend that. I have tried to present the important issues needed to evaluate the printer for your use.

I personally rate it extremely high. It appears to be the least expensive, full function, medium speed letter quality printer available today. It is rumored to be built to Radio Shack's specifications and that the manufacturer can not sell it to any others for two years.

Don't forget that a letter quality printer is very mechanical -- lots of moving parts. Access to service is necessary. The Radio Shack service support is worth some value. That plus the low price makes it extremely cost effective.

Although I criticize the current software support from Radio Shack, I feel condident that as new software comes to the market the DE II will operate with it nicely. The DE II has the important features required to be a "real" word processor/letter quality printer.

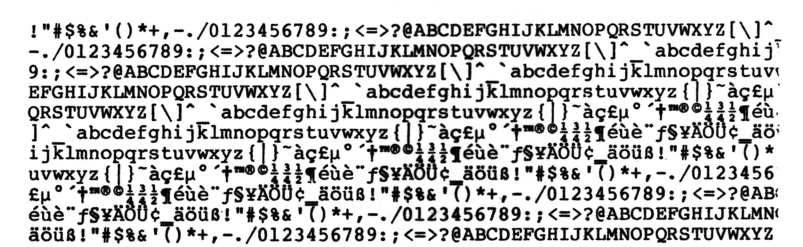

FIGURE 1

Orange County
TRS-80 ™
Users Group

2531 E. Commonwealth
Fullerton, CA 92631

First Class