IPSO FACTO

Issue #3
November 1977

(A publication of the Association of Computer Experimenters)

Table of Contents

Editor, de facto : Tom Crawford
Invaluable Assistants : Jo-Ann Van Bergen, Wayne Bowdish and all contributors to this issue.

All newsletter correspondence should be sent to:

Tom Crawford
50 Brentwood Dr.
Stoney Creek, Ontario
CANADA L8G 2W8

Editor's Remarks — November 24, 1977

I must begin this issue of IPSO FACTO with an apology. It is 2 weeks late, relative to my promise in the membership letter dated October 10, but I think the wait is worthwhile.

This issue is almost a special issue on cassette interfacing. There are 4 major articles on the subject. In addition, we are planning 2 more for the next issue: a PC board layout for a KC Standard interface, including a UART and some extensions to the Keyboard Editor to perform cassette I/O.

If you are interested in the Keyboard demonstrated by Pat Anthony at the first Club meeting, you will find sufficient information here to build one of your own. This design is reputed to solve all bounce problems, and in addition has acoustic feedback of keypress.

If you're tired of watching flashing LED's, build some of Fred Feaver's Hexadecimal readouts. They are an excellent example of utilizing existing devices in a slightly unorthodox fashion to arrive at a cost-effective solution to an existing problem (Whew! That means they're cheap and easy to build!). And when you get tired of these, you can have a go at connecting a TVT 6 to your TEC-1802. Some notes are in this issue; I expect more details to follow.

Some people have inquired about the CPU clock speed in "Music and Micro" in Issue #1. Read Ken Smith's article concerning the 2 grades of 1802 CPU's to find out if yours will run fast enough to stay in tune.

Finally, there is a note from one of our more distant club members, who lives in Texas, concerning his method of attacking the problem of insufficient software for the 1802.

The club executive recently burned the midnight oil to arrive at a Constitution. This Constitution was then accepted, with slight modifications unanimously by the members present at the Club meeting held on Nov. 24/77 at the Stelco Engineering Dept. Auditorium. One of the modifications was to change the official name of the club to the Association of Computer Experimenters (ACE). The full Constitution, as accepted, is printed elsewhere in this newsletter, for your reference.

We also discussed the possibility of setting up programming classes for interested people. Ken Smith was appointed by the Executive as Education Co-ordinator; his position was ratified at the meeting, as required by the new Constitution.

As a result, our next meeting will begin with an Introductory 1802 programming class at 7 P.M., with regular club business commencing at 8 P.M. Ken asks that only those seriously interested in solving their programming problems should attend at 7 P.M.

Our next meeting will be on Tuesday, January 10, 1978, at the Spectator auditorium (Main St. W., near Hwy. #403; the auditorium is just inside the main entrance). I understand there are no coffee machines, so you may want to bring your own refreshments. There will be no meeting in December.

We are in correspondence with several computer clubs in the U.S.; arrangements have been made to exchange newsletters with 2 of them: the Tulsa Computer Society (Oklahoma) and the COSMAC ELF Newsletter club

Editor's Remarks cont'd

(Texas). In addition, our Newsletter goes to BYTE, KILOBAUD, POPULAR ELECTRONICS and RADIO-ELECTRONICS. Did you see our write-up in November BYTE, pg. 212?

Postal Codes

I have been advised that Postal Codes will be required soon on all first and second class mail. Many members have not supplied their Postal Code; please check your mail label. If your code is missing, send us the corrected label ASAP. Thanks.

What Happened to ...

Someone suggested a Calculator chip interface. How's it coming? Want some help? How about a paper tape reader? Does anyone have one running? Please send the details. (I understand there is some software for the 1802 available on paper tape). Anyone running a CRT display? (Don't say no, 'cause I heard different). We would really appreciate an article on interfacing it to the 1802 (Even to an ELF; it can't be that much different from a TEC-1802. Besides, I understand we have a significant number of members with ELF's).

What about all you people who bought Memory Expansion Boards from TEKTRON? What software are you running in all that memory?

Any word on the Niagara-St. Catharines club?

Several people have shown active interest in using Wayne Bowdish's Cross-assembler (it runs on a PDP 11/10). I understand Bernie Murphy has his own running on an IBM System 370. Anyone else interested? (An assembler is a necessity for good software development).

Is anyone working on an S-100 bus card for the 1802? Please tell all.

What Might Have Been.

We hoped to have a review of Tom Pittman's Tiny BASIC for the 1802 in this issue, but 2 things held us back. The first was an ordering error; we received Tiny BASIC for the 6502 instead. On paper, however, the software looks good; it is definitely well documented. We'll see how it runs when the correct program arrives. Incidently, this software is supplied on paper tape only; see what I meant earlier about tape readers?

Another article we missed was coverage of TEKTRON's 7K RAM board. The board is not quite ready, and Eugene won't make promises he can't substantiate. I think you'll see it next time, though. I've seen it running and it looks good. (This is also the other reason we couldn't review tiny BASIC: insufficient memory).

The 1802D vs. The 1802CD — Ken Smith

One of the interesting properties of CMOS logic is its switching time and speed dependence on operating voltage. This is true for RAM's, ROM's, logic chips and CPU's. Is there any difference between the D and the CD chips for maximum clock frequency? Probably not, at least at 5V. The D chip is not really faster, it just has a higher voltage (15V) rating than the CD, hence the greater speed capability. If highest speed is to be obtained from the D chip, it will have to be run at 10V, no doubt about it. In my experiments, my CD chip ran up to 4.50 MHZ (at 5V) before it "crashed". Actually it doesn't crash, it just stops, and resumes operation when the frequency is lowered again. Does this mean that the music program at 4.4 MHZ would run on a CD chip? Probably on most people's CD chip, but there is no guarantee. The 3.2 MHZ figure from RCA is a guaranteed maximum. The absolute maximum will probably take the form of a normal distribution, if we measured several chips. One important thing; the clock waveform must have a 50% duty cycle since the 1802 uses both edges of the clock.

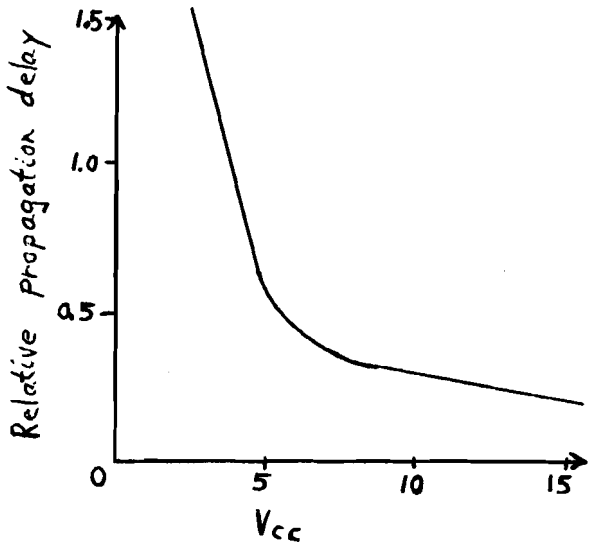


Figure 1.

On my 1802 board, I use a 3.58 MHZ colour burst crystal. This is the cheapest XTAL around. I would highly recommend using this frequency for 5V operation for those who want higher speed and more stable frequency than the 1 MHZ RC oscillator.

Figure 1 shows propagation delay (relative) vs. V_{CC}. Note that the delay is reduced by a factor of 2 when V_{CC} is increased from +5V to +10V. Compare to the 1802 speed of 3.2 MHZ @ 5V and 6.4 MHZ @ 10V. This graph is from a National CMOS data book.

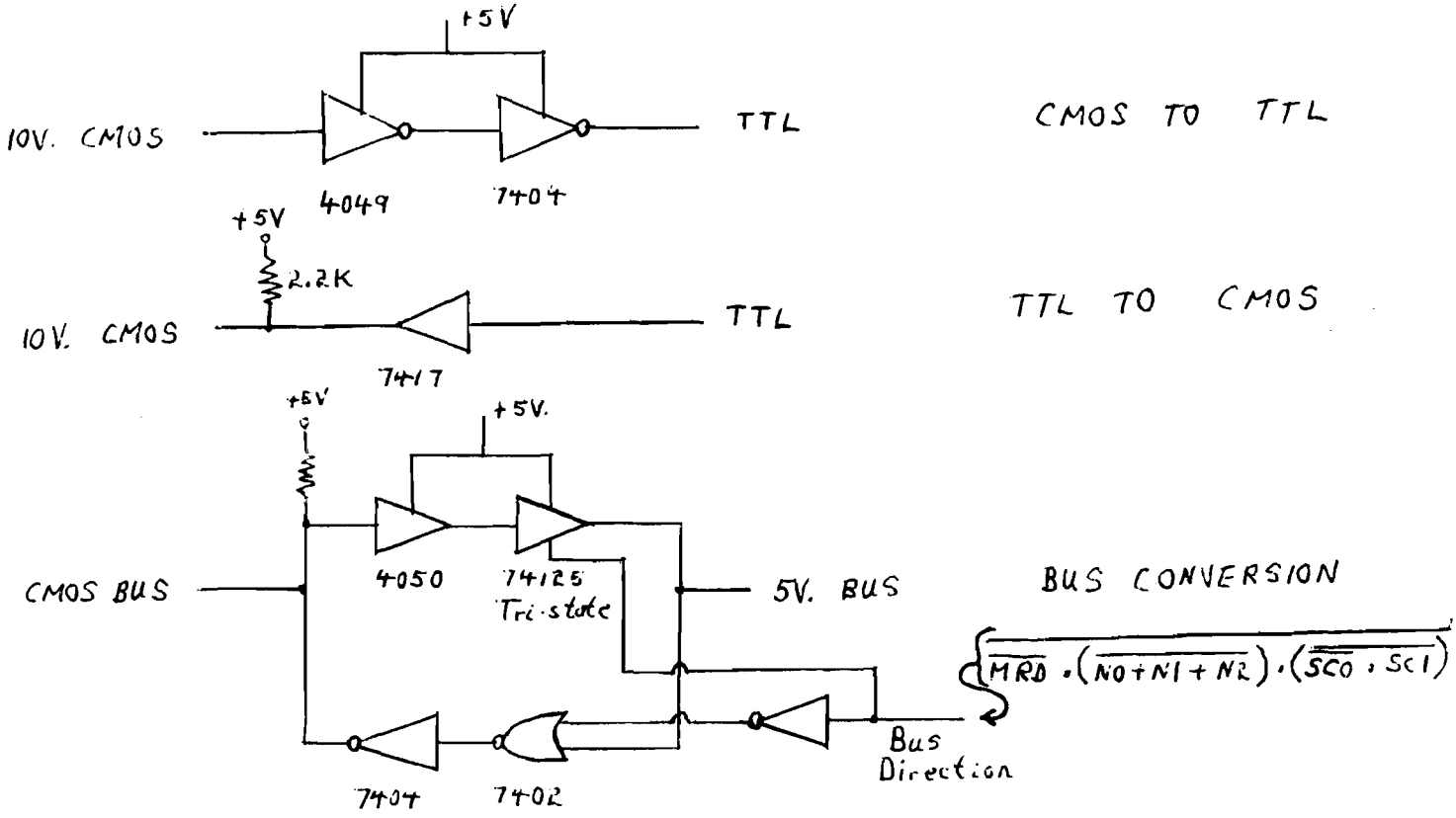
The 10 Volt Bus

To get more speed, we raise the voltage to 10V, but TTL and NMOS memories don't like 0-10V signals. We must do a voltage level conversion

cont'd

The 10 Volt Bus cont'd

when operating the 1802 at 10V and other boards at 5V. Where it really hurts is to convert a 10V bidirectional bus to a 5V one. Note that the effort to get higher speed is not worthwhile to the average person. 3.58 MHZ is good enough. RCA has on the drawing boards the 1802S, using SOS-CMOS technology with an instruction time of 1 μSec., probably at 10V. This implies a 16 MHZ clock! For this, 280 NS memories are needed.

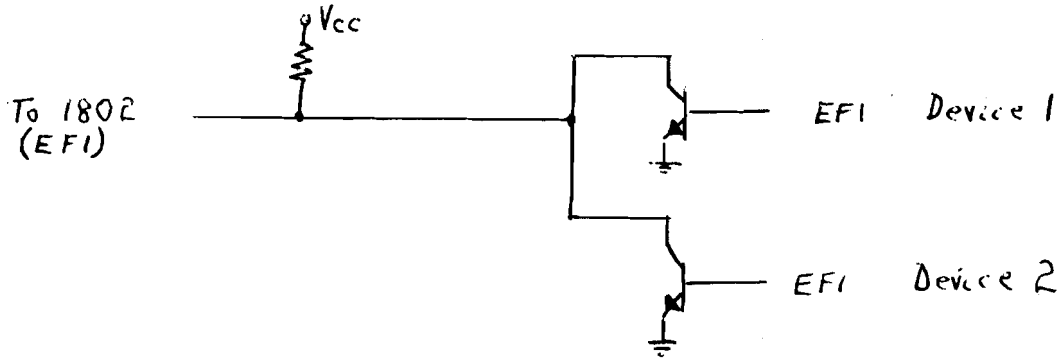


As one can see, the bus conversion is nasty business. In order to run the TEK-1802 board at 10V, the 2101 RAM's must be removed and the LED's would have to be isolated from the 10V supply and run at 5V (and the D chip used). Contrary to what RCA, Motorola and National say, CMOS will drive one standard TTL load without using buffers such as the 4049. When CMOS drives TTL low, the voltage is about ¼ V or less. Defective TTL which draws excess input current won't work.

Flag Re-Use

The four flags are quickly used-up in an 1802 based system. A possibility is to use the same flags for I/O devices which are not used at the same time eg. a cassette interface and an A/D converter. For using common flags, the flag lines from I/O devices can be OR-tied.

cont'd



I/O Device Independence

A problem in many systems is that there are several output devices which accomplish almost the same thing, the difference being in speed and giving hard copy or not. eg. 33 teletype, silent 700, TV display, etc. What device the user uses depends on the budget availability. When establishing an operating system, a definite advantage is to use software which is I/O independent. All the user must do is supply the proper interface. The 1802 flags are useful here. The device flag is high when the device is busy. The computer loops until the flag is reset, then transfers another character. I added a busy flag to a TVT-1, rather than using a 33 mSec wait loop in software.

(Ed. Note: It may be possible to raise the speed of a "D" chip to 6.4 MHZ by raising the potential of only V_{DD} (pin 40) to +10V. Leaving V_{CC} (pin 16) at +5V means that all I/O to the CPU chip is done at 5V levels, while internal logic runs at 10V. All level conversion is done inside the CPU chip. See RCA's CDP1802D Data sheet and specs. TC)

Letters to the Editor

(Ed. Note: The following letters have been edited slightly to conserve space. The Editor apologizes for any errors or omissions which result. TC)

Dear Mr. Crawford:

... I am a graduating senior in electrical engineering at the University of Missouri-Rolla. My 1802 system was started this summer, ... The system is based on multiple CDP 1802D processors in a master-slave multi-processor configuration. Separate processors are to be used for a terminal, disk interface and cassette interface. ... At the present time, I am developing some much-needed system software using an emulator written on a Microdata 1600 system. Some sort of operating system will be necessary eventually (!!! TC). My cassette interface is capable of using the Kansas City standard although I normally use ANSI bi phase encoding (often called Tarbell).

cont'd

Letters to the Editor cont'd

Is anyone there working on a dynamic memory interface for an 1802 running at 6.4 MHZ? ... One of the major problems with memory is the interface between the 10V processor and the 5V memory ...

Dennis R. Keats, Box 212 RHA, Rolla, MO 65401

(Dennis, it sounds like you've been quite busy. Your letter hints at many intriguing things, and raises a few questions, such as: What kind of cassette drive and software are you using? What cassette file structures have you considered, and which will you be using? What language is your emulator written in, and is it transferable to say, a PDP 11/10 without too many problems? Will you write an article for us covering the design and operation of your emulator? I'm afraid I don't know anyone working on dynamic memory (yet), but keep in touch. Thanks for your letter. TC)

Hi Tom:

... Thanks for the newsletters, they are among the best I have seen. The original 1802 newsletter by Ed Robertson in N. Carolina (Triangle Area Computer Club. TC) lasted 2 issues and really didn't provide much information. Enclosed are a couple of reprints from that newsletter.

My system is patterned after the ELF that appeared in Popular Electronics. I currently have: Hex Readouts, Hex Keyboard, 256 byte keyboard monitor in ROM, 1K static RAM and a cabinet. I am working on a Cassette Interface, a Polymorphics video interface, and an ASCII keyboard. (Gary sent a picture; its a good-looking cabinet. TC)

I plan to use dual 22 pin and S-100 boards. The video board is S100 and at $25 for an 8K memory board it is too good to pass up. The cassette board I laid out myself based on the circuit in March and April '76 BYTE by Percom Corp.

I sent copies of the Percom board and data as well as a schematic of an 8K memory. (Please see me if interested. TC)

One of the hardest things to do when you "roll your own" micro is to picture how it should go together when you expand. ... In other words, do it right the first time and you won't have to rewire ... If anyone is interested in the hex keyboard monitor, I will sell a 1702 PROM with the monitor in it for $10. I won't need it any longer, as I am going to the ASCII keyboard. It is being wired as in May 77 BYTE.

A friend of mine wrote the hex monitor as well as an ASCII monitor, cassette loader program, and video monitor. He currently has 8K of RAM, a video display and cassette interface, and is running Tiny BASIC.

... One thing I am sure you have learned already and that is that you could put out a 200 page newsletter every month and still not satisfy everyone. If addresses and work and home phone numbers (with permission) are printed, individuals can contact one another and not be frustrated waiting for the next newsletter. (How about it? TC) ... Please advise as to my monetary responsibility for the newsletter.

Sincerely, Gary Banko, Willowbrook Apts. 4B
Fishkill, N.Y. 12524

Mr. Tom Crawford,

Today I received the material you sent me. I was very impressed with all the interest and effort it takes to undertake a club, a newsletter, and everything else associated with keeping things together. Congratulations; and thank you very much for sending the material.

I am enclosing some with this letter, also. Last summer I decided it was time for me to get involved in uC's myself. About that time a computer club was being formed at work (Litton Industries, Van Nuys, CA), an article in Popular Electronics covered a minimal system design for a uP (the 'ELF'), and an article in Aug. 76 BYTE of Suding's TV Interface. Since I had just completed my first CMOS design, and was highly enthused with CMOS, the natural choice was to start with the 1802. Ordering the 1802 was easy; waiting for it took from Sept. til the end of Jan. 77. Fortunately, we had an 1802 evaluation kit at work which I used the uP chip to checkout my CPU board. I have since built a simpler, smaller UC, not too unlike the ELF, except that it has a video i/F (26 char X 12 lines --using a color crystal, 3.579 MHZ, which limits the horiz. resolution to about 26 char.).

Last November I wrote letters to several publications trying to find 1802 enthusiasts. Well the April 77 BYTE paid off; see pgs. 116,7. I shortly started getting letters from all over the country. So decided to make up a list of names/addresses and distribute to all who had written to me. Enclosed is a much updated list.

I have built into a 5" attache case the following: an ASCII keyboard, the encoder for it (my CMOS design, using about 35 microamp! Yes!), a CPU card with the uP and 1 k RAM (21L02's) and sockets for 1 K PROM (1702's) and a video interface with 32 char. X 16 lines, plus a 4 k RAM (purchased from S D Sales, Texas), and a homebrew power supply. Have not gotten too far with the operating system yet, as writing programs without an assembler and then burning EPROM's, then trying to debug; well, you can understand the turnaround time and fuss. So, decided to build the ELF which has allowed easier writing/debugging. So, I can start putting all the software together. As soon as I get a tape loader and the operating system working, I will start using Tiny Basic. Enclosed also is some info and 1802 Basic. I've purchased it, but have yet to make use of it.

Regarding the computer club at Litton, everyone seemed interested at first. But it didn't satisfy them, as there were too many wasted meetings on the wrong emphasis; like the club charter, too many sessions were consumed in getting it formalized; further it was run by the programmers, and to get any decisions on 'what hardware to get' ended up like the proverbial 'design done by committee'; and for me it moved too slowly. So I decided to go my own way, because at the time there weren't too many interested in the 1802, some would even make fun of you for mentioning the 1802, as they were used to 8080 archetecture/instructions and were inflexible in acknowledging any advantages of the CMOS 1802. True, had the 8080 been CMOS, or better the Z-80, I would have really liked that; I've done some programming with the 8080 and like it. I expect soon to see some fantastic uP chips coming out, maybe with VMOS, probably with 16 bit instructions, etc. I wish also the 1802 were much faster; see pop. elec. recent issue of Don Lancasters' TVT-6 design using the Up to handle most of the timing.

cont'd

... Next, will be a cassette interface design; yup, CMOS everywhere possible. Then need to write a program to handle the format/deformatting. (KC Standard, I hope. TC)

Do you know of anyone with an assembler? I've been thinking of writing one, but know that is quite an undertaking; even a dis-assembler is, and that is the easy direction.

By the way, in one of the 'letters to editor' a reference to a tape reader was made. I have a comment; there are several inexpensive readers, hand pulled, optical types available. One, I recently purchased is quite nice; RAECO is the company, for about $35, assembled. Has been listed in BYTE and several other magazines.

I could go on for pages. However, there are a few other things to accomplish this evening. I feel the letter is rather fragmented in many thoughts. But wanted to respond and let you know a bit of my situation/ accomplishments. If I hadn't sent out all the polaroids of my uC, I'd have included one this mailing. If possible, I would appreciate somehow getting your newsletter; do you plan any 'subscriptions'? Please let me know.

Until next time, I will close for now. Best wishes on your effort.

Sincerely, Harley A. Shanko
15025 Vanowen St., Apt 209
Van Nuys, CA 91405, USA

(Thanks for your letter, Harley. How does that 4K RAM board work? I understand the price is quite low. I would also like to know what clock frequency you are using. Would you care to share some of your hardware designs? They sound like excellent material for some newsletter articles. In return, I can point you towards an assembler. It is a cross-assembler, written mostly in Fortran, and runs on a PDP-11. Write to Wayne Bowdish; his address was given in the 2nd newsletter, which you should have received by now. He plans on producing a resident assembler, as soon as someone around here comes up with enough memory to run it in! TC)

Dear Tom:

I heard about you from Les Solomon at Popular Electronics. I've built the Neutronic's ELF II computer and would like to join your users' group for the RCA 1802. Would love to obtain your publication "IPSO FACTO", sounds just great. Could you please send me a copy, a membership application, anything you have, I'll send any money required by return mail.

Very truly yours, Russel Rhine
52-82 73rd St.
Maspeth, N.Y. 11378

Dear Tom:

Terry Wolfe of N.Y. says that you may be putting out some publications/ programs/etc. for the ELF. If this is the case, I would appreciate finding out about them, and would certainly forward any remuneration requested for same.

Sincerely, Michael Rutkaus, Siler Rte.
Box 396, Winchester VA, 22601

P.S. My daughter and I built the ELF II.

Letters to Editor cont'd

Dear Tom:

I am very grateful for the literature you sent me, I found it most interesting and informative. Your group has a lot of very good ideas and I hope they continue to do so. In your letters column (issue #2), Mr. Skodny mentioned a need for an inexpensive hex read-out. This is a good idea, but on page 122 of the March BYTE, James Hogenson describes a display which is more versatile. The sixteen digit display would be inexpensive and handy as an output device. It is programmable and can be used as the output for a number cruncher, (in decimal), or to monitor various states of your computer and provide data on registers in use, present memory address, etc. I noticed that your group plans to interface their computers to a calculator chip. Great idea. I have been thinking about this for a while and have been unable to solve one major problem. It should be simple to have the computer control analog ports to activate the calculator chips keyboard inputs, but I have been unable to figure a way to have the calculator chips output available to the computer. (Any thoughts on this one out there? TC) ... The chip need not be used for math at all. It could be used to increment a counter that acts as a real time clock, or to control counters for various computer games. Someone with greater hardware knowledge than I may be able to come up with an answer that would enable the chip to be used with a great deal more versatility.

Some of the projects of your members are what I have been thinking about for a while. The memory expansion, 4K Basic, and especially the monitor program that you mentioned (TECBUG). When my system is completed, as far as hardware is concerned, I plan to have 12-16K of RAM, a monitor program in ROM, a sixteen digit programmable display, a pixie graphics display, and an ASCII keyboard, running a version of BASIC (4K preferrably). Presently my system consists of a COSMAC ELF being built on a board supplied by Quest Electronics. I wish I had known about the TECTRON board before I started buying parts, I would have bought it.

I would like to receive your magazine and am willing to subscribe to it. Could you please send me a pinout diagram of the 22 pin edge connector attached to your board. (Done. TC) I would like to make my computer compatible with the other products from TECTRON and also those thought up by owners of the product.

Yours truly, David Brady, Box 353
Angus, Ontario, CANADA
L0M 1B0

Dear Tom:

A very interesting comment was made at the first club meeting regarding the fact that there are many people having difficulty with the fundamentals of programming, despite the course. I think this is discouraging many people and is a stumbling block for those wanting to do something with their system. It would be worthwhile to consider giving assistance to a group of people in the form of an informal gathering, employing those (like myself) knowledgable as assistants. It is a rather unfortunate fact that the 1802 is one of the most difficult micro-processors to learn that I have encountered, due to its unusual architecture ...

Ken Smith, 12 Sylvia Cres., Hamilton

(An excellent idea, Ken. I hereby nominate you to organize such an undertaking. Do you accept? You may count me in as one of your instructors (although not the only one, I hope! TC)

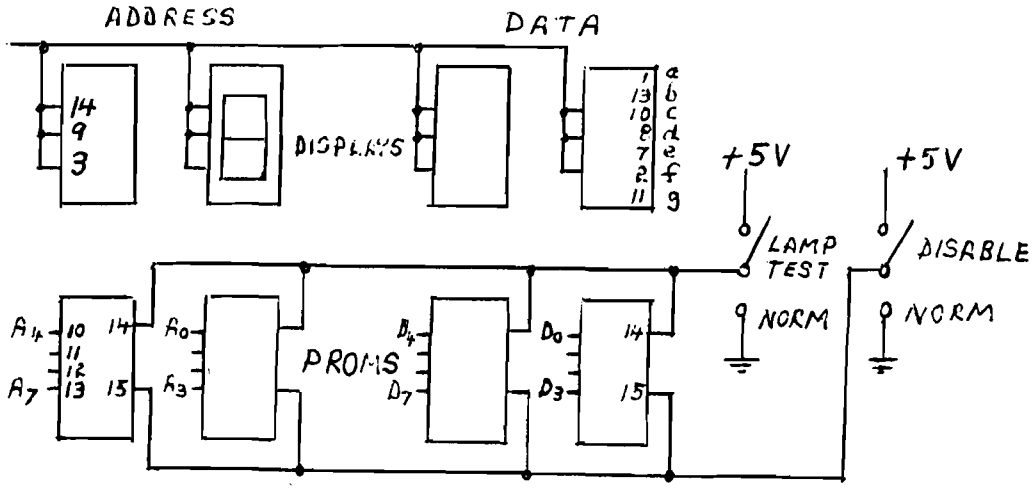
A HEXADECIMAL DISPLAY

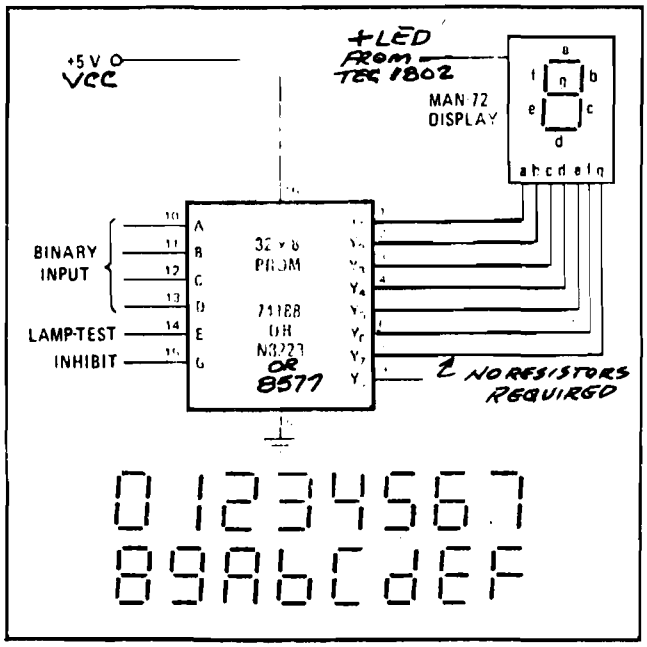
by Fred Feaver

A simple hexadecimal display is available from Hewlett-Packard in their 5082-7359 but this costs about $30. per digit and an inverter is required for each binary input to interface it to the TEC 1802.

A much less expensive system can be made from a Standard 7 segment LED display with common anode such as MAN-1 and a PROM (8223, 82S23, 74188 or DM 8577). The PROM must be programmed to invert the binary outputs from the TEC-1802 and to encode them for application to the 7 segment display. This involves approx. $7 per digit. For the 256 byte memory, 2 digits are required for memory and 2 for data. Multiplexing requires more I'Cs.

A connection diagram and truth table are given below:





TRUTH TABLE AND PROGRAM FOR THE HEXADECIMAL DISPLAY

INHIBIT	LAMP TEST	D B8	C B4	B B2	A B1	DISPLAY	PROGRAM IN MEMORY Y1 a	Y2 b	Y3 c	Y4 d	Y5 e	Y6 f	Y7 g	Y8
0	0	0	0	0	0	0	0	0	0	0	0	0	1	X
0	0	0	0	0	1	1	1	0	0	1	1	1	1	X
0	0	0	0	1	0	2	0	0	1	0	0	1	0	X
0	0	0	0	1	1	3	0	0	0	0	1	1	0	X
0	0	0	1	0	0	4	1	0	0	1	1	0	0	X
0	0	0	1	0	1	5	0	1	0	0	1	0	0	X
0	0	0	1	1	0	6	0	1	0	0	0	0	0	X
0	0	0	1	1	1	7	0	0	0	1	1	1	1	X
0	0	1	0	0	0	8	0	0	0	0	0	0	0	X
0	0	1	0	0	1	9	0	0	0	0	1	0	0	X
0	0	1	0	1	0	A	0	0	0	1	0	0	0	X
0	0	1	0	1	1	b	1	1	0	0	0	0	0	X
0	0	1	1	0	0	C	0	1	1	0	0	0	1	X
0	0	1	1	0	1	d	1	0	0	0	0	1	0	X
0	0	1	1	1	0	E	0	1	1	0	0	0	0	X
0	0	1	1	1	1	F	0	1	1	1	0	0	0	X
1	X	X	X	X	X	(OFF)	1	1	1	1	1	1	1	1
0	1	X	X	X	X	8	0	0	0	0	0	0	0	X

1 = HIGH 0 = LOW X = DON'T CARE

Remember the hex symbol. Binary inputs to the PROM produce a seven-segment representation of hexadecimal-code symbols. The PROM costs about $3, and can be programmed quickly at practically no cost if an automatic programing machine is available.

(Ed. Note: Fred sent a cover letter with his article; I think you will find it as interesting as his displays. How about writing an article on your PROM programmer, Fred? TC)

Dear Tom:

I have written a few notes on my hexadecimal display and have included a connection diagram and truth table. I programmed my 8577 PROMS in a home made semi-automatic programmer - micro second timing and regulated currents and voltages are required.

I am presently working on a larger display to accommodate the full 65K of memory as well as the 8 bytes of data. I believe this is the lowest cost display available.

Yours very truly, Fred Feaver
105 Townsend Ave.
Burlington, Ontario
L7T 1Y8

Items for Sale

1) Computer System for Sale: RCA CDP18S004 Development System. The system is in working order, and consists of:
   - Card cage with 33 slot mother board
   - CPU and 1K bytes of RAM
   - Clock module; TTY Interface
   - Bus separators; Power supplies
   - Burned in PROM's containing the extended version of RCA's monitor program.

   The list price is $3200.; offers will be considered. Call Robert La Gat at (201) 455-4490 during the day, or write him at Allied Chemical Co., Information Systems and Services, Columbia Rd. and Park Ave., P.O. Box #1039R, Morristown, N.J., U.S.A. 07960. More information is available on this system from this newsletter's Editor.

2) Does anyone have an 1802 kit for sale used? There are several people looking for used kits - please let me know about them.

3) Wanted: an 1802D CPU chip (working), by trade for an 1802CD chip plus cost difference. Contact Ken Smith, 12 Sylvia Cres., Hamilton. Phone: 545-8149.

4) Wanted: The recent issue of Dr. Dobb's Journal (DDJ) which contains the article showing mods to SWTP CT-1024 (otherwise known as TVT-2). These mods provide for 64 characters/line and hardware scrolling. If you have this issue and will lend it to me, please call. Tom Crawford Phone: 662-3603 Thanks.

Alternate Keyboard System for TEC1802

by Patrick R. Anthony

This keyboard system offers foolproof data input without having to wait for key bounce settling time. It uses a standard keyboard scanning process, providing data direct from a BCD counter and a latch strobe via a 16 input multiplexer, scanned by the counter, and a high pass filter- upon key closure. The DMA strobe is delayed by the carry-out port of the counter until the end of each keyboard scan, to avoid data race problems.

To further aid accurate data input, the key strobe pulse is also used to trigger a flip-flop driving a 45 ohm speaker- providing a key stroke recognition.

In the case of the TEC-1802 system this circuit replaces the functions provided by:- 3/4(U23),U26,½(U16),U18,U20,1/6(U14),1/6(U22),¼(U19),Q1,D1, D2,D3,D4 and associated resistors, but separate control switches must be provided.

To faciliate assembly the remaining ½(U16) function is moved to the keyboard assembly. This allows use of the U16 socket for all of the keyboard connections, except data which is provided by U18's socket, via DIP plugs and ribbon cable. The DC supply can be picked up at either socket, but it is recommended that the speaker supply be returned directly to the supply source.

Note that the balance of U16 is used in the DMA delay circuit, along with ½(U20).

Two schematics are enclosed:- (1) shows the keyboard connected to the TEC-1802 and (2) shows the keyboard alone for parts identification and connection.

Some Notes on a TVT-6 to TEK-1802 Interface

by Tom Crawford

The TVT-6 is a very low cost hardware/software combination which allows the contents of a block of micro-computer memory to be displayed in ASCII on a TV set, using the micro-processor to generate most of the necessary timing. The TVT-6 was developed apparently for a 6502-based KIM micro computer by Don Lancaster; details of the unit can be found in July and August 1977 Popular Electronics magazines. What we want to do is use an 1802-based TEK-1802 micro computer instead of a KIM.

It will be necessary to re-write the software, including the special SCAN instruction. It will also be necessary to add some hardware to the micro computer, to provide 2 structures not presently available on a TEK-1802. One of these is a set of 16 address lines all available simultaneously to the TVT-6. The other is a set of bus drivers, 8 bits wide, to allow the CPU chip to be isolated from the RAM chips, while still enabling the RAM output. This is required to implement the "up-stream tap" on the data bus. It will also be necessary to provide an enable signal to activate the bus drivers we have added, for normal micro computer operation. Throughout these notes, I will be assuming you have read both parts of the TVT-6 article in Popular Electronics. If you don't have a copy of the articles, check with the local library. In Hamilton, back issues of Popular Electronics can be borrowed from the Reference Library on James St., or the Kenilworth branch of the Public Library. Please don't "permanently borrow" back issues; someone else may want them after you. All libraries have photocopy machines.

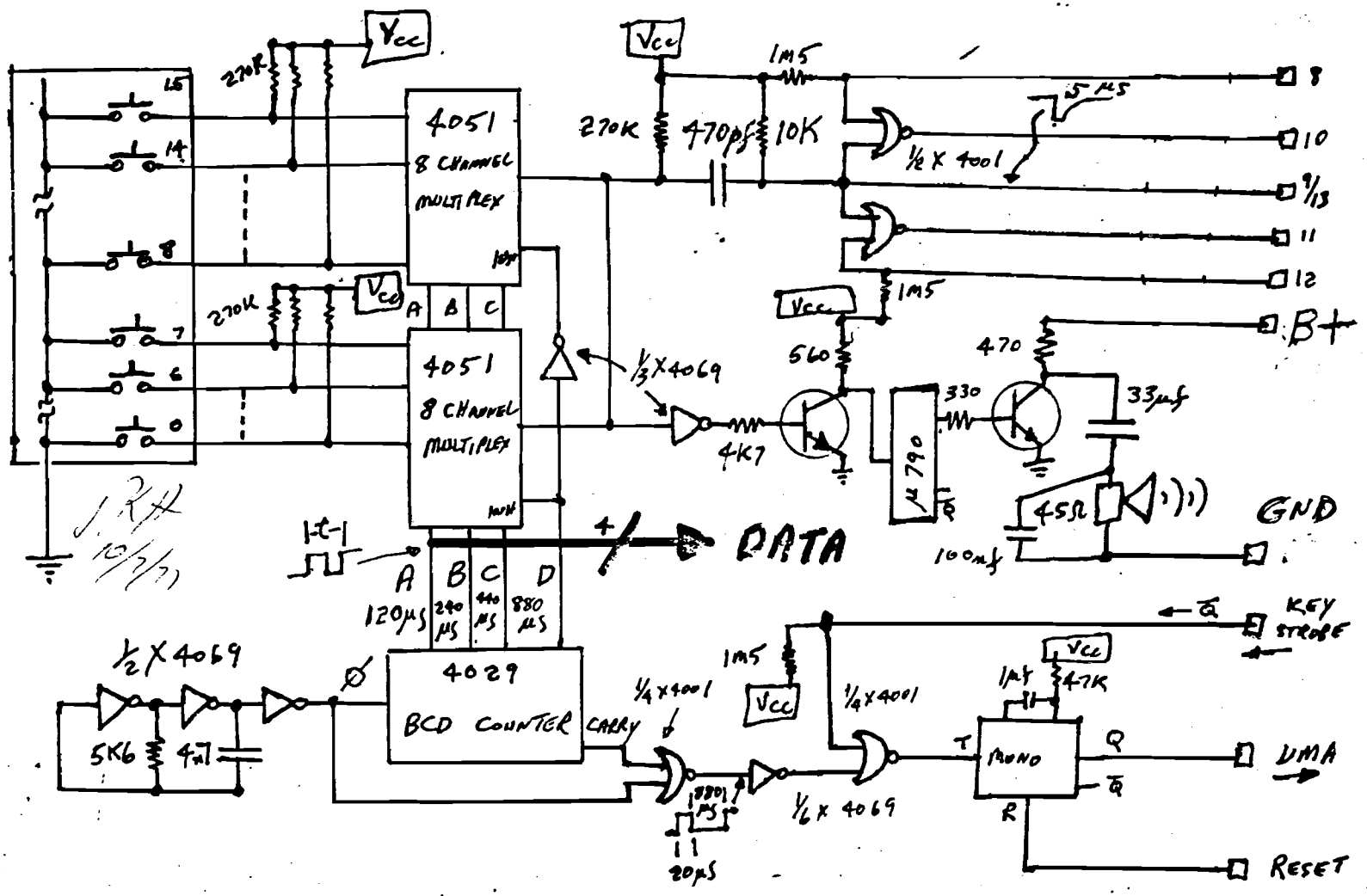
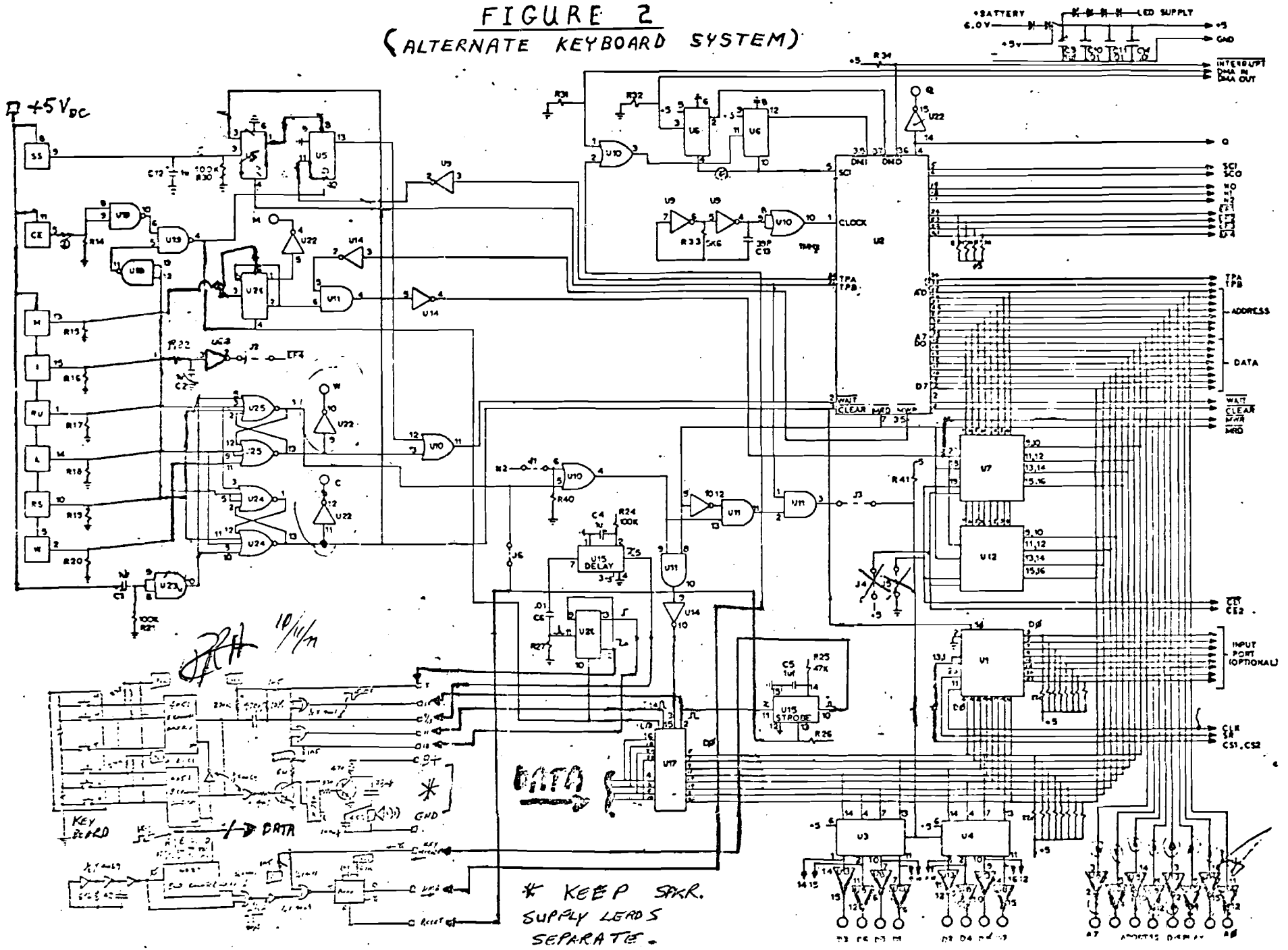


FIGURE 1

(ALTERNATE KEYBOARD SYSTEM)

FIGURE 2
(ALTERNATE KEYBOARD SYSTEM)



Let me be discouraging for a moment. I don't think this will be easy. Also be aware that the TVT-6 puts some limitations on your micro computer. It can do nothing else while the display is appearing (not as big a problem as it seems though). The TVT-6 is mapped into 24K bytes of memory space, leaving 40K bytes for normal programming. And you will have to run your micro computer as fast as possible (6.4 MHZ) to approximate the performance offered by a TVT-6 - KIM combination.

The primary consideration is the length of time required for 1 horizontal scan of a TV raster system. For 525 lines fully interlaced every 1/30 second, each horizontal line takes about 63.5 micro-seconds. Into this time you must squeeze memory addressing for all the characters you want to display per row, plus a horizontal synch pulse, retrace, and some calculations to determine what to display on the next line.

As it happens, the critical calculations can be done faster on an 1802 than in a KIM; 18 machine cycles VS the KIM's 26 cycles. While these calculations are going on, the TVT-6 hardware will look after the horizontal synch pulse and the retrace. We should now turn our attention to the special SCAN instruction.

For the KIM, the SCAN instruction consists of 15 LDY instructions, followed by an RTS (looking only at 32 characters/row for the moment). An LDY is normally a 2 byte instruction, which loads the data in the second byte into the CPU's Y-register. The PROM which contains the LDY instructions, however, ignores the least significant bit of the memory address; hence every 2 consecutive PC values from the CPU are mapped into the same location of the PROM. Therefore only 1 byte in the PROM is required to store a 2 byte instruction. The catch is that you must be satisfied with using the same binary pattern for the data byte as you used for the op code byte.

For the special SCAN instruction for use in the 1802, one of the Group 3 instructions can be used (refer to page 88-89 of the RCA CDP1802 Users Manual for details of Group 3 instructions). It must provide the necessary memory address line increment-by-1, at the rate of once per machine cycle. I suggest the LDI instruction ($F8_H$) be used. Note that this will load hex data F8 into the D-register each time the special SCAN instruction is used, destroying, whatever data was there previously. Since the 1802 does not have the equivalent of an RTS instruction, then the last instruction must be a SEP. If we assign registers at this point, then R14 can be the SCAN calling routines' PC, R15 can be the SCAN instruction PC, and therefore the last location of the PROM contains the instruction SEP R14. The other 31 locations each contain LDI (although only the last 16 instructions in the PROM are used for 32 characters/row.

The SCAN instruction in the KIM accesses 32 consecutive memory locations in 32 machine cycles using 15 LDY's and 1 RTS instruction. This means the RTS instruction must access 2 consecutive memory locations, even though the RTS is only a 1 byte instruction! This sounds impossible at first, but close examination of the 6502 architecture shows how it is done. The 6502 uses a technique known as "pipelining" to speed up execution. An RTS is a 6 cycle instruction. The first machine cycle is used to fetch the RTS instruction from memory. The second machine cycle does 2 things: it decodes the RTS instruction, and fetches the next consecutive memory location, even though it isn't required for an RTS instruction, and will be discarded. This pipelining of data into the 6502 CPU results in a single byte instruction (RTS) accessing 2 consecutive memory locations.

The SEP instruction we are going to replace the RTS with, only provides 1 memory access in 2 machine cycles; during the second machine cycle the memory address lines are in an undefined state. Therefore it will be necessary to add some extra hardware to detect this 32nd machine cycle (the second cycle of the SEP instruction), and to blank the TV screen when it occurrs. Otherwise, we will be displaying garbage characters in the 32nd character position of each row on the TV screen. This means also that we will be displaying only 31 characters/row when the TVT-6 is in the 32 character mode, and 63 characters/row in the 64 character mode. This is not a serious limitation; if you leave the TVT-6 in the 64 character/row mode, then the actual # of characters/row is under software control, and can be changed in increments of 2 characters from 1 to 63. I will discuss this in a later article, after we get the basic TVT-6 working as 16 rows of 31 characters/row.

Now it is time to look at the software which uses the special SCAN instruction to implement the TV display. Listing 1 shows a program which should produce the 16 row by 31 character display; we will get into more sophisticated programs, such as 16 X 63 character displays, scrolling, cursor control, etc., once this program has been tried out (and perhaps corrected). Some precautionary notes are in order at this point.

The program in listing 1 must run continuously during the time that you want characters displayed; if you stop it or interrrupt it, the TV display will go blank until you start it again. An interesting point to consider is this: if you could arrange to run the program just often enough to produce every second frame of the TV display, would you notice any difference? Would the display flicker? What precaustions do you need to take to maintain horizontal and vertical synch? If someone could resolve this, then the CPU's time could be shared; 50% for maintaining the TV display, and 50% for other work such as running BASIC programs, keyboard monitors, etc.

If you intend to obtain more memory to provide a place to store the characters to be displayed, consider this:

```
16 rows X 32 characters  =  512 bytes
16 rows X 64 characters  = 1024 bytes (1K)
32 rows X 64 characters  = 2048 bytes (2K)
```

Note that although we can only display a maximum of 63 characters on a row, the 64th byte must still be provided. You won't be able to display the data stored in every 64th byte, so you can use them for miscellaneous data storage. They look just like ordinary memory locations to the CPU. The same comments apply to every 32nd byte, if you restrict yourself to the 32 character/row mode of the TVT-6.

One nice advantage of the 1802 is that the program in listing 1 is pure code; it doesn't modify itself. Therefore the program can be stored in ROM, and doesn't have to be placed back into memory each time you turn the power back on. (Don't put it in ROM until you're sure it works right, though). This is not true of the software used to run the TVT-6 from a KIM (6502), a 6800, 8080 or Z80.

I have shown some suggestions for the TEK-1802 hardware additions and changes I feel are necessary, in Figure 1. Note that I have assumed the addition of an extra 1K byte of RAM to hold the characters to be displayed. This shouldn't require any changes to the CPU card other than those required to run the CPU at 6.4 MHZ.

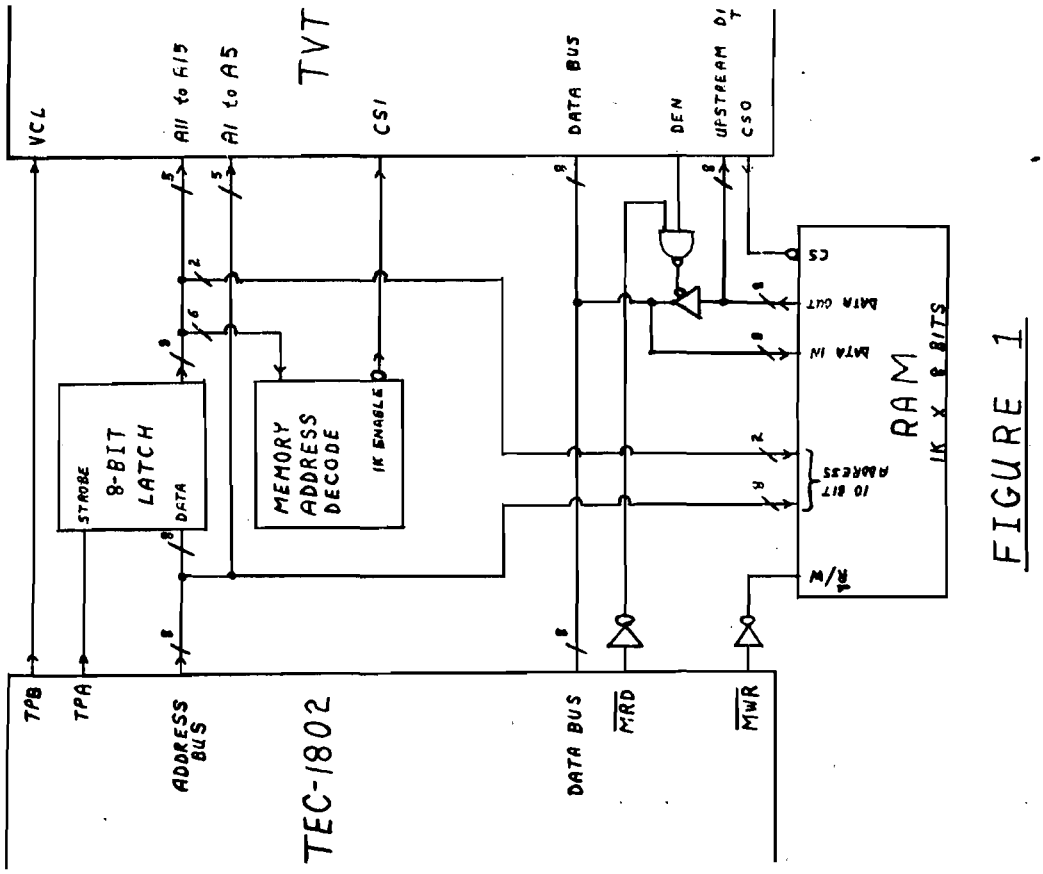


FIGURE 1


```
1                       .TITLE          TVT6-1
2
3
4                                   LISTING 1
5
6
7       MICRO  - RCA1802D   START - BR "START"    DISPLAYED  - X200-X3FF
8       SYSTEM - TEK-1802   END   - INTERRUPT     PROG. SPACE - 0100-015A
9
10
11              16 LINE X 31 CHARACTER/LINE TVT6 DISPLAY PROGRAM
12                      (FULLY INTERLACED RASTER)
13
14
15      NOTES:  1. THIS IS A TRANSLATION OF THE PROGRAM LISTED IN TABLE II
16                 ON PAGE 50 OF AUGUST 1977 POPULAR ELECTRONICS.
17              2. IT IS ASSEMBLED STARTING AT PC=0100, BUT CAN BE MOVED
18                 ELSEWHERE IF DESIRED. PROGRAM MUST NOT CROSS A PAGE BOUNDARY.
19              3. TVT6 MUST BE CONNECTED AND SCAN MICROPROGRAM PROM(IC1)
20                 MUST BE IN CIRCUIT FOR PROGRAM TO RUN.
21              4. TVT6 LENGTH JUMPER MUST BE IN "32" POSITION.
22              5. PC REGISTER UPON ENTRY MUST BE 14 (HEX E) IF PROM IC1
23                 CONTAINS "SEP R14" IN ADDRESS 31.
24              6. PROGRAM IS ENTERED AT LABEL START. R12 HI BYTE MUST BE
25                 INITIALIZED TO 00 HEX PRIOR TO ENTRY AT START.
26              7. CPU CLOCK MUST RUN AT 6.4 MHZ. 1 MACHINE CYCLE =
27                 1.25 MICRO-SECONDS.
28              8. THIS SOFTWARE IS SLAVED TO THE TV'S HORIZONTAL OSCILLATOR
29                 FREQUENCY. ALL LOOPS EMPLOYING THE SPECIAL SCAN INSTRUCTION
30                 ARE 50 CPU CYCLES LONG, AND THE SCAN INSTRUCTION MUST USE
31                 CYCLES #11 TO 44 INCLUSIVE (INCLUDING CALL) WITHIN EACH
32                 LOOP. ANY DEVIATIONS FROM THIS WILL CAUSE TEARING OF
33                 THE DISPLAY.
34              9. THE MEMORY SPACE TO BE DISPLAYED MAY BE THE UPPER HALF
35                 OF ANY 1K MEMORY BLOCK WHICH LIES IN NORMAL PROGRAM
36                 MEMORY SPACE. THIS BLOCK MUST BE CONNECTED TO THE
37                 "UPSTREAM TAP" DESCRIBED IN THE NOTES.
38              10.EACH ROW OF DISPLAYED CHARACTERS REQUIRES 10 LINES
39                 OF HORIZONTAL SCANNING. 264/265 LINES COMPRISE 1
40                 EVEN/ODD FRAME.
41
42
43
44
45      REGISTER USE:
46              R15 - PC DURING SPECIAL SCAN INSTRUCTION
47              R14 - MAIN LINE PC
48              R13 - WORKING STORAGE FOR R15 VALUE DURING BLANK SCANS
49              R12(HI BYTE) - INTERLACE FLAG (80 = EVEN FRAME,
50                                             00 = ODD FRAME)
51              R12(LO BYTE) - VERT. BLANKING LOOP COUNTER
52
53      .SLW
```

```
                                                                                    # CYCLES
                        ;
                        ;
                        ;
        = 01 00                 .ORG    #0100           ; ESTABLISH PC START VALUE
                        ;
0100  7C C0             LOOP1:  ADCI    #C0             ; RESTORE LINE POINTER                          2
0102  BD                        PHI     R13             ; PUT SCAN INST. HI ADDRESS AWAY                2
0103  BF                LOOP2:  PHI     R15             ; AND USE FOR SCAN                              2
0104  8D                        GLO     R13             ; GET SCAN INST. LO ADDRESS                     2
0105  AF                        PLO     R15             ; AND USE FOR SCAN                              2
0106  9D                        GHI     R13             ; GET HI BYTE OF REG. 13 INTO D.                2
0107  DF                        SEP     R15             ; /// CHARACTER SCANS 1-8///                    2 + 32
0108  9D                        GHI     R13             ; RESTORE D-REGISTER CONTENTS                   2
                                                          INCREMENT CHARACTER SCAN COUNTER; TEST
                        ;
0109  7F 88                    SMBI    #88             ; DONE 8 LINES YET?                             2
010B  33 00                    BPZ     LOOP1           ; NO.  GO DO NEXT LINE.                         2
                        ;                       TOTAL CYCLES = 50; =62.5 MICRO-SECS.
                        ;                       (FOR EACH PASS OF ABOVE LOOP)
010D  8F                        GLO     R15             ; YES.  EQUALIZE 2 CYCLES                       2
010E  F8 80                     LDI     #80             ; SET UP R15 FOR 2 BLANK LINE SCANS             2
0110  BF                        PHI     R15             ; FOR SPACE BETWEEN CHARACTER ROWS              2
0111  F8 00                     LDI     #00             ;   (= 8000 HEX)                               2
0113  AF                        PLO     R15             ;                                               2
0114  DF                        SEP     R15             ; /// CHARACTER SCAN 9 ///                      2 + 32
0115  8D                        GLO     R13             ; INCREMENT LO ADDRESS BYTE BY 20 HEX           2
0116  7C 1F                    ADCI    #1F             ;   (DF = 1 AT THIS POINT)
0118  AD                        PLO     R13             ;                                               2
                        ;                       TOTAL CYCLES = 50; = 62.5 MICRO-SECS
                        ;
0119  9D                        GHI     R13             ; INCREMENT HI ADDRESS BYTE BY 3; PLUS          2
011A  7C C8                    ADCI    #C8             ; CARRY FROM LO BYTE ADDITION; PLUS RESET       2
011C  BD                        PHI     R13             ; VS. BIT; SAVE RESULT                          2
011D  FA 04                     ANI     #04             ; IF CARRY SETS BIT 3 OF D, THEN GO TO          2
011F  3A 27                     BNZ     CONT2           ; CONT2, SINCE 16 ROWS ARE DONE.                2
0121  DF                        SEP     R15             ; ///CHARACTER SCAN 10 ///                      2 + 32
0122  9D                        GHI     R13             ; GET HI BYTE OF REG. 13 INTO D.                2
0123  7F C0                    SMBI    #C0             ; PRESET FOR ENTRY TO NEXT CHAR. ROW.           2
0125  30 00                    BR      LOOP1           ; AND GO TO START NEXT CHARACTER ROW            2
                        ;                       TOTAL CYCLES = 50; =62.5 MICRO-SECS (ROWS 1-16)
                        ;
0127  DF                CONT2:  SEP     R15             ; /// CHARACTER SCAN 10/// (IF ROW=17)          2 + 32
0128  9C                        GHI     R12             ; GET INTERFACE FLAG (BIT 8)                    2
0129  FB 80                    XRI     #80             ; TOGGLE IT                                     2
012B  BC                        PHI     R12             ;         AND RESTORE FOR NEXT FRAME,           2
                        ;                       TOTAL CYCLES = 50; =62.5 MICRO-SECS (ROW 17)
                        ;
012C  3A 35                     BNZ     CONT3           ; JUMP IF EVEN FRAME                            2
012E  F8 50                     LDI     #50             ; PREPARE R15 FOR ODD FRAME VERT. SYNCH         2
0130  BF                        PHI     R15             ;                                               2
0131  0F                        LDN     R15             ; /// ODD  FRAME VERT. SYNCH ///                2
0132  F8 67                    LDI     #67             ; LOAD SHORT # OF VB SCANS(104 DEC.)            2
0134  AC                        PLO     R12             ; PUT IN LOOP COUNTER                           2
0135  F8 50             CONT3:  LDI     #50             ; PREPARE R15 FOR EVEN FRAME VERT. SYNCH        2
0137  BF                        PHI     R15             ;                                               2
0138  C4                        NOP                     ; EQUALIZE 12 CYCLES                            3
0139  C4                        NOP                     ; CONT.                                         3
013A  C4                        NOP                     ; CONT.                                         3
013B  C4                        NOP                     ; CONT.                                         3
013C  9C                        GHI     R12             ; GET INTERLACE FLAG                            2
013D  32 45                     BZ      CONT4           ; JUMP IF ODD FRAME (INTERLACE = 0)             2
013F  F8 68                     LDI     #68             ; LOAD LONG # OF VB SCANS (105 DEC.)            2
0141  AC                        PLO     R12             ;                                               2
0142  8F                        GLO     R15             ; EQUALIZE 4 CYCLES                             2
0143  8F                        GLO     R15             ; CONTINUED                                     2
0144  0F                        LDN     R15             ; /// EVEN FRAME VERT. SYNCH ///                2
0145  F8 1E             CONT4:  LDI     #1E             ; PRESET R15 FOR SHORT VB SCAN                  2
0147  AF                        PLO     R15             ;    (SCAN INSTRUCTION IS                       2
0148  BF                        PHI     R15             ;                                               2
0149  DF                        SEP     R15             ; /// 1ST VERT. BLANKING SCAN ///               2 + 2
014A  C4                        NOP                     ; EQUALIZE 6 MACHINE CYCLES                     3
014B  C4                        NOP                     ; CONTINUED                                     3
                        ;                       TOTAL CYCLES = 50; =62.5 MICRO-SECS (1ST VB SCAN)
                        ;
014C  F8 00             LOOP3:  LDI     #00             ; PRESET R13 FOR TOP OF NEXT FRAME              2
014E  AD                        PLO     R13             ; (SCAN ADDRESS 8200 HEX ), AND                 2
014F  AF                        PLO     R15             ; RESET R15 FOR REGULAR VB SCANS                2
0150  F8 82                     LDI     #82             ;                                               2
0152  BD                        PHI     R13             ;                                               2
0153  DF                        SEP     R15             ; /// REGULAR VERT. BLANKING SCANS ///          2 + 32
0154  2C                        DEC     R12             ; DECREMENT                                     2
0155  8C                        GLO     R12             ;         AND TEST VB LOOP COUNTER              2
0156  3A 4C                     BNZ     LOOP3           ; BRANCH IF NOT DONE YET.                       2
                        ;                       TOTAL CYCLES = 50; =62.5 MICRO-SECONDS
                        ;                           (FOR EACH PASS OF ABOVE LOOP)
0158  9D                        GHI     R13             ; PRESET D WITH SCAN ADDRESS HI BYTE,           2
0159  30 03                    BR      LOOP2           ; AND GO START NEXT FRAME.                      2
                        ;
                        ;
                                .END
```


USER SYMBOL TABLE

```
CONT2  0127   CONT3  0135   CONT4  0145   LOOP1  0100   LOOP2  0103   LOOP3  014C

PROGRAM SIZE = 015B
  0 ERRORS DETECTED
```

TVT-6 to TEK-1802 Interface cont'd

In conclusion, I want to emphasis that at this time none of the ideas in this article have been tried out. I will be doing so shortly, and I will let you know in a future article how I fare. My intention is to make up a PC card with the TVT-6 circuitry, 1K of character display RAM, and all the necessary hardware additions required. This card will plug into the "Tekatch" bus, and will minimize the number of hardware changes to the CPU card. I understand some people already have the TVT-6 hardware; if you try out any of these ideas, please let me know if they work or not.

Incidently, my notes apply generally but not directly to the TVT-6L display presented in the June 1977 issue of KILOBAUD magazine. I do have some other comments on this display; contact me if you are seriously interested.

1802 Software Exchange

Ross Wirth

To all 1802 users: The 1802 Exchange.

Very little software for the RCA CDP1802 is currently in the public domain. To remedy this situation I am going to publish a ten page booklet listing available software. If you desire to sell or even give away your software please send me a listing for my review. My booklet will provide a complete description and cost information with a reference number corresponding to a number on an ordering coupon.

I plan to charge $1 for the booklet. This amount will also cover the costs associated with processing the coupons. The use of the coupon will reduce the costs to the person ordering from more than one source.

The publication date is set for early December. Advance orders may be made at $1 per copy. Send all orders, software listing, and other correspondence to:

Ross Wirth
1636 S. 108 E. Ave.
Tulsa, OK 74128

Here is your chance to buy a good selection of software as well as sell some.

(Ed. Note: Ross will be listing IPSO FACTO as a source of 1802 software. In addition, we will print some or all of his booklet, and if warranted by demand, we will try and arrange for distribution of some software through the Club and this newsletter. Good idea, Ross. TC)

CUTS - Computer Users Tape System

by Robert M. Marsh

(Ed. Note: The following article has been extracted from the "Computer Bits" column of the March 1976 issue of POPULAR ELECTRONICS. My thanks to Mr. Marsh. TC)

The lack of standardization is the bane of many industries. For example, three basic four-channel audio systems (SQ,QS, and CD-4), instead of a universal system, have impeded progress in that field. The same holds true for computer hobbyists, where a host of methods for exchanging programs or data have been introduced, including the HIT system published in POPULAR ELECTRONICS, September 1975.

Rather than stifle this user-created "program explosion" a group of hobby computer manufacturers and other interested parties (POPULAR ELECTRONICS, among them), met in Kansas City, MO, last November to explore standardization in general and hopefully to agree on a single method of recording data. There was general agreement that cassette tape represented the best route to go for a hobbyist computer-data exchange system. These tapes are low-cost and widely available, and cassette machines are owned by most people.

The use of inexpensive cassette recorders was not viewed as a serious limitation as long as the record/playback exchange method adopted allowed for certain inherent machine deficiencies. The two most common considerations with low-cost cassette machines are: (1) the automatic level control incorporated in some machines, and (2) variations in average speed, nominally 1 7/8 inches/second. Both drawbacks could be easily overcome, it was decided.

Another important consideration in using low-cost cassette tapes is that some tapes would likely cause drop-outs (momentary loss of signal) due to a lack of uniform distribution of oxide particles. At this time, the user would have to "certify" the best tape brand and model for him to use. There are also "data cassettes" certified by tape manufacturers. Prices are not too much higher than those for consumer premium tapes.

Cassette Data Recording Methods

Various methods have been used by computer enthusiasts and manufacturers to record data on audio cassette recorders. These fall into five categories: (1) simple tone burst, (2) pulse-width modulation such as used in the POPULAR ELECTRONICS HIT program, (3) frequency shift keying (FSK) as used in radio- teletypewriter or phone-line communications modems, (4) double-frequency pulse recording as used in most floppy disc systems, and (5) phase encoding as used in ANSI standard magnetic tape transports of all major computer manufacturers.

Most of these methods record data serially; that is, one bit after another. Serial recording requires a conversion from parallel to serial form (and vice versa) when used with a computer. Fortunately, most computers and terminals already have a standardized serial communications channel that transmits in a form called "non-return to zero" (NRZ), shown in Fig. 1A.

Cassette Data Recording Methods cont'd

Tone-burst (or cw) recording may be the simplest way of recording data, where data "1" is the presence of a tone and data "0" the absence of a tone, as shown in Fig. 1B. Because this system is basically an amplitude-modulation scheme, and very susceptible to noise, reliability suffers above 150 bits per second.

Pulse-width modulation may be recorded in its pure form (Fig. 1C) or as a burst of tone with varying duration, as used in the HIT system (Fig. 1D). Both methods are self-synchronizing and are highly independent of speed and amplitude variations. However, in the original HIT proposal, data was recorded synchronously so that each data word had to follow the previous word immediately, thus making HIT impractical for use with stand-along asynchronous terminals such as TV typewriters and teleprinters. In addition, "pure" pulse-width modulation is patented as a data recording method, which might be seen as a drawback by manufacturers.

Ordinary frequency shift keying (FSK), shown in Fig. 1E, is by far the most common method used to transmit data over phone lines and radio links. It would be a useful feature of a cassette recorder interface if it could transmit data over phone lines as if it were a FSK Bell-103 compatible modem. However, while FSK is fairly insensitive to AM noise and level changes, it is susceptible to loss of data when overall frequency changes exceeding $\pm$5% of the nominal value occur. The 5% frequency, or speed tolerance is not sufficient for reliable data storage on many cassette recorders. In addition, FSK is more expensive to implement than many other methods.

Double-frequency recording (DFR), shown in Fig. 1F, is often used on disc memories at high data rates. When used on a cassette, however, it requires a relatively high bandwidth for a given data rate. This method is insensitive to speed variation since each bit is self-clocked, but it is only moderately free from problems created by noise and amplitude changes. DFR is, therefore, not as reliable as other methods at data rates higher than 500 bits/second, making future expansion and improvement difficult.

Phase encoding has many variants and has been in use in many different types of magnetic tape data systems for many years. The most common forms are Bi Phase-L, usually called "phase encoding" and Bi Phase-M, often called "Manchester" code. Both methods are self-clocking and, at first glance, resemble simplified FSK. In fact, phase modulation does create a form of frequency modulation. All phase-encoded methods are independent of frequency changes over a wide range, and can be made highly resistant to AM noises and level shifts.

Bi Phase-L is shown in Fig. 1G. You can see that there is a transition in the middle of each bit cell and that the polarity of the transition determines whether the bit is a logic 1 or 0. Bi Phase-M, or Manchester, shown in Fig. 1H, has a transition at the beginning of each bit cell. Logical 1's have another transition in the middle of the cell, whereas logical 0's do not.

Manchester code is extremely easy to generate, decode, and synchronize, and is the basis for the CUTS (Computer Users Tape System) recording method proposed as an outgrowth of the meeting in Kansas City.

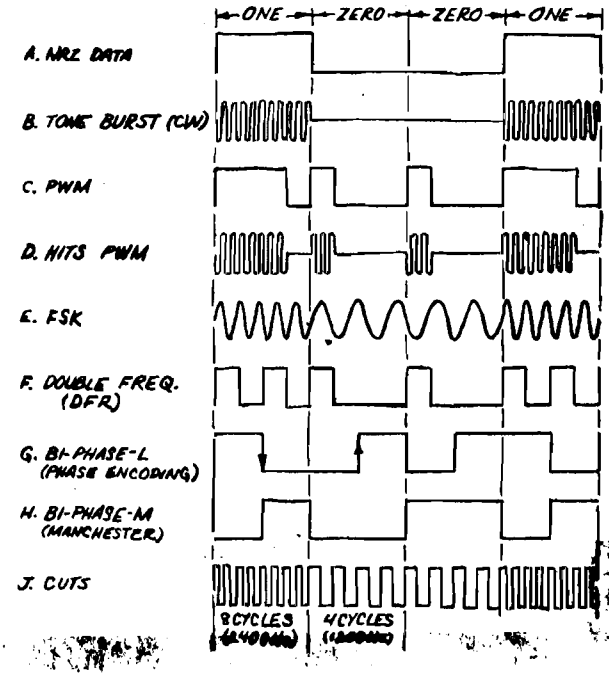


Fig. 1. Methods of recording data on cassette recorders.

COMPARISON CHART

	Level, Noise Tolerance	Frequency, Speed Tolerance	Self clock-ing	Cost*	Future Upgrading**	Remarks
CW (Fig. 1B)	Poor	Poor	No	Very Low	No	Susceptible to noise. Reliability suffers over 150 baud.
PWM (Fig. 1C)	Good	Very Good	Yes	Low	To 1500 Baud	Patented. Requires higher bandwith than Bi-phase for same baud rate.
HITS (Fig. 1D)	Good	Very Good	Yes	Low	To 600 Baud	Requires higher bandwidth than CUTS for same baud rate (see PE, Sept. 1975).
FSK*** (Fig. 1E)	Very Good	Poor-Fair	No	Moderate	To 450 baud (with Bell 103 tones)	Can be transmitted over phone lines to any modem.
DFR (Fig. 1F)	Moderate	Good	Yes	Low to Moderate	To 800 Baud	See "Computer Hobbyist" Vol. 1, No. 5, 6, 1975.
Bi-Phase-L (Fig. 1G)	Very Good	Very Good	Yes	Low to Moderate	To 1500 Baud	Proposed ANSI standard. Subject to phase inversion.
Bi-Phase-M (Fig. 1H)	Very Good	Very Good	Yes	Low to Moderate	To 1500 Baud	Widely used. Easily decoded.
CUTS (Fig. 1J)	Very Good	Very Good	Yes	Low	To 1200 Baud	Very easily decoded. Can be transmitted by phone to other CUTS units.

*Cost is estimated for devices which do not require a CPU for reading and writing. Cost may be quite low if CPU decodes mostly by software.

**Fututo upgrading implies minimal hardware modification at low cost.

***Especially Bell-103 compatible.

Cassette Data Recording Methods cont'd

The CUTS method employs a variation of the Manchester code in which a logical 1 consists of eight cycles of 2400 HZ, and a logical 0 is four cycles of 1200 HZ. A 4800-HZ clock is derived from the recorded data itself as the tapes are read, and is used to clock a UART (Universal Asychronous Receiver Transmitter) which performs the serial/parallel and parallel/serial conversions necessary to interface with the computer's data bus. It is not necessary to use a UART. Also, in some simple applications, a less expensive circuit can be used.

The standard data rate is 300 bits/second, and can be expanded to 600 or even 1200 bits/second with slightly higher error rates. Each bit is self-synchronizing because every bit time frame starts with a positive transition and contains an even number of tone cycles. Each data character is resynchronized by a logic 0 start bit that precedes the data bits. Therefore, data can be transferred asynchronously from any computer, terminal or modem with a serial data channel, as long as the serial channel is set up for 300 bits/second, eight data bits, and two "stop" bits.

Recording Method

The following specifications were adopted with the goal of optimizing versatility, reliability, low cost, and future expandability.
Mode: asynchronous by character.
Character Format: 11 bits; one start bit (a 0); least significant data bit first (if less than 8 bits are used as with Baudot 5-level code, then all bits not specified by the code will be set to 1). The interval between characters, if any, will be 1's.
Modulation Method: 1's will be 8 cycles of 2400 HZ; 0's will be four cycles of 1200 HZ tones. Sine-wave signals are preferred, although not always necessary.
Leader: Five seconds of continuous 2400 HZ (all 1's) will precede any block of valid data. At least 30 seconds of continuous 2400 HZ should be recorded at the beginning of each cassette. When multiple blocks are recorded, there will be a 5-second gap between them.
Motor Control: The interface should provide for switching the tape recorder motor so that the computer can start and stop the machine under program control.

... Based on observations made at the meeting, most manufacturers agreed to shelve their personal systems for the good of the industry, although some might still offer their systems, with CUTS made available as an alternative.

Cassettes and Computers

Tom Crawford

Cassette tape is one of the least expensive and most flexible means of mass program storage available to the computer hobbyist. It makes use of a common and well understood appliance: the portable cassette recorder. These units are available for as little as $25 each, and can easily provide 100,000 bytes of storage on a $5 tape, which can be written or read at a rate of 30 bytes/second.

The hobbyist needs to add 2 things to his cassette recorder in order to connect it to his computer. These are the hardware interface and the software driver routines. I intend to let someone else handle the software; this article will concentrate on the hardware.

A cassette interface is required to allow your computer to write data onto a cassette tape, and then read that data back from the tape to the computer, without errors. This is important; the most expensive cassette system in the world is of no use to you if it introduces bad bits into the data stream. Naturally, we want to build an inexpensive interface, but we don't want any errors either.

Let's consider the sources of bad bits. Assuming you can solder, and you use sound logic and audio practices in your interface, there are still 2 variables beyond your control: the speed regulation of the cassette recorder, and the drop-out rate of the tape.

THE TAPE:

Magnetic tape consists basically of a layer of iron oxide on a plastic film. If the oxide gets scratched, or the tape gets bent or stretched, or the oxide layer was poorly made, you will get drop-outs. At a data rate of 300 baud, each bit is recorded on a section of tape only 6/1000 of an inch long. It doesn't take much to scratch this much oxide off, or to have it missing in the first place. There is only one solution to the problem of dropped bits: buy the best tapes you can afford, no longer than type C60 to prevent stretching, and take good care of both your tapes and your recorder head. Certified digital quality cassettes are best, but these are very expensive (20 dollars) and may be hard to get. Top grade audio cassettes will do a good job, and there are ways to certify them yourself, if required.

THE RECORDER:

Now on to the problem of tape recorder speed. First it is necessary to introduce the idea of Framing, and the UART. In order to store a byte of data onto tape, it is necessary to convert the data from parallel to serial. The usual practice is then to add a Start bit in front of the first data bit, so you know where the data starts, and a Stop bit after the last of the 8 data bits, so you are sure of leaving the serial transmission medium in a known, or Marking state, which allows you to detect the beginning of the next Start bit. This set of 10 bits is known as a frame, and is illustrated in Figure 1a.

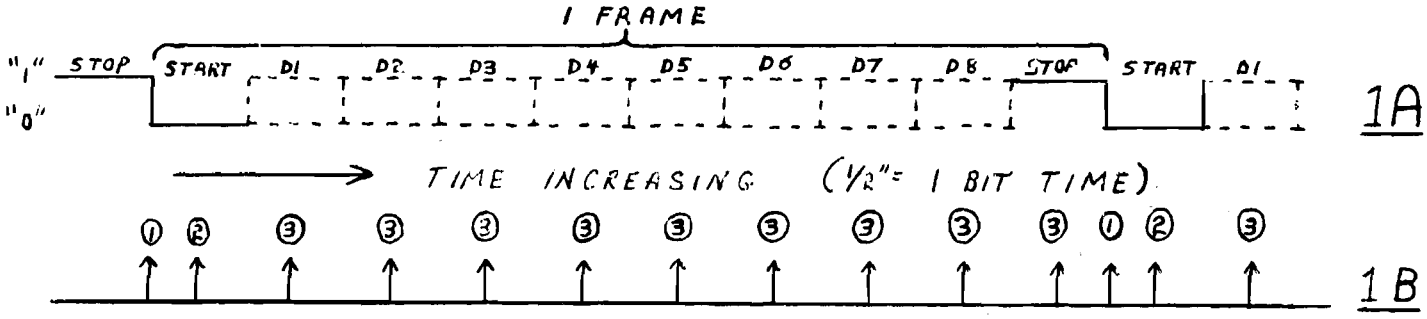


FIGURE 1

This is one form of what has become known as the Universal Asynchronous Receive/Transmit (UART) format. Many variations are possible; you can use STOP bits, an odd or even parity bit, and from 5 to 8 data bits. It is important that the receiver knows what format to expect, and at what Baud Rate to expect the data. The Baud rate is simply the # of bits (of any type) which will be received per second.

A UART function can be implemented in either software or hardware; the procedure is identical in either case. Data reception starts when the serial input signal changes from Marking (logic "1") to spacing (logic "0"). This is indicated in Figure 1B by the vertical arrow marked #(1), which co-incides with the beginning of the Start bit. If the serial input is still at a logic "0" ½ bit time later (arrow marked #(2)), then the Start bit is considered valid. Otherwise, the start bit is considered bad, and the logic returns to looking for the leading edge of a start bit.

If the start bit is valid, then the UART logic will look at ("sample") the serial input at time intervals exactly 1 bit time apart, as determined by the Baud rate, until all required data, parity and stop bits are obtained. (arrows marked #(3)). At this point the start and stop bits can be stripped off the received data, and the data byte presented in parallel for use of the computer.

I may seem to have wandered from the subject of recorder speed tolerance, but we have finally arrived at the point I wish to make. Remembering that the sample point for the Stop bit is a function of the leading edge of the Start bit, and the fixed Baud rate used by the UART receiver, it is easy to see that the actual rate of the serial data input to the UART cannot vary by more than ½ bit time per 10 bits, or ± 5%, if we wish to prevent data errors within a 10 bit frame. A more practical limit is ± 4%, to allow for some distortion in the serial data string.

Does your cassette recorder have sufficiently good speed regulation to meet this requirement? I know my El Cheapo doesn't. If you still think it does, then consider that this ± 4% speed regulation must be maintained between new batteries and old, high line voltage and low, dirty head and clean head, new cassette and old cassette, and between your recorder and your friends, if you are going to exchange tapes. Still think you can maintain ± 4% regulation? What about tape stretch, and replacing your dropped (busted) recorder with a new El Cheapo?

Sure, there are ways to overcome all of these objections, but they all cost money, and one of my prime objectives as a computer hobbyist is to spend as little money as possible. There is a (relatively) simple way to overcome the ± 4% speed regulation limitation. Basically, it involves recording Clock information along with data onto that cassette tape, so that when you play the tape back, you can tell the UART exactly how fast the serial data is arriving, instead of using a fixed Baud rate.

Don't go out and buy a stereo cassette recorder yet, though; thats expensive and unnecessary. There is a way to record both Clock and Data on the same tape track of that El Cheapo. What I'm talking about now is known as the Kansas City (KC) Standard, or BYTE Standard, for computer hobbyist data recording (So-called because the Editors of BYTE magazine first suggested it at a convention meeting in Kansas City).

... cont'd

I'm stuck on this technique, for a number of reasons: it offers speed tolerance of +45%, -30%, its fairly inexpensive to build, it is easy to understand and repair, and it has a minimum of critical set-up adjustments. It also allows software exchange via cassette tape, since out of the dozens of cassette interfaces in use on the hobbyist level today, at least half are KC Standard-compatible. (My own estimate). If this isn't enough, consider: the input/output of the interface is directly compatible with any standard UART chip, which will perform all frame error checking and serial-to-parallel or parallel-to-serial conversions, for about $5 extra. Admittedly, 300 Baud (30 bytes per second) is somewhat slow, but it sure beats typing a program in by hand. And for you speed freaks, the identical interface, used with a modified Software UART function, can be run successfully at speeds up to 1200 Baud! (You might need a medium quality recorder, though).

THE INTERFACE:

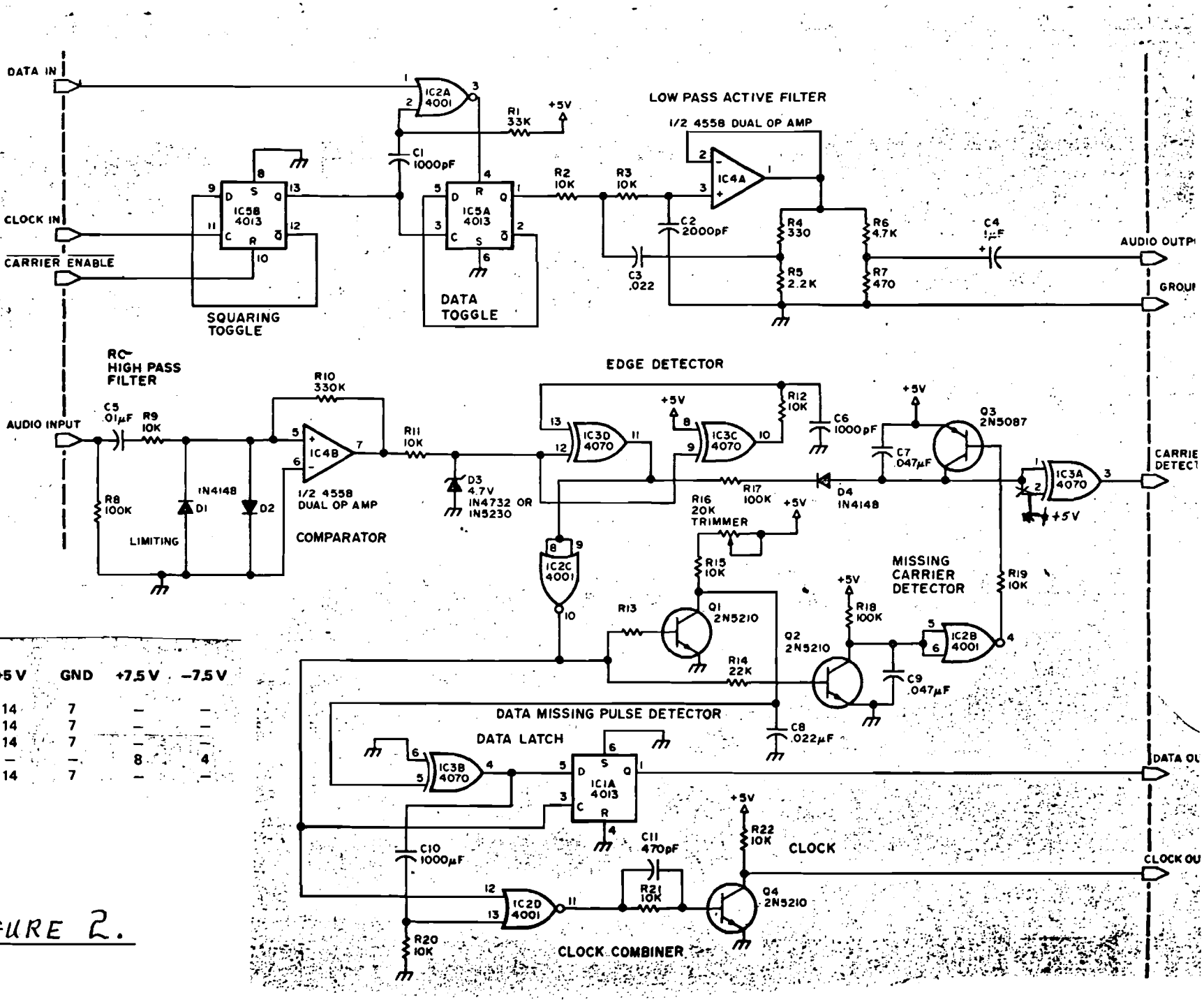
There are several versions of the KC Standard cassette interface available; one is by Don Lancaster, and can be found in his book "TV Typewriter Cookbook". Another was designed by Gary Kay of Southwest Technical Products Co. (SWTP), and can be found in the December, 1976 issue of BYTE magazine, pg. 98-108, or in the May 1977 issue of INTERFACE AGE magazine, pg. 29-31. (Beware of the schematic error on pins 1 and 2 of IC3A in these references, though). I have built and used the SWTP cassette interface circuit without problems, so I will describe it here. Figure 2 and most of the technical description have been extracted, with many thanks, from Gary Kay's article in December 1976 BYTE.

HOW IT WORKS:

The modulator works by feeding a 4800 Hz (16 times 300 Hz) clock into the toggle provided by IC5b. The division by 2 in this flip flop insures a 50% duty cycle required by the modulator. The carrier enable input provides a means of suppressing audio output from the modulator. IC5a divides the frequency by two once more if the data-in line is high and simply follows the clock frequency if the data-in line is low. This gives a 1200 Hz tone for a low state and a 2400 Hz tone for a high state. The resulting output is then fed into two pole active filters provided by the 4558 operational amplifier section IC4a, where it is converted to a closer approximation of a sinusoidal audio waveform which is more easily handled by audio recorders.

Incoming audio data is first fed into a high pass filter consisting of R8 and C5 and then onto comparator IC4b. The comparator's 0.5V hysteresis reduces the possibility of false triggering. The $\pm$ 7.5V zener regulated power buses feeding IC4 are necessary to stabilize this hysteresis value. Zener diode clamp D3 limits the comparator output to CMOS compatible voltage levels. Each time the comparator changes state the exclusive OR functions of IC3c and IC3d generate a 5 μs active low pulse at IC3 pin 11. This pulse is repetitive when data is being received and pulls capacitor C7 down to ground through diode D4. The level on capacitor C7 is in turn inverted and buffered through IC3a to form the carrier detect signal. This same active low pulse is inverted by IC2c where it feeds four separate circuits.

... cont'd



Number	Type	+5 V	GND	+7.5 V	−7.5 V
IC1	4013	14	7	–	–
IC2	4001	14	7	–	–
IC3	4070	14	7	–	–
IC4	4558	–	–	8	4
IC5	4013	14	7	–	–

FIGURE 2.

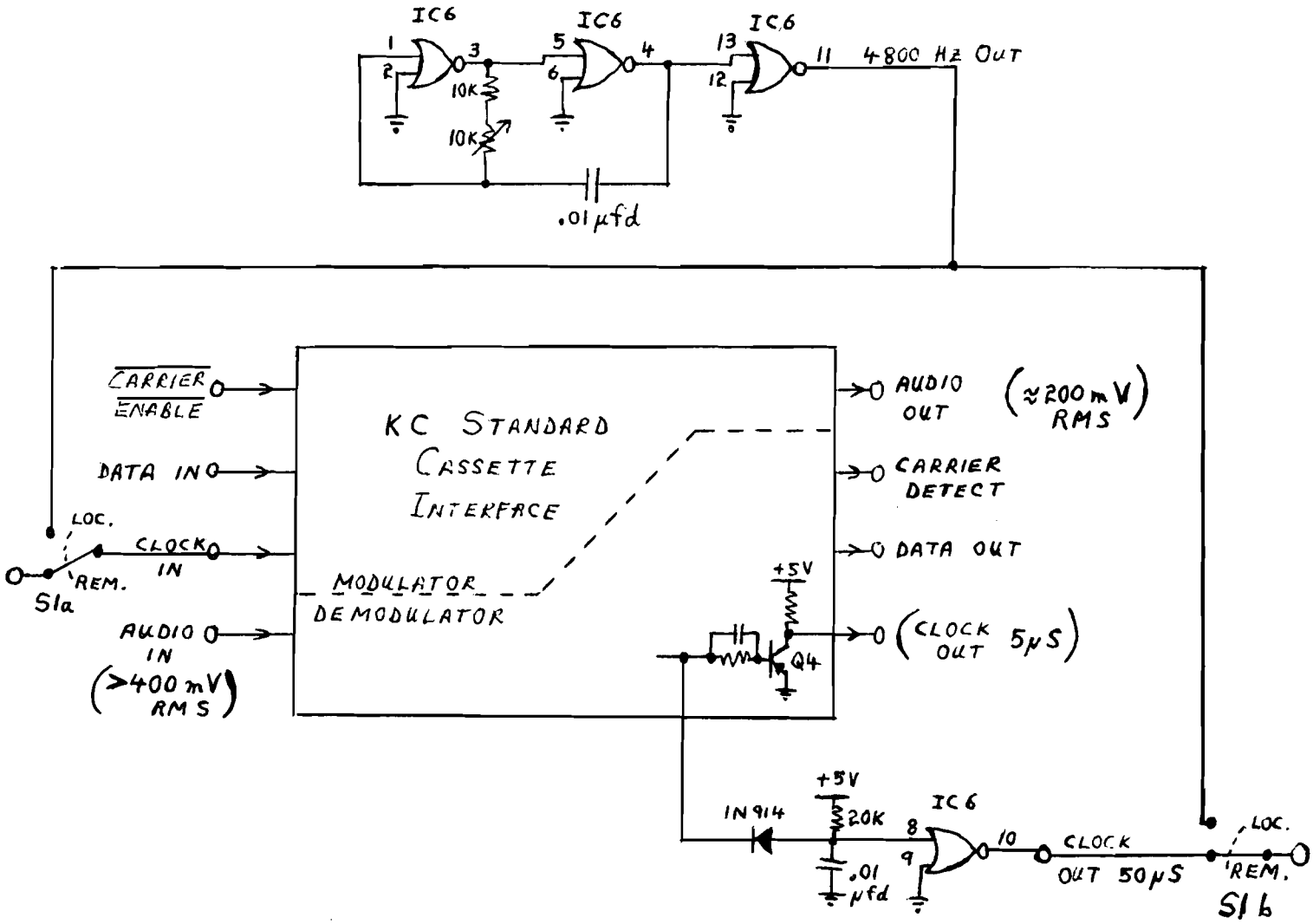

IC6
IC6
IC6
4800 Hz Out
10K
10K
.01 µfd
CARRIER ENABLE
DATA IN
CLOCK IN
AUDIO IN
(>400 mV RMS)
LOC.
REM.
S1a
KC STANDARD
CASSETTE
INTERFACE
MODULATOR
DEMODULATOR
AUDIO OUT
(≈200 mV RMS)
CARRIER DETECT
DATA OUT
(CLOCK OUT 5µS)
+5V
Q4
IN914
+5V
20K
.01 µfd
IC6
CLOCK OUT 50µS
LOC.
REM.
S1b

FIGURE 3

The first is a missing pulse detector composed of transistor Q2 and inverter IC2b.

This forces the buffered carrier detect signal low whenever several cycles of audio carrier are missed. The second circuit is also an adjustable missing pulse detector but this one times out whenever 1200 Hz data is being fed into the demodulator. The 20k ohm trimmer resistor R16 sets the period for this timer. The third circuit driven by this pulse is the clock input to flip flop 4013 IC1a which outputs the demodulated data. This data out line is high when 1200 Hz audio is being demodulated and low when 2400 Hz audio is being demodulated. The fourth circuit fed by the pulse is IC2d which with the addition of the output of IC3b synthesizes the 16X clock out data. Take note that although on the average this clock is accurate, it jitters by design.

After I constructed this interface as shown in Figure 2, I made a couple of changes to it. These are shown in Figure 3.

The first change is necessary if you want to run this interface using an EF line to detect the received Clock signal. This received Clock turns out to be a pulse only 5 μ S long; far too short to detect using a software loop. So I replaced the received clock transistor line driver (Q4) with a mono-stable circuit to stretch out the pulse width to about 50 μ S. This requires adding another 4001 IC to the circuit, which I have labeled IC6.

Since there were now 3 spare gates available in IC6, I decided to add a local oscillator to the circuit, to produce the 4800 Hz signal required when recording data onto the tape. This results in somewhat simpler software, since the clock no longer needs to be generated by a software timing loop. Notice that the clock can be switched in and out of the circuit; by using the received clock signal line during recording, both the record and receive software drivers can use the same time delay subroutine, but this requires that the local clock be switched in for recording and switched out for playback.

In conclusion, I feelthat the KC Standard cassette interface circuit I have described in this article meets all the necessary requirements for cheap, convenient program storage on tape, and provides the user with the capability for software exchange, and for recording at faster, non-standard Baud rates, if desired.

References

1- KC Standard: Feb. and March 1976 issues of BYTE magazine.

2- "The Designer's Eye View of the AC-30", by Gary Kay, SWTP Dec. 1976 BYTE magazine, pg. 98-108.

3- "Some Comments on the AC-30", by Carl Helmers. December 1976 BYTE magazine, pg. 100-101.

4- "UAR/T - Universal Asynchronous Receiver/Transmitter", product description by General Instrument Corp., describing the AY-5-1013 UART chip. 12 pages. March, 1974.

5- "A Software Controlled 1200 BPS Audio Tape Interface", by Carl Helmers. April 1977 BYTE magazine, pg. 40-49.

RCA 1802 - KC Standard Cassette Interface Test Routine

Al Dunlop

A specific test routine is an invaluable tool for checking out and debugging a system which will eventually be used in a computer system. Using a relatively uncomplicated program to test a peripheral device, in this case a 'KANSAS CITY'/Biphase Cassette Tape Interface, enables you to check out the interface hardware and concentrate on its problems. Testing a hardware device with a complicated user written program can be a trying experience especially when both hardware and software contain bugs.

The following test routine, written by the Editor and optimized by the Author, operates in two modes. The first mode reads data from the keyboard when the 'I' control is pushed and outputs the 8 bit byte using a software asynchronous transmitter. The second mode reads 8 bit bytes using a software asynchronous receiver and displays the bits on the DATA leds. The Hex code #OD, the ASCII Carriage Return character, is used to terminate each mode.

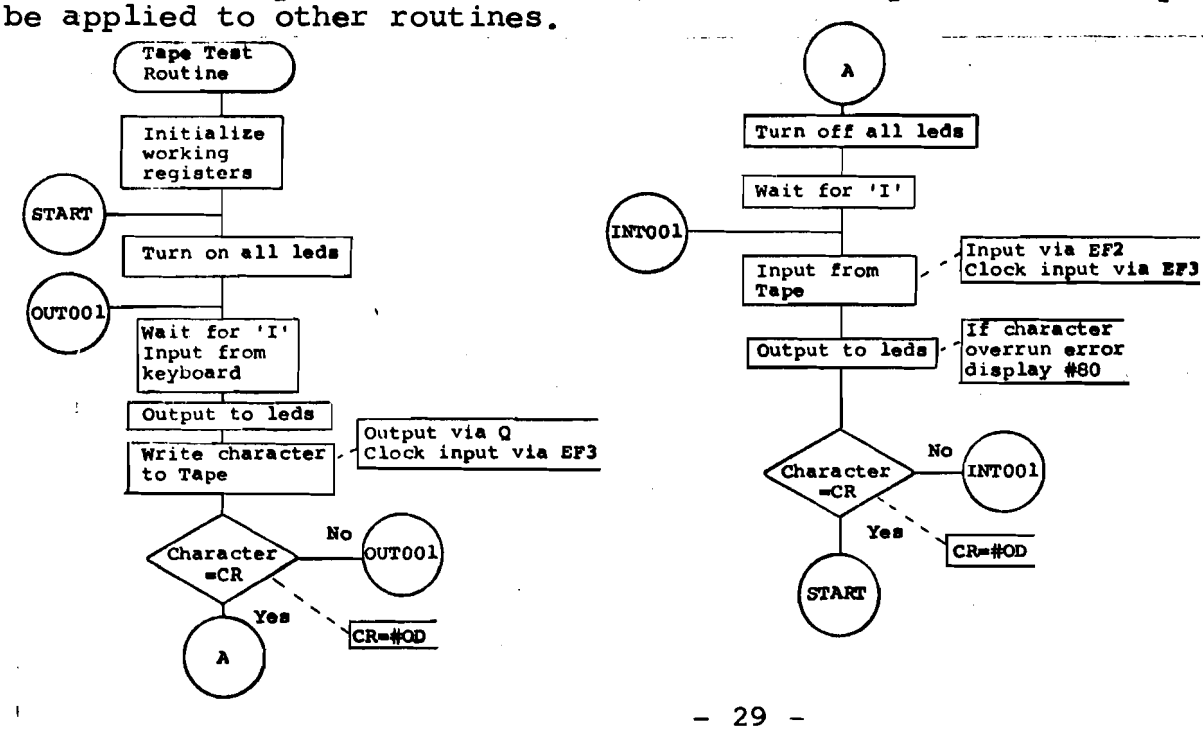
The software asynchronous receiver/transmitter uses an external hardware generated clock which is input via EF3. This clock is obtained from the KC interface (see "Cassettes and Computers" article in this issue).

This method of clocking the asynchronous receiver/transmitter was employed to make its operation independent of the CPU clock frequency. The accompaning flowchart indicates the usage of the test routine. Note that the bytes will be displayed during playback (tape read) at the same rate at which they were entered during recording (tape write).

The tape write mode may be operated in a continuous mode by replacing the instruction at label Y with a B4 X and the instruction at label Z with a BN4 OUT001. Now when two hex characters are entered via the keyboard the 8 bit byte will be output continuously as long as the "i" button is held down. When "i" is released, the program will return to OUT001 and await entry of new data via the keyboard. The Tape Read section of the test program is not used in the continuous write mode.

This continuous write mode provides a signal for use when trouble shooting with an oscilloscope.

The software asynchronous receiver/transmitter parts of this program can be applied to other routines.



```
                        .TITLE  --- TAPE TEST ROUTINE ---
                        ;  - - - - - - - - - - - - - - - - - -
                        ;  THIS ROUTINE IS USED TO TEST THE
                        ;   KC STANDARD CASSETTE INTERFACE
                        ;  - - - - - - - - - - - - - - - - - -
                        ;  NOTES: 1 - I/O LINE ASSIGNMENT:
                        ;             EF4 - "I" PB
                        ;             EF3 - CLOCK IN (XMIT & RECEIVE)
                        ;             EF2 - DATA IN
                        ;              Q  - DATA OUT
                        ;         2 - THE DELAY TIME IS INDEPENDENT OF
                        ;             MICRO-PROCESSOR CLOCK SPEED.
                        ;         3 - SUBROUTINE CALLS AND RETURN PERFORMED
                        ;             USING P-REGISTER SWITCHING.
                        ;
                        ;
0000  F8 00             START:  LDI     #00     ; INITIALIZE HI BYTE OF SOME REGS.
0002  B2                        PHI     R2      ;  -X-REGISTER (HI BYTE ALWAYS #00)
0003  B6                        PHI     R6      ;  -DELAY ROUTINE POINTER
0004  F8 61                     LDI     DEL200  ; (TAPE DELAY ROUTINE ADDRESS)
0006  A6                        PLO     R6      ;  -INIT.LO BYTE....DELAY ROUTINE
0007  E2                        SEX     R2      ;  R2 IS DATA POINTER
0008  F8 FF             START:  LDI     #FF     ;
000A  A2                        PLO     R2      ;                -X-REGISTER
000B  52                        STR     R2      ; PLACE #FF AT LOCATION #FF  *****
000C  64                        OUT4            ; TURN ALL LEDS ON
000D  22                        DEC     R2      ; RESTORE X REGISTER
000E  37 0E             OUT001: B4      @       ; WAIT FOR INPUT FROM KEYBOARD
0010  3F 10                     BN4     @       ;
0012  6C                        INP     4       ; GET KEYBOARD DATA BYTE
0013  64                        OUT4            ; DISPLAY ON LEDS
0014  22                        DEC     R2      ; RESTORE X REGISTER
0015  A9                        PLO     R9      ; SAVE FOR LATER USE
0016  B8                X:    PHI     R8      ; SAVE FOR OUTPUT TO TAPE
0017  F8 08                     LDI     8       ; SET UP BIT COUNTER.
0019  A8                        PLO     R8      ;
001A  7A                        REQ             ; SEND START BIT
001B  D6                        SEP     R6      ; DELAY 1 BIT TIME
001C  98                OUT005: GHI     R8      ;
001D  76                        SHRC            ; GET NEXT BIT TO OUTPUT
001E  B8                        PHI     R8      ;
001F  CF                        LSDF            ; IS DATA BIT A ONE?
0020  7A                        REQ             ; NO,  ITS A ZERO: RESET Q
0021  38                        SKP             ; BYPASS THE  SEQ  INSTRUCTION
0022  7B                        SEQ             ; YES, ITS A ONE:  SET Q
0023  D6                        SEP     R6      ; DELAY
0024  28                        DEC     R8      ; DECREMENT BIT COUNTER
0025  88                        GLO     R8      ;
0026  3A 1C                     BNZ     OUT005  ; DONE 8 BITS YET?  NO
0028  7B                        SEQ             ; YES: SEND STOP BIT.
0029  D6                        SEP     R6      ; DELAY 1 BIT TIME
002A  89                        GLO     R9      ;                (CONTINUOUS WRITE)
002B  FF 0D             Y:    SMI     #0D     ; IS THIS CR?          ( B4  X     )
002D  3A 0E             Z:    BNZ     OUT001  ; NO, GET ANOTHER BYTE ( BN4 OUT001)
```



```
                        ;
                        ;            TAPE READ
                        ;
                                GHI     R2      ; GET #00
                                STR     R2      ; SET UP TO TURN OFF
                                OUT     4       ;         ALL DATA LEDS.
                                DEC     R2      ; RESTORE X REGISTER
                                B4      @       ; WAIT FOR I
                                BN4     @       ;
                        INT001: LDI     8       ; INITIALIZE BIT COUNTER
                                PLO     R8      ;
                        INT005: B2      @       ; TAPE INPUT, IS THIS START BIT? NO.
                                LDI     8       ; YES SET UP DELAY 1/2 BIT TIME
                                INC     R6      ; R6 = R6 + 2   TO POINT TO
                                INC     R6      ;       DEL200 + 2   (IE. DEL205)
                                SEP     R6      ; R6 WILL POINT TO DEL200 UPON RETURN
                                B2      INT005  ; IS START BIT STILL VALID? NO.
                        INT010: SEP     R6      ; YES, DELAY 1 BIT TIME
                                ADI     0       ; RESET  DF FLAG
                                BN2     INT015  ; IS NEXT BIT A ZERO?  YES
                                SMI     0       ; NO.  SET  DF FLAG
                        INT015: GHI     R8      ;
                                SHRC            ; SHIFT DF INTO RECEIVED BYTE,
                                PHI     R8      ; AND SAVE RESULTS.
                                DEC     R8      ; DECREMENT BIT COUNTER
                                GLO     R8      ;
                                BNZ     INT010  ; DONE 8 YET? NO.
                                SEP     R6      ; YES. DELAY 1 BIT TIME FOR STOP BIT
                                GHI     R8      ; GET DATA BYTE FOR DISPLAY
                                B2      INT020  ; STOP BIT OK? YES.
                                LDI     #80     ; NO. REPLACE WITH INDICATOR
                        INT020: STR     R2      ; SEND TO LEDS
                                OUT     4       ;
                                DEC     R2      ; RESTORE X REGISTER
                                SMI     #0D     ; IS THIS CR?
                                BNZ     INT001  ; NO
                                BR      START   ; YES, GO TO OUTPUT ROUTINE
                        ;
                        ; *** DELAY SUBROUTINE ( 1 BIT TIME AT 300 BAUD IS 16 PULSES
                        ;                        USING INTERFACE 4800 HZ CLOCK) ***
0060  D0                DELRTN: SEP     R0      ; RETURN TO MAINLINE
                        ;   ACTUAL DELAY IS 16 CLOCK PULSES, BUT ONE PULSE MAY PASS
                        ;                UNDETECTED WHEN EXECUTING PROGRAM.
0061  F8 0F             DEL200: LDI     15      ; THIS DELAYS FOR 15 CLOCK PULSES
0063  36 63             DEL205: B3      @       ; IS EF3=0? NO.
0065  3E 65                     BN3     @       ; YES. IS EF3=1? NO.
0067  FF 01                     SMI     1       ; YES. SUBTRACT 1 FROM D-REGISTER
0069  3A 63                     BNZ     DEL205  ; DELAYED FOR 1 BIT TIME YET? NO.
006B  30 60                     BR      DELRTN  ; YES. RETURN TO CALLER.
                                .END
```


USER SYMBOL TABLE

```
DEL200 0061   DEL205 0063   DELRTN 0060   INT001 0037   INT005 003A   INT010 0043   INT015 004A   INT020 0057
OUT001 000E   OUT005 001C   START  0008   X      0016   Y      002B   Z      002D
```

PROGRAM SIZE = 006D
0 ERRORS DETECTED

A Standard Format for Cassette Data

Wayne Bowdish

This article will propose a standard method of formatting data on a cassette tape. The data format requires standardization for the same reason that the recording method requires standardization - to allow the efficient exchange of recorded programs and data. When both the recording method and the data format have been decided, a standard tape handler can be developed and distributed (possibly in ROM). It may even be included as part of a standard operating system.

Before I get into the standard, I'll define some functions which the hardware and software must perform. Then some methods of implementing these functions will be discussed. Finally, a standard will be discussed. This proposed standard should not be considered final. I know that many readers will have ideas of their own, possibly even working systems. Let's hear from you! Anyway, on with the article.

First some desirable hardware characteristics.

The hardware should allow:

- 2 or more cassette drives.
- software controlled motor start/stop.
- use a common, widely used, recording method.
- at least a 300 baud transfer rate.
- a hardware UART function is desirable.
- carrier or data detect should be available to the software.
- should be relatively inexpensive.

The software characteristics are as follows:

- allow 2 or more cassette drives.
- must be able to read and write cassettes.
- if a hardware UART function is not available, this function must be performed in software.
- must handle a wide variety of data.
  i.e. ASCII data files, memory dumps, etc.
- should easily interface to user programs.
- should supply utility functions (possibly as separate programs) such as:

  SEARCH for a given file on the tape.
  LIST the contents of a file.
  COPY a file from one tape drive to another.
  LOAD a file into memory.
  DUMP memory into a file.
  CREATE a file from an auxiliary peripheral such as a keyboard.

To implement these functions, a hierarchy of hardware and software is required. Figure 1 is a diagram of this hierarchy. The lowest level consists of the actual hardware required to record, and read back, data on the tape. The next level consists of a set of routines which perform such basic functions as reading or writing a byte of data or starting and stopping the drive motors. A software parallel to serial (and vica versa) conversion may also have to be done and appropriate start and stop bits may have to be added (and removed) from the data byte.

The next level consists of routines which interface to the users programs. These routines perform higher order functions such as reading or writing a block of data, creating and deleting files, loading and dumping memory, etc. More complex functions such as listing the contents of a file or copying a file from one tape to another may be implemented as separate programs.

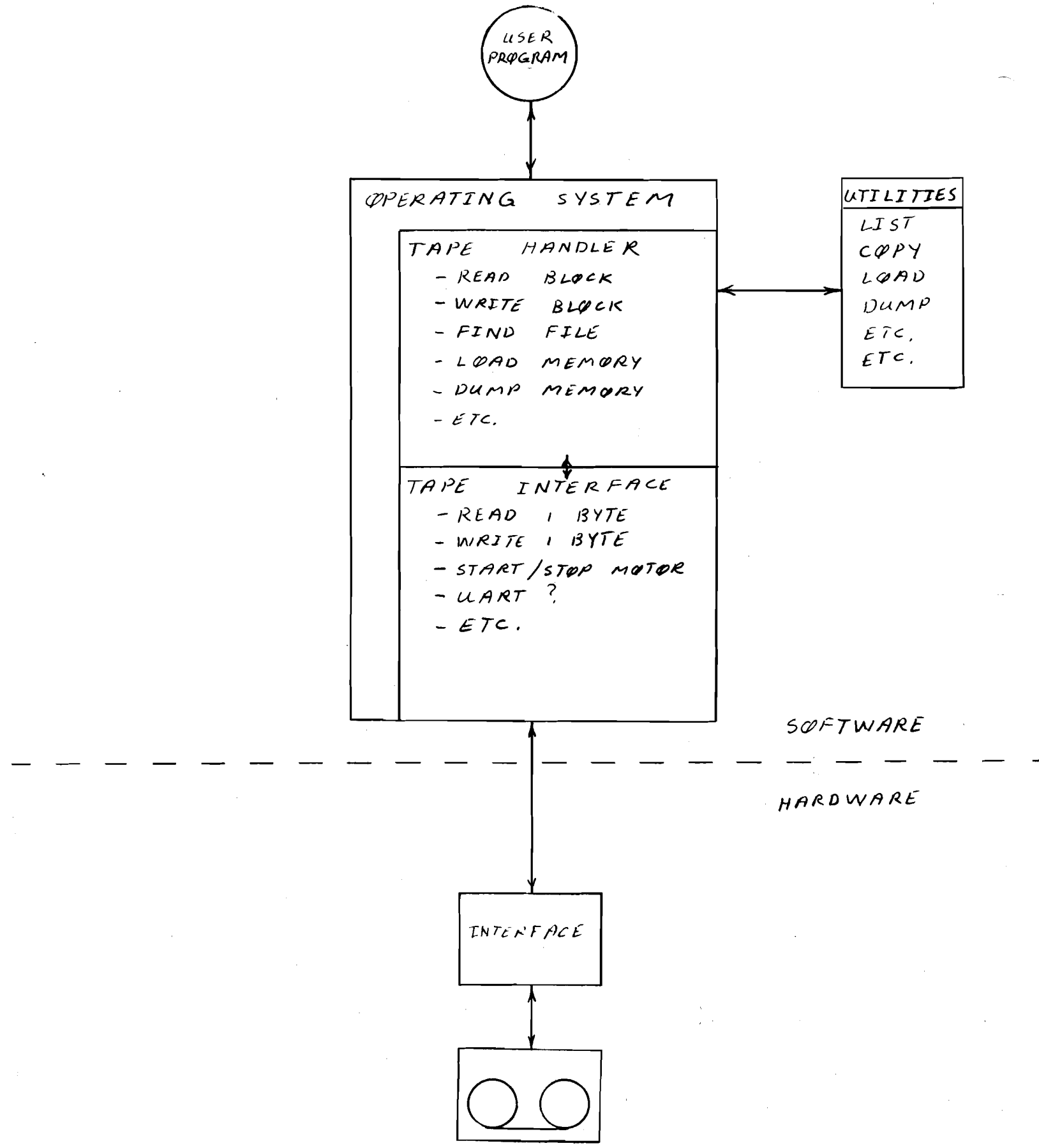

USER PROGRAM
OPERATING SYSTEM
TAPE HANDLER
- READ BLOCK
- WRITE BLOCK
- FIND FILE
- LOAD MEMORY
- DUMP MEMORY
- ETC.
TAPE INTERFACE
- READ 1 BYTE
- WRITE 1 BYTE
- START/STOP MOTOR
- UART ?
- ETC.
UTILITIES
LIST
COPY
LOAD
DUMP
ETC.
ETC.
SOFTWARE
HARDWARE
INTERFACE

FIG. 1

The highest level of the hierarchy consists of the users applications programs. Some examples might be a program to print mailing lables, an assembler, or a data collection program.

Only the lowest level of the software should be dependent on the actual hardware configuration. Higher levels of software should be independent of the hardware. This allows upgrading of the hardware without having to rewrite large volumes of software.

Several other articles in this newsletter will discuss the actual tape interface hardware and software. In this article, I will concentrate on a data format which will allow us to perform the higher level functions such as creating and manipulating files of data on the tape. Future newsletter articles will discuss the actual methods of implementing these functions.

Before going any further, I will discuss the concept of a file and some characteristics (or parts) of a tape file in general. A file is a collection of related data. For instance, a file may contain a program, a collection of assembler (or BASIC or whatever) statements, or it might be a mailing list consisting of a collection of names and addresses. A file might even contain a dump of your machines' memory which will be reloaded at a later date. Each item in the file, i.e. a line of code or an address, is called a record. A file may contain only a few bytes, and therefore several files could be stored on one tape, or a file may be very large and require several tapes.

The software must "know" quite a bit about the tape file in order to effectively handle it. Some characteristics are previously defined and therefore "built in". For example, the number of start and stop bits and the order in which individual bits are written. Other characteristics are not known in advance and are stored within the file. A file name and size (number of bytes) are examples.

Four components of the tape data format are:

- a data block size and format
- an inter record gap (IRG) size
- a file header format
- a sentinel file format

I'll discuss each of these in order. Blocking data means dividing the data into sections (blocks) and writing these blocks onto the tape. Each block is of the same size and they are separated by a short length of unused tape (called an inter record gap). The unused tape allows a space for stopping and restarting the tape drive. Why would you want to start and stop the tape? Eventually, applications will advanced past the simple load memory/dump memory stage to more advanced data processing applications. Most of these applications will generate or process data at a rate different from that of the cassette drive (In fact, the tape handler may not even allow simultaneous processing and reading or writing of data). Some examples of this might be:

- transferring data from a keyboard to the tape (how many people can type 30 characters per second).
- printing a mailing list on a teletype at 10 character per second.
- a remote data collection system which writes only a few pieces of data a minute.

Figure 2 shows this tape layout.

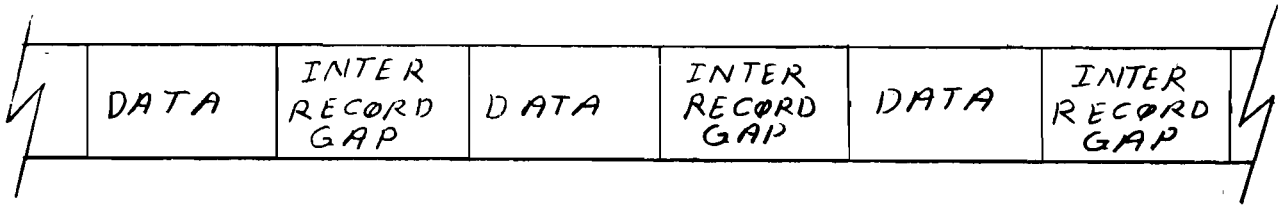


FIG 2.

Now about the header. As stated earlier, several files may be stored on one tape. How does your computer find these files. The answer is to use a header at the beginning of each file. A header is a block of data which contains information relating to the file. Some items might be a file name and a file type, i.e. is it a program or data? There are other important items but I'll list them in the proposed standard section.

Finally, there is the sentinel file. This is a file which indicates the end of a tape. The computer has no way of knowing whether a 30 minute or 60 minute tape is actually in the cassette drive so some means of indicating when the end of the tape is reached is required. The sole function of the sentinel file is to indicate when the end of the tape has been reached.

Figure 3 shows a complete (if somewhat shortened) tape.

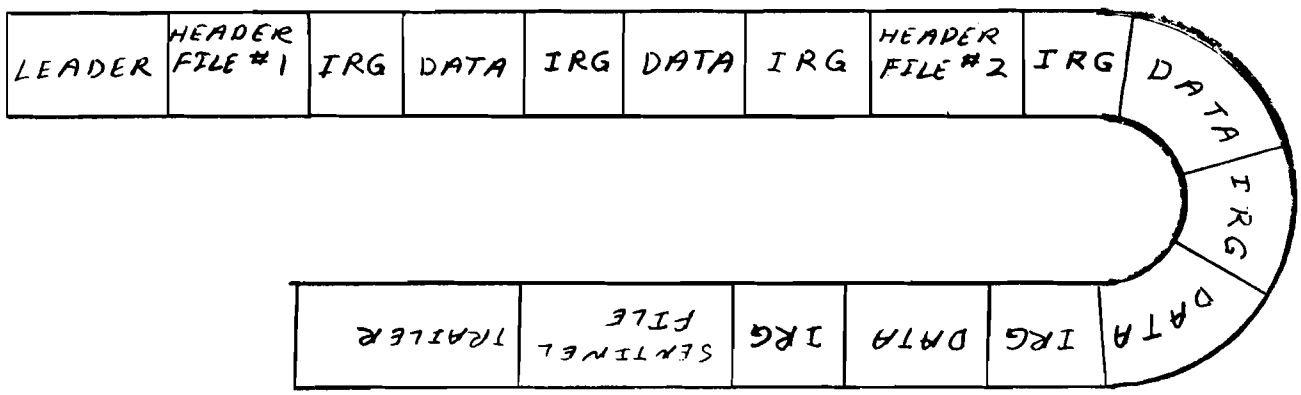


FIG 3

A PROPOSED TAPE DATA FORMAT

After this somewhat lengthy preamble, a data format will now be presented.

General Format

Data will be stored on the tape in files. A file will consist of a fixed length header followed by a number of data blocks.

Each data block and the header will be separated by a fixed length inter-record gap (IRG).

The end of the tape will be denoted by a file consisting of a file header with no following data blocks. This file is called a sentinel file.

A simplified file format will also be defined for cases where the application does not require the advanced file handling capability. This simplified format will be recognized and handled by the full format handlers.

The following five items will be defined:

- the inter record gap (IRG) size.
- the data block size and format.
- the header block size and format.
- the sentinel file format.
- the simplified file format.

IRG

As can be seen from figure 4, the amount of data which can be stored on a tape for a given data block size is dependent on the IRG length. The IRG must be long enough to allow the tape to come to a complete stop and then to restart and come up to full operating speed. Running the tape for about 2 seconds without writing data should probably generate an IRG of sufficient length. Some experimentation with a variety of tape units will be required to determine an acceptable IRG length.

DATA Block

As can be seen from figure 4, the amount of data which can be stored on a tape, for a given IRG length, is dependent on the data block size. As the block size is increased, the storage capacity increases.

When a block of data is read into memory, a block of memory must be reserved for the data block. Similarly, when a block is to be written it must first be set up in memory. This memory area is usually called a buffer. Many applications will require several buffers. This fact favours a smaller block size.

A tradeoff between memory requirements and tape capacity must be made. A good compromise appears to be a block size of 128 or 256 bytes.

The data block must also contain some additional information. The format is as follows:

- flag indicating the last block in the file — 1 byte
- count of the actual number of bytes used — 1 byte
- data bytes — 128 or 256 bytes
- an error detection code (CRC) — 1 byte

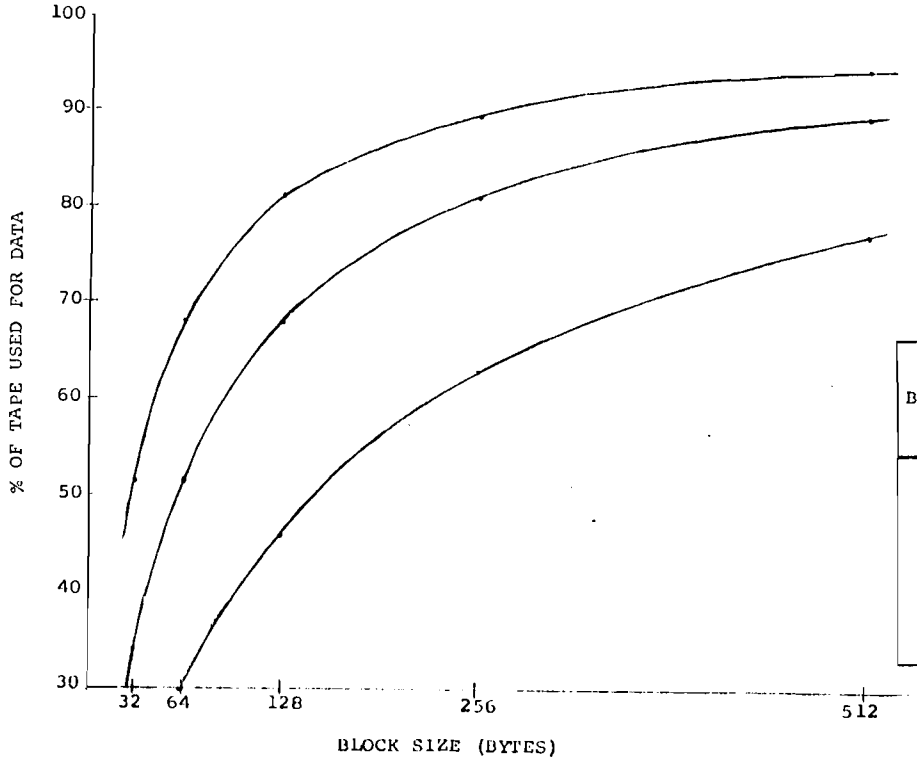
The flag byte would contain a 1 if this is the last block in the file and 0 otherwise.

The byte count is required since blocks may only be partially filled. This would normally be true for the last data block of a file.

An error detection code is useful in detecting errors during reading of the tape. One possibility is summing the data bytes with end around carry.

TAPE UTILIZATION

ASSUMPTIONS
- 60 minute cassette tape (C60)
- 300 baud recording rate
- MAXIMUM CAPACITY = 108,000 bytes



TAPE CAPACITY (K BYTES)

BLOCK SIZE	IRG SIZE 1 SEC %	1 SEC BYTES	2 SEC %	2 SEC BYTES	5 SEC %	5 SEC BYTES
32	52	55.7	35	37.6	18	19.0
64	68	73.5	52	55.7	30	32.1
128	81	87.5	68	73.5	46	49.7
256	90	96.7	81	87.5	63	68.1
512	94	102.	90	96.7	77	83.5

Fig. 1

HEADER Block

The header contains information about the file. This information is as follows:

- file type	1 byte
- start address	2 bytes
- transfer address	2 bytes
- file name	6 bytes
- spares	4 bytes
- error detection code	1 byte

Sentinel File

The sentinel file is a special type of header block which contains the following:

- file type (=HEX FF)	1 byte
- 4 bytes for compatability (= 00)	4 bytes
- spares	10 bytes
- error detection code	1 byte

Simplified Format

In some cases, the microprocessor will be used in dedicated systems where a minimal hardware and software configuration is desirable. In these cases the full format support will not be required. A simplified format could be used to load the memory with the specific application program. This format will be recognized by the full format handler.

The file consists of a 16 byte header followed immediately by the data bytes. There may be from 1 to 65,536 data bytes. Immediately following the data would be an error detection code byte. The header would consist of the following:

- file type (simple format indicator)	1 byte
- start address	2 bytes
- byte count	2 bytes
- spares	10 bytes
- error detection code	1 byte

Conclusion

This is the proposed format. Several areas still require further definition. The file types should be defined. One possible standard is to use ASCII characters such as

A	- an assembly language file
B	- a Basic language file
D	- a data file
S	- simplified format file
HEX FF	- sentinel file

Another area to consider is the error detection code. This code should be easily to compute but should detect all errors. Simply summing the data bytes with end around carry may be an acceptable method.

Finally, I would like to repeat myself. (from the first page). This spec. is not final. Let's get some feedback! What other formats are there? Have you any ideas for improvements? Let's work together and come up with a flexible, easy to use system (that doesn't gobble up memory).

The WOM Tester

(Ed. Note: Reprinted from ELECTRONIC DESIGN #21, Oct. 11, 1977. TC)

We regret the delay in our publication of the April 1 announcement of the new Tektronix WOM tester. The system is intended for use with the Signuteks 9046 X N random-access write-only memory or equivalents. The system is unique in its testing neu devices (which enhance or deplete regardless of gate polarity).

The tester provides fast, clean pulses by use of Pushme-Pullyou Drivers. Quadra-state comparators allow flexibility in checking WOM outputs with only 12 clock phases.

The system uses TEKTEST IXX, Version 999.99, which is neither upward or downward compatible with other test languages due to its unique language structure. Users can be freed of problems with hexadecimal, octal and other cumbersome languages, and revel in the simplicity of ones and zeros. No compilers, assemblers or translators are needed, so there is ample space for test programs.

The address ports of the WOM are multiplexed with a negative frequency to achieve optimum resolution. The drain on the WOM is tested by monitoring a 5-gallon bucket.

Handlers are easily interfaced to this tester with only 500 signal lines. The system is capable of binning into two categories: one for NO-GO DIPs, the other for FAILED I^2R. If the I^2R mode occurs, it's time to call the power-distribution agency to find out how many substations went "WOM" as they exceeded maximum dissipation.

The instruction set includes:

BH	Branch and Hang
IIB	Ignore Inquiry and Branch
TDB	Transfer and Drop Bits
DO	Divide and Overflow
SRZ	Subtract and Reset to Zero
PI	Punch Invalid
SSJ	Select Stacker and Jump
FSRA	Forms Skip and Run Away
RASC	Read and Shred Card
SRSD	Seek Record and Scar Disc
BST	Backspace and Stretch Tape
RIRG	Read Inter-Record Gap
UER	Update and Erase Record
SPSW	Scramble Program Status Word
EIOC	Execute Invalid Op Code
EROS	Erase Read-Only Storage
PBC	Print and Break Chain
CM	Circulate Memory
MLR	Move and Lose Record
CRN	Convert to Roman Numberals
DMPK	Destroy Memory Protect Key
D€	Divide and Conquer
EIP	Execute Programmer Immediate
LCC	Load and Clear Core
HCF	Halt and Catch Fire
IDC	Initiate Destruct Command

Depending on configuration, this machine will sell competitively from $1.98 to something less than the national debt.

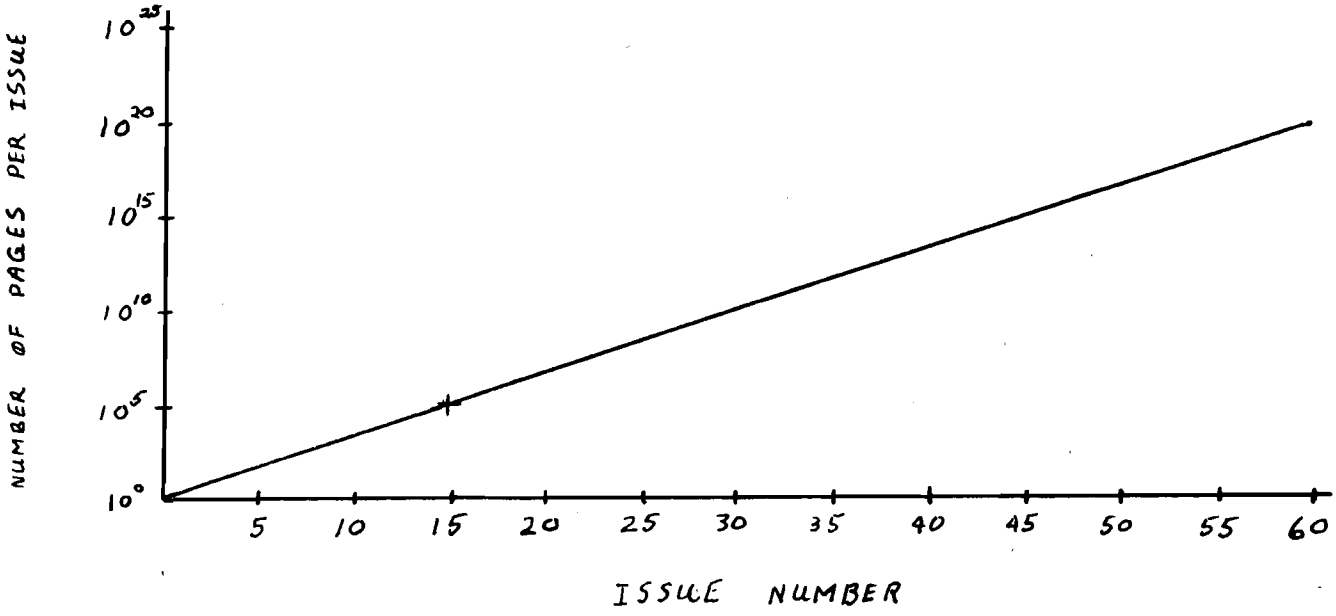
A Note from a Worried Member

Tom;

To date, each issue of the newsletter has approximately doubled in size. As can be seen from the table below the paper consumption for the newslette will soon exceed the total world paper production. This should happen about issue number 30. This is only about 4 years in the future! If this should happen there would be dire social and economic upheavals, not the least of which would be a lack of toilet paper in public washrooms.

I therefore propose that a maximum upper limit be set on the size of the newspaper. A limit of about 10^5 pages would probably be an acceptable limit. This limit would be reached at about issue number 14. A newsletter of this size would probably supply a good stimulus to the paper industry. The effects on the post office system can only be guessed.

In closing, I suggest that we act immediately to avoid this potential world-wide problem.



Errata

Wayne Bowdish

Inevitably any publication will make an error. True to form, the last issue of IPSO FACTO had some errors in the morse code program listing on page 19. There are 3 corrections to make as follows:

BYTE ADDRESS	OLD CODE	NEW CODE
002B	SEQ 7A	SEQ 7B
0031	REQ 7B	REQ 7A
0048	RESET: LDI #0B F8 0B	RESET: LDI KRBIN F8 11

Our thanks to Joe Kolodziej for spotting these errors.

This also brings out an error in the USERS MANUAL FOR CDP-1802 COSMAC MICROPROCESSOR (MPM 201A). The opcodes for the REQ and SEQ instructions are reversed in the INSTRUCTION SUMMARY table on page 102. The values should be

REQ 7A
SEQ 7B

These are listed correctly in the INSTRUCTION REPERTOIRE section on pages 39 and 40.

CONSTITUTION

of

THE ASSOCIATION OF COMPUTER EXPERIMENTERS

ARTICLE 1. Name
The name of the club shall be "The Association of Computer Experimenters" hereinafter called the Club.

ARTICLE 2. Location
The headquarters of the Club shall be located within the City of Hamilton, in the Province of Ontario.

ARTICLE 3. Purpose and Object
To promote and encourage interest and activities associated with personal computing applications and it is hereby declared that any funds or other accretions to the Club shall be used in the promotion of the objects of the Club; this shall not be construed as precluding the Club from acquiring land, buildings, vehicles, equipment, supplies and chattels for promotion of its objects.

ARTICLE 4. Membership

4.1 Membership of the club shall be of two (2) classes.
1) Honorary Lifetime Member. The executive committee may elect, from time to time, as honorary lifetime members, persons or organizations who are well known for their interest in personal computing applications and/or for their service to the Club.
2) Full Member. Open to persons interested in personal computing applications.

4.2 The Club by majority vote of those present at any regular, special or annual meeting, may levy upon the membership such dues or assessments as shall be deemed necessary for the business of the Club within the terms of this Constitution and by-laws.
Non payment of such dues or assessments shall be cause for expulsion from the Club within the discretion of the executive committee.

ARTICLE 5. Executive Committee

5.1 There shall be an executive committee consisting of a past president, and an elected president, secretary/treasurer, editor and executive-at-large as deemed necessary by the president.

5.2 All members of the executive shall be full members of the Club.

5.3 A majority of the executive committee shall constitute a committee quorum.

5.4 Meetings of the executive committee shall be at the call of the president.

5.5 The excutive may, of its own power, decide and act upon all matters of the Club except that;
1) It may not commit the Club or members of the Club to any binding and continuing action or policy without ratification by a regular, special or annual meeting.
2) It may not commit the Club to any financial obligation beyond the existing funds of the Club.
3) It may not cause a reversal or negation of a legally established decision of a regular, special or annual meeting.

ARTICLE 6. Meetings

6.1 Regular meetings shall be held at such time and place as shall be properly determined by the executive committee.

6.2 Special meetings may be called by the president upon the written request of any five (5) members of the Club, or by an approved motion of the executive committee.
Notice of all such meetings shall be sent to all members informing them of the meeting and the agenda of business. Such notices to be sent by mail at least seven (7) days before the time set for the meeting. Only such business as is designated by the notice shall be transacted at such meetings.

6.3 The annual meeting shall include the election of the executive committee and consideration of reports of the president, treasurer and chairperson of all committees.
The treasurer shall present a statement of accounts.

6.4 The president or in his absence, the secretary/treasurer or in his absence, a member appointed by the executive committee shall preside at all executive, regular, special or annual meetings of the Club, and shall vote only in the event of a tie.

6.5 A quorum for a special or annual meeting shall be 25% of the membership or 20 members whichever is less. A quorum is required for passing financial and constitutional motions.

6.6 All full members in good standing shall be elegible to vote.

6.7 Unless otherwise provided, questions arising at any meeting shall be decided by a majority vote of members present.

ARTICLE 7. Elections

7.1 Elections for all executive committee positions shall be conducted at the annual meeting to be held in the month of May of every year.

7.2 For a person to be accepted as a nominee for an executive committee position he shall be in good standing with the Club.

7.3 The president shall name two (2) scrutineers and an election shall be held by secret ballot for all the executive positions.

7.4 The new executive shall take office at the conclusion of the annual general meeting.

ARTICLE 8. Vacancies & Executive-at-large
Vacancies and executive-at-large on the executive committee shall be filled by appointment by the executive but shall be subject to ratification at the next meeting of the Club.

ARTICLE 9. Committees
The executive committee shall have the power to appoint such ad hoc and standing committees as it sees fit, subject to ratification at the next meeting of the Club.

ARTICLE 10. Amendments to the Constitution
This constitution may be amended by a two-thirds (2/3) majority of the full members present at a special or annual meeting of the Club providing that notification has been mailed to all members of the intent to amend the Constitution at a specified time. Such notice shall outline the proposed changes to the Constitution.

ARTICLE 11. Duties of the President
The president shall chair all meetings, be responsible for appointing executive-at-large, and act as official representative of the Club.

ARTICLE 12. Duties of the Secretary/treasurer
The secretary/treasurer shall keep a record of the proceedings of all meetings, carry on correspondence, arrange for notification of members of every special and annual meeting of the Club, except when excused by action of the by-laws. The secretary/treasurer shall control and supervise the collection and receipt of all monies payable to the Club and shall keep an accurate account of all monies received and expended. All cheques of the Club shall be signed by the secretary/treasurer. He shall at the expiration of his term of office turn over everything in his possession belonging to the Club to his successor.

ARTICLE 13. Audit
13.1 The books of the Club shall be audited annually at the end of the fiscal year by two (2) auditors appointed by the executive committee.
13.2 The auditors shall not be members of the executive committee.
13.3 The fiscal year of the Club shall be from the first day of June to the following last day of May.

ARTICLE 14. By-laws
A special, regular or annual meeting of the Club may by majority vote, institute and amend by-laws for its own government provided that such by-laws are consistent with the provisions of the constitution.

ARTICLE 15. Extra Mural Relations
The Club may collaborate with other organizations for the furtherance of common objectives.

RTICLE 16. Rules of Order
In all matters not covered by this Constitution, Robert's Rules of Order shall apply.