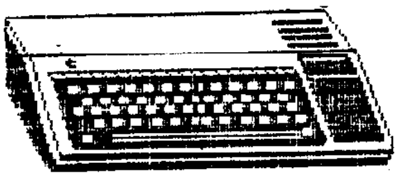
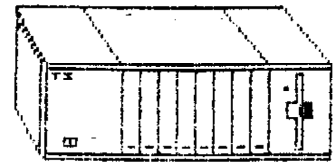
NEW JUG NEWS



NEW JERSEY USERS GROUP



Vol.5 No.2 *Monthly Publication of the New Jersey Users Group* FEBRUARY 1986

****

MEETING

****

FEBRUARY 10 MONDAY 7:00

7:00 - 8:00 BASIC SIG WILL MEET

8:00 - GENERAL MEETING--**GARY** -demonstration of TI-ARTIST V2.0

!!!

MEMBERS MUST PAY DUES THIS MONTH TO RECEIVE MARCH ISSUE

Star (*) on mailing label means you have paid for 1986

****

The New Jersey Users Group meets on the second Monday of each month in the Metuchen Library.

****

OFFICERS

President............Steve Citron..686-5619
Vice-Presidents......John Bonito...653-2637
Bob Costello..663-4512
Mel Gary......828-5407
Bob Guellnitz.382-5963
Secretary............Carol Sudol...494-3781
Treasurer............Marv Shuldman.821-8158
Newsletter Editor....Mel Gary......828-5407
Software Library.....Dave Green....463-9133
Software Review......Dan Ferst.....536-4255
Advanced Prog. Sig...Jay Holovacs..356-3150
Basic SIG............Bob Haefeli...572-2828

****

SUBSCRIPTION FREE WITH PAID MEMBERSHIP, TO USERS GROUPS AND SELECTED VENDORS

****

FEBRUARY 1986

SUNDAY	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY
						1
2	3	4	5	6	7	8
9	10 GENERAL MEETING	11	12	13	14	15
16	17	18 STEERING COMMITTEE	19	20	21 NEWSLETTER DEADLINE	22
23	24	25	26	27 ADVANCED PROG.	28	

President:

Steve Citron
981 Townley Ave.
Union, NJ 07083

Send Dues To:

Marv Shuldman
28 Tyndall Rd.
Kendall Pk., NJ 08824

Write For Application:

Bill Dubrow
21 Seaward Ave.
Metuchen, NJ 08840

MYARC UPDATE

By J. Peter Hoddie

(d/l from CompuServe by mel gary)

A few weeks after completing my 22 page report on the TI Faire in Chicago, I recieved a copy back from Lou Phillips himself. He had gone through my article with a pen and made comments, clarifications, and corrections. Instead of editing the original file (which I believe is 100% accurate with respect to what was said in Chicago) I am appending this file which contains Phillips' comments. I also spoke with Lou on the phone and got some further information. Some of this is very technical and some of it is very trivial. My main purpose through all of this has been to bring to you as much information on the new Myarc computer as possible. I plan to stay in regular contact with Lou Phillips so if you have any other questions, let me know and I'll pass them on.

The 48K of internal ROM I mentioned includes both library routines and also the BASIC interpreter. The 8K of mouse support is not mouse support but Operating System support. The mouse support is built into the video chip. Thus the reason for using the MicroSoft mouse is that the support is built right into the hardware.

The 9995 microproccessor, although faster then the 9990 will still be working under conditions similar to those that governed the 9990. This means that like the 9900 it has 256 (or 128 words) of internal "0 wait state" RAM which was used for scratch pad in the 99/4A. The 9995 has the same 256 bytes of "0 wait state" RAM and it will be used similarly. It will be used primarily as work space registers when the machine is running in the Myarc mode. The remaining memory (up to 2 megabytes) will be "1 wait state memory" (i.e. 666ns or 2 clock cycles).

Phillips says that at this point there are no specific plans to develop a card to allow internal placement of the Speech Synthesizer (there is no connection for it on the Myarc machine) as is done in CorComp's Triple Tech card. He says that Myarc is "only considering" such a product.

Phillips felt it neccesary to clarify why a new connector to the PE Box was needed. He says 1) "The II flex cable has only 16 bits of address!! To get at 512K (which the TI PEB was designed to support) we need 3 more address line!" Furthermore, Phillips plans to be able to address 2 megabytes of memory which means that 2 more lines are needed. I have seen the pin outs for the side port on the 99/4A and it does have the address lines to support 512K as Phillips has pointed out. However, he has made it clear that these lines were not fully implemented in the PE Box hardware and so he can't use them. Thus he needs to have another cable. On this point, I must admit, I am not 100% clear. 2) "In order to perform more 'exotic' control and machine code debugging, more of the signals are now going to the PEB (i.e. IAQ, HOLD, and other video capabilites)." 3) "Everyone says they want a round, more 'flexible' cable, therefore here it is!"

Phillips also pointed out that he is not using the 9938 video chip developed by TI, but a chip very similar to it now coming out of Japan.

Phillips was also quick to point out the the quad density disk controller was "only a re-layout of their present controller" and not a new product. I guess this means that we can hope to see it in the near future and without the problems that can often accompany new products.

I submitted two and half pages of questions to Phillips and the results are reproduced below.

Q: Will a new assembler be released to support the new features of the 9995?
A: We are looking for someone to write one.

Q: Will more utilites such as VMBW, DSRLNK etc. be available?
A: They will be implemented as XOP's (eXtended OPeration). Over 100 are planned and most are currently implemented in some form in the new Extended BASIC II. This will allow for integration from XB II to the new machine with relative ease.

Q: Will the GPL interpreter be any faster?
A: It should be between 2 and 3 times faster.

Q: Will Craig Miller's new GPL assembler and dis-assembler still work?
A: We have been in contact all along. (this should mean that the GRAM Kracker will also still be useable!)

Q: Will a technical assistance line be available?
A: If economically feasible. (this I don't like. Although Phillips has promised full documentation of the machine, things could get very sticky unless there is an established way of getting answers to questions that users may have.)

Q: Are there any other problems with compatablilty besides the changes in the KSCAN routines?
A: I hope not. (Phillips has said that they have encountered none so far but even just the change in the KSCAN routine renders such programs as Fast-Term and Danny Michael's Screen Dump useless without modification. Keep your fingers crossed.)

Q: Will current programs with an assembly/X-BASIC mix work with the new X-BASIC, especially when all the changes in variable space allocation are considered?
A: Too early to tell but XB-II will be a gooood test. (In a later conversation Phillips said that the initial release of XBII would NOT support passing variables between XB-II and assembly but that a future release in a month or so would).

Q: Will all the old "scrach pad" RAM use remain the same - at least in 99/4A mode?
A: There will be RAM there but the highspeed RAM is at >F000 in the 9995.

Q: How will one switch from 99/4A mode to Myarc mode?
A: From the operating system, you are in Myarc mode. In BASIC there are 2 calls to ASSEM (one old, one new).

Q: Will CorComp cards, particularly their disk controller, work with the new machine?
A: Yes, but we sure DON'T like what they do on power-up. Maybe Craig Miller (can fix this). (Although Phillips is justified in his view that CorComp is wrong in what they do at power-up, I only hope that he doesn't use this as an excuse to shut out thousands of CorComp users. Hopefully a solution will be found.)

Q: Will XB-II support ALL grapics modes?
A: It will support most but not all such as Pattern (multi-color) mode.

Q: Will the commands to put text on the screen (i.e. DISPLAY AT, PRINT, etc.) work in bit map mode or will there be separate commands for this?
A: In XB-II HCHAR, VCHAR, and two new commands CALL WRITE(X,Y,A$) and CALL VWRITE(X,Y,A$). On the new computer DISPLAY AT will work in bit map.

Q: Have user defined subprograms been retained?
A: Not in release 2.0 of XB-II. They will be in release 2.1. The hooks are already there.

Q: Will user defined subprograms execute faster then in TI Extended BASIC?
A: Yes. Everything is 2.3 times faster.

Q: In the BASIC editor will there be a way to search (and replace) a certain piece of text?
A: No.

Q: Will there be some sort of EXEC command facility to allow a "batch" file to be created?
A: Yes. In version 2.1 you can use DIS/VAR 80 files as input.

Q: Will it be possible to use the function keys in BASIC to enter key words?
A: The function keys are only supported with the TERMCHAR function in BASIC.

Q: Will the floating point routines maintain the high level of accuracy of TI's routines?
A: We are using the same routines.

Q: Will functions be available to convert from integer to floating point and back?
A: This is done automatically like IBM Fortran G.

Q: Will there be an easy way to catalog disks from BASIC? Initialize, rename, etc.?
A: The catalog feature is in our controller already. The new computer will support access to the disk operating system from BASIC.

Q: C has been mentioned as the next language: Have you contacted Clint Pulley who has ready written a small C for the 99/4A?
A: We are in contact with a developer for C. (Phillips has further clarified this. He has said that they have helped out a person who is developing a small C but at this point there is no formal arrangement between the two. Furthermore, he has not dealt with Clint Pulley.)

Q: Will a print spooler and RAM disk be built into the new computer?
A: It is alread in our 128/512K cards. It will be built into the new computer.

Q: Is the TI 32K card useless or can it still be used as RAM?
A: Useless.

Q: Will there be a BASIC compiler?
A: It is a high priority. As of yet, it is not started however. (Phillips has further stated that due to the vast array of XOPs available that the task of writing a compiler for any language will be considerably simplified.)

Q: Will it be possible to time and date stamp files?
A: This is already done in the MDS/100 (Winchester hard disk system) and will be done in future products.

Q: Will a reset switch be in place instead of wearing out the on/off switch?
A: A buffered keyboard will be used with special keys.

Q: Will it be possible to auto boot off of disk on power-up instead of dealing with title screens?
A: Yes, on the new computer.

Q: Will there still be joystick and cassette support?
A: Yes.

Q: How about a switch like on the Apple //c to allow use of a DVORAK keyboard?

A: I doubt it.

Q: Will more levels of interrupts be available then the 2 on the 99/4A?
A: Yes. The 99/4A only used 0 and 1. The 9995 uses 0,1, and 4.

Q: How about an internal 300/1200 baud modem?
A: Possible card in late '86.

Q: Anything new in the sound chip to allow for more complicated sound effects and music?
A: No. We must use the same chip for compatability.

Q: Will "TI-FORTH" work?
A: Probably. (I don't like this answer at all. TI-Forth is not so exotic that any special problems should occur.)

Q: Will there be a fan and will there be problems with extended use?
A: No fan. Should have no problem with extended use.

Q: Who will service the new computer?
A: We are NOW setting up capabilities in the Southwest (3 places), California, Chicago, Atlanta, and looking for more.

Well that is the end of it. I hope this has answered some more of your questions about the new Myarc computer. No definate release date has yet been set but Phillips seems to be aiming for the March/April time period. All we can do is wait.

C-TUTORIAL-1

By Warren Agee

(d/l from CompuServe by mel gary)

This is the first in (hopefully) many tutorials on the C language. Bear in mind that I am *not* an experienced C programmer. In fact, c99, by Clint Pulley, is not only the first C compiler for the 4A, but is also my first exposure to the C language. This tutorial, as well as any future tutorials, will be short and sweet. No lengthy explanations. You can spend $20+ for a book on C for that. I will focus on specific problem areas that I have experience in learning C, in hopes to make the language less mysterious and more understandable to everyone interested.

As I stated above, I will not be going into detailed explanations of all facets of C. As such, I am assuming that the reader knows how to type in a C program and run the compiler to generate runable object code.

LOCAL AND GLOBAL VARIABLES

In c99, there are two basic "storage classes" of variables: local and global (more commonly called external and automatic). In a BASIC program, all variables are global: once the contents of a variable is changed, it is changed throughout. In Extended BASIC, you have the ability to write user-defined subprograms, using the SUB statement. With this feature, you can incorporate local variables; these have no effect on the outside, or calling program, even if the variable names occur elsewhere in the program. The same effect can be easily achieved in C. Any variable you use in a C program must be declared before it can be used, and is done the following way:

```
int a;   /* This defines an integer variable "a" */
char string;   /* this defines a character variable "string" */
```

Now where these declarations appear in the program determines whether they are automatic (local) or external (global). Basically, there are three basic parts to a C program: the header, the main() function, and the collection of user-created functions. The header contains, among other things, any #include statements which direct the compiler to load in other files. The header is the first thing that appears in a C program, before the main() function. If a variable declaration is included in the header, then it is an external variable. This means that you may use that variable anywere in your program, anywhere in your functions. If you change the value in one function, it changes throughout program. To create an automatic variable (local), declare it *inside* the function where it will be used. In effect, that variable will live only in its "home" and no where else in the program.

Here is a short C program that is quite useless, but illustrates the concept of automatic and external variables, as well as the three parts of a C program.

```
/* A C PROGRAM which illustrates storage classes */

/* The following is the HEADER */

#include dsk1.stdio  /* loads in a separate file i/o library */
int index;  /* An external integer variable */
char string;  /* An external character variable */

/* The following is the main() function */

main()  /* start of main() function..every program must */
{       /* contain this function! This is executed*/
        /* when the program is run */
```

```
    int x; /* An automatic (local) integer variable */
    putchar(12)   /* clears screen */
    puts("Please enter your name:");
    gets(string);   /* Even though "string" was not
declared in the */
                /*  main() function, it is EXTERNAL, so it
can */
                /* used anywhere.  */

    display();   /* Calls display(), a  user-written
function.  */
    puts("\n\nThat's all, folks!");
    }

    /* The following is a user-defined function */

    display()
    {
    int  boogie;   /* boogie is local to display() only.
*/
    puts("\n\nYour name is: ");
    puts(string); /* Since string is external, it can be
used */
    }               /* anywhere.  */
```

As you can see, since "string" is external, I used it in 2 separate functions (main() and display()) and I only declared it once, in the header. "x", on the other hand, is declared in main(). This means I cannot refer to "x" in any other function without declaring it again. "x" lives within main() only. The same goes for "boogie"...only the display() function knows that "boogie" exists.

Some other brief points: when you *call* another function, place a semicolon after the function name. This tells the compiler that line is a statement. To *define* a function, DO NOT put a semicolon after the function name. This tells the compiler that it will be defining a new function. Furthermore, a function body is enclosed in braces. Following this logic, you can see that main() is a function, and because there is no semicolon following the name, I am indeed defining the main() function. But main() is a very special function. When this program is compiled and run, the main() function is executed first. It is from this function that you call all other functions. It is the MAIN program. Other than this, it behaves as a normal function. The body is enclosed in braces. In fact, you can call main() just like a normal function. If, somewhere in another function, you insert a main(); (include the semicolon), the main program will run again! You can even call main() from inside main()! This is called recursion, where the main program calls itself. Look at this:

```
main()
{
  puts("Recursion");
  main();
}
```

This program will run forever, printing "Recursion" and then calling itself over and over again.

Well, that's it for now. I have briefly covered the difference between automatic and external variables, and how to declare each. Since the only C compiler available right now for us is c99, my comments have been limited to what can be done with that compiler. In other "true" compilers, you have many more storage classes from which to choose, and you have more flexibility in how to declare them. I also quickly covered the difference between calling a function and defining one. Admittedly, this first tutorial was VERY basic, but we all have to start somewhere! In the future, I plan to cover how to handle strings and pointers, two sticky situations at best! If you have any comments or questions, please direct them to me, Warren Agee 70277,2063. Have a ball!
"string" */

DUES ARE DUE THIS MONTH FEBRUARY

SPEECH

By Ronald Albright

(d/l from CompuServe by mel gary)

The more I read about the "new" developments and software for other machines, the more impressed/infuriated I become with Texas Instruments. Whether you realize it or not, TI was light-years ahead of the remainder of the home computer industry in virtually everything except, of course, consumer marketing and common sense. One of the features which remains the industry leader and is, at the same time, the most neglected and overlooked feature available for our machine is the Text-To-Speech access. With the speech synthesizer and the Terminal Emulator II cartridge (or disk-based Text-To-Speech program for XB), you have a feature unrivaled on any other machine. Sure, others have "speech" and some even boast "unlimited vocabulary", but, if you have ever heard these facilities on another machine, you realize how far ahead TI was (and still is) in synthetic speech. What I would like to do in this article is give you an overview of speech synthesis on the TI and, hopefully, revive some interest in this incredible facility.

The chip used in our speech synthesizer is the TMS5220, a p-channel MOS device packaged in a 28-pin DIP. It is a second-generation speech chip, which followed the TMS5100 used in the Speak and Spell toys appearing in 1977. While the TMS5220 is capable of all 3 types of synthetic speech (linear prdictive coding, wave-form modulation, and phoneme-stringing), our machine uses the most memory-efficient form linear predictive coding, or LPC (but has capability for allophone-stringing). LPC in our machine requires a small 3K memory to hold the 128-allophone library, 7K to accomodate the 650-rule TEXT-TO-SPEECH set for translating English-language text into allophonic equivalents and for contouring inflections with the help of pitch modifiers to help make the speech more natural. The allophone library and the rules for stringing them are held in the TEII GROM chips. The synthesizer holds the speech chip and the resident speech vocabulary (memory location >9000).The system is not perfect (as you may have learned HOPEFULLY by experience) but even with this small ROM requirement, TI achieved 92% translation accuracy. You can correct the remaining 8% with changing text.

Let us digress for clarity. Of what do we speak when we discuss allophones? Allophones are the most fundamental of any of the other linguistic components, including phonemes, diphones, and morphs. An analysis of the English language shows that about 40 allophonic sound characteristics can provide the needed variations for all 45 standard phonemes. For example, the phoneme for the letter "p" in English is rounded and aspirated in the word "poke", rounded and unaspirated in "spoke", aspirated in "pie", slightly aspirated in "taper", released in "appetite". These acoustically different "p"'s - so-called voiceless bilabial stops - are allophonic variations of the phoneme "p". Thus, allophonic speech produces better quality than phonemics, because the allophones provide most of the subtle variations each English phoneme can encompass AND use each variation in the appropriate relationship. Phonemic speech sounds mechanical and is limited, allophonic speech is much better though still not perfect...the transition between allophones make the speech sound unnatural and intonations are charcteristically monotonic. But allophonic speech is an ideal compromise based on size of vocabulary, memory requirements, quality and versatility of speech.

So, knowing that we use a allophonic speech system, how does it work, in generalities? Text, from keyboard input, is convereted into the appropriate allophones, which are then converted into LPC data which activates the TMS5220 to generate immediate speech. Well, its not quite that simple. For the text to be converted to the "appropriate" allophones, rules must be applied; 650 rules, to be exact. The rules, based on a U.S. Navy Laboratory system, are complex to say the least. For example, in the process of translating the word "space", the allophone-stringing algorithm looks first at the "s" and supplies an initial allophone for /s/. But for the "p", it finds a rule where the left environment is an "s". Also, since the "p" is not a final sound, the algorithm translates the "p" accordingly. Next the rule is invoked that applies to an "a", where the right-sided environment consists of a single consonant and the word ends with a word-final silent "e". This rule selects the appropriate "long-a" allophone. Finally, the rule for 'ce' inserts an /s/ component in the allophone string to replete the 'c' in the text; the rule says the 'e' is silent. As we have stated, 92% of the time, the rules work...not bad! Compound words give it problems, often easily corrected by hypenating...e.g. "base-ball".

Not only does the TI system convert text to component allophones, it also, through the rule set, translates secondary and primary speech-stress points into pitch variations. Contouring algorithms divide sentences into two major stress profile types: a falling mode, where the pitch level drops following a primary stress point (as occurs in a normal sentence making a statement), and a rising mode, which occurs in sentences terminating in a question mark. This adds even more normal quality to speech. Remember how many times your have heard "Ready to start?"...notice how the pitch varies in a rising tone on 'start'.

So, all in all, a very complex system that TI engineers gave us. We have sparse but utilitarian documentation in the TEII manual. It discusses, ever so briefly) how to access both the text-to-speech via "OPEN

#1:'SPEECH', OUTPUT" and the allophone library directly, through "OPEN #1:'ALPHON',INTERNAL". It briefly defines the manual override feature to vary pitch and slope through the "//XX YY". Perhaps this feature deserves more comment.

You can vary greatly the pitch and slope of speech through the use of the //xx yyy command. I have heard a sparse few program where the computer actually sings. The most recently published was the "ABC SONG" seen in Tigercub Tips (Jim Peterson, Tigercub Software, 156 Collingwood Ave., Columbus, Ohio 43213). Look at that program, and see how Jim changes the pitch and slope to poduce synthetic singing. The key formula is one which were the slope is calculated from the set pitch through Y(slope)=(X(pitch)/10). We are told in the manual (p. 34), that this gives the best results. So, by changing the pitch to simulate singing of notes and adjusting the slope through this formula, we can approach singing. Further, we can set stress points in our own text through use of " ' " (sets primary stress point in a sentence), " _ " (sets secondary stress points within a sentence, and ")" (shifts stress points within a word). So, we need not rely on the 92% accuracy TI accomplishes with the rule set...we can achieve realism approaching 100% with manual symbols input within our text.

Through "OPEN #1:'ALPHON', INTERNAL" we can directly access the 125 alphons (but we said 128; 126 and 127 are pauses) in the TEII Grom library. They are listed in the manual with a rather Spartan description of their use. They are strung together as CHR$ statements; CHR$(10)&CHR$(22)&CHR$(x)....Again, we are allowed to change pitch and slope through manual input. This time, by sending a CHR$(252)&CHR$(xx), where the CHR$ statement following a CHR$(252) sets a new pitch and CHR$(251)&CHR$(yy) where CHR$(251) changes slope to the following CHR$(yy). Stress points can be set with CHR$ numbers 253 (primary stress with rising contour), 254 (primary stress with falling contour), and 249 (secondary stress point). While you can change pitch and slope of allophones, the only way (I know of) to increase the duration of the sound is to string allophones, i.e. CHR$(N)&CHR$(N)&CHR$(N) to increase the duration of allophone "N" three fold. A way to implement the RPT$ function in BASIC would do the trick!

What follows is a marvelous application for what we have learned about speech and allophones. There are other ways to use the marvelous utility of speech. I hope we can revive interest in the easily accesable facility and incorporate its technology into more programs.

PROGRAM 1 by Howie Rosenberg
==========

```
100 CALL CLEAR
110 DIM X(23)
220 DATA 1,3,4,5,7,13,14,15,32,37,59,64,
        69,75,76,77,79,81,83,85,93
230 OPEN #1:"ALPHON",INTERNAL
240 RESTORE
250 FOR M=1 TO 21
260 READ X(M)
270 NEXT M
280 FOR M=1 TO 21
290 N=X(M)
300 PRINT M
310 A$=CHR$(252)&CHR$(21)&CHR$(N)&CHR$(N)&CHR$(N)
    &CHR$(N)&CHR$(N)&CHR$(N)&CHR$(126)
320 B$=CHR$(252)&CHR$(16)&CHR$(N)&CHR$(126)
330 C$=CHR$(252)&CHR$(11)&CHR$(N)&CHR$(126)
340 D$=CHR$'(252)&CHR$(5)&CHR$(N)&CHR$(126)
350 E$=CHR$(252)&CHR$(54)&CHR$(N)&CHR$(N)&CHR$(N)
    &CHR$(N)&CHR$(N)&CHR$(N)&CH R$(1 26)
360 F$=CHR$(252)&CHR$(50)&CHR$(N)&CHR$(252)
    &CHR$(48)&CHR$(N)&CHR$(252)&CHR$( 45)&
    CHR$(N)&CHR$(252)&CHR$(38)
370 ARP$=A$&B$&C$&D$&E$&F$&CHR$(N)&CHR$(126)
380 PRINT #1:ARP$
385 PRINT X(M)
390 NEXT M
400 CLOSE #1
```

PROGRAM 2
==========

```
100 OPEN #1:"ALPHON",INTERNAL
110 DATA 54,53,52,51,50,48,47,45,44,
        43,41,39,38,37,34,32,30,29,
        28,26,25,24,22,21,20
120 DATA 19,18,16,14,12,11,10,8,7,6,5
130 DATA 1,3,4,5,7,13,14,15,32,
        37,59,64,69,75,76,77,79,81,83,85,93
140 DIM X(36)
150 DIM A(12)
160 FOR Y=1 TO 36
170 READ X(Y)
180 NEXT Y
190 FOR Z=1 TO 12
200 READ A(Z)
210 NEXT Z
220 FOR AA=1 TO 12
230 S=A(AA)
240 FOR Y=1 TO 36
250 N=X(Y)
260 PRINT #1:CHR$(252)&CHR$(N)&CHR$(S)
270 NEXT Y
280 NEXT AA
```

ASSEMBLY

By Jay Holovacs

What Goes on Here?
(or an introduction to the Status Register)

The status register is the third user accessable register (after the program counter and workspace pointer) that actually resides on the 9900 chip, and is referenced by all conditional instructions and is also available to your program or the debugger as required.

The value contained in this register is displayed by the debugger (in hex) after each breakpoint. To interpret this value, you must first break each digit down into its 4-bit binary equivalent (A=0110; 4=0100; 15=1111 etc). The first 6 bits indicate the logical and arithmetic results of the last comparison, and provide an indication as to what is occuring in the program.

Bit positions are numbered from 0 to 15 in TI nomenclature with 0 being the most significant (leftmost) bit:

Bit 0: Logical Greater Than--For the comparison of two unsigned 16 bit or (in the case of byte operations such as CB) two unsigned 8 bit values. If the destination is greater than the source this bit is set to 1.

Bit 1: Arithmetic Greater Than--compares values, 15 bit word or 7 bit byte interpreting the first bit as a sign. For example, >7FFF is greater than >8000 because it signs positive (if this is unclear, look at the binary representations: >7FFF=0111111111111111 and >8000=1000000000000000).

Bit 2: Equal--is set to 1 only if both source and destination are exactly equal.

Bit 3: Carry--is set to 1 if the last operation a carry was generated. It is important to understand when this bit is set so it can be used properly.

Addition and Incrementing: this would simply be carrying a one to the left beyond the most significant byte, like the grade school definition.

Shift Left: if a one from the most significant digit is 'rolled' off the left edge.

Subtraction and Decrementing: these are tricky in that they work counterintuitively (and opposite to certain other microprocessors like the 8085 or Z80) in that the carry bit is NOT set when we would normally 'borrow' in the traditional sense. Essentially think of the operation as adding the negative of the number being subtracted (therefore the sign bit is 1) to the destination. For example, DEC R1 is the actually the addition of -1 (>FFFF) to the value in R1, you can see that the only time that a carry does not occur is when R1 starts as 0 and proceeds to -1 (>FFFF). Therefore if you are constructing a loop with a decrementing counter, JOC LOOP would jump back to LOOP until the counter value passes 0.

Bit 4: Overflow--when you are working with signed numbers, the overflow bit is set if the result is inconsistent with the sign convention. For example, adding the two positive values >7FFF + >7FFF produces FFFE (-2) because the sign bit has been 'clobbered' by an overflow. The overflow bit is equipped with logic to detect this condition and can be checked by your program when integrity of a signed result is important. In general, the carry bit reflects the condition of the numbers primarily as 16 and 8 bit unsigned values, and the overflow bit treats them as signed 15 and 7 bit values, but the 9900 does not really differentiate between them, each bit is set or reset appropriately. It is therefore up to the programmer to keep track of how a value is to be treated and use the appropriate flags.

Bit 5: Odd Parity--This bit is set to one if the number of ones in the result is odd, otherwise it is reset to 0. While useful for data communications and some I/O software, this is otherwise not heavily used in assembly programming.

Bit 6: Extended Operation--the 9900 has potential for 15 pseudo instructions to be implemented in software and behave essentially like a BLWP instruction except no address is given because the addresses at which to find the new workspace and program counter are calculated from the XOP value. During the time that one of these is being executed the extended operation bit is set. This feature is not particularly useful in a machine with architecture like the 99/4A and in fact not all consoles even have provision for using this feature.

Bits 12-15: Interupt mask--Hardware interupts, signals for the CPU to stop what it is doing and handle a specific routine, can be assigned to one of 15 different levels in the 9900. Any interupt with a higher number than the mask value will be ignored. Because the CRU has its own interupt masking system which is more sophisticated than the 9900 system, only two interupt masks have any meaning on the 99/4A; LIMI 0 and LIMI 2. A few functions (notably sound generation) require that interupts be enabled briefly fairly frequently, but most programs will run well with 0 in the interupt mask all the time.

****

Conditional branching instructions (Jump On Carry; Jump if Greater Than, etc.) can access the status register directly for their operations, but such values as parity and overflow do not have their own instructions. To read these flags (or to access the other flags from your program) use the STST (store status) instruction to place the contents into a specified register. There individual bits can be tested using an ANDI with a mask and examining the result. (For example, to check for Equal flag, AND the status value with 2 (0010), a result of 2 indicates that the flag is set.)

In addition to the compare instructions which affect all comparison bits, the 9900 automatically compares the result to zero for arithmetic and other instructions. This avoids the need for a separate compare instruction in the the decrement loop example previously. (Note that the DEC must be just before the JOC, otherwise the status may be altered by intervening instructions.)

Not all instructions affect all status bits, so be sure to check the E/A manual concerning the instruction in question.

NO DUES
NO STAR
NO MARCH ISSUE

FOR SALE

Texware Adventure Editor

Use this cartdridge to write and edit your own adventure games. Bill Dubrow is selling the E/A version for $20. Contact him at 463-9415.

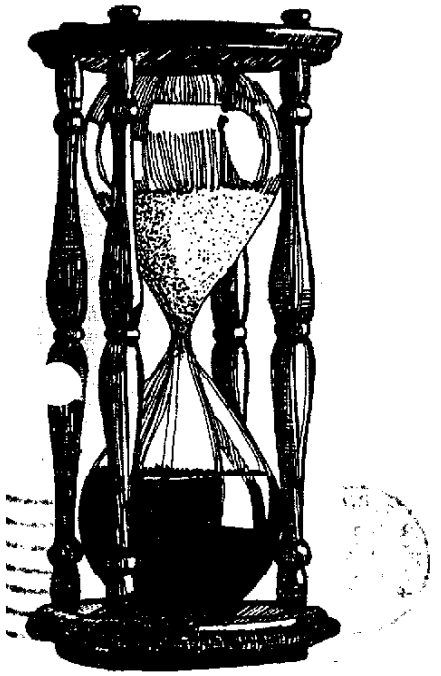
PRESIDENT'S MESSAGE

Steve Citron

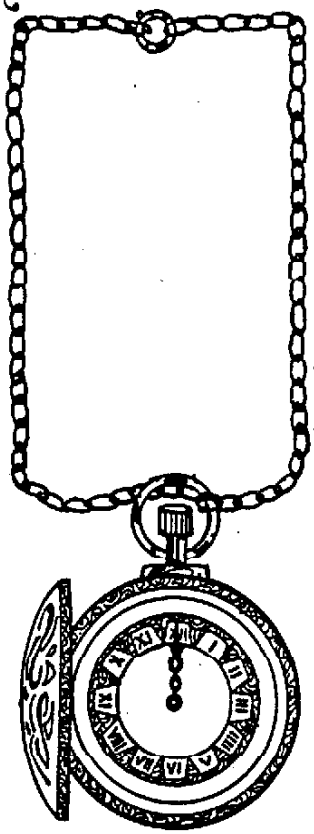
The progress of the TIC OFF has been astounding, yesterday I received an inquiry from a TI user in Spain. because of the large quantities of mail we have opened a P.O. Box (address below).

We will be reorganizing our SIG's for 86 some please complete the Questionairre in last months newsletter and give or send it t[illegible]ll Dubrow. One of the things we are considering are SIGS in TELECOMMUNICATIONS, PASCAL, etc. If you are looking for any TI [illegible]ent or want to sell TI equipment please contact me. I have presently a number of P-boxes available, consoles and P-cards with software. I receive a large number of calls each month and as a service to our members will keep an as available list if you are liooking for anything, etc.

T.I.C. O.F.F. - 86
P.O. BOX 255 DEPT .N
ROSELLE PARK, N.J. 07201



COORDINATOR.....	Steve Citron.....1	201 686-5619
ASST. COORDINATOR		
Advanced sales		
User Group Coordinator..	Jeanette Shader....1	201 929-0532
Bulletin Boards	Jeanette Shader....1	201 929-8161
	Bill Reiss.....1	201 679-0549
VENDORS		
Commercial/Flea Market.	Randy Evans.....1	201 549-5926
PROGRAM		
SPEAKERS ETC.....	Bob Costello.....1	201 663-4512
LEGAL	Kevin O'Boyle.....1	201 762-4696
	Carney Mims.....1	914 961-5993
Publicity	Henry Heins.....1	201 358-9057
	Bill Dubrow.....1	201 463-9415
TRAVEL COORDINATOR	Tillie Blahosky....Monarch Travel	
	1429 US Route 22	
	Mountainside, NJ 07092	
	1 201 654-6211	
Location Liason	Bob Guelinitz 1	201 382-5963 home
	1	201 241-8902 off.
	Roselle Park H.S.. 1	201 241-4550
AREA COORDINATORS		
New Jersey	Jeanette Shader ...1	201 929-0532
	John Simtkins 1	609 939-6028
Delamare	Jack Shattuck 1	302 764-8619
Pennsylvania	Tom Burke 1	215 927-4495
	BBS.....1	215 927-6432
Maine	Mark Ridout.....1	207 797-2104
New York	Art Byers 1	914 528-5402



DAY OF SHOW BULLETIN BOARD... 1 201 241-8902 SAT.MARCH 15, 1986 ONLY
A bulltein board will be set up in one of our exhibition rooms to keep the rest of the world,(those who because of (.....) are unable to attend on their own free will) informed as to the progress, news of the show
etc.

OTHER NEWS:
Second Metropolitan Regional 99'er Conference. will be held on Saturday April 12, 1986 from 1 PM to 6 PM. AT: Queensboro Community College Bayside Queens NY

All User Groups in The Tri-State Metro area are invited. User Group Demos, etc. If you are interested in attending or demonstratin give Steve a call as soon as possible will will try to arrange car pools.
For further information call or Write:

Jeffrey B. Semeo
137-77 45 th Avenue
Flushing N.Y.
11355
1 718 961-5399

HARDWARE SIG IN COOPERATION WITH NEW JUG NORTH WILL START ON SUNDAY FEBRUARY 2
FOR FURTHER INFORMATION CALL:
MIKE BOLITON 845-6317
275 ECCLESTON PLACE
MAYWOOD NJ

NEW JUG NEWS

NEW JERSEY USERS GROUP

DIRECTIONS

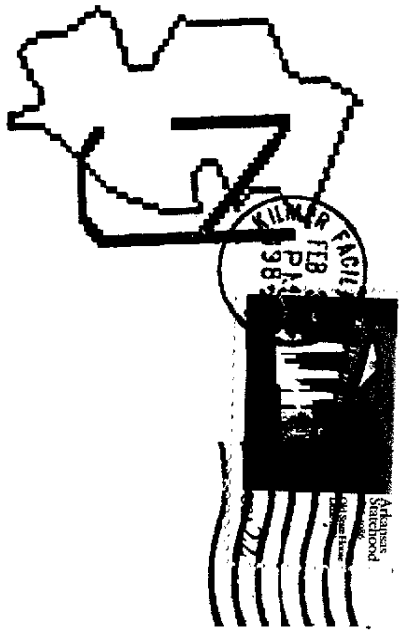
Take Garden State Parkway to exit 131, bear right toward Metuchen on Route 27 (Middlesex Ave.) until you reach the fourth traffic light (Main St.). You will have passed the Library. Turn right at light on to Main St., go one block and turn right onto Library Place, proceed half way up block; Library is on right. Park only on left side of street or on cross street (Linden Ave.). Don't use employees' parking lot.

Or from Route 287:

1. Take 287 south to Metuchen Exit;turn left off exit;
2. Bear right at fork;
3. Road will eventually bear to left;
4. At third light, turn left onto Main St.;
5. Go one block and turn right onto Library Place;
6. Library is 3/4 block down; use rear entrance.

Mel Gary
49 Pine Grove Ave.
Somerset, NJ 08873

DALLAS TI H. C. GROUP
1221 MOSSWOOD PL.
IRVING, TX 75061



THE GREAT AMERICAN

T. I. C. O. F. F.

- 1 9 8 6 -

Texas Instrument Computer Owners Fun Festival

9:30 --- 4:00

MARCH 15, 1986

* Roselle Park High School *

Roselle Park, New Jersey

FEATURING: New MYARC Computer

Educational Program — Hardware and Software Demos

Flea Market — 20,000 square feet vendor area

Lectures, class and workshops — Cafeteria with seating

User Group Meetings and Exchanges — comfortable auditorium

S.I.G. Meetings - Special Interest Groups — Sound system

Software, Freeware, Public Domain Libraries

Ample parking, convenient to major highways

Roselle Park Railroad Station 1 block away

BE THERE THE IDES OF MARCH

For Further Information Contact:

BOB GUELLNITZ
STUDENT COUNCIL ADVISOR
ROSELLE PARK HIGH SCHOOL
1 201 241-8902 OR 241-4550

STEVE CITRON - COORDINATOR
P.O. BOX 255
ROSELLE PK, NJ 07204
1 201 686-5619

-OR-

JEANETTE SHADER
1 201 929-0532
1 201 929-8161 BBS