TEXAS INSTRUMENTS

The Inner Limits

by Glenn Davis

Remember last time that I told you that the Assembler is your friend? And you didn't believe me? Another way to make the Assembler more friendly is to name your registers. This just requires that you add an EQUate to the beginning of the file:

```
TOTAL     EQU 0
COUNT     EQU 1
POINTER   EQU 2
```

And the like. TI Forth uses this trick in its source code. It makes for very readable programs, as you'll see below. One thing you must watch out for: R0 cannot be used for index addressing as in MOV R8, @ARRAY(TOTAL). While it looks innocuous, this statement will always address @ARRAY, and never any other part of the list.

In addition to the alphabetic and numeric characters, "$" and "#" may also be used in names for labels. For example, $KEY is valid, and so is VAL#1. Other special characters will return errors.

Primed values can be used in expressions too, just like constants. Another way to find the right VDP address for loading the standard character set using GPLLNK is: ' '*8 + VDP-BAS. Try it. There are other applications of course.

Speaking of VDP addresses, certainly you've had to calculate screen addresses to display messages. Use an expression in the form Row*screenwidth + column. This allows expression of the screen position like Forth does (BASIC starts at one, while these will start numbering from zero). The fifth row down, second column in the 40-column mode would use: 4 + 1. An entire write sequence would be:

```
MESSAGE  TEXT  'Display this message'
MSGEND   EQU   $
SCRWDT   EQU   40 - SCReen WIDTh
START    EQU   $
         LI    0,MESSAG
         LI    1,4*SCRWDT+1
         LI    2,MSGEND-MESSAG
         BLWP  @VMBW
```

Incidently, these registers do not necessarily need to be named. VMBW always uses the same registers, so little would be gained by naming them. As Leo Brodie, a prominent Forth programmer wrote in his recent book *Thinking Forth*, only consider using "tips" as they apply in your situation.

On another front (I think that one has been worn to death) let's talk about speed. The fastest RAM is the only CPU RAM in the console which is called "scratch pad RAM" (at addresses >8300 through >83FF). The workspace for the GPL interpreter is at >83E0, while the Sieve of Eratosthenes below uses >8300. All 8K of ROM is on the 16-bit bus too. Calling a ROM routine (when possible) will be faster than calling user-written routine that uses the same code because this is faster memory. Besides putting important things on the 16-bit bus by using scratch-pad RAM there are a few things you can do to speed up routines. Namely, by using a different addressing mode.

For instance, using register indirect addressing (*Rx) is faster than indexed addressing [@SYMBOL (Rx)]. When you've got a loop whose index is incremented, this can add significant speed. It takes a few more instructions to set it up, but those within the loop are executed faster. Using DEC or DECT and a JNE is about 16% faster than using CI, C, or CB and JNE. Here is an example of a slow loop to clear a region of memory:

```
SIZE     EQU   8192
BUFFER   BSS   SIZE
START    EQU   $
         CLR   R1
         LI    R2,SIZE
LOOP     CLR   @BUFFER(R1)
         INCT  R1
         DECT  R2
         JNE   LOOP
         RT
```

This version is much faster:

```
SIZE     EQU   8192
BUFFER   BSS   SIZE
START    EQU   $
         LI    R1,BUFFER
LOOP     CLR   *R1+
         CI    R1,BUFFER+SIZE
         JNE   LOOP
         RT
```

And this is a FAST loop:

```
SIZE     EQU   8192
BUFFER   BSS   SIZE
START    EQU   $
         LI    R1,BUFFER
         LI    R2, SIZE
LOOP     CLR   *R1+
         DECT  R2
         JNE   LOOP
         RT
```

As with any other language, most often there are several ways to code a particular algorithm. There were three, and many more exist that are entirely different. Which is most "correct?" It depends on what restrictions you're under. Don't vote on the first one, though. My vote is for the third example; it is clear and fast.

Now turn your attention to the Sieve of Eratosthenes example code (Program 1). For those of you not familiar with this program, it is now a "classic" benchmark. *BYTE* magazine first presented the Sieve in September 1981 and did a follow-up article in January 1983. Ever since then,

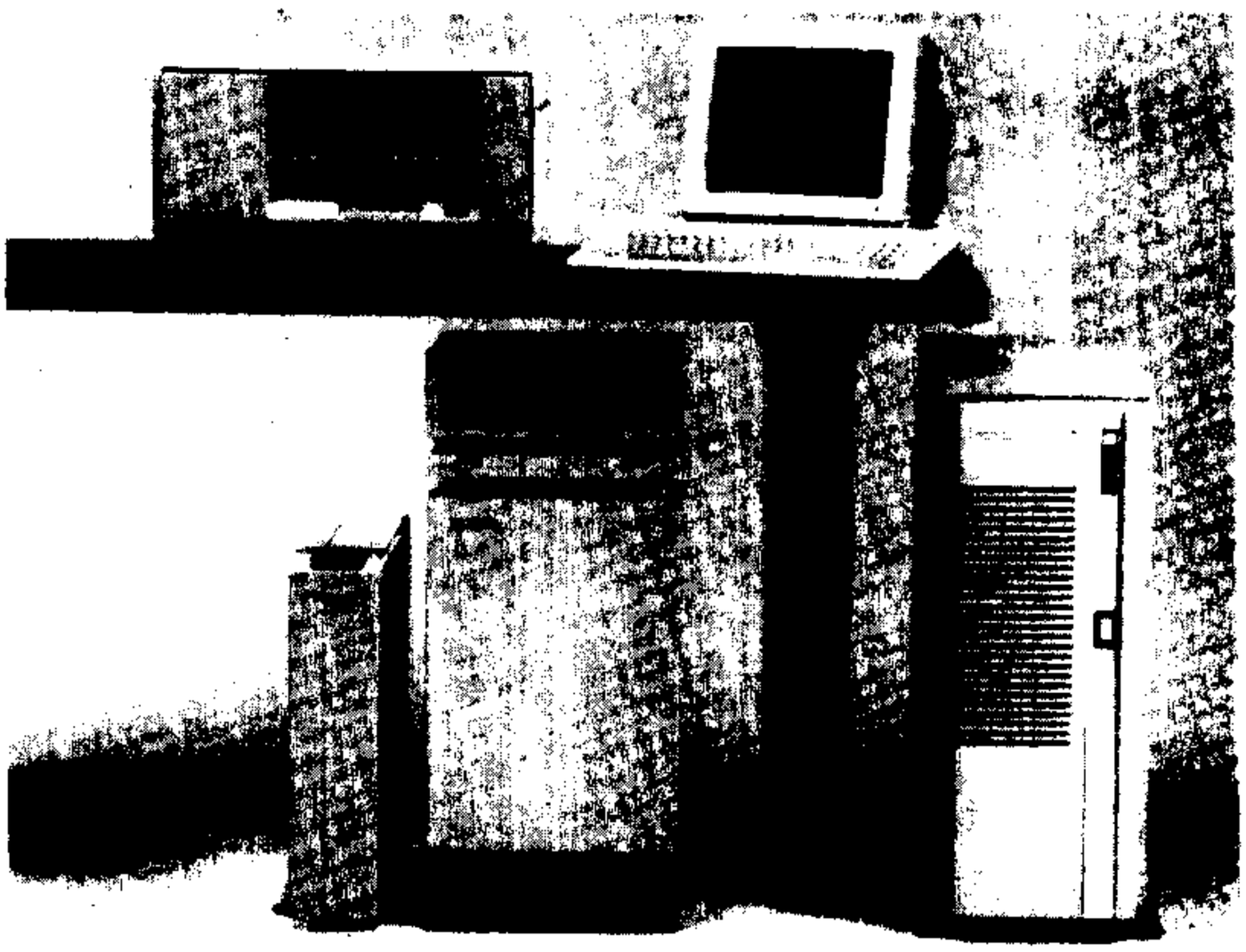
continued on page 92

```
*
*
******
*      Sieve of Eratosthenes
*  in TMS9900 assembly language
*        by Glenn Davis
******
*
*



        DEF  START
TRUE    BYTE 1
FALSE   BYTE 0

SIZE    EQU  8190
FLAGS   BSS  SIZE+1
        EVEN

SIEVWS  EQU  >8300
GPLWS   EQU  >83E0
SAVE11  BSS  2

PRIME   EQU  0
I       EQU  1
K       EQU  2
COUNT   EQU  3
ITER    EQU  4
REG     EQU  5
BOOLE   EQU  6

        REF  GPLLNK,XMLLNK,VMBW,KSCAN
STATUS  EQU  >837C
KEY     DATA >2000
FAC     EQU  >834A
STR     EQU  >0014
CIF     EQU  >2300

SCRWDT  EQU  32

MSG     TEXT ' primes'
MSGEND  EQU  $
SIEVE1  TEXT 'Eratosthenes Sieve'
SIEVE2  EQU  $
WORK1   TEXT 'Working...'
WORK2   EQU  $
PRMPT1  TEXT 'PRESS ANY KEY TO CONTINUE'
PRMPT2  EQU  $
        EVEN

START   MOV  11,@SAVE11    SAVE SYSTEM
        LWPI SIEVWS
```

```
        LI   0,7*SCRWDT+7
        LI   1,SIEVE1           PUT UP INTO MSG
        LI   2,SIEVE2-SIEVE1

        BLWP @VMBW

        LI   0,9*SCRWDT+11
        LI   1,WORK1            PUT UP WORKING MSG
        LI   2,WORK2-WORK1

        BLWP @VMBW

        CLR  ITER
WHILE   INC  ITER               WHILE(ITER<=10)
        CI   ITER,10
        JH   END                {

        CLR  COUNT              COUNT=0;
        CLR  I                  I=0;
        MOVB @TRUE,BOOLE

        LI   I,FLAGS
INIT    MOVB BOOLE,*I+          WHILE(I<=SIZE) FLAGS[I]=TRUE;
        CI   I,FLAGS+SIZE
        JL   INIT

        CLR  I
        MOVB @FALSE,BOOLE
END1    CI   I,SIZE             WHILE(I<=SIZE)
        JH   FINISH             {

        MOVB @FLAGS(I),REG
        JEQ  ELSE               IF(FLAGS[I])=FALSE

        LI   PRIME,3            {
        A    I,PRIME            PRIME=3+I+I;
        A    I,PRIME

        MOV  I,K
        A    PRIME,K            K=I+PRIME;

        AI   K,FLAGS
END2    CI   K,SIZE+FLAGS       WHILE(K<=SIZE)
        JH   DONE
        MOVB BOOLE,*K           {FLAGS[K]=FALSE;
        A    PRIME,K            K+=PRIME }

        JMP  END2

DONE    INC  COUNT              COUNT++; }
```

```
ELSE    INC  I                  I++;
        JMP  END1               }

FINISH  JMP  WHILE              }
END     CLR  1                  |
        MOVB 1,@STATUS

        MOV  COUNT,@FAC
        BLWP @XMLLNK            CONVERT TO FP
        DATA CIF

        CLR  REG
        MOVB REG,@FAC+11
        BLWP @GPLLNK            CONVERT FP TO STRING
        DATA STR

        LI   0,15*SCRWDT+10     VDP ADDR

        MOVB @FAC+11,1          CPU ADDR
        SRL  1,8
        AI   1,>8300

        MOVB @FAC+12,2          # BYTES
        SRL  2,8

        BLWP @VMBW              WRITE # TO SCREEN

        LI   0,15*SCRWDT+15
        LI   1,MSG
        LI   2,MSGEND-MSG

        BLWP @VMBW              WRITE MSG TO SCREEN

        LI   0,23*SCRWDT+3
        LI   1,PRMPT1
        LI   2,PRMPT2-PRMPT1

        BLWP @VMBW

$KEY    BLWP @KSCAN             WAIT FOR KEY
        MOVB @STATUS,0
        COC  @KEY,0
        JNE  $KEY

        CLR  0
        MOVB 0,@STATUS          RESET GPL STATUS BYTE

        LWPI GPLWS              RETURN
        MOV  @SAVE11,11
        RT

        END
```

TI Announces New 32-Bit Series Of Computer Systems



TI Series 1000 Business System

A new generation of computer systems was announced recently by Texas Instruments. The UNIX-based Business System 1000 Series will extend the company's computer line into a new class of high-speed, high-performance business applications.

The first member of the series, TI's Business System 1500, was unveiled during a press conference given in conjunction with the first VAR Congress, a trade show for computer resellers sponsored by the tenants of the Dallas-based computer market, InfoMart.

The Business System 1500 provides the most expandability, flexibility and processing power currently available in a UNIX-based, multi-user business computer. It includes multiple high-speed 32-bit processors, the ability to support up to 128 terminals and 4 billion bytes of memory address space. In addition, the Business System 1500 has an operating system based on AT&T UNIX System V that has been enhanced for multiprocessing and ease-of-use in commercial applications.

"The Business System 1500 is the solution for customers who want to take advantage of the UNIX V environment or who require large numbers of terminals for business applications," said Wally Rhines, president of TI's Data Systems Group. "The Business System 1500 combines advanced technology with industry standards in a computer that gives customers unprecedented configurability and power. With the addition of the Business System 1500," Rhines continued, "TI can support from one to 128 users through a broad offering of business computing products."

The Business System 1000 Series will continue TI's traditional strategy of family compatibility. Currently a COBOL System V programming language provides a compatible COBOL environment between the Business System 1500 and TI's Xenix-based Business-Pro computer. Applications written in Mico Focus Level II Cobol ET, VS COBOL Workbench or TI COBOL System V can be run on both TI's mini and micro products. This gives TI customers numerous options in configuring sytems to run COBOL applications, without the need to re-program applications software.

Flexible, High-Capacity Hardware

The keys to the Business System 1500's flexibility and configurability are the seven-slot chassis and a system bus that allows the use of multiple processors; the system can be tailored to the user's current needs, and processors can be added as requirements change. Each processor board contains a high-performance 32-bit 68020 microprocessor running at 16.67 megahertz and 16 KB cache memory for increased computing efficiency and extremely high throughput. Separate intelligent processors manage mass storage and peripheral input and output which frees the central processors for application processing. (Both the chassis and bus design were developed from TI's Explorer artificial intelligence computer.)

Each processor board uses 256K DRAM chips and advanced surface mount technology to provide 2 MB of on-board, dynamic random access memory; and add-on memory board can boost memory to 4 MB for each processor board. The multiple processors plug directly into a 32-bit NuBus system bus, which allows the processors to efficiently share the system load. The NuBus has a 37.5 megabyte-per-second transfer rate with a 100-nanosecond clock speed, making it one of the fastest system buses currently available. The NuBus provides a full 32-bit path for both physical memory and data transfer, and a virtual address space of 4 billion bytes (4 gigabytes).

The Business System 1500 can have up to 3.6 gigabytes of mass storage using two types of intelligent mass storage controller boards that plug into the NuBus chassis. A Small Computer Standard Interface (SCSI) board supports up to seven 5¼ inch 140 MB Winchester disk drives with a 60MB cartridge tape backup. This disk has a 30 millisecond access time. Another dual-ported mass storage controller board combines SCSI interface with a Storage Module Drive (SMD) interface to support up to seven 5¼ inch 140 MB disks and up to two 9-inch 515 MB Winchester disks with an average access time of 20 milliseconds. This disk is used in TI's current Business System 600 and 800 computers, preserving any investment that customers may already have in that peripheral. A single slot intelligent Communications Carrier Board supports EIA devices, modems, parallel printers, auto call units, and wide or local area communications options.

Terminal Concentrators Expand User Count

Terminal concentrators are used to increase the number of users attached to the system. The terminal concentrators are intelligent units designed for local terminal connection. Each terminal concentrator contains a microprocessor that off-loads part of the network management function from the system processor(s). Up to 16

continued on page 172

The Inner Limits
continued from page 91

BYTE has used the Sieve to test computer systems and language interpreters or compilers. What does it do? It finds prime numbers *fast*. If you missed this in your math education, it starts by eliminating all known non-primes (all even numbers, evenly divisible by two), and then the multiples of the next primes (multiples of 3, 5, 7, 11...). See the articles referenced below.

The Sieve can be coded in any language that can index arrays of numbers. This version is in 9900 assembly language. It is the fastest one I could create without changing the algorithm. (Note that there is a difference between modifying the algorithm and modifying the code used to implement the algorithm.) At 10.85 seconds for all ten iterations (an IBM-PC takes about 6 seconds), it is by far the fastest language for the TI. For comparison, c99 v2.00 takes 88.5 seconds, TI Forth takes 173, and BASIC takes nearly 4,000 seconds!

First look at the label MSG. This uses the TEXT directive to write a string of bytes into memory. The single quotes (apostrophies) delimit the string. The label MSGEND indicates the end of the string. The difference between these is the number of bytes in the string. While MSGEND and SIEVE1 both indicate the same address, they shouldn't necessarily have the same name since latter changes or additions may change the relationship of the strings. If a string is added between them, *every* occurence of either string would have to be found. Just below the label START is a sequence to write similar messages to the screen. The screen position is 7,7 (counting from zero) and SIEVE2-SIEVE1 gives the length of the string. If the string is ever changed, the program automatically changes with it. Otherwise every occurence of that string being written would have to have the count changed. Long programs would make that task difficult, to say the least.

Near the label INIT the routine sets the byte-array to true as the comments in "C" indicate. This is a very tight, fast loop. Near the labels END1 and END2 is the main part of the Sieve that strikes out multiples of primes. After the label END more messages are written to the screen and several system utilities are called (XMLLNK and GPLLNK) to do conversions. At $KEY the Sieve waits for a key press and uses COC (Compare Ones Corresponding) to make the comparison.

Type in all comments that are listed with any source code that you type in. When you have to come back to it later, or have somebody else make modifications to *your* code these comments will allow much easier modification. All too often I see people typing in routines and leaving the comments out. Though this may save some time in typing it in, it will cost you dearly later. While it may not be important on Sieve it's a good habit to get into.

I've covered the inner workings of the Sieve quickly because many of the things covered in the last "Inner Limits" are included in the Sieve, including the naming of registers to improve program readability. INC COUNT is used instead of the equivalent INC R3. Study this version of the Sieve. Figure out why things were coded the way they were. Can you improve on it? If so, I'd like to hear from you. The best way to learn assembly is to study code and then write your own. Good luck. ●

The TI Forum

by Ronald Albright
& Jonathan Zittrain

I was in trouble. My best planet, 522.a. had been left undefended and I was in danger of losing it. "You have messages waiting," confirmed the battle computer, as I prepared to assume command of my ship. Sure enough, planet 522.a had left an impassioned plea for help.

I throttled up past light speed to get to 522.a as soon as possible. Unfortunately, Ming the Merciless, in command of a battle cruiser that could consume quite a few scoutships like the one I was flying and still call it a light meal, had spotted me zipping across the galaxy. "Message from ship number 9082," declared the computer, It reads: "I have chosen you for a space duel. Eat photons." No sooner had I loaded my ship's own meager photon torpedo tubes when my ship was knocked silly by several of Ming's industrial-quality missiles. The resulting battle was not pretty, especially since I succeeded only in dumping my remaining fuel instead of attempting to flee or fight Ming. By the time I had returned with a brand new scout ship (my old one having been reduced to galactic metal chunks) I was informed that planet 522.a had surrendered in order to save civilian lives. It was not one of my best evenings as an interplanetary entrepreneur.

My experience with Ming actually happened. I really did own (and then lose) planet 522.a. At least, it certainly seemed that way. I was actually hooked into a nationwide computer network, the CompuServe Information Service, through my much smaller TI-99/4A computer, and was competing with real people.

In past articles the wonders of telecommunications have been discussed. From the simple act of connecting one's computer to a phone line, the concepts of electronic bulletin board systems and information retrieval networks were born. Through large computers that could have hundreds of people busily interacting with each other simultaneously came the electronic conferencing concept as well. One user could type a message, and have it instantly transmitted to the other users awaiting it.

Multi-player gaming takes the conferencing process one step further. The game described above was "MegaWars III," developed by Kesmai Corporation and licensed to CompuServe. Says John Taylor of Kesmai, "The idea of multi-player games is to provide users with an environment, which they can then act within." In the case of MegaWars III, the environment is that of a galaxy. Commands are available to the user which enable control of his or her "spaceship," as well as to manage acquired "planets." The multi-player concept comes into play when users can actually "see" the ships and planets of others, and proceed to capture or destroy them, as Ming so mercilessly did to my own ship. The presence of a real-time radio aboard each ship allows users to talk to each other, trading hints or making

continued on page 201

Toolkit
continued from page 198

Toolkit. Incidentally, the present Version 2.00 clears up some bugs that appeared in an earlier version.

Chapter 2 is a very valuable summary of differences between IBM BASICA (sometimes called GW-BASIC) and Sanyo BASIC. This chapter alone is worth the price of the Toolkit if you have been struggling to convert programs to or from Sanyo BASIC. I've devoted a chapter in my Sanyo book (and two articles in a Sanyo magazine) to this subject, and found things in this chapter I was not aware of!

Chapter 3 offers patches to correct bugs in the IBM BASIC Compiler BASCOM.CMD, BASCOM.LIB and BASRUN.EXE programs. Compiling a BASIC program speeds it up considerably (sometimes 30 times or more), and, within limits, makes it transportable between MS-DOS machines. I've never compiled a progran, so I can't comment.

The Bottom Lines

If you do any BASIC programming with a Sanyo 55X, especially if you are translating to or from IBM PC BASIC, this Toolkit will be immensely useful. An IBM (GW-BASIC) version of Toolkit is available if you do BASIC programming and have no desire to support Sanyo BASIC. It has five programs (PF-DEFINE and BTA CONVERTER, which are strictly for the Sanyo, are omitted), with the additional advantage that the IBM version of BASIC CROSS-REFERENCE is able to read both ASCII and tokenized files.

BASIC Programmer's Toolkit
Version 2.00
MVP Software
1035 Dallas SE
Grand Rapids, MI 49507
Phone: (616) 245-8376
Price: $39.95 + $3.00 shipping

TI Forum
continued from page 200

threats.

"The key word is "social," says Lee Winter, a former game operator of "GameOp" for MegaWars III. "People have a topic to discuss, and can discuss it—all while playing the game." Winter commanded the "Death Star," a special ship that enforces a semblance of law for the players. "Users can generally do whatever they like within the bounds of the game, but we try to lend a little bit of support to the new user, who otherwise may be quickly destroyed by the more experienced users."

Taylor agrees. "Certain restrictions are required for playability. The software must take some part in limiting the ability of some of the more ruthless players to take advantage of the new players." A favorite pastime for some players is "Imp hunting," whereby a player hides outside a declared neutral area (an Imperial planet), and shoots at anyone appearing there.

"The game allows for anonymity," says Winter. "Aside from a player's chosen handle, you really can't tell anything about him, her, or it." Such anonymity brings out both the best and worst in people, he states. "It's a chance to sit down and do whatever you like. No one sees the person behind the terminal, and that's a license to be more outgoing and unrestrained. After spending a hard day at the office, straining to be polite to someone you would rather throttle, this is a chance to load and fire a torpedo at him—without really hurting anything except egos."

In this way, the method of communication—clacking away at a computer keyboard and watching the messages from others on a screen—seems to add another dimension instead of taking away one. The fact that one can be wearing pajamas and sipping coffee while in the midst of battle is something entirely new, and worth experiencing.

Tony Savage, known to fellow gamers as "White" in MegaWars III and "Cail" in "Island of Kesmai" (another multi-player game, based around wizards and dungeons instead of planets and lasers) has similar feelings about the multi-player games. "I play them because they're social. There's someone—not just a computer—on the other end, if not a whole group of people," she says. The male/female ratio seems to be heavily in favor of the former, she notes. "I've had other players refuse to shoot at me [in MegaWars III] after they found out I was female. Of course, that doesn't mean I won't shoot at them!"

Multi-player games tend to develop their own personalities. With Island of Kesmai, one can simply zip into the dungeon and fight the computer-generated and -controlled monsters, regardless of the presence of other players. Even the loners can't help but return with "war stories," though, Taylor says. Whether the actual play is performed alone or with a few fellow adventurers, a community spirit inevitably develops. Patricia Fitzgibbons, administrator of the Multi-Player Games Forum (an electronic "hangout" for players on CompuServe where they can leave messages to each other and archive helpful hints), remembers the time that the rather powerful dragon in Island of Kesmai was finally conquered by a small group of players. "Forum members went nuts for days! Everyone wanted to know what happened, who did what, etc. Tales of valor abounded!"

MegaWars III actually has a team structure built into the software. One player can declare himself to be a team leader, name his team, and begin to draft members. A team score is then calculated, and posted along with the scores from other teams. The leader of the top team at the end of a game (MegaWars III games run in four week cycles; most other multi-player games don't have such sharply defined beginnings and endings) is then crowned emperor, sent an actual trophy, and allowed to attach "FE" (meaning "former emperor") to his or her handle. The top individual scorer (scores are based on a combination of combat expertise and planetary management) is then named president, and sent another trophy, says Winter. For Winter, his tenure

continued on page 203

LISTING 1

```
index(s,c)
char *s,c;
{
 int n;
 n=1;
 while(*s) {
     if(*s==c)
       return(n);
     else {
       ++s;
       ++n;
     }
 }
return(0);
}

strncpy(s1,s2,n)
char s1[],s2[];
int n;
{
 int a;
 for(a=0;a<n;a++) {
    s1[a]=s2[a];
    if(s1[a]==0)
      break;
 }
 if(a=n)
   s1[a]=0;
}
```

LISTING 2

```
in_string(string,cnt)
char string[];
int cnt;
{
 int i,c;
 for(i=0;i<cnt;i++) {
    c=getchar();
    if(c==-1)                  /* ctrl-z */
      return(-1);
    if(c==10) {                /* enter */
      string[i]=0;
      return(i);
    }
    if(c=='\b') {              /* backspace */
      i=i-2;
      putchar('\b');
      continue;
    }
    string[i]=c;
 }
 string[i]=0;
return(i);
}
```

LISTING 3

```
get_string(buff,n)
char *buff;
int n;
{
 char temp[81];
  /* change this to the max. length of the string */
 int u;
 u=in_string(temp,n);
 if(u==-1)
   return;
 strcpy(buffer,temp);
 return(u);
}
```

TI Forum
continued from page 201

as GameOp has provided insights on quite a bit of human nature as the struggle for the titles commences. "It's almost like class warfare. On the top, you have the most experienced players, who are quite serious and organized in their team leadership. They sometimes require team members to have a phone number where they can be reached in emergencies, in case some extra firepower is needed to retain a planet or capture a new one.

"From the top two or three teams, there are then five or six other teams, without a real chance at winning, but who play just for some quick shooting or rampaging. At the bottom there's always the teams started by players who've become a bit overzealous in becoming leaders, when they themselves have only begun to play." Fitzgibbons agrees, declaring that some team members have been associated with their teams for literally years.

As might be gathered from the description of the games, they are quite an investment in time, and since on-line time is charged at an hourly rate, they're also an investment in money. "I suppose I could have put a down payment on a house," pines Oberon, another well-known player. "Then again, I've never had this much fun paying bills."

The structures of the various games also require different minimum investments of time from the players. In MegaWars III, a fairly heavy usage over a four-week period is necessary to win. "The amount of money in one's entertainment budget is definitely a factor," says Lavrenti Kutuzov, an avid MegaWars III player and assistant administrator for the Multi-Player Games Forum. "Those who have the most money can afford to spend the most time playing the game. Some spare time is also needed!" Detailed formulae for planetary management have also been developed, since there are so many variables. The player can become quite serious, if he or she wants to. On the other hand, games like "You Guessed It!" or MegaWars I are not as intense. "I like MegaWars I so much better [than some of the other games]." says Gary Shook (also known as Admiral Arrakis), "because each game there is no more than five or six hours in duration. I can jump in, have my fun, and jump right out again." The type of commitment desired is a major factor in deciding which game to become involved in. "You Guessed It" is a real-time trivia game, where players match wits (with a real "audience" as well) and gain credits for actual prizes. Even in Island of Kesmai, one need only play when convenient. "I think you have to decide what will give you the most bang for your gaming dollar," says one player. "I like to enter a dungeon in Island of Kesmai, meet other players by chance, and then leave whenever I choose."

Whomever the player may be, the comment is always the same: "I don't know what I did before I started playing! To think I actually enjoyed sitting and staring at a TV set! [slight pause, as the speaker realizes that's what he is still doing while playing the games on the computer]. Well, at least here I get to COMMUNICATE with people. I get to forget about everything else for awhile, and really have some fun."

I waited impatiently as the computer digested my command to attack 522.a. I had spent my last amount of galactic money on the attack force, and was determined to recapture the planet. "There is no resistance. The base surrenders!" declared the computer. I leaned back in my chair and took another sip of coffee. It hadn't been such a bad evening after all.

Changing Your Clock...

Found this pearl in the *R/D Newsletter* (Ryte Data, 210 Mountain Street, Haliburton, Ontario, Canada KOM 1S0; $14/year). As usual, the latest issue of this great monthly carried the following information about changing the clock crystal of the TI consoles to speed up console function (and, thus, program speed). As usual, try it at your own risk. But it does work.

"To speed up a standard TI console by approximately 19%, do the following:

Obtain a 14.318 MHz crystal, a one-pole—two-position switch, 3 pieces of wire, about 6 inches long and a soldering iron. Unsolder one lead of the existing 12 MHz crystal and solder in one of the wires. This wire will be the "pole" wire (i.e. it goes to the center connection of the 3 connections on the switch. Solder another wire to the bare lead of the 12 MHz crystal.

One lead of the 14.3 MHz crystal is soldered to the last wire, the other lead is soldered on the 12 MHz's lead that is connected to the PC board. In other words, all that is being done is a switch from one crystal to the other via the switch. The two wires coming from the crystals are simply soldered to the switch so that in one position the 12 MHz is selected and in the other the 14.3 MHz crystal is selected. Simple, heh? It turns out the wire lengths, type, etc. is not critical. I used 22 gauge solid wire 6 inches long and a mini-toggle switch. Here comes the neat part: It is possible to switch clock speed while the program is running! (at least in BASIC as that is all I have tried so far...). The hardest thing is trying to find a good location for the switch. I put mine peeking out of the cooling slots on top of the console."

Being the usual cynic that I am, I asked one of the world's greatest TI gurus, Paul Charlton (author of Fast-Term and currently a major force in developing the new Myarc computer), about this hocus-pocus. Paul had the following comments:

"Well, changing the crystal is fairly easy, I think I got my crystal from a Radio Shack store...16.000 MHz...problems you may see with this modification:

1) Some programs do i/o to the VDP chip too quickly, and they'll generally screw up...though 'tis no worse than my 16-bit modification.

2) A more serious problem: the RS232 baud rates get changed by a factor of 33%...I had to special modify a version a Fast-Term to take this frequency change into account, also this modification only works on consoles which have a 12.000 MHz crystal to start."

Paul further said that the oldest consoles had a 48 MHz clock crystal and those after early 1983 had the 12 MHz crystal. The clocks speed should be clearly printed on the crystal. Again, do the modification only if you understand what you are doing. Thanks to Bruce Ryan at Ryte Data and Paul Charlton of Myarc for that information.

More On c99...

I have been totally overwhelmed at the response to the c99 information presented in your TI Forum. Your letters have been great. I have to insist, though, that if you want a personal reply, you must send a self-addressed, stamped envelope. We promise that IF you do that, we will write back. The c99 material has been very popular and we appreciate your letting us know you like it. We do plan to continue our support of this language in the future.

continued on page 205

TI Forum
continued from page 203

One of the really nice things about using Compuserve's TI Forum is that you get to fraternize with some real geniuses. The Forum (really, the TI Special Interest Group or GIS) is frequented by a large number of the real "movers-and-shakers" in the TI community. One such genius is Warren Agee. Warren lives in Lavonia, Michigan, and is, truly one of the most knowledgeable c99 programmers I have met. He has written the first commercial program written entirely in c99 (the soon-to-be-released "Article Filer" from Asgard Software, P.O. Box 10306, Rockville, MD 20850 and numerous tutorials and program source code files (available on Compuserve's Forum for downloading). He has also taught me everything I know about this language. A real programmer and teacher. He offers as an exclusive to the *Computer Shopper* TI Forum the following information. Remember, you read it here first!

Warren Agee's c99
Tips And Tricks

Compuserve ID 70277,2063

BUGS found on the c99 disk:

The following string functions, found in the file STINGFNS on the c99 2.0 disk, do not work properly. The corrections are shown in Listing 1.

Two **NEW functions:**

x = in_string(string,cnt)

A replacement for gets(). "string" is the char array that will hold the string."cnt" is an integer that specifies the maximum length of the string to be inputted. The function returns the actual length of the string. Returns a -1 if a ctrl-z is hit. Backspace is supported, as in gets(). This function is shown in Listing 2.

The second function requires in_string and strcpy to be defined in the same program, and is called just like in_string. It is also an input routine; however, if one enters a ctrl-z, and text that was in the string prior to calling get_string remains unchanged. See Listing 3.

The functions "in_string" and "get_string" were developed by Warren and are extremely useful. Try them out. Warren is now working on the first truly relational database for the TI 99/4A and it will be written in c99. Thanks to Warren for giving the Forum this exclusive.

Fairware Does It Again!

Canada (and Ontario in particular) has really produced some incredible Fairware products. Disk Manager 1000, the c99 Compiler, and others. Well, the Canadians have done it again. RAG Software (R.A. Green, 1031 Chantenay Drive, Gloucester, Ontario, Canada K1C 2K9) has released a Macro Assembler. This package comes as 2 disks and contains extensive on-disk documentation. This package, like c99, makes assembly language more friendly. By using the macro library which comes with the package, you can accomplish several assembly language statements with one macro, thus speeding and simplifying your code. It's kinda hard to explain (there is an excellent review of the package by John Clulow in the July *MICROpendium*), but it's almost as easy as c99 and just as fast. Another major programming tool for the TI. Check it out.

Gary Cox, one of the driving forces behind one of the top TI users groups in the country, the Mid-South Users Group (P.O. Box 38522, Germantown, TN 38183; you can subscribe to their terrific newsletter for $10/year) also offers some unique Fairware for the usual disk and pre-stamped, self-addressed mailer. Weather Forecaster, the only program of its kind for the TI. does just what the title suggests and with graphics. He also has a BASIC database. Try and send $'s if you use them. Thanks for sharing Gary. ●

Expert Systems
continued from page 204

cially available for only a few year, the technology has proven its reliability and effectiveness in field use. A few applications have been abandoned, but most have become proven successes. Costs are no longer a barrier, many excellent hardware and software products are readily available, and many companies offer experienced assistance.

The risks are small, the rewards are great, the time is now.

Viewpoints

Question: What educational and professional backgrounds are good foundations for careers in knowledge engineering?

Bill Turpin, manager of AI applications for TI's Data Systems Group:

"A technical background is very helpful because a knowledge engineer is really a modern version of a systems analyst. So, computer skills are essential, problem solving skills are essential—and yet, many people trained in those disciplines lack some of the interpersonal and social skills that are also necessary. Therefore, I wouldn't say that you would have to have one of those technical degrees. It might be that a person in the area of geology or art would have enough technical skills to be able to do the job and might even be better at some of the social skills and interviewing skills...maybe even a combination of journalism and computer programming."

Michael Smith, member of the group technical staff for TI's Industrial Systems Division, and knowledge engineer for the Campbell Soup expert system (See November 1985 edition of *AI Letter*):

"I was just talking to a friend of mine about that, and one thing we're noticing is that people with backgrounds in education and psychology are becoming knowledge engineers. I read a quote by Ed Feigenbaum* in *Building Expert Systems*, and he said the "knowledge engineer practices the art of bringing the principles and tools of AI research to bear on difficult applications problems requiring experts' knowledge for their solution," which would imply that the knowledge engineer has to be a computer scientist of some sort. But it seems to me that recently some people are calling knowledge engineers people who simply go out and extract knowledge and relay that knowledge to somebody who can put it into the computer."

*Edward A. Feigenbaum is a professor of computer science and a principal investigator of the Heuristic Programming Project at Stanford University. He is a co-author of *The Fifth Generation*.

From: AI Artificial Intelligence Letter published by Texas Instruments Data System Group.