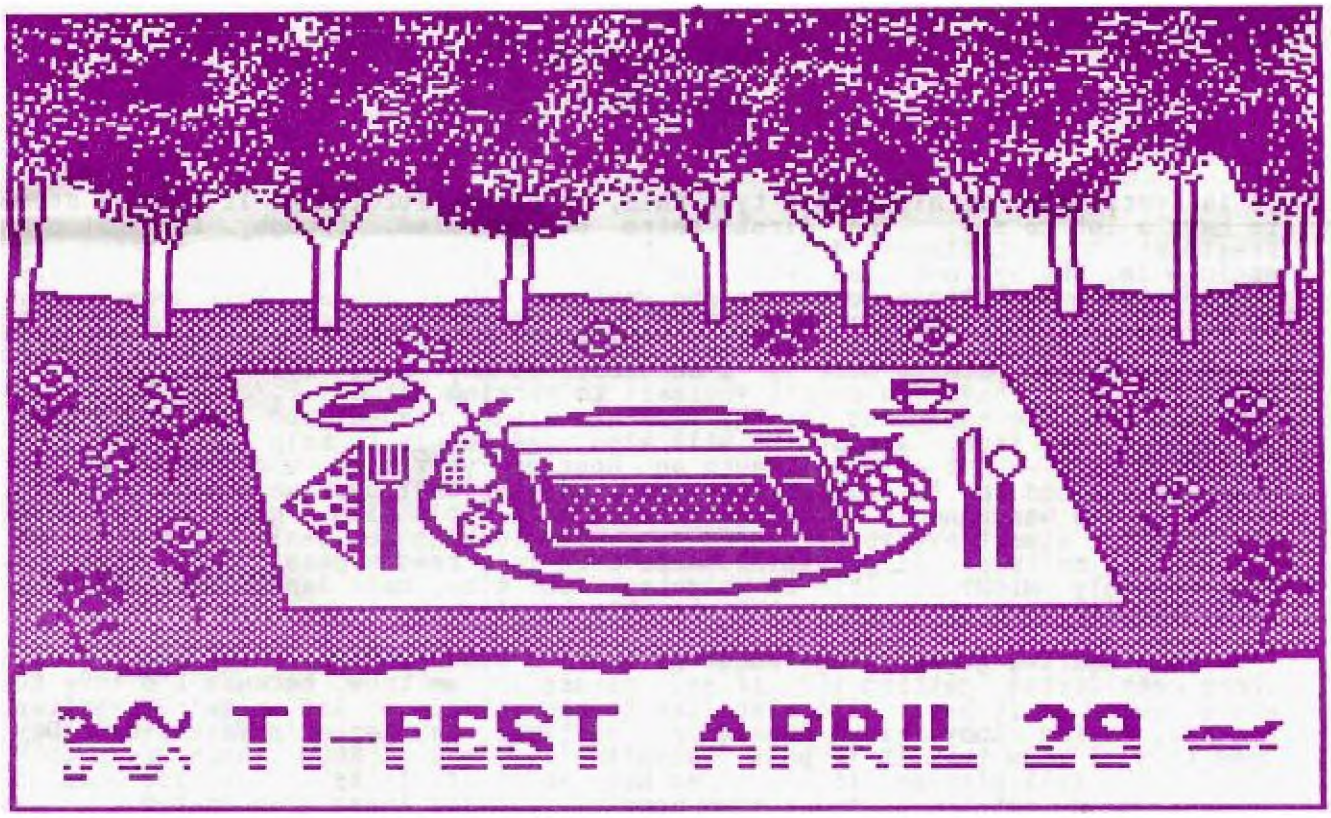
The Ottawa T.I.99/4A Users' Group

NEWSLETTER

VOLUME 8 NUMBER 4.....April 1989



DON'T FORGET THE MEETING -- April 4, 1989

P.O. BOX 2144,STATION D,OTTAWA
*** ONTARIO,CANADA K1P 5W3 ***

COMING EVENTS

Beginners' Assembly Language Class	April 1, 1989 1:00 p.m.	Merivale High School For more information Contact Bill Sponchia.
April Meeting:	April 4, 1989 7:30 p.m.	Merivale High School
4th Annual TI-FEST	April 29, 1989	Merivale High School Contact Jane Laflamme for details or to volunteer your help.
May Meeting:	May 2, 1989 7:30 p.m.	Merivale High School
Newsletter Deadline:	April 15, 1989	(or any time before that!)

EDITOR'S NOTES
from
Ruth O'Neill

It is very late at night as I type this, so please forgive me if I don't seem to have a lot to say. The first thing on my mind, though, is that our President, Jane Laflamme, deserves a huge round of applause for getting her article in. She has not been well, lately, and I did tell her not to worry about getting an article done, but she went and did it anyway. Well done, Jane. As well as applause, Jane also deserves more support for the FEST. The most important thing to remember is that the FEST is so much more enjoyable for those who participate to make it a success. It may sound strange, but it is true. You don't have to commit yourself to working all day... crews to set up and take down the equipment for the FEST are always needed, and you would have your whole day free, almost. We will also need people to help out in the used equipment room -- not all day; even an hour of your time would help. People bring new items in throughout the FEST, and the VERY best way to be there to snatch up the bargains is by being there to help. It's also a great way to meet the people; almost everybody spends some time looking for deals on "pre-owned" hardware or software. If you think there even is a remote possibility that you just possibly MIGHT be able to volunteer some time, call Jane TODAY. She will find a job for you that suits what are able to offer.

I've been hearing great things about Al Beard's 9640 FORTRAN. Is anybody out there considering getting it? If so, please let me know, because I'd love to see a review of it here. I'd also like to see reviews of Art Green's TI-Writer upgrade, Carfax Abby, Floyd Donaldson's software, Ray Kazmer's Valentine's Day game (Yes, I know that it is past Valentine's... but I KNOW there are people out there still playing it.) --- so how about it, folks? There are LOTS of great programs out there, and I know some of you have those I've listed.

Geneve owners: ABASIC is out and up on the Networks, and for those who found lots of bugs in the first release, a new version is up on the nets, with all known bugs as of March 27, 1989 corrected. While you're there, look for LOADERMAKER, a little utility Charles Earl released as fairware recently. It's for Geneve users, and you can use it to create GPL loaders for your favorite E/A option 5 programs. It makes loading some software from MDOS nearly painless. Hope you like it.

The President's Two Cents' Worth
by Jane Laflamme

I am sorry for my absence in group activities in the last couple of weeks, but another bug managed to catch me this year. For one who usually has had good health, I am finding this year quite a difficult one... I would like to thank Bill Sponchia who has kept things under control while I have been indisposed. You are a real "brick"!

Thank you Jack McAllister! I hear that your seminar for console-only users was quite a success...

The meeting for April will feature the Horizon RAM disk and its many uses. (If you would like to show us how you use your RAM disk, bring it to the meeting and show us!)

The TI-FEST -- my mind boggles at the thought. Because of my sickness and other unforseen problems, we are far behind schedule. Al Palmer has been calling for systems, but we still require more. Al also will be asking for your help which I need desperately; people to help set up on the day, plus crew chiefs. A meeting will be scheduled in about two weeks for all the volunteers. To date I have had four. I know others are planning to help, but you haven't called me! I don't know who you are!! Four people cannot put on a show of this size. Please set my mind at rest and let me know.

For you out-of-towners who are planning to attend:

"Official" hotel - Talisman Motor Hotel, 1376 Carling Ave., Ottawa, Ont., K1Z 7L5 (613) 722-7601. If you make reservations on your own (invitations to individuals and user groups are on their way to you by the time you read this), please mention you are with the **TI-FEST**. Our hospitality suite will be free if we book a required amount of rooms, so this is very important. And don't forget to come to the hospitality suite whether you stay at the hotel or not. The conversations held in that room are not to be missed!

The event will be held, as in the past, at the Merivale Highschool Cafeteria on Merivale Rd. (corner of Viewmount) and run from 9 a.m. to 5:30 p.m. We have confirmation from several high profile TIers from Canada and the U.S.

A dinner will be held at the Talisman on Saturday at a cost of $25. This has proven to be quite a popular event and a way of winding down the festivities. As tickets are limited, make your reservations early, please.

If you wish to attend and haven't heard from us by mid-April, please write to me c/o Laflamme & Wrigley (the address is in the advertisement at the back of this Newsletter) and let me know how many will attend. If you require us to book your hotel accommodation, tell us how many rooms and how many tickets for the dinner you require, and enclose a with cheque for the dinners please! (Made payable to the Ottawa TI-99/4A Users' Group).

We are looking forward to seeing both new and old friends on the weekend of April 28th, and let's hope the weather is as good as it was on Easter Weekend!

'Til next month...

VOLUNTEER FOR THE TI-FEST!

Call Jane and offer her your support today!!!! She deserves it, and you will enjoy the FEST all the more for it. Many choice positions are STILL available, but they are there for a limited time only! They will be gone by April 30, and it will be too late to say, "I would have helped!" Call NOW! Operators are standing by to take your pledge.

BROWSING THE LIBRARY
--with DAVE MORRISON

I was very pleased to see that our Newsletter is receiving more contributions. Of course, I am particularly pleased because it means that I can continue saying very little each month - and get away with it! Actually, there is not much that can be said about any Library at any time, other than to report on new acquisitions - none this month!, activity - very quiet (other than sales at monthly meetings).

As Spring approaches, we also approach our Fourth Annual TI-FEST. The last three were outstanding successes and from all accounts the latest should be a real knock-out! The Library will be available from the minute the doors open, right through to closing time. If you wish, you may peruse the catalogue; leave your list and attend seminars, visit various vendors, etc., and when you return, your disks will be ready. For you out-of-towners, now is the time to send me your list of requirements. Incidentally, please make sure that you specify whether you need single or double-sided disks!

In anticipation of your requirements, I have been (and will be) busy, copying disks which I think will be in demand. Unfortunately, I have no way of knowing the version number of the latest updates so that the version I offer is the latest update that the Library has received. This is also hint that I would like to be advised when any disk has been up-dated!

For those of you who attended last month's meeting but were too slow or too tired to take advantage of Jim Patterson's free offer, I will have a few more of his excellent TIGERCUB MUSIC disks available on Tuesday, April 4th.

THE TI-FEST USED EQUIPMENT SALE
notes from Lucie Dorais

Last year, I wrote some notes in the Newsletter concerning the best way to organize your stuff for the sale, to no avail: at 9 o'clock on the morning of the Fest, there were hordes of selling people at the door, some with nothing prepared, no price marked on the objects, no lists for the sales people... Worse, there were also hordes of buyers ready to grab your goodies! Some nightmare, but the rest of the day went OK, many people found what they wanted, and the sellers were happy.

This year, I will not be able to spend the day at the sale; I got one name for help, but this is not enough: those who have visited the "garage sale" in the past years know the atmosphere... Six people for the rush hours (morning, around noon when the Montreal bus arrives, end of day) seems like a minimum.

So this note is intended to 1) find some help (phone me, or leave a message on our Texlink); 2) repeat my instructions on how to help us sell your goods:

(1) PLEASE mark every piece CLEARLY with your name and maximum price wanted.

(2) PLEASE prepare a list in advance, in duplicate (one for us, one for you), following this example. If you don't want us to discount the item at some point during the day, please write "NO" in the "Minimum price" column.

NAME: _____

	ITEM name	MAXIMUM price	MINIMUM price	SOLD for
1 -	_____	$_____	$_____	$_____
2 -	_____	$_____	$_____	$_____

etc.

A Look At Assembler Language -- Switches
by R. A. Green

Most programs have switches and some have a lot of them. Most switches have only two values: YES/NO, TRUE/FALSE, ON/OFF. It is this type of switch that we will look at. Most program options are this type of switch. For example, the Assembler itself has may switches: compressed text - ON/OFF, listing - ON/OFF, cross-reference - ON/OFF, etc.

It first may seem very simple to set and test such switches, but we will see that with some thought it can be done at less cost than your first try. OK, let's look at the obvious way.

```
*
* Switch Definitions
*
ON      TEXT  'N'       Value for ON
OFF     TEXT  'F'       Value for OFF
*
SW      BSS    1        A typical Switch
*
* Setting Switch Values
*
        MOVB  @ON,@SW   Set switch ON
        MOVB  @OFF,@SW  Set switch OFF
*
* Testing a Switch Value
*
        CB    @SW,@ON   Is switch ON?
        JEQ   ISON      Jump yes
*
        CB    @SW,@OFF  Is switch OFF?
        JEQ   ISOFF     Jump yes
*
* Testing a Switch Using the IFB macro
* Generates exactly the same code as above
*
        IFB   @SW,EQ,@ON,ISON   Jump if ON
        IFB   @SW,EQ,@OFF,ISOFF Jump if OFF
*
* Reversing a Switch Value
*
        IFB   @SW,EQ,@ON,WASON
        [illegible]
[illegible]
```

Before we evaluate the "cost" of doing switches this way I should point out that both the "storage" and "time" cost of a comment statement is exactly zero! You can and should use lots of them in your programs. (We defined these "costs" in last month's article.)

The storage cost of the switch itself is one byte. Setting a switch has a storage cost of 3 words and a time cost of 5 words. Testing a switch has a storage cost of 4 words and a time cost of 6 words. Reversing a switch value has a storage cost of 11 words and a time cost of 17 words.

I know you have already devised a better way to do switches, but remember I said the "obvious" way, now comes the "non-obvious" way (which will of course become obvious before we are done). You should read up on the ABS instruction, which gives the absolute value in a word of storage. When you read up on it you should notice that the STATUS bits are set according to the value of the word before the instruction is executed. That doesn't affect us now although it is interesting (as an aside, what other instructions set status according to the before value, what instructions do not set the status). Note, however, that the instruction:

```
ABS   @A
```

as well as giving the absolute value of A will tell us whether the value was zero or non-zero. Also notice that if A was zero it remains zero and that if A was non-zero it remains non-zero (with its sign possibly changed). If switches had a value zero when OFF and a non-zero value when ON, then

```
ABS   @SW
JNE   ISON
```

is a better way to test switches (less cost) than our initial try. It is convenient that the 9900 has a easy way to set a word zero or non-zero:

```
CLR   @A      Sets A to zero
SETO  @A      Sets A to >FFFF, non-zero
```

Thus setting switches this way has less cost than our first try. So far we have improved on setting and testing, can we do as well on reversing the value of a switch? Consider this:

```
ABS   @A
DEC   @A
```

This is certainly getting non-obvious, but looking closely, after the ABS instruction our switch A has the value zero or +1. The DEC makes the zero into -1 (non-zero) or makes the +1 (non-zero) into zero, thus reversing the switch value. Compare the costs of reversing a switch this way to our original method -- quite a saving.

It's time for another plug. A program full of non-obvious switch setting, testing and reversing is going to be a program that is hard to read. I certainly don't like non-obvious programs. Can we use non-obvious switches that are cost effective and still have a readable program. Of course we can! We write a MACRO that hides the actual code, and at the same time reduces the number of lines we have to write and type (the third cost of a program). With the appropriate macro written we could code the following.

```
* Setting Switches
      SW    @A,ON         Set switch A on
      SW    @A,OFF        Set switch A off
      SW    @A,REV        Reverse switch A
* Testing Switches
      IFSW  @A,ON,ISON    Jump switch A on
      IFSW  @A,OFF,ISONF  Jump switch A on
```

I probably should leave the coding of the "macro definition" as an exercise for the reader, but this time I won't. Note that the following two macro definitions look surprisingly like small programs themselves.

```
$MACRO SW             Start of macro defn
$REM   &P0 is the label field
$REM   &P1 is the switch name
$REM   &P2 is how to set the switch
$GOTO  &P2            Case: ON, OFF, REV
$LABEL ON             Case ON, Set switch ON
&P0    SETO  &P1
$EXIT                 Macro generation done
$LABEL OFF            Case OFF, Set switch OFF
&P0    CLR   &P1
$EXIT                 Macro generation done
$LABEL REV            Case REV, Reverse switch
&P0    ABS   &P1
       DEC   &P1
$END                  End of SW macro defn


$MACRO IFSW           Start of macro defn
$REM   &P0 is the label field
$REM   &P1 is the switch name
$REM   &P2 is the test value, ON or OFF
$REM   &P3 is the jump label
&P0    ABS   &P1      Test the switch
$GOTO  &P2            Case: ON, OFF
```

```
$LABEL ON              Case ON, Jump if sw is ON
       JNE   &P3       Jump sw ON (non-zero)
$EXIT                  Macro generation done
$LABEL OFF             Case OFF, Jump if sw is OFF
       JEQ   &P3       Jump sw OFF (zero)
$END                   End of IFSW macro defn
```

One or two final notes. Notice that the switch name, the operand of the ABS or DEC instruction, is a "general address" and thus can be specified in any of the 5 general address forms. Notice also that the target of the IFSW must be close since a JMP is used. You may want to modify the IFSW macro to add a "long jump" option, say by adding an "L" suffix to the ON or OFF.

Next month we'll look at "calling subroutines".

"Make it possible for programmers to write programs in English and you'll find that very few programmers can write in English."

TEACHING PR BASE (Part One)
by BILL SPONCHIA

Editor's note: The following article, which we will be running as a series, is not a replacement for the PR-Base documentation. It is a set of notes from Bill's "How to Run" seminars for those who wanted to learn PR-Base. We are publishing it so that, for those who want one, it can serve as a guide for those who may run similar seminars in the future. Used in conjunction with the PR-Base documentation, it could also serve as a set self-teaching guidelines. I hope you will find it useful.

"HOW TO RUN"

Program: PRBASE Version 2.0

Section I - General Information

Requirements:

(a) Hardware - Disk Drive (minimum - 1 single sided)
32K card
Module (XB or E/A)
RS232 card and Printer (not absolutely necessary but ...)

(b) Software - (1) Program diskette, which must be named PRBASE, containing the following files:

CHAR - special character set
CRT:1
CRT:2 - 3 files making CREATE program
CRT:3
LOAD - for loading from XB
PRB:1
PRB:2 - 3 files making PRBASE MANAGER
PRB:3 program
PRBASE - to give initial menu
UTIL1 - for loading from E/A Option 5 or TI-Writer Option 3

(2) Diskette containing PRBASE documentation. File called PRB:DOC

(3) Initial Use - blank diskette for data file
Subsequent Use - diskette containing data file

Run From - XB, E/A Option 5 or TI-Writer Option 3

Booting up

- After booting up you will get the menu:

1. Database Creation	- must be used the initial time plus whenever changes are made
2. Data Management	- used to put data into the database and to work with the data file
3. Exit	- to get out of PRBASE

Section II - Initial Use of Program

1. Press "1" to get into the Create program.

   After the title screen is displayed, the colours may be changed by pressing "F" for foreground colour and "B" for background colour

2. Press any key (except F or B) to get the "CREATION MENU"

1. Select Data Disk Drive	- to indicate drive data file is in
2. Format Data Diskette	- to format diskette for PRBASE
3. Design Data Screen	- to create data screen
4. Design Tabular Reports	- to create reports forms
5. Design Mailing Labels	- to create labels
6. Set Printer Codes	- to set up printer codes
7. Set System Options	- to set system information
8. Exit	

When you are in one of these options, pressing FCTN 9 will bring you back to this menu. When initially running the program you must use Options 1, 2 and 3 in that order. The other 4 options can be used in any order and do not have to be used initially; however, for convenience they are described in this section.

Option 1 - it will ask which disk drive # will contain the data file diskette. Enter the drive # (1 to 5) and you will automatically be sent back to the menu. The default drive number is 1.

Option 2 - this is to format the diskette for PRBASE use. When going into the option, a series of questions will be asked and then the diskette will be formatted.

Note 1 - this step must be carried out even if the diskette has already been initialized with a disk manager.
Note 2 - if formatting double-sided, you must later go into Option 7 to designate your data disk as double-sided. Failure to do this will result in being able to use only one side.
Warning: in subsequent use of the Create program for this data file do not use this option.

Option 3 - this option is used for designing the data screen.

i) At the bottom of the screen is a row of graphics with another row of characters directly under them. To use the graphic designs press CTRL ? (where ? = the character under the graphic design you wish. The use of graphic designs is strictly for cosmetic purposes and need not be used.

ii) You are allowed to define up to 32 fields totalling 255 characters. If you try to use over the maximum of either when you try to save the design you will be told that you have exceeded the maximums and you will have to re-edit the screen.

iii) To define the size of a field use braces "{}" (FCTN F and G) or brackets "[]" (FCTN R and T). When determining the size of the field, the braces or brackets are considered part of the field. Therefore, the smallest field possible is 2 characters. Brackets ([]) - used to ensure input to field

will be in Upper Case regardless of what was input. Braces ({}) - allows for the input of both Upper and Lower Case.

iv) Field names can be included on the screen display. These will not be part of the data but will appear whenever the screen is displayed. These are not considered part of the 255 characters you are limited to in your input.

```
eg. - LAST NAME:  [            ]
field name  ----field-------
```

v) When the screen creation is completed, it can be saved to diskette by pressing FCTN 6. A series of questions will be asked, then the information will be saved.

vi) At this point, a print out of the screen you have just created may be obtained. It is recommended that you print it out for future reference.

vii) Special key functions:

FCTN 1 - deletes character under cursor and shifts all characters one to the left
FCTN 2 - inserts a space character under the cursor and shifts all characters one to the right
FCTN 3 - erases the data screen display
FCTN 5 - tabs the cursor 5 spaces to the left
FCTN 7 - tabs the cursor 5 spaces to the right
FCTN E - moves cursor up one line - stays on same column
FCTN X - moves cursor down one line - stays on same column
ENTER - moves the cursor to the beginning of the next screen line
Note: FCTN 6 and FCTN 9 are described elsewhere.

(to be continued...)

FAST EXTENDED BASIC

LUCIE DORAIS

Before I start this month's column, I have to make a correction in last month's PROGRAM WRITER: in line 780, the X has to be replaced by an L; the whole line should be:

```
780 IF SEG$(A$,L,1)=C$ THEN L=L-1 :: A$=SEG$(A$,1,L) :: GOTO 780
```

I changed the listing so many times, that I hope only this line escaped my vigilance; if there are more bugs, please tell me.

Do you know that Barbie is 30 years old? To celebrate, I created this little program (also because I sewed a whole wardrobe for my niece's Barbie, and had a lot of fun matching the clothes, colors, etc.). No, the Mattel company is not sponsoring the program; the title was DOLL BOUTIQUE until I badly needed a blank character to use instead of the re-defined "/" and had to sacrifice the "D" (see line 550); the "@" is already used for the "Y": see line 130.

```
100 REM ** BARBIE BOUTIQUE ** L. Dorais / Ottawa U.G. / March 1989
110 REM
120 CALL CLEAR :: CALL SCREEN(12) :: CALL MAGNIFY(3) ::
    CALL CHARPAT(47,A$,58,B$,89,C$) :: CALL CHAR(64,C$)
130 DISPLAY AT(1,7):"BARBIE BOUTIQUE" :: DISPLAY AT(21,5):" OPENING THE
    STORE": :"   IF @OU USE THE JO@STICK     RELEASE THE ALPHA LOCK"
140 DIM DR(15),DC(15),SZ(20),US(4),UC(4)
150 DATA 0,00C6C600004438,0,0,0F1F3E78F0E0C0C0,C0C0C0C06060E0E0,
    E0F0F83C1E0E0606,0606060C0C0E0E0E
160 GOTO 480 :: AR,BC,C,CC,CR,K,N,NC,S,V,X :: CALL HCHAR :: CALL VCHAR ::
    CALL GCHAR :: CALL COLOR
170 CALL SPRITE :: CALL LOCATE :: CALL DELSPRITE :: CALL C :: CALL CC ::
    CALL CUR !@P-
180 DATA 071F3F7F7F7F7F7F,C0F0F8FCFCFCFCFC,7F7F7F3F1F0F0703,FCFCFCF8F0E0C080
190 DATA 0000000000010303,033F7FFFFFFFFFFF,80F8FCFEFEFFFFFF,0000000000008080
```

```
200 DATA 070F1F1E3C787020,FFFF7F3F3F3F3F3F,FFFFFDF8F8F8F8F8,C0E0F0F0783C1C08
210 DATA 1F1F1F3F3F3F7F7F,F0F0F0F8F8F8FCFC,7E7E7E7E7E7C7C7C,FCFCFCFCFC7C7C7C
220 DATA 7C7C7C7C7C787878,7C7C7C7C7C3C3C3C,7878787878787830,3C3C3C3C3C3C3C18
230 DATA 183C7EFFFF7E3C18,0,0,0
240 DATA 0000003F3F3F0000,0,000000F8F8F80000,0
250 DATA 1C1E1E1E1F1F1F3F,3F3F3F3F3F1F1F0F,1C3C3C3CFCFCFCFE,FEFEFEFEFEFCFCF8
260 DATA 0,0103030707020000,070F1F1F3F7F7FFF,FFEFC78707070703
270 DATA 0707078F8FDFFFFF,FFFFFFFFFFFFFFFE,0080C0C0E0F0F0F8,FCBE1E0F07020000
280 DATA 3C7FFFFFFFFFFF7F,7F3F0F0707070707,00078FFFFFFFFFFF,FFFFFFFFFFFFFFFF
290 DATA E0F0F8F8F8F8F8F0,F0E0800000000000,0,0
300 DATA 0000000307070F0F,1F1F1F3F3F3F7F7F,FFFFFFFFFFFFFFFF,FFFFFFFFFFFFFFFF
310 DATA 808080E0F0F0F8F8,FCFCFCFEFEFEFFFF,0,0
320 DATA 0F0F0F1F3F7F7F7F,7F7F7F7F7F7F7F7F,F8F8F8FCFEFFFFFF,FFFFFF7F7F7F7F7F
330 DATA 7F7F7F7F7F7E7E7E,7E7E7E7E7E000000,7F7F7F7F7F3F3F3F,3F3F3F3F3F000000
340 DATA 1F1F3F7FFFFFFFFF,FFFFFFFFFFFFFFFF,F0F0F8FCFEFEFEFE,FEFEFEFEFEFEFEFE
350 DATA 1F1F1F3F7FFFFFFF,FFFFFFFFFEFE0000,F0F0F0F8FCFEFEFE,FEFEFEFEFEFE0000
360 DATA 0103030101030341F,7FFFFFFFFF7F3F1F,80808038FC3800C0,FCFEFFFFFFFEFCF0
370 DATA 001F3F3F3F3F3F00,0,00FCFEFEFEFEFE00,0
380 DATA 8,16,8,17,9,16,9,17,10,15,10,16,10,17,10,18,11,15,11,16,11,17,11,18
390 DATA 12,16,12,17,13,16,13,17,14,16,14,17,15,16,15,17
400 DATA 15,26,16,26,15,27,16,27,5,26,6,26,5,27,6,27
410 DATA 8,23,9,23,8,24,9,24,8,25,9,25,8,26,9,26,8,28,9,28,8,29,9,29,8,30,9,
    30,8,31,9,31
420 DATA 10,3,11,3,10,4,11,4,10,5,11,5,10,6,11,6,14,3,15,3,14,4,15,4,16,3,
    17,3,16,4,17,4
430 DATA 10,7,11,7,10,8,11,8,14,7,15,7,14,8,15,8
440 DATA 4,4,5,4,4,5,5,5,5,7,6,7,5,8,6,8
450 DATA 86,121,1,74,120,1,74,110,2,74,126,2,73,118,2,73,134,2,89,116,2,
    89,132,2
460 DATA 89,120,2,105,120,2,89,121,1,89,121,1,45,120,1,52,120,1
470 ! ** display **
480 CALL CC(32,63,1) :: CALL D("  FINDING THE DOLL") :: CALL CC(88,143,1)
    :: CALL COLOR(2,10,5,3,10,5,4,10,5) :: X=8
490 CALL COLOR(X,13,1) :: X=X+1 :: IF X<15 THEN 490 ELSE X=13
500 CALL VCHAR(6,X,61,13) :: X=X+1 :: IF X<21 THEN 500
510 CALL CC(40,59,2) :: CALL SPRITE(#16,32,2,54,125) ::
    CALL SPRITE(#17,36,11,56,121) :: CALL D("CHOOSING THE CLOTHES") ::
    CALL CC(88,143,2)
520 CALL HCHAR(8,23,32) :: CALL VCHAR(10,6,32,2) :: CALL HCHAR(16,26,32,2)
    :: CALL D(" SORTING THE COLORS")
530 DISPLAY AT(7,2):"HATS";:: DISPLAY AT(19,1):"BOTTOMS" ::
    DISPLAY AT(11,23):"TOPS" :: DISPLAY AT(17,23):"BELT"
540 CALL C(6,11) :: CALL C(14,11) :: CALL C(6,22) :: CALL C(14,22)
550 V=8 :: CR=25 :: CC=121 :: US(1),US(2),US(3),US(4)=20 :: SZ(20)=1
    :: UC(1),UC(2),UC(3),UC(4)=13 :: BC=5 :: X=2
560 READ DR(X),DC(X),SZ(X) :: X=X+1 :: IF X<16 THEN 560 ELSE CALL
    SPRITE(#1,60,16,CR,CC) :: CALL CHAR(68,A$,71,B$)
570 DISPLAY AT(21,1):RPT$("XZ",14):"MOVE CURSORG ARROWSDJO@STICK":
    "ACTIONG ENTERDFIRE   QUITG Q":"  k NEW COLOR   < TAKE OFF"
580 ! ** game **
590 CALL KEY(3,K,S) :: IF S<>0 THEN 620
600 CALL JOYST(1,X,Y) :: IF ABS(X)+ABS(Y)=4 THEN K=X+2*Y ::
    K=9+(K=-4)-(K=-8)-2*(K=8) :: GOTO 630
610 CALL KEY(1,K,S) :: IF S=0 OR K<>18 THEN 590 ELSE 690
620 IF K=81 THEN CALL CLEAR :: END
630 IF K=11 OR K=69 THEN CALL CUR(CR,-V,17) :: GOTO 680
640 IF K=10 OR K=88 THEN CALL CUR(CR,V,137) :: GOTO 680
650 IF K=8 OR K=83 THEN CALL CUR(CC,-V,9) :: GOTO 680
660 IF K=9 OR K=68 THEN CALL CUR(CC,V,233) :: GOTO 680
670 IF K=13 THEN 690 ELSE 590
680 CALL LOCATE(#1,CR,CC) :: GOTO 590
690 AR=1-(CC+4>128)-(CR+4>72)-(CR+4>72) :: CALL GCHAR((CR+4)/8,(CC+4)/8,C)
700 IF C=107 AND CC<180 THEN 770 ELSE IF C=60 THEN GOSUB 760 :: GOTO 590
    ELSE IF C=61 THEN 800
710 IF C<88 THEN 590 ELSE N=INT(C/4)-20 :: IF INT(N/2)<>N/2 AND SZ(N)=2
    THEN N=N-1 :: C=C-4
720 GOSUB 760 :: CALL SPRITE(#N,C,UC(AR),DR(N),DC(N))
730 IF SZ(N)=2 THEN CALL SPRITE(#N+1,C+4,UC(AR),DR(N+1),DC(N+1))
740 US(AR)=N :: GOTO 590
750 ! ** subroutines **
760 CALL DELSPRITE(#US(AR)) :: IF SZ(US(AR))=2 THEN CALL
    DELSPRITE(#US(AR)+1) :: RETURN ELSE RETURN
```

```
770 NC=UC(AR)+1 :: IF NC=17 THEN NC=2 ELSE IF NC=10 THEN NC=11
780 CALL COLOR(#US(AR),NC) :: IF SZ(US(AR))=2 THEN CALL COLOR(#US(AR)+1,NC)
790 UC(AR)=NC :: GOTO 590
800 BC=BC+1 :: IF BC=12 OR BC=10 THEN BC=BC+1 ELSE IF BC=17 THEN BC=2
810 CALL COLOR(2,10,BC,3,10,BC,4,10,BC) :: GOTO 590
820 !@P+ ** user-def subs **
830 SUB CC(X,Y,Z)
840 IF Z=1 THEN READ A$ :: CALL CHAR(X,A$)ELSE READ A,B :: CALL HCHAR(A,B,X)
850 X=X+1 :: IF X<Y+1 THEN 840
860 SUBEND
870 SUB C(R,C) :: CALL HCHAR(R,C,107) :: CALL HCHAR(R+2+4*(R=6),C,60)
    :: SUBEND
880 SUB D(A$) :: DISPLAY AT(21,5):A$ :: SUBEND
890 SUB CUR(X,Y,Z) :: X=X+Y :: IF X=Z THEN X=Z-Y
900 SUBEND
```

Playing is easy: the doll is at the centre of the screen, on a blue backdrop; around her is the "boutique", divided into four departments: two HATS, three TOPS, four BOTTOMS and one BELT. To dress the doll, move the cursor with the arrow keys or joystick # 1; you can use both at any time, no need to decide when you start the program. When the cursor is positioned over the garment you want, just press ENTER or the FIRE BUTTON, and it will immediately appear on the doll. To change the doll's backdrop color, just position the cursor over it, then press ENTER/FIRE.

There are two special characters into each department: a green square to change the color of the garment on the doll (since they are sprites, their color is independent from the character colors of the clothes hanging in the store), and a pink button to take the garment off; both work with the ENTER key or the FIRE BUTTON. The only "real" key that needs to be pressed is the "Q" for Quit, which makes this program very easy to use for small children.

You might want to use PGMWRITER, published last month, to type this program... lots of DATA, which redefines almost all the characters. Careful planning had to be done, because I use sprites to display them on the doll, and single characters to display them in the boutique. The sprites are magnified with a factor of three, which means that each sprite is made up of four characters; that is why some characters are defined as empty: they will be part of the sprite, but are not needed to define the actual garment.

To help you (and myself) understand and debug the program, I prepared the following chart. All the clothes use one or two sprites, whose number is also used by the arrays DR and DC (dot-row and dot-column for display on the doll) and SZ (garment size; array is 20 because I needed a dummy value to start the program: I use a dummy sprite #20, that never appears on the screen). The "line" columns below refer to the program line where you can find the relevant character definitions and locations on screen (the "boutique"), and the sprite positions on the doll when she gets dressed. She herself (Body) is made up of characters, but her face and hair are sprites, displayed in front of her head.

SPRITE #	USED FOR	CHAR DEF. char #	CHAR DEF. line	CALL HCHAR line	CALL HCHAR rows	CALL HCHAR cols.	DISPLAY SPRITE line	DISPLAY SPRITE drow	DISPLAY SPRITE dcol
1	Cursor	60- 63	230	--	---	---	560	25	121
2	Belt	88- 91	240	400	15-16	26-27	450	86	121
3	Sleeveless top	92- 95	250	400	5- 6	26-27	"	74	120
4	Sweater	96- 99	260	410	8- 9	23-24	"	74	110
5	"	100-103	270	"	8- 9	25-26	"	74	126
6	Blouse	104-107	280	410	8- 9	28-29	"	73	118
7	"	108-111	290	"	8- 9	30-31	"	73	134
8	Wide Skirt	112-115	300	420	10-11	3- 4	"	89	116
9	"	116-119	310	"	10-11	5- 6	"	89	132
10	Pants	120-123	320	420	14-15	3- 4	460	89	120
11	"	124-127	330	"	16-17	3- 4	"	105	120
12	Straight Skirt	128-131	340	430	10-11	7- 8	"	89	121
13	Short	132-135	350	430	14-15	7- 8	"	89	121
14	Flower Hat	136-139	360	440	4- 5	4- 5	"	45	120
15	Pillbox Hat	140-143	370	440	5- 6	7- 8	"	52	120
16	Face	32- 35	150	--	---	---	510	54	125
17	Hair	36- 39	150	--	---	---	510	56	121
--	Body	40- 59	180-220	380-390	8-15	15-18	--	--	--

The display of the screen starts on line 480; it takes a while to read all the DATA with the SUB CC, so some messages appear on the screen (thanks to the SUB D) to keep your mind busy. The CC sub is used both to CALL CHARs (if the third parameter passed is a 1, as in line 480) and to CALL HCHARs (parameter is 2, as in line 510: first the body, then, after the sprites for face and yellow hair, the clothes); first and second parameters are the starting and ending characters. I made an important discovery: a user-defined sub can READ DATA anywhere in the program!

This program uses no FOR-NEXT loops, since they tend to slow the program flow -- thus the many "X=" found in the screen display portion of the program. For example, line 500 draws a blue rectangle as a backdrop for the doll; instead of FOR X=13 TO 20 :: CALL VCHAR(6,X,61,13) :: NEXT X, we define X=13 in the preceding line, then do the VCHAR, then increment X; if lower than 21, redo the line; when X reaches 21, go to the next line.

Line 520 puts some space characters on the screen to replace those empty characters that are part of some clothes in the boutique, because when we pick a garment, we don't want the program to react to an empty character. Line 540 puts the special characters (for NEW COLOR and TAKE OFF) in each of the four departments of the store. Line 550 defines the cursor "Velocity" as 8 pixels, i.e. jump one character at a time (if you modify it to go faster or slower, you MUST also modify the border checks for the cursor movements in lines 630-660, third parameter); CR and CC are the starting positions of the cursor; the arrays US (Used Sprite) and UC (Used Color) keep in memory the number of the sprite displayed on the doll (only one from each department) and its color, to be used when we erase the garment with a CALL DELSPRITE (to change it or to take it off, sub in line 760), or when we change its color (sub in lines 770-790). The Used Sprite is initialized to 20, a dummy sprite, with a size of 1, and the starting color is dark green (13), same as the clothes in the store (you can change it to your taste, since it affects only the sprites' color, not the actual character sets). Background color is initialized as dark blue (5).

The program finally reads the DATA for the display-row/column and size of each garment (sprites #2 to 16) in line 560, a trick that allows for fast "dressing" of the doll, before displaying the cursor and writing some instructions at the bottom: please type as is, remember the "G", "D" and "@" are redefined as ":", "/" and "Y". The repetition of "XZ" draws a nice line of... belts.

The main part of the program is **game**. It starts by a CALL KEY, with keyboard 3 to get action from the arrow keys with alpha-lock up or down, and with or without pressing the FCTN key. If a key is pressed from the keyboard, go to line 620 to read it; if none, check the joystick! The program will only react if the joystick is moved in one of the four main directions, which will always give us an absolute total of 4 when the two parameters are read (see the XB manual; sorry, programming it for eight directions is a bit more complicated, since we also use the keyboard; it could be done with some thinking... I leave it to you!). If the joystick was moved up, down, right or left, we get the equivalent arrow key values (8-11) with some brainteasing calculations (try it, remembering that the value of the expression inside the parentheses is -1 if true and 0 if false), then send the program to move the cursor in lines 630-660. But the JOYST function does not check for the fire button: this is done is line 610, where keyboard #1 refers to joystick #1; this line will react only if the key pressed is 18, which is the fire button (it is also "Q" on the split keyboard, but don't press it instead of FIRE: you will Quit the program!) and will go to line 690, which also deals with the ENTER key checked in line 670. If no key was pressed, or the joystick was not used, the program returns to the CALL KEY in line 590.

Lines 620-670 move the cursor according to the key values (8-11 if you also press FCTN, or ASCII letters ESDX if not); the four moves, accomplished with the SUB CUR which also checks for the screen limits, all go to a CALL LOCATE for the cursor in line 680, while if you press ENTER (K=13) you go to line 690; if you press any other key, back to the main CALL KEY. Key "Q" (Quit) was dealt with in line 620.

When you press ENTER, the program has to find out the character under the cursor and react accordingly: this is done by lines 690-740. We first find the store department the cursor is in, or the AREa, derived from its dot-row/column; again, a relational expression gives us the answer (the areas are: HATS=1, TOPS=2, BOTTOMS=3, BELT=4); that value will be used by the US and UC arrays to hold the new sprite number N (see line 740) and new sprite color

NC (lines 770-790). Once the general area has been defined, we can CALL GCHAR under the cursor; the character C is located under the centre of the cursor, thus the addition of four pixels to the cursor positions: they refer to the upper left pixel of the cursor sprite.

Character 107 is the green square used to change the color of the garments; we also check the cursor column, because this character is also part of the blouse with puffy sleeves, displayed beyond dot-column 180 (I faced a character shortage, which also explains why the doll has only one belt, two hats, and no shoes...). Character 60, the "take off" button, is the same character as the cursor, but is colored like the body because it is part of one of the body char. sets; if you change the backdrop color, the buttons will change too! Character 61 is the backdrop, an empty character from the same body set. If the character under the cursor is part of a garment, it will have a number above 87; if GCHAR gave us a character below that value, we are over a space, or the doll's body, or over a department name, so we go back to the CALL KEY in line 590.

But if you press the ENTER/FIRE BUTTON over a garment, the program needs to know the starting character number of the first (or only) sprite to display it, and the corresponding sprite/array number: this is all done in line 710 by finding the value N, which is the sprite number. We divide the character by 4, then subtract 20 to get a value between 2 and 15: look at the chart above to see how each sprite was planned; remember that lower numbered sprites appear in front of higher numbered ones, therefore the belt is sprite #2, and the cursor, which has to pass in front of everything, is #1. If the garment is made up of two sprites (as told by the value in array SZ), the character picked by GCHAR might be part of the second sprite, i.e. the odd-numbered one, so we get the right value for both character and sprite with subtractions. What if the character picked is not a multiple of four, i.e. not a sprite root?? No problem: under CALL MAGNIFY(3), CALL SPRITE will always start with the "next smallest [character] number that is evenly divisible by four" (XB manual).

Line 720 first takes off the garment displayed on the doll (GOSUB), then displays the new one in the same UC color as the previous one, at the dot-row/column specified by the DR/DC arrays for that sprite; since a sprite containing a blouse with puffy sleeves covers more body area than a sleeveless one, the precise location for each piece of garment had to be determined by the programmer at some point... If the garment is made up of two sprites, line 730 will display the second one, in the same color of course... Position of the second sprite also needs to be kept in an array, since the second sprite can be at the right of the first one (wide skirt, some tops) or below it (pants).

A last note on changing the colors of the garments or the backdrop (lines 770-810): both subroutines simply cycle through the TI colors, skipping color X, transparent, and pale red (10), used for the body; in addition, the backdrop color will never be pale yellow (12), the screen color.

TI BASIC continued from March
by **Steven Shaw**

GLOSSARY

ARRAY A collection of variables referenced by a subscripted number.

ASCII 'American Standard Code for Information Interchange' - standard code numbers for the characters used by the computer.

BASIC 'Beginners All-purpose Symbolic Instruction Code'. An easy to use and widely used type of programming language.

BAUD 'Bits per second'- refers to the speed at which data is transferred to and from the computer.

BINARY Our normal numbering system is DIGITAL, and uses numbers 0 to 9. A BINARY system uses only numbers 0 and 1 ,or OFF and ON. The computer works internally with signals which are OFF or ON.

BUG An error in a program, which causes incorrect or unwanted operation.

BYTE A group of 8 binary numbers (called BITS) used in computing to represent a character or command. Also used as a means of measuring a computers memory capacity.

CONSTANT Used to describe a number or STRING, and distinct from a VARIABLE.

CURSOR A flashing character used by the computer to indicate that it is waiting for an input.

DISK A mass storage device used to store programs and data files.

FILE A collection of data records.

HEXADECIMAL A numbering system with base 16. Instead of using number 0 to 9, it uses numbers 0 to 15, but letters A to F are used instead of 10 to 15.

LOOP A program line, or lines, which are performed a specified number of times.

PROGRAM A series of commands which tell the computer what to do.

RAM 'Random Access Memory' Temporary storage in the computer, used for your programs. The contents are not retained when the console is switched off.

RECORD A collection of data elements. A group of records form a FILE.

RESERVED WORD A word used by the computer as a command or function. Such words cannot be used as variable names.

ROM Read Only Memory. Permanent memory which retains its contents when the console is switched off. Contains the operating system of the computer.

RUN An instruction to the computer to execute a program in its memory.

SCROLL Movement of the screen display by one line upwards.

SOFTWARE A name given to computer PROGRAMS.

STRING A series of letters, numbers or symbols, treated as a single unit. A single number may be treated as a number OR as a string but cannot be both at once. 2 is a number. "2" is a string.

VARIABLE A name or label which has a value which may be altered during a program.

HOTLINE NUMBERS

The executive has expressed a desire to assist all members should you have some problems or questions, want to do some library swapping or borrow a book. This will be the place to look. Listed here are the members of the executive, committee heads, and others in the group willing to help in their specialized areas. Of course, if you wish to be placed on the list, just give me a call. I know there is a lot of expertise within our Group, so I hope to add to this list. Please respect normal hours unless you specifically know that someone doesn't mind a call at 3am, or use the BBS to leave a message at 738-0617, 24 hours a day, 7 days a week.

JANE LAFLAMME.....PRESIDENT.....(H) 837-1719 or (W) 745-2225

AL PALMER.....VICE PRESIDENT.....594-9216

MARCELLE GIBSON.....SECRETARY.....233-2384

BILL SPONCHIA.....TREASURER.....523-0878

MICHAEL TAYLOR.....PAST PRESIDENT.....831-0143

PETER ARPIN.....SYSOP.....523-0017

RUTH O'NEILL.....NEWSLETTER EDITOR.....234-8050

TONY HOPKINS.....ADVERTISING.....746-4463

DAVE MORRISON.....LIBRARY CHAIRMAN.....737-4889

JACK McALLISTER.....CASSETTE LIBRARY.....225-6989

HENRI MONAT.....ARCHIVES.....824-0941

LUCIE DORAIS.....MEMBERSHIPS.....232-0393

BOB BOONE.....HARDWARE/SOFTWARE.....(705) 476-9391

ART GREEN.....ASSEMBLY HELP.....837-1955

DICK PICHÉ.....TECH.....521-8667

CLUB BBS.....SET MODEM TO 8N1.....738-0617

COME TO THE TI-FEST!

Don't miss the Ottawa TI event of the year!!!! Plan to come yourself, and make sure all your TI-friends know about it, so they can come, too! Mark April 29th, 1989 on your calendar today!

Contact: Jane Laflamme
5480 Canotek Road
Unit 16
Gloucester, Ontario
Canada
K1J 9H6

Phone: (613) 837-1719 (home)
(613) 745-2225 (work)
(613) 744-4784 (fax)

For out-of-town visitors, the official hotel is the Talisman Motor Inn. If you make reservations, please be sure to mention that you with the TI-FEST.

FROM

P.O. BOX 2144,STATION D,OTTAWA
*** ONTARIO,CANADA K1P 5W3 ***