mini'app'les

apple computer user group newsletter

VOL IV No 7 JULY 1981

CALENDAR CALENDAR CALENDAR

WHICH	WHEN	WHERE	WHAT
PASCAL Note 1	Wed July 1 7:30pm	Minnesota Federal 9th Ave S Hopkins	Regular PASCAL Special Interest Group Meeting.
Nibble "Subscribers" Note 4	Wed July 8 7:30pm	Home of Ron Androff 1725 Crest Ridge Lane Eagan	Status Meeting - Work Assignments
REGULAR MINI'APP'LES Note 3	WEDNESDAY JULY 15th 7:30pm	UNIVERSITY MINNESOTA ST. PAUL CAMPUS Near State Fair Room 1345 Bldg 412 (See map inside)	Bill DeCoursey demonstrating the Video Digitizer DG-65 which converts TV images to HIRES.
Fort Snelling Note 2	Mon July 20	Nakomis Community Ctr Minnehaha Parkway	Programming Special Interest Group Meeting.
PASCAL Note 1	Wed Aug 5 7:30pm	Minnesota Federal 9th Ave S Hopkins	Regular PASCAL Special Interest Group Meeting.
Fort Snelling Note 2	Mon Aug 10	Nakomis Community Ctr Minnehaha Parkway	Programming Special Interest Group Meeting.
Nibble "Subscribers" Note 4	Wed Aug 12		
REGULAR MINI'APP'LES Note 3	WEDNESDAY AUG 19th 7:30pm	HENNEPIN SOUTHDALE LIBRARY 7001 York Edina.	
Genealogy	Oct 10th	Minn. Historical Soc	Genealogy Conference.

Note 1. Contact- Keith Madonna
2. Dave Nordvall
3. Chuck Thiesfeld
4. John Schoeppner

MINI'APP'LES INFORMATION

MINI'APP'LES OFFICERS

President	Stephen K.Johnson 6053 Wentworth Ave S. Minneapolis, Minnesota, 55419	869-3447
Past President and Newsletter Editor	Daniel B.Buchler 13516 Grand Avenue S. Burnsville, Minnesota, 55337	890-5051
Vice President	Chuck Thiesfeld 8416 Xerxes, Bloomington, Minnesota, 55431	831-0009 830-5020
Treasurer	Marilyn Thomas 2735 Irving, Minneapolis, Minnesota, 55408	872-7669
Secretary	Ron Androff 1725 Crest Ridge Lane, Eagan, Minnesota, 55122	452-5230

MEMBERS OF THE BOARD

Membership Co-ordinator	Ann Bell 8325 39th Avenue N. New Hope, Minnesota, 55427	544-4505
Newsletter Editor	Dan Buchler	890-5051
Bibliographer	Chuck Boody	933-5290
Librarian	Terry Pinotti	786-7118
MECC Librarian	Dave Nordvall	724-9174
Program Editor	Ken Slingsby	507/263-3715
Software Distr. - Mail and Software Sales	Hugh Kurtzman	544-7303
Hardware Sales	Al Peterman	721-3295
Disk Sales	Peter Gilles	475-3916
Publicity Co'tor	A.Michael Young	884-2841
Education Co'tor	Chase Allen	435-6245
Spcl Interest - Pascal	Keith Madonna	474-3876
Spcl Int. - Geneology	Bill Decoursey	574-9062
Spcl Int. - Nibble	John Schoeppner	455-8613
Spcl Int. - Z80/CPM & Meeting Hdw Support	Rick Gates	735-0373
Technical Advisers	Dave Laden	489-8321
	Jim White	636-4865
Assistant Librarians:	Bill Decoursey	see above
Assistant Prog Editors:	Tom Edwards	927-6790
	Rick Gates	see above

This is the Newsletter of Mini'app'les, the Apple II Users' group of the Twin Cities of Minneapolis and St. Paul.

Questions

Please direct questions to appropriate board member or any officer. Technical questions should be directed to one of the Technical Advisers listed here.

Membership

Applications for membership should be directed to the Membership Co-ordinator.

Dues are $10/year thru July; $5/year in July/Aug/Sept. After Oct 1st, $10 buys membership for current and next year. Members receive a subscription to this newsletter and all club benefits.

DOMs

DOMs (Diskettes of the Month) are available at meetings or contact Software Sales coord'r.

Newsletter Contributions

Please send contributions to the Newsletter Editor. Hard copy binary or text files (ASCII coded) are prefered, but any form will be gratefully accepted. Deadline for publication is the Wednesday before the 1st Wednesday of the month in which the item might be included. An article will be printed when space permits if, in the opinion of the Newsletter Editor, it constitutes suitable material for publication.

Advertising rates

Rates are as follows:

Full Page	$30/issue
Half Page	$20/issue

Circulation 450 (approx)

EDITORIAL

Summer is here! Whats that got to do with this newsletter? Plenty! People are busy at this time of the year. Vacation, yard work, outdoor activities of all kinds make it hard to get to one's Apple. Our staunch regulars (David Laden and Ron Androff) have contributed as usual, plus Jim White and the MECC information from Dave Nordvall, but thats it. Your editor has been away most weekends, so the promised article on Word Processing has'nt materialized. Besides, I've spent all my spare Apple time working on a new Intelligent Terminal Program for use with the Honeywell Multics Timesharing system.

Because of the above, this seemed a good opportunity to bring you a selection of material from another Apple II user group. This time it is the Dallas Apple Users Group. We featured the Chicago Northwest Suburban User Group earlier this winter.

This editor beleives that a Newsletter should be as original as possible. But as a user group, we owe it to our members to distribute as much information as possible. There is no practical way of giving everyone a copy of every user group newsletter that we receive. Hence, we extract the best material from another newsletter and reproduce it for you in this newsletter.

TURNING THE PAGES continued from page 18

RECREATIONAL COMPUTING -- MAY/JUNE 1981

Sesame Place Learning, Playing and Using Computers by Tony Bove and Cheryl Rhodes. Pages 8-15 and 22-28. Sesame Place uses 55 Apple computers in the computer area of the park. The article also includes an interview with Joyce Hakansson and Dennis Sullivan, the computer games coordinator and director of computer programming respectively.

Games: Apple Fun by Louis K. Bell. Page 38. Two short Applesoft programs called "Sketch Pad" and "Sum of the Digits."

Computer Anatomy for Beginners: Taking the First Step by Mike Gabrielson and Marlin Ouverson. Page 44. This is an introduction to a few computer buzz words.

DOM #5

Details of the contents of DOM#5 were published in the last edition of Mini'app'les. This disk will be for sale at the next meeting (July). Due to rules of the Library, the disk was not sold at the June meeting in the Hennepin Southdale Library, and, for the same reasons, will not be for sale at the August meeting. You may also purchase the disk by mail. See page 2.

RENUMBER

by Jim White
from the Apple Answer Book

Renumber is a very powerful tool for developing programs. However, if you use it, you may find some strange alterations in your program. RENUMBER may have done it. What happens is that the number after "*" sometimes is mistaken as a line number and RENUMBER renumbers it! So, if you had a line:

```
10 LET A = B * 10
```

it might renumber as:

```
20 LET A = B * 20
```

The fix is:

For RAM Applesoft	For ROM Applesoft
]LOAD RENUMBER	]LOAD RENUMBER
]POKE 14342,172	]POKE 4815,172
]POKE 14343,171	]POKE 4816,171
]SAVE RENUMBER	]SAVE RENUMBER

BULK PAPER

by Dan Buchler

The best price, we know of, for regular 9 1/2 by 11 inch single part tractor feed 20lb paper is at

Custom Business Forms
Hennepin Avenue
Minneapolis.

They make the paper and sell it at wholesale prices (less than $15 for a box of 2000 sheets). You can buy it in lots of one! Meanwhile, if anybody who lives in BUrnsville wants some, I have a few extra boxes.

BOARD MEETING JUNE

Meeting was called to order at 7:45. Our treasurer reported our financial status:

CASH ON HAND: approx $2000
ASSETS: 350 disks
LIABILITIES: approx $50.

We discussed meeting places and reafirmed our June meeting site and that July's meeting would be held at the U-of-M St. Paul campus. We also concluded that we should continue the search for meeting places. Requirements: Seat at least 250 people, be available until at least 10 o'clock, have a source of electricity, satellite rooms if possible, and have a low cost.

Zim Computer had a greivance concerning our printer sale. Seems they sold a printer to one of our members, delivered the unit and were paid; only to have the member return the printer demanding his money back so he could purchase it through the club. A most unfortunate situation. The consensus seemed to be that we would try to include the local dealers in all of our special purchases if possible but that it was in our charter to get the best deal for our members too.

Motion - That we make DOMs available to Personal Business Systems for resale to Min'app'les members on a trial basis. Passed.

Resolution - That all computers at the meeting will be turned off during the business meeting and program except for club business, and that said computers be made available to club members for club business on a timely basis, and finally that copying of copywrited software is prohibited.

Motion - That Hugh Kurtzman and Dave Nordvall are authorized to purchase discs needed for MECC and DOM from Peter Gilles after existing supplies are exhausted. Passed.

The Association of Twin Cities Personal Computer Clubs are planning a computer faire for the fall time frame. Their next meeting is Wednesday June 24.

MEETING MINUTES JUNE

Meeting was called to order at 7:37 by our president.
No old business.
No new business.

Our next meeting wil be at the St. Paul Campus at the classroom office building. Terry Pinotti didn't make it to the meeting, unfortunely, so detailed directions are yet forthcoming.

Ms. Kathryn Thomas announced the Compucamp - a computer camp to be held at Bethel College in St. Paul for children from 10 to 16 years of age. There will be three sessions:-

August 2-7
August 9-14
August 16-21.

The camp features Apple computers and CDCs Plato system.

Al Peterman announced an Epson MX-80 printer purchase, probably the last for the summer. Order cut-off Tuesday June 23rd. He also requested anyone interested in modems or floppy drives at about $200-220 and $300-350 each respectivly to contact him. He wants to see if there is enough demand to warrant a bulk purchase.

Meeting was adjourned at 8:00.

John Riskin put on an excellent program on a subject dear to my heart - word processing. And as a bonus our Max Coe showed us very graphically the speed difference between assembly language and basic. Many of us are aware basic is slow, but I was amazed by the demonstration. How slow it is and how much versatility and power we give up for programming ease. Thanks to both gentlemen for an enjoyable and informative evening.

MX-80 NEWS

We're still waiting for the opportunity to order the Dot Graphics option. People rpresenting the Epson Factory either give no information or say wait a couple of weeks.

As stated before, if you want to be on the list for the first bulk order of the Graphics Option, Call Dan Buchler at 890-5051. Ask to be placed on the MX-80 Graphics List!

EXTRACTS FROM THE APPLE-GRAM NEWSLETTER OF THE DALLAS APPLE USERS GROUP

The articles on the following pages are taken from the Apple-Gram, the monthly newsletter of the Dallas, Texas, Apple Users Group.

These articles are only a small sample of an excellent publication. We have many programs in our library that originated in the Dallas Group.

The criteria for selection was:
Quality
Stand-alone
Not too long The Apple-Gram has published many excellent articles which accompany large program listings. We thought it better to leave the larger programs out. They take up alot of space and they can be obtained from our user bank or a disk-of-the-month.

Incidentily for those of you not familiar with the group, Bob Sander-Cedarlof is the author of the "SC" Assembler.

We thank the Dallas Apple Users Group for the opportunity to publish these articles. We also thank the Northwest Suburban Users Group for the article by Tim Hartley, republised in the Apple-Gram.

The table of contents for these articles is included with the table of contents for Mini'app'les.

Speedier GO TO.....Bob Sander-Cederlof

The Applesoft manual, on page 120, says: "During program execution, when Applesoft encounters a new line reference such as 'GO TO 1000' it scans the entire user program starting at the lowest line until it finds the referenced line." The conclusion is that you should place frequently referenced lines as early in your program as possible. That is why you frequently place all your frequently referenced subroutines down in the low line numbers.

However, Applesoft is not really quite that dumb. Before starting its search at the beginning of your program, it first checks to see if the referenced line is greater in value than the current line. If so, it starts right after the current line! That is the good news.

The bad news is that the test is made in an incomplete way. Only the high-order bytes of the two line numbers are compared. This means that if you are on line 1022 and you GO TO 1023, the search will start back at line 0. But if you GO TO 1024, the search will start immediately after line 1022. Put another way, if INT(current line number/256) is less than INT(referenced line number/256), then search from the next line forward. Otherwise, go back to the beginning.

MID$ vs LEFT$ & RIGHT$ Mike Firth

I always use MID$ and never use LEFT$ and RIGHT$. The reason is that MID$ will tolerate 0 length strings while LEFT$ & RIGHT$ will not. If I want to remove trailing blanks (which can be typed in or created by MID$, see below), I might write this one line routine : 100 Q=LEN(X$) : IF Q>0 THEN IF MID$(X$,Q, 1) = " " THEN X$= MID$(X$,1,Q-1): GOTO 100

If X$ starts off as all blanks, then it ends as a null string. If Q=0 then this command X$ = LEFT$(X$,Q) will produce an error and a null string cannot be created.

Another reason for using MID$ is that it is tolerant of length problems the other way. If X$="MIKE" and you try Y$=LEFT$(X$,8) you will get an error. If you use MID$ in the same situaton, you will get a four letter string. Just for the fun of it, note that the use of MID$(X$,6,4) will result in a null string.

TWO WARNINGS

If MID$ is used to take a portion out of the middle of a string, and that portion ends in blanks, the blanks will remain. (Which is how I got involved in this problem: I broke a long string down into a an array and ended up with a lot of blanks stored on the disk.) For example, MID$("JOHN SMITH 827-7734", 1,15) will result in fifteen character string that has five blanks at the end.

You may have wondered why, in the sample line 100 above, I didn't say IF Q>0 AND MID$(Q$,Q,1)=" " THEN The problem here is that while MID$ will tolerate a 0 as the second number it will not as the first. And while in some BASICs which allow only one AND test, the testing stops if the first item is false, in AppleSoft apparently all of the expression is evaluated before a judgement is made.

In the example, if Q is 0 (a null string), the error ILLEGAL QUANTITY is given when AND is used. When the form IF Q>0THEN IF MID$.... is used, the second test, while equivalent to AND, is not done until the first one is true. The memory cost is one additional byte.

Character Codes Again.....Bob Sander-Cederlof

Here is a one line Applesoft program that will print the neatest little chart of the complete Apple screen codes you ever witnessed!

```
100 HOME : PRINT "   0 1 2 3 4 5
 6 7 8 9 A B C D E F": PRINT
:CH = 0: FOR I = 0 TO 15: PRINT
" ";:B = PEEK (40) + PEEK
(41) * 256 + 1: POKE B,I + 176
 + 7 * (I > 9): FOR X = 2 TO
32 STEP 2: POKE B + X,CH:CH =
CH + 1: NEXT X: PRINT : NEXT I
```

It is on the August Disk of the Month, but if you didn't get one of those priceless volumes, go ahead and type it in! Then try explaining how it all works!

I found another neat one in the NSAUG Harvest, August, 1980, on page 4. (NSAUG is the Northwest Suburban Apple Users Group, in Illinois.) It is in an article by Mark Pump, a name you will likely see a lot of in time to come. Get into the monitor, where the prompt character is "*", and type the following:

```
* N C000 34:0 (cr)
```

That is, space, letter N, "C000", space, "34:0", space, RETURN. If you did it correctly, you will see a continuously scrolling screen full of "C000- 0D" lines. Now type any character on the keyboard, and you will see the hexadecimal appear where the "0D" was! Try it! You'll LOVE it!

Apple II File Processing .....Dr. Robert F. Zant

(The following material is condensed from the viewfoils Dr. Zant presented at the September Apple Corps meeting.)

Sequential files

A record is the amount of data read by a single INPUT command. Records are terminated by a carriage return. The symbol "▼" represents a carriage return in these examples:

```
THESE ARE EXAMPLES OF RECORDS.▼
SOMETIMES RECORDS ARE COMPOSED OF A SEQUENCE OF VALUES.▼
SUBDIVISIONS OF A RECORD THAT EACH CONTAIN▼
A SINGLE VALUE ARE CALLED FIELDS.▼
IN APPLESOFT, THE FIELDS ARE USUALLY SEPARATED BY A COMMA.▼
THESE ARE THE FIRST, 2ND, AND 3RD FIELDS IN THIS RECORD.▼
THE FOLLOWING RECORD HAS FOUR FIELDS.▼
YES,47.5,A4,23▼
```

Notice that a record may contain a mixture of numeric and string data. A comma can be used to separtae fields in Integer BASIC only when the values are numeric.

An alternative to using commas is to force the corresponding fields in each record to contain the same number of characters. The record can then be read as a single string value, and the subdivided using the MID$ function.

Now that you know what a record looks like, let's look at some examples.

```
100 INPUT A$
200 INPUT NUM, AMT
300 INPUT NUM, AMT, DEX$
```

Records to match these INPUT statements might look like this if entered from the keyboard:

```
NO▼
1024,347,54▼
987,1456.81,SUPPLIES▼
```

If the same records were to be read from disk, the exact same data including the commas and carriage returns would have to be on the disk.

Problem: How do we get the data onto the disk in the correct format?

Answer: The PRINT statement is used with the commas explicitly stated.

```
100 PRINT A$;",";AMT;",";DES$
```

```
90 C$=","
100 PRINT A$;C$;AMT;C$;DES$
```

Problem: PRINT statements normally display data on the screen. How do we write data onto the disk?

Answer: Before executing a PRINT statement we redirect output to the disk by executing a DOS WRITE command.

Example:
```
80  PRINT CHR$(4);"WRITE FILE1"
90  C$=","
100 PRINT A$;C$;AMT;C$;DES$
110 PRINT CHR$(4)
```

Statement 110 redirects output back to the screen.

Now that we know how to write data onto a disk, how is it read off of the disk? Simple, use the INPUT statement.

Problem: INPUT statements normally read from the keyboard. How do we READ from a disk?

Answer: Before executing an INPUT statement we redirect input from the keyboard to the disk by executing a DOS READ command.

Example:
```
90  PRINT CHR$(4);"READ DISK2"
100 INPUT NUM, AMT, DES$
110 PRINT CHR$(4)
```

Statement 110 redirects input back to the keyboard. (Any printed record beginning with Ascii 4 cancels a DOS READ or WRITE command.)

Hey, this is interesting! But I have noticed a problem. We have used the DOS commands "READ DISK2" and "WRITE FILE1". What are "DISK2" and "FILE1"?

Well, they are the names of TEXT files on the disk. These names would appear on a catalog with a "T" in front of them to denote them as text files.

Problem: How does DOS know what slot/drive/volume contains the text file that is to be read or written?

Answer: Before we read or write on a disk file we must inform DOS as to its location by executing a DOS OPEN command.

Example:
```
100 PRINT CHR$(4)"OPEN FILE1,S6,D1,V0"

200 PRINT CHR$(4)"OPEN DISK2,S4,D2,V10"
```

The DOS OPEN command locates the specified file and prepares it for reading or writing beginning at the first record on the file. The DOS APPEND command is used instead of OPEN when you wish to start writing at the <u>end</u> of a file.

Example:
```
100 PRINT CHR$(4)"APPEND CHECKS,S6,D1"
```

You may also change your position in a file without reading or writing. The DOS POSITION command moves your place in a file <u>forward</u> over the number of records (carriage returns) you specify by the "R" parameter.

Example:
```
200 PRINT CHR$(4)"POSITION BOOKS,R3"
```
will skip over the next 3 records in the file named "BOOKS".

When we are finished reading and writing a file, we cannot just end the program. We must first terminate file processing with the DOS CLOSE command. If a file is named in the CLOSE command, only that file is affected. If no file is named, then all files are closed.

Examples:
```
100 PRINT CHR$(4)"CLOSE BOOKS"
200 PRINT CHR$(4)"CLOSE"
```

Warning: Terminating a program without closing all files can result in the loss of data from a file.

The DOS EXEC command is a special command. It is not used in the regular reading and writing of data files. Rather the EXEC command causes a file to be opened and read immediately (no INPUT command required). The records on the file are immediately executed just as if they were statements or commands entered through the keyboard. This allows you to store a commonly executed sequence of commands, or a commonly used sequence of statements on a text file, and call them up by merely typing "EXEC" and the file name.

When the EXEC command is issued from within an executing BASIC program, the EXEC file is not processed until after the BASIC program terminates.

Example: The file "MESSAGE" contains the following records:
```
IF N=1 THEN PRINT "ARRAY LIMIT EXCEEDED"▼
IF N=2 THEN PRINT "ILLEGAL VALUE ON FILE"▼
IF N=3 THEN PRINT "INVALID PASSWORD"▼
```

Then the Nth error message can be printed by the following statement (note that the value of N has already been set):

```
1000 PRINT CHR$(4)"EXEC MESSAGE":END
```

The Apple DOS manual does not use the term "record" with sequential files. The manual demonstrates writing only one value per record, and hence uses the term "field" instead of "record". Note that if a record contains only one value, it is composed of only one field.

RESCUING DAMAGED FILES

Mike Firth

I seem to have a great talent for saving a program to which I have made dozens of changes just after I have destroyed the end of it in some way so that when I reload it and list, I get endless trash that can not be DELeted.

Recently when this happened, I tried something that rescued me. I typed in the routine shown on page 76 of the DOS manual to list a program out as a text file. I converted the routine to a one line addition as line 1. When I did this, I had to hit RESET to get the cursor] back, but the line was in. In the routine, I only listed the lines I knew were good. When I EXECed the file back in, it was rescued.

Changing Volume Numbers.....Tim Hartley

[Reprinted from the HARVEST Newsletter, published by the Northwest Suburban Apple Users Group, Cincinnati, Ohio, December, 1980.]

Often we want to change the volume number of a diskette so we might be able to differentiate it from one that currently has the same number.

Obviously one solution is to re-initialize all of our disks and use new volume numbers. Then we could transfer all programs from the old disk to the new. Alas, this is very tedious and time consuming. The solution, as well as the problem, lies in the fact that the volume number is written in the header for each and every sector on the diskette when it is first initialized. Then, anytime DOS accesses a sector, the volume found in that sector's header is stored in one of DOS's little memory locations. I did devise a way of using a "pseudo" volume number that works quite well.

First of all, here are two POKE's which should be either part of the HELLO program or POKEd in manually:

```
POKE 44476,193
POKE 44477,179
```

Once they have been entered, the CATALOG command will display the "pseudo" number instead of the real volume number encountered for the sector.

Here is a short program which will allow you to modify the "pseudo" number on all of your disks:

```
]LIST
10 TEXT : HOME
20 POKE 44476,193: POKE 44477,179
30 CALL 45047: REM READS IN VTOC
40 INPUT "ENTER NEW VOLUME NUMBER:  ";VL
50 POKE 46017,VL
60 CALL 45051
70 PRINT CHR$ (4)"CATALOG"
```

The POKEs and CALLs are for a 48K machine and they must be modified for a smaller one. It works on either DOS 3.2.1 or DOS 3.3.

INVERSE & FLASHING MODES FOR APPLE PASCAL TEXT DISPLAY

By Ron DeGroat

The assembly language procedures shown in Listing #1 allow the user to display text in FLASHing, INVERSE and NORMAL modes with Apple Pascal in much the same way that Applesoft BASIC does.

These procedures may be directly linked to the Pascal host (see example program in Listing #2) or stored in a UNIT in the SYSTEM.LIBRARY along with other useful utilities (such as PEEK and POKE routines, q.v., "The Multi-Lingual Apple" column in the Feb. and Mar./Apr. 1980 issues of Call A.P.P.L.E.). For more information about the control codes ($CØ83 and $CØ88) see Appendix D of the Apple Language System Installation and Operating Manual.

It should also be noted that since the Apple uses the top two bits of the ASCII character code to select INVERSE and FLASHing modes, lower-case letters cannot be displayed in these modes.

LISTING #1: INVERSE, NORMAL & FLASH PROCEDURES

```
.PROC INVERSE
LDA ØC083  ;SELECT 2ND 4K BANK
LDA ØC083  ;AND WRITE-ENABLE
LDA #ØØ
STA ØD8ED  ;CLEAR BITS 6 & 7
LDA ØC088  ;SELECT 1ST BANK & WRITE-
RTS        ;PROTECT, THEN RETURN
;-------------------------------
.PROC NORMAL
LDA ØC083
LDA ØC083
LDA #8Ø
STA ØD8ED  ;SET BIT 7 FOR NORMAL MODE
LDA ØC088
RTS
;-------------------------------
.PROC FLASH
LDA ØC083
LDA ØC083
LDA #4Ø
STA ØD8ED  ;SET BIT 6 FOR FLASH MODE
LDA ØC088
RTS
```

(Footnote to Listing #1: For Pascal newcomers, constants in Apple Pascal assembly language must start with an integer, Ø to 9. If the first digit is greater than 9, i.e., A, B, C, D, E or F, the number must be prefaced with a Ø, as shown in Listing #1.)

LISTING #2: EXAMPLE PASCAL PROGRAM

```
PROGRAM TESTSTUFF;

PROCEDURE INVERSE;  EXTERNAL;
PROCEDURE NORMAL;   EXTERNAL;
PROCEDURE FLASH;    EXTERNAL;

BEGIN
  GOTOXY(Ø,1Ø);
  WRITE ('THIS SHOULD BE ');
  FLASH;
  WRITELN ('FLASHING');
  NORMAL;
  WRITELN;
  WRITE ('AND THIS SHOULD BE ');
  INVERSE;
  WRITE ('INVERSE');
  NORMAL;
END.
```

How the Modification Works:

Listing #3 shows a section of code stored on disc in Block No. 4 of SYSTEM.APPLE file. When loaded into memory, this code resides in the second 4K bank of RAM, which must be selected and write-enabled with the control codes described above before any change can be made.

LISTING #3: CODE EXCERPTS FROM SYSTEM.APPLE

```
D8E8: E9 2Ø SBC  #2Ø  ;CONVERT TO U.C.
D8EA: 29 3F AND #3F  ;CHAR. MASK
D8EC: 9  8Ø ORA #8Ø  ;MODE SELECT:
                     ;  4Ø = FLASH
                     ;  ØØ = INVERSE
```

The character mask clears the top 2 bits of the character, and the next instruction sets the bit necessary for the mode selected. Thus, by changing the contents of memory location D8ED, as indicated we can select either FLASHing, INVERSE or NORMAL modes.

TIMING TRICK Mike Firth

I recently wrote a routine into one of my programs that would setup an interval timer. This is a program I leave in the computer while I work on other things and I needed an interval timer.

Of course, I had to have a loop to do the timing and I had to set the exact value so it would count off minutes. After trying a few values at random, I suddenly realized that all I had to do was start the loop running and stop it with a control C or by hitting RESET (on Applesoft) then print the value of the counter. Then I reset the end of the loop to that value and tested.

This is the program I wrote. It permits an exit at any time, stopping the timer. At the end of the interval the beeper sounds. During the timing the current minute and the total are displayed.

```
1900 REM  TIMER
1905 PRINT : INPUT " NUMBER OF MINUTES";Q :PRINT
1910 FOR J= 1 TO Q : VTAB 21 : PRINT J" OF "Q
1920 FOR Q1= 1 TO 4500 : Q2= PEEK (-16384): IF Q2>127 then 1945
1925 NEXT
1940 NEXT : Q$ = CHR$(7) : FOR J=1 TO 50 : PRINT Q$; : NEXT
1945 POKE -16368,0 : RETURN
```

NOTES: If you leave off the ; in line 1940, the screen will be cleared and the beeps will be further apart in time.

A nothing loop to count off a minute takes about 50,000 counts (e.g. FOR J=1 to 50000 : NEXT)

The PEEK in line 1920 checks the keyboard. The POKE in 1945 clears the strobe. If you leave out the POKE and your menu (or where ever this routine returns to) uses GET to enter the choice, then whatever key breaks the timing is input with the GET, with good or bad results.

ON ERR -- Punt?.....Robert F. Zant

Have you ever been three levels deep in GOSUB's and discovered that you need to intercept error conditions? If you have then you know that the use of 'ONERR GOTO' will damage the return linkages for the GOSUB's.

Those of you who do not give up easily may have also discovered that Apple provided a solution for this problem (see page 82 of the Applesoft manual). The solution consists of a small machine language routine that can be placed anywhere in RAM and then called by your program after an error has occurred.

Now the purpose of this note...you do not need the little machine language routine. It already exists in the Applesoft interpreter! Simply CALL -3288 (11055 for RAM Applesoft) after an error and before a RETURN.

PICK A NUMBER Mike Firth

I happen to like to combine data entry and program flow control. So you will often find a statement in one of my programs that looks like this one:

```
INPUT "ENTER PAYMENT (OR Ø TO QUIT)";Q
```

One of the nuisances of BASIC is that if the normal input requires three items, say a day, month and year, my technique requires that three items, not just one, be entered to satisfy the INPUT command.

```
INPUT "ENTER DAY, MONTH, YEAR (OR Ø,Ø,Ø)";D,M,Y
```

One way of curing the problem is using three INPUT commands, but that adds three RETURNs to be pushed also. Further, I had a situation in which sometimes a single entry was OK and other times I needed two numbers (blanking a single line or a group of lines, if you want to know).

My solution is given below. This subroutine collects up to 10 numbers, stopping when the return key is pushed. It expects numbers to be separated by commas. It could be modified to collect a limited set of numbers,whether or not a return was pushed, by setting K outside the routine. It could also permit separation by spaces or something else (such as slashes for date entry 5/16/80). Note that error checking is not done in the routine, although letters will be converted to Ø values. Errors should be checked for outside the routine. Also, note that the back space arrow does not produce reasonable results in this version.

```
790  REM MULTI NUMBER ENTRY
791  K= 10
792  FOR J=1 TO K_ : Q(J)=0
793  GET Q$ : PRINT Q$ : IF Q$=CHR$(13) THEN RETURN
794  IF Q$="," THEN NEXT J  :RETURN
795  Q(J) = VAL ( STR$( Q(J) ) + Q$ ) : GOTO 793
```

Normal return will have a set of numbers in the array Q, with J giving number of numbers.

P.S. While I was writing this article, another version of this routine occurred to me, one that would permit use of the backspace and forward arrows. You will note that the structure is much the same. It is really a "parsing" routine, that breaks apart the line.

```
790  REM MULTI NUMBER ENTRY IN STRING
791  K=10
792  INPUT Q$   : J=1 : FOR J= 1 TO K : Q(J)=0 : NEXT J
793  FOR Q=1 TO LEN(Q$)
794  IF MID$(Q$,Q,1)="," THEN J=J+1 : IF J>K THEN RETURN
795  Q(J) = VAL ( STR$( Q(J) ) + MID$(Q$,Q,1) ) : NEXT Q :RETURN
```

ANSWER: 2058 with Applesoft in ROM. If you guessed 38376 or something near that, you skipped on saving memory space. If the line 2 INPUT Q$ is added to the program and the equivalent line is typed in, your answer will result, but Applesoft saves memory space by pointing to the actual line in the program. The same thing happens with DATA statements. Thus you can save considerable memory space by assigning any repeatedly printed string to a variable.

Radix Sort

Harry L. Pruetz
709 Caldwell St.
Yoakum, TX 77995

I found the article in the August AppleGram on Sorting techniques interesting, but I missed the classic "radix" sort technique. The more recent history of the radix sort includes the old IBM card sorters. All of the sort techniques included in the August article by Bob Sander-Cederlof were of the type called minimal-storage sorts; that is, they require hardly any additonal memory beyond the data array to be sorted. The radix sort, on the other hand, requires extra memory equal to the amount taken by the original data.

An Applesoft progrm is listed here which implements the radix sort for up to 1000 items. For 50, 100, and 200 items, the sort times in seconds are 15, 29, and 59. Note the linear dependence on N, the number of items to be sorted.

Two equally dimensioned arrays are needed, since, for example, all of the numbers may be odd. The sort may be made descending by changing the equal test in line 150 to a not equal test.

Line 30 sets B$ to a control-G, or bell, which is used to signal the end of the sort for timing purposes.

Lines 10-60 set up the sort and the random data to be sorted.

Line 100 waits until any key is pressed to start the sort.

Lines 110-190 perform the radix sort.

Lines 1000-1020 print out the sorted data.

Line 150 does most of the work for this sort. It is essentially comparing one bit of the data on each pass. The data must be type integer, and be in the range from 0 to 999.

```
]LIST

10 T = R = I = RAD = 0
20  DIM R(1000),RO(1000)
30 B$ = ""
40  INPUT "NO. OF ITEMS = ";N
50  IF N < 1 OR N > 1000 THEN 102
    0
60  FOR I = 1 TO N:R(I) =  INT (1
    000 *  RND (1.0)): NEXT I
100  GET C$
110 MAX = 1000
120 RAD = 1
130 NE = 0
140 NO = 0
150  FOR I = 1 TO N:R = R(I):T =
     INT (R / RAD): IF T = 2 *  INT
    (T / 2) THEN NE = NE + 1:R(N
    E) = R: GOTO 170
160 NO = NO + 1:RO(NO) = R
170  NEXT
180  FOR I = 1 TO NO:R(NE + I) =
    RO(I): NEXT
190 RAD = 2 * RAD: IF RAD < MAX THEN
    130
999  PRINT B$
1000  FOR I = 1 TO N: PRINT R(I),
    : NEXT : PRINT
1010  GOTO 40
1020  END
```

Trivia Quiz

Mike Firth

The AppleSoft manual tells us that at locations 105 and 106 there is a pointer to simple variable space which ends at a location stored in 107 and 108. Within this space, in blocks of seven memory locations, are stored the names and values of real and integer variables and the names and pointers to string variables. Thus the following would allow us to examine the blocks:

```
FOR J= PEEK(105) + PEEK(106)*256  TO PEEK(107)+PEEK(108)*256 STEP 7
FOR K= 0 TO 6 :  L(K)= PEEK(J+K)   :  PRINT L(K)" ";:NEXT K : PRINT
  ....
NEXT J
```

Each of the blocks is in a different format requiring conversion to characters or multiplication. The following is the formula for a string variable.

```
IF L(0)<128 AND L(1)>127 THEN PRINT CHR$(L(0));CHR$(L(1))" "L(2)" "L(3)+
  L(4)*256
```

Which will print the name (first two characters) of the variable, the length of the string and the memory location. *(Note: my manual says that variable names are stored with the first letter greater than 127--negative--and the second less--positive--for a string variable, but testing reveals the reverse is true as included above.)*
Actual printing of the string will result with the following line.

```
FOR M=L(3)+L(4)*256  TO  L(2)+L(3)+L(4)*256 : PRINT CHR$(PEEK(M)); :NEXT
```

Now, with all that back ground, I will offer the quiz question after one more fact "Strings are stored in order of entry from HIMEM: down."

Suppose I put all of the fragments above in a program along with this line:

```
1 Q2$="THIS IS Q2$; A TEST LINE"
```

When I run the program, I will get printed lines something like this:

```
Q2 24 ##  0 0
THIS IS Q2$; A TEST LINE"
```

Except, of course, where I have typed ##, there will be a number of the memory location of the string. On a 48K machine with DOS loaded, HIMEM= 38400.

QUESTION: What will be the value of ## in the above program?

DOWN LOAD

Mike Firth

If you have tried getting data from another computer, you may have run into the same problem I did: While the data seems to be coming over the phone lines and is displayed on the screen, it doesn't get into the strings that you want to save it in.

I called Apple and their solution did the trick, sort of, so I was able to load data from a Cyber 6600 and save it on my Apple disk, using a communication card and a modem.

I was told that some computers send out NUL characters at the start of transmissions. If these get in the start of an Apple string, the operating system thinks there is nothing in the string. Apple suggested several GET statements to capture the NUL's.

I was able to get rid of them another way, as the Cyber operating system has a ROUT command which sets the number of RUBOUT (NUL) characters sent. Normally this is used to increase the number to allow for a slow carriage return on a printer. I could set it to zero and capture my data.

Another Little Window.....Ray Thompson

Here is an Integer BASIC version of the "little window" titling program in last month's newsletter. Notice the "exit" I have added to get rid of the blank line in the window, and to get the cursor out!

```
>LIST
1300 REM TITLE SUBROUTINE**********
1302 REM (IDEA:B. SANDER-CEDERLOF....
     INTEGER VERSION: R.M. THOMPSON)
1304 DIM T$(80): TEXT : CALL -936

1306 FOR I=5 TO 16
1307 FOR T=10 TO 31
1308 VTAB I: TAB (T): PRINT "*";

1309 NEXT T
1310 NEXT I
1311 FOR D=1 TO 300: NEXT D
1312 POKE 32,10: POKE 33,20: POKE
     34,5: POKE 35,15: CALL -936
     : VTAB 15
1313 FOR D=1 TO 300: NEXT D
1314 REM (MORE,BELOW)
1316 FOR I=1 TO 4: CALL -380
1317 IF I=1 THEN T$="THIS IS THE WAY
     YOURMESSAGE WILL BE DIS-PLAYED I
     N THE WINDOW.....
     "
1318 IF I=2 THEN T$="FLAGS CAN BE SET
      TO PRINT A PARTICULAR  STRING I
     N INVERSE...AS YOU SEE HERE.....
     "
1320 IF I=3 THEN T$="EACH STRING T$ W
     ILL HOLD FOUR LINES. THEPRINTING
      SPEED CAN  BE ADJUSTED WITH 'D'
     "
1322 IF I=4 THEN T$="::::::::::::::::
     :::::----------------:::::::
     ::::::::::::----------------------
     "
1324 FOR J=1 TO 80:T=J
1326 IF J>20 THEN T=J-20
1328 IF J>40 THEN T=J-40
1330 IF J>60 THEN T=J-60
1332 IF I=2 THEN CALL -384
1334 TAB (T): PRINT T$(J,J);
1336 FOR D=1 TO 50: NEXT D
1340 NEXT J
1342 NEXT I
1345 CALL -380: PRINT "**************
     ******"
1346 POKE 32,0: POKE 33,40: POKE
     34,0: POKE 35,23
1350 VTAB 14: CALL -958
1355 FOR D=1 TO 3000: NEXT D
1360 LIST 1300,1314
1365 END
```

TESTING RANDOMNESS MIKE FIRTH

While there are statistical ways of testing a random function on a computer, here is one that is quick and dirty. You have to create an array, like this:

```
100 DIM A(100) : FOR J=1 TO 100 : A(J)=INT(RND(1)) : NEXT J
```

Then you have to sort it. I'll leave the method up to you. Then run this line:

```
200  FOR J=1 TO 99  : R= A(J+1)-A(J)+R : S= A(J)+S : NEXT :S=S+A(100)
          PRINT R/99, S/100
```

The two numbers printed are the average interval and average number in the array. Ideally the answers would be 1 and 50 for a truely random array. Larger arrays are more likely to approach the ideal.

TURNING THE PAGES

with David E. Laden

Well, eventhough TURNING THE PAGES received processing time this month, it did not get enough to "catch up" and so it is still a little behind.

One note before we get to the PAGES: I would still appreciate hearing from more of you, the members of Mini'app'les, concerning this column. As published in the MAY 1981 newsletter, I would like to know what you think of this column, its format, and contents. Give me a call (my number is listed inside the front cover) or catch me at the next meeting.

BYTE -- MAY 1981

Byte's topic for May is Software Piracy.

Editorial: How Can We Stop Software Piracy? by Chris Morgan. Pages 6 and 10.

rdware Review: The Epson MX-80 and -70 Printers by Kevin Cohan. Pages 22-34.

Byte's Arcade: Tranquility Base by Robin Moore. Pages 112-114. Asteroids in Space and Planetoids by Oliver Holt. Pages 116-120. These games for the Apple are reviewed.

Using Page Two with Apple Pascal Turtle Graphics by Bruce Wallace. Page 122.

Washington Tackles the Software Problem by Christopher Kern. Pages 128-138.

Legal Protection for Computer Hardware and Software by Stephen A. Becker. Pages 140-146.

Using Interrupts on the Apple II System by George M. White. Pages 280-294.

Digital Plotting with the Apple II Computer by Dr. Richard C. Hallgren. Pages 296-314. The Houston Instrument Hiplot digital plotter is used.

Recursion and Side Effects in Pascal by Robert Morris and James Perchik. Pages 316-324.

A File Catalog System for UCSD Pascal by Edward Heyman. Pages 408-427. This program will maintain a master listing of all your Pascal files and diskettes.

COMPUTE! -- MAY 1980

Computers and Society by David D. Thornburg. Pages 12-14. The book "Without Me, You're Nothing" is reviewed.

Computer Aided Instruction, Boon or Bust? by Alfred D'Attore. Pages 18-20. A commentary on the future of Computer Aided Instruction (CAI).

Using Named GOSUB And GOTO Statements In Applesoft BASIC by M.R. Smith. Pages 64-68. This is another use for the Ampersand (&). Machine language program and Applesoft demo program included.

Commas, Colons And Quote Marks Too by Craig Peterson. Page 69. A three line input anything routine written in Applesoft.

Generating Lower Case Text On The Apple II Plus Using The Paymar Chip by David Shapiro. Pages 70-72. This is an Applesoft routine used to convert Applesoft statements.

CREATIVE COMPUTING -- MAY 1981

Fantasy Games - Part 2 by David Lubar Pages 20-24. Some adventure games for the Apple are described.

Buyer's Guide to Small Business Computers. Pages 26-27. This is a comparison chart of 11 computers.

An Open Letter To: The Personal/Small Business Computer Industry From: A Businessman Subject: A Serious Problem in Your Industry by David Henderson. Pages 40-45. This article includes A "Reply From Apple Computer, Inc. by Phil Roybal and an "Editorial Comment."

Tero's Apple by Jan Sand. Pages 62-63. "A special adaptation of the paddle control for the severly handicapped..."

Six Programs for the Investor by Linda Barkaszi. Pages 76-80. Six packages including three for the Apple are described.

Break Even Analysis with VisiCalc by George Blank. Pages 96-98.

Executive Privilege by Leland D. Young. Pages 136-142. This article describes how EXEC files may be used to back up a diskette. Program listings included.

Bombproof Data Entry - A Revision by Susan Luca. Page 144.

How to Solve It - With the Computer Part 7 by Donald T. Piele. Pages 146-154.

Applesaurus & The World Brain by Derek Kelly. Pages 158-166. This program named "Thesaurus Analyser" is written in Applesoft BASIC.

Apple Cart by Chuck Carpenter. Pages 200-207. This months topics include the DOS 3.3 System Master Diskette and "Lo-res Block Letters" in Applesoft among other things.

KILOBAUD MICROCOMPUTING -- MAY 1981

Kilobaud Microcomputing features word processing in this issue.

Processing Written Words by Dennis Bathory Kitsz. Page 32. An overview of word processing.

Inspiration From the Muse by David C. Goodfellow. Pages 53-54. Super-Text II from Muse, a word processing program for the Apple, is reviewed.

A Simple Text Processor by Henry Simpson. Pages 80-85. An Applesoft BASIC program listing is included.

The Animated Apple II by Edward Burlbaw. Pages 112-114. The author reveals additional HI-RES pages for your Apple. Applesoft examples included.

A Tiger's-Eye View Of Computer Graphics by Jim Hansen. The author looks at the IDS 440 and 445 Paper Tiger printers.

MICRO -- MAY 1981

MacApple by David Lubar. Pages 9-11. This machine language "routine allows substitution of unreserved control keys as shorthand for commonly used Integer BASIC commands."

Applesoft Variable Dump by Scott D. Schram. Pages 23-24. This machin language routine will dump values of Applesoft variables to help in debugging.

How Microsoft BASIC Words by Greg Paris. Pages 31-37.

Apple Memory Maps, Part 2 by Peter A. Cook. Pages 45-56. This part contains the machine language routines used to produce the memory maps presented in part 1.

Protecting Memory from DOS by Glenn R. Sogge. Pages 81-82. This article tells you how to reserve memory above DOS.

PERSONAL COMPUTING -- MAY 1981

The Ins and Outs of High-Level Languages by Paul Bierman. Pages 10-14 and 67-73. BASIC, COBOL, FORTRAN, ALGOL, Pascal, PL/1, PILOT, LISP, and Ada are explored.

A User's Guide To Operating Systems by Alan Boyd, Phillip Good, and Stanley Veit. Pages 27-32, 83-87, and 108. Systems covered included CP/M, Apple's DOS, Apple Pascal, and SOS for the Apple ///.

Educational Computing: Are Computers Hazardous To Your Child's Health? By Carol Klitzner. Pages 34-35 and 95.

Understanding Interpreters and Compilers by Ken Mazur. Pages 39-42, 77-81, and 110-116. Interpreters and compilers are explained with a cute analogy of an imaginary foreign clerk working in an office.

Personal Computers: Products for Every Need. Pages 45-55 and 74. Personal Computing looks at computers from nine manufacturers in this first of a two part series.

Generating Lower Case Characters with Pascal by Sam Gaylord. Pages 65-66 and 91. An Apple Pascal program called GFRPRNT is included which utilizes Pascal's high resolution character set.

Continued on page 3

MINI'APP'LES MECC ORDER

MECC disks may be ordered only by Mini'App'Les members who are residents of Minnesota. The diskettes were developed for the Minnesota schools by Minnesota Educational Computing Consortium, a tax-funded agency, and you are not to make copies for non-residents.
We refer non-residents to MECC Publications 612/376-1118.

No.	Title	Vol.	Special Emphasis	Quantity	No.
701	MECC Demo	1		--------	701
702	Elementary	1	Mathematics	--------	702
703	Elementary	2	Language Arts	--------	703
704	Elementary	3	Social Studies	--------	704
705	Elementary	4	Science	--------	705
706	Mathematics	1	Senior High	--------	706
707	Science	3	Middle School	--------	707
708	Science	1	Senior High	--------	708
709	Science	2	Senior High	--------	709
710	Social Studies	1	Senior High	--------	710
711	Social Studies	2	Senior High	--------	711
712	Music Theory	1		--------	712
714	Business	1		--------	714
715	Teacher Utilities	1		--------	715
716	Aestheometry	1		--------	716
717	Agriculture	1		--------	717
718	Driver's Education Industrial Arts	1		--------	718
719	Elementry	5	Language Arts (Prefixes)	--------	719
720	Programmer's Aid	1	**	--------	720
724	Shape Tables	1	HGR Utilities **	--------	724
725	Elementary	6	History	--------	725
726	Spelling	1		--------	726
727	Special Needs	1	Handicapped Spelling	------	727
728	Spelling	2	Adult words	--------	728

** The Shape Tables and Programmer's Aid require the support booklets to be used. We will sell the disk and book as a package only, to be delivered at the the club meetings.

PRICES

Disketts	$5.00 each		----------
Disk & Booklet	$8.50 each		----------
Mailing	$2.00 each	(Disks only)	----------
			==========
		Total	----------

Name ----------
Address ----------
Phone ----------

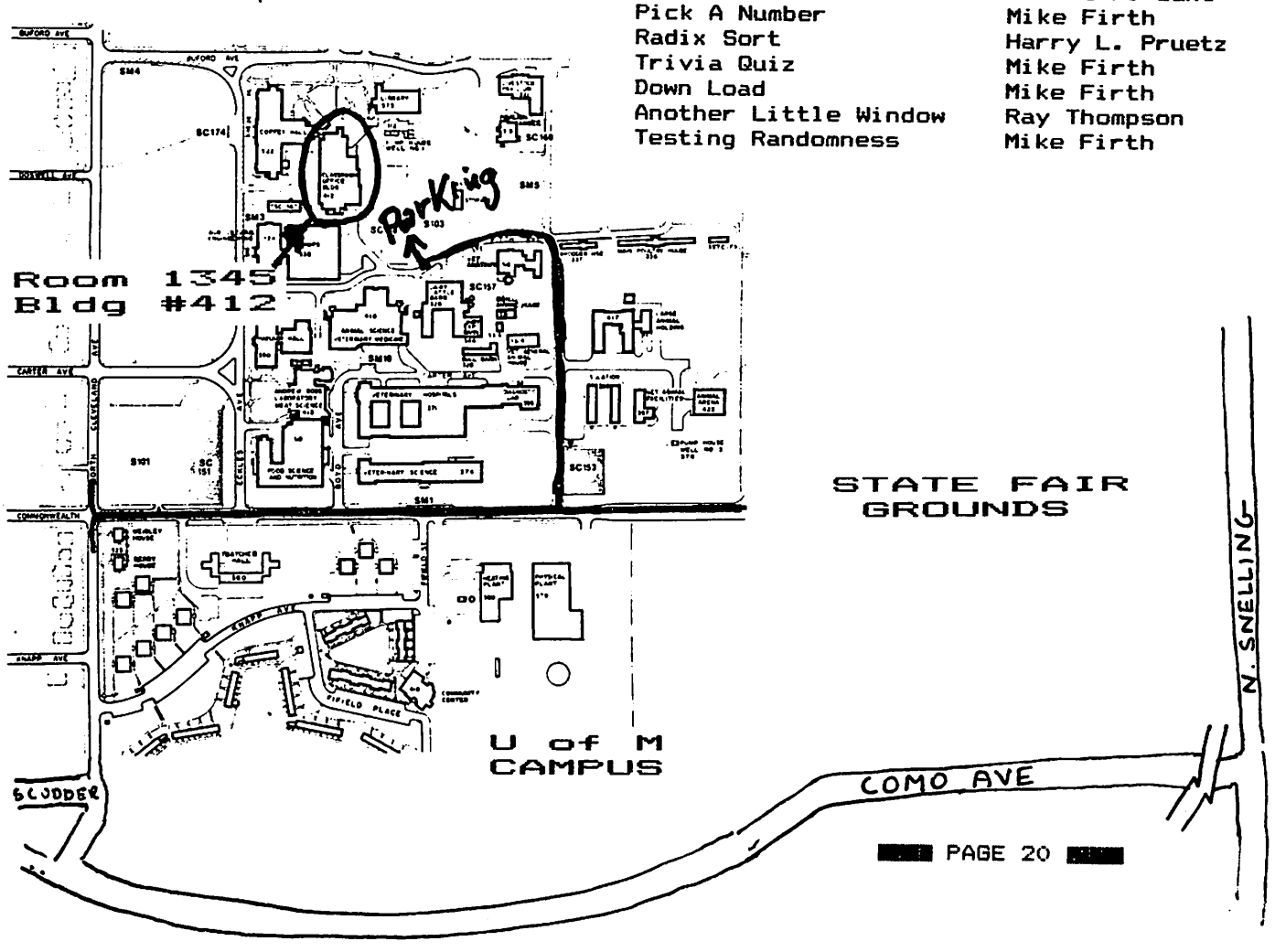
I understand that I am not to make copies for non- residents.

Signed ----------

NOTE MECC DISKS ARE IN DOS 3.3 !

IN THIS ISSUE

EXTRACTS FROM THE APPLEGRAM



MINI'APP'LES
13516 Grand Avenue South
Burnsville
Minnesota, 55337

Bulk Rate
U.S. Postage
PAID
Hopkins, MN
Permit 631