N.Y. 99er

INTERFACE

FEBRUARY 1989 $1.25

CONVERSIONS!

CHANGING PRB TO TIB...
IT'S SOOOO EASY!!

SEE PAGE EIGHT

INTERFACE CONTENTS

WNY 99'ERs USERS GROUP EXECUTIVE BOARD

PRESIDENT.....Robert Coffey Jr.
VICE PRESIDENT.....Rich DiChristina
SECRETARY.....Mary Ann Robson
TREASURER.....Jim Collins

COMMITTEE CHAIRMEN

PROGRAM.....JoAnn Suttner
LIBRARY.....Robert Coffey
SERGEANT AT ARMS.....Hollie Brashear

INTERFACE EDITORIAL STAFF

EDITOR.....Harry Brashear
CONTRIBUTING EDITORS....Greg Mioducki------Robert Coffey

CLUB BBS *THE 39 STEPS* 1(716)837-2818

INTERFACE is a publication of the WESTERN NEW YORK 99'ers USERS GROUP. It is intended as a source of news, information and instruction. Articles and opinions expressed in this newsletter do not necessarily reflect the opinions of either the editorial staff or members of the users group.

INTERFACE will appear monthly except bi-monthly from June to September, and will be available to all paid members of the users group.

Articles submitted for publication must be received by the second week of the month to be included in the following month's issue. For those submitting articles prepared by the TI-Writer word processor package, please submit a disk containing the file to be published. The disk will be returned to you. The inclusion of articles will be solely at the discretion of the editorial staff. The editors further retain the right to edit any article submitted.

Advertising space is available at modest rates. For information on rates, write to the INTERFACE at the address below. Classified ads must be limited to eighty (80) characters including spaces and are free of charge to all paid up members. Address all inquiries to:

WNY 99'ers USERS GROUP
c/o 84 Glenhaven Dr.
Amherst, NY 14120

The WNY 99'ers USERS GROUP meets on the second Wednesday of every month . For information call Bob Coffey, President, at (716)833-4714. Yearly Dues are $20 per family and $15 for an associate (Will attend no more than three regular monthly meetings).

Members are entitled to receive the monthly newsletter, have access to the software library and attend all general meetings, seminars and workshops sponsored by the Group.

THE PRES'S PLACE

BOB · COFFEY · PRESIDING

It was a long time coming, but here we are at the 5th Anniversary of our newsletter, Interface! I've had the pleasure of writing something for this publication since its inception five years ago. Even our grand-daddy, HTB, can't lay claim to that.

It has been a long five years, and our club has seen many ups and downs in those years. The officers that brought this club together originally are all gone. Paul Thomas, Paul Whiddon, Bruce Rechin, and Steve Royce formed this club out of their love of what this computer can do, and we now continue to cultivate the orginization that they planted so many moons ago. My hat is off to you gentleman who had the vision, talent, and guts. (Editor's Note: MICROpendium also has their 5th anniversary with their February 1989 issue, so we are in good company!)

* * *

Our friendly neighbors down south in the West Penn 99'ers have some interesting news for you. Eric Zeno has come up with a board that will let you put Extended Basic, 32k Memory expansion, speech syntehsizer, and a battery-backed clock all inside the console! While requiring no extra power, it will have a RESET circuit and will virtually eliminate lockups. Eric is looking to see how many TI'ers are interested before comitting himself, so if you are interested, write (please include an SASE) or call him. Eric Zeno, 414 Highland Road, Pittsburgh, PA, 15235, (412) 371-4779. He hopes to offer the board for about $15, and could become as popular as the Horizon RAM disk kit became.

* * *

GREG$ TIPS was planned for this issue, but we ran out of space, so you can look forward to it next month. REAR WINDOW will also return next month.

-BASIC BASICS-

GETTING STARTED IN PROGRAMING

Delay routines in REAL TIME

Paul E. Scheidemantle

Well folks, it's back to programing subroutines this month. We are going to setup a delay routine for basic and extended basic, which can be used in real time... as we know it. This will take the guesswork out of timing loops and having to fiddle with the numbers to get the delay just right.

First I'd better explain what I'm talking about when I mention a loop. A for next loop is setup as such: FOR A=1 TO 100 :: NEXT A . What happens here is that the value of "A" is incremented by 1 each time the program sees the NEXT A statement, until finally, it has reached the "TO" value in the statement. 1..2..3..4..5, etc., right on up to 100 (or whatever figure you choose). This particular loop is referred to as a delay routine.

Most delay routines as above are set up just as you see them with numbers ranging from 1 to 100 or higher. The idea is to make the computer wait (delay) before doing anything else. This could be a timer for a title screen, between notes in music, or whatever you might want to stay on the screen for a certain length of time.

So the idea here will be to design a special loop that works in REAL TIME, instead of those cryptic numbers that we have to continually guess at.

The first thing we will need to do is find out how long it takes the computer to count through a loop. Then we can find out how far we have to count to get increments of seconds. Shown below is a small program that was run to find out this information. The call sound statements were used to give an audible sound at the beginning and end of the routine making it easy to time. I used a stop watch in hundredths of seconds to do this. If you are using your watch and have only even seconds to work with, you might want to increase the length of the loop (make it 100000 instead of 10000) to get a more accurate figure.

FOR BASIC:

```
100 CALL SOUND(-10,880,0)
110 FOR A=1 TO 10000
120 NEXT A
130 CALL SOUND(-10,880,0)
```

This routine required 28.28 seconds to run. Now b dividing 10,000 by 28.28 we get a value of 353.6067893. B sure to do this the easy way and use the computer. Type i "PRINT 10000/28.28" (without the quotes), and the number wil just pop up like magic.

FOR EXTENDED BASIC:

```
100  CALL SOUND(-10,880,0):: FOR A=1 TO 10000 :: NEXT A :
CALL SOUND(-10,880,0)
```

We do the same thing for extended basic, except in a multiple statement line. The result is 33.76 seconds, givin us a value of 296.2085308. (For Myarc XBII users the second are 38.10 with a value of 262.4671916).

Gee now what do we do with these great figures now that w have them. Well I'm certain most of you are way ahead of m and have already gone on to the next article. But for those o you who were kind enough to stay with me and listen to m ramblings... here comes the answer.

If we multiply the value we got in the above example by an number, we get the values for our soon-to-be loop that come out in seconds (not cryptic numbers). Below are two example o subroutines to show you how they are used.

First we will look at the basic routine, and here is how i works. We will use two variables; SECONDS and A. In line 14 we give the variable SECONDS the value of 30 (30 seconds or an positive number you may wish to use here). In line 150 w GOSUB to line 500 (the delay subroutine). Line 160 jus separates the main program from the subroutines. Now to lin 500... We set up a loop called "A" in which we count from on (1) to 353.61 (number required to complete one second i basic), times the value given to the variable SECONDS. Lin 510 sends the program back to line 500 to increment "A" by on (1), until it equals the high number (second part of th perimeters (1 was the first part)). Finally line 520 return control to the main program at the point right after where th GOSUB command was used.

```
140 SECONDS=30
150 GOSUB 500
160 END
500 FOR A=1 TO (353.61*SECONDS)
510 NEXT A
520 RETURN
```

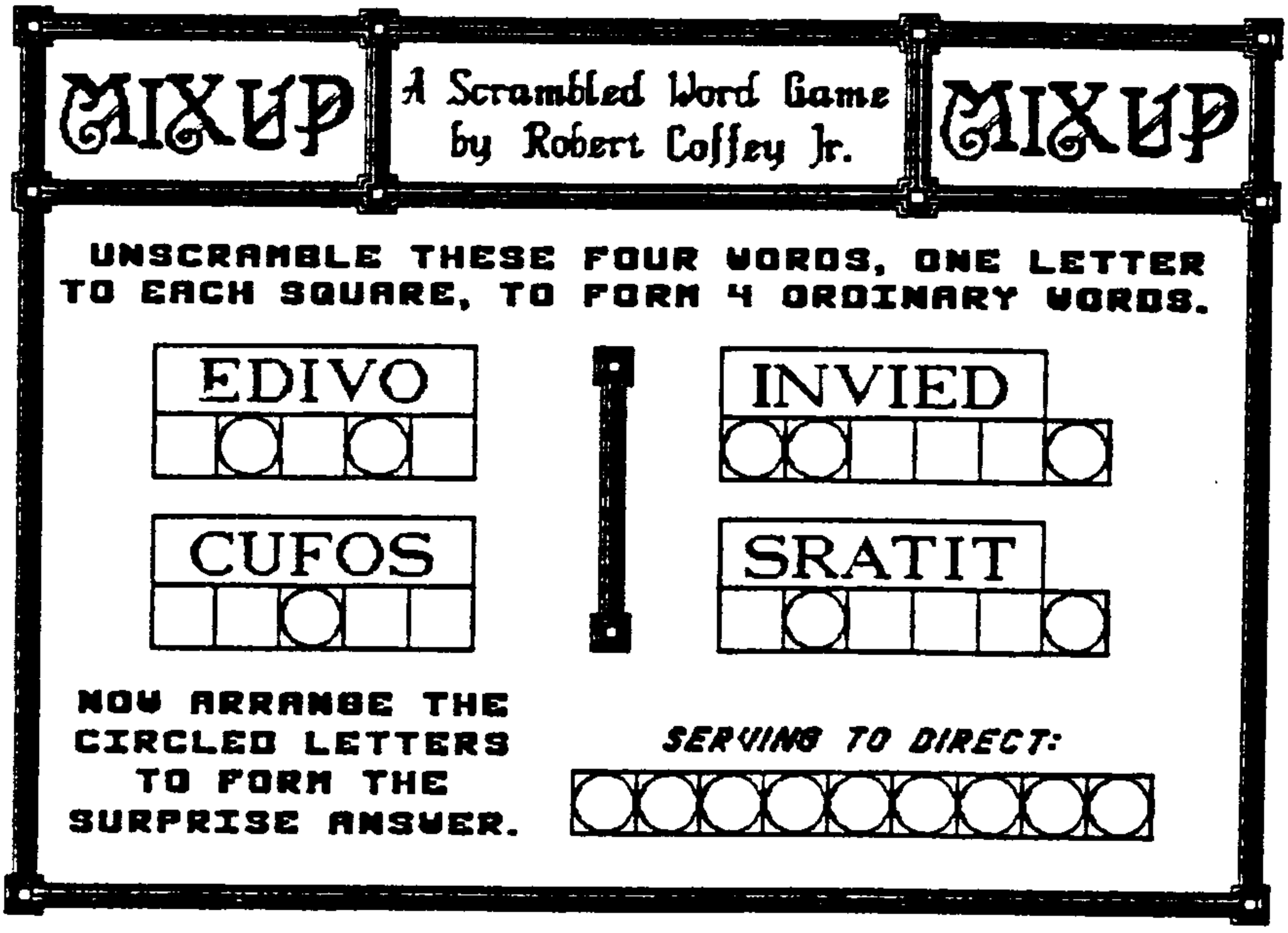
Now to the extended basic routine, and here is how works... We will use two variables; SECONDS and A. In li 110 we call our subprogram DELAY and pass the value of t number of SECONDS we want. (In this case, 30 seconds, or a positive number you may wish to use here). Line 120 jus separates the main program from the subprogram(s). Now to li 500, our subprogram. We set up a loop called "A" in which count from one (1) to 296.21 (number required to complete on second in extended basic) times the value given to the variab SECONDS. The NEXT A tells the program to go back and increme "A" by one (1) until it equals the high number (second part

the perimeters (1 was the first part)). Finally the SUBEND returns control to the main program at the point right after where the CALL command was used. Now you have a delay routine that works in REAL TIME! If you were using Myarc's XBII then replace the value 296.21 with 262.47, and you should be all set to run.

```
110 CALL DELAY(60)
120 END
500 SUB DELAY(SECONDS):: FOR
A=1 TO (296.21*SECONDS):: NE
XT A :: SUBEND
```

I hope that these routines help you with your programing and that this article will help you undérstand where all those cryptic numbers come from in programs. Another thing you should keep in mind is that the extended basic subprogram in line 500 can be saved in a merge format, for future use with other programs. There is hardly any reason to have to do this more than once. This can be done by saving the program thus SAVE DSK1.FILENAME, MERGE (the result is a DV/163 file on disk). Then in the future you can merge it into a program by first loading the program and then typing in MERGE "DSK1.FILENAME" and it will now be part of the program. It should be noted that when a file is merged to a program that it will overwrite any existing lines with the same number, so be sure to give your routine a high enough line number that this does not happen.

I guess I've said this many times but here we go again.... With a blank (or backup) disk and a little imagination you can do no wrong! So have fun and enjoy your computer!



NEWS + VIEWS (SCANDAL:OPINION)

This could be the best news in some time folks. THERE IS A VERY STRONG POSSIBILITY THAT TI-ARTIST WILL BE GETTING A FACELIFT! My source for this information is TI-Source, the Texaments bulletin board. Steve Lomberti, Texaments owner, has requested that you drop in and tell him what you would like tc see changed. Before I get into that though, I would like tc venture a guess as to how this happened. It is also a back handed compliment to our present community.

I had heard that Chris Fariaty, Artist's programmer, would never do anything for the TI again - I don't know why such a statement was made. However, in case you hadn't noticed, it's Chris's dad that created TI-Base, the hottest selling item going right now. I am sure that by this time, sales on TIB are approaching, or have already surpassed 2000 copies. That's a fair piece of change! I don't know who is going to do this face lift, but it's for sure, somebody decided that the community is a lot different today than it was three years ago. That's when Artist came out. We buy good stuff now if we intend to use it, and that says a lot for todays TIer.

Anyway, off the soapbox... here is a list of stuff I would like to see. If you agree or disagree, let's have your opinions.

First of all, and most importantly, don't CHANGE too much, but very defiantly add to the program. Based on the modular approach that we are using with TELCO and Press, we can go ape as far as we want to with new functions.

The one change the program needs most to remove the case sensitivity from the I/Os. Nothing makes me crazier that to be moving stuff around and have to go in and out of the alpha lock.

Add some new "brushes". There are only a half dozen or so in there now. Take a look at Picasso or Joy-Paint and see all they have to offer.

Add more textures to work with.

How about a spray can effect. You could replace the "Point" icon very easily with that.

I would like to see two zoom stages, kind of like MY-Art. The first step could use a quarter of the screen, the second could be the same as it is now. I prefer the zoom mode of Picasso, but I doubt if that would be possible in Artist.

I would like to see a shape change for the circle and disk functions as Graphx has. Circles are nice, but it's not a perfect world, so there are more ellipses.

Make saving color in the picture file a selected option, or devise some automatic method of detecting color.

As much as I would like to see a larger screen area, I'm afraid that the large scale move and copy functions would

suffer. Check it out though. I find the small scale functions of this type in Joy-Paint totally unacceptable.

It needs better drivers for todays class of printers.

Could a light pen be designed for it that really worked??

Put an eight pixel eraser and a single pixel draw in the enhancement section.

Get rid of the Alpha-numeric feature in the drawing space, or, design it to load different one-high fonts... THAT'S ALL, no size changes. (What a waste of memory that is!!)

Make the whole thing memory image so that it can be loaded with anything, from anywhere.

DON'T change the instances or font loads. They are deeply rooted and we need them.

That's about all I can think of right now. If you have some other ideas, let me know. I intend to upload this article to TI-Source and I will be happy to add to it.

L8r... HTB

DISK OF THE MONTH

Diskname DOM:02/89 Total Sectors 358 Free Sectors 8

Filename	File Type	Size	Pro	Comment
!READ-ME!	D/V	80	4 Yes	Read this text file!
BOOT	PGM	6656	27 No	Main Menu
CHARA1	PGM	1030	6 Yes	Character file for BOOT
GROG/2	I/V	254	67 Yes	Maze of Grog, by Ray Kazmer
GROG/2LOAD	PGM	28	2 Yes	Loads GROG/2
GROG/DOCS	D/V	80	30 Yes	Docs for GROG/2
GROG/HS	D/V	80	2 No	High score file for GROG/2
KEYM-LOAD	PGM	30	2 Yes	Loads KEYMUSIC
KEYMUSIC	I/V	254	55 Yes	Keyboard Music by J. Brylinski
LOAD	PGM	798	5 Yes	Loads BOOT
LOADER	PGM	4751	20 Yes	Misc. loader
MANDEL-DOC	D/V	80	9 Yes	Docs for MANDELZOOM
MANDELZOOM	PGM	7937	33 Yes	Mandelzoom, part 1.
MANDELZOON	PGM	4652	20 Yes	" " , part 2.
MANDELZOOO	PGM	6144	25 Yes	" " , part 3.
PANORAMA_P	PGM	6144	25 Yes	Fractal picture for MANDELZOOM
PANORAMA_w	PGM	54	2 Yes	Data info for PANORAMA_P
Q20P	PGM	3803	16 Yes	20 Questions by T. Pellitieri

HOW DO I DO THAT?
HARRY'S DO-DO COLUMN

GOING FROM POINT "PRB" TO POINT "TIB"
or
"Converting the Club's database"

With the introduction of Vrs 2.01 of TI-Base, we can pull our records out of PR-Base and easily convert them to our bright new baby, TIB. This requires three programs, the one on either end (PRB and TIB) and a "C" language program called PRB-DF (that's PR-Base to Display Fixed). This program can be had from our library, or, if you like, you can download it from Texament's bulletin board.( ) I would recommend that you get the library version though since there are other TIB tutorials with it.

The first thing you should do is go into your PR-Base CREATE program. After CREATE is loaded, put your data disk (the one with the records on it) into the drive and bring up the report generator. The first screen you get after laid out your parameters is the one that shows all of your field sizes. This is the one you need to print out for TIB reference. It will look similar to FIG I at the end of the article.

Kill the report function with FCTN 9 when it has finished printing out, but don't drop out of PRB CREATE yet. It might also be a good idea to print out the screen format of your present data as well. This is done with the CREATE DATA SCREEN function. It will grab the screen it finds on the data disk assuming that you want to upgrade it. As soon as you have it comes up, press FCTN 6 and you will be able to print out the screen. It will look similar to FIG II.

Kill PRB, you're finished with it forever. Now, you must use the above mentioned PRB-DV program to convert your PRB records to the new format. This is an EA option 5 entry program so be prepared for that.

Since there are only about four inputs required from the user in this program, it's very easy to run. The main thing you should know is how many bytes your enter database uses. This can be determined by adding up all of the field-size counts from your PRB printout. There is one other question it asks that you may have trouble with also - You may/may not know, but, there is an upgraded version of PRB called 2.1 that Mike Dodd did for us. That one starts the records at sector 12 instead of 10 as the earlier versions did. The conversion program asks which sector to start with so check and see which version you have. I would also suggest that you use a clean disk to run the records to.

It's now time to hit up TI-Base in all it's glory. I'm going to assume that you have a nodding acquaintance with the general program by now, at least, enough to "turn it on".

After TIB is loaded, set up your status as you would like it and then type the command, CONVERT. The prompts will ask for the name of the file you wish to convert (the one you used in the TIB-DV program), and then a new name for the conversion.

You are safe in using the same name if you like because TIB will ad a "/D" suffix to it. At this point you are shifted to the CREATE function of TIB to set up your record fields.

This is very important... You MUST set up all of your field sizes exactly as they were in PR-Base, and, they must be in the same order. This is where your printouts from TIB become important. USE THEM!! To bring this home to you, you must understand what you are now dealing with, record wise.

When you set up your new file with PRB-DV you will get strings of text up to 255 characters long. (That's as many as you can have with PRB, 255 characters, or, one sector.) If you have a copy of WriterEase you can load this file into it and see that you have columns of information, perfectly alined. TIB must take this line of text and break it into the various fields you require. If you haven't set. the fields to their exact length, you will break it in the wrong spots. This could lead to the beginning characters of a field winding up as the trailing characters of the previous field. You could edit each record after it's into TIB but, that's a lot of work that can be avoided with a little accuracy.

Now, just so that I can keep you total confused, and dependent on this article, I will tell you about an exception to the rule.

Take a look at the PRB data screen, FIG 2. You will notice that the area under "JOINED" is broken into two fields, one for the month and one for the year. This is stupid! I did it years ago and I have no idea why. Anyway, I set this up in TIB as a single eight character field. (Notice that it represents two 4 character fields in PRB.) I got away with it because the two fields were right next to each other. I was also able to add a one hundred character "NOTE" field at the END of the record because there would be nothing to load into it anyway. You can play around as much as you want, just as long as you fully understand the way the record is set up.

After TIB is finished with the conversion, your next command must be "RECOVER". TIB will look through the new records and gather information such as the number of records in the file. This information is then sent to a statistics file with the suffix of "/S". If you don't do this, and try to open the file later, you will find that TIB will have no idea of what you're talking about.

Once all of the above is complete, you will have a working TIB file. Check it out by EDITing a few of the records. If you find that things are a little screwed up, and it's a big file I would suggest you start all over again with TIB CONVERT. This time make sure your fields are the right size.

As a final note, I want to suggest that you put TI-Base into your ram disk. The sort function is very slow on the outside from a floppy, and the program itself is slow to load... it's a whopper! I broke one of my rams into two equal segments and dumped the program into #7 and the data into #8. The improvement in speed was tremendous and the program is now a real joy to work with. Those of you that still don't own a ram had better think about it. I can guarantee you that most of the important software coming is going to be hard pressed to do without one.

FIGURE I

Report Format Design

#	Location	Size	#	Location	Size
1	242	10	17		
2	253	3	18		
3	257	5	19		
4	263	4	20		
5	267	4	21		
6	272	4	22		
7	333	24	23		
8	413	16	24		
9	434	3	25		
10	490	27	26		
11	567	17	27		
12	588	9	28		
13	650	8	29		
14	665	11	30		
15	730	14	31		
16			32		

FIGURE II

```
  0 01234567890123456789012345678901234567890
 40 **** WNY 99'ERS MEMBERSHIP BASE ****
 80
120 ****************************************
160 *NXT PY DAT*TYP* NUM * JOINED *EQUIP*
200 ****************************************
240 *[        ]*[ ]*[   ]*[  ][  ]*[  ] *
280 ****************************************
320 *LAST NAME *[                       ]*
360 ****************************************
400 *FIRST NAME*[                ]*INT*[ ]*
440 ****************************************
480 *ADDRESS*[                          ]*
520 ****************************************
560 *CITY*[                ]*ST*[        ]*
600 ****************************************
640 *COUNTRY*[          ]*ZIP*[          ]**
680 ****************************************
720 *PHONE #*[                ]*************
760 ****************************************
800
```

BINARY SEARCH TREES - An Application
by Thomas R. Pellitieri

There are many ways of organizing data in computers so as to access it in a reasonable manner. One such organization scheme is the Binary Search Tree. A Binary Search tree is a data structure which tends to divide the data into two pieces (binary) in order to locate (search) the data quickly. Each division may be further sub-divided, giving the tree structure.

The reason this data structure is called a "tree" is that the structure, when drawn, looks like a tree. It has a starting point, or root, with two branches. These branches can be labeled "Yes" and "No", or "Previous" and "Next", or whatever. As one follows the branches, each branch splits again into two. At the end, when there is no more splitting, the "Leaf Node" is reached, and that is the final answer.

Binary Search Trees are used in many applications, most of which involve some sort of decision making or maintaining ordered data. One such application is a game program called "Twenty Questions".

Most of you have probably played Twenty Questions with a friend: one person thinks of an object or person, and the other must guess it by asking questions which may be answered either "Yes" or "No". This game lends itself nicely to Binary Search Trees, since the list of objects gets divided into two by the "Yes" or "No" answer.

In the case of a question tree, the root is the first question to be asked. At each branch, there is another question. Each leaf is the name of a person. For each question, there is a Yes-Branch, which is followed if the player answers "Yes" to the question, and a No-Branch, which is followed if the player answers "No". After the branch has been followed, it is just like being at the root again, only further up in the tree. When you reach the leaf of the tree, there is nowhere else to go, and that is the answer that the computer will guess.

Similar trees can be used to alphabetize a list of words. If the word matches the word at the root, it is already in the tree. If not, either follow a "Backward" or "Forward" pointer to find the word in alphabetical order. If the word is not found in the tree, it is added at the appropriate point. I will cover this in more detail in a future article (I promise, Harry!)

The program which accompanies this article takes the part of the interrogator in a game of Twenty Questions where the player must think of a person. The computer will ask the questions that it has and follow its data structure to the end, where the name of a person is available as a guess. If the person on the list is the one that the player was thinking of, then the computer will congratulate itself. Otherwise, it will ask the player for information which will allow it to expand its knowledge.

Let's look at the program in detail. I will point out some of the programming "tricks" which are TI-specific as well as the details of the Binary Tree.

```
140 L9=1022 :: DIM Y(1023),N(1023)
150 DISPLAY AT(12,9)ERASE ALL:"20 QUESTIONS" :: DISPLAY AT(2
4,4):"PRESS ANY KEY TO GO ON"
160 CALL KEY(0,K,S):: IF S=0 THEN 160
170 DISPLAY AT(1,1)ERASE ALL:"This is the game of Twenty Qu
estions.  You will choose a famous person and I will try to
guess who it is."
180 DISPLAY AT(6,1):"If I guess correctly, we can play
again.
If I make a bad guess, then you will help meto learn by tel
ling me who"
190 DISPLAY AT(10,1):"you were thinking of and by giving me
a question that I may ask to distinguish my guess from you
r person!"
```

Line 140 sets a variable L9 to the effective size of the data base. This should always be an even number. You can increase the size by changing this line. Be aware, however, that disk space is also a limiting factor. I found that you can use up to 1135 entries if you start with a blank single-sided disk. This gives a grand total of 568 possible people in the computer's file, and should take no more than 10 questions to guess, provided the file is balanced. Line 140 also dimensions two arrays. Notice that the dimension is one more than the value of L9.

Line 150 displays a title screen, and line 160 waits for a key press. Lines 170-190 display the instructions. Please notice that the display string is very long. The TI-99 will automatically go to a new line after the first 28 characters are displayed, so multiple lines can be squeezed into a single DISPLAY AT statement.

```
200 DISPLAY AT(16,1):"What disk has the questions?" :: DISPL
AY AT(18,13):"DSK1" :: ACCEPT AT(18,16)VALIDATE(DIGIT)BEEP S
IZE(-1):D$
210 DISPLAY AT(20,5):"Create new files? N" :: ACCEPT AT(20,
24)VALIDATE("YyNn")SIZE(-1):A$
220 F$="DSK"&D$&".QUESTION" :: P$="DSK"&D$&".LINKS"
230 ON ERROR 9000 :: OPEN #1:F$,RELATIVE,DISPLAY ,UPDATE,FIX
ED 64
240 OPEN #2:P$,RELATIVE,DISPLAY ,UPDATE,FIXED 16
250 IF A$="N" OR A$="n" THEN 290
260 PRINT #2,REC 0,USING 8900:1,1
270 PRINT #2,REC 1,USING 8900:0,0
280 PRINT #1,REC 1:"George Washington"
290 INPUT #2,REC 0:S1,L1
300 FOR I=1 TO L1
310 INPUT #2,REC I:Y(I),N(I)
320 NEXT I
330 ON ERROR 9100
```

These lines determine (from the player) where the question file is located, and whether or not to create a new file. Notice the VALIDATE clauses in the ACCEPT AT statements, which allow us to avoid testing for invalid values. Notice also that we have a SIZE(-1) and a Default Response. This helps prevent

invalid inputs, and allows a user to quickly get into the program.

Line 220 builds the File Names by adding the disk number to the default names. The files are then opened, and if new files were requested, the program sets up with only one person: George Washington. Line 260 indicates that the first question is in record 1 and that there is only 1 record in the question file. Line 270 indicates that Record 1 is a leaf in the tree. This is indicated by the pair of zeros for "Yes" and "No" pointers. Lines 290-320 read the data from the file.

```
340 GOSUB 1000
350 DISPLAY AT(12,8)ERASE ALL:"Play Again? Y" :: ACCEPT AT(
12,21)VALIDATE("YNyn")SIZE(-1)BEEP:A$
360 IF A$="Y" OR A$="y" THEN 340 ELSE 9900
```

These lines call the "Play" subroutine (GOSUB 1000) and find out if the player wants to play again. If so, the program loops, otherwise it exits.

```
1000 REM PLAY THE GAME
1010 Q=0 :: S=S1 :: P=-1
1020 IF Y(S)=0 THEN 1100
1030 REM ASK A QUESTION...
1040 P=S
1050 GOSUB 2000
1060 IF A=1 THEN S=Y(S)ELSE S=N(S)
1070 GOTO 1020
```

This section is the main playing loop. Line 1010 initializes the tree pointers. Q is the number of questions, S is the current State, and P is the previous State, or "root". When the Yes Branch of the current state is 0, the leaf node has been reached. Otherwise, the Previous State is set to the current state and the question is asked (GOSUB 2000). The next state is determined by the answer, and the program loops.

```
1100 REM GUESS THE ANSWER
1110 GOSUB 2000
1120 IF A=1 THEN 1700
1130 IF L1<L9 THEN 1200
1140 DISPLAY AT(2,1)ERASE ALL:"You must be thinking of somep
erson I don't know! I can not remember anything more!"
1150 DISPLAY AT(6,4):"(My mind is too full!)" :: GOTO 1800
```

```
1700 REM CHEER FOR THE COMPUTER
1710 DISPLAY AT(18,1):"I guessed it correctly!"
1800 REM WAIT FOR A KEY PRESS
1810 DISPLAY AT(24,2):"Press any key to continue"
1820 CALL KEY(0,K,S):: IF S=0 THEN 1820
1830 RETURN
```

These lines ask a final guess and determine what to do afterwards. If the computer is correct, it goes to 1700 and waits for a key press. If the computer was incorrect and the

file limit (from line 140) is reached, a warning message is displayed and the computer waits for a key press. If the computer guesses incorrectly and there is room in the file, it goes on to the next block of code.

```
1200 REM LEARN A NEW PERSON
1210 DISPLAY AT(2,1)ERASE ALL:"You must be thinking of somep
erson I don't know about! What is that person's name?"
1220 DISPLAY AT(9,1):"(Use two lines if necessary)"
1230 ACCEPT AT(6,1)SIZE(28)BEEP:N$
1240 M$="" :: IF LEN(N$)>25 THEN ACCEPT AT(7,1)SIZE(28)BEEP:
M$
1250 N$=N$&RPT$(" ",28-LEN(N$))&M$ :: M=L1+1 :: Y(M)=0 ::N(
M)=0 :: PRINT #1,REC M:N$ :: PRINT #2,REC M,USING 8900:0,0
1260 DISPLAY AT(9,1):"Enter a question which may be used to
distinguish my guess from your person:"
1270 DISPLAY AT(16,1):"(Use two lines if necessary)"
1280 ACCEPT AT(13,1)SIZE(28)BEEP:R$
1290 T$="" :: IF LEN(R$)>25 THEN ACCEPT AT(14,1)SIZE(28)BEEP
:T$
1300 R$=R$&RPT$(" ",28-LEN(R$))&T$ :: IF POS(R$,"?",1)=0
THEN GOSUB 4000
1310 L1=L1+2 :: PRINT #1,REC L1:R$ :: DISPLAY AT(2,1)ERASE A
LL:"Pretend you are thinking of"
1320 GOSUB 3000 :: DISPLAY AT(3,1):N$&"."
1330 DISPLAY AT(6,1):"Please answer this question:"
1340 DISPLAY AT(8,1):R$ :: DISPLAY AT(11,10):"Answer:" :: AC
CEPT AT(11,18)SIZE(1)VALIDATE("YyNn"):A$
1350 IF A$="Y" OR A$="y" THEN Y(L1)=M :: N(L1)=S ELSE Y(L1)=
S :: N(L1)=M
1360 IF P=-1 THEN S1=L1 :: GOTO 1390
1370 IF Y(P)=S THEN Y(P)=L1 ELSE N(P)=L1
1380 PRINT #2,REC P,USING 8900:Y(P),N(P)
1390 PRINT #2,REC 0,USING 8900:S1,L1 :: PRINT #2,REC L1,USIN
G 8900:Y(L1),N(L1)
1400 RETURN
```

Time to add a new leaf to the tree. Lines 1210-1250 get the new name and add it to the files as a new leaf node (Y(M) and N(M) are 0). Lines 1260-1300 get a question to be used in future games to distinguish between the original guess and the new name. Lines 1310-1340 store the question and ask the player to answer the question. This answer is used in lines 1350-1390 to determine the correct paths for the two leaves.

If the player answers "Yes" to the new question, the Yes-branch must point to the new leaf, and the No-branch to the old one. If the answer is "No", the branches must go the other way. The record number in the file of the answer is placed in the appropriate spot (line 1350). If there was no previous question, this is the first question and the Starting Question number (S1) is updated. Otherwise, the appropriate branch of the previous question must be set to indicate the new question instead of the old answer. If the player answered "Yes" to the last question, the Yes-branch for that question will now point to this new question (line 1370-1380).

```
2000 REM ASK THE NEXT QUESTION
2010 Q=Q+1
```

```
2020 DISPLAY AT(2,1)ERASE ALL:"Question #";STR$(Q);":"
2030 IF Y(S)=0 THEN 2060
2040 LINPUT #1,REC S:Q$
2050 DISPLAY AT(5,1):Q$ :: GOTO 2090
2060 DISPLAY AT(5,1):"Are you thinking of"
2070 INPUT #1,REC S:Q$
2080 DISPLAY AT(6,1):Q$;"?"
2090 DISPLAY AT(11,10):"Answer: Y"
2100 ACCEPT AT(11,18)VALIDATE("YNyn")SIZE(-1):A$ :: A=0 :: I
F A$="Y" OR A$="y" THEN A=1
2110 RETURN
```

This is a subroutine which asks the next question and gets an answer. In line 2020, STR$(Q) is used to avoid the automatic spacing which BASIC includes around numeric output. Line 2030 decides if a question is to be asked, or a guess is to be made. Lines 2040-2050 copy the question from the file. Lines 2060-2080 do the same thing, but ask "Are you thinking of?" since they ask the final guess. Lines 2090-2100 ask and get the answer to the question.

```
3000 REM TRIM TRAILING BLANKS FROM N$
3010 N1=LEN(N$)
3020 IF SEG$(N$,N1,1)=" " THEN N1=N1-1 :: GOTO 3020
3030 N$=SEG$(N$,1,N1):: RETURN
```

These lines help out the input by trimming trailing blanks from N$.

```
4000 REM ADD A QUESTION MARK TO R$
4010 R1=LEN(R$)
4020 IF SEG$(R$,R1,1)=" " THEN R1=R1-1 :: GOTO 4020
4030 R$=SEG$(R$,1,R1)&"?" :: RETURN
```

These lines do the same thing as those at line 3000, except they add a question mark to the end of the question. Please notice that the actual questions are stored in the file with question marks, but the answers are stored without final punctuation.

```
9000 REM ERROR - ABORT
9010 IF A$="y" OR A$="Y" THEN 9030
9020 DISPLAY AT(22,1):"There are no files on that disk.  Pl
ease create them." :: GOTO 9040
9030 DISPLAY AT(22,1):"There is no disk available on that d
rive.  Try again."
9040 ON ERROR 9999
9050 GOTO 9900
9100 REM ERROR PROCESSING
9110 CALL ERR(Z1,Z2,Z3,Z4):: PRINT " ":"Error number";Z1;"/"
;Z2:"occurred at line";Z4
9120 GOTO 9040
9900 REM CLOSE FILES AND END
9910 CLOSE #1 :: CLOSE #2
9999 END
```

These are the error handlers and exit routines. Line 9000 is executed when the files cannot be opened. If the files were being created and we get here, it is assumed that an invalid disk drive was specified, otherwise it assumes that the files were not available on the specified disk. Line 9100 handles other errors. These should not occur. Line 9900-9999 close the two data files and exit the program.

A few words should be devoted here to artificial intelligence. This program can develop a rather sophisticated data base, but the player is the one who determines how sophisticated it is. With the proper "training", the computer can become a very good player. As stated earlier, there is room on a single-sided disk for 568 names. On the average, the computer should be able to guess a name by the eleventh question. This assumes that the questions and the names are "well-balanced".

Consider the following starting plays. Since the data base is brand new, question #1 will be "Are you thinking of George Washington?". Since you probably won't be, you will type in the name of the person you were thinking of. Let us assume that it is Martha Washington. A question is required to distinguish between Martha and George Washington. The BEST questions are always as general as possible! One possible question would be "Was he the Father of his Country?", while another question could be "Are you thinking of a man?". The first question is VERY BAD, since it divides the possible set of people into two groups such that George Washington is in a group by himself, and all other possible people are in the other group. The second question is very good, since it immediately limits the thought to either Men or Women.

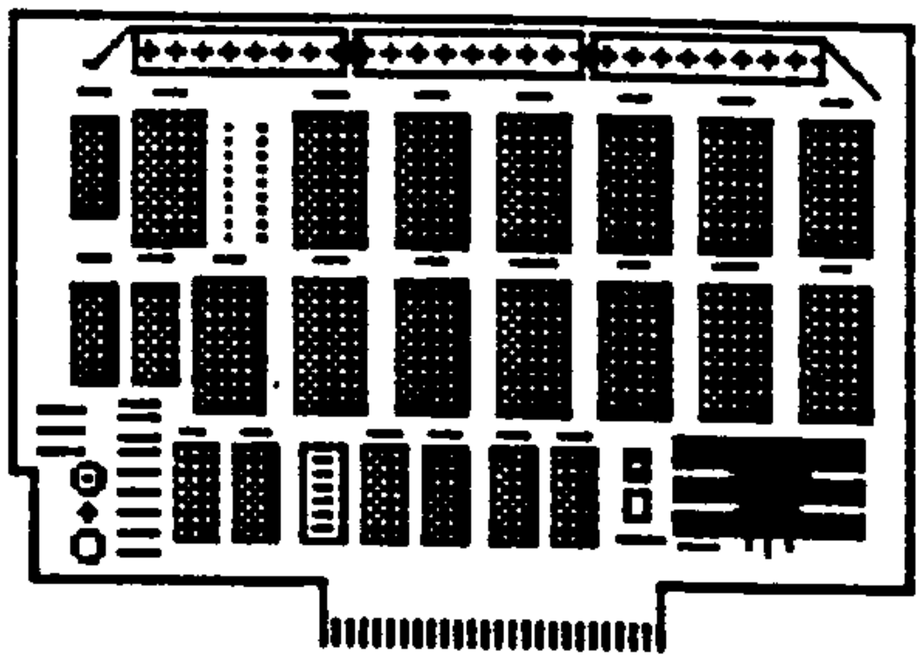
Note that the order in which names are added is also important. Try for the first few questions to come up with widely divergent answers. Think about how you would play the game and the questions you would ask early on. Then choose your first few names so as to have the computer ask those questions.

For myself, I like to ask "Are you thinking of a man?" as the first question, since it immediately divides the group of possible answers into two roughly equal parts. The next question I like on each side is "Did s/he live in the 20th century?", which breaks the group down basically into recently living or long since dead. My first three names tend to be Martha Washington, a living actress, and a living actor (e.g. Meryl Streep and Sylvester Stallone). This allows the computer to have a good pair of starting questions, and helps to balance the question tree.

Perfect balance is difficult to achieve, since you must keep track of every person in the tree, and know who they all will be before you start to get a completely balanced tree. If you can think of 30 famous people and start off by putting them in a balanced tree, your computer should seem fairly intelligent to the average person. Remember to keep the questions as general as possible, and the balance should be maintained for quite some time!

<30>

THE HORIZON WAY



There hasn't been a whole heck of a lot really new with horizon of late, but I would like to run over a couple of things that have been brought to my attention.}

The first thing is the running of Multiplan from the HRD. I had never been involved with this process before so when the subject came up, I had to call Bud Mills to get the following info.}

As you know, Multiplan looks for a disk name of TIMP to run from, but it can only look through four drives to find what it wants. Since most people call their ram drive #5, this creates a problem.}

You must be using CRU 1000, nothing below that, in order for the MP system to work properly. Make sure you have switch #1 turned on before setting up for TIMP.}

I want to make it VERY clear to you that my method of setting up may not be yours. I don't want anyone getting bent and telling me that this doesn't work. . . that you do it some other way! The configuration of my system is as follows: CorComp controller, 2-DSDD drives, 2-384K rams. . . that's the stuff that counts, but it's all optional. I am only letting you know because of the possible quarks this config. may be causing me. I also am using ROS and MENU Vrs 7.3, the last one made.}

1> Bring up the file named CFG on Vrs 7.3 of the John Johnson menu system.}

2> Using the edit mode (E), set the ram drive numbers to 3 and 4. It would probably be a good idea to load the ROS even if it's already there. I do that on GPs every time I need to reconfigure. MAKE SURE YOU TURN THE POWER-UP OFF! If you leave it on, you will get a "File Not Found" warning from the MP module.}

3> Using the configuration mode (C), set drive #3 to be named TIMP and format 360 sectors for the Multiplan program files.}

4> Set up drive 4 for your data files, naming it what ever you like and just format the remaining sectors for the data. (The number of remaining sectors will depend on the size of your ram.)}

5> Leave CFG and bring up DM1000. By the way, I recommend that you always use the DM1000 Vrs that came with you HRD system. Load all the files having to do with the Multiplan program into drive #3. Copy your data templets to drive #4. After this is finished, you can put whatever else you want to into drive #4, but keep the MENU on drive #3. If you need the MENU, you can just cut to BASIC and CALL MENU. This is real handy if you need to catalogue a disk or something.}

Again, DO NOT set the HRD for auto power up as this will get in the way of the MP files. If you don't use anything else but MP, then you can leave off the Menu system.}

At this point you should be all set to load MP from the cartridge. MP will not search through the floppies until it gets to #3 to load the start-up templet. This is because the cartridge looks for CRU 1000 first, as opposed to floppy drive #1.}

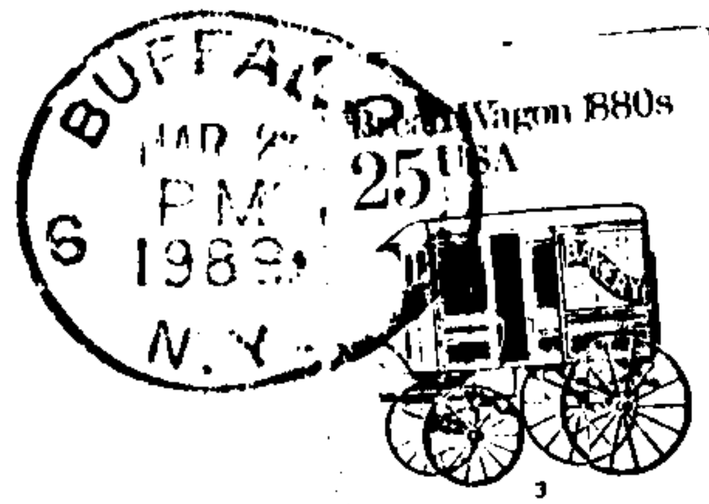
The next thing that must be done is to change the default file for your templets. As soon as MP is loaded, press (T) for Transfer, then (O) for Options, press CTRL A to go to system and type in "DSK4" then enter. This will allow all of your named data files to Save to, and Load from drive 4. That's all there is to it.}

As a final note - I would NEVER leave important MP data files on the ram without backup. No matter how reliable I feel the HRD is, IT DOES GO DOWN, and when it does. . . you lose. Leave your work in for the next time by all means, but always, always back it up to a floppy when you're finished. It's the smart thing to do.}

You can also use the above method to put PR-Base on your ram, but you MUST use the load program, not go directly to the memory image files. I still haven't figured out a way to put the PR data in the ram though.}

Speaking of databases, I have been talking to a few people about TI-BASE in the HRD. From what I have been told, using it for the data files creates an, almost, "no wait" situation for file access. I'll be working on this and will let you know more about it later.}

WNYDOER
2753 MAIN ST
NEWFANE NY
14108



DEFEAT
MU SCULAR DYSTROPHY
SUPPORT MDAA

TO:

***SMAUG/99
RT 4, BOX 23
BREWTON, AL
36426

FIRST CLASS MAIL