LONG ISLAND 99ER USERS GROUP

VOL #12 NO.03 MAR, 1993 $2.00

MEETING DATE: MAR 12, 1993

LONG ISLAND SOUND

EDITOR: FRANCIS J. BUBENIK JR.

HAPPY ST. PATRICK'S DAY

ESTABLISHED APRIL-1983

* 1993 FAIR SCHEDULE *

Compiled by FranK J. BubeniK Jr. (NL Editor)

LONG ISLAND SHOW MARCH 7TH

MAR 7, 1993 (SUN) HOFSTRA UNIVERSITY. HEMPSTEAD BLVD MEADOWBROOK PKWY TO M-4 - 1/2 MILE PAST NASSAU COLISEUM TURN RIGHT INTO HOFTRA UNIVERSITY FITNESS CENTER. HEMPSTEAD, NY. $8.00. 10AM/3PM. 500 TABLES. 200 DEALERS. KGP.

COME TO THE FAIRS.

TI APR 17, 1993 (SAT) BOSTON-NORTHEAST COMPUTER FAIR. WALTHAM HIGH SCHOOL, WALTHAM, MA. BOSTON COMPUTER SOC. TI99/4A UG. Contact: RON WILLIAMS, 14 EAST ST. AVON, MA. 02322. OR MIKE FRANCIS (617) 965-5653. *ONLY EAST COAST TI SHOW*. ROUTE 128, TOTTON POND ROAD FOR 1.5 MILES EAST TO LEXINGTON STREET, THEN .5 MILES NORTH TO WALTHAM H.S. 10AM/4PM. $3.00 ADMISSION.

TI MAY 14/15, 1993 (FRI/SAT) LIMA MULTI USER GROUP CONFERENCE. OHIO STATE UNIVERSITY LIMA CAMPUS. INFO:CALL CHARLES GOOD. (419) 667-3131 EVENINGS. WRITE: LIMA 99/4A UG. P.O.BOX 647,VENEDOCIA, OH 45894.

TI NOV 12/13, 1993 (FRI/SAT) *11TH ANNUAL CHICAGO INTERNATIONAL FAIRE*. ELK GROVE HOLIDAY INN.HOTEL RES. 708) 437-6010 CODE "IWF". JIM DEARDS NEW CHAIRPERSON.

TI NOV 14, 1993 (SUN) MILWAUKEE TI FAIRE. INFO CALL GENE HITZ (414) 535-0133.

LITI 99ERS NEWSLETTER IS NOT RESPONCIBLE FOR CANCEL-ATIONS. CALL THE NUMBERS BELOW TO VERIFY TIME AND DATES BEFORE YOU GO.

* Ken Gordon Productions (KGP) SHOWS COST $8.00- $1.00 discount cards are sent to those people on there mailing lists. Call (800)631-0062 OR (908)297-2526 for info. (rev 2/22/92).

* Tri-State Computer Fairs (TSCF) SHOWS COST $6.00- $1.00 discount cards available by mail. Call Robert Barlow (201) 533-1991 for info. (rev 1/7/93).

Don't miss the bargains on computers, software, IC's, peripherals, printers, monitors, parts, supplies and books.

XB MISCELLANEOUS #22
BY EARL RAGUSE

ELEMEMTARY XB PROGRAMMING (CONT) THIS ARTICLE IS 5 IN A 8 PART SERIES THAT STARTED WITH XB MISCELLANY #18. IF YOU ARE A NEW READER, AND DO NOT HAVE ACCESS TO REST OF THIS SERIES, PLEASE CONTACT ME OR THE NEWSLETTER EDITOR.

THE OPPOSITE PAGE SHOWS THE ENTRY MODULE, THE SAVE MODULE, AND THE DISPLAY MODULE. IT DOES NO GOOD TO ENTER STUFF IF WE CAN'T SAVE IT, ITS ALSO NICE TO BE ABLE TO SEE WHAT WE HAVE DONE. ALSO AT THE END IS THE DISPLAY SUBROUTINE, WE NEED ALL OF THESE TO DO WHAT WE PLAN TODAY.

THE INPUT MODULE LINE 1100, STARTS WITH THE USUAL CLPUT STATEMENT OF THE MODULE PURPOSE. THEN 1110 USES, DISPLAY AT TO PROVIDE SOME INSTRUCTIONS.

LINE 1120 IS A CONDITIONAL, IF YOU HAVE LOADED A FILE, YOU HAVE SOME RECORDSIN MEMORY, HENCE N HAS A VALUE. LINES 1130-1140 CONTINUE WITH THE INFO, AND GET AN INPUT FROM YOU VIA CALL GKEY(Q,ROW), SEE LINE 4000, XB MISC #21. NOTE THAT IT USES CALL KEY(3,K,S). THIS FORCES ALL INPUTS TO BE UPPERCASE. IT ALSO DISPLAYS WHAT YOU SELECTED ON ROW 24 SO YOU CAN'T ARGUE WITH IT. THE KEYPRESS IS RETURNED AS THE ASCII VALUE, IN Q.

LINES 1150-55 PROVIDE MORE INFO AND ANOTHER CHOICE. IF YOU WANT AN APPEND FILE, THE YES CHOICE IS STORED IN APP, ELSE RECORD NUMBER N=0.

LINES 1160-1180 PERFORM THE ACTUAL DATA ENTRY. 1160 COMPUTES THE RECORD NUMBER, IT USES CALL CLS(4,12) TO CLEAR PART OF THE SCREEN, SEE LINE 4300, FROM LAST TIME, THEN USES A FOR - NEXT LOOP TO COMPUTE AND DISPLAY THE LINE NUMBERS, USING THE INDEX TO POSITION THE DISPLAY. IT ALSO PRINTS THE RECORD NUMBER. 1180 HAS A FOR - NEXT LOOP THAT DOES THE ACCEPT AT, NOTICE THE SIZE(-24). ALSO NOTE HOW THE LOOP INDEX I IS USED TO PUT EACH LINE OPPOSITE THE NUMBER SET UP IN 1160.

LINE 1190 PUTs UP A PROMPT, GO CHECKS WHAT WE WANT, AND IF WE ACTUALLY PRESSED AN ALLOWED KEY (IF Q=0 THEN 1190) AND ON TO BRANCH TO THE RIGHT PART OF THE PROGRAM.

LINE 1300-1390 IS THE SAVE MODULE. LINE 1310 CHECKS N TO SEE IF WE, IN FACT, HAVE ANY DATA IN MEMORY, AND IF NOT, GIVES US A NASTY PROMPT, AND WE GO TO 1390 PAK, AND ULTIMATELY RETURN TO MENU.

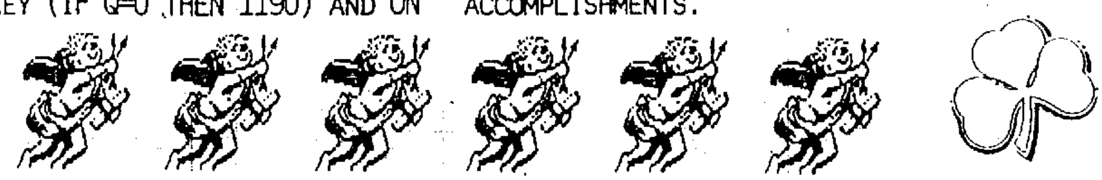
I AM GOING TO PASS UP EXPLAINING THE SAVIT SUBPROGRAM NOW, IT WORKS. TRUST ME.

LINES 1200-1290 ARE THE DISPLAY MODULE. THE FIRST TWO LINES 1200-1205 ARE OLD HAT, RIGHT? LINES 1210-1215 USE CLPUT, PUT AND ACCEPT AT TO PROMPT AND GET A VALUE FOR J, THE RECORD NUMBER OF INTEREST. IF J<>0, WE ARE OFF TO 1280 TO DISPLAY IT, ELSE WE MAKE J=1, SO WE CAN SEE THEM ALL.

LINES 1220-1230 PARTIALLY CLEAR THE SCREEN, GOSUB 2100, THE DISPLAY ROUTINE, DISPLAYS THE SELECTED RECORD, 1230 PUTs UP A PROMPT (NEXT LAST ALL QUIT), 1240 GO CHECKS, AND ON SELECTS THE RIGHT COURSE OF ACTION. IF WE SELECT NEXT, WE GO TO 1250, AND DIR=1, IF WE SELECT EITHER CASE WE GO ON TO 1265 WHERE J=J+DIR. DO YOU SEE THAT IS EQUIVALENT TO FWD AND REV? J IS CHECKED IN LINE 1265 TO SEE IF IT IS WITHIN LIMITS OF 1 TO N, IF NOT THEN, OFF TO LINE 1170 WHERE WE GET "THAT'S ALL", PAK AND RETURN TO MENU (140). IF WE HAD SELECTED ALL WE GO TO LINE 1275, SET PAR=1 AND ON TO 1280 TO DISPLAY THE RECORD J. IF WE HAD SELECTED ALL, AND THUS PAR=1, THEN CALL WAIT (400) IS EXECUTED TO GIVE US TIME TO READ THE RECORD, THEN GOTO 1210 AND START OVER AGAIN WITH A NEW RECORD NUMBER.

LET'S TAKE A CRACK AT SAVIT, IT EXPECTS FIL$,REC$(,),N,APP) IN THE CALL STATEMENT. IT WILL RETURN VALUES IN ALL OF THESE VARIABLES. SAVIT IS MOSTLY OLD STUFF, IT USES PUT, CLS, AND GKEY TO GET THE DRIVE NUMBER IN D$. IT USES DISPLAY AT, AND ACCEPT AT TO GET FIL$ (FILENAME), I HAVE TALKED NO END ABOUT FILE OPENING STATEMENTS, SO I WILL SAY NO MORE. IF YOU NEED HELP CALL ME.

LINE 4150 USES A DOUBLE FOR - NEXT LOOP TO WRITE (PRINT) ALL THE RCORDS TO THE DISK. LINE 4160 JUST ADVERTISES ITS ACCOMPLISHMENTS.



THIS IS THE ENTRY MODULE

```
1100 CALL CLPUT("DATA ENTRY
SECTION",2)
1110 DISPLAY AT(4,1):"YOU MA
Y HAVE UP TO 5 LINES  PER RE
CORD. USE ARROW KEYS, AND TH
E ENTER KEY TO ACCEPT."
1120 IF N THEN 1150 ELSE DIS
PLAY AT(10,1):"IF YOU INTEND
 TO APPEND      RECORDS TO AN
 EXISTING FILE,YOU MUST FIRST
 LOAD THAT    FILE."
1130 CALL PUT("IF THIS IS AN
 APPEND FILE,    ",16):: CALL
PUT("PRESS A, ELSE PRESS ANY
 KEY  ",18)
1140 CALL GKEY(Q,24):: IF Q=
ASC("A")THEN APP=1 :: GOTO 1
400 :: ELSE APP=0
1150 CALL PUT("DATA ENTRY SE
CTION",2):: DISPLAY AT(8,1):
"YOU WILL OVER WRITE EXISTIN
GRECORDS, IF THAT IS NOT WHA
T"
1155 DISPLAY AT(10,1):"YOU W
ANT, PRESS A FOR APPEND : :"
PRESS A, ELSE PRESS ANY KEY"
:: CALL GKEY(Q,24):: IF Q=AS
C("A")THEN APP=1 ELSE N=0
1160 N=N+1 :: CALL CLS(7,12)
:: DISPLAY AT(8,7):"RECORD N
UMBER";N :: FOR I=1 TO 5 ::
DISPLAY AT(I+9,1):I :: NEXT
I :: CALL KEY(5,K,S)
1180 CALL KEY(5,K,S):: FOR I
=1 TO 5 :: ACCEPT AT(I+9,3)S
IZE(-24):REC$(N,I):: NEXT I
1190 CALL PUT("EDIT  MORE  Q
UIT",24):: CALL GO("EMQ",Q):
: IF Q=0 THEN 1185 ELSE ON Q
 GOTO 1180,1160,140
```

THIS IS THE SAVE MODULE

```
1300 CALL CLPUT("SAVE DATA S
ECTION",2)
1310 IF N<1 THEN CALL PUT("Y
OU HAVE NO DATA TO SAVE",12)
:: GOTO 1390
1320 CALL SAVIT(FIL$,REC$(,)
,N,APP)
1390 CALL PAK :: GOTO 140
```

THIS IS SAVIT

```
4100 SUB SAVIT(FIL$,VAR$(,),
N,APP)
4110 CALL PUT("WHAT DRIVE#",
12):: CALL GKEY(Q,24):: IF (
Q-48)<1 THEN 4110 ELSE D$=CH
R$(Q):: CALL PUT("WHAT FILE
NAME",14)
4120 DISPLAY AT(16,10):FIL$
4130 ACCEPT AT(16,10)SIZE(-1
0);FIL$ :: OPEN #3:"DSK"&D$&
"."&FIL$,OUTPUT,DISPLAY,VARI
ABLE 80
4140 !PRINT #3:N
4150 FOR J=1 TO N :: PRINT #
3:CHR$(32):: FOR K=1 TO 5 ::
 DISPLAY AT(12,6):"SAVING RE
CORD";J :: PRINT #3:VAR$(J,K
) :: NEXT K :: NEXT J :: CLO
SE #3 :: N=J-1
4160 DISPLAY AT(12,1)ERASE A
LL:"    I SAVED";N;"RECORDS"
 :: SUBEND
```

THIS IS THE DISPLAY MODULE

```
1200 CALL CLPUT("DATA DISPLA
Y SECTION",2)
1205 IF N<1 THEN CALL PUT("Y
OU HAVE NO DATA TO DISPLAY",
12):: GOTO 1290
1210 CALL CLPUT("IF YOU KNOW
 THE RECORD",8):: CALL PUT("
NUMBER, ENTER THAT NUMBE
R",10):: CALL PUT("ELSE ENTE
R 00",12):: ACCEPT AT(14,13)
:J
1215 IF J<>0 THEN 1280 ELSE
J=1 :: PAR=0
1220 CALL CLS(6,20):: GOSUB
2100 :: GOTO 1230 ! DISPLAY
RECORD
1230 CALL PUT("PRESS FIRST L
ETTER TO",23):: CALL PUT("NE
XT  LAST  ALL  QUIT",24)
1240 CALL GO("NLAQ",Q):: IF
Q=0 THEN 1240 ELSE ON Q GOTO
 1250,1260,1275,140
1250 DIR=1 :: GOTO 1265
1260 DIR=-1 :: GOTO 1265
1265 J=J+DIR :: IF J<1 OR J>
N THEN 1270 ELSE IF PAR THEN
 1280 ELSE 1220
1270 CALL CLS(22,23):: CALL
PUT("THAT'S ALL",22):: CALL
PAK :: GOTO 140
1275 PAR=1 ! PRINT ALL RECOR
DS
1280 IF J>N THEN 1270 :: CAL
L CLS(6,20):: GOSUB 2100 ::
IF PAR THEN CALL WAIT(40
0):: GOTO 1265 ELSE CALL PAK
 :: GOTO 1210
1290 CALL PAK :: GOTO 140
```

THE DISPLAY SUBROUTINE

```
2000 !
2010 ! SUBROUTINE AREA
2020 !
2100 DISPLAY AT(7,10):"RECOR
D";J :: FOR K=1 TO 5 :: DISP
LAY AT(K+9,1):K;REC$(J,K
):: NEXT K :: RETURN
```

What is AMS, and Why is it Here?
A monologue by Chris Bobbitt

The Introduction

A wise man once said true wisdom can be attained by learning from your mistakes. Considering the number of mistakes I've made over the years, at least I can say I've had ample opportunity to become wise.

The largest "mistake" I made in the 10 years Asgard Software has been in existence was Press. It wasn't a complete mistake - I'm not going to apologize for the vision Charles Earl and I shared of what a word processor should be. I will readily admit, however, that the way we went about developing it and marketing it was all wrong.

Lots of little mistakes became apparent in my quest to discover What Went Wrong. Soul-searching aside, one of the biggest reasons was actually technical. Contemplation of the technical problem led to both a fundamental realization of, and appreciation for, The Problem.

The Problem

Press was designed to be the ultimate in modular code - the program would literally load individual functions into memory as needed, and reuse the space when the functions were no longer needed. In this way, it had more in common with mainframe programs than software for home computers. This level of modularity is what would have made Press possible - and without it Press was impossible.

As you may have already guessed, the fact we couldn't get this "Memory Manager" to work was the reason that Press didn't work. A large part of this was due to the fact that memory is scarce on the 99/4A. In order to work, Press needed all the memory it could get - so the Memory Manager was written to take advantage of all sorts of other types of memory: supercarts, the Mini-Memory and even some RAM-disks (such as a Rambo equipped Horizon RAM disk). While this was fine in theory, in practice it was a mess.

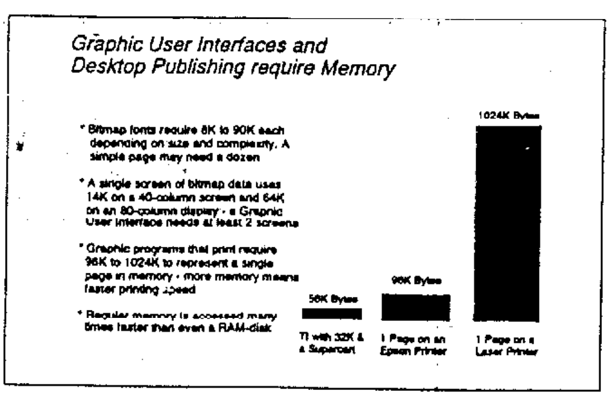
Because of the complexity of using some of these devices as memory for programs and data, more code was devoted to accessing memory beyond the standard 32K than all of the other code in the Memory Manager combined. Because only a small number of features could be implemented on a standard 32K 99/4A, and taking advantage of memory beyond the 32K resulted only in reams of buggy program code, Press collapsed under its own weight.

While it would have been possible to make a version of the program that ran in the standard 32K, we had promised the program would do much more, and neither Charles nor I really wanted to release a program that only did half of what we told everyone it would do for 2 years - so the project died, and The Problem first slapped us in the face:

"The 99/4A desperately needs more memory accessible to programs"

Our experience proved that the only real program-accessible memory for the 99/4A is the 32K card, and to a degree a supercart/Mini-Memory. You have to face it, while 32K was a lot of memory in 1979, not a single PC program today will run in it. Heck, 64K became standard in 1982 when the Commodore 64 was released. The average PC or Mac sold today is equipped with 4096K of RAM. The average PC word processor requires 640K to run minimally, and the next generation of word processors will no longer run in less than 1024K.

To illustrate how memory requirements have ballooned in the last decade, take a look at the following chart:



Desktop publishing was chosen as an example because, with the advent of programs like Page Pro 99, it has become one of the most popular things the 99/4A is used for. As someone who develops graphics software, I've been painfully aware of the 99/4A's memory limitations for years - especially as we've tried to expand the capabilities of our programs.

Why do 99/4A programs need extra memory? Very simply: to allow more data and more program functions to be in memory at once.

What is the difference between extra memory and a RAM-disk? While this is discussed in more depth below, the answer is also simple: RAM-disks are really only faster disk drives. *While they can hold programs and data, this information is only accessible when a program physically loads a piece of it from the disk. With more real memory, the program itself can be much larger, and more data can be stored in memory where it is accessed quickly before it has to go to the relatively slower disk drive.* As fast as they are, RAM-disks are slow compared to storing data in memory. More memory also allows programmers to write large programs that contain lots of data (such as a Graphics User Interface) without frequently referring to a disk drive.

In essence, more memory is needed so that more complex things can be done - and a RAM-disk is only a partial substitute at best for more memory.

So, now that we knew The Problem (the lack of real memory for 99/4A programs), The Search was on for The Solution.

The Search

After discovering The Problem we decided to evaluate all of the memory devices available for the TI-99/4A to see if any of them were The Solution. Our experience helped us to define the criteria any memory system should meet before it was usable as "real memory":

1. It had to offer memory within the normal programming area. This is so that it would be easier to adapt existing programs to take advantage of it.
2. It had to offer memory in usable "chunks" that were large enough to store a significant amount of program code and data, yet small enough where a single bank of RAM didn't take all of the standard memory area.
3. It had to offer a lot more memory than the standard 32K - the more the better.
4. It had to be invisible to, or at least not conflict with standard hardware and software.
5. It had to usable by average programmers - and not just super hackers.
6. It had to be inexpensive.

In our opinion, the ideal memory system would also:

7. Be invisible to the programmer - he or she would simply write a large program and let the memory card figure out how to fit it into memory.

To make a long story short - none of the memory devices on the market met these conditions. Supercarts and the Mini-Memory were limited to a certain amount of memory at a certain location. The only device that even offered a glimmer of the kind of memory needed was the Rambo, and it was both inflexible and a programmers nightmare to work with. All other RAM-disks also failed on one or more points.

After an exhaustive search, we decided that if we wanted to write more sophisticated software, we had no choice but to build a device that offered the capabilities we needed.

Using the Geneve and the un-released TI-99/8 as models, which both could be expanded to 2048K, Asgard Peripherals (the hardware division of Asgard Software) began a two-year odyssey of exploration, frustrating dead-ends, and back-tracking before we were able to construct a memory system that met the six criteria above, and could (with some work) even meet the last condition - a memory system that was invisible to the programmer.

The Solution?!

It's funny how sometimes the hardest questions have the simplest answers. In searching for a solution we almost entirely re-invented the wheel before we realized that TI had already done it for us. We discovered that they had built, and continued to produce, a single chip that did most of the work of expanding the memory of the 99/4A.

Variations of this device (known as a "Memory Mapper") are found in virtually every 9900-family computer product TI ever built and sold, with the exception of the 99/4A. A variation is even used in the Geneve and the TI-99/8.

The 99610 memory mapper (its original designation) is elegant in its simplicity. It takes the 16-bit address space of the 9900 processor and turns it into a 24-bit address space. In other words, it makes the 9900 think it has up to 16Mb (or 16384K) of memory instead of 64K. It does this by allowing a programmer to put 4K banks (or blocks) of memory anywhere within the normal address space of the 9900 processor.

The jist of this is that at a stroke this single chip met all of our first six conditions.

Besides offering memory within the normal programming space, it also offers it in 4K blocks that are easily manipulated. Further, it turns out, most software written by TI was also designed to work with blocks of the same size - and so it would be a lot easier to adapt existing TI software (Extended BASIC, etc.) to take advantage of mapper memory than any other kind of memory.

The mapper obviously allows for a lot more memory than 32K, but just as importantly, to the computer a memory card using the mapper is no more non-standard than a 32K card - and won't conflict with any device except those that try to provide 32K to the computer.

A final advantage of the mapper is that, for programmers, it is about the simplest way to use memory beyond 32K. Assembly programmers only need a few lines of code and other programmers a single command to push blocks of memory in and out of the 32K area - or even potentially have it done for them automatically (depending on the language).

After discovering the mapper it was only a matter of time before we built a prototype, the AMS, that would provide up to 512K of RAM to the 99/4A accessible through the mapper.

What is the Asgard Memory System?

- A family of fully compatible memory cards for the Peripheral Expansion Box
- Fully compatible with all existing TI software - works as a 32K card to regular programs
- Transparent to all disk controllers and some RAM-disks, as well as all other devices
- Provides memory to 99/4A in 4K banks - exactly what TI specified for the 99/4A & 99/8
- For AMS aware programs, provides additional memory on demand up to 16Mb in a few machine cycles
- Easy to adapt existing software and languages to take advantage of it - if it works with a Supercart it can work with AMS

AMS 128K-512K
Available now

AMS 4Mb-16Mb
Available Q3 '93

The AMS also met our sixth criteria - it is a cheap way to add 128K to 512K of usable program space to the 99/4A. Considering an 8K supercart costs $25 or so, $120 for 16 times as much RAM is a bargain. The AMS also allowed us to prove our concept, and get the technology embodied by the mapper quickly into the hands of programmers so that they could become familiar with the technology. Further, you can do real things for the 99/4A, even with "only" 128K.

Finally, the AMS has allowed us to begin work on a software system that brings our last goal - a memory system that is transparent to programmers - within reach. Several programming languages in development will take advantage of the memory without the programmer specifically writing program code to do so.

The Comparisons

Inevitably, there will be comparisons between AMS and other memory systems. While you can compare AMS to other devices in terms of the amount of memory, it is impossible to compare it in terms of how the memory is used. The only memory device that comes close is a supercart. Like a supercart, the AMS provides real memory for programs, and not disk storage. Unlike a supercart, it offers much more than 8K of RAM.

Comparing the AMS to a RAM-disk, as mentioned above, is like comparing apples and oranges. RAM-disks are "solid-state disk drives", meaning memory chips that are controlled by software to emulate a disk drive. The AMS, by comparison, is memory that can be directly used by programs to store data or program code which can be used or acted on without "loading" it first.

To illustrate the difference between RAM-disk memory and AMS memory, a good example of would be a database program. With TI-Base, for instance, if you have a 2000 record database you may be able to store (perhaps) 50 records in memory at once. Whenever TI-Base sorts the database, it has to physically load each of the 2000 records, 50 at a time, into the same space in memory. It creates an "index" of those records, which it then saves to disk after it has completed loading all of those records. This "index" tells the program the order the 2000 records are in. With a database like Personal Record Keeping, you would be limited to maybe 100 records altogether, since it doesn't allow you to have more records than you can fit into memory.

If you had a database designed to work with AMS, *all 2000 records could be in memory at once*. A sort would be instantaneous as the records could be put into sorted order within memory - or an index built in memory without loading anything first. The same database could be sorted in a tiny fraction of the time, and searching the database would be instantaneous. This same comparison would apply in any situation where you have a lot of information to store - a word processor, graphics program, etc. Because all of the data physically resides in memory, it can be accessed and used much faster.

In addition to providing memory for data, AMS also provides memory for larger programs. Because all of the program can be loaded into memory at once, managing memory with AMS is less difficult than loading parts of the program from disk - and takes much less code. The AMS memory card functions almost exactly like memory beyond 640K does on PC compatibles. Programming a 99/4A equipped with the AMS (or its sibling) is no more difficult than writing PC programs larger than 640K.

While technically, AMS is fundamentally different from other cards - some people have been tempted to compare it with other memory systems on the basis of the amount of memory offered. While the AMS offers as much memory as some other memory systems (the Foundation has been mentioned), again, there is no comparison. *The AMS is only memory card for the 99/4A that offers a memory mapper - and a memory mapper is the only way to truly expand the amount of memory available for programs and data on the TI-99/4A.*

The Future

The AMS is currently in the classic "chicken-before-the-egg" dilemma. If you were to buy one and plug it in, it only works as a 32K card until you run a program designed to use it. Why would anyone want to buy a card where there is currently no software designed for it? Conversely, while would anyone write a program for a device nobody owns?

These are legitimate questions, and they deserve honest answers.

From a programmer's perspective, if AMS was just another RAM-disk for the 99/4A, I would agree. However, the AMS is a real technical breakthrough in 99/4A memory expansion. We have good reason to believe that every programmer that has hit the 32K memory barrier in the past is (and if they aren't, should be) interested in writing programs for AMS. There are really no other options if you want to make larger and more powerful programs.

For programmers, the AMS is the first and only true memory-mapped memory device for the 99/4A. *There is no easier way to write programs larger than 32K.* Further, the AMS was designed to the only real standard that ever existed for expanding 99/4A memory - the one TI specified for the unreleased TI-99/8. The AMS provides memory to the 99/4A the same way memory is provided to the 99/8. The AMS is the only true memory expansion system for the 99/4A aside from a 32K card or a supercart.

While we can't make programmers take advantage of the card, its potential to make so many projects that were impossible in the past possible, or even easy, will interest any programmer who wants to make better programs.

From an average person's standpoint, the issue is more complicated.

AMS was designed and implemented by a software company. Therefore, its guaranteed support from at least one software company - one that is in the process of writing a considerable amount of software designed to take advantage of it. While initially most of our software will be AMS versions of our other products, these versions will not only be faster and more capable, but ultimately, may evolve into much bigger, more powerful programs. Further, programming languages designed to make writing software for AMS even easier are also in development - including a full assembler package and a new Extended BASIC cartridge. While we can't speak for other software developers, virtually every product in our future will take advantage of AMS one way or another.

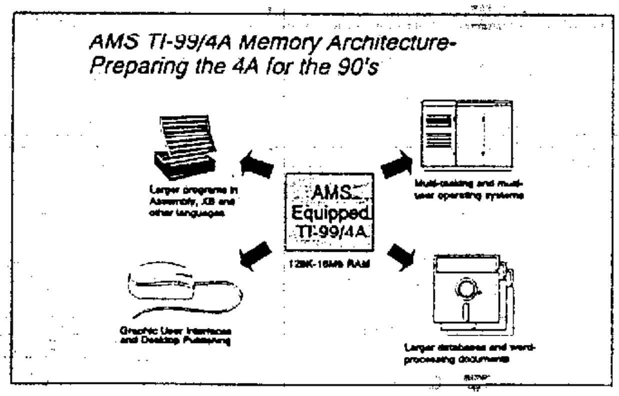
Also as a software company, we have a better idea of what programmers need in order to write programs for the AMS. In order to allow other developers to work with AMS, we've placed all programming specifications and source code into the public domain, and we are providing AMS systems with documentation at cost to any programmer that wants one. We already have mailed documentation to most prominent TI software developers.

From the point of view of a customer, the AMS is being produced by a company with an almost 10-year track record in the TI community - and a solid record of keeping our promises (well, all but 1 or 2 of them). We are in it for the long run, and so we will do whatever we can to support and develop this technology.

Examples? We are currently working on the next generation AMS card which allows 1024K to 16384K of memory for the 99/4A - potentially 8 times the memory of the Geneve. A wide array of development software will be included with the device - which will be fully compatible with its predecessor. To protect buyers of the current card - they will be able to trade it in on the next card when its available.

Finally, we can offer one last form of "buyer protection". We are committed to making this memory device the standard extended memory system for the 99/4A. Asgard intends to license the design for a low, one-time fee to any company, user group or individual that would like to make AMS-compatible cards. *We designed this card because we needed it to do the software we wanted to do - not to get into the hardware business.* We are negotiating with several people right now, and if all goes well, in the future you should be able to buy compatible cards from a number of vendors.

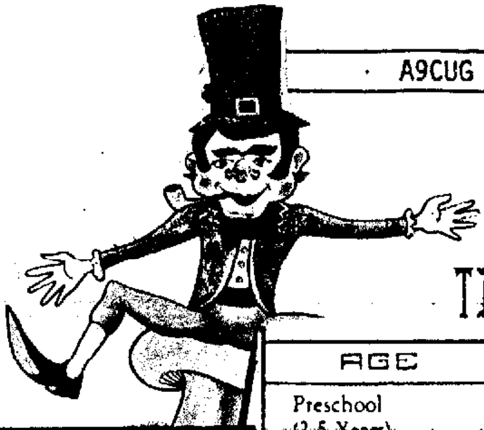
If all of these reasons don't convince you, the best reason of all is the potential of what can be done with more memory on the 99/4A. More memory will open up a wide variety of new applications that are difficult to impossible on the 99/4A:



As the chart illustrates, AMS makes lots of things possible that weren't before, including:

- Multi-tasking and multi-user operating systems
- Full-scale business packages
- Graphical user interfaces and true desktop publishing
- Full-size word processors, databases and spreadsheets
- Modern, full-size programming languages such as C and ever more capable Extended BASICs
- Great advances in graphics, speech and music software - including multi-media, digitizing, fax software, and more

All in all, AMS will let the 99/4A live up to its full potential as a computer by eliminating its most serious problem - the lack of memory available for programs and data.



HAPPY ST. PATRICK'S DAY TO ALL!

SELECTION GUIDE FOR TI EDUCATIONAL SOFTWARE

AGE	SUBJECT	PROGRAM
Preschool (2-5 Years)	Early Learning	Early Learning Fun Early Logo Learning Fun
Early Elementary (5-7 Years)	Reading	Early Reading Reading Fun
	Spelling	Hangman
	Math	Number Magic Addition/Subtraction I Addition/Subtraction II Numeration I
	Art	Video Graphs
Middle Elementary (8-9 Years)	Reading	Beginning Grammar Reading On Reading Roundup
	Spelling	Scholastic Spelling Levels 3 & 4
	Math	Multiplication I Meteor Multiplication Division I Alligator Mix Minus Mission Alien Addition
Late Elementary (10-12 Years)	Reading	Reading Flight Reading Rally
	Spelling	Scholastic Spelling Levels 5 & 6
	Math	Demolition Division Dragon Mix Numeration II
	Music	Music Skills Trainer Computer Music Box
Early Elementary to Junior High (5-14 Years)	Math	Addison-Wesley Computer Math Games II, III, IV, VI
		Milliken Math Series: Addition, Subtraction, Multiplication, Division, Integers, Fractions, Decimals, Percents, Laws of Arithmetic, Equations, Measurement Formulas
	Computer Programming	TI LOGO II
Junior High to Adult	Logic	Video Chess
	Typing	Touch Typing Tutor
	Physical Fitness	Physical Fitness
	Business	Market Simulation (Disk)
	Computer Programming	Teach Yourself BASIC Beginner's BASIC Tutor Teach Yourself Extended BASIC

NUM	TITLE	MODULE	NOTES
P203	c99 V4.0 EXE	E/A	c99 PROGRAM
P204	c99 V4.0 DOCS	XB	c99 DOCUMENTATION
P205	SAVEXT	XB	SAVE YOUR PROGRAM AFTER LOCK-UP
P206	VECTOR BASE V1.0	XB	D BASE + SPREAD SHEET + DOCS
P207	TIPS LABEL V1.3	XB	GRAPHICS ON LABEL(EPSON TYPE PRINT)
P208	TIPS V1.8 DSSD	XB	TIPS SHOW + DOCS
P209	SECTOR ONE	E/A	READ AND REPAIR SECTOR ONE
P210	DISK PRINT 2	XB	DISK CATALOG WITH COMMENTS
P211	PLATO UTILITIES	--	PLATO UTILITIES
P212	INVESTMENT HELPER	XB	INVESTMENT HELP. DOCS.
P213	Q BASE V1.0	XB	DBASE + DOCUMENTATION
P214	FILE UNDERSTANDING	E/A	IDENTIFIES TYPE OF FILES ON DISK
P215	TERR*WARE GAMES	XB	JOKER POKER/BLK JACK/WHEEL-FORTUNE
P216	TI 99ER CARD GAMES	XB	5 CARD GAMES
P217	CHINESE CHESS	XB	JOYSTICK - DOCS
P218	MONTE CARLO V4.3	XB	ROULETTE - DOCS
P219	LITI NL INDEX	XB	LOCATING ARTICLES IN LITI99ER NL
P220	FUNNELWEB V4.40	XB	40/80 COLUMN DSSD SYSTEM DK 8/24/91
P221	FUNNELWEB V4.40	XB	40/80 DOCUMENTATION/ADD SYSTEM FILES
P222	FUNNELWEB V5.00	XB	80 COLUMN EDITOR 12/92
P223	FUNNELWEB V5.00	XB	C.GOOD'S DOCS 80 COLUMN
P224	TI-SWEEPER	XB	MICROSOFTS MINESWEEPER GAME FOR TI
P225	MAC-LABELS V2.6	XB	ED MACHONIS'S LABEL PROGRAMS
P226	CHAINLENK SOLITAIRE	XB	SOLITAIRE GAME. GREAT.
P227	TAPE TO DISK XFER	E/A	TRANSFERING TAPE PROGRAMS TO DISK.
P228	TURBO COPY V2.0	E/A	FAST COPY PROGRAM
P229	T.I. TOOLS	E/A	COPIES ANY DISK.

NUM	TITLE	MODULE	NOTES
P230	DM 1000 V6.1	XB	LATEST DISK MGR 1000/SW99ERS
P231	TI-ARTIST PICTURES	E/A	LELAND PIPER #09 PICS/REQ TI-ARTIST
P232	TI-ARTIST PICTURES	E/A	LELAND PIPER #05 PICS/REQ TI-ARTIST
P233	TLC-THE HEALTH GAME	XB	WALTER BLOOD'S FIRST AID GAME.GREAT.
P234	NEWSLETTER PRINTER	XB	ART GIBSON V2.2 NEWSLETTER EDITIOR.
P235	TI-101 ARTICLES	XB	JACK SUGHRUE'S EDUCATION ARTICLES.
P236	DEFRAGMENTER V1.1	XB	MARK SCHAFER FAIRWARE/FIXES DISKFILE
P237	THE ALTMAN LIST	XB	LIST OF ALL TI FAIRWARE (ARCHIVED)
P238	ANIMATOR DEMO	XB	DEMO ASGARD ANIMATOR(BRAD SNYDER)
P239	REDISKIT V1*1	XB	CORCOMP DISK COPIER.
P240	G.B.S. V3.1E	XB	SECTOR EDITOR/WILL REBUILD SECTOR 0
P241	TIME LINE 99	XB	BILL GASKILL HISTORY OF TI99/4A
P242	40 COLUMN UTILITY	XB	BRAD SNYDER'S UTILS. VER 2.3
P243	XB UTILITY VOL#2	XB	HARRISON XB UTILITIES. GREAT.
P244	CSHELL 99 DEMO	XB	DEMO ONLY. JOE F. ROSS PROGRAM.
P245	TINYGRAMS V2.0	XB	ED MACHONIS TINY PROGRAMS.
P246	USEFULL PROGRAMS	XB	USE ON BUB MILLS HORIZON RAMDISK
P247	OPA NEWS #2	XB	3/92 CATALOG OF HIS SALE ITEMS.
P248	HOLE CURRENTS	XB	XB TUTORIALS BY EARL RAGUSE.
P249	THEORY OF DARK	XB	EARL RAGUSE TI ARTICLES JAN 90
P250	MISC XB PROGRAMS	XB	EARL RAGUSE XB#8 V3.0. GREAT.
P251	V9938 80 COLUMN	XB	NEEDS 80 COLUMN BOARD/MISC PROGRAMS
P252	GIF PICTURES X02	XB	REQS. 80 COLUMN BD/DEE & STACY
P253	FUNNELWEB 80	XB	80 COLUMN (ARCHIVED) DSSD.
P254	GIF PICTURES X01	XB	80 COLUMN REQ. X RATED NUDES. DSDD
P255	GIF PICTURES G01	XB	80 COLUMN REQ. GIRLS/MISC PICS. DSDD
P256	TI-ARTIST PICTURES	E/A	REQ TI-ARTIST. WAR PLANES. DSDD
P257	XHI PROGRAM	XB	80 COLUMN PROGRAM-REQ 80 COLM BRD
P258	XHÍ DOCUMENTATION	DV80	80 COLUMN DOCS FOR ABOVE PROGRAM

MAY BE YOUR LAST NEWSLETTER. We have not heard from you since 9/92. IF WE DON'T HEAR FROM YOU SOON. THIS WILL BE LAST NL.

FIRST CLASS MAIL

DALLAS TI HOME COMPUTER M.BUSH*
P.O. BOX 29863
DALLAS TX
75229 USA

9209 <- LAST NL RECEIVED 2

LONG ISLAND SOUND
The Newsletter of the
LONG ISLAND 99er's U. G.

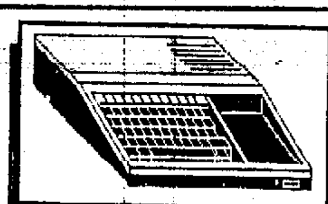
EDITOR: FRANK BUBENIK,JR

TONY IANNACONE.PRESIDENT
JERRY STOCKLER.VICE PRES
FRANK BUBENIK..SECRETARY
BOB LAWSON.....TREASURER

MEETINGS HELD ON SECOND FRIDAY EACH MONTH AT TONY'S HOME 242 SOUTH BROADWAY LINDENHURST,NY

PHONE: (516) 938-1095
BBS NO:(516) 661-3643

< < MAILING ADDRESS > >
93 MYERS AVE
HICKSVILLE, NY
11801-2424



BOSTON COMPUTER SOCIETY'S TI 99/4A USERS GROUP

PRESENTS

The New England TI99/4A Home Computer Fair

NEW ENGLAND'S PREMIER COLLECTION OF TI SUPPORT

Hard Drives, Ram Disks, 80 Column Upgrades, Databases Desk Top Publishing, High Res Graphics, Communications Word Processing, Spreadsheets, Games and much, much more.

SATURDAY APRIL 17, 1993
10AM-4PM
WALTHAM HIGH SCHOOL
617 LEXINGTON ST.
WALTHAM, MA

$3.00 Admission ($2.00 for BCS Members-Kids under 10 Free)