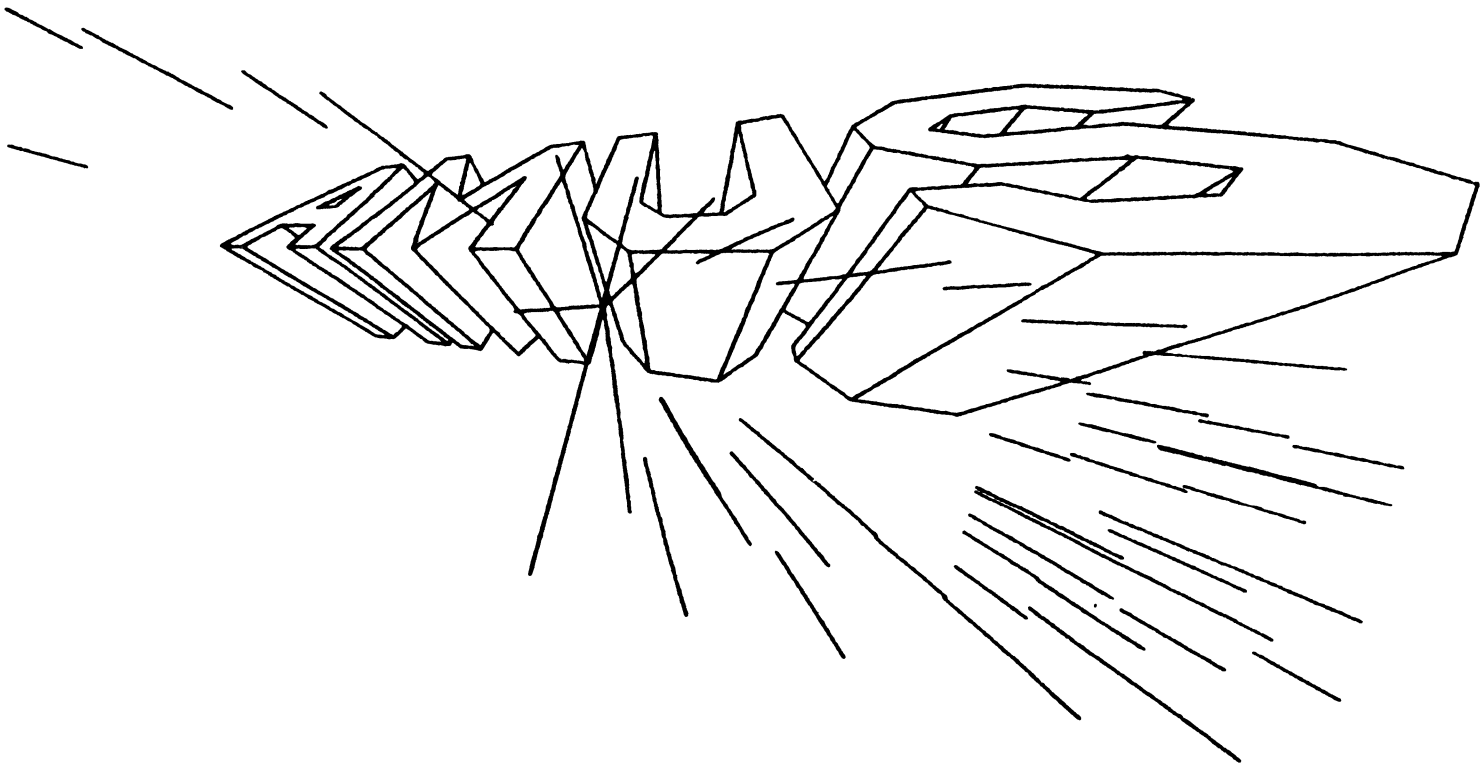

Newsletter

VOL 4 NO 7 July 1981

ALPHA MICRO USERS SOCIETY
934 B Pearl Street
Boulder, CO 80302
(303) 449-6917

AMUS BOARD OF DIRECTORS

OFFICERS

President
Stephan K. Elliott
Ecosys Corporation
Boulder, Colorado

1st Vice President
Eugene Platt
North America Title Co.
Houston, Texas

2nd Vice President
Thomas Cross
Cross Communications
Boulder, Colorado

Secretary-Treasurer
Pat Seitsinger
Dunn, Moore & Associates
Brighton, Colorado

MEMBERS

Bob Fowler
Alpha Information Systems
Palo Alto, California

Michael Lewis
Dravac
Mahwah, New Jersey

William L. Miller
William L. Miller & Associates, Inc.
Miami, Florida

Jim Taylor
Jim Taylor & Associates
Boulder, Colorado

STAFF

Sharon Greene, Secretary
Adam Fedor, Network Librarian
Pat Seitsinger, Newsletter Editor
Steve Elliott, Manager

AMUS office hours (for the summer) are from 8:00 a.m. to 4:00 p.m. mountain time. Our overworked Adminstrative Assistant (that women's libber talk for secretary) is Sharon Greene who is happy to assist you with any question you might have about AMUS, or the Alpha Micro Computer. If she doesn't know the answer to your question, she will try to direct you to someone who does.

The AMUS Newsletter is published monthly by AMUS, 934-B Pearl Street, Boulder, Colorado 80302. Subscription rates are $10.00 per year. Each member representative receives a one year subscription, the cost of which is included in the annual dues.

Additional copies and back issues of the newsletter may be ordered from Sharon Greene. Bug fixes, articles, letters, reviews of software and information about Alpha Micro applications are happily accepted. Material must be received by the 20th of the month for inclusion in the following month's edition.

Each member is allotted one free page of advertising in the newsletter each year; further advertising is at the rate of $50.00 per page with a minimum of 1/4 page. Ads must be camera-ready, black and white copy and be accompanied by the appropriate fee. Send all advertising to the editor at 1401 E. Bridge Street, Brighton, CO 80601.

The Alpha Micro Users Society Network is a computer system meant to give members access to information and other Alpha Micro users with similar interests. It consists of an Alpha Micro computer with a Hawk disk drive, a 300 baud modem, a 1200 baud modem, and 160K of memory. AMUS members are given an individual account and password on the Network so that they may receive personal electronic mail. Contact Sharon Greene for your account and password. Many thanks to Alpha Micro Systems of Irvine, California; North America Title Co. of Houston, Texas; and the Byte Shop of Reno, Nevada who have donated equipment and software to the Network.

AMUS has a library of programs which have been donated by members for distribution to other members. Programs are available either through the AMUS Network, or, if you prefer, we can make floppy or Hawk cartridge copies and mail them to you. Orders may be placed through Sharon Greene.

Second Class Postage paid at Boulder, Colorado 80302, ISSN 0273-8708, USPS 567-330.

We are planning a massive revamping of the AMUS Network as soon as Mike Lewis sends us the new version of the DRAVAC Time Sharing and Security System (TSASS). Once TSASS is up we will be able to put many demonstrations of software packages on the system so that you can try out software packages on the AMUS system before plunking down your hard earned money for a package that isn't quite what you expected. We already have the COBOL package from Angusglow in England ready and waiting, and we are anticipating several other major items in the near future. If you have a piece of software that you would like to have placed on the AMUS Network so that other members can see what you're up to, please contact Adam Fedor or myself so that we can explain how we plan to set up the demonstrations.

We would also like to remind you to send in information about yourself and your dealer for us to include in the upcoming AMUS directory. The form is in the back of this newsletter. We think that a directory of members will be a very useful tool when you're looking for someone else that has similar interests or hardware that you can consult when you're trying to figure out a knotty problem that someone somewhere has obviously already solved.

Steve Elliott

* * * * * * * * * *

AMOS VERSION 4.5

Version 4.5 is out to dealers and is being delivered with all new systems. A preliminary bug list is being prepared by our resident expert, Bob Fowler, and should be in the August newsletter.

4.5 has some really nice features - Electronic Mail, for one. I haven't had a chance to test it yet, but from the manual, it looks good.

The Basic compiler reports syntax errors as it compiles, rather than waiting until the end. Nice if you want to stop it when you find some stupid typo you've introduced. If an error occurs during the compile, a run program is not produced. A new error trapping procedure allows the program to continue if an error is followed by a Control-C. An INCLUDE verb allows you to call external routines without including them in the source code. FORCED'RANDOM forces a disk access every time your program reads a file record even if that record is already in memory. COMPIL optionally reports any unmapped variables it encounters during the compile.

TRM.DVR is no longer required in system memory, but must be somewhere if you're calling it from basic programs.

A terminal driver building program allows the user to build their own terminal drivers. The user is asked questions about the terminal and a source program is created.

An object file library generator allows you to define libraries of object file routines which all assembly language programmers on your system can make use of.

Also included are magnetic tape backup programs, drivers for the new AM-120 board and software support for the AM-710 (128k memory).

Now for the bad news. The monitor is 1000 bytes larger....BUT, there's also a small version of RUN.PRG named RUNSML.PRG which you can use in place of RUN if your basic programs do not use the trigonometiric functions of EXP, LOG, LOG10, FACT, or exponentiation (^) operations. RUNSML is 1000 bytes smaller than RUN. If you implement the newer version of run and one of the functions are called, you will see the error message,

?Unsupported function

or, if using error trapping, the error number is 35 (I don't think this number is yet documented).

Nested INCLUDES do not always report errors. Bob Fowler says you must clean up the lowest hierarchy of the included procedure before you implement the next hierarchy.

Sorting of sequential files now requires that each variable be separated by a comma rather than spaces or tabs. Bob is working on this one - more about it next month.

Should the stack fill, just at the point it fills, you get some hinky error message which relates to the next statment to be executed. It never gets to the point of reporting stack overflow. (This one took a minute to figure out.)

More next month...

Incidentally, the documentation is EXCELLENT!!! Alpha Micro seems to get better and better in this area. (Remember the Pre-Release Notes ?)

Pat

Structured AlphaBASIC
By Steve Elliott

One of the nicest things about the Alpha Micro computer is AlphaBASIC. It allows you to do several things which make programming, and especially structured programming, very easy to accomplish.

What is structured programming?

It's a list of narrow minded, constraints fabricated by some academecian, which produce bulky, redundant, inefficient code and inhibit the creative energy of programmers who otherwise would be able to go about their business.

or

Its the fig newton of some old codger's imagination who couldn't follow his own logic, and decided that it would make life easier if he just outlawed the GOTO statement.

or

It's another attempt by management to interfere with the process of programming.

or

It's an organized approach to the process of designing, creating, and documenting programs, meant to make the products of an organization more cohesive, understandable, and easliy modified.

We like to think of it as closer to the last definition. In our shop, we have been using Structured Programming techniques because the programmers like the idea. It does not seem to squeeze the creative juices dry, nor does it seem to create inefficient code. It does create longer source code listings, since we use longer variable names than most other software houses we have observed. The whole idea is to build code in such a way that it can be tested throughout various stages of its creation, can be easily read by other programmers without a ten hour introduction to the problem, and can be easily modified later.

This is accomplished by setting down specifications which everyone follows when they are bulding a program about where certain common functions will occur, requiring that variable names mean something (NEW'ITEM'COST vs NI'CST), and that every subroutine have only one entry point and one exit. We also set up standards for indentation so that subroutine titles, loops, and IF THEN logic are easily recognized.

This article outlines most of the conventions that we at Ecosys Corporation use so that you can see how this idea might work. We don't want to impose any of our constraints on you, nor do we think that we have the ultimate answers to all the problems encountered in programming, but we do believe that we are more efficient at creating programs than other folks who are not using structured programming techniques.

Actually, the hardware folks have been using modular design for years with great success. Take your AM-100 computer for example. At the highest level, you have the COMPUTER, the box you point at when you're showing off your new expensive toy to the astounded neighbors. Within it are several separate components, the AM-100 card (the CPU), the memory card(s) and hee disk controller(s). The components are independant in that they have only one way into and out - the pins at the bottom that follow the conventions of the S-100 Buss layout. You can take out a memory card and replace it with a faster memory , and the system will chug along happily. Subsets of the memory card are the chips on the card. Each chip might be able to handle a few bytes of storage. If one goes bad, we can take it out and replace it with a new one within a matter of moments. This is possible because the parts are modular, and are constructed in such a way that there is only one way to get information into, and one way to get information out of the device.

If you outlined the construction of the memory portion of the COMPUTER it would look like this:

```
I.  THE COMPUTER
    A.  The Memory card
        1. The chips on the memory card
```

The heirarchy in this example is easy to see, and the relationships between the items is easily believable. relationships in software are sometimes harder to see, but they are there none the less. If we were to outline how a random access file file is constructed in AlphaBASIC, it might look like this:

```
I.  THE FILE
    A.  Records withing the file
        1. fields within the records
```

If we similarly outlined the process used to add information into a file it might look like this:

```
I.  OPEN UP THE FILE
    A. Find the next available record
       1. Enter information into each field
    B. Store the record
II. CLOSE THE FILE
```

As you can see, with this kind of hierarchy, and with modular type construction, we could change information about a field without worrying about what kind of effect

the change would have on locating the next available record. If we want to stretch this metaphor to its illogical limits, we could say that computer hardware is being built using modular techniques, while much of the software is being created with the same kind of techniques that were used to wire together radios in the 1930's. If you've ever looked at the underside of an old tube type radio you'll see wires that look more like spaghetti than anything else. If you look at a lot of the software being built today you'll see the same spaghetti logic at work.

These are the rules which we loosely adhere to. (loosley, because there is always an exceptional case, and also because we don't want people to spend hours coding a complex maneouver that could easily be done by stretching a rule just once).

1. No line numbers. They just take up valuable space and time. If you need to go back and insert something into a program, numbers can slow down the process by forcing you to renumber blocks of code within a program. The only drawback that we can see to this is that if you have an error, RUN won't report where the error occurs. If this happens during testing and we're really having trouble, we number the program, RUN it to find the problem, and then go back to the unnumbered code to fix it up.

2. Always use labels. Since there aren't line numbers, this is the only way you can go. The labels should also reflect that is going on in that section of code. Reasonable labels are: FETCH'NEW'RECORD'NUMBER:, PAUSE:, MAIN'MENU:, OR UPDATE'MASTER'FILE:. Unreasonable labels are: DD3:, LOOP:, or X: because they aren't informative. Think about what the program would look like to you if you had never seen it before, and were assigned to go in and change something.

3. Use meaningful variable names. VGX, X, and DR1 are easy to type, but don't give you any idea what they stand for. While you're enmeshed in the process of creating a program, VGX is obvious as the acronym for the volume of gas exchanged between two companies in a month. Six months from now when you return to the program, it's gibberish and you'll spend an hour just getting refamiliarized with the program's vocabulary. When we do use single letter variables, it's usually for a temporary counter in a FOR NEXT loop or a variable that is returned from a LOOKUP and then checked in the very next line.

4. Indent like crazy. (meaningfully). We've copied our indentation rules from two sources: The Little Book of Basic Style by John Nevison, and the conventions used by PASCAL. Naturally we don't exactly follow the rules of either, but we have borrowed some of the more useful (to us) hints:

   a. Labels go all the way to the left.

   b. Everything within a labeled routine gets indented one TAB to begin with. This sets labels off and makes them easy to find.

   c. Everything that occurs within a FOR NEXT loop gets indented, including nested loops. i.e.:

```
DISPLAY'THE'RESULTS:
        FOR N = 1 TO 100
            CALL PRINT'A'LINE
            FOR R = 1 TO 3
                PRINT ASC$(7)
            NEXT R
            PRINT "LINE #"; N; "HAS BEEN PRINTED"
        NEXT N
        RETURN
```

   This makes it easy to spot loops, and nested loops.

   d. Use PASCAL-like indentations for IF THEN ELSE constructions. This is where AlphaBASIC really shines. (also where you can get errors which can't possibly occur and do absolutely impossible things). All of this is accmplished with the use of the '' and the ': ' pair. The format of a simple IF THEN ELSE is as follows:

```
        IF relation
            THEN
                DO THIS :
                AND THIS :
                AND ALSO THIS
            ELSE
                DO SOMETHING ELSE :
                NOT TO MENTION DOING THIS
```

   Notice the use of just the '' when you want to go to the next section of the statement, and the use of the ': ' pair if you want to accomplish more than one thing within a THEN or an ELSE. Caution! if you forget one of the ''s at the end of a line, or put a space after it, you will get startling, bogus results. This kind of format allows you to specify exactly what you want to happen, unlike most BASIC programming we've seen where conditions are checked to decide what aart of

the next few lines of code should be skipped. IF THEN ELSE statements can also be several layers deep. Anything goes as long as you don't exceed 512 bytes for one statement. Look over the following statement:

```
BOY'SCOUT'SELECTION:
  IF APPLICANT'SEX = "M"
    THEN
      IF AGE => 12 AND AGE <= 16
        THEN
          IF DUES = "PAID"
            THEN
              CALL INITIATE'INTO'TROOP
```

5. Since the resulting programs rely heavily on subroutines which are called by IF THEN logic, they tend to have a lot of IF THEN decisions at the top of the program, and a lot of subroutines at the bottom. We think that the smaller the subroutines are, the better. Why?, because the more specific a subroutine is, the easier it is to use because it will just do one thing and do it well. Also, you might be able to use it in another program later by using the YANK capabilities of VUE to collect a lot of subroutines that you are familiar with into a program like putting together a model of the Empire State Building with toy building blocks.

6. Be very frugal with GOTO's. We only use GOTO's in two places: 1) to construct a loop, in which case the GOTO points up a few lines, and 2) to blast out of a program altogether. This might happen if you lookup a file you are about to create and find one already in existance and the user does not wish to destroy the already existing file: GOTO QUIT.

7. Lay out where certain types of subroutines are going to reside within the program. Our programs use the following format (usually):

   A. MAPS go at the top (naturally)

   B. Any files that are going to be used are looked up, and then opened

   C. the main menu, or the main logic comes next which cascades into about three more levels.

      1. The first level deals with files

      2. the next with records

3. and the next with fields

4. Error checking on fields, calculations, and odd subroutines such as screen control reside on the lowest level

8. Under normal conditions, if you call a subroutine, it can be found on the next level down. This won't hold true if the subroutine is called from several different levels, but at least you won't spend vaulable time looking all over for a subroutine.

9. Use English-like spacing on line wherever possible. We have trained ourselves to read English with spaces between words. Programs are easier to read if they follow the same conventions. Which of the following lines of BASIC is easier to understand?

```
VOLUME=SQUARE(1,7)+AREA'SIZE+SQUARE(2,7)
```

or

```
VOLUME = SQUARE(1,7) + AREA'SIZE + SQUARE(2,7)
```

If you find yourself mentally inserting spaces into the first line, then that takes time, and time is what we're trying to save since programmers are expensive. It might take a little more effort to add spacing when a statement is entered in a program, but as we all know, you don't just write a line once and then never reference it again.

10. Only enter more than one command per line if it makes sense. There's nothing like finding a secret trap door embedded deep within a line as in the following example:

```
TERM = 7   IF J > 4 RETURN : AMOUNT = 12000
```

It may seem obvious in this example, but within a long program, a programmer could spend hours trying to find why AMOUNT is always zero. We use several commands on one line if they are all one related thought; initializing values, tabbing and clearing lines, pausing for a while. All of the following make sense:

```
TERM = 7   : INTEREST = .19  :    AMOUNT = 12000
? TAB(10,1); TAB(-1,10); TAB(10,10);
FOR X = 1 TO 1000 : NEXT X
```

These don't:

```
TERM = 7 : FOR N = 1 TO 12 : J = J + N
INPUT "Enter your choice ", CHOICE : CALL ADJUST
CALL ENTRY : IF R > 52 CALL NEW'YEAR : NEXT N
```

By now you should be getting the idea. We want to create programs that make sense to the eye, follow a logical, hierarchial format, and have independent, well defined modules. It usually takes writing about three medium sized programs before structured programing comes naturally. Most of the confusion you will encounter, disregarding indentation rules, will be centered around two essential questions:

1. What is the best way to phrase an IF THEN ELSE question - IF X = Y THEN, or IF X # Y THEN? This varies with the situation. Usually if you think about both possibilities, you'll discover that one requires less testing and less calls than the other. It's also a good idea to try to construct IF THEN ELSE statements so that false conditions are encountered as soon as possible, dropping through to a RETURN. This will make your programs run faster since less checking will be done.

2. If a aariable is checked in a higher level subroutine that is set in a lower level subroutine, should I clear the variable in the high, or the low level subroutine? We usually clear variables within subroutines where they get set. This way, if you need to use the subroutine in another program later, you only have to copy the subroutine and it is complete within itself.

Once you get familiar with structured programming techniques, we believe that you will find that programs are developed faster, it's fun YANKING in subroutines that you're familiar with, AND you have more confidence in the code; you KNOW that it's right, and even if a bug does develop, you immediately know where to go to correct the problem.

CUSTOM DESIGNED MANAGEMENT AID PACKAGE

SELECT AT LEAST TEN OF THE PROGRAMS LISTED BELOW AND WE WILL WRITE A MAIN DRIVER PROGRAM WHICH WILL DISPLAY A MENU CONSISTING OF THE PROGRAMS THAT YOU HAVE SELECTED.

Future Value of An Investment
Future Value of Regular Deposits (Annuity)
Regular Deposits
Accrued Interest on Bonds
Apportionment by Ratios
Bayesian Decision Analysis
Continuous Interest Compounding
Nominal Interest Rate on Investments
Regular Withdrawals from an Investment
Minimum Investment for Withdrawals
Effective Interest Rate on Investments
Financial Statement Ratio Analysis
Lease/Buy Decision
Markov Analysis
Net Present Value of an Investment
Nonlinear Breakeven Analysis
Present Value of a Tax Deducation
Profit Sharing Contributions
Program Eval & Review Tech (PERT)
Rule of 78's Interest
Swedish Machine (Queuing Theory)
Syndicated Investment Analysis
Current Value of a Treasury Bill
Critical Path Method (CPM)
Depreciation Switch
Economic Order Quantity
Economic Production Quantity
Financial Mgmt Rate of Return
Earned Interest Table
Initial Investment
Depreciation Rate
Depreciation Amount
Salvage Value
Internal Rate of Return
Discount Commercial Paper
Principal on a Loan
Regular Payment on a Loan
Last Payment on a Loan
Payoff Matrix Analysis
Remaining Balance on a Loan
Term of a Loan
Annual Interest Rate on a Loan
Mortgage Amortization Table
Transportation Algorithm

THIS PACKAGE INCLUDING SOURCE CODE AND COMPLETE DOCUMENTATION IS AVAILABLE ON FLOPPY OR HAWK DISC. FOR FURTHER INFORMATION OR TO PLACE AN ORDER CONTACT CONTEMPORARY CYBERNETICS GROUP, 1204 WILLOW GREEN, NEWPORT NEWS, VA 23602 (804) 599-4749. WE ACCEPT MC AND VISA.

ESTABLISHED
1968

(201) 794-1114

aim DATA PROCESSING
BUSINESS COMPUTER SALES & SERVICE
BOOKKEEPING SERVICES

21-07 Maple Avenue
Fair Lawn, N. J. 07410

May 19, 1981

Dear AMUS Member;

AIM is a full service Alpha dealership serving the greater Metropolitan New York area. We provide hardware maintenance, programming and consulting in addition to the sale and installation of Alpha Micro systems. We will also provide an Alpha Micro system for any show that you would like to demonstrate Alpha at in the New York area. If you have a client with offices in the greater Metropolitan area we will work with you in the sale and installation of the system as well as the maintenance.

We offer four application packages that you should be aware of.

ASIMS - AIM STATIONERS INFORMATION MANAGEMENT SYSTEM

ADOMS - AIM DENTAL OFFICE MANAGEMENT SYSTEM

AVOMS - AIM VETERINARY OFFICE MANAGEMENT SYSTEM

ARIMS - AIM RETAIL INFORMATION SYSTEM

In addition to the above listed packages, we offer an enhanced version of the Alpha Business Accounting System to provide automatic restart for all batch updates and sorts. For the interactive processes we have added code to prevent bracketed records and other enhancements in the area of recovery control and restart.

We have enclosed a stationery brochure for your review. The packages we offer are all fully documented.

Pricing:

Documentation only (hard copy) - $50.00

Demo 5 Meg cartridge (run modules and documentation) - $150.00

Software License for you to install system with our support via Teleprocessing - $2,500.00

Source License - call us for terms

GOOD LUCK AND GOOD SELLING!

AIM - Authorized Alpha Micro Dealer

CHAMPAIGN COMPUTER COMPANY, INC.

PLEASE CALL OR WRITE FOR INFORMATION .....

DISK #4: ACCOUNTING PLUS INVENTORY CONTROL SYSTEM (APICS): over-the-counter invoicing with Inventory Control, complete file-locking. Perfect for parts dealers, etc. Handles back/orders. Enhanced AlphaAccounting may be interfaced to APICS. The ideal package!

DISK #6: "GOODIES": Includes source for many subroutines, drivers, and misc. programs. Complete list of programs available upon request.

DISK #9: MUNICIPAL UTILITIES BILLING SYSTEM (MUBS): Handles all types of meters, also sewer and/or garbage collection charges, may be interfaced to AlphaAccounting. Purchased gas adjustments, taxes, bond issues, budget billing, may chain meters, block billing or bill by routes, Also for electric and gas companies.

DISK #10: PRINTER DRIVERS: Includes object code for printer drivers for Teletype Model 40, Diablo 1600 series, Qume Sprint 5, and NEC 5510, 5515, 5520, and 5525 printers. Supports bold print, underlining, full print speed. Turns off Teletype Model 40 chain when printer is not in use. Bi-directional printing for full use of your printer!

DISK #11: SUPER ALPHA-ACCOUNTING MODULES: Written as it should have been; map statements changed, complete file locking, add vendors on the fly, etc. Source for AR, AP, PR, GL modules, debugged, and enhanced. Complete list of enhancements available upon request. Buy AR, AP, and PR; get GL free! Please contact CCC for details. Save time, money, and patience! Get it bug-free from CCC!

ALPHA-ACCOUNTING MANUALS: Complete user's documentation for the standard AlphaAccounting software. Available in hardcopy or source for AR, AP, PR, GL, IC/OE. Worth a look!

CHAMPAIGN COMPUTER COMPANY, INC.
P.O. BOX 350
MAHOMET, IL 61853
Phone (217) 586-4131

Available for a limited time. A collect of programs, routines, etc.

Please Direct inquiries to:

Paul Gayeski 522 S.E. 29th Ave. Portland, Oregon 97214

Phone: 1-503-231-9233 <= Jean will tell you the price.

Pascal...

```
   WRDSIN PAS 44  <= Produces a BASIC Source diagnostic listing...
   ADDCHR MAC 1   <= Adds character to string...
   PMIN1  MAC 1   <= Retreives minimum integer value...
   PMAX1  MAC 1   <= Retreives maximum integer value...
   PSORT1 MAC 9   <= Sorts array...
   PSRCH1 MAC 4   <= Searches array sequentially...
   PSRCH2 MAC 6   <= Searches array binarily...
   PFILL1 MAC 2   <= Fills field with character...
   PRPLC1 MAC 6   <= Replaces characters in range...
   PRPLC2 Mac 6   <= Replaces characters in set...
```

Assembler to programs...

```
   ARCHV1 MAC 33  <= Documentor lists comments in first block of files.
   CLRMEM MAC 5   <= Clears memory at or following a particular module.
   CMDLST MAC 5   <= Lists all jobs command file status...
   FORST1 MAC 5   <= Creates & initializes tree structure memory module.
   GTMEM1 MAC 4   <= Gets memory module...
   JOBINF MAC 48  <= Displays or lists info on any job in system.
   JOBLST MAC 7   <= Displays some job states...
   LOADIF MAC 2   <= Loads module(s) if not in system or user memory...
   PSWRDS MAC 5   <= Displays accounts and passwords (Priveledged)
   SEQPPN MAC 20  <= Sequences Account...in a variety of ways...
   SETIE  MAC 2   <= Set image-echo suppress..
   SETC   MAC 2   <= Set control-c interupt...
   SHWPPN MAC 17  <= Shows account in a variety of ways...
   SYSMDL MAC 4   <= Lists system modules and size...
   YNKEY  MAC 11  <= Command file routine to get YN response...
```

Basic Subroutine...

```
   CMNSWP MAC 6   <= Exchange data bettween module and basic map area...
   CTRLC1 MAC 2   <= Set control-c interupt...
   DATEID MAC 12  <= Proofs date returns day in year, week & names day, month.
   FILL1  MAC 6   <= Fills string with multiple characters...
   FILL2  MAC 5   <= Fills string with 1 character...
   FILL3  MAC 2   <= Fills indirect address...(use with SYSRCH)
   GTCHR3 MAC 14  <= Get character display set while waiting...
   IMGECS MAC 2   <= Set Image-Echo suppress..
   JOBS1  MAC 8   <= Lists other jobs or suspend, revive, control-c them.
   KILCMD MAC 1   <= Kills command files...
   LKUP1  MAC 14  <= Wild card file lookup...
   LKUP2  MAC 10  <= Lookup Accounts...
   LCLEKO MAC 2   <= Set local echo-echo suppress...
   MSORT3 MAC 11  <= Sorts array..
   MSRCH4 MAC 5   <= Searches array...
   RPLC1  MAC 11  <= Replace characters in/out of set or range...
   SETCON MAC 2   <= Set control-c interupt (Pageable as prg or sbr.)
   SETPRG MAC 2   <= Sets correct program name into system JCB
   SKIPTO MAC 2   <= Skip to line in command file following specified label.
   SYSRCH MAC 3   <= Searches for module...
   TREE1  MAC 5   <= Tree structure lookup, add, delete
   UDATE1 MAC 1   <= Get system date...
   UTIME1 MAC 2   <= Get system time...
   VOLID1 MAC 3   <= Retreives volume id info...
```

All routines are in source code for your enhancement. Comment block at start of listing and additional documentation included. Single user and network prices. Available on diskette in STD or DSA only. Updates or revisions by source code listings.

contemporary cybernetics

We are offering a series of programs designed with the business manager in mind. Each program is accessable through a main driver routine which allows you to easily select the program you wish to run. The following is a copy of the main menu display:

MANAGEMENT AIDS

1) Accrued Interest on Bonds
2) Apportionment by Ratios
3) Bayesian Decision Analysis
4) Continuous Interest Compounding
5) Current Value of a Tresaury Bill
6) Critical Path Method (CPM)
7) Depreciation Switch
8) Economic Order Quantity
9) Economic Production Quantity
10) Financial Mgmt Rate of Return
11) Financial Statement Ratio Analysis
12) Future Value of an Investment
13) Internal Rate of Return
14) Lease/Buy Decision
15) Markov Analysis
16) Net Present Value of an Investment
17) Nonlinear Breakeven Analysis
18) Payoff Matrix Analysis
19) Present Value of a Tax Deduction
20) Profit Sharing Contributions
21) Program Eval & Review Tech (PERT)
22) Rule of 78's Interest
23) Swedish Machine (Queuing Theory)
24) Syndicated Investment Analysis
25) Transportation Algorithm

This package including source code and complete documentation is available on floppy or Hawk disc.

For further information or to place an order contact CONTEMPORARY CYBERNETICS GROUP, 1204 Willow Green, Newport News, Virginia 23602, (804) 599-4749.

IAC IN AGENCY COMPUTERS

A T T E N T I O N A L P H A D E A L E R S

* HERE IS YOUR OPPORTUNITY TO PENETRATE THE INSURANCE MARKET *

The insurance market is large, local, and waiting to be tapped by YOU. Of the approximately 68,000 Independent Insurance Agencies in the country, fewer than 10% have been automated.

The local insurance agency is an ideal customer for an ALPHA. The agency's main product is paperwork, at which a computer is much more cost effective than a staff of clerks. The reason this industry has not yet been fully automated is the peculiar nature of its business. A number of generalized systems have been adapted to it, but the low priced systems which most agents can afford are not suited to their needs.

The IAC insurance software just released for the ALPHA can be used by YOU to change this situation in your neighborhood. If you can take care of the ALPHA, our software will take care of all the requirements and idiosyncracies of the agents' business. Selling it should be easy since it is demonstrably a superior product and it is already known in the industry. Our systems are listed in the industry sponsored Insurance Institute of Research (IIR) Automation Guide under the IMA corporate name.

The IAC software system has been eight years in the making. It was designed and implemented on larger machines. We discovered the ALPHA last year and were amazed by its performance and ability to handle all our software which was designed for large sophisticated minicomputers. The system is now fully implemented on the ALPHA and is available at a reasonable license fee to dealers.

We offer a fully documented data-base system which handles all agency functions. The system is menu-driven, and its user characteristics are controlled by parameter files, providing extensive program adaptability. In addition to being a complete agency system, it also includes interfaces to word processing, and to the IIR Agent-Company interface pilot.

The IAC system can help you succeed in this potentially lucrative market. We will support you with marketing brochures, program installation, and user manuals. Because we have been working with this industry for many years and are the prime developers of the system, we can answer all your questions and give you support and advice when you need it.

Write to us or call Mike for further information.

15 BROOKFALL ROAD • EDISON, NEW JERSEY 08817 • PHONE (201) 572-0798

Lambda Biosystems, Inc.
7342 SOUTH ALTON WAY • SUITE F
ENGLEWOOD, COLORADO 80112
(303) 773-8484

NEW PRODUCT ANNOUNCEMENT

CPU-CPU PARALLEL INTERFACE AND SUPPORT SOFTWARE: CPU-LINK

GENERAL DESCRIPTION:

CPU-LINK is a software system utilizing the parallel ports on the AM100T*. Up to four (4) remote and one (1) central CPU can be linked to pass data from both random and sequential files as well as any block of information. One 50-pin ribbon cable links the CPU's in a 'daisy chain'.

FEATURES:

+ average speed approx. 50kilobaud (higher possible)
+ complete ACK/NAK sequence assuring no loss of data
+ automatic resync if send or receive CPU gets interrupted
+ eliminate most OPEN files (and buffers) in remote programs
+ uses XLOCK* for simultaneous use
+ CPU's can be arranged in priority sequence
+ support software 'crunches' exiting programs into new version
+ automatic real-time correction of transmission errors
+ actual transmission done in XCALL* subroutine
+ send binary data (exact image as MAPED*)
+ flexible for conversion to any parallel boards (PIO)
+ uses no system if desired (monitor memory)

AVAILABILITY: Diskette (CDC) or hard disk (HAWK/PHOENIX)*

COSTS: $1250.00/ system used

EDUCATION: Workshop will be scheduled in the Denver area

NOTES: (1) This software will be protected with some type of 'pirate stopper' (DRAVAC DR-100)**
(2) Quantity discounts will be negotiated

* ALPHA MICROSYSTEMS

** DRAVAC LTD

AMERICAN BUSINESS DATA SYSTEMS

REDBAL™

RELATIONAL DATA BASE MANAGEMENT SYSTEM
FOR THE ALPHA MICRO

With REDBAL, it's a snap to build and use small data files. Even non-technical users can create files; edit, select, sort, and merge them; and print reports from them. The User Interface is guided by clearly worded menus and question-and-answer dialogs. The versatile File Editor allows you to add, change, delete, display, and search for records in any file. The Report Generator queries you for the report format (file name, page dimensions, titles, column headings, print fields), then stores the format description for future retrieval.

Programmer Interface: Basic programmers can access data through a group of Basic language subroutines that allow storage and retrieval of individual fields or entire records.

Specifications: REDBAL handles any number of files, each with up to 30 data fields per record. Four types of data fields: single ASCII character, floating point number, variable length string (100 char max), and paragraph (2048 char max). (The File Editor includes a Paragraph Editor that lets you modify individual characters, phrases and lines.) Fields can later be added or deleted, even if a file already contains data. Relational operators: select, project, join. Written in AlphaBasic. Runs in 32K user memory, with RUN.PRG (or RUNSML.PRG) in system memory. Recommended maximum file size: 2,000 records (larger files are permitted, but access is slow). No security or multi-user features ... yet.

Uses: We originally developed REDBAL to handle the data dictionary for one of our consulting projects. It now has over 50 users world-wide, and we've heard nothing but praise from them. It's great for any filing and reporting job that lends itself to tabular data. It is being used for mailing labels, marketing data bases, bibliographies, and a host of other applications. It's a joy to use: you can produce simple systems almost as fast as you can design them.

Special Bonus: To help develop REDBAL, we created a special librarian program called LIBRA [tm]. It copies the REDBAL subroutines — and any other source files such as file maps — into Basic programs. It has a "macro" facility for altering files as they're copied. Copy files can be nested up to 10 levels deep. LIBRA also renumbers Basic programs. It successfully handles continuation lines. Options are provided: Compile after copying; number only paragraph labels (to shrink large run files); delete all copy files (to shrink source files before using VUE); cross reference of all copy files in ppn; copy file library ppn. It's written in assembly, and it's FAST. You get LIBRA as part of REDBAL, or you can purchase it separately.

Unbelievably Low Price: Single-user license, $175 (includes source code and documentation). LIBRA only, $50. Manual, $25. AMS or STD format single density diskettes (please specify). For other media, please contact us. California residents please add 6% sales tax. Dealer inquiries invited. Send payment with your name, address and phone number to:

AMERICAN BUSINESS DATA SYSTEMS
890 Osos, San Luis Obispo, CA 93401
(805) 543-5487

What runs four times faster than BASIC?

FORTRAN 77 for the Alpha Micro Computer!

Some FORTRAN 77 programs run as much as ten times faster than they would if written in BASIC. Most average more than four times faster — letting you really use the Alpha Micro Computer's incredible processing power.

And you can use the wide range of FORTRAN programs written over the past 35 years — for engineering, mathematics, scientific and statistical uses. With the features and extensions of the ANSI 77 standard, FORTRAN 77 now has the flexibility to implement the structured design of modern business applications.

FORTRAN 77 is a full implementation of the ANSI standard, not a microcomputer subset. FORTRAN 77 for the Alpha Micro Computer offers features that make it one of the most flexible and powerful FORTRAN versions:

- Six data structures, including enhanced character manipulation
- Reentrant object code — all jobs share one runtime package
- Compiles in only 22K of memory
- Three pass compiler for fast execution — compiles directly to machine code with no pseudo code
- Dynamic memory allocation and virtual paging — to execute large programs in small amounts of memory

FORTRAN 77 is also fully compatible with Alpha Micro files generated using BASIC programs, so you can choose the programming language most appropriate to your application while maintaining the integrity of your data base.

FORTRAN 77 has been successfully installed, tested, and is in use throughout the U.S.A. and internationally. It is available with full documentation and includes the compiler, utilities, a mathematics library, and over 70 scientific subroutines. It even includes a utility that lets you compile programs directly from AlphaVue.

For more information, call or write:

SOMA INTERNATIONAL
4425 Jamboree Road, Suite 280
Newport Beach, CA 92660
(714) 752-6513
TELEX: 65-5383 Answer Back PHOTOBANK

STARR COMPUTER SYSTEMS INC.™

6126 Melissa Lane • Omaha, Nebraska 68152 • (402)571-1722

Dear **Teletype 40 Printer Users:**

Starr Computer Systems and D. K. Engineering can provide you with an interface driver to run your Teletype Model 40 Printer from your AM300 board, with full printer control at 9600 baud.

FULL CONTROL!
FULL SPEED!

This software is entirely interrupt driven, contains no time consuming null insertions or timing loops and runs with minimal impact on the system. The Teletype motor is under complete software control, and will start and stop automatically.

This software is used in place of the AM300.IDV, but with added features that allow full control of one or more Teletype printers from Teletype ports while providing normal control for all other ports.

GET FULL PERFORMANCE FROM YOUR PRINTER !!!

We have exhaustively researched the problem of interfacing the Teletype to the Alpha Micro. Our systems design approach allows the Teletype to run at 9600 baud, with full motor control, and will never miss a character.

Includes program TTY300.IDV and full documentation.

Sincerely,

Dick Starr

Richard J. Starr
President

SIZE OF AMOS MONITORS AFTER SYSGEN

This document is meant to aid you in calculating (precisely) the total room necessary to configure a given kind of AMQS system, without having to actually go through the whole sysgen process. It can be used by a salesman & a prospective customer to figure out whether a particular kind of system will actually fit in a prescribed amount of core. It may also be used by the system programmer to "juggle" system size requirements on paper, without having to kick any users off the system to do so.

In general, the AMOS resident monitor consists of a what we shall refer to as the "Base Monitor", whose size (in bytes) is a constant for any given AMOS release. The size of this Base Monitor is found at location #ZSYDSK (see SYS.MAC); simply type "PRINT WORD(136)" in BASIC to get this number. The SYSTEM.MON found in PPN [1,4] consists of the Base Monitor plus an additional area whose size is usually about 4000 bytes (see tables for exact size in each AMOS release). This additional area is where the Disk Driver is contained, and where new drivers are inserted by MONGEN. Most drivers are less than 1000 bytes, and the remainder of this 4000+ byte area is left unused. On the system disk, this unused area actually takes up disk room (thus resulting in "apparent" SYSTEM.MON sizes of 13000+ bytes), but during sysgen time this unused area is eliminated in core, so that only the Base Monitor and the Disk Driver occupy (precious) memory.

To determine the exact total size of an AMOS monitor, you must first have the following information :

(1) The AMOS Release Number (e.g. "4.4")

(2) The Disk Driver used in the Monitor (e.g. "PERAMS")

(3) A listing of the SYSTEM.INI file (only those lines that preceed the line "SYSTEM" effect the size of the AMOS monitor).

The AMOS Release Number is needed because the sizes of various programs change with each release. The Disk Driver must be known, because each driver has a different size. Every single line that preceeds the line "SYSTEM" in a SYSTEM.INI file will add to the size of the AMOS monitor at sysgen time; it is possible to insert lines into this part of the SYSTEM.INI file that do not affect the monitor size, but they do not have to preceed the "SYSTEM" line (they can be moved to the area following "SYSTEM").

Page 2 contains a summary of the various contributions to the monitor size. The various constants on page 2 are correct for AMOS versions 4.0 thru 4.4A, you will note that some of them have changed during this time period, and there is no guarantee that they will not change in the future. Also, I have not at this time checked any earlier versions.

Page 3 contains the size (in bytes) of various programs for several AMOS releases, with room for future expansion. The user may wish to add a few more programs to this table, based on his own particular needs.

CONTRIBUTIONS TO AMOS MONITOR SIZE

(-) SYSTEM.MON — Size Of Base Monitor (=WORD(136))
+ Size of Disk Driver
+ 2

(1) JOBS — For each jobname in a JOBS line :
+ 292 bytes

(2) TRMDEF — For each TRMDEF line :
+ 70 bytes
+ in-width buffer size
+ in-buffer size
+ 2 x (out-buffer size)

For each different terminal driver :
+ 16 bytes
+ Size of terminal driver (e.g. HAZEL.TDV)
For each different interface driver :
+ 16 bytes [I only checked this on 4.2 and after]
+ Size of interface driver (e.g. AM300.IDV)

Note : If a TRMDEF line uses the same IDV and TDV as previous lines, then the monitor size is not increased by these drivers a 2nd time. The pseudo drivers PSEUDO and NULL contribute no additional room to the monitor, but any pseudo-terminal still requires 70 bytes + the buffer sizes (see above).

(3) MEMDEF — + 18 bytes for the first MEMDEF line
+ 12 bytes for any additional MEMDEF lines
SYSMEM — + 10 bytes if present (AMOS 4.3 or later)

Note : This data only reflects some simple PIISCEON memory board sample cases; more elaborate cases require more memory.

(4) DEVTBL — [following is for AMOS 4.0 to 4.3]
+ 18 bytes for the first device (NOT including DSK0)
+ 8 bytes for each additional device

[following is for AMOS 4.4]
+ 32 bytes for the first device (NOT including DSK0)
+ 16 bytes for each additional device
+ 60 bytes for each bad track device (including DSK0)

Note : Line "DEVTBL DSK1,AMS0,AMS1" has 3 (not 4,2,1) devices.
Note : Bad track devices currently include only Phoenix Disks

(5) BITMAP — For each BITMAP line without the "/S" option
+ 26 bytes (AMOS 4.0 to 4.2)
+ 34 bytes (AMOS 4.3 to 4.4A)
+ 2 x (size of bitmap in words)

Note : Any shared BITMAP line counts as only 1 BITMAP line above

(6) QUEUE — + 10 x (# of extra QUEUE blocks) (AMOS 4.0)
+ 16 x (# of extra QUEUE blocks) (AMOS 4.1 to 4.4A)

(7) SYSTEM prog — For each program made resident :
+ 12 bytes
+ Size of program

(-) SYSTEM — [ends all changes to monitor size]

SOME PROGRAM SIZES (in bytes)

Program Name	3.2	3.3	3.4	4.0.0	4.0.2	4.1	4.2	4.2.5	4.3	4.4	4.4.A
SYSTEM.MON	13122	13122	13122	13256	13236	13328	*13494	13494	13520	13778	13778
Base Size	?	?	?	?	8920	9012	*9178	9178	9334	9426	9426
Unused	?	?	?	?	4316	4316	4316	4316	4186	4352	4352
Disk Drivers											
HWK500.DVR	-	-	468	486	486	486	486	496	530	530	530
SMD410.DVR	-	-	-	-	-	-	-	542	570	664	664
PERAMS.DVR	472	472	478	478	478	478	*	*	*	*	*
PERSTD.DVR	476	506	512	512	512	512	*	*	*	*	*
WNGAMS.DVR	336	340	346	346	346	346	*	*	*	*	*
WNGSTD.DVR	370	374	380	380	380	380	*	*	*	*	*
200DVR.DVR	-	-	-	-	-	-	844	844	848	852	884
Other Drivers											
AM300.IDV	342	342	342	342	342	342	342	342	342	342	342
AM310.IDV	-	-	-	-	-	-	-	-	408	374	374
IMSIO.IDV	100	100	100	100	100	100	100	100	100	100	100
ADM3.TDV	268	268	246	246	300	300	300	300	300	300	300
HAZEL.TDV	288	288	294	294	278	314	314	314	314	300	300
SOROC.TDV	288	288	288	288	300	300	300	300	300	300	300
MEM.DVR	-	-	-	308	308	298	298	298	298	298	298
MTM.DVR	1028	1028	1028	1028	1028	1028	1028	1028	1028	1028	1028
RES.DVR	-	-	-	-	-	196	196	196	196	196	196
TRM.DVR	188	188	252	252	252	252	252	252	252	252	252
AMS,HWK,STD	(look up under original name under Disk Drivers, above)										
Other Progs.											
BASIC.PRG	11002	11056	11056	11198	11198	11332	11492	11492	11526	11756	11756
RUN.PRG	9658	10440	10502	10982	11000	11064	11234	11234	11310	11494	11494
DYSTAT.PRG	902	902	902	902	902	902	902	902	902	902	902
TODCNV.PRG	524	524	524	524	524	524	524	524	524	524	524
EDIT.PRG	2844	2844	2844	2994	2994	2994	2994	2994	3000	3030	3030
VUE.PRG	-	-	-	5960	8604	9310	9308	9308	9308	14314	14314
ISAM.PRG	4782	4660	4694	4846	3918	3912	4258	4258	4270	4270	4270
XLOCK.SBR	218	218	218	218	218	218	218	218	218	218	218
FLOCK.SBR	-	-	-	-	1018	1078	1078	1078	1078	1078	1078
BASORT.SBR	-	841	877	878	878	878	878	878	878	1064	1078
AMSORT.PRG	-	1886	1952	2018	2118	2118	2118	2118	2130	-	-
AMSORT.SYS							-	-	-	1702	1702
FLTCNV.PRG	-	498	498	498	498	498	498	498	498	498	498

Notes (*) : Monitor Sizes --- Some 4.2 Persci-AMS Monitors were released that were bad; they had a monitor size of 13364 bytes, and a base monitor size of 9050 bytes.
Driver Sizes --- From 4.2 on, all floppy drivers are created out of 200DVR.DVR and have the same size as 200DVR.DVR

letters

Gary W. Cage, M.D.
5411 W. Cedar Lane, Suite 102A
Bethesda, MD 20014

For some time I have had an intermittent problem which I think I have resolved and the information may be useful to other users.

I am running three terminals and two printers concurrently. The program to print the bills is generally run from terminal 1 or job1. As these are completed, it is then spooled to one of the printers. When the bill is set up, the bill program also enters a transaction on that patient's file to indicate that the bill was sent. Recently I have worked on the bill program so that it runs much faster. On recent billing days, however, we have found that several bills would be printed satisfactorily and then the system would go down. The spool program is set up to wait if insufficient queue blocks are available; however, I also use xlock when I am writing to the patient's file to indicate the bill has been sent. I finally figured that as queue blocks were being used up, when the locks were to be put into effect and no queue blocks were available, the system crashed. If this doesn't make sense to other users let me know.

At any rate, I modified the program to check for queue blocks before any locks were put into place. This is done by finding the number of queue blocks which is at location 144 and checking this before continuing. The follcwing is the small insert which I made to show me the number of queue blocks available and to wait until sufficient numbers were available. I arbitrarily picked 10.

```
20190 TERMFIL:
20192 QFRE=BYTE(144)
20194 PRINT QFRE; " Q FREE "
20196 PRINT TAB(-1,3);
20198 FOR X=1 TO 10:PRINT TAB(-1,5):NEXT X
20199 IF QFRE<10 GOTO TERMBIL
```

I am happy to report that in recent times the system no longer crashes.

I would like to take this opportunity also to say that it is desirable that the prices of software be given with information. Quite often the software is too expensive to consider whereas at other times it is a real bargain and to have to inquire to every supplier everytime you are interested is rather cumbersome.

The ads are helpful but I think that larger commercial suppliers should expect to pay a fee for entering their ads whereas small users passing information about software they have developed should not have to pay for entering the information.

I have usually been reticent to notify AMUS of problems that I have since often I think think it is some stupid mistake that I'm making in my own software. However, quite often I then read a letter which alerts me to the fact that the problem is actually Alpha Micro Software and not my problem. Readers should be encouraged to report their experience even if they are not certain of the cause since others user can then report similar problems and the situation looked into more vigorously if more than one person has a similar problem.

Thank you for all the fine work at AMUS.

Gary W. Cage, M.D.

* * * * * * * * * *

C L A S S I F I E D

TI810 PRINTERS

Brand New - Still in Original Cartons

Upper/Lower Case

$1,250 c.o.d. Paper Tray $15.00

Paul Eliseo
Basic Research
Youngstown, Ohio
(216) 782-8858

* * * * * * * * * *

CDC HAWK PACKS

Includes lifetime Warranty.

1 - 3	$90.00 each
4 - 7	$85.00 each
8 +	$80.00 each

Comtemporary Cybernetics
1204 Willowgreen
Newport News, VA 23602

* * * * * * * * * *

FOR SALE TO BEST OFFER

Alpha Micro 64K CPU, 6 Serial I/O Ports, 10mb CDC 9427 H Cartridge Disk, 2 disk packs, 2 Hazeltine 1500 terminals, 1 TI810 RO printer.

Harry Chalekian
1608 S. Main
Racine, WI 53403
(414) 637-6238

* * * * * * * * * *

TELETYPE MODEL 43 PRINTER

New with chess case, 12 extra ribbons and 2 cases of paper, manuals, etc. Original cost - over $1500. Will sell for first $1,000.

Eugene Levine, President
Gene Levine Associates
Levine Building
Edgewood, TX 75117
(214) 896-1681

* * * * * * * * * *

USED HARDWARE

3 - Persci 277 Drives	$ 750	each
2 - IMS Power Supply	125	each
1 - VDM-1 board and Sanyoi monitor	200	(both)
2 - AM200 Controller	250	each

All hardware was purchased new 3 years ago. Only one Persci drive and AM200 controller were used by client. The other hardware was used for development only and has not been used for about 1 1/2 years.

Lambda Biosystems, Inc.
7342 S. Alton, Suite F
Englewood, CO 80112
(303) 773-8484

* * * * * * * * * *

WANTED

Geneological Package for tracing family trees.

Fred Hougardy
Bakersfield Association for
Retarded Citizens
Bakersfield, CA
(805) 834-2272

* * * * * * * * * *

WANTED

1040 Income Tax Preparation Programs

Frank Scofield
Scofield & Scofield
United Bank Tower, Suite 800
Austin, TX 78701
(512) 476-8371

* * * * * * * * * *

WANTED

2 - Wangco Floppy Disk Drives

Ray Downen
Ozark Computer Services
Box 1065
Joplin, MO 64802
(417) 782-0814

* * * * * * * * * *

WANTED

Wangco Floppy Disk Drives; AM-100 CPU's

Gary S. DeCamp
Western Computer Systems, Inc.
P. O. Box 10191
Eugene, OR 97440
(503) 485-4222

Sometimes the Dragon Wins

If you would like your name to appear in the AMUS Member Directory, please fill out this form and return to Sharon Greene, AMUS, 934 Pearl Suite B, Boulder, CO 80302. Regardless of whether you want your name to appear, please be sure and fill out the information about your dealer and send it to Sharon.

NAME _____

COMPANY _____

ADDRESS _____

CITY _____ STATE _____ ZIP _____

COUNTRY _____

PHONE _____ DATE _____

COMMENTS:

MAJOR INTERESTS _____

_____

_____

AREAS OF EXPERTISE _____

_____

_____

ADDITIONAL COMMENTS _____

_____

WHO IS YOUR DEALER?

COMPANY _____

ADDRESS _____

CITY _____ STATE _____ ZIP _____

COUNTRY _____ PHONE _____

CONTACT PERSON _____

COMMENTS: _____

_____

Index, July, 1981

ALPHA MICRO USERS SOCIETY MEMBERSHIP FORM

Please fill out as much information as possible.

Name _____ Company _____

Address _____ City _____

State _____ Zip Code _____

Business Phone _____ Home Phone _____

Circle one: Own Lease Thinking

Check all applicable: Dealer ___ OEM ___

User: Corporate ___ Individual ___

Describe equipment: _____

_____

AMUS may use my name for mailing lists ___

Make checks payable to AMUS

Annual dues are $35.00 per member.

For more information call Sharon Greene at 303/449-6917 or write AMUS.

Second Class Postage paid at Boulder,
Colorado 80302, ISSN 0273-8708, USPS 567-330.

AMUS
934 pearl, suite b
Boulder, CO 80302