A2-Central™

formerly Open-Apple

January 1993
Vol.8, No.12

A journal and exchange of Apple II discoveries

ISSN 0885-4017
newstand price: $3.00
photocopy charge per page: $0.20

BASIC: The Peoples' Language

by Stephen Brunier, Ron Lewin, and Morgan Davis

Talking to the Computers. The development of computer languages can be broken down into three stages. The earliest computers were programmed in binary code. Binary code is simply the zeroes and ones that make up the actual language of every computer from small micros to the largest mainframes. The earliest computer programmers were forced to program a computer in the language the computer *understood*, and not in a language that was natural for the human programmer.

It became apparent that much of the work done by the programmer could be done by the computer. This lead to the development of assembly language. By using an assembler, the programmer didn't have to worry about about memory addresses or what value caused a particular processor action to occur. The programmer could work with things most familiar, and the assembler would make the necessary conversions for the computer. Assembly language was stage one of computer language development.

Although this was a major achievement, there were still serious shortcomings. No matter how good assemblers became, they still required the human to have an intimate knowledge of the computer and required a great deal of time, energy and skill in writing the programs.

The second stage in program language development, the creation of higher level languages, solved many of the problems inherent in assembly language. The first of the higher level languages, Fortran, was developed by IBM in the middle '50s at the cost of about 300 man years of labor. Fortran was intended for use in science and math and was little more than what its name said it was, a formula translator.

A major breakthrough had been made, however. Higher level languages were possible and techniques had been developed that allowed for more sophisticated computer languages. Programs could be written in something that looked a lot like a human language; knowledge of the computer hardware was no longer required. Other programming languages were soon to follow.

In 1960 the computer language Algol was defined. Although Algol is little used today, it was to become the godfather of structured languages. Structured languages were developed because it was discovered that a well-defined programming structure greatly reduced the total cost of software development. Structured programming was the third major step in computer language development.

The best known of the direct descendants of Algol is Pascal. In fact, Pascal is so similar to Algol that we're not certain it is really another programming language. C, although significantly different from Algol, still retains the basic programming structure originally defined in Algol and is probably the most widely used programming language today.

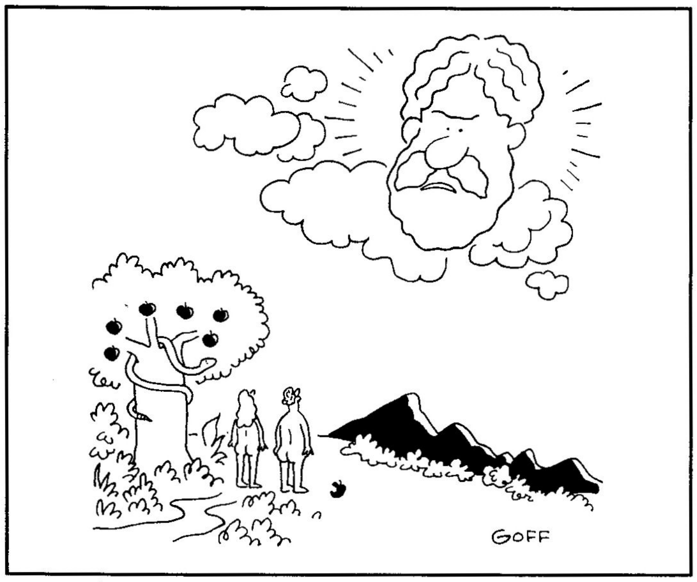
BASIC was defined in the middle '60s as a teaching tool before the concept of structured programming had caught on. The creators of BASIC had created a language that was so easy to learn that learning the language itself didn't complicate the learning of computer programming significantly. Unfortunately, the creators of BASIC also created a programming language with a serious shortcoming; programs written in BASIC where difficult to decipher.

The main purpose of this brief history of programming languages is to point out their primary function; that of communication devices between the human and the computer. Without these devices, computers would still be coded in binary. For a great number of people on the Apple II, BASIC is this communication device.

Why is BASIC so easy to learn when compared with Pascal and C? Probably because BASIC comes closest in structure to a naturally spoken language. When we first learned to speak, we weren't concerned with grammar. We learned words and what they meant; sentences followed naturally. This is the manner in which BASIC is constructed. With Pascal and C we must first understand some basic grammar before we can even begin to program.

Here's an example. Let's take the English sentence, "The little boy went to school on a bicycle," and write this sentence as it would be written, given the rules of Pascal and C.

```
Start of Discussion.
Start of Paragraph.
Start of Sentence.
Nouns: boy, school, bicycle.
Verbs: went.
Adjectives: The, little, a.
```



"AND IN ADDITION, YOUR PUNISHMENT WILL INCLUDE POWER SURGES WHILE YOU SAVE FILES."

```
Prepositions: to, on.
The little boy went to school on a bicycle
End of Sentence.
End of Paragraph.
End of Discussion.
```

In BASIC, the actual sentence is all that's necessary while in Pascal and C much more must be explicitly stated. While this has distinct advantages in a programming language, it also makes the language much more difficult to learn. Imagine how difficult English would have been to learn if you had to know what a noun was before you could say "mama" or "papa." BASIC doesn't place this burden on the "infant" programmer.

For a time, BASIC lost out to Pascal and C. Although Pascal and C are more difficult to use, they offered more advanced features, such as special structured programming commands. Structured programming did away with line numbers by allowing the flow of the program to be controlled by special loops and subroutines. Structured programming techniques gives the programmer greater organization within a program and better control over program flow.

For example, a typical Applesoft BASIC program looks like:

```
10 HOME
20 INPUT "Type in any animal's name: "; A$
30 IF A$ = "DOG" THEN PRINT "I like dogs.": GOTO 20
40 IF A$ = "CAT" THEN PRINT "I don't like cats." : END
50 PRINT "The ";A$;" is a good animal.": GOTO 20
```

An equivalent "structured" program is:

```
HOME
REPEAT
   GOSUB Ask_Animal
UNTIL Animal$ = "CAT"
END
ROUTINE Ask_Animal
   INPUT "Type in any animal's name: "; Animal$
   IF Animal$ = "DOG" THEN BEGIN
      PRINT "I like dogs."
   ELSE BEGIN
      IF Animal$ = "CAT" THEN BEGIN
         PRINT "I don't like cats."
      ELSE BEGIN
         PRINT "The ";Animal$;" is a fine animal."
      ENDIF
   ENDIF
RETURN
```

While the Applesoft program is smaller and more compact, it can be harder to follow. This might not be apparent in the short program above but in much longer programs it soon becames obvious, especially if you must return to work on a program 6 months after you've written it. It is much easier to remember what "GOSUB Ask_Animal" does than what "GOSUB 10203" does. This example also illustrates some other advantages of structured programming, such as the existence of special flow control commands like IF/THEN/ELSE, REPEAT/UNTIL, and many more.

One other major point of structured programming styles is the existence of local and global variables. Local variables are variables whose values can only be seen and referenced within a specific part of a program, often called a PROCEDURE or FUNCTION. Global variables are variables whose values can be seen, shared, and referenced throughout an entire program, including within FUNCTIONs and PROCEDUREs. The advantage of procedures and functions using local variables is that they create independent programs-within-a-program. Within these programs you can use whatever variables you want without fear of using the same variable somewhere else. Variables can be passed between PROCEDUREs and FUNCTIONs (to give parameters to, or receive results from them) using special commands. This allows you to make libraries of PROCEDURE and FUNCTION modules that can be brought into a program as needed to perform special tasks. An example of this:

```
PROGRAM "Divide"
GOSUB Get_Input [ "Nominator", Nominator ]
GOSUB Print_Number [ Nominator ]
REPEAT
   GOSUB Get_Input [ "Denominator", Denominator ]
   GOSUB Print_Number [ Denominator ]
   IF Denominator = 0 THEN PRINT "Denominator may not be 0."
UNTIL Denominator <> 0
Answer = INT (FN Divide [Nominator, Denominator])
PRINT "The integer result of the division is: ";Answer
END
PROCEDURE Get_Input [ GLOBAL Name$, Value ]
   PRINT "Please enter the "; Name$
   INPUT Value
END PROCEDURE
PROCEDURE Print_Number [ GLOBAL Name$, Value ]
   PRINT "The "; Name$; "is "; Value
END PROCEDURE
FUNCTION Divide [ Numerator, Denominator ]
   Result = Numerator / Denominator
END FUNCTION [ Result ]
```

Get_Input and **Print_Number** are two PROCEDUREs which perform operations on the variables passed to them within the [ ] brackets. Get_Input asks the user to enter a number, and returns the number in the variable Value%. Because Value% was designated as a GLOBAL variable, the main program can see the result. However, if it was designated a LOCAL variable, the main program would not see the changes made within the PROCEDURE.

FUNCTION Divide operates somewhat differently. FUNCTIONs are meant to be used within mathematical formulas and return a mathematical result. So when FN Divide [Numerator, Denominator] is invoked in a formula, the value of the variable passed back at the end of the FUNCTION (in this case called "Result") will replace the FN Divide in the original formula.

Although the structured program in this case is far longer than an equivalent Applesoft program, you can see the benefits in this style of programming for longer and more sophisticated programs. In fact, this highlights one of the major differences in Applesoft and structured languages, so this is a good time to start looking at Applesoft again.

Applesoft is probably the most popular language for the Apple II because it is built in to almost every Apple II ever made and it is fairly easy to learn. It's the first choice for most people who begin to dabble in programming. There are plenty of free programs available in Applesoft and it is the "common language" of Apple II programmers everywhere. I doubt there is any Apple II programmer who can say they didn't start programming in Applesoft.

However, significant advances have been made in programming languages and the capabilities of your Apple II since Applesoft was introduced in the Apple II-Plus over 12 years ago. While native Applesoft is easy to learn, it is severely limited. It lacks the structured programming capabilities of most modern programming languages and it can't access the advanced features of today's Apple II. We wouldn't be surprised to hear its line editor has driven people to inflict grievous harm upon their poor computers.

Applesoft is also an interpreted language; as Applesoft executes your program, it translates your program, command by command, into the real instructions your computer has to execute at the machine language level. This makes debugging easier but it significantly slows down execution. Even a simple loop like:

```
10 PRINT "HI MOM"
20 GOTO 20
```

will cause Applesoft to re-translate each line and re-execute the translated instructions each time through the loop. Most other languages today are compiled; before you can run your program, the computer converts the full program to machine language. Once the conversion is complete, the program can execute more quickly because there is no translation being done while the program is running. Compilers also have techniques to optimize compiled code, enabling your program to run more efficiently. The drawback is that a complier introduces a delay for the programmer, who must compile a program repeatedly during development and debugging. Compilation can take anywhere from a few seconds to a few minutes depending on the size of your program. Compiled programs are also more difficult to debug because they cannot be interrogated during the running like an interpreted program can.

All this said, while Applesoft does have drawbacks, it is still my first choice for a quick and dirty programs, such as the "Divide" program above. Don't let others convince you there is anything fundamentally wrong with BASIC. On almost every computer today, modern BASICs like QuickBASIC are still some of the most popular programming environments around. On the Apple II, BASIC has been given a new lease on life through third-party products which have emerged to address all these issues for the BASIC programmer and extend the power found in languages like C and Pascal to the BASIC environment.

Today there are many choices for the BASIC programmer on the Apple II; some based on Applesoft, and some which go beyond Applesoft. There is no room to discuss all the products available; we're going to try to cover some of the main ones which are still sold and supported.

One large group of tools for BASIC programmers consist of add-ons to Applesoft based on Applesoft's "&" command. The ampersand can access machine language routines to perform available functions. Tools based on the & command extend BASIC to allow access to extra memory, access to Super Hires graphics and sound on the Apple IIGS, and other specialized functions. However, these types of tools still operate under the general limitations of Applesoft. There are many publishers of these kinds of tools and you can often see advertisements for them in the mail-order pages of Apple II magazines.

If you're simply looking for a way to speed up an existing Applesoft program, you may want to look at the Beagle Compiler, available from Quality Computers or Resource Central. It can compile straight Applesoft programs into machine language, making them much faster. The main drawback to the Beagle compiler is that it doesn't really add any capabilities to Applesoft and is incompatible with the & based products which do.

***Micol Advanced BASIC (MAB),* is a tool for BASIC programmers that does not operate within Applesoft at all.** *MAB* is a complete development environment for structured BASIC programming that includes a full screen editor, compiler, linker, and utilities. There are two versions of *MAB*, one for the Apple IIe/c (128k enhanced required) and one for the Apple IIGS. Both versions allow BASIC programmers to write fully-structured BASIC programs (the examples above are actual *MAB* programs) using the full screen editor, and then compile them to machine language using the compiler/linker. *MAB* offers all the features of high-level structured BASICs including local and global variables, optional use of line numbers, libraries of PROCEDUREs and FUNCTIONs, integer, real and boolean variables, and more. Each version of the language offers commands that allow full access to the features of the Apple II they are made for. For example, the Apple IIe/c version allows up to 70k combined variable and pro gram space (more than twice Applesoft's limit), and access to double high resolution graphics. The IIGS version has mouse and windows commands, super hires graphics commands, multiple voice sound commands, access to the toolbox and GS/OS, and much more. In both cases, the language is designed to function as closely to Applesoft as possible to make it easy to learn and use. For instance, on the IIGS version the command to plot a dot on the screen in super hires graphics is HPLOT. Sound familiar?

Both versions of *MAB* are an integrated environment for editing, compiling and debugging programs. From the editor, two keypresses will save your program, automatically compile and link it, and start executing it. If there's an error, you'll be returned to the editor with the cursor on the offending line.

The main criticism of our product has been that it operates under its own shell and does not use the *Orca* shell from Byteworks. This means that *MAB* programs cannot be linked into programs written using the Byteworks products. We acknowledge this limitation but feel that our own shell is far easier to use, and this ease of use outweighs any drawbacks to the 3 percent of users who may want to link programs written in *MAB* and *Orca* products together.

It is our opinion that *MAB* offers the best features for Apple II programmers who want the easiest, fastest path to writing high level structured programs taking advantage of all the features of modern Apple IIs. If you're already a BASIC programmer, *MAB* builds on your knowledge and adds to your power, unlike Pascal and C, which require you to throw away all your Applesoft experience and start from scratch.

We feel it is important to point out that we (the authors of this article, except for the following section about *MD-BASIC*, which was written by Morgan Davis) are the authors of *Micol Advanced BASIC.*. We'll avoid the temptation to monopolize this whole article with information about *MAB*, and allow Morgan Davis to give you information about *MD-BASIC*, his fine BASIC development tool:

There are varying degrees of sophistication from which to choose when programming in BASIC on the Apple II. You can use simple Applesoft BASIC, as thousands have, or a highly structured, compiled BASIC. *MD-BASIC* fits somewhere in between. It offers features found in both, while eliminating the drawbacks inherent in Applesoft. But with all of its power and modern features, *MD-BASIC* creates ordinary Applesoft BASIC programs. If that is all you need, there is no better tool than *MD-BASIC*.

The typical development cycle of an *MD-BASIC* program goes like this: First, create your source code using the powerful features of an editor. Editing is integrated into *MD-BASIC* (or you can use any editor you prefer, including AppleWorks). Next, *MD-BASIC* translates your source into Applesoft BASIC, only the output is highly optimized. Your program is smaller and faster than one created the old fashioned way. Finally, you capitalize on the immediacy of Applesoft for testing and debugging. When done, issue the BYE command to return to *MD-BASIC*. It's that simple.

***MD-BASIC* offers two working environments: a desktop interface and a command line mode.** The native Apple IIGS desktop interface sports multiple windows for editing, an integrated source translator, and a BASIC-to-source converter that creates source code from your existing BASIC programs. The command line mode gives *ORCA* or *GNO/ME* shell users access to *MD-BASIC's* features through typed commands or shell scripts that automate software development. You get the best of both worlds with one package.

Developing programs with *MD-BASIC* offers many advantages. You program in a familiar environment, using a modern, high-level BASIC language (illustrated earlier in this article). The programs you create can be run on any Apple II. They are optimized to save memory and execution time. They can incorporate extensions to the BASIC language, provided by a plethora of ampersand commands. And, they are compatible with the *Beagle Compiler* and *RADE*, the Real-time Applesoft Debugging Environment.

***MD-BASIC* requires an Apple IIGS for development only.** The Morgan Davis Group provides phone and online support through these services: GEnie, America Online, BIX, the Internet and the Pro-Line network. Check these services for a free demo of MD-BASIC, written in MD-BASIC with source code included.

We hope you've learned something from this article, and that is: No matter if you program in Applesoft, *MAB*, *MD-BASIC*, or use any other BASIC tool, BASIC is still one of the best choices for programming the Apple II. It is easy to use and learn, can be powerful and is always fun. Just as our trusty Apple II has evolved and can still do everything most other PCs can do (and often better!), BASIC has evolved and can still do everything most other languages can, often better and easier!

The Morgan Davis Group can be reached at 10079 Nuerto Lane Rancho San Diego CA 919 77-7132 619-670-0563, 619-670-9643 Fax, 619-670-5379 BBS, GEnie: Morgan-Davis and Micol Systems 9 Lynch Road Willowdale, Ontario Canada M2J 2V6 Tel: (416) 495-6864 Fax: (416) 731-1052 GEnie: Micol.System, Internet: ronl@ter-ranet.cts.com. *MD-BASIC*, *MAB* IIe/c & *MAB* IIGS available from Resource Central and other mail order sources.

Marketing Educational Software 2

by Phil Shapiro

When marketing educational software to an individual school or school district, the first challenge is to make contact with the person in the school or school district whose job is to supervise educational technology. Such first contacts are usually made by phone.

Thankfully, you don't need to go on a wild goose chase to locate the name of the educational technology coordinator (sometimes referred to as the Director of Computer-Assisted Instruction 'CAI'). A set of commercial reference books, the "CIC School Directories," published annually by Market Data Retrieval, lists the addresses and phone numbers of just about every public and private school in the nation. But that's just for starters: these treasure-trove reference books offer a cornucopia of other information that can assist you in your marketing efforts.

Most importantly, from the perspective of a software publisher, these directories give fairly specific information as to the number and models of computers being used at individual schools. Using these directories, you can target your marketing efforts at those school districts with more than 25 Apple II's at each of their schools.

The directories tell you the general number of Apple II's at a given school by classifying the number of computers into four groupings: 1 to 4, 5 to 9, 10 to 24, and 25 plus. If the school has a mixture of Commodores and Apples II, the predominant model of computer is indicated in boldface.

Thumbing through the directories, you'll see page after page of schools with 10-24 Apple II's. In the more affluent school districts, you'll see a whole column of elementary schools with 25+ Apple II's.

At the top of each school district listing the directories list the names of the major administrators for that school district. Each person's job responsibilities is indicated by one or more code numbers. (These code numbers are all listed on a flap at the end of the directory.) The person in charge of computer-assisted instruction is listed with a code number of 73 beside their name.

Calling the school district's main voice number will get you started in tracking these persons down. Sometimes these administrators work in the central administration office, in which case you'll be connected directly to their extension. But in many cases these persons do double duty working at one of the schools in the district. In such a case you'll have to try reaching them at their school location.

Once in a while the person listed as being in charge of CAI, is actually responsible for data processing in the school district. So before you launch into any conversations, it pays to inquire first as to whether you have indeed reached the person who handles educational software for the district.

You may be wondering, at this point, about the price of buying these "CIC School Directories." The complete set of 51 directories sells for a hefty $950. However, you can buy individual state directories for $33 to $60.

The less populous states, such as Wyoming and Hawaii, barely have 50 pages of listings in their directory. The more populous states, such as California, Texas, or New York, have several hundred pages of listings in their directories.

With California's strong support of computer-assisted instruction, it would make sense to target your first marketing efforts there. Other states with strong commitments to CAI include: Arizona, Oregon, Washington, Minnesota, Iowa, Michigan, Maryland, North Carolina, and Florida. (These states come to mind as being frequently mentioned in magazine articles about CAI.)

When you reach the CAI administrator for a school district, your first goal is to get the district to evaluate your software. The standard procedure is to offer to send a free evaluation copy, with no strings attached. (School districts are naturally weary about having to keep track of which materials need to be returned to which publishers.)

During your conversation with the CAI administrator, you might inquire as to the procedures the district uses to evaluate new educational software. Some districts pass the software around to various classroom teachers, asking each teacher to fill out an evaluation form. Other school districts evaluate new software by sending it to a committee of professional educators.

Large city school districts tend to have a more formal, bureaucratic procedure for evaluating new software. Smaller school districts tend to evaluate the software informally.

You would do well to include an evaluation form and self addressed, stamped envelope along with your software. Typical evaluation forms would give evaluators the opportunity to judge the software's educational content, design, interface, and documentation. Ending the form with a general "Comments" section can help elicit valuable feedback about the strengths and weaknesses of your software.

You can expect evaluations to take a minimum of two months. Some school districts may take as long as three to four months to complete an evaluation.

For further information about the "CIC School Directories," you can contact Market Data Retrieval at: 16 Progress Drive, Shelton, CT, 06484. Phone: 1-800-243-5538. Market Data Retrieval also offers "mailing list" direct marketing services, where their company will assist you in sending out your literature to a targeted set of school districts.

If you live near a big city library, it's possible that the library may have a copy of these "CIC School Directories" on their reference shelf. Another place to look would be at a college "school of education" library.

(The author is the founder of Balloons Software (5201 Chevy Chase Parkway N.W., Washington, D.C. 20015, 202-244-2223), a new educational software company. The company's first product, Number Squares, has been licensed to three school districts. The company's second product, Big Text Machine, a language arts creativity toolkit,

Miscellanea

Apple sinks ship but fishheads seen floating to surface. The late breaking but not totally unexpected news from Apple Computer, Inc. is that as of December 15, 1992, the Apple IIGS will be removed from Apple's price list. It's sad and then again it's not. Equate it to the death of a loved one who has spent years of pain and suffering. Most dealers did everything in their power not to sell one anyway. When was the last time you saw an Apple II on **your** dealer's shelf? So now when they tell you that Apple doesn't even make the Apple IIGS anymore they won't be talking nonsense. It's a fact. But, in a sense, all that's changed is the price list.

The Apple IIe, on the other hand, is still being manufactured and sold! And don't let your dealer tell you otherwise. How many of you knew in 1984 that the 128K Apple IIe would outlast the 128K Macintosh by more than half a decade? (We did! We did!)

And we still have third-party developers like Seven Hills, Econ Technologies, WestCode, CV Technologies, Big Red Computers, Quality Computers, and more. We still have newsletters, magazines, user groups and online support. We still have the machines. And, oh, we still have Bill Heineman.

Despite this decision, Apple wants to show its support to the established base of Apple IIGS users. One way they are going to accomplish this support is with a brand new issue of the Apple II Software newsletter that has been edited by Cynthia Field for the past two years. Previously, it was available only through user groups, but this year it will be advertised in *InCider* Magazine at Apple's expense and made free for the asking. The newsletter is a compilation of Apple II software packages currently available with sources to purchase them.

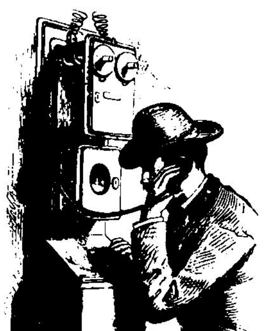
One major source of frustration for Apple II owners has been the almost nonexistence of dealers who were willing or able to repair their computers. It's been reported that Apple is considering addressing this concern with a service plan along the lines of the Powerbook program. It would be a mail order type of repair service for Apple II's. You put your broken computer back in the box and ship it off to Apple. They repair it and send it on back. Other future points of light in the future include the upcoming System Software 6.0.1 update, System 6.1 (but probably not beyond that), the Ethernet card, and the existence of the Apple II Continuation Engineering Group at least for another year. Add to that the fact that the Apple direct mail catalog includes Apple II hardware, peripherals and third-party products and you can't yell too loudly. They didn't **have** to do anything.

Nevermind! Maybe to help balance the Apple IIGS news comes the good news from *InCider* that their plans to greatly diminish Apple II coverage was a bit premature. The word on the street now is that they will indeed continue to cover the Apple II, only the format of the magazine will change with the March 1993 issue. From what I can gather, Mac coverage will be encapsulated in its own section, making it easier to locate the parts of the magazine that are relevent to your computer.

Coming soon to a hard disk near you is one of two utilities designed to compress the data on your hard drive, thereby giving you more space for the money. Econ's *AutoArk* (P.O. Box 195356, Winter Springs, Fla. 32719 is already shipping with mostly great reviews. A few bugs were discovered and squashed. Seven Hills' contender is called *HardPressed (2310 Oxford Road, Tallahassee, Fla 32304)* and is just about ready to go into beta. It will be interesting to see the two in a ring once they're both ready for action. In a time when support is supposedly dwindling here are two companies with products we sorely need.

Help wanted. The Cooper Institute for Aerobics Research (12330 Preston Road, Dallas, Texas, 214-701-8001 voice, 214-991-4626 fax) is looking for an experienced Apple IIe programmer to do subcontract work. The project involves writing/converting a character-based software package that runs under MS-DOS to run on an Apple IIe/IIGS. The program is a physical fitness testing and reporting package designed to be used in schools and other youth agencies. If you're interested in this project contact Dr. Marilu Meridith at 800-635-7050 or at the above numbers.

Apple II users are a fervent and diverse crowd. Reports of the death of the Apple II have been going on for years and it's been quite a roller coaster ride. You know what? The ride isn't over. Apple may have stopped selling the IIGS but we're still using them. Resource Central and many other companies are still supporting them. ***A2-Central*** is and always will be an Apple II newsletter. It's an everchanging newsletter but Apple II all the way. We get letters from all over the world, complementing us for articles that are technically oriented, criticizing us for articles that are not technical enough, or criticizing us for articles that are too technical to understand. We get feedback from some that we spend too much time and space on Apple IIGS material and too little space on the 8-bit stuff. And visa verse. Because we try to cater to the needs of our subscribers, you're bound to find something different each month. If there is something you would like covered, write and tell us. That's the only way we'll know. — edr



Ask (or tell) Uncle DOS

The following message was taken from the A2 RoundTable on GEnie where it was posted following an emotional response to the report that Apple Computer, Inc. intends to drop the Apple IIGS from its dealer price list as of December 15, 1992. Lunatic was present at a meeting between members of the Bay Area Apple II User Group, the User Group Connection, and John Santoro, marketing representative for Apple USA's K-12 group.

Mostly the facts

Now that the cat's out of the bag, allow me to add a bit more information that I gleaned from that meeting and from talking to John Santoro later that evening at a user group meeting.

Apple wanted to do a IIGS Card for the Mac LC and looked into it, but found that they couldn't make one for less than the cost of the Mac itself.

Apple will not license the Apple II ROMs to a third party for three major reasons: 1) It is proprietary technology, 2) Apple is now has a competing product to the Apple II, the Mac LC, 3) The Apple logo must be licensed with the ROMs. Any action that a third party takes while using the Apple logo reflects on Apple Computer, Inc.,whether good or bad. This is unacceptable.

No new Apple II hardware is expected to be developed by Apple Computer, Inc. after the Apple II Ethernet card is released.

System software tweaks such as printer drivers for new Apple printers are expected to continue. One large thrust will be continued network and printer compatability.

HyperCard GS and Apple IIGS System Software 6.0 were done partially to give Apple IIGS users "a taste of the Mac" in the hope that they would eventually migrate to that platform.

Apple is actively investigating a PowerBook-style mail-in service plan for all Apple II users.

The much-rumoured "ROM 4" Apple IIGS was to have the following features: Built-in 40MB hard drive, built-in SuperDrive, 2 Meg RAM, System 6.0 tools in ROM, DMA SCSI port, HyperCard GS bundled with the machine. It would have retained the 2.8 Mhz processor speed of the previous versions of the IIGS.

Part of the delay in the development and release of System 6.0 was because of the cancellation of the "ROM 4."

A re-engineered Apple IIGS was investigated, solely to lower production costs (continued production could be justified for a longer time with lower costs) but it never even got past the idea stage.

Finally, with all this darkness, I'd like to add a little light. Regardless of the actions of Apple Computer, Inc., A2 and A2Pro, the Apple II RoundTables on GEnie, have pledged to do everything we can to support you, the Apple II user, for as long as you continue to call this system. We will be here, as strong as ever, and in fact growing in size as we continue to add direct online support from more third-party Apple II companies, for many, many years to come. As long as you, the users of A2 and A2Pro, continue to support us by your presence here, we will support you in any way we can. There is no question as to where **our** loyalties lie. We all love our Apple IIs, and we will not give them up! Apple II Forever!

Lunatic E'Sex
Head Sysop in Charge of Promotions
A2 and A2Pro RoundTables on GEnie

(Permission is granted to reproduce this message as long as it has not been changed or edited in anyway.)

Hardware is what they call it...

I need IIGS diagnostic software!!!

J. B. Duvall
Beijing, China

Short question, short answer: there isn't any. Well, not much, and it might not help you beyond what you may already have.

The IIgs equivalent of a PC's POST (Power-On Self Test) is activated by using control/option/open-apple/reset. Let this run for a few minutes and it will cycle through testing of most of the machine's components. Chances are that if it doesn't catch something, then the system itself isn't broken. What's missing from the equation is the meaning of the codes the diagnostics return, which we hereafter paraphrase from Apple II IIGS Technical Note #95, 'ROM Diagnostic Errors'.

***There are basically twelve tests.** If all tests are completed successfully then you hear a series of tones and see a "System Good" message. Otherwise, you should see a message of the form "System Bad: AABBCCDD" at the lower left hand side of the screen and another (staggered) copy of the AABBCCDD code at the upper right which is there in case a RAM problem muddles the lower left display.*

The 'AA' is the test number. The remaining codes vary in meaning for the tests, which are:

***AA = 01: ROM test.** BB will indicate the checksum at failure. DD will be 01 if the test ran into bad RAM in which case the error code will be reported similarly to the RAM Test error codes.*

***AA = 02: RAM test.** BB is the bank number of the failure (or $FF if an ADB Tool call failed) and CC is the bit pattern at the failure. (See also ROM test above.) Also notice that the test checks only motherboard RAM which is soldered in on the IIgs.*

***AA = 03: Soft switches and state register test.** BB will be the state register bit, CC is the low byte of the softswitch address (in the $Cx page).*

***AA = 04: RAM address test.** BB is the bank number of the failure (or $FF if an ADB Tool call fails). CCDD will be the address of the failure within that bank.*

***AA = 05: Speed test.** BB will be 01 if the speed is stuck slow; 02 if the speed is stuck fast.*

***AA = 06: Serial test.** BB indicates the type of error; 01 for register read/write, 04 for transmit buffer empty status, 05 for transmit buffer empty failure, 06 for "all sent" status failure, 07 for received character available status, or 08 for bad data.*

***AA = 07: Clock test.** If DD is 01 then a fatal error occurred (test is aborted).*

***AA = 08: Battery RAMTest.** If BB is 01, then the address test failed and CC is the bad (battery RAM) address. If BB is 02 then the non-volatile RAM failed and CC and DD will contain the pattern and address at failure respectively.*

***AA = 09: Apple Desktop Bus (ADB) test.** BBCC is the bad checksum. If DD is 01 then the ADB tools ran into a fatal error and the checksum was not calculated.*

***AA = 0A: Shadow Register test.** If BB is 01 then text page 1 shadowing failed. 02 means text page 2 failed, 03 means there was an ADB tool call error (apparently a pretty common alternative for the diagnostics?), and 04 indicates there was an error in the Power On Clear bit.*

***AA = 0B: Interrupts test.** BB is 01 for a VBL interrupt time-out, 02 for a VBL IRQ status failure, 03 or 04 for a 1/4 second interrupt problem, 06 if it was the VGC IRQ, and 07 if it was a scan line interrupt glitch.*

***AA = 0C: Sound test.** DD is 01 for a RAM data error, 02 for a RAM address error, 03 for a data register failure, 04 for a control register failure, and 05 for an oscillator interrupt time-out.*

Before you scream "That doesn't tell me much" notice that in most cases these codes let you diagnose the subsystem where the problem exists. Given that most subsystems on the IIGS are integrated circuits that you can't build in your basement, that may be all the information that's pertinent. You can't 'repair' a bad Ensoniq chip; all you can do is replace it.

The things the Apple IIGS built-in diagnostics won't check are expansion devices; printers, modems, disk drives, and even any expansion memory you've added (the diagnostics only check motherboard memory). Since there's no way to anticipate what someone is going to stick into a slot or port, there's no universal way to check these items.

*The prominent exception is memory cards; obviously a "generic" memory checking utility is possible as far as finding unreliable memory. However, such a program can't identify which chip is bad on your board; doing this would require knowing how the board is laid out physically and manufacturers don't use the same layout (even some that look the same actually access the chips in a different "order"). I usually recommend the **MemoryTest.CL** application that comes with **AppleWorks GS**; it seems to be best at detecting problem RAM. Not everyone has (or needs) **AppleWorks GS**, and it would be nice if Claris would offer this utility separately for a few dollars. If not, maybe someone else will write one.*

*The other common problem areas are disk drives. The normal problems are disk speed on 5.25 drives and alignment. Disk speed for 5.25s can be checked with several available utilities including **Copy II Plus** (choose 'Verify" and then 'Drive Speed'); 3.5 drives are self-adjusting; if the speed is off, it means replacing the mechanism. Testing alignment requires an oscilloscope and an alignment disk to check and adjust properly; software methods are not reliable.*

*Apple had some dealer diagnostics, but in my experience they weren't particularly more useful than using the internal diagnostics and a memory test program. In all cases, we either ended up swapping subsystems or taking instruments to the system to fix it. These **are** the diagnostic tools when you have eliminated basic problems.*

Basically, "canned" diagnostics are only useful for determining a relatively limited set of common, easily interpreted problems. They're the computer equivalent of a car's gauges and warning lights. Beyond that, you (or a tech) has two choices: swap subsystems until the system starts working, and (where possible) troubleshoot a bad subsystem at the component level. On a car, most people usually take it to a mechanic at that stage. For computers, the equivalent is an electronics technician; unfortunately, finding one well-versed in Apple IIs is about as easy as finding a good Jaguar mechanic. There's not much we can do about that.

*Assuming you find (or are) a tech, the next step is documentation. Schematics for the IIGS are in the **Apple IIGS Hardware Reference** (the Second Edition has schematics for both the ROM 01 and ROM 03 motherboard versions). Schematics for the IIe and IIc (respectively) are in the **Apple IIe Technical Reference** or the **Apple IIc Technical Reference**. **Sam's ComputerFacts** also publishes troubleshooting schematics for several Apple II models and peripherals, and Jim Sather's **Understanding the Apple II** and **Understanding the Apple IIe** are excellent references for the theory of operation of the II and IIe hardware. Beyond that, Apple doesn't release much low-level technical information on their systems, and most third-party "troubleshooting" guides have disappeared because not enough people were buying them to keep them in print.—DJD*

Forever fonts

Concerning the Punjabi font inquiry in the November 1992 issue, there are two solutions. Our Punjabi friend can create his own font of choice using *GS Font Editor* from Beagle Bros or *Font Factory* from Seven Hills Software (Seven Hills Software, 2310 Oxford Road, Tallahassee, FL 32304-3930).

If he needs a ready-made font, he could buy one from Ecological Linguists, Box 1516, Washington, D.C. 20003. Order the font in True Type format for the Macintosh and then use *Pointless* (Westcode Software, 15050 Avenue of Science, Suite 112 San Diego, Calif 92128) to convert it into a suitable Apple IIGS font. The company carries fonts for various other languages.

Dr. Shree Datye
Akureyri, Iceland

Know any good monospaced fonts in multiple point sizes? Normal fonts or True Type fonts for use with *Pointless* would be acceptable. I write newsletters and sales lists and have found that monospaced fonts work out better.

Don Zahniser
Kent, N.Y.

With the help of Marty Knight of II Productive we located a variety of monospaced IIGS fonts. Apple M, Andover, Courier, Digital, Elite, Flow II, Hood River, IBM Klone, Kendall, Lamoni, New Monaco, Pica, Santa Monica, Tiny Font, and Wash DC , to name a few. Of those, I was only able to find Andover and Kendall in multiple point sizes. All is not lost, however. ***Font Factory GS*** *(Seven Hills Software 2310 Oxford Road Tallahassee, Fla. 32304-3930) will allow you to scale a font to create a new size. If anyone out there knows of some TrueType fonts that are monospaced, let us know about them.–edr*

More on midi

Thank you for a fine article on the IIGS and MIDI by Dennis Doms. I would like to add a few things, if I may.

He mentions the volume item in synthLAB's "Setup" menu and the fact that you can select the output volume for each of synthLab's sixteen possible instruments. I have found that when playing back your sequence from the Finder Extra, synthInit, that these volume settings are ignored. This may be true for other Midisynth sequence players as well, but I don't know that for sure. The way around this is to make the volume adjustments on the instruments your sequence uses in synthLab's instrument editor and then save the instrument file.

Also. he writes that synthLab wasn't meant to displace commercial programs. This is true but unfortunately there are no commercial (or shareware) sequencing programs available for the Apple IIGS. Passport has discontinued *MasterTracks Pro* and *MasterTracks Jr. MasterTracks Pro* was in need of an update before being dropped — it crashes when you try to access NDA's and suffers from a poor timing resolution.

Michael Nickolas
Lynn, Mass.

SynthLab's "volume" setting doesn't change MidiSynth, only the data (which is why you have to save the data to make the changes "permanent" for other programs.—DJD

Another basic

Does anyone out there still use *ProBASIC* by Alan Bird? It come on the back of *Program Writer.* I am new to ***A2-Central*** and recently bought all the back issues. I was pleased to see *ProBASIC* discussed in Volume 2 Number 8 page 62 b-c.

It is a very powerful version of ProDOS's BASIC system and has many awesome features not mentioned in the article. I would like to find the latest version with the bugs fixed and any of the module disks. I have tried numerous places and even attempted contacting Alan Bird but to no avail. Are there any other *ProBASIC* users who would like to exchange ideas and/or other modules?

On another note, in ***A2-Central***, Volume 5, Number 4, there is an article about Lego versus Fischer. Does anyone know the address or phone number for Fischer and if the *Fischertechnik* kits are still available?

Lane Wacholz
New Richland, Minn.

You'll be happy to learn that Alan Bird recently released ProBASIC and the accompanying module disk as freeware. They are only two of the thousands of available files waiting for you to download in A2Pro's library on GEnie. The file numbers are #2882 and #2883 respectively. If you don't telecommunicate or have a GEnie account, you might write to Dean Esmay and beg him to include it on our monthly disk.

This is probably a good time to reiterate the importance of modems and telecommunication for any platform of computer, especially for Apple II users. In a time when support for Apple II is dwindling, the online networks remain a vital and viable source for all sorts of technical information; as well as shareware, freeware and public domain programs. The technical expertise on GEnie (and other services like America Online, I'm sure), is incredible. It's not uncommon for someone to post a question to a complex (or not so complex) problem and have good, sound answers from a variety of people in less than 24 hours. For the tight-budgeted, most local user groups have bulletin boards set up that can be accessed with no fees or hourly rates involved. The answers to most technical questions can be only an electronic phone call away.

In my book, the acquisition of a modem and appropriate telecom software just might take precedence to even a hard drive. Bundles containing everything you need are available from us or any number of other mailorder companies. As many power-user types upgrade to the newer and faster modems available today, used 2400-baud modems become easy to purchase second hand. Aside from the classified section of your local paper, user groups would be another point of contact for such purchases. Of course, if you've got a friend who's already "modemized," the online services are probably the best place to find second hand equipment.—edr

Apple Writer minus & plus

I was quite astounded to see in the October 1992 issue of ***A2-Central***, two pages dedicated to reviving *Apple Writer* and the message that there will be more to follow. I feel that I may be at great risk here in offending those *Apple Writer* stalwarts but a voice for progress must be heard.

The Apple II certainly was, and many would argue, still is one of the finest machines to be placed on the microcomputer market. I would say that the main reason for this is the attempt to produce that "user-friendliness" so unknown in the PC world. There is no doubt that *Apple Writer* had its day and that it possesses many features that AppleWorks does not (well, at least not without splashing out for the add-ons). However, aside from the fact that it is now freeware, I feel that *Apple Writer* should be buried along with the dinosaurs, simply because there is a better package that will achieve all that *Apple Writer* does and more, whilst retaining that usability we get with AppleWorks. There was a letter about the *Fulltext Pro-80* package in January 1992 and since then I sent you some details of the package which I reviewed some time ago. It is not the purpose of this letter to expound the qualities of *Fulltext* but more to point out a few features which make it of more value than *Apple Writer* could ever be.

Firstly, the package boots into a main "control" screen via which all file, dictionary, comms, etc. functions are accessed. Pressing E(dit) takes you to the edit mode, which presents a blank screen as in *Apple Writer* except telling you to press Open-Apple-H for help (memory resident, unlike *Apple Writer*) and providing a text window for your control codes.

Ron Evry states that you have to be careful not to exceed the 1024k limit of the character buffer. In *Fulltext*, if it is exceeded, it asks you if you wish to save it to disk, or does so automatically if you possess a RAM disk. Ron states that a unique feature of *Apple Writer* is that it is memory resident. *Fulltext* is also entirely mem ory resident (including help screens) but what's more, takes advantage of any additional memory to load in any other relevant files to a RAM disk if available by using an internal macro. The files it loads are prefixed with PRO. so this not only allows you to select which modules you require but also enables you to lead in your other favourite data of program files at the same time.

Printing of *Apple Writer* documents was men tioned. You can have the most ideal word processor in the world, but if it doesn't give you a hard copy, what's the point? Printing has got to be *Apple Writer's* major downfall, simply because it treats all control codes as characters, screwing up line length, pagination and anything else you have carefully formatted. I remember trying patches to eliminate this but they were only partially successful. Using *Fulltext,* control codes do not have this attribute-i.e. typing Open-Apple-B for bold will not move the cursor on the edit screen but you will see the code entered in the "codes window" at the base of the edit screen (unless you specify the whole screen to be in "raw" mode). In addition, the text is formatted on the screen exactly as it will come out on the printer-yep, save those rainforests...

File saving and loading is also made easier by a G(et) command at the control screen so no messing about with typing in lengthy prefixes reminiscent of the old ProDOS quit routine!

Okay, to be honest, the only thing which Ron talks about that *Fulltext* cannot achieve is a "peek" at a document on disk without exiting the current document in memory. However, since you should always save your document anyway. and it is so quick to load it in again, not being able to "peek" presents no inconvenience.

As well as the above, the program boasts storing its data as ASCII files (with a header), has a database function (also using ASCII files) and, here's the crunch, is equally at home on the IIe/c as the IIGS. Your only problem is that it is now unavailable in Europe or the US. It is however widely used on Apple II's in New Zealand (its country of origin) and may be obtained from there (Spacific Software, P.O. Box 8035, Dunedin, New Zealand).

The Apple II is a great machine. Let's not bury it by loading up *Apple Writer.*

Greg Carson
Rhosesmor, Mold

Hey, Greg! I thought you said the purpose of your letter was not to expound the virtues of ***Fulltext Pro! Fulltext*** *sounds like a fabulous program but what good does it do us if you have to live in New Zealand to get it? After our last correspondence, I contacted Spacific Software in the hopes of obtaining a review copy. After a few letters and faxes back and forth, they informed me that they were not interested in promoting and supporting the product outside of New Zealand. If you can sell them as well as you can sell us, I'd still be interested in seeing it. We get lots of requests from people looking for a good word processor that isn't as pricey as AppleWorks.*

Apple Writer *is available and it's free. What more can one ask? We've gotten lots of letters from people with positive feedback on* ***Apple Writer*** *in general and our coverage of it. We've gotten only one piece of negative feedback and you just read it. <grin>*

The last couple of issues of A2-Central have been quite nostalgic. I started writing a book on multivariate statistical analysis in 1985. When I finally finished it five years later, I was still using *Apple Writer IIe* with patches by Minuteware and Don Lancaster (remember *Stretchifier?)* and Applesoft, augmented by Microsparc's *Ampersoft.* All of this under DOS 3.3. As your readers have correctly noted, there are better things out there now but those seemed great when we were using them. (Ampersoft is still pretty good when you are working with matrix algebra routines.)

For the record, all of this still works on the Mac LCII, with the exception of some patches.

J. Edward Jackson
Rochester, N.Y.

A2-Central™

© Copyright 1992 by
Resource Central, Inc.

Most rights reserved. All *programs* published in ***A2-Central*** are public domain and may be copied and distributed without charge. Apple user groups and significant others may obtain permission to reprint *articles* from time to time by specific written request.

Publisher: **Tom Weishaar**
Editor: **Ellen Rosenberg**

with help from:
Denise Cameron, **Dennis Doms**, **Sally Dwyer**, **Dean Esmay**, **Mary Johnson**, **Jeff Neuer**, **Becky Smith**, **Jean Weishaar**

A2-Central,—titled ***Open-Apple*** through January, 1989—has been published monthly since January 1985. World-wide prices (in U.S. dollars; airmail delivery included at no additional charge): $34 for 1 year; $60 for 2 years; $84 for 3 years. All back issues are currently available for $2 each; bound, indexed editions of our first six volumes are $14.95 each. Volumes end with the January issue; an index for the prior volume is included with the February issue.

The full text of each issue of ***A2-Central*** is available on 3.5 disks, along with a selection of the best new public domain and shareware files and programs, for $90 a year (newsletter and disk combined). Single disks are $10.

Please send all correspondence to:

A2-Central
P.O. Box 11250
Overland Park, Kansas 66207 U.S.A.

A2-Central is sold in an unprotected format for your convenience. You are encouraged to make back-up archival copies or easy-to-read enlarged copies for your own use without charge. You may also copy ***A2-Central*** for distribution to others. The distribution fee is 20 cents per page per copy distributed.

WARRANTY AND LIMITATION OF LIABILITY. We warrant that most of the information in ***A2-Central*** is useful and correct, although drivel and mistakes are included from time to time, usually unintentionally. Unsatisfied subscribers may cancel their subscription at any time and receive a full refund of their last subscription payment. The unfilled portion of any paid subscription will be refunded even to satisfied subscribers upon request. OUR LIABILITY FOR ERRORS AND OMISSIONS IS LIMITED TO THIS PUBLICATION'S PURCHASE PRICE. In no case shall our company or our contributors be liable for any incidental or consequential damages, nor for ANY damages in excess of the fees paid by a subscriber.

ISSN 0885-4017 GEnie mail: A2-CENTRAL

Last time in *A2-Central-on-disk:*

Directory: /A2.ON.DISK.9211/

Filename	Blocks	Description
GENERAL.STUFF	1	For Apple II users
ASHAR.OS.BOOT	4	AppleShare Boot program
APPLE.HISTORY7	865.	Complete Apple II Hist.
GRAFORTH.COMMND	19	GraForth Command List/Help
GRAFORTH.PROG	116	Lost Classic Forth lang
HYPERC.SYSTEM	17	Correct, improved
		HyperC IIgs system
HYPERC.HGR.LIB	38	HiRes Graphics Library
RATIONAL.PERSON	123	Demo of The Rational
		Person program
IIGS.STUFF	1	For Apple IIgs users
BOOM.OOH.LAUNCH	29	OOW, neat sound & anima
		tion on launch
DOS.3.3.IIGS	49	Amazing DOS 3.3 hard drive
		util for GS
FIX.FIND.FILE	5	Fixes a bug in System
		6.0's FindFile
GS.SHRINKIT.1.1	209	GS ShrinkIt, IIgs archiv
		er, version 1.1
MAKIN.COPIES	46	Appropriate sound
SYS6.SOUNDPATCH	6	Fixes bug in System 6.0
		sound player

Last time in Studio City

Directory: /STUDIO.CITY.18A/

Filename	Blocks	Description
HOME.STACK	311	
HS.SYS16	288	
FS.DEMO	1	Flight School #2,
DEMO.12.92	21	The Auto Exit card
HYPER.SOUNDS	1	
TYPE.ONE.KEY	10	
WONKY.SYNTH	11	
KEYOW	27	
LETTER	1	
HELP	99	
TEXTURES	1	New ideas for creat
HYPERTEXTURES	545	ing backgrounds by
		--Vern Mastel
TTT	1	Learn stack design
TIC.TAC.TOE	109	and play games too
		--by Bill Lynn
ELEM.MATH	1	Quality time with
ADDITION.GAME	491	your kids and learn
ADDITION.STACK	401	SimpleScripting at
HELP.STACK	41	the same time--by
MATH.HOME	338	Joel Helton
START.HERE	70	
HYPER.ART	1	
AUSTRALIA	26	More maps of the
INDIA	26	world by Michael
MOONSCAPE	45	Giltzow
MOUNTAINS1	22	
WEST.INDIES	22	
WARTS.AND.ALL	31	Off beat stuff by
		M.A. Tryzna