So What Software™

Call Box Edition

Newsletter

Number 3

July 1990

What's new?

New functions are coming soon to Call Box BASIC. The new upgrade will be V2.1 and should be available in September. C.B.P.A. members have received V2.0.1 and will be getting V2.1b3 in their July mailing packet. This "beta" version will support some of the new features of Call Box BASIC.

The new features are Standard File Operations, GS/OS commands, Loader commands including the ability to load and run P16 shell applications directly from within Call Box BASIC programs and the Taskmaster command will be expanded to include double and triple click detection. There will be other additions and corrections as well but one of the several changes will be the startup time which should be nearly 2000% faster!

We hope to have a new launching shell completed by that time which works more gooder than it does now and the editors should support Merlin sourcecode output as well as some other function enhancements.

As with all Call Box upgrades this one is available only through C.B.P.A. (*Call Box Programmers Association*). The Call Box TPS version 2.1 should be available commercially sometime in late September or early October.---

We have a new "logo" for the 90's (*see the one at the top of this page*). The logo was changed for two reasons. 1. The original design was difficult to reproduce due to its irregular shape which made it hard to work into packaging designs. 2. The new style lends itself beautifully to laser output desktop publishing and fits better into generally rectangular packaging and promotion designs. It still has the same general look and feel of the old logo but we feel it looks more "*professional*" if there is such a thing.---

A new programming tool for Applesoft BASIC is on the market and it is called MD BASIC. This program allows you to write BASIC programs using symbolic references and extended language functions in any wordprocessor type of program including APW or its own editor. The really good thing about MD BASIC in our eyes is that it is totally compatible with Call Box BASIC and one enhances the functionality of the other... kind of a symbiotic relationship.

Now Call Box BASIC programs can be written which incorporate all the features of the MD BASIC language enhancements and can be written on your word processor. This gives you unprecedented power and flexibility.

MD BASIC is available from The Morgan Davis Group directly or from A2 Central.---

Morgan Davis Group
10079 Nuerte Ln.
Rancho San Diego CA. 92078-1736
(619)670-0563

A2 Central
P.O. Box 11250
Overland Park KS. 66207
(913)469-6502

SoftDisk Wants You!

SoftDisk Publishing Inc. wants to publish your Call Box BASIC programs. They pay actual money (well... checks) for software they use on their monthly Apple IIgs SoftDisk. Contact Jay Wilbur at SoftDisk on just how you submit software for consideration.

It is just my guess, but I think that games will be the big winner for publication on SoftDisk, Altho, any well written program will be a valuable asset. I can think of no better way for you to publish your work and get some money for your efforts in the deal without having to commit your life savings and all your time to publishing it yourself. Let's see... Getting paid for your work, that sounds real good!---

SoftDisk Publishing Inc.
(Jay Wilbur)
606 Common Street
Shreveport LA. 71101
(318)221-5134

I Knew That...

Congratulations to **John Jordan** from National City California. He was the first person to call with the answer for the April game contest. He answered "*Spinning Beachball Cursor*" and the acceptable answers are: Spinning Beachball Cursor, Beachball Cursor or Long Processes Cursor. This cursor was introduced to give you an indication that somethings going on when the computers off doing its thing. It is designed to say to you "*No... I haven't crashed, it's just that this process is taking a little time.*"

John wins an original game by Ed Rambeau called Roulette. As for all the others... better luck next time.---

I Did'nt Know That...

Many Apple IIgs owners are not aware of several built-in functions available in their computers. Many of these functions are poorly documented, if at all.

I will cover Some of these features to fill in the gap for those of you who have not yet acquired your own reference manuals.

I Did'nt Know That...*(continued)*

You press OPEN APPLE-CONTROL-ESCAPE to enter the CDA Menu. Minimally you will see Control Panel, Alternate Display Mode and Quit. There are 2 other built in Classic Desk Accessories in the IIgs that do not appear here, at least not yet. Quit the CDA Menu and from Applesoft type CALL -151 which will put you into the monitor. Next press a SHIFT-3 (the # sign) followed by RETURN and then press CONTROL-C to reenter Applesoft.

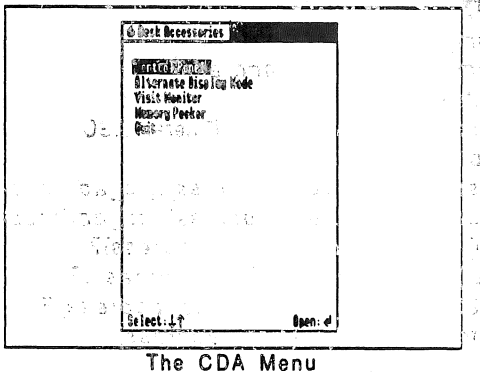
When you enter the CDA Menu again you will see two extra desk accessories listed... namely Visit Monitor and Memory Peeker.

These desk accessories add a new dimension to your programming environment. Having access to the monitor during the execution of your program and having the ability to get a readout on how the memory is allocated in your IIgs at the current time are very valuable assets for debugging and exploring your system and software.

If you are making a Call Box BASIC program you are making a program that utilizes all of the available memory and most of the internal operating system. Sometimes things do not work as planned (Did I say sometimes?) especially if you are reading, writing or shuffling around memory, which is what most all programs do. The problem in your program most probably lies in the fact that you clobbered (overwritten) some important piece of memory that you did not intend to overwrite.

The best way to find this kind of thing is to catch it in the act! If you have a problem like this make sure to invoke these two desk accessories at the beginning of your programming session and then while running your program and just before the problem occurs, enter the control panel and select the Visit Monitor desk accessory. From here you can examine any and all memory in the machine aside from being able to alter, move, search, compare and even assemble code, albeit in a limited fashion. All this is possible while your program sits in suspended animation, waiting for your return. You can examine variables, strings, interpreter code, global page variables, DOS etc. etc. and even change something behind the scenes in your program for experimental "what if" situations. When you are done fiddling around you press CONTROL-Y and your back in the CDA Menu Simply press ESCAPE followed by a RETURN and you are back in your BASIC program continuing on from where you interrupted execution to go into the CDA Menu.

You can interrupt your program as often as you wish without ill effects, and most times you will need to "intervene" at least twice... once to check or set something and once to examine the results after the something has been exercised.



The CDA Menu

The other hidden desk accessory is the Memory Peeker and its function is to list all the blocks of memory currently allocated by the Memory Manager. All of the memory in the machine is dolled out by the Memory Manager... actually you can do anything you want to any memory and the Memory Manager will not stop you from using "already spoken for" memory. Its function is to keep a list of the memory that applications (*you*) ask for and tells you if the memory that you want to use is already in use. This list of allocated memory blocks is what Memory Peeker displays.

I will cover the Memory Peeker in the next issue, as for this issue I will concentrate on the Monitor.

Visit Monitor-

This Classic Desk Accessory is the same as doing a CALL -151 from Applesoft. The system Monitor firmware is a program that you can use to create and test your own machine-language programs for the Apple IIgs. From the Monitor you can create programs that utilize various system resident subroutines as outlined in the Apple IIgs firmware reference manual appendix C. When you create your own programs or use the Monitor to examine programs that others have created, various features of the Monitor firmware assist you in your task.

The Apple IIgs Monitor provides commands that...

* Manipulate memory, including examining memory; entering changes in either ASCII or hexidecimal form; moving. comparing, or filling blocks of memory; searching for specific patterns.
* View and change the execution environment (microprocessor registers and flags)
* Execute programs from the monitor
* Step through and trace program execution
* Perform miscellaneous tasks such as setting the display to inverse or normal video, displaying or setting the time and date, redirecting input and output, performing hexidecimal arithmetic, returning to BASIC via cold or warmstart
* Invoke the mini-assembler
* Invoke the disassembler

You enter all Monitor instructions in the same format: Type a line on the keyboard and press return. The Monitor accepts the line using the I/O subroutine GETLN . A monitor instruction can be up to 255 characters followed by a carriage return.

A Monitor command can include 4 kinds of information: memory-bank number, addresses, data values, and command characters. You type addresses, memory-bank numbers and data values in hexidecimal notation.

The microprocessors in Apple II computers prior to the Apple IIgs could address memory only in an address range from 0 to 65535. The Apple IIgs, on the other hand, can address up to 256 banks or 65536 memory locations each. Thus there is a need for a memory-bank address qualifier for the Monitor commands. You will see the complete address represented as {*bank*/*address*}, where *bank* is to be specified as two hexidecimal digits and *address* as four hexidecimal digits.

When the command you type calls for an address, the Monitor accepts any group of hexidecimal digits, automatically providing leading zeroes to fill out the width of the field of digits.

The following is a list of the commands and syntax possible in the Monitor...

Terms

Destination	An address (with optional bank) that serves as a data destination
from_address	An address (with optional bank) at one end of a range of addresses
to_address	An address (with optional bank) at the other end of a range of addresses
start_address	An address (with optional bank) at which the monitor will start an operation
val	An 8-bit (1 byte) value specified as two hexidecimal digits
val16	A 16-bit (2 byte) value specified as four hexidecimal digits
val64	A value expressed as up to eight hexidecimal digits
val10	A value expressed as decimal digits
mm/dd/yy	Three 8-bit values separated by forward slashes
hh:mm:ss	Three 8-bit values separated by colon elements

Commands for viewing and modifying memory

Display single memory location	{*from_address*}
Display multiple memory locations	{*from_address*}.{*to_address*}
Terminate memory range display	Control-X
Modify consecutive memory	{*destination*}:{*val*} {*val*} {"literal ASCII"} {'flip ASCII'} {*val*}
Move data in memory	{*destination*}<{*from_address*}.{*to_address*}M
Verify memory contents	{*destination*}<{*from_address*}.{*to_address*}V
Fill memory (zap)	{*val*}<{*from_address*}.{*to_address*}Z
Pattern search (specified in four ways, can combine)	\{*val*}\<{*from_address*}.{*to_address*}P
	\{'123t}\<{*from_address*}.{*to_address*}P
	\{"literal ASCII"}\<{*from_address*}.{*to_address*}P
	\{*val16*}\<{*from_address*}.{*to_address*}P

Commands for viewing and modifying registers

Examine registers	Control-E
Modify accumulator	{*val16*}=A
Modify X register	{*val16*}=X
Modify Y register	{*val16*}=Y
Modify D register	{*val16*}=D
Modify DBR register (bank)	{*val*}=B
Modify program bank register	{*val*}=K
Modify stack pointer	{*val16*}=S
Modify process status	{*val*}=P
Modify machine-state register	{*val*}=M
Modify quagmire register	{*val*}=Q
Modify 16/8-bit accumulator mode	{*val*}=m
Modify 16/8-bit index mode	{*val*}=x
Modify native/emulation mode	{*val*}=e
Modify language card bank	{*val*}=L
Modify ASCII filter mask	{*val*}=F

Miscellaneous commands

Begin inverse video	I
Begin normal video	N
Change time and date	=T=*mm/dd/yy hh:mm:ss*
Display time and date	=T
Redirect input links	{*slot*}Control-K
Redirect output links	{*slot*}Control-P
Change screen display to text	Control-T
Change cursor	Control-^{*new_cursor_character*}
Convert decimal to hexidecimal	={*val*}
Convert hexidecimal to decimal	{*val*}=
Perform hexidecimal math	{*val*}+{*val*} {*val*}--{*val*} {*val*}*{*val*} {*val*}_{*val*} + = add -- = subtract * = multiply _ = divide
Quit monitor	Q

Farwell to Applefest ?

What if you gave a show and no one showed up? Well... that is just what happened at this springs Applefest.

The promoters (*Cambridge Marketing*) had a bright idea by putting on an Applefest that included Mac's Comodores and the dreaded IBM's. To rub salt in the wounds they changed the location to Sommerset NJ.... WHERE??? The response to this by the Apple II community was to not show up in protest, a tactic that seems to be all to effective. The turn out was a fraction of what the attendance has been for both the Boston and San Fransisco shows in the past.

No other computer in the history of computers has such a cult following as the Apple II has and the organizers of these shows had better understand this if they wish to continue this tradition.

Loyal Apple II people do not consider any other computer worthy of consideration, especially MAC's and IBM's... anyway these computers have their own shows and there is no reason to step on our toes. I guess that these people know little or nothing about computing and about loyalty which is non-existant in other computer lines.

All in all it sounds like unabridged greed to me and not any concern for the betterment of the art. To steal an old saying... If dynamite were brains they could'nt even blow their nose, and this is being nice.

Some unfounded rumors have surfaced that Sculley is pushing a yet to be released low end Mac that runs Apple II software (not GS software) and that this is intended to replace the II line. Once again Apple ignores the reason for its existance and attempts to tell us what's good for us and I personally will not listen to such treason! If Apple Computer Inc. wants to abandon all 4+ million of us fine... As far as So What Software is concerned we will continue to support the IIgs regardless of what corporate decisions come down from Cupertino. You can not fool with mother nature, and you certainly can not fool with the Apple II... take Applefest as an omen.

-- William Stephens

Miscellaneous commands (continued...)

Jump to coldstart BASIC	Control-B
Jump to warmstart BASIC	Control-C
Jump to user Vector	Control-Y
Set full native mode	Control-N
Go (begin) program in bank $00	{start_address}G
Execute from any memory bank	{start_address}X
Restore registers and flags	Control-R
Resume execution	{start_address}R
Perform a program step	{start_address}S
Perform a program trace	{start_address}T
Disassemble (list)	{start_address}L
Enter the mini-assembler	!

The mini-assembler

The Apple IIgs mini-assembler included in the Monitor program allows you to enter machine language programs directly from the keyboadr. ASCII characters or hex values can be entered into the mini-assembler program exactly as you enter them from the Monitor. The mini-assembler does not accept labels; you must use actual values and addresses.

When you enter the mini-assembler, the Monitor prompt character changes from * to ! (the mini-assembler prompt) and assembles the first line of code (if a line of code is typed on the same line as the exclaimation point that caused the mini-assembler to be entered).

The mini-assembler saves one address, that of the program counter. Before you start typing a program, you must set the program counter to point to the location where you want the mini-assembler to store your program. Do this by typing the address followed by a colon. Then type the mnemonic for the first instruction in your program, followed by a space and the operand for that instruction.

The mini-assembler converts the line you typed into hexidecimal format, stores it in memory beginning at the location of the program counter, and then disassembles it again and displays the disassembled line. The prompt is then displayed on the next line. The mini-assembler is now ready to accept the second instruction in your program. To tell it that you want the next instruction to follow the first, don't type an address or a colon; type a space and the next instruction mnemonic and operand and then press return. Typing an additional return will exit the mini-assembler.

If an instruction line has an error in it, the mini-assembler beeps loudly and displays a caret (^) under or near the offending character in the input line. If you forget the space before or after a mnemonic or include an extraneous character in the hexidecimal value or address, the mini-assembler rejects the input line. If the destination address of a branch instruction is out of the range of the branch (more than 127 locations distant from the address of the instruction) the mini-assembler flags this as an error.

An address consists of one or more hexidecimal digits. The mini-assembler interprets addresses the same way the Monitor does: If one, three, or five digits are entered, a preceeding zero is automatically entered as well. For example, the instruction LDA #1 is assembled as A9 01. A dollar sign ($) is ignored by the mini-assembler and may be omitted when typing.

Branch instructions, which use the relative addressing mode, requirethe target address of the branch. The mini-assembler automatically calculates the relative distance to use in the instruction. If the target address is more than the allowable distance from the current program counter, the mini-assembler sounds a beep, displays a caret (^) under the target address, and does not assemble the line.

If you give the mini-assembler the mnemonic for an instruction and an operand and the addressing mode of the operand cannot be used with the instruction you entered, the mini-assembler will not accept the line.--

How to interact with So What Software

1. By mail... this is the best way for us because listings can be sent and a permanent record can be refered to. It also gives us some time to reflect upon your suggestion / problem and respond.

Send mail to:
So What Software
Call Box Edition
10221 Slater Ave. Suite 103
Fountain Valley, CA. 92708

2. By modem... this is the next best way for us much the same reason as "By mail..." is.

Send E-mail to:

Applelink
Mail stop: D1462

GEnie
Mail stop: SO-WHAT

3. By hotline.. this is the quickest but is the most time intensive for us... please use it only in a pinch!

Hotline number:
(714) 963-2344
Call between 6 and 10 PM P.S.T. weekdays only.