RAMWORKS

USER MANUAL

APPLIED ENGINEERING
P.O. Box 798, Carrollton, Texas 75006

CONGRATULATIONS!

You now own APPLIED ENGINEERING'S RAMWORKS CARD. Because of the time and care taken in the design and manufacture of your RAMWORKS card, we are sure that you will enjoy the use of it for many years to come.

This manual was written with a Z-80 Plus card, RAMWORKS card, and Wordstar using an Apple IIe.

1.0

CONTENTS

NOTICE: Your RAMWORKS card functions the same way as the Apple computer 8Ø column and extended 8Ø column cards function. This was done to insure that the RAMWORKS card would be compatible with all software written for the Apple cards. If you already have Apple's 8Ø column card, you may still refer to that manual regarding operation of our 8Ø column functions. If you already have Apple's extended 8Ø column card, you may still refer to that manual regarding auxiliary memory control. In other words, the RAMWORKS card functions the same way that the Apple extended 8Ø column card functions except for the additional command that switches betweeen the RAMWORKS multiple 64K banks.

HOW TO CONFIGURE YOUR RAMWORKS CARD

Revision A Apple IIe's are not capable of double high resolution graphics. To find out what Revision Apple you have, look at the letter behind slots 3, 4, and 5. Most rev A apples were sold in February of 1983. If you bought your Apple after that it is probably rev B. All rev B or later Apples can do the double high res. If you have a rev A Apple, current Apple policy will allow you to upgrade to a rev B motherboard at no charge. If you want to use the RAMWORKS card in a rev A Apple, you can; however, you'll have to break a "solder switch" on the RAMWORKS card. This solder switch is located on the component side of the circuit card down near the gold plated edge card connector and is labeled DH. You can use an exacto knife to open this solder switch for use with a rev A Apple. If you ever upgrade to the rev B, you can close the switch with a solder blob.

SOFTWARE COMPATIBILITY

One person and one person only determines what's compatible with what, and what works with what. This person decides how much memory will be used, how many disk drives you need and determines ALL parameters surrounding the operation of the computer. That person is the programmer. The programmer decides whether the program will use 2K of RAM or 1 meg. This is why your choice of a RAMWORKS card was a wise one because it can be configured in sizes ranging from 64K to 1 meg and a 128K version of the RAMWORKS card is $50 less than the 64K Apple extended card.

The RAMWORKS card is 100% software compatible with the Apple extended 80 column card. Programs designed to use the Apple extended card will function normally when used with the RAMWORKS card. Additionally, RAMWORKS is 99.9% compatible with a 192K card manufactured by another company. We say 99.9% because this memory card has a screen flickering problem (and it costs more!) that must be corrected with somewhat complicated software. We have tested the RAMWORKS card with software written for this other board and have encountered no difficulties. However, it is theoretically possible that such a problem could exist. Software written for your RAMWORKS card will not normally function correctly with this other 192K card because programmers do not take the extra steps necessary to correct the screen flickering because the RAMWORKS card corrects the problem in hardware (patent pending).

ADDING MEMORY TO THE RAMWORKS CARD

Your RAMWORKS card can be expanded to a maximum of 1 megabyte of memory, a maximum of 512K on the main RAMWORKS card and another 512K on a "piggyback" card. If you ever order the RAMWORKS piggyback RAM expander card, please specify the revision level of your RAMWORKS card when ordering. The revision level of the RAMWORKS card is located in the upper right hand corner of the RAMWORKS card. If you add more memory, you can use either 64K or 256K dynamic RAMs. Each row of RAM chips must contain all 64K RAMs or all 256K RAMS but if you wish, each row may be different. For example, in one row you can have all 64K RAMs and in another row all 256K RAMs for a total memory of 32ØK.

When adding more memory, please note that all the RAMs install notch side down (when the card is plugged in the computer). Applied Engineering also sells RAM chips. Please call for competitive pricing. We use only the highest quality RAMs available and we will warranty them for 3 years. Generally, when RAMs are purchased from electronics vendors they come with no warranty and are sold "as is". If you wish, you may return your RAMWORKS card to Applied Engineering for a memory upgrade. We will install the RAMs and retest your card at no additional cost. If you want us to do the RAM upgrade for you, please call for a return authorization number before sending your card. If you purchase the RAMWORKS card with 128K memory and wish to expand the memory, two options are available to you. You can remove one row of 64K RAMs and replace it with 256K RAMs or you can use the piggyback RAMWORKS expander card which plugs into the back of the RAMWORKS card. Before taking your RAMWORKS card to higher levels of memory, please make sure that the software you're using is capable of recognizing the additional memory. Almost without exception, commercially prepared programs will recognize at least 64K memory on the RAMWORKS card. Applied Engineering has developed several software products that modify popular commercially written software to use more memory on the RAMWORKS card. The programs that we are currently capable of expanding to higher levels of storage using our software products include Apple Works, Visicalc IIe, Advanced Visicalc IIe. Applied Engineering also has available progams that will turn the RAMWORKS card into an ultra high speed solid state disk drive. These programs are compatible with DOS 3.3, Pro-DOS, Apple PASCAL, CP/M 2.2, CP/AM 4.Ø. Please see our catalog for more information.

It is very easy to install the RAMWORKS card in your Apple IIe computer following the steps listed below.

First, turn the power to your Apple IIe computer <u>OFF</u>.

Any attempt to install the RAMWORKS card into the computer with the power on will **damage** the card and your computer! Now place the extended 80 column card, with the integrated circuit side up on a table in front of you making sure the gold connectors on the card are pointing toward you. The words KEYBOARD END will be on the left end of the card and this is a reminder that this end of the card will be nearest the keyboard of the computer. With keyboard end of the card identified, the extended 80 column card can then be plugged into the computer with the gold connectors fitting into the Apple's auxiliary slot which can be located by refering to page 14 of the Apple IIe Owner's Manual. A step by step procedure follows :

1) **TURN THE POWER OFF**

2) Remove the cover to your Apple by lifting the rear edges and and sliding the cover backwards.

3) Touch the large metal box on the left side of your Apple. This will discharge any static charge you may have and should be done everytime that you plan to touch anything inside your Apple IIe.

4) Check inside the Apple IIe and make sure the small red light at the rear of the computer is not glowing. If it is, this means that the power is **STILL ON.**

5) Locate the auxiliary slot in the Apple IIe. It is labeled AUX. CONNECTOR. If you cannot find it, refer to page 14 of your Apple IIe Owner's Manual.

6) Hold the card with the edge labeled KEYBOARD END actually pointing toward the keyboard and with the integrated circuit side of the card facing away from the power supply (the large metal box). Press it carefully into the auxiliary slot of the computer. The connectors will enter the slot with some friction and will then seat firmly.

7) Replace the cover of the Apple IIe by inserting the front lip under the front of the opening and pressing down on the rear corners until the cover snaps into place.

8) Your RAMWORKS card is now installed.

WHAT YOU SHOULD READ

If you only use commercially prepared programs, you need not read any further because the rest of this manual concerns itself with information used by programmers. Thousands of commercially prepared programs will use your RAMWORKS with no effort on your part at all. If you do program, please continue reading for information on 80 column display as well as information on auxiliary memory control.

USING THE RAMWORKS CARD

The startup procedure for displaying 80 columns of text on the Apple IIe depends on which operating system you plan to use.

STARTUP WITH PASCAL OR CP/M

When you are using either the USCD Pascal or CP/M operating systems on your Apple IIe they will automatically start up using 80 columns of text. When Pascal or CP/M are booted, they check for an 80 column text capability of your Apple IIe. If the operating system finds that your Apple IIe has 80 column capability, then it will use it.

Note: When Pascal or CP/M are booted they will come up in either 80 columns or 40 columns. The operating system will stay in that mode until it is rebooted. There is no way to switch between the two text column modes without causing a system error.

STARTUP WITH BASIC OR DOS

With Apple BASIC or DOS, just start up the computer as you normally would. When you are ready to use 80 columns of text, type PR#3 and press the RETURN key. This will activate the 80 column text capability of the computer. There are three escape sequences that effect the operation of the 80 column text mode. They are:

1) Press ESC key and then 4 to temporarily enter 40 column text

2) Press ESC key and then 8 to return to 80 column operation.

3) Press ESC key and let it go, then press the CTRL key and then while still holding it down, press the letter Q to deactivate the 80 column mode.

Another way to deactivate the the 80 column mode is to press control-reset. Whenever reset is performed the RAMWORKS card will automatically return to 40 column text mode. One thing that may surprise you the first time you deactivate the 80 column text mode with a reset, is that gibberish will seem to appear on the screen. Don't be alarmed, this comes from the fact that every even numbered column of text in the 80 column mode is stored in the aux. memory on the extended 80 column card. When a reset is performed, this additional memory is not used and therefore every odd numbered column of the 80 column text is displayed on the 40 column screen. If you enter some lines of text in 80 column mode and then perform a reset, you can see that every other letter has vanished leaving half of the former line.

NOTE: To tell which text mode the Apple IIe is in, look at the cursor. If the cursor is a blinking checkerboard, then the computer is in 40 column mode. If the cursor is a solid non-blinking block, then the computer is in 80 column mode.

DISPLAY DIFFERENCES BETWEEN 40 AND 80 COLUMNS

INVERSE command in 40 column text mode will only allow uppercase letters to be displayed in inverse. In 80 column mode the inverse command will display both lower and uppercase letters in inverse.

FLASH command will not work in 80 column text mode. If used in 80 column mode strange results will occur. To recover from this, use either the normal or inverse commands.

TABBING in 80 column mode, HTAB and comma tabbing only work up to 40 columns. To position the cursor anywhere on an 80 column line, use the command POKE 1403,N (POKE 1403 cannot be used in 40 column mode), where N is the horizontal position that you wish the cursor to start at.

NOTE: These differences that have been stated are not because of any faults with the MemoryMaster IIe card, but lie in the fact that when the Apple II computers were first developed, they did not provide an 80 column text mode and therefore did not include the proper routines for 80 column text. The above differences are the best that can be accomplished without totally redesigning the Apple computer. Also, all of the commands shown above are the same for even Apple's 80 column cards.

THE UPPERCASE-RESTRICT FEATURE

In Apple BASIC you are restricted to using uppercase letters for your programming. If you use any lowercase letters, the BASIC language will give you a syntax error. In uppercase-restrict mode you will be able to enter lowercase letters into print or input statements of your programs or anyplace where letters appear between quotation marks.

To activate the uppercase-restrict mode press the ESC key and then the letter R. Enter a BASIC program just as you would normally do, but whenever you type a beginning quotation mark all letters that follow will be in lowercase. To make any one or all of these letters in uppercase just hold down the shift key like an ordinary typewriter. When you type the ending quotation mark all letters that follow are in uppercase once more. This feature relieves much of the work when lowercase letters are desired in a BASIC program output. To deactivate the uppercase-restrict mode, type the ESC key and the letter T.

UPPERCASE-RESTRICT

ESC R to activate uppercase-restrict.
ESC T to deactivate.

ESCAPE MODES

The escape mode allows for the easy editing of typing mistakes. The escape mode enables you to move the cursor around the text screen without changing anything on it. This is very helpful when a misspelled word appears in the middle or end of a line. You can move to that position and make the correction without retypeing the whole line.

To enter the escape mode, press the key labeled ESC. The cursor will become an inverse plus symbol (in 40 column mode the cursor remains the same). This indicates that you are now in the escape mode. Some escape sequences will perform their function and then turn off the escape mode and others will perform their function and leave the escape mode on. This is handy when you have many escape functions to do, since it will relieve you from having to press the ESC key each time. To turn off the escape mode at anytime, just press the space bar and the cursor will return to normal.

In the following table all of the escape sequences are listed with their function. The notes off to the side refer to special circumstances that are listed below.

(1) These sequences turn off the escape mode.
(2) These sequences leave the escape mode on.
(3) These sequences are only available 80 column text mode.

ESCAPE SEQUENCE TABLE

ESCAPE SEQUENCE	FUNCTION	NOTES
ESC @	Clears text window and homes the cursor.	
ESC A	Moves the cursor one line right.	1
ESC B	Moves the cursor one space left.	1
ESC C	Move the cursor one space down.	1
ESC D	Move the cursor one line up.	1
ESC E	Clears the text to the end of the line.	
ESC F	Clears the screen from the current cursor position to end of screen.	
ESC I	Move the cursor up one line.	2
ESC J	Move the cursor one space left.	2
ESC K	Move the cursor one space right.	2
ESC M	Move the cursor one line down.	2
ESC R	Turns on uppercase-restrict mode.	3
ESC T	Turns off uppercase-restrict.	3
ESC 4	Switches from 8Ø column mode to 4Ø column mode.	3
ESC 8	Switches from 4Ø column mode to 8Ø column mode.	3
ESC CONTROL-Q	Turns off the 8Ø column mode capability.	3

NOTE: The escape sequences above using the I, J, K, and M keys can be replaced with the arrow keys in the appropriate direction.

CONTROL CHARACTER SEQUENCES

In the Apple IIe computer, there are several special key sequences that produce some valuable functions. These control sequences can be very helpful in a basic program for they allow you to access functions that are not in BASIC language. To get a control sequence, first hold down the key labeled CONTROL and while still holding it down press the other key that will produce the function desired. After you have done this, release both of the keys. In the following table there are several notes:

(1) Only available when 80 column card is on.
(2) Will only work from the keyboard.
(3) Will only work from a program.
(4) Will not work in the BASIC language.

CONTROL CHARACTER TABLE

CONTROL CHAR.	APPLE IIe NAME	ASCII CODE	WHAT IS DONE	NOTES
CNTL-Q	BELL	7	Makes a tone for .1 sec.	
CNTL-H	BACKSPACE	8	Moves cursor one space left, if at the left edge of line it goes to the right edge of the line above it.	
CNTL-J	LINE FEED	10	Moves the cursor one line down.	
CNTL-K	CLEAR EOS	11	Clears from the cursors position to the end of the screen.	1
CNTL-L	CLEAR	12	Moves the cursor to the upper-left corner of the screen and clears the screen.	1
CNTL-M	RETURN	13	Acts as if the RETURN key was pressed.	
CNTL-N	NORMAL	14	Sets to normal display mode.	1,3
CNTL-O	INVERSE	15	Sets to inverse display mode.	1,3
CNTL-Q	40 column	17	Sets to 40 column display.	1,3
CNTL-R	80 column	18	Sets to 80 column display.	1,3
CNTL-S	STOP LIST	19	Stops listing of characters on the screen, until a key is pressed.	1,2
CNTL-U	QUIT	21	Turns off the 80 column text and homes the cursor.	1,3
CNTL-V	SCROLL	22	Scrolls the display down one line.	1
CNTL-W	SCROLL UP	23	Scrolls the display up one line.	1
CNTL-Y	HOME	25	Moves the cursor to the upper left position of the display.	1
CNTL-Z	CLEAR LINE	26	Clears the line from cursor position.	1
CNTL-\	FORWARD SPACE	28	Moves the cursor one space to the right, opposite backspace.	1
CNTL-]	CLEAR EOL	29	Clear the line from cursor to the end of the line.	1
CNTL-^	GOTOXY	30	With the next two characters, minus 32 as X and Y values of the screen location for the cursor to move to.	1,4

NOTE: For more information about these control characters refer to the Applesoft BASIC reference manual.

To demonstrate the use of control sequences here is a program that uses the scrolling and bell control function.

```
10 FOR I = 1 TO 10
20 PRINT CHR$(22); : REM SCROLLS THE SCREEN DOWN
30 NEXT
40 FOR I = 1 TO 10
50 PRINT CHR$(23); : REM SCROLLS THE SCREEN UP
60 NEXT
70 PRINT CHR$(7) : REM SOUND THE BELL
```

This program will move anything that is on the TEXT screen down 10 lines. After this it will move anything that is on the screen up 10 lines. When the text on the screen is in its same location the bell will sound. Any text that scrolls off the screen is lost, but the text in the middle of the screen will move down and then up.

USING THE AUXILIARY MEMORY

In order to access any memory outside of the computers main memory, bank switching must be performed. One kilobytes (1K) of memory in the active bank of aux. memory serves as the 80 column display memory (with two banks this can lead to problems that will be discussed later). The remaining 63K bytes of memory of that bank are free for program or data storage. If you use the 40 column display mode, the full 64K bank is free for use.

DISPLAY PAGES

The Apple IIe generates its video from data stored in specific areas of memory called DISPLAY PAGES. The 40 column text and LOW RESOLUTION graphics use TEXT PAGE 1 and TEXT PAGE 2, located at addresses 1024 to 2047 ($400 to $7FF) and 2048 to 3071 ($800 to $BFF) respectively, in the main memory. The RAMWORKS uses a combination of TEXT PAGE 1 and the same page in the auxiliary memory to display 80 columns.

The 6502 microprocessor can address a total of only 64k bytes at a time. To be able to expand the available memory, bank switch must be used. Bank switching is when all or a portion of the memory space is replaced by another portion of memory not normally available. Various locations in memory control which portion of RAM is being used. The locations are called SOFT SWITCHES. The soft switches that switch the 48K memory bank ($200 to $BFFF) from main to auxiliary memory are called RAMRD and RAMWRT. Another soft switch is ALTZP which will switch the memory locations between 0 and 511 ($0 to $1FF) and between 53248 and 65535 ($D000 to $FFFF) to the auxiliary memory.

Each soft switch has a pair of memory locations dedicated to it. One location selects main memory, the other selects the RAMWWORKS auxiliary memory. The read and write functions may be set independently, making it possible for a program to read a byte from a location in main memory and write a different byte at the same location in the auxiliary memory. RAMRD selects the main or auxiliary memory for read operations, while RAMWRT does the same for write operations.

The ALTZP switch selects its memory areas between main and auxiliary memory for both reading and writing. This is done for the fact that zero page memory 0 to 511 ($0 to $1FF) is very important to the operation of the computer and if reading and writing did not work on the same memory location the computer would probably crash.

***********CAUTION***********

Do not manipulate the soft switches until you fully understand the effects they can have on your program. If you're not careful, you can crash your program. For example, a program in the auxiliary memory which uses standard I/O links to call built in I/O routines (like putchar which puts characters on the text screen) will crash because those routines are in main memory, which in this case, is switched out.

To use the main memory, RAMRD and RAMWRT must be off. To utilize the auxiliary memory, these switches must be on. TEXT PAGE 1 and HI-RESOLUTION graphics page 1 of the 48K auxiliary memory are controlled by RAMRD and RAMWRT. However, it is possible to control these areas separately by using the soft switches 80STORE, PAGE2, and HIRES independently.

If the 80STORE soft switch is activated, then the PAGE2 soft switch can select either main memory or the auxiliary memory. If the HIRES soft switch is off, then the PAGE2 soft switch selects display page 1 from either the main or auxiliary memory. If the HIRES soft switch is on, then the PAGE2 soft switch selects HI-RESOLUTION graphics page 1 and TEXT PAGE 1 from main or auxiliary memory.

To allow the RAMRD and RAMWRT to control the entire memory space from $200 to $BFFF, the 80STORE switch must be off. If the 80STORE switch is on, the TEXT PAGE will be unaffected by RAMRD and RAMWRT (the TEXT screens will not change when RAMRD and RAMWRT change). PAGE2 switch will control TEXT PAGE 1 regardless of how RAMRD and RAMWRT are set if 80STORE is on and HIRES is off. If HIRES is on then the HI-RESOLUTION page is displayed and RAMRD and RAMWRT have no effect on the HI-RESOLUTION page or the TEXT PAGES. PAGE2 then controls both TEXT PAGE 1 and HI-RESOLUTION graphics page 1 regardless of how RAMRD and RAMWRT are set if 80STORE and HIRES are on. Therefore, it is important to make sure the 80STORE and the HIRES switches are correctly set.

Determining the setting of these soft switches can be done by reading a memory location. If the byte located at $C013 has its high bit set to 1 (128 or higher) then RAMRD is on, which means the auxiliary memory is activated for reading. If the high bit at that location is set to 0 (127 or lower), then RAMRD is off, which means the 48K of main memory is enabled for reading. Location $C014 should be read to determine the status of RAMWRT. If the byte at that location has its high bit set to 1, then RAMWRT is on indicating that the auxiliary memory is enabled for writing. If the high bit at that same location is 0, then RAMWRT is off meaning that the 48K block of main memory is enabled for writing. A chart of the soft switches discussed is shown below:

SOFT SWITCH TABLE

NAME	FUNCTION IF THEN	LOCATION DECIMAL	HEX	NOTES
RAMRD	On Read Aux. 48K	49155	$C003	Write
	Off Read Main 48K	49154	$C002	Write
	Read RAMRD Switch	49171	$C013	Read
RAMWRT	On Write Aux. 48K	49157	$C005	Write
	Off Write Main 48K	49156	$C004	Write
	Read RAMWRT Switch	49172	$C014	Read
ALTZP	On Aux. Stack, Zero Page, and Bank-Switched Memory	49161	$C009	Write
	Off Main Stack, Zero Page, and Bank-Switched Memory	49160	$C008	Write
	Read ALTZP Switch	49174	$C016	Read
80STORE	On Access Page 1X	49153	$C001	Write
	Off Use RAMRD, RAMWRT	49152	$C000	Write
	Read 80STORE Switch	49176	$C018	Read
PAGE2	On Access Aux. Memory	49237	$C055	1, Write
	Off Access Main Memory	49236	$C054	1, Write
	Read PAGE2 Switch	49180	$C01C	Read
HIRES	On Access High-Resolution Page 1X	49239	$C057	2, Write
	Off Use RAMRD, RAMWRT	49238	$C056	2, Write
	Read HIRES Switch	49181	$C01D	Read

NOTE: Page 1X means the same page in the auxiliary memory. Read and write stand for the type of memory operation. (1) When 80STORE in on, the PAGE2 switch works as shown: when 80STORE is off, PAGE2 selects High-Resolution Page 2. (2) When 80STORE and HIRES is on you can use PAGE2 switch to select between High-Resolution Page 1 and 1X.

DOUBLE HI-RESOLUTION SCREEN

To use Double Hi-Res graphics you must have an Apple IIe motherboard rev. B or later and a RAMWORKS card with at least 64K of RAM. Current Apple computer policy is that owners of rev A Apple IIe motherboards can exchange them for rev B Apple IIe motherboards. If you have a rev A and do not want to exchange it, a "solder switch" can be broken on the RAMWORKS card so that it will function in the rev A mother board. This solder switch is located on the component side of the RAMWORKS card near the gold edge card connector and is labeled "DH". If you open the solder switch using a exacto knife or similar device, it can be reclosed later with a solder bridge if needed.
NOTE: There is only one double Hi-Resolution Page on the Apple IIe. Page flipping between page 1 and 2 that was done on the regular Hi-Res Pages to smooth the display appearances cannot be done. Some other ways such as fast machine language block moves of Hi-Res Pages may work if page flipping is needed.

SUBROUTINES DEALING WITH AUXILIARY MEMORY

There are built-in auxiliary memory subroutines which are valuable in writing assembly language programs which use the RAMWORKS auxiliary memory. These subroutines allow the programmer to bypass the soft switches and still utilize the auxiliary memory. The procedure for using these subroutines is the same as for using the I/O subroutines, which is discussed in Chapter 3 of the Apple IIe Reference Manual. The two main subroutines are AUXMOVE at location $C311, and XFER at location $C314.

AUXMOVE

AUXMOVE can be used to copy blocks of data from the main to the auxiliary memory or vice versa in your machine language and BASIC programs. The data addresses must be put into byte pairs in page zero ($0000 to $00FF) and the carry bit set to select the direction, either from main memory to auxiliary memory, or auxiliary to main memory, before this routine is called. AUXMOVE cannot be used to copy data in main memory to main memory or auxiliary to auxiliary memory. Also, don't use AUXMOVE to move memory in zero page, page one, or the banked-switched memory ($0000 to $01FF or $D000 to $FFFF). AUXMOVE uses zero page to move the data and this would produce strange results.

Three pairs of bytes are used for passing addresses to this subroutine. These are A1, A2, and A4. The function of each of these pairs are shown in the table on the next page.

AUXMOVE PARAMETERS

NAME	MEMORY LOCATION	PURPOSE
Carry		Ø = Aux. to Main Memory 1 = Main to Aux. Memory
A1L	$3C	Source Start Address, Low-byte
A1H	$3D	Source Start Address, Hi-byte
A2L	$3E	Source End Address, Low-byte
A2H	$3F	Source End Address, Hi-byte
A4L	$42	Destination Start Add, Low-byte
A4H	$43	Destination Start Add, Hi-byte

AUXMOVE uses the carry bit to select the direction in which the data is to be copied. To copy from main to auxiliary memory with RAMWORKS card, the carry bit must be set to one. To copy data from the extended 8Ø column card to main memory, the carry bit must be set to zero. When all of the parameters have been set AUXMOVE can be called at location $C311. After the subroutine is finished the data will have been copied and the A, X, and Y registers will contain their original values.

XFER

XFER is a built-in routine which can be used to transfer control of the computer to and from subroutines or programs located in the RAMWORKS auxiliary memory. Three parameters must be set up prior to using XFER. These parameters indicate the address of the routine you wish to execute, direction transfer (main to auxiliary or the reverse), and identity of zero page and stack areas you wish to use.

The transfer address must be put into the two bytes at $3ED and $3EE. The low order byte is put in location $3ED, while the high order byte is put in $3EE. The carry bit controls the direction of the transfer (just like in AUXMOVE). If the carry bit is set to one, then the control is switched to a program in main memory. If the carry bit is set to zero, then the control is switched to a program in the auxiliary memory. The overflow bit is used to select which zero page and stack is desired. The overflow should be set to zero in order to use main memory. If the overflow bit is set to one then the auxiliary memory zero page and stack will be used.

Control to the XFER routine should be passed by a machine language jump instruction or a BASIC call function rather than a subroutine call to location $C314. This should be done after all the parameters are set up correctly. The accumulator and the transfer address on the current stack are saved by XFER. XFER then sets up the soft switches for the parameters you have selected and then jumps to the new program.

Do not forget to save the stack pointer somewhere in the current memory before using XFER. The stack pointer should be restored upon regaining control; otherwise, a program error may occur.

XFER PARAMETER TABLE

NAME OR LOCATION	PURPOSE
$3ED	Program Starting Address, Low-byte
$3EE	Program Starting Address, Hi-byte
Carry Flag	Ø = Transfer from Auxiliary to Main Memory
	1 = Transfer from Main to Auxiliary Memory
Overflow Flag	Ø = Use Zero Page and Stack from Aux. Memory
	1 = Use Zero Page and Stack from Main Memory

NOTE: In the above discussions, any references to switching between the main and auxiliary memory zero page and stack, also apply to the bank-switching memory between $DØØØ to $FFFF since ALTZP soft switch is used.

SWITCHING BETWEEN 64K BANKS (128K RAMWORKS or larger)

Memory Implementation

The Apple IIe has 64K of main memory which can be expanded by adding a RAMWORKS card, which contains another 64K of memory. This extended memory is called auxiliary memory and is accessed by using softswitches built-in to the Apple IIe. The softswitches (see page 1Ø) control access to different sections (address ranges) of memory.

The extra memory on the RAMWORKS is implemented as extra 64K auxiliary banks of memory (see figure on next page). The selected 64K bank of memory is accessed in exactly the same way as the 64K auxiliary bank on the extended 8Ø column card, through the use of softswitches.

The programmer selects which bank of 64K is used by writing the bank number into the bank register, located at address $CØ73, (49267) on the RAMWORKS. The program can then use the built-in softswitches on the IIe to bank in portions of the selected 64K bank. This implementation allows the programmer to directly transfer data between any auxiliary bank of memory and main memory. Data transfers between different 64K banks on the RAMWORKS must be done using an intermediate move to main memory (or a byte at a time using the accumulator).

Bank Ø contains the 8Ø column and double hires video pages. These video pages are implemented in exactly the same way as the extended 8Ø column card. The video information is only displayed from the bank Ø video pages of the RAMWORKS, even though another 64K bank may be active (patent pending) selected by the bank register. This prevents an 8Ø column video display from changing every time another 64K bank of memory is selected (by changing the bank register at $CØ73). For example, if the program needs to access data from bank 1, then the 8Ø column display will continue fetching characters from bank Ø, while the program reads/writes data into bank 1.

Bank Configuration

If 64K RAMs are used the main board of the RAMWORKS contains banks Ø and 3, while the extra piggyback board contains banks 1 and 2. If 256K RAMs are used then bank Ø of the main board becomes banks Ø,4,8,12 and bank 3 becomes banks 3,7,11,15. Bank 1 of the piggyback board will become banks 1,5,9,13 and bank 2 contains 2,6,1Ø,14.

The bank register ($CØ73) is initialized as zero during power-up (but not after a reset - see below).

The bank register address is shared with the system. Writing to the bank register also triggers the paddle strobe, which is used to read the paddles. To ensure that the paddles are read properly, wait 3 ms. after writing to the bank register before starting the paddle read routine.

The contents of the bank register can't be read. The program must keep track of the current bank number. It can do this by storing the bank number in a reserved location inside each bank.

Programming Suggestions

Reset Vector

All programs should start by initializing the bank register to Ø. Since the card cannot detect a hardware reset, the software must force a reset instead. All the software reset routine has to do is store a Ø in the bank register. This can be done by intercepting the reset softvector at $3F2-$3F3 and changing it to point to the new reset routine.

```
LDA $3F2              ;STORE OFF OLD RESET SOFTVECTOR
STA $3ØØ              ; IN $3ØØ.3Ø1
LDA $3F3
STA $3Ø1
LDA #$Ø2              ;SET RESET VECTOR TO POINT TO $3Ø2
STA $3F2              ; ($3F2 -> $3Ø2)
LDA #$Ø3
STA $3F3
JSR $FB69             ;INITS NEW VALIDITY CHECK BYTE
                      ;SEE IIe REFERENCE MANUAL PG. 81
LDA #$A9
STA $3Ø2              ;RESET ROUTINE
LDA #$Ø               ;LDA #Ø
STA $3Ø3
LDA #$8D
STA $3Ø4
LDA #$73              ;STA $CØ73
STA $3Ø5
LDA #$CØ
STA $3Ø6
LDA #$6C
STA $3Ø7
LDA #$ØØ              ;JMP ($3ØØ)
STA $3Ø8
LDA #$3Ø
STA $3Ø9
```

Interrupts

Since the auxiliary memory card may be switched in when an interrupt occurs, any routines that use the extra memory must be prepared to process interrupts. For example, since the interrupt vector is located at $FFFE, then each auxiliary bank of memory contains an interrupt vector. If the expanded RAMWORKS contains 256K of memory then there are four different interrupt vectors on the card (located at $FFFE in each bank). Here are two of the possible methods for handling interrupts.

One simple method is to lock out interrupts whenever auxiliary memory is used. The optional ProDrive ramdisk driver already does this for you. The instruction to lock out interrupts is ENI and to enable interrupts CLI.

Another method which processes interrupts is to install an interrupt handler in each bank of memory. This handler should save the state of which bank is enabled, then jump to the interrupt handler in main memory. The handler finds the bank number by reading a specific location in memory, which was initialized with the bank number when the program was loaded. For example, location $FFF0 would be reserved in each auxiliary bank to store the bank number of that auxiliary bank. During initialization location $FFF0 of auxiliary bank 1 would be set to a 1, $FFF0 of auxiliary bank 2 would contain a 2, $FFF0 of auxiliary bank 3 would store a 3, etc.

There must also be a byte reserved in the 48K space, since $FFF0 of the auxiliary banks can only be read if the alternate language card (also stack and zero page) is enabled. This other reserved byte could reside in the slot 3 screen hole in auxiliary memory, such as $47B.

If the interrupt handler in main memory plans on using a specific bank of auxiliary memory then it must save the number of the auxiliary bank which is currently active, before changing to the new bank. For example, if the interrupt handler needs to write to the 80 column screen then it must make sure that auxiliary bank 0 is active before writing. Before changing the auxiliary bank register to 0, the interrupt handler must look in auxiliary memory and determine which auxiliary bank is active. In the previous example this could be done by enabling the alternate language card (assuming it isn't already enabled) reading $FFF0, and restoring the proper language card.

Criticism of this manual is welcomed at all times. Welcomed are any comments which will enhance the content or format. Individuals that wish to contribute software are encouraged to do so.

Correspondence regarding this document should be addressed to:

APPLIED ENGINEERING
P.O. BOX 798
CARROLLTON, TX. 75006

Other Apple Peripherals by APPLIED ENGINEERING:

TIMEMASTER II H.O. Clock
8 Bit 8 Channel A/D
12 Bit 16 Channel 9 Gain A/D
16 Bit Input/Output Board
Z-80 Plus CP/M Card
RAMWORKS
Music Synthesizer
VIEWMASTER 80

and more on the way, much more............

APPLE PERIPHERALS ARE OUR ONLY BUSINESS
That's why we're so good at it!