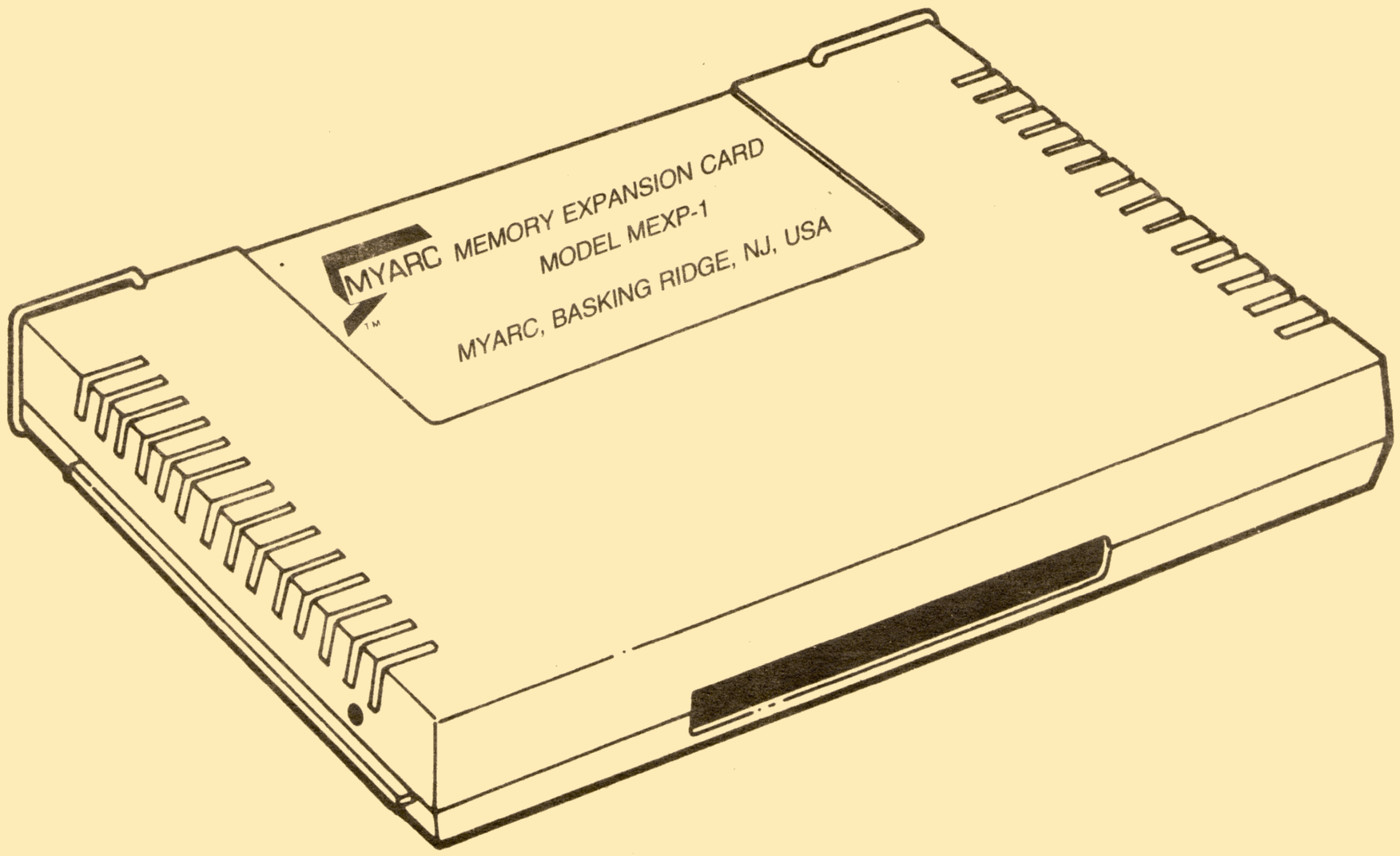
MYARC

T.M.

32/128/512K Memory Expansion Card

Model No. MEXP-1



MYARC, INC.
Basking Ridge, NJ

COPYRIGHT

DISCLAIMER OF WARRANTY

MYARC, INC. makes no representation or warranties with respect to the contents hereof and specifically disclaims any implied warranties of merchantability or fitness for any particular purpose. MYARC, INC. software is sold or licensed "as is." The risk as to its quality and performance is with the buyer and not MYARC, INC. Further, MYARC reserves the right to revise this publication and to make changes in the content hereof without obligations of MYARC to notify any person of such revisions or changes. MYARC also reserves the right to make design revisions or changes without obligations of MYARC to notify any person of such revisions or changes.

32/128/512K Memory Expansion Card

Model No. MEXP-1

PLEASE NOTE

This User Manual provides the reader with important information for effectively utilizing all three functions of the 128/512K MEXP-1 (32K memory expansion, RAM-disk, and print spooler).

If you have purchased the 32K version MEXP-1, your Card will function as a 32K memory expansion only. Accordingly, you need only read the following three sections:

Scope of This Manual

Introduction

Set-Up Instructions.

TABLE OF CONTENTS

WARNING

Your new 128/512K MEXP-1 provides the 32K memory expansion that increases the RAM capacity of your T199/4A to 48 Kbytes.

YOU CANNOT USE, AND YOU MUST REMOVE ANY OTHER 32K MEMORY EXPANSION CARD FROM YOUR PERIPHERAL EXPANSION BOX.

SCOPE OF THIS MANUAL

This manual describes the installation and operation of MYARC's Model MEXP-1 32/128K Memory Expansion Card for use with TI's PHP1200 Peripheral Expansion System.

Portions of this manual i.e., those sections dealing with technical descriptions of the TI 99/4A Home Computer System functions, their operation, and detailed instructions for system operation by the user, have been reproduced directly from TI's PHP1240 Disk Memory System and the PHP1260 32K Memory Expansion Card Manuals. Where appropriate, modifications have been made in these instructions for applicability and use with MYARC's RAM-disk Memory and Print Spooler Systems. These two systems are *not* included in TI's PHP1260 Memory Card.

Sections on the RAM-disk and Print Spooler Systems, which were developed by MYARC, were written by, and are the sole responsibility of MYARC.

WE RECOMMEND THAT FIRST-TIME USERS READ THE ENTIRE MANUAL BEFORE PROCEEDING WITH INSTALLATION AND OPERATION.

INTRODUCTION

THE MYARC 32/128/512K Memory Expansion Card, Model MEXP-1 is further improvement and technical advancement over TI's PHP 1260 32K Memory Expansion Card.

In addition to the 32 Kbytes of expansion memory provided in TI's PHP 1260, MYARC's Model MEXP-1 provides a RAM-disk function with very fast RAM-speed type access times and a print spooler function which permits the user to continue operating the computer console while *simultaneously* using the RS 232 and printer to print out up to (n) Kbytes of data or information stored on the print spooler.

As with the PHP1260, the 32K of expanded memory is designed for use with TI Extended BASIC, Editor/Assembler, TI LOGO, or any other *Solid State Software** Command Module designed to use the additional memory, as well as the UCSD p-System**. (For information on whether or not the 32K memory can be used with a module, refer to the Owner's Manual for that module).

To utilize the 32K Memory Expansion function in your MEXP-1, the TI Extended BASIC Command Module or another specialized Command Module MUST be inserted into the computer console. The TI BASIC computer language which is built into the 99/4A Console and certain other software packages *cannot* make use of the 32K Memory Expansion function.

For RAM-disk and print spooler operation, Command Modules may be used but are *not required.*

*"Solid State Software" is a trademark of TEXAS INSTRUMENTS, INC.
**"UCSD p-System" is a trademark of the Regents of the University of California.

(n)= 96 Kbytes in relation to 128K Memory Expansion Card
and
(n)= 480 Kbytes in relation to 512K

SET-UP INSTRUCTIONS

The steps involved in inserting the Memory Expansion Card into the Peripheral Expansion System and then checking its operation are included in this section. Please read this material completely before proceeding.

Note: The Peripheral Expansion System has eight slots into which accessory cards can be inserted. The Peripheral Expansion Card must occupy slot number 1. (For information on setting up the peripheral system, refer to the Peripheral Expansion System owner's manual.) If you have a Disk Memory Drive in the Peripheral Expansion System, the Disk Drive Controller Card must be in slot number 8. Other cards can be inserted in any of the remaining slots.

CAUTION

These electronic components can be damaged by static electricity discharges. To avoid damage, do not touch the connector contacts.

Once you've unpacked the unit, you're ready to insert the Memory Expansion Card into the Peripheral Expansion System. (Save the packing material for storing or transporting the unit.)

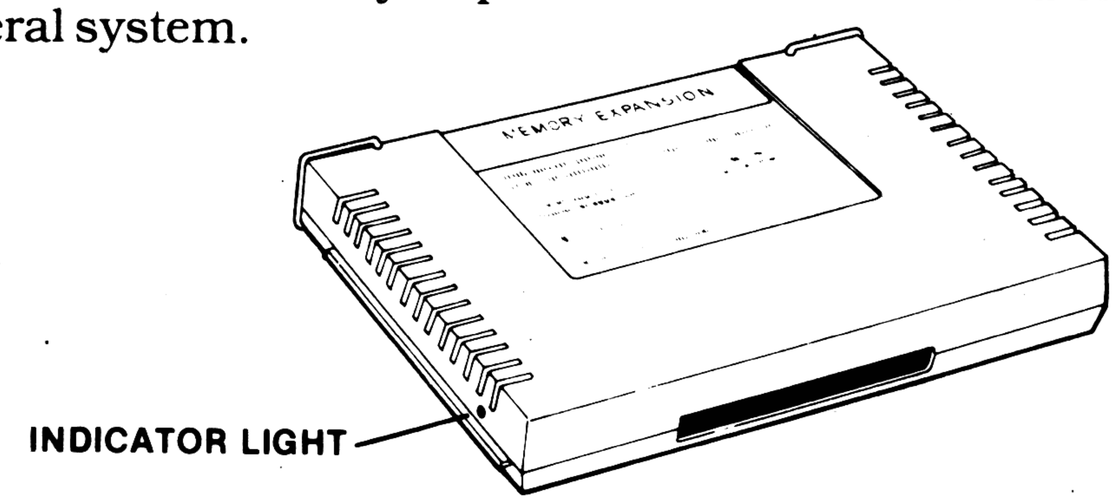
WARNING

Your new 128/512K MEXP-1 provides the 32K memory expansion that increases the RAM capacity of your T199/4A to 48 Kbytes.

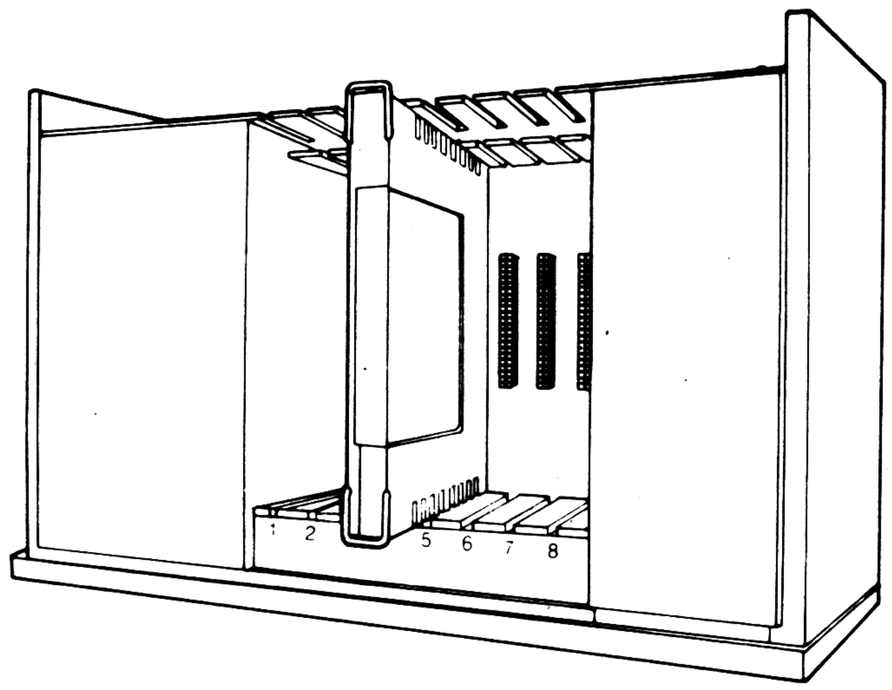
YOU CANNOT USE, AND YOU MUST REMOVE ANY OTHER 32K MEMORY EXPANSION CARD FROM YOUR PERIPHERAL EXPANSION BOX.

Inserting the Memory Expansion Card

1. First turn off the computer console and all attached devices.
2. WARNING: TO AVOID DAMAGING ACCESSORY CARDS, WAIT TWO (2) MINUTES AFTER TURNING OFF THE UNIT FOR THE POWER TO DISCHARGE BEFORE PROCEEDING.
3. Remove the top from the peripheral system by lifting the back edge of the top and pulling up.
4. The label identifying the Memory Expansion Card is on the top of the card. On the front of the card is an indicator light that is seen from the front of the peripheral system when the card is active. Hold the Memory Expansion Card with the indicator light facing the front of the peripheral system.



5. Carefully align the card with the desired slot and press the card firmly down into the slot.



6. Replace the top on the Peripheral Expansion System by sliding the front edge under the extension on the front of the unit and firmly pressing down on the back edge of the top. Do not run the system without the top in place because the top ensures proper ventilation. *Note:* If the top does not fit properly, remove the card and realign it in the slot, remembering to press down firmly until the connection is made.

CAUTION

Always disconnect the Peripheral Expansion System before moving the computer console. The cable connecting the console and peripheral system is not designed to support the weight of the units. To prevent damage, always disconnect all devices before moving any part of your Home Computer system. For long distance moves, remove all cards from the Peripheral Expansion System. Then, repack the devices in their original packing material.

Testing the Memory Expansion Card

1. The power switch is located on the front of the Peripheral Expansion System in the lower left-hand corner. Turn on the peripheral system, monitor, and console in that order.
2. A light should briefly come on in the position where you have inserted the Memory Expansion Card. Each time the computer system accesses the Memory Expansion Card, its corresponding light comes on. Note that the intensity and duration of the light varies, depending on the operation being performed within the system.
3. If the light does not come on, the corresponding card may not be properly inserted. Repeat the "Set-Up Instructions" procedure. If you still have difficulty, see "In Case of Difficulty" on page 24.

Performing the "On-Board" Diagnostics

In order to invoke the "on-board" diagnostics, type "CALL RDTEST" on the key board in either TI Basic or TI Extended Basic modes. In both cases, all data in the memory expansion card will be erased. This test will provide a message on the screen for each bank of 32K bytes describing the status of each bank. It requires about 25 seconds to test a 32K bank of memory.

Removing the Memory Expansion Card

1. Turn OFF the computer console, Peripheral Expansion System, monitor, and any other attached accessories.
2. Wait two minutes and then remove the top from the peripheral system.
3. Using both hands, pull up to remove the card from its slot in the peripheral system.

A QUICK LOOK AT THE 32/128K MEMORY EXPANSION CARD

The MEXP-1 card provides three functions as follows:

1. 32 Kbytes RAM—As with the PHP1260, the card adds 32 Kbytes of random access memory (RAM) to the 16 Kbytes of RAM available in the 99/4A computer console. This expanded memory is designed for use with TI Extended BASIC and other Command Modules (see Introduction, Page 3).

2. Exceptionally high speed access, in-line storage memory.

   RAM-disk memory can be utilized as if it were a floppy diskette. Since the RAM-disk is a solid state, integrated circuit device, speed-of-access time to store and retrieve is magnitudes faster than access times in the floppy disk memory system.

   Like a floppy disk memory system, with TI BASIC you can:

   - Save programs.
   - Store and retrieve data.
   - Update data.
   - Open and close files
   - Delete files

   The RAM-disk memory CAN be used with the DISK MANAGER cartridge from Texas Instruments, or the MYARC DMIII diskette. Of course, it can only be used after it has been partitioned.

   As with a floppy diskette, the RAM-disk must be "initialized", however, unlike a floppy diskette, the RAM-disk must be "initialized" *each* time the computer is turned back on after power shutdown (or power interruption). RAM-disk can only be initialized by CALL PART or the R Command of DMIII.

   Unlike a floppy diskette, the RAM-disk must be continuously powered to maintain it's stored memory. AT EVERY POWER SHUTDOWN OR POWER INTERRUPTION ALL STORED MEMORY IS ERASED.

3. Print spooler—Also not in TI's PHP1260, the print spooler buffers up to the maximum memory partitioned (allocated) to it for buffering data to the printer or any other device that is outputted from the RS232. Special software in the RAM-disk operating system permits *simultaneous* printout *and* operation of the computer. The remainder of the 96K memory not partitioned for print-spooler is available for the RAM-disk.

   As with the RAM-disk, power shutdown or interruption erases *all* memory then stored in the print spooler.

INITIALIZING

Under the section "A Quick Look at the 32/128/512K Memory Expansion Card" it was stated that the RAM-disk must be "initialized" every time the computer is turned back on after power shutdown or interruption.

Actually this "initialization" step is more accurately a "partitioning" step, where the total 96 Kbytes of available RAM must be partitioned (i.e., allocated) between the RAM-disk and the print spooler. The procedure is described in the following section "Partitioning Memory to RAM-disk and Print Spooler".

Hereafter throughout this manual, the term "partition" will be used in preference to "initialize".

PARTITIONING MEMORY BETWEEN RAM-DISK AND PRINT SPOOLER.

Before using the RAM-disk (RD) and/or print spooler (PS) it is necessary to specify the amount of memory that is to be partitioned between the two functions. The amount of memory available for partitioning is the total storage capacity (128K) *minus* expansion memory (32K) or 96 Kbytes.* You may select any whole number values of memory for RAM-disk and print spooler *provided* the *sum* of the two values equals 96.

In order to use RAM-disk and/or print spooler you must "partition" after *every* power startup.

To partition, ENTER:

CALL PART (x,y)

where x equals Kbytes alloctaed to RAM-disk, and
y equals Kbytes allocated to print spooler

AND

where x PLUS y is equal to 96.

ADDITIONAL PARTITIONING INSTRUCTIONS FOR 512K ONLY

The amount of memory available for partitioning is the total storage capacity (512K) MINUS expansion memory (32K) or 480 Kbytes.

You may select ANY whole number value for the print spooler and any whole number value, UP TO THE MAXIMUM VALUE OF 400 for the RAM-disk, provided the sum of the two values equals a total of 480.

CALL PART (x,y)

x = 400 and y = 80

or

x = o and y = 480

Any number combination between the two extremes, detailed above, can be used when partitioning the 512K Memory Expansion Card.

*For operational purposes, a small amount of memory is used for variable temporary storage. This storage is not available to the user.

SPECIAL COMMANDS FOR RAM-DISK and PRINT SPOOLER

RAM-disk

The device name for RAM-disk is "RD".

1. Once the card has been partitioned between RAM-disk (RM) and Print Spooler (PS), a simple command instructs the RAM-disk to emulate a floppy diskette or "turns off" the emulation as follows:

   CALL EMDK (1 or 0)

   where

   CALL EMDK (1) instructs RD to emulate DSK1.

   and

   CALL EMDK (0) "turns off" the DSK1 emulation.

   The RAM-disk portion of the 128/512K memory expansion card can also be used to emulate not only "DSK1", but also DSK2, DSK3, DSK4, DSK5. The command to perform this is: **CALL EMDK (X)** where X is either 0 to turn off the DSKX emulation or 1-5 to emulate either one of disk drives 1-5.

2. A volume or diskette name is given to RD (RDSK1) as follows:
   CALL VOL ("volume-name")
   The purpose of a volume name is to allow a program to access the RAM DISK in the form of DSK.VOLNAME.FILENAME. This is useful in applications such as MULTIPLAN where the software looks for "DSK.TIMP.filename" for the program files.

3. To get a directory listing of all files in the (RAM-disk) directory from the TI BASIC or Extended BASIC monitor, ENTER: **CALL RDDIR**

SPECIAL COMMANDS for RAM-DISK and PRINT SPOOLER

Print Spooler

The device name for print spooler is SP, SP/1, SP/2, or SPPIO corresponding to RS232, RS232/1, RS232/2 or PIO ports respectively.

To abort the print spooler, in BASIC Command Mode, enter: **CALL ABPS**

The print spooler may be used only for OUTPUT and you can address the print spooler with only one port at a time. Before using another RS232 port, the previous buffer must have been emptied.

To use the PRINT SPOOLER portion of ram, you first need to partition some of the memory for this task. You may "spool" output to your printer connected to either the PIO, RS232, or RS232/2 ports of the computer. To perform this, simply change the names used in the "OPEN" statement to SPPIO, SP (or SP/1), or SP/2 respectively. To abort this command, simply type CALL ABSP to abort spooling to any of the 3 ports. Please be advised not to use the same ports as used in spooling as are used in normal I/O. This will cause the spooler to become imoperative. Also, do not use the PIO and RS232 ports at the same time when spooling to one of them. This also will cause spooling to stop.

One last work about print spooling on the TI 99/4(A) is in order. When the print spooler is invoked, the data going to the printer is actually sent first to the RAM MEMORY allocated to the spooler and then to the printer. The data is sent to the printer on demand and allows the computer to perform another task at the same time. The "SPOOLING" is transparent to the user, in that it steals CPU cycles not being used by the computer when the computer is idle such as during data entry or the like. In actuality your TI 99/4(A) now becomes a "MULTITASKING" computer similar to more advanced professional or mini-computers.

RAM-DISK MEMORY OPERATION WITH TI BASIC

Your RAM-disk memory system can be used conveniently with TI BASIC. This section discusses saving/loading programs and processing file data with programs.

Saving and Loading Programs

The RAM-disk system stores and retrieves TI BASIC programs quickly and efficiently. You can store up to approximately 34 seventy-line programs, roughly equivalent to a SD, SS diskette.

The SAVE command is used to store a program in RAM-disk. The format is

```
SAVE RD. program-name
```

"RD" is the RAM-disk. The *program-name* is any valid identifying name you want to give your program. Valid program names can be up to ten characters long and can include any character except the period and the space character.

For example, to store a program called COLORDEMO in RAM-disk, you would enter this command (with the program in memory):

```
SAVE RD.COLORDEMO
```

The OLD command is used to read a program from RAM-disk. The format is:

```
OLD RD.program-name
```

When you enter the OLD command, the program you designate by name is read into the computer's memory. You can then run it, list it, or edit it—just as you would use a program entered from the keyboard.

NOTE

In MYARC's RAM-disk memory system the device name for RAM-disk is "RD". Accordingly when entering commands and BASIC statements always use "RD" for "device" in all such commands and statements for RAM-disk.

File Naming Conventions

A filename can be up to ten characters long and may contain any character except the period and the space character. However, the TI 99/4A best recognizes characters with ASCII codes of 32 through 95. For best results, use only upper-case characters A through Z and other characters with ASCII codes of 95 and lower (excluding the period and the space character) to name your files.

File Processing

There are seven main TI BASIC statements that are used to access files in RAM-disk. They are OPEN, CLOSE, INPUT, PRINT, EOF, RESTORE, and DELETE. The following discussions of each of these statements relate to their use with the RAM-disk system.

OPEN — The OPEN statement prepares a TI BASIC program to use data files stored in RAM-disk. It provides a link between a file-number used in a program and the file in RAM-disk, and it describes a file's characteristics so that a program can process or create the file. If the file already exists, the description that is given in the program must match the actual characteristics of the file.

The OPEN statement has the following general form:

```
OPEN #file-number:"device.file-name" [,file-organization] [,file-type] [,open-mode]
[,record-type]
```

The *file-number* and *device.file-name* must be included in the OPEN statement. The other information may be in any order or may be omitted. If an item is omitted, the computer assumes certain defaults, which are described below.

- *file-number* — The *file-number* (1 through 255 or an expression) is assigned to a particular file by the OPEN statement. (File number 0 is the keyboard and screen of the computer. It cannot be used for other files and is always open.) You may assign file numbers as you wish, with each file having a different number.

  The *file-number* is entered as a number sign (#) followed by a number or a numeric expression that, when rounded to the nearest integer, is a number from 1 to 255 and is not the number of a file that is already open.

- *device.file-name* — The *device* refers to RAM-disk in which a particular file is stored. The *file-name* may be any valid file name.

- *file-organization* — The records in a file can be accessed either sequentially or randomly. Records accessed sequentially are read or written one after the other. Records accessed randomly can be read or written in any order, including one after the other.

  To indicate which access method you wish to use, enter either SEQUENTIAL for sequential accessing or RELATIVE for random accessing. If you are creating a file, you may optionally specify the number of records on a file by following the word SEQUENTIAL or RELATIVE with a number or a numeric expression. If you do not specify the *file-organization*, the default is SEQUENTIAL.

- *file-type* — Files can be stored in RAM-disk either in easily readable ASCII characters or in machine-readable binary form. If the information is going to be printed or displayed for people to use, ASCII format is usually a better choice. In most cases, binary records are preferred because they take up less space and are processed faster by the computer.

  To specify that you wish the file to be in ASCII format, enter DISPLAY. (The length of a DISPLAY-type record is limited to approximately 150 bytes.) To specify binary format, enter INTERNAL. If you do not specify a *file-type*, the default is DISPLAY.

- *open-mode* — This entry instructs the computer that the file may be both read and written upon (UPDATE), may only be read (INPUT), may only be written to (OUTPUT), or may only be added to (APPEND).

  If a file is marked as protected, it cannot be written to and may only be opened for input. Also, APPEND mode can only be specified for VARIABLE length records. If you do not specify an *open-mode*, the computer assumes the default UPDATE.

  Note: If an unprotected file already exists on RAM-disk, specifying an *open-mode* of OUTPUT to the same file name writes over the existing file with the new file. You can prevent this by opening in the update mode and reading all the existing records so that you move to the end of the file or by using the RESTORE statement with the proper record number.

- *record-type* — File records may be all the same length (FIXED) or may vary in length (VARIABLE). If they are all FIXED, shorter records are padded to make up the difference. Any that are longer may be truncated to the proper length. Files that have FIXED-length records are processed faster than files with VARIABLE-length records but usually take up more space in the memory.

  If you like, you may specify a maximum length of a record by following VARIABLE or FIXED with a numeric expression. The maximum length for a VARIABLE file is 254 bytes, and the maximum for a FIXED file is 255 bytes. If you do not specify a record length, the default is 80.

  RELATIVE files must have FIXED-length records. If you do not specify a *record-type* for a RELATIVE file, the default is FIXED.

SEQUENTIAL files have either FIXED or VARIABLE-length records. If you do not specify a *record-type* for a SEQUENTIAL file, the default is VARIABLE. A file with FIXED-length records may be reopened for either SEQUENTIAL or RELATIVE access.

The following are examples of OPEN statements.

OPEN#1:"RD.MYFILE"	Creates or reopens a file in RAM-disk with a name of MYFILE. The file is a SEQUENTIAL file in the UPDATE mode with DISPLAY format and VARIABLE length records having a maximum length of 80 bytes. (These are the default attributes assigned by the computer.)
OPEN #23:"RD.X",RELATIVE 100,INTERNAL,OUTPUT,FIXED 80	Creates or reopens a file named X in RAM-disk. The file is a RELATIVE file in the OUTPUT mode with INTERNAL format and FIXED-length records having a maximum length of 80 bytes. Initially, 100 records are made available for the file, if enough room exists in RAM-disk.
OPEN #243:A$,INTERNAL	Creates or reopens a file with a name of ABCD if A$ equals RD.ABCD. The file is a SEQUENTIAL file in the UPDATE mode with INTERNAL format and VARIABLE-length records having a maximum length of 80 bytes.

CLOSE — The CLOSE statement discontinues the association between a file and a program. After the CLOSE statement is performed, the file is not available unless it is opened again with an OPEN statement. Files may optionally be deleted by adding :DELETE to the end of the CLOSE statement.

The CLOSE statement has the following general form:

```
CLOSE #file-number [:DELETE]
```

The *file-number* is the number which you used in the OPEN statement to open the file.

If you do not close a file, data on it may be lost. If a program ends due to a BREAK statement, by your pressing **CLEAR**, or because of an error, files may not be closed even if you have a CLOSE statement later in the program. However, you can close the files properly by entering CLOSE *#file-number*, NEW, or BYE if you wish to leave BASIC. Editing the program also automatically closes any open files.

Note: If you leave TI BASIC by pressing **QUIT**, data may be lost. Leave TI BASIC only by entering BYE when you are processing files.

INPUT — The INPUT statement allows you to read data from RAM-disk. It only can be used with files opened in INPUT or UPDATE mode.

The INPUT statement has the following general form:

```
INPUT #file-number [,REC record-number]:variable-list
```

The *file-number* and a *variable-list* must be included in the INPUT statement. The *record-number* may optionally be included when reading random-access files.

- *file-number* — The *file-number* is the number assigned to a particular file by the OPEN statement. The *file-number* is entered as a number sign (#) followed by a number or a numeric expression that, when rounded to the nearest integer, is a number from 1 to 255 and is the number of a file that is open.

- *record-number* — A *record-number* refers to the record on the file which you want to read. The *record-number* can only be specified for RELATIVE files. (SEQUENTIAL files are read in sequential order.)

- *variable-list* — The *variable-list* is the list of variables into which you want the data from the file to be read. It consists of string or numeric variables separated by commas.

The following are examples of INPUT statements.

INPUT #1:X$	Puts into X$ the next value available in the file that was opened as #1.
INPUT #23:X,A,LL$	Puts into X, A, and LL$ the next three values from the file that was opened as #23.
INPUT #11,REC 44:TAX	Puts into TAX the first value of record number 44 of the file that was opened as #11.
INPUT #3:A,B,C	Puts into A, B, and C the next three values from the file that was opened as #3. The comma after C creates a pending input condition. When the next INPUT statement using this file is performed, one of the following actions occurs. If the next INPUT statement has no REC clause, the computer uses the data beginning where the previous INPUT statement stopped. If the next INPUT statement includes a REC clause, the computer terminates the pending input condition and reads the specified record.

PRINT — The PRINT statement allows you to write data into RAM-disk. It can only be used with files opened in OUTPUT, UPDATE, or APPEND mode.

The PRINT statement has the following general form:

```
PRINT #file-number [,REC record-number] [:print-list]
```

The *file-number* must be included in the PRINT statement. The *record-number* may optionally be included when writing to random-access (RELATIVE) files. The *print-list* is also optional.

- *file-number* — The *file-number* is the number assigned to a particular file by the OPEN statement. The *file-number* is entered as a number sign (#) followed by a number or a numeric expression that, when rounded to the nearest integer, is a number from 1 to 255 and is the number of a file that is open.

- *record-number* — A *record-number* refers to the file record you want to write. The *record-number* can only be specified for random files (RELATIVE).

- *print-list* — The *print-list* is the list of values that you want to put on the file. It consists of string or numeric variables or constants separated by commas, colons, and semicolons.

The following are examples of PRINT statements.

`PRINT #1:X$`	Puts the value of X$ into the next position of the file that was opened as #1.
`PRINT #23:X;A;"TIMES 4"`	Puts the value of X, A, and "TIMES 4" into the next record in the file that was opened as #23.
`PRINT #11,REC 44:"TAX"`	Puts the string constant "TAX" into record number 44 of the file that was opened as #11.
`PRINT #3:A;B;C,`	Puts the values of A, B, and C into the next three positions in the file that was opened as #3. The comma after C creates a pending print condition. When the next PRINT statement is performed, one of the following actions occurs:

If the next PRINT statement has no REC clause, the computer places the data immediately following the previous data.

If the next PRINT statement has a REC clause, the computer writes the pending print record onto the file at the position indicated by the internal counter and performs the new PRINT statement as usual.

EOF — The EOF (end-of-file) function indicates whether there is another record to be read from a file. The EOF function has the following general form:

```
EOF(file-number)
```

The value of the *file-number* must correspond to the number of an open file.

The EOF function always assumes that the next record is going to be read sequentially, even if you are using a RELATIVE file.

The value returned by the EOF function depends on where you are in the file. If you are not at the end of the file, the value returned is 0. If you are at the end of the file, the function returns a value of 1. If the RAM-disk is full and you are at the end of the file, the function returns a value of −1.

The following are examples of the EOF function.

`PRINT EOF(3)`	Prints a value of 0, 1, or −1, according to whether you are at the end of the file that was opened as #3.
`IF EOF(27)<>0 THEN 1150`	If you are at the end of the file that was opened as #27, control is transferred to line 1150.

The usual way to keep track of the last record in RELATIVE files is to maintain a "dummy" record as the first record in the file. This record contains the number of records in the file. Each time you change the length of the file, you must update this record.

RESTORE — The RESTORE statement is used to position you at a specified record on a file. The statement has the following general form:

```
RESTORE #file-number [,REC record-number]
```

The *file-number* must be included in the RESTORE statement when it is used with files. The *record-number* may optionally be included.

- *file-number* — The *file-number* is the number assigned to a particular file by the OPEN statement. The *file-number* is entered as a number sign (#) followed by a number or a numeric expression that, when rounded to the nearest integer, is a number from 1 to 255 and is the number of a file that is open.

- *record-number* — The *record-number* indicates which record on the file you want to access. A *record-number* is allowed only with RELATIVE files.

The following are examples of the RESTORE statement.

RESTORE #6	Resets the file pointer to the first record on the file that was opened as #6 so that the next INPUT or PRINT statement referring to the file will access the first record.
RESTORE #23,REC 12	Resets the file pointer to the thirteenth record on the file (remember that the first record is number zero) so that the next INPUT or PRINT statement referring to the file will access the first record.

With RELATIVE files, RESTORE is generally used only to position the internal counter at the end of the file since the record that you want to read or write can be specified in the INPUT or PRINT statement.

DELETE — The DELETE statement is used to delete files. It has the following general form:

```
DELETE "RD.file-name"
```

"RD" refers to RAM-disk in which a particular file is stored. The *file-name* may be any valid file name.

Note: Files can only be deleted if they are not protected.

Cataloging Files

By accessing the file index in RAM-disk, you can use TI BASIC to read a catalog of disk contents. The RAM-disk-index (or catalog) file is an unnamed, INTERNAL-format, FIXED-length file. The following is an example of an OPEN statement that accesses the catalog on drive one:

```
100 OPEN #1:"RD.",INPUT,RELATIVE,INTERNAL
```

Note: *File-name* is omitted, since the catalog is an unnamed file.

Every record in the catalog file contains four items: one string and three numeric values, written in INTERNAL format. There are exactly 128 records in the file, numbered from 0 through 127.

Record #O contains information about the RAM-disk. The string (up to 10 characters in length) indicates the name of the RAM-disk, and the numerical items give the following information:

- the record type (always a zero for record #0)
- the total number of sectors in RAM-disk
- the number of available sectors in RAM-disk

Records 1 through 127 contain information about the corresponding files in the index. The string item is the file-name, and the numeric items are as follows:

- The file type (a negative value if the file is protected)
  1 = DISPLAY/FIXED data file
  2 = DISPLAY/VARIABLE data file
  3 = INTERNAL/FIXED data file
  4 = INTERNAL/VARIABLE data file
  5 = BASIC program or other "memory image" data

- The total number of sectors allocated for the file
- The total number of bytes per record

A type-5 file always has a zero (0) as the number of bytes per record, since this measurement does not relate to memory image data.

A sample program listed at the end of this section illustrates how to display the RAM-disk catalog.

Sample Programs

Three programs are included in this section. The first illustrates some of the techniques that can be used for file management, showing how to open a file, write records randomly to it, and read records randomly from it. The second program shows how to save the contents of the screen on a file and put it back on the screen later. This is useful if you have created a graphics screen and want to be able to store and retrieve it. The third program shows how to read the catalog from the RAM-disk using TI BASIC.

GENERAL PROGRAM — The following program illustrates general file handling techniques for a random-access file.

The first section, lines 100 through 120, gives the entry instruction.

```
100 CALL CLEAR
110 OPEN #2:"RD.GENFILE",RELATIVE 50,INTERNAL
120 PRINT "ENTER 'XXX' TO LEAVE ENTRY"::
```

The next section, lines 130 through 210, allows entry of up to 50 records, numbered 1 through 50. Line 160 checks to see if the last record has been entered. If 50 records have been entered, line 210 informs the user that the file is full.

```
130 REM ENTRY SECTION
140 TOTAL=0
150 INPUT "RECORD "&STR$[TOTAL+1]" ":BUFFER$
160 IF BUFFER$="XXX" THEN 220
170 PRINT#2,REC TOTAL+1:BUFFER$
180 TOTAL=TOTAL+1
190 IF TOTAL<50 THEN 150
200 REM PRINT FULL MESSAGE
210 PRINT "FILE FULL"
```

The next section, lines 220 through 320, allows the user to choose what to do.

```
220 REM CONTROL SECTION
230 PRINT ::"1=PRINT FILE"
240 PRINT "2=PRINT A RECORD"
250 PRINT "3=CHANGE A RECORD"
260 PRINT "4=ADD A RECORD"
270 PRINT "5=LEAVE THE PROGRAM":::
280 INPUT "ENTER YOUR CHOICE?":X
290 PRINT
300 IFX>1 THEN 220
310 IF X<5 THEN 220
320 ON X GOTO 1000, 2000, 3000, 4000, 5000
```

The next section, lines 1000 through 1060, prints the entire file in sequential order.

```
1000 REM PRINT A FILE
1010 RESTORE #2,REC 1
1020 FOR COUNT=1 TO TOTAL
1030 INPUT #2:BUFFER$
1040 PRINT COUNT;BUFFER$
1050 NEXT COUNT
1060 GOTO 220
```

The next section, lines 2000 through 2040, prints a specific record.

```
2000 REM PRINT A RECORD
2010 GOSUB 10000
2020 INPUT #2,REC RECNUM:BUFFER$
2030 PRINT RECNUM;BUFFER$
2040 GOTO 220
```

The next section, lines 3000 through 3040, allows a record to be changed.

```
3000 REM CHANGE A RECORD
3010 GOSUB 10000
3020 INPUT "ENTER NEW DATA?":BUFFER$
3030 PRINT #2,REC RECNUM:BUFFER$
3040 GOTO 220
```

The following section, lines 4000 through 4050, allows a record to be added.

```
4000 REM ADD RECORD
4010 IF TOTAL=50 THEN 210
4020 TOTAL=TOTAL+1
4030 INPUT "ENTER DATA?":BUFFER$
4040 PRINT #2,REC TOTAL:BUFFER$
4050 GOTO 220
```

Lines 5000 through 5020 close the file at the end of the program.

```
5000 REM LEAVE THE PROGRAM
5010 CLOSE #2
5020 END
```

The last section, lines 10000 through 10050, is a subroutine used for user input of a record number.

```
10000 INPUT "WHICH RECORD?":RECNUM
10010 IF RECNUM<1 THEN 10030
10020 IF RECNUM<=TOTAL THEN 10050
10030 PRINT "INVALID RECORD NUMBER"
10040 GOTO 10000
10050 RETURN
```

SAVE AND RESTORE A SCREEN — Lines 100 through 210 of the following program can be used as a subroutine in other programs when you want to save something you've created on the screen. (Be sure that you change the file name each time you use the program so that you don't write over a file that you have previously saved.) This program only saves and restores the characters that are on the screen. Information about redefined characters and colors is not saved.

```
100 REM SAVE A SCREEN
110 REM ADD AT END OF THE PROGRAM THAT PRODUCES THE SCREEN
120 REM
130 REM CHOOSE A UNIQUE FILENAME
140 OPEN #20:"device.file-name",INTERNAL
150 FOR ROW=1 TO 24
160 FOR COLUMN=1 TO 32
170 CALL GCHAR[ROW,COLUMN,X]
180 PRINT #20:X
190 NEXT COLUMN
200 NEXT ROW
210 CLOSE #20
```

The following runs as an independent program. It recalls the contents of a screen that has been previously saved and then puts it on the screen.

```
100 REM REPRODUCE A SCREEN
110 OPEN #30:"device.file-name",INTERNAL
120 FOR ROW=1 to 24
130 FOR COLUMN=1 to 32
140 INPUT #30:X
150 CALL VCHAR[ROW,COLUMN,X]
160 NEXT COLUMN
170 NEXT ROW
180 CLOSE #30
190 REM DELAY TO GIVE TIME TO LOOK AT THE SCREEN
200 FOR DELAY=1 to 10000
210 NEXT DELAY
```

Note: Remember to use "RD" for "device" throughout these sections.

READ THE CATALOG — The following program enables you to read and print the catalog for RAM-disk from TI BASIC. Lines 100 through 160 set up a single-dimension array of five items corresponding to the five types of files.

```
100 CALL CLEAR
110 DIM TYPE$(5)
120 TYPE$[1]="DIS/FIX"
130 TYPE$[2]="DIS/VAR"
140 TYPE$[3]="INT/FIX"
150 TYPE$[4]="INT/VAR"
160 TYPE$[5]="PROGRAM"
```

The next section opens the file, reads the RAM-disk information for record #0, and displays it on the screen.

```
210 OPEN #1:"RD.", INPUT, RELATIVE,INTERNAL
220 INPUT #:A$,J,J,K
230 DISPLAY "RD-RAM-DISK NAME=";A$
A$:"AVAILABLE=";K;"USED=";J-K
```

The remainder of the program reads the remaining information in the index, formats it, and displays the catalog on the screen.

```
240 DISPLAY:"FILENAME SIZE TYPE P":
"                              ".
---------- ----- --------- -";
250 FOR LOOP=1 TO 127
260 INPUT #1:A$,A,J,K
270 IF LEN[A$]=0 THEN 350
280 DISPLAY:A$;TAB(12);J;TAB[17];
TYPE$[ABS[A]];
290 IF ABS[A]=5 THEN 320
300 B$=" "&STR$[K]
310 DISPLAY SEG$[B$,LEN[B$]-2,3];
320 IF A›0 THEN 340
330 DISPLAY TAB[28];"Y";
340 NEXT LOOP
350 CLOSE #1
```

APPENDIX A: Error Codes in TI BASIC

The normal error codes for your computer are given in the *User's Reference Guide*. The following codes are for errors that relate to the RAM-disk system.

Error codes are two-digit numbers. The first digit indicates the command or statement involved in the error, and the second digit tells you the type of error.

FIRST DIGIT	COMMAND/STATEMENT	FIRST DIGIT	COMMAND/STATEMENT
0	OPEN	5	OLD
1	CLOSE	6	SAVE
2	INPUT	7	DELETE
3	PRINT	9	EOF
4	RESTORE		

SECOND DIGIT	TYPE OF ERROR
0	The device specified could not be found.
1	FILE WRITE PROTECTED — Remove protection via the Disk Manager.
2	BAD OPEN ATTRIBUTE — One or more OPEN options were illegal or didn't match the file's actual characteristics.
3	ILLEGAL OPERATION — Should not be generated by the disk system. It could be caused, however, by attempting to perform some illegal file operation, such as requesting INPUT from the Thermal Printer.
4	OUT OF SPACE — The RAM-disk is full, or you are trying to open more files than are allowed.
5	ATTEMPT TO READ PAST END OF FILE.
6	DEVICE ERROR — May occur if the RAM-disk is not partitioned or is damaged. It also may occur if the system had its power disconnected during a previous print.
7	FILE ERROR — The indicated file or RAM-disk doesn't exist or you are trying to read a BASIC file as if it were data.

APPENDIX B: Accessing Memory with Assembly Language

To access the additional 96K of RAM (in addition to the standard 32K) with assembly language, the following is a useful discussion on its operation.

The 128K RAM is divided into 4 banks of 32K each. When enabled, each bank occupies memory locations >2000->3FFF and >A000->FFFF. The mechanism to select a particular memory bank is determined by CRU bits >1002 and >1004. When these bits are zero, the normal 32K memory expansion is enabled. The other 3 banks are selected by setting the proper combination of bits. The following examples show how to select each of the four memory banks from assembly language executing from "PAD" RAM:

Select bank 2:

```
LI    R12,>1000
SBO   1
SBZ   2
```

Select Bank 3:

```
LI    R12,>1000
SBZ   1
SBO   2
```

To Select bank 4:

```
LI    R12,>1000
SBO   1
SBO   2
```

To Select "standard" 32K RAM:

```
LI    R12,>1000
SBZ   1
SBZ   2
```

IN CASE OF DIFFICULTY

If the Memory Expansion Card does not appear to be working properly, check the following.

1. *Power* — Be sure all devices are plugged in. Then turn on the power to the units in the proper sequence: disk drives and Peripheral Expansion System first, followed by the console and the monitor.

2. *Card Position* — Turn the power off, wait two minutes, and remove the top of the peripheral system. Verify that all cards are inserted properly and then replace the top.

3. *Cable* — Check to be sure that the proper cables are being used. Check the cables for loose or broken leads. Check to see that the cables are properly connected, right side up.

4. *Software* — Be sure all commands and statements are used as described in this manual. If the system works properly with commands but not with a program, the problem is probably with the program. Especially check the use of OPEN, INPUT, and PRINT.

5. *Peripheral Expansion System* — Check for proper connection between the console and peripheral system.

6. *Home Computer* — Check to see that the Home Computer works properly with all accessories disconnected.

7. If none of the above procedures corrects the difficulty, consult "If You Have Questions or Need Assistance" or see the "Maintenance and Service Information" section of the *User's Reference Guide*.

IF YOU HAVE QUESTIONS OR NEED ASSISTANCE

If you have questions concerning Model MEXP-1 32/128K Memory Expansion Card operation or repair, contact first the dealer from whom you purchased the equipment.

Your dealer will be able to quickly answer most questions. If your dealer doesn't have an immediate answer, the dealer will either contact MYARC or suggest you contact us directly by mail or phone.

Our address and telephone number are:

MYARC, INC.
P.O. Box 140
Basking Ridge, N.J. 07920

(201) 766-1701

Please Note: This telephone number is not a toll-free number and collect calls cannot be accepted.

90-DAY LIMITED WARRANTY

THIS MYARC MEXP-1 MEMORY EXPANSION CARD WARRANTY EXTENDS TO THE ORIGINAL CONSUMER PURCHASER OF THE ACCESSORY.

WARRANTY DURATION

This MEXP-1 Memory Expansion Card is warranted for a period of 90-days from the date of the original purchase by the consumer.

WARRANTY COVERAGE

This MEXP-1 Card is warranted against defective materials or workmanship.
THIS WARRANTY IS VOID IF THE ACCESSORY HAS BEEN DAMAGED BY ACCIDENT, UNREASONABLE USE, NEGLECT, IMPROPER SERVICE OR OTHER CAUSES NOT ARISING OUT OF DEFECTS IN MATERIALS OR WORKMANSHIP.

WARRANTY DISCLAIMERS

ANY IMPLIED WARRANTIES ARISING OUT OF THIS SALE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE LIMITED IN DURATION TO THE ABOVE 90-DAY PERIOD. MYARC, INC. SHALL NOT BE LIABLE FOR LOSS OF USE OF THE HARD-WARE OR OTHER INCIDENTAL OR CONSEQUENTIAL COSTS, EXPENSES, OR DAMAGES INCURRED BY THE CONSUMER OR ANY OTHER USER.

Some states do not allow the exclusion or limitation of implied warranties or consequential damages, so the above limitations or exclusions may not apply to you in those states.

LEGAL REMEDIES

This warranty gives you specific legal rights, and you may also have other rights that vary from state to state.

WARRANTY PERFORMANCE

During the above 90-day warranty period, your Memory Expansion Card will be repaired or replaced with a new or reconditioned unit of the same or equivalent model (at MYARC's option) when return is authorized by MYARC and the unit is returned by prepaid shipment to MYARC, INC. at the address shown below. The repaired or replacement unit will be warranted for 90 days from date of repair or replacement. Other than the shipping requirement, no charge will be made for the repair or replacement of in-warranty units.

SHIPPING INSTRUCTIONS: If you believe that your unit requires servicing, please contact MYARC *before* you return your system. We will try to analyse and may be able to solve your problem without need of returning the unit. Please obtain a Return Authorization number from us before you ship the unit back.

MYARC strongly recommends that you insure the unit for value, prior to shipment.

MYARC's ADDRESS:

MYARC, INC.
241 Madisonville Road
Basking Ridge, NJ 07920

MYARC, INC.

Basking Ridge, N.J.

PRINTED IN U.S.A.