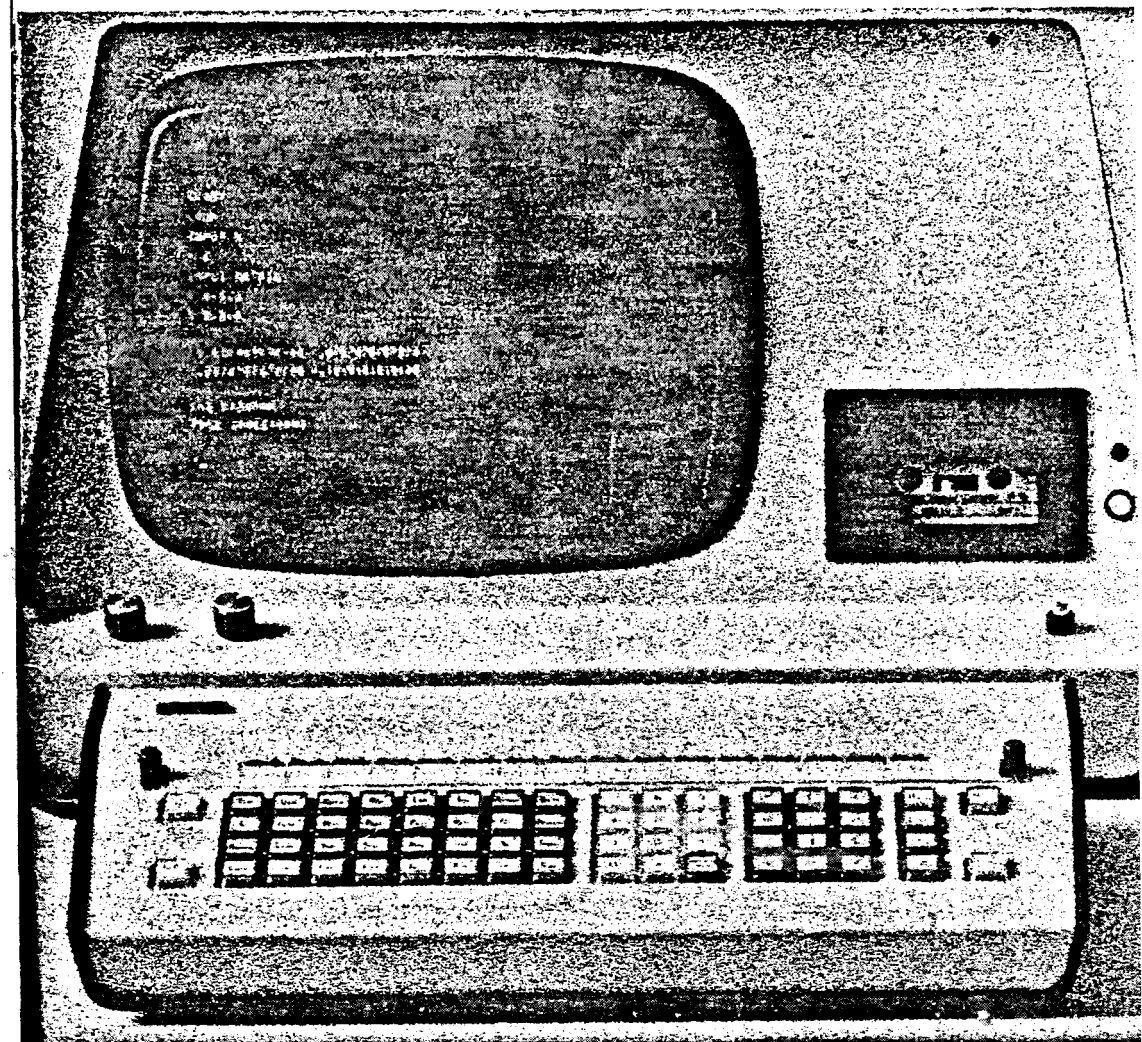
WANG

SYSTEM 2200
TAPE CASSETTE DRIVE
REFERENCE MANUAL
(Models 2217 & 2218)

SYSTEM 2200



System 2200 Tape Cassette Drive (Models 2217 & 2218)

Reference Manual

WANG LABORATORIES, INC.
836 NORTH STREET, TEWKSBURY, MASSACHUSETTS 01876. TEL. (617) 851-4111. TWX 710 343-6769. TELEX 94-7421

Disclaimer of Warranties and Limitation of Liabilities

The staff of Wang Laboratories, Inc., has taken due care in preparing this manual; however, nothing contained herein modifies or alters in any way the standard terms and conditions of the Wang purchase agreement, lease agreement, or rental agreement by which this equipment was acquired, nor increases in any way Wang's liability to the customer. In no event shall Wang Laboratories, Inc., or its subsidiaries be liable for incidental or consequential damages in connection with or arising from the use of this manual or any programs contained herein.

WANG LABORATORIES, INC.
836 NORTH STREET, TEWKSBURY, MASSACHUSETTS 01876, TEL. (617) 851-4111, TWX 710 343-6769, TELEX 94-7421

HOW TO USE THIS MANUAL

The Wang System 2200 Tape Cassette Drive Reference Manual provides an introduction to the operation of all Wang 2200 Series cassette drives including the Model 2217 Single Tape Cassette Drive, the Model 2218 Dual Tape Cassette Drive, the Model 2216/2217 Combined CRT Executive Display/Single Tape Cassette Drive, the Model 2216A/2217 Combined Upper Lower Case CRT Display/Single Tape Cassette Drive, and the Model 2220 Console-CRT/Keyboard/Single Tape Cassette.

Chapters 1 and 2 are recommended reading for all users of System 2200 configurations equipped with either a single tape cassette drive or multidrives. Information in both chapters is fundamental for operators and programmers alike.

In Chapter 3, the procedures and programming techniques for recording and reading data files are presented in great detail. System 2200 users who do not plan to work with data files immediately should postpone reading Chapter 3 until the information contained therein is needed.

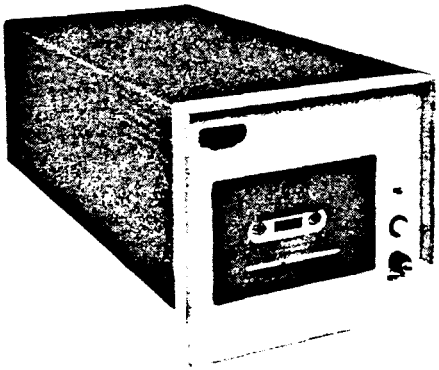
Chapters 4, 5, and 6 emphasize programming techniques rather than operational procedures. The chapters are independent of each other and can be read or studied in any order.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

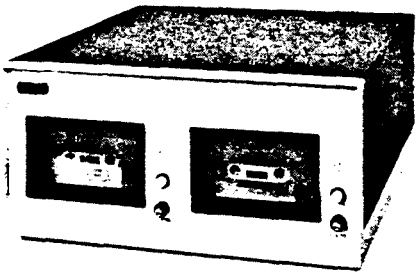
LIST OF FIGURES

LIST OF TABLES



Model 2217

Single Tape Cassette Drive



Model 2218

Dual Tape Cassette Drive

Figure 1-1. System 2200 Tape Cassette Drives

CHAPTER 1

GENERAL INFORMATION

1.0 INTRODUCTION

Tape cassette drives are available in two models as peripheral devices for the Wang System 2200: the Model 2217 Single Tape Cassette Drive and the Model 2218 Dual Tape Cassette Drive. See Figure 1-1.

The single cassette drive is available either as a separate physical unit or housed in the chassis of the CRT (Cathode Ray Tube) Executive Display. As a separate unit, the single drive is called the Model 2217 Single Tape Cassette Drive. Housed in the CRT chassis of a combination unit, the single drive is included in each of the following units:

a) the Model 2216/2217 Combined CRT Executive Display/Single Tape Cassette Drive,

b) the Model 2216A/2217 Combined Upper/Lower Case CRT Display/Single Tape Cassette Drive, and

c) the Model 2220 Console-CRT/Keyboard/Single Tape Cassette.

With the exception of Model 2220 units, a connector cord from the single drive plugs into a peripheral controller board labeled Model 2217. The board, like those for the other System 2200 peripherals, is mounted in the CPU (Central Processing Unit) chassis. For Model 2220 units, a connector cord from the single drive plugs into a special two-outlet controller board designed for the keyboard and the cassette drive.

The Model 2218 Dual Tape Cassette Drive is always a separate physical unit, housing two tape cassette drives whose connector cords plug into one double-outlet peripheral controller board labeled Model 2218. The drives are operated independently under BASIC language control in the Immediate Mode or Program Mode by using the unique device address assigned to each drive. With both drives plugged into one controller board, the Model 2218 offers advantages over two Model 2217 units. In addition to the smaller purchase price, space is freed in the CPU for an additional peripheral controller board.

A maximum of six tape cassette drives can be used with one System 2200 Central Processing Unit. Several configurations are possible when combining Model 2217 units (counting as one drive each) with Model 2218 units (counting as two drives each) to increase the number of drives addressable by the System 2200. Moreover, cassette drive units can be added to the system when the need for a greater tape cassette manipulation capability develops, provided the maximum of six drives is not exceeded.

Whether a System 2200 configuration consists of one tape cassette drive or several drives, an unlimited number of tape cassettes can be used to store program files and data files.

1.1 UNPACKING, INSPECTION, AND INSTALLATION

A Wang Service Representative, upon notification, unpacks all equipment and inspects the units for shipping damage.

Installation of Model 2217 and Model 2218 tape cassette drives is the responsibility of a Wang Service Representative.

1.2 POWER-ON PROCEDURE

1. Turn ON the power switch located on each peripheral. Include the CRT. Usually, the rocker type power switch is located on the back of each unit next to a round fuse knob. Figure 1-2 shows the power switch and fuse on the back of a Model 2217 cassette drive. Only one switch activates the Model 2216/2217 combined CRT and cassette drive unit; the switch is on the lower left rear corner of the CRT housing next to the fuse knob. Illumination of the cassette drive light, visible through the door of each cassette drive, confirms that power is on.

2. Turn ON the main switch located on the Power Supply Unit. Illumination of the Power Supply Unit Light confirms this step.

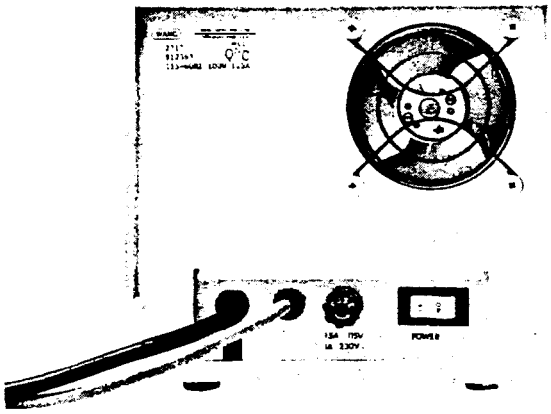


Figure 1-2. Power Switch and Fuse on Back of Model 2217 Single Tape Cassette Drive

Activating the Power Supply Unit, Master Initializes the System 2200. Then, as soon as the message shown in Figure 1-3 appears on the CRT Executive Display, the system is ready for operation.

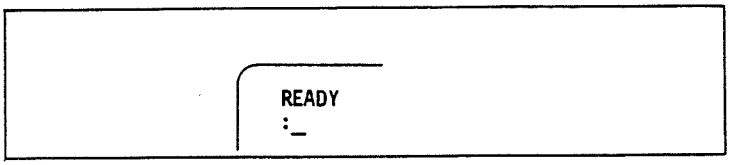


Figure 1-3. System 2200 Ready Condition Display

If a system failure occurs, touch the RESET key on the keyboard in the System 2200 configuration.

If the ready condition is not restored, Master Initialize the system again (i.e., turn the main power switch off and then on again).

If the system remains nonfunctional, repeat the Power-ON procedure before calling your Wang Service Representative.

> NOTE:
>
> When the System 2200 is Master Initialized, all memory is cleared. The primary devices (i.e., the peripherals with default addresses) are selected automatically for all subsequent input/output operations. If a cassette is mounted on a cassette drive, information stored on the tape is not affected.

1.3 CASSETTE STORAGE AND HANDLING

Wang cassettes are packaged in individual plastic storage boxes. When not in use, cassettes should be kept in their storage boxes or other enclosed containers.

The transparent leader on the magnetic tape is exposed along "windows" on the upper edge of each cassette. During tape recording or reading operations, the magnetic tape passes by these windows. Care should be exercised when handling cassettes to minimize exposure to dirt and dust which can affect recording or reading operations adversely. Periodic cleaning of the cassette read/write head is recommended (see Appendix E).

For easy removal of a cassette from its storage box, view the spiked surface of the box as the bottom. The narrow edge on this surface permits unobstructed access to the cassette. When returning a cassette to its box, always place it (label side up) over the spikes for easy handling thereafter (see Figure 1-4).

Figure 1-4. Cassette in Storage Box

1.31 Mounting a Cassette

The procedure for mounting a cassette on a cassette drive is the same for all drives whether the drive is a Model 2217 single drive (housed in the CRT chassis or in a separate unit) or one of the Model 2218 drives housed side-by-side in a dual unit.

First, open the cassette drive door by pressing the white Door Release button located at the right (see Figure 1-5). If the door does not open, gently press the top of the door inward to deactivate the double lock. If the door still does not open, press the Rewind button and repeat the standard procedure.

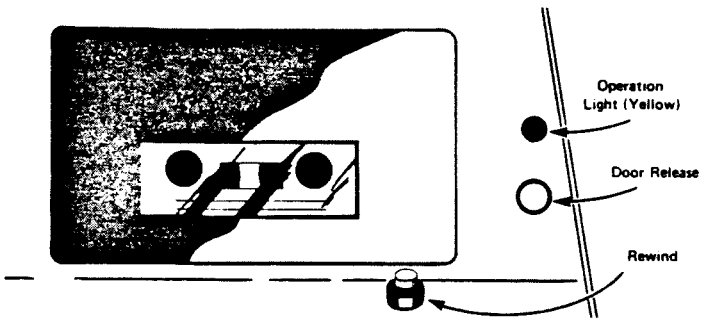


Figure 1-5. Cassette Drive Door and Manual Controls

Slip the cassette (held with its label side facing the user) into place between the brackets on the cassette drive door as shown in Figure 1-6.

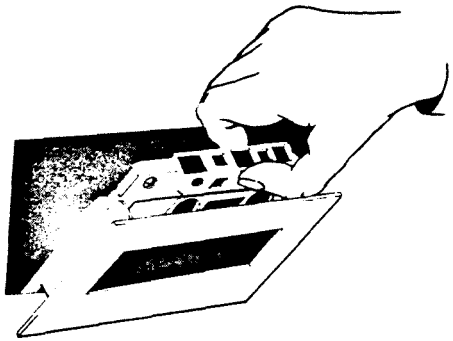


Figure 1-6. Mounting a Cassette

Once the cassette is in the cassette drive door, close the door. Then, rewind the cassette by pressing the Rewind button on the cassette drive housing. Rewinding, at this point, is a precautionary measure to ensure exact positioning of the tape at the beginning before use.

1.32 Removing a Cassette

Before removing a cassette from a drive after use, rewind the tape; a cassette should not be removed until the tape is rewound completely. Then, press the Door Release button on the drive housing. If the door does not open, gently press the top of the door inward to deactivate the double lock.

1.4 REWINDING A TAPE CASSETTE

To rewind a tape cassette, either:

1) press the Rewind button on the cassette drive housing, or

2) enter the word REWIND on a System 2200 keyboard and then touch the EXECUTE key.

Both methods can be used during Immediate Mode operations. The second method (with a specified line number) is used for Program Mode operations.

In programs written for multi-cassette-drive applications, the device address or file number of the drive to be accessed is included in the REWIND statement using a special format; e.g., REWIND /10C or REWIND #3. Otherwise, when the system encounters a REWIND statement, the cassette drive last selected for TAPE-class operations is activated (see Section 1.9).

1.5 MOTION INDICATORS FOR CASSETTE DRIVES

When a cassette drive is in motion, the Operation Light (yellow) on the cassette drive housing is illuminated. Also, the motion is visible through the cassette drive door window.

> Note:
>
> 1. Do not attempt to remove a cassette when a drive is in motion.
>
> 2. To interrupt cassette drive motion and regain control of the system, touch the RESET key.

1.6 RECORDING/READING OPERATIONS FOR ONE-DRIVE CONFIGURATIONS

Recording programs on tape cassettes and reading programs from cassettes to memory are straightforward operations in the System 2200. Some how-to-save (recording) and how-to-load (reading) procedures for one-cassette-drive configurations are introduced here. Additional information is given in Chapter 2.

In a typical one-drive configuration, the cassette drive is housed in the CRT chassis (Model 2216/2217). However, the procedures described here are applicable to any one-drive configuration since the device address of the drive is the default address 10A (see Sections 1.8 and 1.9).

To record a program already entered in the system memory:

1. Mount an unused or scratch cassette on the drive.

2. Enter the command SAVE and touch the RETURN (or EXECUTE) key.

Automatically, the program text currently in memory is recorded on the cassette in as many 256-byte physical records as necessary. A program file is formatted with a header record, program records (as many as needed), and a trailer record. (Furthermore, the program text remains in memory.) When the recording operation is completed, the System 2200 ready-condition-colon appears on the CRT Executive Display. Then, the cassette can be rewound, removed from the drive, and stored until the program is needed again.

An operator can read the recorded program into memory as follows:

1. Mount the cassette containing the recorded program on the drive.
2. Enter CLEAR and touch the EXECUTE key.
3. Enter LOAD and touch the EXECUTE key.

After the reading operation is completed, the ready-condition-colon appears on the CRT Executive Display. Then the cassette can be rewound, removed from the drive, and stored until the program is needed again.

To record a program on a cassette without overwriting (and thereby destroying) program files already on a cassette, use a SKIP statement (see Appendix C) in addition to the SAVE command. For example, to add a third program file to a cassette already containing two program files, proceed as follows:

1. Mount the cassette on the drive.
2. Press the Rewind button (to ensure exact positioning of the tape at the beginning).
3. Enter SKIP 2F and touch the EXECUTE key.
4. Enter SAVE and touch the EXECUTE key.
5. Rewind and remove the cassette.

Similarly, to read a program known to be the fifth program on a cassette, proceed as follows:

1. Enter CLEAR and touch the EXECUTE key.
2. Mount the cassette on the drive and press the REWIND button.
3. Enter SKIP 4F and touch the EXECUTE key.
4. Enter LOAD and touch the EXECUTE key.
5. Rewind and remove the cassette.

1.7 IDENTIFYING RECORDED PROGRAMS

When recording a program on a cassette, a name (from one to eight bytes in length) can be assigned to the file. By naming a file when it is recorded, an advantage is gained. A cassette can be searched for a particular file referenced by name.

To assign a program file name during a recording operation, enclose the desired name in double quotation marks in the SAVE command. For example, to identify a particular program as PROG3, enter the command:

```
SAVE "PROG3"
```

and then touch the EXECUTE key. Automatically the name is stored in the header record when the program file is formatted by the System 2200. Likewise, to identify another program on the cassette as METHOD A, use the command:

```
SAVE "METHOD A"
```

where the blank embedded in double quotation marks is counted as one of the eight bytes.

At a future time, the cassette can be searched for a specified file, referenced by name, without using a SKIP statement. For example,

```
LOAD "PROG3"
```

or

```
LOAD "METHOD A"
```

During execution of a statement of the form LOAD "name", the cassette is searched forward only for the specified program file. Whenever a header record is encountered during the search, the name in the header record is compared to the specified name. If a match (including any embedded blanks) occurs, the program text in the file is read into memory.

If the current tape position (with respect to the cassette read/write head) is already beyond the location of a specified program when execution of a LOAD statement begins, the specified program cannot be found. Therefore, unless the sequential position of a particular program is known to be beyond the current tape position, always rewind a cassette before initiating a search.

See Chapter 2 for more information on recording/reading program files. See Chapter 3 for information on recording/reading data in logical records and data files.

1.8 CASSETTE DRIVE ADDRESSES

An understanding of device address codes and device selection techniques is important, especially for System 2200 configurations with more than one cassette drive. The essential features of cassette drive addresses apply to other peripherals in the system.

For program control, the device address of every System 2200 peripheral unit is specified by a three-hexadecimal-digit code of the form xyy. The first hexadecimal digit (the leftmost or x digit) represents the Device Type. The last two hexadecimal digits (yy) represent the Preset Address of the specific controller board into which the peripheral is plugged.

A list of device address codes for System 2200 peripherals is given in Appendix A. The list provides a standard for unique address codes; for example, six unique device address codes are reserved for cassette drives: 10A, 10B, 10C, 10D, 10E, and 10F.

The last two hexadecimal digits in the device address correspond to the 8-pole setting on an address switch on the printed circuit board which controls a cassette drive. The 8-pole address switch is not visible after the controller board is installed in the CPU. Each Model 2217 board has one device address switch. Each Model 2218 board has two device address switches. Address switches are set by the factory before shipment or by a Wang Service Representative when a controller board is installed.

Normally, a System 2200 configuration with a Model 2216/2217 combined CRT Executive Display/Single Tape Cassette Drive is set up with the address code 10A for the "first" cassette drive, the drive housed in the CRT chassis. If a separately housed Model 2217 single drive unit is added to the system, the address code 10B is assigned to that drive. If, alternatively, a Model 2218 dual drive unit is added to the Model 2216/2217, the addresses 10B and 10C are assigned to those drives.

On the other hand, a System 2200 configuration with a Model 2216 (CRT Executive Display only) is assigned cassette drive addresses as follows. The first separately housed Model 2217 drive added to the system is set up with the address 10A. If, alternatively, a Model 2218 is added to a system having no Model 2217 units, the addresses 10A and 10B are assigned to the drives in the dual unit.

Address uniqueness for cassette drives is essential. Furthermore, the address 10A is significant. Address 10A is assigned to one cassette drive in each configuration because 10A is one of the five default addresses of the System 2200 (see Section 1.9).

To avoid confusion when operating a system with more than one tape cassette drive, attach a label near the door of each drive showing its address code. Keep in mind, however, that the address of a tape cassette drive is determined by the preset address of the controller board into which the drive is plugged. The model number and device address code (or codes, for the dual drive Model 2218) are given on each controller board.

1.9 CASSETTE DRIVE SELECTION

When the System 2200 is Master Initialized, the "primary devices" are selected automatically for input/output operations. The keyboard (with address 001) and the CRT (address 005) are two of the five primary devices. The cassette drive with address 10A is another. The complete list of default addresses is given in Appendix B (see Table B-1).

Users of a System 2200 equipped with only one cassette drive can mount a cassette on the drive and instruct the system to perform any cassette operation without explicitly specifying the device address of the drive (since the preset address of the drive is the default address 10A).

Thus, no direct or indirect address specification is needed in actual cassette statements (see Appendix C, Table C-1). Also, no SELECT statement of the form

```
SELECT TAPE 10A
```

is needed in a program to be executed exclusively on a one-cassette-drive configuration.

Users of a System 2200 equipped with more than one cassette drive can use drive 10A as though the multidrive configuration is a one-drive configuration. After a multi-cassette-drive system is Master Initialized, the system accesses drive 10A for execution of all subsequent cassette operations unless one of the following conditions exists:

1. a different address is specified directly or indirectly in a cassette statement, or

2. a different address has been assigned to the TAPE-parameter by execution of a SELECT statement.

For example, after execution of the statement

```
SELECT TAPE 10B
```

the drive with address 10B becomes the "current" cassette drive. Later, if the system executes a REWIND statement, the cassette on drive 10B is rewound. However, if the system executes a REWIND /10C statement, the cassette on drive 10C is rewound. The address specified within a cassette statement is always honored first. When no address is specified, the drive last selected for TAPE-class operations is assumed.

In Appendix B, the general format of the SELECT statement is discussed along with examples. The I/O operations are summarized in Chart B-1, showing the eight classes into which these operations are grouped. The I/O class parameters identifying the groupings are given. A study of the chart shows that the class parameter for all I/O operations related to cassette drives is the parameter TAPE.

For a multi-cassette-drive system, the SELECT statement can be used to assign cassette drive addresses to file numbers. For example, in a configuration with three cassette drives having the device addresses 10A, 10B, and 10C, the statement

```
SELECT #1 10A, #2 10B, #3 10C
```

assigns the address 10A to #1, 10B to #2, and 10C to #3. Only the file numbers one through six can be assigned a cassette drive address.

The term "file number" defines the special notation #n (where n = 1, 2, 3, 4, 5, or 6). Any reference to a "file number" should not be confused with the terms "program file" and "data file" in this manual or in the System 2200 A/B Reference Manual.

A file number, if used as a parameter in a SELECT statement, requires a device address (with no comma between). When executed, the SELECT statement assigns the device address to the file number; but the assignment does not alter the current address assigned to TAPE-class operations. Several file number assignments can be made in one SELECT statement (as illustrated above) by separating each successive pairing of a file number and device address from the previous pairing by a comma.

After a cassette drive address is assigned to a file number by a SELECT statement, the system accepts the file number as an "indirect device address specification" in a cassette statement. For example, if the statement REWIND #3 appears in the logic of a program (after a file-assignment-SELECT-statement), the cassette drive assigned to #3 is implied.

CHAPTER 2

FORMATTING PROGRAM FILES

2.0 INTRODUCTION

Automatic formatting of information recorded on tape cassettes is a feature of the System 2200. Information is recorded in 256-byte blocks called physical records and each physical record is recorded twice to ensure the integrity of the information.

Program text or data can be recorded on tape cassettes. However, as a general programming practice, programs and data are stored on different cassettes. To facilitate recovery of specific programs, each program is formatted as a "file" automatically. BASIC language statements used to record/read program files are discussed in this chapter.

Statements used to record/read data are discussed in Chapter 3.

2.1 PROGRAM FILES

When the System 2200 executes a SAVE command, program text currently in memory is recorded on tape in the format of a program file. A program file consists of three types of physical records: a header record (only one), program records (as many as required to store the program text), and a trailer record (only one). See Figure 2-1. The header and trailer records identify the beginning and end of program text when the tape is read. The three types of records are described in detail in Sections 2.11, 2.12, and 2.13.

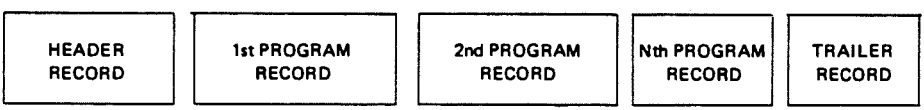


Figure 2-1. Program File Format

Each physical record in a program file contains one control byte, a maximum of 254 bytes of valid information, and one EOB (end-of-block) byte or EOF (end-of-file) byte. A byte is an 8-bit sequence of the binary digits zero and one.

The System 2200 uses several different 8-bit codes as control bytes. The codes specify information such as the following:

1. Whether a physical record belongs to a program file or a data file.

2. Whether a record is a header record or not,

3. Whether a record is a trailer record or not,

4. Whether a nonheader-nontrailer data record is an intermediate or a final record in a logical record (see Chapter 3).

2.11 Header Record

The first byte in a 256-byte header record is a System 2200 control byte which identifies the physical record as the header record for a program file.

The next eight bytes in a header record contain the program file name, up to eight characters. If no name is specified in the SAVE command, the system supplies a dummy name. If a name with less than eight characters is specified, the system fills the unspecified bytes with dummy characters which are ignored when checking the program name.

The tenth byte in a header record is the special EOB (end-of-block) byte which indicates the end of valid information in the record.

The remaining storage capacity beyond the EOB byte (246 bytes) is not used when the program is recorded. Extraneous information beyond the EOB byte in the header record is ignored when the record is read.

2.12 Program Record

Each program record begins with a control byte which identifies the 256-byte block as a program record.

Next, the program record contains a line of program text (the first line of text if the record is the first program record in the the file). The number of bytes of storage required for each line of program text varies from one text line to another but does not exceed 192 bytes. (A line of program text, when entered, is restricted to 192 characters).

As many complete lines of program text as possible are included in one program record. A partial line of program text is not stored in a program record.

When insufficient bytes of storage remain for another line of text, the system automatically supplies a special EOB (end-of-block) byte. Any remaining storage capacity in the record is not used.

Additional program records are formatted, as required.

2.13 TRAILER RECORD

A trailer record begins with a control byte which identifies the 256-byte block on tape as the final physical record of a program file.

Lines of program text may be included in the trailer record. If so, the final line of program text is the last line of text in the trailer record.

Then, the record contains the special EOF (end-of-file) byte used by the system to designate the end of the program file.

Any storage capacity beyond the EOF byte in the trailer record is not used.

2.2 RECORDING A NAMED PROGRAM

A name is not required when a program is recorded on tape. However, if a name is specified, an advantage is gained. The system can search a cassette for a particular program file referred to by name.

Before recording a particular program, decide on a descriptive program name consisting of eight bytes or less. Use any desired combination of characters: alphabetic, numeric, or special characters (except double quotation marks). Make the name unique, as well as descriptive; for example, PROGRAM1, COMP5, TAXCALC, *PROG, CALC%.

Now, proceed as follows:

1. Mount a new or scratch cassette on drive 10A or on the drive last selected for TAPE-operations (if different from the default address 10A).

2. Touch the Rewind button to position the tape exactly at the beginning.

Next, assuming the program to be recorded is currently in memory and the desired name for the program file is COMP5,

3. Enter SAVE "COMP5" and touch the EXECUTE key.

The system automatically formats a program file with the specified name recorded in the header record. As soon as the recording operation is completed, the ready-condition-colon appears on the output device currently selected for CO (console output) operations -- usually the CRT Executive Display (by default). Finally,

4. Rewind, remove, and store the cassette.

2.3 SKIPPING AND BACKSPACING PROGRAM FILES

The System 2200 provides the capability to skip or backspace a specified number of program files on a cassette (whether the files are named or not). The following statements can be used in the Immediate Mode or Program Mode (assuming the cassette is mounted on the default drive):

```
SKIP mF
BACKSPACE mF
```

where "m" represents an integer or a mathematical expression. If an expression is specified, the system evaluates the expression and then truncates the result to an integer which must be ≥ 1. See Appendix C for the general forms of the SKIP and BACKSPACE statements.

Upon execution, a SKIP statement advances the tape from its current position until "m" trailer records have been read. The read/write head is positioned at the end of the mth trailer record. Therefore, if the number of programs already recorded on a cassette is known, a SKIP statement can be used to position the cassette read/write head at the end of the last recorded program before recording an additional program (as illustrated in Section 1.6).

Upon execution, a BACKSPACE statement reverses tape motion from its current position until "m" header records have been read. The read/write head is positioned in front of the mth header record.

2.4 CASSETTE LABEL UPDATING

An up-to-date reference list of the names of programs recorded on a cassette, in sequential order, should be maintained. The label on the cassette is a good place to maintain such a list. By consulting the list, an operator can determine the number of files to skip before recording another program on the cassette, thereby avoiding the danger of overwriting and destroying existing programs.

After a program is recorded on tape, add the program file name to the list on the cassette label.

2.5 PROTECTING A TAPE CASSETTE

Unless provision is made to position the read/write head beyond any program or data files of interest, a new program may be recorded accidentally on a portion of tape containing an important program or data. The overwritten program or data is destroyed.

A technique is available to prevent accidental overwriting of programs or data. Each tape cassette is equipped with a pair of orange plastic tabs on the bottom edge. If the tabs are turned inward to expose square holes at each end of the bottom edge, the cassette becomes a "protected tape" (not to be confused with a protected program file).

Program files already recorded on a protected cassette can be read into memory but no new files can be recorded until the tabs are reversed and the holes are covered. Actually, only the tab on the right (when viewing the label side) need be covered to record a program.

If a SAVE command is issued while a protected tape cassette is mounted on the cassette drive, an error message (Code 50) appears on the CRT Executive Display below the line containing the command. Code 50 indicates the cassette is protected and recommends mounting another cassette or covering the exposed holes on the bottom of the present cassette. If the plastic tabs are missing, masking tape can be used to cover the holes.

2.6 RECORDING A PROGRAM - DETAILED CHECKLIST

A checklist for recording programs on a tape cassette is given below. Any program can be used to test the procedure. For sample programs, refer to the System 2200A/B BASIC Programming Manual.

1) Clear the memory.

2) Enter the complete program text in memory, using the keyboard.

3) Mount the cassette on the cassette drive in the CRT housing (or the separate drive with address 10A).

4) Press the Rewind button on the housing to ensure exact positioning of the tape at the beginning.

5) If other programs are already recorded on the cassette, determine the number of files to be skipped and enter the appropriate statement:

    SKIP mF

    replacing "m" by the proper integer.

6) Decide on a descriptive name for the program (eight bytes or less) and enter the specified name in the following statement:

    SAVE "name"

    then touch the EXECUTE key.

After tape motion ceases and the ready-condition-colon appears on the CRT, the program is recorded on the cassette. Now continue as follows:

7) Rewind and remove the cassette from the drive.

8) Update the cassette label by adding the name of the new program.

9) Store the cassette in its box or a covered container.

2.61 Recording Portions of a Program

Sometimes a subroutine or a selected portion of a program currently in memory is sufficiently general to warrant its recording as a separate program file. If so, the System 2200 provides such a capability.

For example, to record Lines 350 through 620 of a program currently in memory as a separate program file named SUB1 enter:

```
SAVE "SUB1" 350, 620
```

and touch the EXECUTE key.

If only one line number is specified in a SAVE command, the system records the program text starting with the specified line number and continuing to the end of the program currently in memory. (See Appendix C.)

2.7 PROTECTING A PROGRAM FILE

If desired, a program can be recorded as a "protected program file" by including the optional parameter P in the SAVE command. A protected program file can be loaded into memory from tape and executed by standard procedures, but the text cannot be listed nor rerecorded from memory.

> Note:
>
> After a protected program has been loaded into memory and executed, the memory should be cleared by Master Initializing the system or issuing a CLEAR command (with no parameters). Otherwise, subsequent programs loaded or entered into memory cannot be listed or saved.

2.8 RETRIEVING A NAMED PROGRAM

If a program file was named when recorded on a cassette, the program can be read into memory without first positioning the cassette read/write head in front of the header record of the desired program. Upon execution, a statement of the form:

```
LOAD "name"
```

with the actual name of a program file specified in double quotation marks, advances the tape from its current position. When a program file header record is encountered, the name stored in the record is compared with the name specified in the LOAD statement. If the names match, the program text is read into memory. If the names do not match, the search continues.

> Note:
>
> Since a cassette tape is read forward only from its current position when execution of a LOAD "name" statement begins, a cassette should be rewound before a search is initiated unless the sequential position of the desired program is known to be beyond the current position of the tape with respect to the read/write head.

2.9 PROGRAM RETRIEVAL - ALTERNATIVE METHODS

Check the sequential list of programs on a cassette label. Then, use either Method A or Method B to retrieve (i.e., read into memory) a particular program.

Method A: Determine the exact program-file-name from the cassette label. Specify the name in the statement

```
LOAD "name"
```

and touch the EXECUTE key.

Method B: Determine the number of files to be skipped to reach a desired program from the current tape position. Substitute this number in place of the lower case m in the statement:

```
SKIP mF
```

and touch the EXECUTE key. Then enter the statement:

```
LOAD
```

and touch the EXECUTE key.

Since no program file name is specified in Method B, the system reads the next program into memory without first checking the name stored in the header record.

CHAPTER 3

FORMATTING DATA FILES

3.0 INTRODUCTION

Data are recorded on tape cassettes in 256-byte physical records formatted automatically by the System 2200 upon execution of DATASAVE statements. When a DATASAVE statement with an argument list is executed, a logical record is created. A logical record consists of one or more physical records; i.e., as many data records as needed to store all the values corresponding to the argument list in the statement. If another DATASAVE statement with an argument list is executed, a different logical record consisting of one or more data records is formatted. A data file is not formatted automatically.

To create a data file, a header record and a trailer record are needed. A separate DATASAVE statement with the parameter OPEN, which requires a file name, is used to instruct the system to format a header record. A separate DATASAVE statement with the parameter END is used to instruct the system to format a trailer record.

In this chapter, programming techniques for recording data in logical records and creating data files are discussed. Also, data retrieval and the rewriting of data records are discussed. Examples of data recording/reading operations for one-cassette-drive configurations are included. Multi-cassette-drive examples are given in Chapter 5.

3.1 RECORDING DATA ON A CASSETTE

DATASAVE statements with argument lists are used to record data on a tape cassette (see the general form in Appendix C, Table C-1). The argument list of a DATASAVE statement can include one or several of the following arguments in any combination:

a) a numeric variable; e.g., A, B1, C5, D

b) an alphanumeric variable; e.g., A$, B2$, M4$

c) a specific element of a one-dimensional numeric or alphanumeric array; e.g., A(2), F1(3), X(10), B$(4), G3$(5)

d) a specific element of a two-dimensional numeric or alphanumeric array; e.g., A(3,4), Y5(2,9), A(1,15), C$(5,7), D1$(4,9)

e) an array designator (an array name followed by a left and a right parenthesis); e.g., A(), P$(), M4$().

f) a mathematical expression; e.g., X*Y-Z, SQR(A↑2 + B↑2), 2*D, X

g) a string function or hexadecimal function; e.g., STR(A$,3,8), HEX(22), HEX(0D0A).

h) a literal string; e.g., "DATA GENERATED BY PROGRAM ONE", "WANG LABORATORIES", "TEST 1"

> NOTE:
>
> When the argument list of a DATASAVE statement contains one or more array elements or array designators, the program logic must specify the dimension of each array in a DIM statement to be executed prior to the DATASAVE statement.

Illustrations of data recording operations are given in Examples 3-1 and 3-2. Duplication of the examples is not recommended since no provision is made in the program logic for reading the recorded data and displaying the result. (Logic for reading recorded data is illustrated in the examples included in Section 3.5). Examples 3-1 and 3-2 illustrate cassette storage requirements imposed by DATASAVE statements (see Table 3-1 in Section 3.2).

Example 3-1

```
10 DIM D(15), C$30
20 A=125.30 :B=17.60 :C$="J. JONES"
30 FOR K=1 TO 15
40 D(K)=.05*K
50 NEXT K
60 DATASAVE A, B, C$, D(3)
```

Line 60 in Example 3-1 instructs the system to record the value for each of the following variables: A, B, C$, and the third element of the one-dimensional array D().

Example 3-2

```
 10 DIM A(40), B(10,8)
 20 FOR J=1 TO 40
 30 A(J)=1200-J*30
 40 NEXT J
 50 FOR K=1 TO 10
 60 FOR L=1 TO 8
 70 B(K,L)=300+5*L+2*K
 80 NEXT L
 90 NEXT K
100 DATASAVE A(), B()
```

Line 100 in Example 3-2 instructs the system to record the value for each element in the one-dimensional array A() and the two-dimensional array B().

3.2 DATA STORAGE REQUIREMENTS IN MEMORY AND ON CASSETTES

In memory, bytes of storage are reserved for the names of variables and arrays in addition to their currently assigned values. For the value of each numeric variable, eight bytes of internal storage are sufficient (even for System 2200 free form values with up to thirteen significant digits). On the other hand, the number of bytes of internal storage allotted for the value of each alphanumeric variable is equal to the specified length (given in a DIM or COM statement) or the default length, 16 bytes, if no specification is made for a particular alphanumeric variable.

For example, upon execution, the statement:

```
30 DIM A$10, B$(4)6
```

assigns a length of 10 bytes to the variable A$ and a length of 6 bytes to each of the four elements of the one-dimensional array B$().

When data is recorded on a cassette, the names of variables and arrays in the argument list of a DATASAVE statement are not recorded. However, the system automatically supplies an SOV (start of value) byte preceding each recorded value. The first bit in each SOV byte identifies whether the value which follows is a numeric or an alphanumeric value. The remaining bits in each SOV byte indicate the total number of bytes in the recorded value.

On a tape cassette, every numeric-variable-value occupies nine bytes of storage (eight bytes plus the SOV byte). Every alphanumeric-variable-value occupies the dimensioned length (any number from 1 to 64 bytes) plus an SOV byte, or the default length (16 bytes) plus an SOV byte.

In Table 3-1, the memory and cassette storage requirements for the arguments in the DATASAVE argument lists in Example 3-1 (Line 60) and Example 3-2 (Line 100) are given as illustrations:

Table 3-1. Memory and Cassette Storage Requirements for Examples 3-1 and 3-2

DATASAVE Statement	Argument List	Variable Type	Memory Storage*	Tape Storage
Example 3-1 (Line 60)	A	numeric	8 bytes	9 bytes
	B	numeric	8 bytes	9 bytes
	C$	alphanumeric	30 bytes	31 bytes
	D(3)	numeric-array-element	8 bytes	9 bytes
			(total=54)	(total=58)
Example 3-2 (Line 100)	A()	numeric array (40 elements)	40x8=320 bytes	40x9=360 bytes
	B()	numeric array (80 elements)	80x8=640 bytes	80x9=720 bytes
			(total=960)	(total=1080)

*Does not include storage required for the name of the variable or array.

When a DATASAVE argument list contains a mathematical expression; e.g., A+5*B, nine bytes of cassette storage are needed for the value of the expression. The nine bytes include the SOV byte and the eight bytes of the numeric value.

When a DATASAVE argument list contains a string (STR) function, the number of bytes of cassette storage required is always one more than the number of consecutive bytes specified in the STR function. For example, STR(A$,3,5) requires six bytes.

When a DATASAVE argument list contains a hexadecimal function, the number of bytes of cassette storage required for the function is always one more than the number of bytes in the function. For example, HEX(220D0A) requires four bytes: the SOV byte, one byte for the HEX(22) character, one byte for the HEX(0D) character, and one byte for the HEX(0A) character.

When a DATASAVE argument list contains a literal string, the number of bytes of cassette storage required for the string is always one more than the number of characters (bytes) enclosed in quotation marks. For example, the literal string "WANG LABORATORIES" requires 18 bytes, including the SOV byte.

3.3 LOGICAL RECORDS

As illustrated in Example 3-2, the total number of bytes of cassette storage required for the argument list of one DATASAVE statement can exceed 256 bytes, the block size of one physical record. For the DATASAVE statement in Example 3-1, the values require 58 bytes of cassette storage. In Example 3-2, the values for the variables in one DATASAVE statement require 1080 bytes of storage, the capacity of more than four physical records. See Table 3-1.

A logical record is defined as the one or more data records (256 bytes each) formatted automatically by the System 2200 when executing one DATASAVE statement. See Figure 3-1. Each data record in a logical record begins with two control bytes (a program record begins with only one control byte).

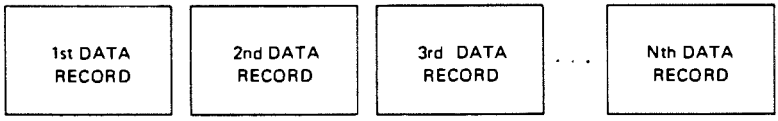


Figure 3-1. Logical Record Format

The first control byte in each data record is one of the two different System 2200 8-bit codes used to identify a physical record as a data record within a logical record. One code is used if the 256-byte block is an intermediate (or the first) record in a logical record; the other code is used if the 256-byte block is the last record in a logical record.

The second control byte in each data record contains the sequential number of the data record in the logical record formatted by a DATASAVE statement. The second control byte is also the second byte in a data record.

Next, each data record contains the values corresponding to some or all of the arguments in the DATASAVE argument list. Every value in the record begins with an SOV (start-of-value) byte indicating whether the value corresponds to a numeric or alphanumeric variable and giving the count (number) of bytes in the value.

A data record never contains a partial value for a variable or an array element. If the value of a variable or an array-element does not fit into the 256-byte block currently being recorded, the value is recorded automatically in the next data record.

Each data record contains a third control byte which identifies the end of valid data. An EOB (end-of-block) byte is supplied by the system following the last value recorded in the data record. If the EOB byte is written before 256 bytes are recorded, the remainder of the record is not used to record data and is ignored when the record is read.

Because each data record contains three control bytes, a maximum of 253 bytes remain for storing values in a data record. Since numeric-variable-values occupy nine bytes while alphanumeric-variable-values occupy as little as two bytes or as many as 65 bytes, the number of bytes of valid data in a record depends upon the number, the type, and the dimension of the arguments in a DATASAVE statement.

Users of the System 2200 do not need to determine the number of data records required by a DATASAVE argument list because the system formats a logical record and its data records automatically. Furthermore, users of the system do not need to determine the precise data record on which a value for a particular variable is recorded.

However, for those users who may be interested in determining the number of data records in a logical record, Example 3-2 is considered again. (See Section 3.1.) In Line 100 of the example, the statement DATASAVE A(), B() instructs the system to record all the elements in the A-array and the B-array. In Line 10, the A-array is specified as a one-dimensional array with 40 elements. Elements of a one-dimensional array are recorded on tape in sequential order: A(1), A(2), A(3),..., A(40). Also, the B-array is specified as a two-dimensional array with 10 rows of 8 elements each. Elements of a two-dimensional array are recorded row-by-row in sequential order: B(1,1), B(1,2),..., B(1,8), B(2,1), B(2,2),..., B(2,8), B(3,1), B(3,2),..., B(3,8),..., B(10,1), B(10,2),..., B(10,8).

The logical record for the DATASAVE statement in Example 3-2 must contain 120 numeric-array-elements, each requiring nine bytes of tape storage (an SOV byte and eight bytes duplicating the value-stored-in-memory). Therefore, a total of 120 x 9 = 1080 bytes must be recorded on tape. However, at most 253 bytes can be recorded in one data record (three bytes are used for control information and no partial value is recorded).

Since every array-element-value in Example 3-2 requires nine bytes, the calculation of the number of values per record is straightforward (253/9 = 28 values, plus one unused byte of storage per record). The format of the logical record for Example 3-2 is given in Table 3-2.

Table 3-2. Logical Record Format for Example 3-2

Data Record	Values Recorded	Number of Bytes
1	A(1) through A(28)	Control bytes: 3 Data bytes: 28x9= 252 Unused bytes: 1 TOTAL 256
2	A(29) through A(40) B(1,1) through B(2,8)	as above
3	B(3,1) through B(6,4)	as above
4	B(6,5) through B(9,8)	as above
5	B(10,1) through B(10,8)	Control bytes: 3 Data bytes: 8x9 = 72 Unused bytes: 181 TOTAL 256

Two more examples are introduced now. The first three lines of the program logic in each example are the same. In Example 3-3, the DATASAVE statement (see Line 50) is outside the FOR/NEXT loop; therefore, only one logical record is formatted when the program is executed. In Example 3-4, the same DATASAVE statement (see Line 40) is inside the FOR/NEXT loop; therefore, a logical record is formatted each time the loop is executed.

Example 3-3

```
10  DIM X(7), Y(7)
20  FOR J=1 TO 7
30  X(J) = 15+J : Y(J) = 25+2*J
40  NEXT J
50  DATASAVE X(), Y()
```

If Lines 10 through 50 in Example 3-3 are entered via the keyboard and the command RUN is given (after a cassette is mounted on cassette drive 10A), the following action occurs. First, space is reserved in memory for the variables in the program; the value of each numeric variable is set to zero (if there were any, each alphanumeric variable would be assigned a space character). Then, Line 30 in the FOR/NEXT loop is executed seven times; each time Line 30 is executed the value of the Jth element of each array is stored in the memory location reserved for the Jth element of each array.

Line 50 in Example 3-3 is executed once. The current value of each of the seven elements in the X-array and the Y-array is recorded on tape in one logical record. Therefore, the logical record contains the values 16, 17, 18, 19, 20, 21, 22 for the X-array followed by 27, 29, 31, 33, 35, 37, 39 for the Y-array.

Example 3-4

```
10  DIM X(7), Y(7)
20  FOR J=1 TO 7
30  X(J) = 15+J:: Y(J) = 25+2*J
40  DATASAVE X(), Y()
50  NEXT J
```

If the command RUN is given after the logic of Example 3-4 is entered and a cassette is mounted on drive 10A, action differs from the description of Example 3-3 as follows. The DATASAVE statement is executed immediately after the Jth element of each array is evaluated. Then, the index J is incremented and Lines 30 and 40 are executed again. Therefore, seven different logical records are formatted.

The first logical record contains the values 16, 0, 0, 0, 0, 0, 0 for the X-array-elements followed by the values 27, 0, 0, 0, 0, 0, 0 for the Y-array-elements.

The second logical record contains the values 16, 17, 0, 0, 0, 0, 0 for the X-array followed by 27, 29, 0, 0, 0, 0, 0 for the Y-array.

The third, fourth, fifth, and sixth logical records are similar and are not described here.

The seventh logical record contains the values 16, 17, 18, 19, 20, 21, 22 for the X-array followed by the values 27, 29, 31, 33, 35, 37, 39 for the Y-array.

Hence, the seventh logical record generated by the program in Example 3-4 is the only logical record which duplicates the single logical record generated by the program in Example 3-3.

3.4 DATA FILES

Unlike the SAVE command which generates a program file, a single DATASAVE statement does not generate a data file. Data files (see Figure 3-2) consisting of a header record, one or more logical records, and a trailer record offer the same advantages as program files when several files are recorded on one cassette. Also, data files simplify the task of positioning the read/write head in front of a particular logical record to be read. However, the header record and trailer record for a data file are not generated automatically by the System 2200.

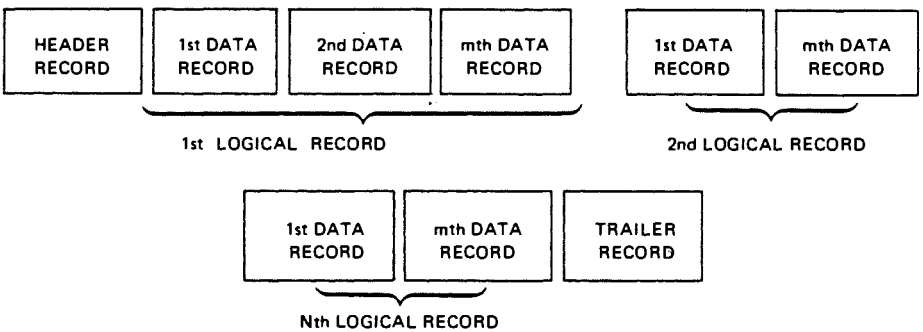


Figure 3-2. Data File Format

To write a header record for a data file, use a DATASAVE statement with the parameter OPEN which requires a specified name (up to eight bytes long) enclosed in double quotation marks. For example,

```
DATASAVE OPEN "TAXDATA"
```

Upon execution, the name TAXDATA is recorded in a data file header record.

Similarly, to write a data file trailer record (after one or more DATASAVE statements with argument lists are executed) use a DATASAVE statement with the parameter END. For example,

```
DATASAVE END
```

No data is recorded in a data file trailer record.

Once a data file is recorded on a cassette, the cassette can be searched by using a DATALOAD "name" statement with the file name specified in quotation marks. The cassette is searched (forward only) until a data file header record with the specified name is found. The read/write head remains positioned at the end of the header record, just in front of the first logical record in the data file. No data values are read.

A sample data file is generated by the logic in Example 3-5. If the example is duplicated by a System 2200 user, a scratch tape should be mounted on the default drive 10A before the program is run. However, the program logic makes no provision for reading the data.

Example 3-5

```
 10 DIM A(200), B(200), S(200)
 20 FOR J=1 TO 200
 30 A(J) = 100+5*J : B(J) = 500-4*J : S(J) = A(J) + B(J)
 40 NEXT J
 50 REWIND
 60 DATASAVE OPEN "CALC1"
 70 DATASAVE A(), B()
 80 DATASAVE S()
 90 DATASAVE END
100 END
```

The data file header record is written when Line 60 in Example 3-5 is executed. The name CALC1 is recorded in the header record. Two logical records are written; one by execution of Line 70, the other by execution of Line 80. The trailer record is written when Line 90 is executed.

3.5 READING CASSETTE DATA

DATALOAD statements with an argument list are used to read cassette data (see the general form in Appendix C). When the System 2200 executes a DATALOAD statement with an argument list, the next logical record on the tape is read if the read/write head is positioned at the beginning of a logical record. (If the read/write head is not in front of a logical record, execution is interrupted by an error message.)

Values read from a logical record (consisting of one or more data records) are assigned sequentially to the memory locations allotted to the arguments in the argument list of the DATALOAD statement under the following conditions:

If the logical record contains more values than required by the argument list, the excess values in the record are ignored (not transferred to memory). Meanwhile, the cassette read/write head moves to the end of the last data record in the current logical record being read, and execution of the DATALOAD statement is terminated.

If the logical record contains fewer values than required by the argument list of the DATALOAD statement being executed, a second logical record is read (unless a trailer record is encountered). The process continues as long as one or more arguments in the argument list remain unmatched by a recorded value (provided another logical record follows the one just completed).

If a data-file-trailer-record is encountered, an end-of-file condition is set. The read/write head is positioned in front of the trailer record. Arguments not yet processed in the argument list retain the values currently in memory and execution of the DATALOAD statement is terminated.

At any point in the reading operation, if the SOV byte for the next recorded value identifies a numeric value when the DATALOAD argument list requires an alphanumeric value (or vice versa), the reading operation is terminated after an error message (Code 43) is displayed. If an Immediate Mode operation is interrupted, the error code appears on the line below the DATALOAD statement being executed. When a Program Mode operation is interrupted, the numbered line of program text is printed out first and the error code appears on the next line of the display. Corrective action is necessary. Either an improper data cassette is mounted and should be changed, or the program text line should be altered.

Now, the program logic in Examples 3-3 and 3-4 is expanded in Examples 3-6 and 3-7 to illustrate the use of DATALOAD statements to read data recorded on a cassette. PRINT statements are included in Examples 3-6 and 3-7 for those System 2200 users who may wish to duplicate the examples and verify the tape recording and reading operations described here.

Example 3-6

```
 10 DIM X(7),Y(7), P(7), Q(7)
 15 STOP "IS SCRATCH TAPE MOUNTED? IF SO, TOUCH CONTINUE & EXECUTE KEYS"
 20 FOR J=1 TO 7
 30 X(J)=15+J : Y(J)=25+2*J
 40 NEXT J
 50 DATASAVE X(),Y()
 60 REWIND
 70 DATALOAD P(),Q()
 80 PRINT
 90 PRINT "VALUES ON THE LOGICAL RECORD ARE:"
100 PRINT
110 FOR K=1 TO 7
120 PRINT P(K),Q(K)
130 NEXT K
140 REWIND
150 END
```

Lines 10 through 50 in Example 3-6 are identical to the logic for Example 3-3, except for the addition of Line 15 containing a reminder to mount a scratch cassette on the cassette drive (default address drive 10A). The logic assumes that the cassette is positioned at the beginning when mounted, so no Rewind operation is programmed before Line 50.

Since data stored in the X and Y arrays remains in memory after the recording operation is executed in Line 50, different locations are reserved in the P and Q arrays for data read when Line 70 is executed. Thus, the credibility of the data printed by Line 120 is ensured without clearing the values in the X-array and Y-array.

When Line 70 is executed, the system reads one logical record. (If the cassette read/write head is not positioned in front of a logical record, execution is interrupted and Error Code 52 informs the operator that a nonheader data record is expected.) Values in the logical record are assigned sequentially to the seven P-array-elements and then the seven Q-array-elements.

Lines 80 through 130 produce the output shown in Figure 3-3. Line 140, when executed, rewinds the tape.

```
VALUES IN THE LOGICAL RECORD ARE:

    16                  27
    17                  29
    18                  31
    19                  33
    20                  35
    21                  37
    22                  39
```

Figure 3-3. Printout from Execution of Example 3-6

Example 3-7

```
 10 DIM X(7),Y(7), F(7), G(7)
 15 STOP "IS SCRATCH TAPE MOUNTED? IF SO, TOUCH CONTINUE & EXECUTE KEYS"
 20 FOR J=1 TO 7
 30 X(J)=15+J : Y(J)=25+2*J
 40 DATASAVE X(),Y()
 50 NEXT J
 60 REWIND
 70 FOR J=1 TO 7
 80 DATALOAD F(),G()
 90 PRINT "VALUES ON LOGICAL RECORD J FOR J=";J;":"
100 PRINT
110 FOR K=1 TO 7
120 PRINT F(K),G(K)
130 NEXT K
140 NEXT J
150 REWIND
160 END
```

Lines 10 through 50 in Example 3-7 are identical to the logic for Example 3-4, except for the addition of the tape-mounting-reminder in Line 15. Since the DATASAVE statement (Line 40) is inside the FOR/NEXT loop, seven logical records are written.

In Line 60, the tape is rewound before the reading operation begins. In Line 10, the one-dimensional F and G arrays (seven elements each) are defined.

Since seven logical records are to be read, the DATALOAD statement (Line 80) is placed inside a FOR/NEXT loop with index J running from one to seven. Each time the DATALOAD statement is encountered, the system begins reading a logical record. Since the argument list of the DATALOAD statement is identical in format to the argument list in the DATASAVE statement, only one logical record is read each time the DATALOAD statement is executed.

Lines 90 and 100 produce a printout heading which documents the value of J each time the DATALOAD statement is executed and a logical record is read. Then, in Lines 110 through 130 the K-loop (nested within the J-loop) produces the seven values of each array corresponding to one value of J. After completion of the J-loop, the tape is rewound (Line 150) and the program terminated (Line 160).

Output from the program in Example 3-7 is not reproduced here. The F and G array values read into memory from seven different logical records are printed. The values from each of the seven logical records are the same as those described in Example 3-4.

Examples 3-3, 3-4, 3-6, and 3-7 show the effects of placing DATASAVE and DATALOAD statements outside or inside a loop. Whether a DATASAVE statement or a DATALOAD statement should be outside or inside a loop depends upon the particular application for which the program logic is designed. The examples presented in this chapter represent no special applications.

> Note:
>
> One general programming practice is recommended. The argument list of a DATALOAD statement used to read a logical record should be identical in format to the argument list of the DATASAVE statement used to write the record. The formats are identical if the number, type, and sequential order of the variables in each list are the same (names may differ).

Carefully formulated departures from the recommended practice of using identically formatted argument lists in DATASAVE and DATALOAD statements can produce satisfactory results. For example, if Line 50 in Example 3-6 is replaced by the following pair of lines:

```
50 DATASAVE X()
55 DATASAVE Y()
```

seven elements of the X-array are recorded in one logical record when Line 50 is executed, and seven elements of the Y-array are recorded in a second logical record when Line 55 is executed. However, the two logical records formatted by Lines 50 and 55 can be read by the DATALOAD statement previously used in Example 3-6:

```
70 DATALOAD P(), Q()
```

Now, because the values in the first logical record do not satisfy the argument list in Line 70, another logical record is read. Furthermore, because the second logical record contains the precise number of numeric values needed to satisfy the argument list in Line 70, the nonidentical argument-list-formats are suitable for reading the recorded data.

3.6 SKIPPING AND BACKSPACING DATA FILES AND LOGICAL RECORDS

Statements used to skip and backspace program files are discussed in Chapter 2, Section 2.3. The same statements can be used to skip and backspace data files as well.

For example, upon execution, the statement:

```
SKIP 5F
```

advances the tape from its current position until five trailer records (whether program file or data file trailer records) have been encountered. The cassette read/write head is positioned at the end of the fifth trailer record encountered.

Similarly, upon execution, the statement:

```
BACKSPACE 3F
```

reverses the tape from its current position until three header records (whether program file or data file header records) have been encountered. The cassette read/write head is positioned in front of the third header record.

Other parameters in the general forms of the skip and backspace statements provide special capabilities when recording/reading logical records. The general forms are as follows:

```
SKIP [#n,  ] {END}
     [/xxx,]  {m  }
              {mF }

BACKSPACE [#n,  ] {BEG}
          [/xxx,]  {m  }
                   {mF }
```

According to the BASIC language syntax summary in Appendix C, vertically stacked items in a general form represent alternatives -- only one of which is to be selected. Items enclosed in brackets are optional. If the stacked items are enclosed in braces, one item must be selected.

The items in brackets in the SKIP and BACKSPACE statements represent alternatives for device address specification. These items should require no clarification in this section since they appear in the general forms of many BASIC language statements for the System 2200. If neither item is specified in a SKIP or BACKSPACE statement, the device last selected for TAPE-class operations (or the default device 10A) is accessed for the operation.

Next, consider the items in braces in the SKIP and BACKSPACE statements. The item mF, where "m" represents an integer or a mathematical expression, has been discussed already in this section and in Section 2.3. Now, observe that "m" can be specified without the parameter F. In such cases, the system looks for control bytes which identify logical records (rather than header or trailer records).

For example, upon execution, the statement:

```
SKIP 5
```

advances the tape from its current position until five logical records have been skipped (or less than five if a trailer record is encountered). The cassette read/write head is positioned at the end of the last data record in the fifth logical record. (Execution is terminated immediately, with an error message, if a program file record is recognized when execution begins.)

Similarly, the statement:

```
BACKSPACE 4
```

reverses the tape from its current position until four logical records have been passed over (or less than four if a header record is encountered). The cassette read/write head is positioned in front of the first data record in the fourth logical record encountered. (Execution is terminated immediately, with an error message, if a program file record is recognized when execution begins.)

The statement:

```
SKIP END
```

can be used to advance the tape to the end of the current data file and position the cassette read/write head in front of the data file trailer record. On the other hand, the statement:

```
SKIP 1F
```

can be used to advance the tape to the end of the current data file and position the read/write head at the end of the data file trailer record.

The statement:

```
BACKSPACE BEG
```

can be used to reverse the tape to the beginning of the current data file and position the cassette read/write head at the end of the header record (that is, just prior to the first logical record in the file). On the other hand, the statement

```
BACKSPACE 1F
```

can be used to reverse the tape to the beginning of the current data file and position the cassette read/write head in front of the header record.

3.7 USING IF END THEN STATEMENTS

An end-of-file (EOF) condition is set during execution of a DATALOAD statement if a trailer record is encountered before enough values are read from tape to satisfy the argument list of the statement. Even though some arguments in the list have been assigned values from tape and others have not, the DATALOAD operation is terminated and program execution continues with the next statement.

An IF END THEN statement provides a technique for program control during the reading of logical records. The IF END THEN statement requires a line number as an argument. The statement instructs the system to branch to the specified line in the program logic if an end-of-file condition is sensed.

By programming an IF END THEN statement (after a DATALOAD statement) and specifying the initial line number of a subroutine, special messages can be printed or special action taken in response to an EOF-condition. During execution of an IF END THEN statement, a branch is made to the specified line only if an EOF-condition currently exists; also, the EOF-condition is reset (removed).

In programs without an IF END THEN statement, the EOF-condition (if set) is reset (i.e., removed) when the system encounters another DATALOAD statement or initiates program execution.

3.8 REWRITING LOGICAL RECORDS

When formatting a logical record, the System 2200 automatically records special timing bits on the cassette tape before recording each data record in duplicate. When a logical record is rewritten using a DATARESAVE statement, the timing bits on the tape are checked to ensure exact positioning of the read/write head before rerecording each data record in the logical record.

To update (rewrite) a logical record, a DATARESAVE statement must be used. Under no circumstances should logical records be rewritten using a DATASAVE statement; tape errors are introduced.

Furthermore, the number of data records in the logical record written by a DATARESAVE statement must be the same as the number of data records in the logical record being updated. To avoid tape formatting errors due to miscalculation of the number of new data records required in the rewritten logical record, the argument list of the DATARESAVE statement should be identical in format to the list used to write the old logical record. To be identical in format, two argument lists must contain the same number and type of arguments, in the same order. The argument names, of course, may be different.

Provision must be made for positioning the read/write head just in front of the logical record to be rewritten before a DATARESAVE statement is encountered by the system. Usually, the following steps are necessary to rewrite a logical record:

1) Search the cassette for the beginning of the desired data file by using a DATALOAD "name" statement. After execution, the read/write head stops at the end of the data file header record, just in front of the first logical record in the data file.

2) Successively read each logical record and test the data in the record -- until the desired record is found.

3) Then, BACKSPACE 1 to position the read/write head in front of the logical record to be rewritten.

4) Use a DATARESAVE statement with a specified argument list to rewrite the record.

> Note:
>
> The DATARESAVE statement is designed to provide a technique for updating temporary data files. Wang Laboratories does not recommend use of DATARESAVE statements as a standard technique for updating permanent data files. Instead, a file backup capability should be developed. Historically, in data processing applications, development of a file backup procedure is a standard and very important consideration.

CHAPTER 4

CHAINING OR OVERLAYING PROGRAMS

4.0 INTRODUCTION

The capacity of the random access memory (RAM) in a standard System 2200 CPU is 4096 bytes (8-bit words). The standard memory is called a 4K memory. The memory can be increased in increments of 4K bytes, up to a maximum of 32K bytes. Approximately, 700 bytes of memory are used by the system for "housekeeping" purposes; the remaining capacity is available to the user.

A label located underneath each CPU identifies the RAM capacity of the unit, in multiples of 4K, by an integer appended to the model number with a dash as a separator. For example, the number 2200B-2 denotes a System 2200B with an 8K RAM. If the label is inaccessible to a user interested in knowing the RAM capacity of a particular unit, he should clear the memory and enter the word END in the Immediate Mode. Two lines appear on the CRT screen; END PROGRAM is followed by FREE SPACE = n, where n represents an integer denoting the bytes of memory available after the 698 bytes required by the system. For example, on the CRT of a system with an 8K RAM, n = 7494 since 2*4096-698=7494.

The capacity of a standard 150-foot (45.7-meter) tape cassette, even with duplicate recording of each record and with necessary gaps between records, is 76,800 bytes (76.8K bytes). Thus, the storage capacity of a 150-foot cassette is more than twice the capacity of a 32K RAM, more than nine times the capacity of an 8K RAM, and more than eighteen times the capacity of a 4K RAM.

Some programs occupy only a portion of the capacity of memory, even for systems with a 4K RAM. Hence, one cassette can store several programs, the actual number depending upon the byte-size of the individual programs.

If an application program is too large for the memory available to a user, the program logic can be divided into segments with LOAD statements inserted to "chain" the segments together. The segments are recorded on one cassette as separate parts (program files). After the first segment is loaded by the operator, the chaining logic loads and executes the sequential parts automatically.

The chaining technique can be employed even when memory capacity is not a problem. Subroutines, stored as separate program modules on one cassette, can be chained to a main program also stored on the cassette. Once the main program resides in memory, the chaining logic loads the subroutines automatically during execution of the main program.

Also, related programs, stored on cassettes mounted on different tape drives in a multidrive system, can be chained together.

Some examples presented in this section illustrate chaining and overlaying techniques using LOAD statements. The examples are brief. No attempt is made to demonstrate the requirement to chain a program since a program may be too large for a 4K RAM system but not too large for an 8K system or a 16K system.

4.1 THE LOAD STATEMENT

The chaining and overlaying techniques presented in this chapter are possible because the LOAD statement event sequence for Program Mode execution is designed with built-in "stop, clear, load and run" operations.

The LOAD statement is the only System 2200 programmable statement designed with an event sequence for Immediate Mode execution which differs from the event sequence for Program Mode execution. Therefore, the LOAD statement appears twice in the list of cassette statements in Appendix C to emphasize the differences associated with execution mode.

The general form of the LOAD statement for Program Mode operations is:

LOAD [#n, /xxx,] ["name"] [line-number-one [,line-number two]]

All the components are optional (see the BASIC language syntax summary in Appendix C).

During Program Mode execution of a LOAD statement the following operations are included:

a) STOP	Stop execution of the current program.
b) CLEAR P [line-one [,line-two]]	Clear program text between and including the specified pair of lines; if only one line is specified, clear that line and all text beyond; if no line is specified, clear all program text.
c) CLEAR N	Clear all noncommon variables from memory.
d) LOAD "name"	Read the specified (or next) program on the cassette into memory.

e) RUN [line-one]

Resume execution beginning with the specified line number or the lowest line number now in memory, if no line is specified. (Execution is interrupted by an error message if the first line of the program read from the cassette does not match line-one, if specified.)

The general form of the LOAD statement for Immediate Mode operations is:

LOAD [#n, /xxx,] ["name"]

During Immediate Mode execution, the event sequence includes only one of the five operations listed for Program Mode execution -- operation (d). Therefore, no information is cleared from memory automatically. Program text read from a cassette overwrites text in memory having matching line numbers. After the program file text is read into memory, the ready-condition colon appears on the CRT Executive Display (unless a different device has been selected for CO-class operations). System control is returned to the operator.

4.2 THE COM STATEMENT

Usually, when a program too large for memory is divided into segments to be chained together, some segments contain variables whose values are determined in previous segments. Similarly, subroutines recorded as separate program files may contain variables whose values are determined in a main program to which the subroutines are to be chained. For this reason, the System 2200 COM statement provides the capability to name and dimension the variables which are "common" to two (or more) chained segments or to a main program and its overlay programs. For example,

```
20 COM Y, B$(2,3)5
```

When executed, a COM statement (which requires at least one variable as an argument) reserves locations in a special area of memory for storage of the specified common variables. Common variables are not removed from memory when LOAD statements or RUN commands are executed (noncommon variables are removed). To remove common variables, a CLEAR or CLEAR V command must be executed or the system Master Initialized.

If numeric or alphameric arrays are included in the argument list of a COM statement, the dimension(s) of each array must be specified in the statement. Also, the maximum length of each element must be specified for alphameric arrays if the length is different from the default length (16 bytes).

The following rules restrict COM statement usage:

1) Common variables must be defined before any noncommon variables are defined (or referred to) in a program.

2) If a particular set of common variables are to be used in several sequentially run programs, COM statements need not appear in any program but the first. (Common variables remain in memory with their dimensions and maximum lengths unchanged; current values of the variables are passed from one program to the next.)

3) COM statements with argument lists containing previously defined common variables may appear in subsequent programs of the chain, but the dimensions and maximum lengths of the variables must be identical to the specifications in earlier COM statements.

4) COM statements with new common variables may appear later than the first program in a program chain.

5) The number of elements in a one-dimensional array or in either dimension of a two-dimensional array must not exceed 255; the total number of elements in a two-dimensional array must not exceed 4096. Even so, a table-overflow-condition (Code 02) can be produced if the text and variable storage requirements of a program are too large for the memory of a System 2200 configuration. A user must reduce the storage requirements when an overflow condition occurs. Memory storage requirements are discussed in Appendix B of the System 2200 A/B BASIC Programming Manual.

4.3 EXAMPLES

Use of COM and LOAD statements to chain or overlay programs is shown in Examples 4-1 and 4-2.

Example 4-1 illustrates a segmented program consisting of three parts recorded on one cassette as separate program files named PART 1, PART 2, and PART 3.

To duplicate Example 4-1, mount a cassette on the default drive 10A. Skip over files already on the cassette, if they are still of interest, by entering the proper SKIP statement. CLEAR the memory, enter Lines 10 through 100, and SAVE "PART 1". CLEAR the memory, enter Lines 415 through 485, and SAVE "PART 2". CLEAR the memory, enter Lines 610 through 680, and SAVE "PART 3". Rewind the cassette after all segments are recorded (do not rewind the cassette after recording individual parts).

To operate Example 4-1, CLEAR the memory, LOAD "PART 1", and RUN the program. When the input is requested, supply a numeric value for the variable N$. Parts 2 and 3 are loaded and executed automatically. Sample input is shown in Figure 4-1 and the corresponding output in Figure 4-2.

Example 4-1. Program Text

```
10 REM PART 1
20 COM X,Y,A(10),N$25
30 INPUT "X=",X
40 FOR J=1 TO 10
50 A(J)=X*J
60 NEXT J
70 INPUT "NAME=",N$
80 Y=LEN(N$)
90 PRINT :PRINT "LOADING SECOND SEGMENT"
100 LOAD "PART 2"

415 REM PART 2
425 PRINT :PRINT "DATA FROM FIRST SEGMENT:"
435 PRINT "X=";X,"LENGTH OF NAME=";Y
445 PRINT "N$=";N$
455 PRINT
465 PRINT "LOADING THIRD SEGMENT" :PRINT
475 REM A() PASSED TO NEXT SEGMENT
485 LOAD "PART 3"

610 REM PART 3
620 DIM B(10)
630 PRINT : PRINT "DATA FROM SECOND & THIRD SEGMENTS:" :PRINT
640 FOR J=1 TO 10 :B(J)=A(J)+J
650 PRINT "J=";J,"A(J)=";A(J),"B(J)=";B(J)
660 NEXT J
680 END
```

```
:LOAD "PART 1"
:RUN
X=? 5
NAME=? J. J. JONES
```

Figure 4-1. Operating Commands and Sample Input for Example 4-1

```
LOADING SECOND SEGMENT

DATA FROM FIRST SEGMENT:
X= 5              LENGTH OF NAME= 11
N$=J. J. JONES

LOADING THIRD SEGMENT


DATA FROM SECOND & THIRD SEGMENTS:

J= 1              A(J)= 5          B(J)= 6
J= 2              A(J)= 10         B(J)= 12
J= 3              A(J)= 15         B(J)= 18
J= 4              A(J)= 20         B(J)= 24
J= 5              A(J)= 25         B(J)= 30
J= 6              A(J)= 30         B(J)= 36
J= 7              A(J)= 35         B(J)= 42
J= 8              A(J)= 40         B(J)= 48
J= 9              A(J)= 45         B(J)= 54
J= 10             A(J)= 50         B(J)= 60
```

Figure 4-2. Output Corresponding to Sample Input for Example 4-1

The A-array and the variables X, Y, and N$ in Example 4-1 appear in more than one part of the segmented program. Therefore, the argument list of the COM statement in Line 20 of Part 1 contains the names of the common variables and specifies the dimension of the A-array and the maximum length of the alphanumeric variable N$. No other COM statement is required for these variables in subsequent parts of the program; however, no difficulty arises when a variable is specified in two different COM statements (if the specification is identical).

In Part 3 of Example 4-1, the B-array is dimensioned in a DIM, rather than a COM, statement since the B-array appears in no other segment (part). If the B-array were used in another part of the program subsequent to Part 3, Line 620 should be written as a COM statement; or, alternatively, the B-array could be included in Line 20 (and Line 620 eliminated).

Lines 100 and 485 contain the LOAD statements which implement the automatic loading and execution functions of the chained program. Lines 90 and 465 are not required; they would not appear in an unsegmented program. Lines 415, 475, and 610 are not required.

Example 4-2. Program Text

```
10 REM MAIN PROGRAM
20 COM L1,L2,L3,N
30 INPUT "LENGTH=",L1
40 INPUT "WIDTH=",L2
50 INPUT "HEIGHT=",L3
60 N=1
70 LOAD "VOLUME" 500,700
90 N=2
100 LOAD "SURFAREA" 500,700
120 N=3
130 LOAD "DIAGONAL" 500,700
140 REWIND
150 END
500 REM START OF SUBROUTINE TEXT
700 REM END OF SUBROUTINE TEXT
1000 IF N=1 THEN 90
1010 IF N=2 THEN 120
1020 IF N=3 THEN 140

    (First Subroutine)

500 REM VOLUME OF RECTANGULAR SOLID
510 V=L1*L2*L3
520 PRINT :PRINT "LENGTH=";L1,"WIDTH=";L2,"HEIGHT=";L3
530 PRINT
540 PRINT TAB(10);"VOLUME=";V
700 REM END OF SUBROUTINE

    (Second Subroutine)

500 REM SURFACE AREA OF RECTANGULAR SOLID
515 A=2*(L1*L2+L1*L3+L2*L3)
525 PRINT TAB(10);"SURFACE AREA=";A
700 REM END OF SUBROUTINE

    (Third Subroutine)

500 REM DIAGONAL OF RECTANGULAR SOLID
550 D=SQR(L1↑2+L2↑2+L3↑2)
600 PRINT TAB(10);"DIAGONAL=";D
650 PRINT
700 REM END OF SUBROUTINE
```

In the text of Example 4-2, a section of the Main Program is delineated by REM statements in Lines 500 and 700. Three different subroutines begin and end with lines numbered 500 and 700. When the main program is executed, the logic provides for the sequential overlaying and execution of each subroutine.

To duplicate Example 4-2, mount a cassette on default drive 10A. Skip files already recorded on the cassette. CLEAR the memory, enter Lines 10 through 1020 of the main program, and SAVE "MAINPROG". CLEAR the memory, enter the text of the first subroutine, and SAVE "VOLUME". CLEAR the memory, enter the text of the second subroutine, and SAVE "SURFAREA". CLEAR the memory, enter the text of the third subroutine, and SAVE "DIAGONAL". Then rewind the tape.

To operate Example 4-2, CLEAR the memory, LOAD "MAINPROG", and RUN the program. When the input is requested, supply numeric values for the length, width, and height. Sample input is shown in Figure 4-3 and the corresponding output in Figure 4-4.

```
:LOAD "MAINPROG"
:RUN
LENGTH=? 30
WIDTH=? 20
HEIGHT=? 5
```

Figure 4-3. Operating Commands and Sample Input for Example 4-2

```
LENGTH= 30      WIDTH= 20      HEIGHT= 5

          VOLUME= 3000
          SURFACE AREA= 1700
          DIAGONAL= 36.400549446
```

Figure 4-4. Output Corresponding to Sample Input for Example 4-2

CHAPTER 5

SINGLE AND MULTI-CASSETTE EXAMPLES

5.0 INTRODUCTION

Some System 2200 applications can be executed entirely in the Immediate Mode or by a one-time-only program. Such applications require no tape cassettes. Other applications require only a single tape cassette on which a program file is recorded. To run the program, the cassette is mounted on drive 10A (or perhaps another drive in a multidrive system), and the program is loaded into memory. After execution is complete, the cassette is rewound, removed from the drive, and stored until needed again.

Still other applications require several tape cassettes. Often, a program file is on one cassette, and one or more data files are on separate cassettes. If such a program is designed for a single-drive configuration, the program file cassette and the data file cassettes are mounted and removed sequentially in accordance with a set of operating instructions for the program. If the program is designed for a multidrive configuration, two or more cassettes are mounted simultaneously; the logic of the program accesses each cassette, as needed, without operator intervention.

Some examples of single and multi-tape-cassette operations are given in this chapter without detailed descriptions of the programming logic. The generation of program files is discussed in Chapter 2, and the generation of data files is discussed in Chapter 3. A study of Chapters 2 and 3, the System 2200A/B BASIC Programming Manual, and the System 2200 A/B Reference Manual should provide answers to questions about the programming logic in Examples 5-1 through 5-4.

The examples in this chapter are simplified versions of some general applications of the System 2200. If duplicated, the examples should be used for instructional purposes only, not as working models for specific applications. The types of applications considered in the examples are:

a) Recording a transaction file (see Section 5.1).

b) Merging two data files (see Section 5.2).

c) Updating a data file (see Section 5.3).

d) Using arrays to fill records (see Section 5.4).

5.1 RECORDING A TRANSACTION FILE

The program text in Example 5-1 is recorded on a program file cassette with a file name; e.g., TRANSACT. The program-file-cassette is mounted on cassette drive 10A. In a multi-drive System 2200 configuration, another cassette is mounted on drive 10B since cassette drive 10B is assigned to file number two when Line 50 in the program is executed. After the cassettes are mounted, CLEAR the memory, and LOAD "TRANSACT". Skip any files of interest on the cassette on drive 10B. Then RUN the program.

If desired, change Line 50 after the program is loaded into memory. A different cassette drive address can be assigned to file number two. Then, all the DATASAVE, DATALOAD, and REWIND statements which contain the Indirect Device Specification (#2) are valid for the new drive assigned to #2 in Line 50.

In particular, to operate the program in a single-drive configuration, proceed as follows:

1) Mount the program-file-cassette on cassette drive 10A, CLEAR the memory, and LOAD "TRANSACT".

2) Rewind and remove the program-file cassette from drive 10A.

3) Mount a data-file-cassette on drive 10A and skip any files of interest already on the cassette.

4) Reenter Line 50 as follows:

```
50 SELECT #2 10A
```

5) RUN the program.

When the program is run, a data file for one day's transactions is created. The date is entered once in the format MM/DD/YY; e.g., May 13, 1974, is written 05/13/74. The operator is instructed to supply the name of the data file by entering a DATASAVE statement with the parameter OPEN, followed by the specified name enclosed in double quotation marks. The account number (alphanumeric, up to five bytes), invoice number (numeric), and amount (or balance due) are entered sequentially for each transaction. The program requests and records information in a loop (Lines 70 through 120) until the operator responds with XXXXX for an account number. A trailer record is written, by the program logic, after the operator enters the special account number XXXXX (to signal the end of valid information) and then gives a carriage-return-only response to the requests for an invoice number and an amount. (The program includes a test for the account number XXXXX in Line 100.) Next, the operator is instructed to supply the address of an output peripheral to obtain a hardcopy of the information recorded in the data file. (The address of the Model 2201 Output Writer is usually 211; the address of the Model 2221 Line Printer is usually 215.)

Example 5-1. Program Text

```
10 REM MULTI-CASSETTE EXAMPLE
20 DIM A$5, D$8
30 REM GENERATE A TRANSACTION DATA FILE
40 INPUT "DATE IN FORMAT MM/DD/YY",D$
50 SELECT #2 10B
60 STOP "ENTER DATASAVE #2, OPEN 'DF MM/DD' WITH MM/DD=MONTH/DAY
, THEN TOUCH EXECUTE KEY. WHEN COLON APPEARS, TOUCH CONTINUE AND
 EXECUTE KEYS."
70 INPUT "ACCOUNT",A$
80 INPUT "INVOICE",I
90 INPUT "AMOUNT",M
100 IF A$="XXXXX" THEN 130
110 DATA SAVE #2,A$,I,D$,M
120 GOTO 70
130 DATA SAVE #2, END
140 REWIND #2
150 STOP "FOR HARDCOPY, SELECT PRINT 215 (OR 211). EXECUTE. WHEN
 COLON APPEARS, TOUCH CONTINUE AND EXECUTE KEYS."
155 STOP "ENTER DATALOAD #2, 'DF MM/DD' WITH MM/DD=MONTH/DAY, TH
EN TOUCH EXECUTE KEY. WHEN COLON APPEARS, TOUCH CONTINUE AND EXE
CUTE KEYS."
160 REM PRINT LOGICAL RECORD NUMBER AND DATA
165 R=0
170 PRINT "RECORD   ACCOUNT   INVOICE      DATE      AMOUNT"
180 PRINT HEX(0A)
190 DATA LOAD #2,A$,I,D$,M
200 IF END THEN 250
210 R=R+1
220 PRINTUSING 230,R,A$,I,D$,M
230 %######     #####  ########  ########  ###,###.##
240 GOTO 190
250 REWIND #2
260 SELECT PRINT 005
270 STOP "END OF PROGRAM"
```

5.2 MERGING TWO DATA FILES

The program text in Example 5-2 is recorded on a program file cassette with a file name (e.g., MERGE). After the program is loaded into memory from drive 10A, remove the program tape from the drive.

Two input data cassettes and one output data cassette are required. Because the program logic is designed for a simplified application, the two input cassettes should contain data recorded in ascending order by account and invoice numbers (as shown in Table 5-1). The input cassettes (containing sample files to be merged) can be generated prior to testing the MERGE program by running program TRANSACT (the program in Example 5-1) twice with a different cassette on drive 10B each time the program is run.

Table 5-1. Sample Input for Example 5-2

Cassette	Account	Invoice	Date	Amount
Data 1	AAAAA	101	05/25/74	2.00
(On one cassette)	AAAAA	107	05/25/74	20.40
	AAAAA	114	05/25/74	56.00
	BBBBB	102	05/25/74	23.98
	BBBBB	104	05/25/74	123.55
	BBBBB	106	05/25/74	99.80
	BBBBB	112	05/25/74	13.43
	BBBBB	113	05/25/74	5.69
	CCCCC	110	05/25/74	34.72
	CCCCC	111	05/25/74	12.48
Data 2	AAAAA	99	05/24/74	23.95
(On different	AAAAA	108	05/24/74	72.90
cassette)	BBBBB	105	05/24/74	6.00
	CCCCC	115	05/24/74	20.00

Now, with program MERGE in memory, mount one data file input cassette on drive 10A and the other on drive 10B. Mount a scratch cassette on drive 10C. The data files on drives 10A and 10B are merged when program MERGE is run. The merged data file is written on the scratch cassette.

Example 5-2. Program Text

```
10 REM TAPE MERGE PROGRAM
20 DIM A1$5,A2$5,D1$8,D2$8,F$1
30 SELECT #1 10A,#2 10B,#3 10C
40 F$ = HEX(FF)
50 REM READ TAPE, UNIT 1
60 GOSUB 310
70 REM READ TAPE, UNIT 2
80 GOSUB 360
90 REM COMPARE ACCOUNT NOS.
100 IF A1$ < A2$ THEN 200   :REM TAPE 1 LOW
110 IF A2$ < A1$ THEN 160   :REM TAPE 2 LOW
120 IF A1$ = F$ THEN 240  :REM IF BOTH = HEX(FF), FINISHED
130 REM SAME ACCOUNT - COMPARE INVOICE NOS.
140 IF I1 <= I2 THEN 200   :REM TAPE 1 LOW OR EQUAL
150 REM TAPE 2 LOW - WRITE RECORD
160 DATA SAVE #3,A2$,I2,D2$,M2
170 GOSUB 360  :REM READ ANOTHER RECORD FROM UNIT 2
180 GOTO 100
190 REM TAPE 1 LOW - WRITE RECORD
200 DATA SAVE #3,A1$,I1,D1$,M1
210 GOSUB 310   :REM READ ANOTHER RECORD FROM UNIT 1
```

(continued on next page)

```
220 GOTO 100
230 REM FINISHED - WRITE "END" RECORD
240 A1$ = "END"
250 DATA SAVE #3,A1$,I1,D1$,M1
260 REWIND #1
270 REWIND #2
280 REWIND #3
290 STOP "END OF PROGRAM"
300 REM READ TAPE, UNIT 1
310 DATA LOAD #1,A1$,I1,D1$,M1
320 IF A1$ <> "END" THEN 340
330 A1$ = F$      :REM HEX(FF)
340 RETURN
350 REM READ TAPE, UNIT 2
360 DATA LOAD #2,A2$,I2,D2$,M2
370 IF A2$ <> "END" THEN 390
380 A2$ = F$    :REM HEX(FF)
390 RETURN
```

5.3 UPDATING A DATA FILE

The program text in Example 5-3 demonstrates the technique of updating a data file using SKIP, BACKSPACE, and DATARESAVE statements. Such a technique should be used to update a temporary file only; the procedure should not be used repeatedly on a particular data file. Furthermore, a duplicate or backup file should be generated before attempting to update a file since valuable data may be destroyed if errors are made during the updating procedure.

The text is written for a data file whose records contain the following information: the account number (A$), invoice number (I), date (D$), and amount (M). Assuming the sequential order of the records in the data file is known, an invoice is located by its position in the file and a payment is applied to the invoice (that is, the "amount" field is changed).

Example 5-3. Program Text

```
10 REM UPDATE
20 REM EXAMPLE OF BACKSPACE, SKIP, DATARESAVE
30 DIM A$5,D$8
40 SELECT PRINT 215
50 PRINT "RCD   ACCOUNT   INVOICE    DATE       AMOUNT   PAID
 BALANCE"
60 SELECT PRINT 005
70 SELECT #2 108
80 REWIND #2
90 R = 1   :REM TAPE IS POSITIONED TO READ RECORD NO. 1
100 T = 0   :REM TOTAL OF PAYMENTS APPLIED
110 INPUT "RECORD NO.",X
120 IF X < 1 THEN 110    :REM NO GOOD
130 IF X > 255 THEN 500  :REM SIGNAL TO END
140 IF X = R THEN 240   :REM ALREADY POSITIONED CORRECTLY
150 REM NOTE THAT BACKSPACE 0 OR SKIP 0 IS INVALID
160 REM SKIP OR BACKSPACE MORE THAN 255 IS ALSO INVALID
170 IF X < R THEN 220  :REM GO BACK TO FIND RECORD
180 REM X > R, GO FORWARD TO FIND RECORD
190 SKIP #2,X-R
200 GOTO 240
210 REM X < R, GO BACK TO FIND RECORD
220 BACKSPACE #2,R-X
230 REM TAPE IS NOW POSITIONED TO READ RECORD
240 R = X   :REM KEEP TRACK OF TAPE POSITION
250 REM READ RECORD
260 DATA LOAD #2,A$,I,D$,M
270 REM DISPLAY CONTENTS
280 PRINTUSING 290,R,A$,I,D$,M
290 %###   #####    ########  ######## ###,###.##
300 REM ENTER PAYMENT
310 INPUT "PAYMENT",P
320 REM CALCULATE NEW BALANCE
330 B = M - P
340 REM ADD TO TOTAL
350 T = T + P
360 REM PRINT TRANSACTION
370 SELECT PRINT 215
380 PRINTUSING 290,R,A$,I,D$,M;
390 PRINTUSING 400,P;B
400 %##,###.##
410 SELECT PRINT 005
420 REM UPDATE BALANCE
430 M = B
440 REM REWRITE RECORD
450 BACKSPACE #2,1
460 DATA RESAVE #2,A$,I,D$,M
470 R = R + 1   :REM POSITIONED AT NEXT RECORD
480 GOTO 110   :REM DO ANOTHER
490 REM END **********************
500 SELECT PRINT 215
510 PRINT "TOTAL PAYMENTS";TAB(44);
520 PRINTUSING 400,T
530 SELECT PRINT 005
540 REWIND #2
550 STOP "END OF PROGRAM"
```

5.4 USING ARRAYS TO FILL RECORDS

Each time the System 2200 executes a DATASAVE statement with an argument list, a logical record is formatted. The logical record consists of as many 256-byte physical records as required to store the values for all the arguments in the list. Each physical (data) record contains three control bytes and up to 253 bytes of valid data. If the required number of storage bytes for a DATASAVE argument list is less than 253, the system formats a logical record consisting of only one 256-byte physical record.

Of interest here are argument lists which require only a fraction of the 253-bytes of available storage space in a data record; for example, one-half (126 bytes), one-third (84 bytes), one-fourth (63 bytes), et cetera. Table 5-2 shows the cassette storage requirement imposed by the DATASAVE argument list in Example 5-1.

Table 5-2. Storage Requirements for the DATASAVE Argument List in Example 5-1, 5-2, and 5-3

Argument	Memory Storage*	Cassette Storage
A$	5 bytes	6 bytes
I	8 bytes	9 bytes
D$	8 bytes	9 bytes
M	8 bytes	9 bytes
		(Total 33 bytes)

*Does not include the number of bytes in memory for the argument name which is not duplicated on tape.

Each time the DATASAVE A$,I,D$,M statement in Example 5-1 (Line 110) is executed, the system writes one logical record consisting of one data record. Only 33 bytes are written in a data record which can accommodate a maximum of 253 bytes of data. By a clever use of arrays, the automatic formatting of the system can be bypassed and up to seven (253/33=7.6) times the amount of data can be recorded on the same tape interval as shown in Example 5-4.

Example 5-4. Program Text

```
10 REM ARRAY
20 REM USING ARRAYS TO FILL DATA RECORDS
30 DIM A$(7)5,I(7),D$(7)8,M(7),D$8
40 SELECT #2 10B
50 INPUT "DATE",D$
60 FOR X = 1 TO 7
70     D$(X) = D$
80     NEXT X
90 K = 1
100 REM DATA INPUT LOOP
110 INPUT "ACCOUNT",A$(K)
120 INPUT "INVOICE",I(K)
130 INPUT "AMOUNT",M(K)
140 IF A$(K) = "END" THEN 180
150 GOSUB 230
160 GOTO 110
170 REM FINISHED
180 K = 7   :REM FORCE WRITING OF BLOCK
190 GOSUB 230
200 REWIND #2
210 STOP "END OF PROGRAM"
220 REM WRITE TAPE SUBROUTINE
230 K = K + 1   :REM INCREMENT SUBSCRIPT OF ARRAY
240 IF K < 8 THEN 280
250 REM WRITE TAPE ONLY WHEN ARRAY IS FULL
260 DATA SAVE #2,A$(),I(),D$(),M()
270 K = 1    :REM START ARRAY AGAIN
280 RETURN
```

CHAPTER 6

OTHER CASSETTE OPERATIONS

6.0 INTRODUCTION

The cassette operations described in Chapters 1 through 5 are programmed using statements available in the System 2200A, 2200B, 2200C, and 2200S. In this chapter, two cassette statements available only in the System 2200B and 2200C are discussed. The statements are DATASAVE BT and DATALOAD BT.

6.1 DATASAVE BT AND DATALOAD BT STATEMENTS

The general forms of the DATALOAD BT and DATASAVE BT statements are as follows:

DATALOAD BT [(N=expression)] [#n, /xxx,] alpha-array-designator

DATASAVE BT [R] [([N=expression],[H])] [#n, /xxx,] alpha-array-designator

where the parameters, defined as follows, are optional if in brackets.

R = resave.

N = 100 or 256, the block size to read/record (default size 256).

H = write timing marks for a header record block.

n = file number (1,2,3,4,5, or 6) assigned to the cassette drive to be accessed.

xxx = device address of the cassette drive to be accessed (address must be preceded by a slash).

alpha-array-designator = an array name plus left and right parentheses; e.g., A$().

One, and only one, alphanumeric array designator must be specified as an argument for each DATALOAD BT or DATASAVE BT statement. All other information is optional. However, before either a DATALOAD BT or DATASAVE BT statement is encountered by the system, the array dimension must be specified in a separate DIM statement.

If the dimension specified for an array is too small for the block size, an error message (Code 60) is printed and corrective action is required. On the other hand, the dimension specified for an array may be larger than the block size; but, if the dimension of the array is too large for the space currently available in memory, a table-overflow-condition occurs. An error message (Code 02) is printed and corrective action is required.

The parameter N = 256 need not be specified when reading, recording, or copying System 2200 cassettes since the default block size is 256 for DATALOAD BT and DATASAVE BT statements. However, to read or copy tape cassettes from the Wang Model 1220 Dual Cassette Typewriter, specify N = 100. The Model 1220 Dual Cassette Typewriter automatically formats information on tape in 100-byte physical records.

Upon execution, a DATASAVE BT statement records a block of data (100 or 256 bytes) on a tape cassette with no control bytes. If the parameter H appears in the statement to signify a header record is being written, a timing mark is recorded on the tape just prior to the data block. If the parameter R appears in the statement to signify a data block is to be rewritten, the tape is backspaced one block automatically before the information is recorded.

If the array length (the product of the number of elements and the maximum length of each element) is greater than the specified block size (100 or 256 bytes), only the first 100 or 256 bytes of the array are recorded. If the array length is less than the specified block size, the entire array is recorded and the remainder of the block contains unpredictable characters.

Upon execution, a DATALOAD BT statement reads the next block (100 or 256 bytes) on tape and stores the information in the specified alphanumeric array. If the parameter N is not specified, the block size is assumed to be 256.

A DATALOAD BT statement reads program files as well as data files. Therefore, special programs can be written to copy the contents of one tape cassette onto another cassette, to read and convert information stored on tape, and to pack information stored on tape. Such programs use many bit and byte manipulation statements available only in the System 2200B and 2200C.

6.2 COPYING TAPE CASSETTES

A sample program to copy the contents of one tape cassette onto another cassette is given in Example 6-1. The program uses only one DATALOAD BT statement and one DATASAVE BT statement in an endless loop.

A 256-byte block of information is read from the cassette mounted on drive 10A each time Line 20 in the program is executed. The information is stored in the four-element X$-array whose elements have a maximum length of 64 characters.

Each time the DATASAVE BT statement in Line 30 is executed, the information stored in the X$-array is recorded in a 256-byte record on the cassette mounted on drive 10B.

Audible clicks indicate the completion of the reading operation for each block. When the end of tape is reached, program execution stops and an error message (Code 49) is displayed. By touching the Rewind button on the housing of each drive, the cassettes are rewound and can be removed.

If the audible clicks cease before the end of tape is reached, an operator can interrupt the program at any time by touching the Reset button on the keyboard. Then, by touching the Rewind button on the housing of each drive, the cassettes are rewound and can be removed.

In Example 6-2, the contents of one cassette mounted on drive 10A are copied on two cassettes mounted on drives 10B and 10C.

Example 6-1. Program Text

```
10 DIM X$(4)64
20 DATA LOAD BT/10A,X$()
30 DATA SAVE BT/10B,X$()
40 GOTO 20
```

Example 6-2. Program Text

```
10 DIM X$(4)64
20 DATA LOAD BT/10A,X$()
30 DATA SAVE BT/10B,X$()
40 DATA SAVE BT/10C,X$()
50 GOTO 20
```

Note:

1. The simplified programs in Examples 6-1 and 6-2 make no provision for recording special timing marks before header records. Therefore, BACKSPACE mF statements cannot be used when working with a cassette copied from another by running either of these programs.

2. Only a cassette containing a single unformatted data file (that is, logical records with no header and trailer records to identify the boundaries of the file) can be copied satisfactorily with the programs in Examples 6-1 and 6-2.

NOTES:

APPENDIX A - DEVICE ADDRESSES FOR SYSTEM 2200 PERIPHERALS

I/O DEVICE CATEGORIES	DEVICE ADDRESSES*
KEYBOARDS (2215, 2222)	001, 002, 003, 004
CRT (2216)	005, 006, 007, 008
TAPE CASSETTE DRIVES (2217, 2218)	10A, 10B, 10C, 10D, 10E, 10F
LINE PRINTERS (2221, 2231) (2261)	215, 216
OUTPUT WRITER (2201)	211, 212
THERMAL PRINTER (2241)	215, 216 (Revised 3/7/74)
PLOTTERS (2202, 2212, 2232)	413, 414
DISK DRIVES (2230-1, -2, -3) (2240-1, -2) (2242, 2243)	310, 320, 330
MARK SENSE (MANUAL) CARD READER (2214)	517
HOPPER FEED CARD READERS (2234, 2244)	629, (029)
PUNCHED TAPE READER (2203)	618
TELETYPE (2207)	019, 01A, 01B INPUT 01D, 01E, 01F OUTPUT
TELETYPE TAPE UNITS	41D, 41E, 41F
TELECOMMUNICATIONS (2227)	219, 21A, 21B INPUT 21D, 21E, 21F OUTPUT
PARALLEL I/O INTERFACE (2250)	23A, 23C, 23E INPUT 23B, 23D, 23F OUTPUT (Revised 3/7/74)
BCD INPUT INTERFACE (2252)	25A, 25B, 25C, 25D, 25E, 25F

*In some cases, more than one device address is listed for each device category. Unless otherwise noted, each peripheral device is assigned a unique address; device addresses are assigned sequentially. Therefore, if a System 2200 has only one device of a particular category (such as a tape drive), it is set up with the first device address listed (10A in the case of the tape drive). If it has two drives, they are set up with device addresses 10A and 10B. Each device address is printed on the interface card which controls that device.

APPENDIX B - DEVICE SELECTION FOR SYSTEM 2200 I/O OPERATIONS

Five address codes are designated as Primary Device Addresses in the System 2200 (see Table B-1). Devices with these default addresses are "selected" automatically whenever the system is Master Initialized (i.e., power is turned off and then on again).

TABLE B-1. PRIMARY DEVICE ADDRESSES FOR THE SYSTEM 2200

DEFAULT ADDRESS	I/O DEVICE CATEGORY	MODEL NUMBER	SELECT Statement I/O Class Parameter
001	Keyboards	2215 or 2222	CI (console input) INPUT
005	CRT	2216	CO (console output) PRINT LIST
10A	Tape drives	2217 or 2218	TAPE
310	Disks	2230 or 2240	DISK
413	Plotters	2202, 2212, or 2232	PLOT

Two or more model numbers in Table B-1 correspond to one default address in several cases. These models should be considered as "candidates" for the primary device address. Since device addresses within a configuration are unique, only one keyboard or one tape drive can be the primary device for a particular device category.

Input/output operations for the System 2200 are grouped in eight classes designated by the I/O "class parameters": CI, INPUT, CO, PRINT, LIST, TAPE, DISK, and PLOT. Chart B-1 identifies the I/O operations in each class, and Table B-1 gives the default device address for these classes of I/O operations.

To change the device address for particular I/O operations, use the SELECT statement. This statement has other uses; but, in every case, it requires a "select parameter" (see the Reference Manual). When selecting I/O peripheral devices, keep in mind that the select-statement-parameter is two-fold. It consists of an I/O class parameter and a three-digit device address.

For example, the statement

```
SELECT PRINT 215
```

instructs the system to access the Line Printer with address 215 for all subsequent output from Program Mode PRINT and PRINTUSING statements.

To reselect the CRT for this output, use the statement

```
SELECT PRINT 005
```

or Master Initialize the system if the memory can be cleared at this point.

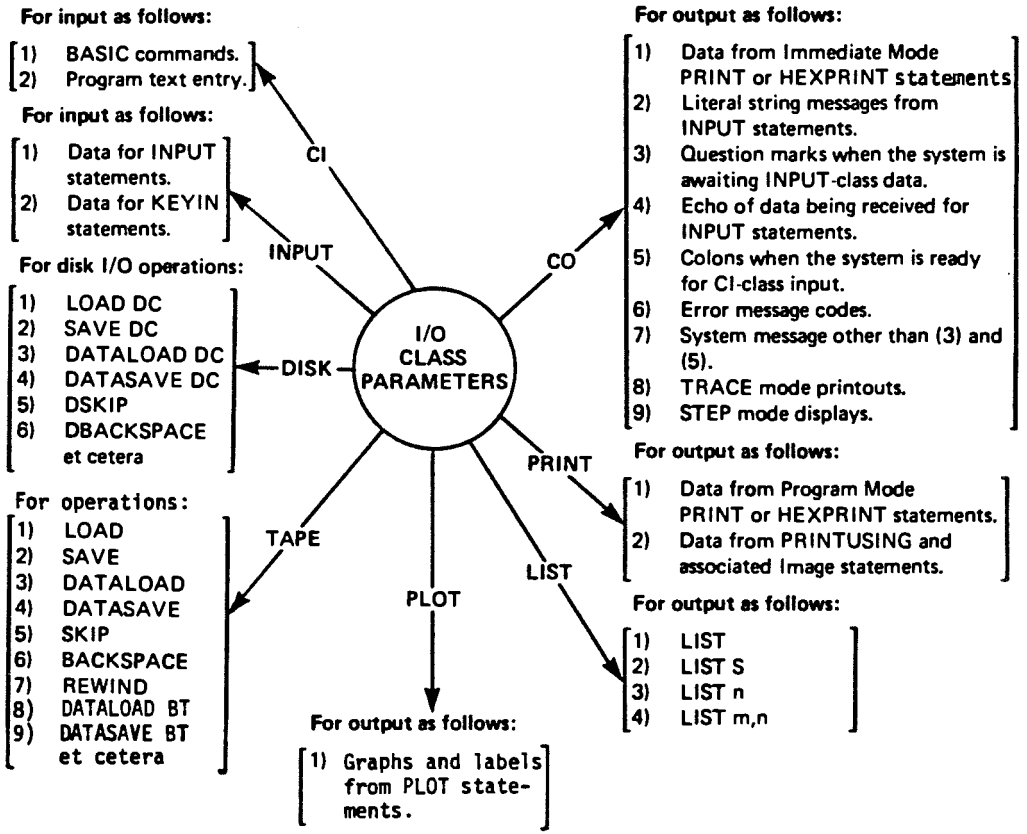
Two or more devices performing different functions can be selected in one statement by using commas as device separators. For example,

```
SELECT LIST 215, PRINT 211, TAPE 10C
```

The SELECT verb "assigns" the specified device address to the specified I/O class parameter. Using a

select-statement is analogous to setting an I/O-class rotor switch which includes the device-address-options for that class. All subsequent I/O operations in the I/O class are "switched" to the designated device until the system encounters another select-statement for that I/O class.

Chart B-1 I/O Operations and Class Parameters for the System 2200



NOTE:

This chart identifies the input/output operations in the eight System 2200 I/O classes. A class parameter (CI, INPUT, CO, PRINT, LIST, TAPE, DISK, or PLOT) with a device address is used in SELECT statements to change or reselect a device for I/O operations. The default address for CI and INPUT operations is 001 (the keyboard). The default address for CO, PRINT, and LIST operations is 005 (the CRT).

APPENDIX C

BASIC LANGUAGE STATEMENTS FOR CASSETTE OPERATIONS

The System 2200 BASIC language statements fall into two major categories: (1) programmable statements, and (2) commands (nonprogrammable statements). The programmable statements, also called "statements", fall into two categories: (a) statements executable in both the Program Mode and the Immediate Mode, and (b) statements executable in the Program Mode only.

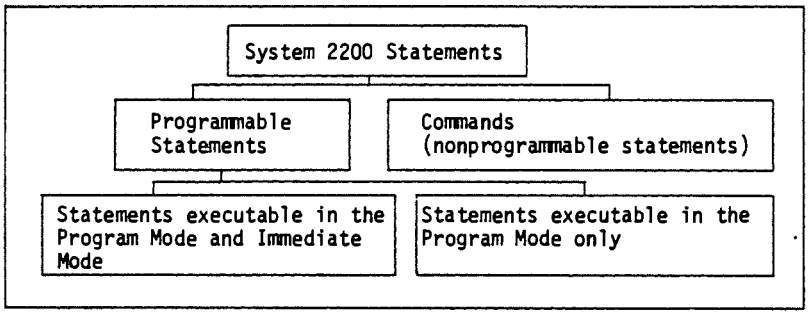


Figure C-1. Categories for System 2200 Statements

The Central Processing Unit for the System 2200 is available in four models, the 2200A, 2200B, 2200C, and 2200S. The CPU models utilize non-identical sets of BASIC language statements. However, every statement in the System 2200A is included in the System 2200B, the System 2200C, and the System 2200S.

The statements used to control tape cassette drive operations are summarized in Table C-1 with the operations listed in alphabetical order for easy reference.

As shown in Table C-1, only one tape cassette operation is controlled by a "command" -- the SAVE command, used to record program files. All other cassette operations are controlled by "statements" executable in the Program Mode and the Immediate Mode. However, one operation (LOAD) appears in the table twice -- once for the Immediate Mode and again for the Program Mode.

Usually, system action during Immediate Mode execution of a statement is functionally the same as the action during Program Mode execution. The LOAD statement is an exception; for LOAD statements only, system action depends upon the execution mode.

Coded notes, referred to by number in Table C-1, contain functional descriptions of the tape cassette drive statements. The notes are found after Table C-2.

The BASIC language syntax and notation for the cassette statements in Table C-1 is similar to the syntax and notation used for all the statements in the System 2200A/B Reference Manual and the System 2200S Reference Manual. The syntax is summarized in Paragraphs 1 through 10 below. The notation is defined in Table C-2.

1. Upper case letters (A through Z) must be written in an actual statement exactly as shown in the general form.

2. Lower case letters or words represent items for which specific information is to be substituted.

3. Hyphens joining lower case words (or words and numbers) signify single items.

4. Vertically stacked items represent alternatives, only one of which is to be selected.

5. When stacked items are enclosed in braces, { } one item must be specified. The braces are not included in an actual statement.

6. When stacked items or single items are enclosed in brackets, [], the items are optional and may be omitted. The brackets are not included in an actual statement.

7. The following characters must be written as shown in the general form, unless otherwise indicated by a note:

comma	,
equal sign	=
parentheses	( )
pound-sign	#
slash	/

8. When an ellipsis, ..., follows an item, the item may be repeated many times successively in an actual statement.

9. Blanks, inserted for readability in the general form, are not required. The System 2200 ignores blanks in an actual statement unless the blanks are embedded in double quotation marks.

10. The sequential order of the components in the general form must be preserved when writing an actual statement.

Table C-1. Cassette Statements

OPERATION	COMMAND or STATEMENT	2200A 2200B 2200C 2200S	2200B 2200C	GENERAL FORM	REMARKS
BACKSPACE	Statement	X		BACKSPACE [#n, /xxx,] {BEG / m / mF}	See notes 1,2
DATALOAD	Statement	X		DATALOAD [#n, /xxx,] {"name" / argument list}	See notes 1,3
DATALOAD BT	Statement		X	DATALOAD BT [(N=s)] [#n, /xxx,] alpha-array-designator	See notes 1,4
DATARESAVE	Statement	X		DATARESAVE [#n, /xxx,] {OPEN "name" / argument list}	See notes 1,5
DATASAVE	Statement	X		DATASAVE [#n, /xxx,] {OPEN "name" / END / argument list}	See notes 1,6
DATASAVE BT	Statement		X	DATASAVE BT [R] [([N=s], [H])] [#n, /xxx,] alpha-array-designator	See notes 1,7
LOAD	Immediate Mode	X		LOAD [#n, /xxx,] ["name"]	See notes 1,8
LOAD	Program Mode	X		LOAD [#n, /xxx,] ["name"] [line-number-one[,line-number-two]]	See notes 1,9
REWIND	Statement	X		REWIND [#n / /xxx]	See note 1
SAVE	Command	X		SAVE [#n, /xxx,] [P] ["name"] [line-number-one[,line-number-two]]	See notes 1,10
SKIP	Statement	X		SKIP [#n, /xxx,] {END / m / mF}	See notes 1,2

Table C-2. Notation for Cassette Statements

Notation	Definition or Function
n	Represents an indirect address specification -- the file number (1,2,3,4,5, or 6) currently assigned the address of the cassette drive to be accessed for the operation.
xxx	Represents an absolute address specification -- the device address (10A, 10B, 10C, 10D, 10E, or 10F) of the cassette drive to be accessed for the operation.
alpha-array-designator	An alphanumeric array name followed by left and right parentheses; e.g., A$(), B5$(), denoting the set of array elements.
argument list	One or more arguments separated by commas. Items acceptable as arguments include the following: alphanumeric or numeric variables, specific alphanumeric or numeric array elements, alphanumeric or numeric array designators (multi-element arguments). In addition, literal strings or mathematical expressions qualify as arguments in DATASAVE and DATARESAVE statements.
BEG	For data files only -- specifies backspacing, within the current data file, to the beginning of the first logical record (the tape is positioned at the end of the header record).
END	For data files only --(a) If executing a SKIP statement, END specifies skipping to the end of the current data file (the tape is positioned at the beginning of the trailer record). (b) If executing a DATASAVE statement, END specifies a trailer record is to be written.
H	Specifies a header record is to be written with a special timing mark recorded in front of the physical record.
line-number-one	Represents the first line of a specified portion of the program text currently in memory. If not specified, the lowest numbered line of the current program is implied.

(continued on next page)

line-number-two	Represents the last line of a specified portion of the program text currently in memory. If not specified, the highest numbered line of the current program is implied.
m	For data files only -- represents an integer (≥1) or a mathematical expression specifying the number of logical records to backspace or skip (the tape is positioned at the beginning of a logical record). During execution, an expression is evaluated and truncated to an integer (which must be ≥1).
mF	For program or data files -- specifies the number of files to pass over, until the tape is positioned as follows: a) at the beginning of the mth header record, if backspacing, or, b) at the end of the mth trailer record, if skipping. where m represents an integer (≥1) or a mathematical expression. An expression is evaluated and truncated to an integer (which must be ≥1).
N	N = s (where s can be 100 or 256) specifies the physical record block size (number of bytes). If not specified, N = 256 is implied.
name	One to eight bytes, consisting of any combination of alphabetic, numeric, or special characters (except double quotation marks).
OPEN	Specifies a data file header record is to be written. The file name to be recorded in the record must be specified in double quotation marks, following the parameter OPEN.
P	Specifies a protected program file is to be recorded (a protected file can be read into memory at a future time but cannot be listed nor rerecorded).
R	Specifies a record is to be rewritten, after the tape is backspaced one block automatically.
s	Specifies the physical record block-size in bytes (only 100 or 256 can be substituted for s).

(End of Table C-2)

Coded Notes for Tape Cassette Drive Statements in Table C-1

Note 1. If neither an Indirect nor an Absolute Address Specification is included in a cassette statement, the device last selected for TAPE-class operations (or the default device 10A) is accessed.

Note 2. If a cassette tape is positioned (relative to the read/write head) within a data file, the following special operations may be useful:

BACKSPACE BEG	To reposition the tape at the end of the header record.
BACKSPACE 1F	To reposition the tape in front of the header record.
SKIP END	To reposition the tape in front of the trailer record.
SKIP 1F	To reposition the tape at the end of the trailer record.

Note 3. a) If a name is specified in a DATALOAD statement, the cassette is searched forward only for a data file header record with a matching name; when found, the tape is positioned at the end of the header record (in front of the first logical record).

b) If an argument list is specified, the next logical record is read. The data values are assigned sequentially to the arguments (for arrays, values are stored in the elements row-by-row). If the logical record contains more values than required by the argument list, the additional values are ignored while the tape advances to the end of the logical record. If the logical record contains insufficient values for the argument list, another logical record is read automatically. If a trailer record is encountered, an end-of-file condition is set; arguments not yet updated retain their former values; the tape remains positioned in front of the trailer record; and statement execution is completed. Insert an IF END THEN statement after a DATALOAD statement, if a transfer to alternative logic is desired in an applications program.

Note 4. A DATALOAD BT statement reads the next block (100 or 256 bytes) of data or program text and stores the information in the specified alphanumeric array. Array dimension(s) must be specified in a separate DIM statement. If the array length (the product of the number of elements and the number of bytes per element) is too small for the block size, execution is terminated by an error message. If the array length is too large for the block size, no error message is issued unless a table overflow condition occurs or one of the following rules is violated: (1) the number of elements in one (or either) dimension cannot exceed 255, (2) the total number of elements cannot exceed 4096, and (3) the number of bytes per element cannot exceed 64.

(Notes continued on next page.)

Note 5. Repeated use of DATARESAVE statements to update a permanent file is not recommended (a file backup capability should be developed). Never use a DATASAVE statement to rewrite a logical record. When using a DATARESAVE statement, be sure the tape is positioned just in front of the logical record to be rewritten. Furthermore, be sure the number of physical records in the logical record to be written by the DATARESAVE statement is the same as the number of physical records in the logical record being updated -- by using a DATARESAVE argument list identical to the argument list of the original DATASAVE statement, as a standard practice.

Note 6. a) If the parameter OPEN is specified in a DATASAVE statement, a data file header record containing the specified name is written. However, if only one data file is being written on a cassette, a header record is not needed.

b) If the parameter END is specified, a data file trailer record is written.

c) If an argument list is specified, one logical record (containing as many physical records as necessary) is written on tape. Argument values are recorded sequentially, as listed (two-dimensional array values are recorded row by row).

Note 7. a) A comma must separate the N and H parameters in a DATASAVE BT statement if both are specified; otherwise, the comma is omitted.

b) If the R parameter is specified (to rewrite a block), the tape is backspaced one block automatically before the block is written.

c) If the H parameter is specified (to record a header record), a special timing mark is recorded on tape ahead of the block. The timing mark is used by the system when backspacing files.

d) If N is not specified, the block size is assumed to be 256 bytes. During execution, the block of data (100 or 256 bytes) is recorded on tape without control information. If the specified array is larger than the specified block size, only the first 100 (or 256) bytes of the array are recorded. If the array is smaller than the specified block size, the data in the array is recorded and the block is filled with unpredictable characters.

(Notes continued on next page.)

Note 8. During Immediate Mode execution of a LOAD statement, no information is cleared from memory automatically. If a program file name is specified, the cassette is searched (forward only) and the name in each program file header record is checked. If the names match, the program file is read into memory. If no name is specified, the next program file is loaded into memory. Program text read from tape is appended to (or overwrites) any text currently in memory. Then, system control is returned to the operator.

Note 9. During Program Mode execution of a LOAD statement, the event sequence includes the following operations:

STOP	Stop execution of the current program.
CLEAR P [line-no.-one [,line-no.-two]]	Clear text between and including specified line numbers; if only one line is specified, clear that line and all text beyond; if no line is specified, clear all program text.
CLEAR N	Clear noncommon variables from memory.
LOAD ["name"]	Read the specified (or next) program on the cassette into memory.
RUN [line-number-one]	Resume execution, beginning with line-no.-one if specified, or the lowest line number. The first line of the program read from tape must match line-number-one, if specified; otherwise execution is interrupted by an error message.)

Restrictions: In multi-statement lines of text, a LOAD statement must be the last statement in the line. A LOAD statement must not be part of a FOR/NEXT loop.

Note 10. The SAVE command records program text currently in memory on tape (automatically formatting a program file). The program file consists of a header record, as many program records as needed, and a trailer record. If a name is specified, it is recorded in the header record. If line-number-one is specified, only the portion of the text beginning with the specified line and ending with the final line (or the specified line-number-two) is recorded; otherwise, all the current text is recorded. If the parameter P is specified, a "protected program file" is recorded; such a file can be loaded into memory but cannot be listed or re-recorded.

(End of Notes for Table C-1)

APPENDIX D

SPECIFICATIONS

	Model 2217	Model 2218
Physical Size		
Height	7-1/8 in. (18.1 cm)	7-1/8 in. (18.1 cm)
Width	7-1/2 in. (19.1 cm)	14 in. (35.6 cm)
Depth	16-1/2 in. (41.9 cm)	16-1/2 in. (41.9 cm)
Site Size (Door Open)	7-1/8" x 7-1/2" x 17-1/2" (18.1 cm x 19.1 cm x 44.5 cm)	7-1/8" x 14" x 17-1/2" (18.1 cm x 35.6 cm x 44.5 cm)
Net Weight	21 lb (9.5 kg)	38 lb (17.2 kg)
Power Cable	8 ft (2.4m)	
Cable to CPU	8 ft (2.4m)	
Power Requirements	115 or 230 VAC ± 10% 50 to 60 Hz ± 1/2 cycle	
Operating Environment	50°F to 90°F (10°C to 32°C) 20% to 80% relative humidity	
Stop/Start Time	.09/.05 sec	
Recording and Search Speed	7.5 in/sec (19 cm/sec)	
Rewind Speed	7.5 ft/sec (2.3m/sec)	
Transfer Rate	326 bytes/sec (including inter-record gaps and dual recording)	
Cassette Capacity	Approximately 300 automatically-formatted, dually-recorded records per 150-ft cassette (approximately 76,800 bytes)	
Supplies	75-ft (22.8m) Cassette (Part Number 174-1250) 150-ft (45.7m) Cassette (Part Number 174-1251) Cleaning Pads (100) (Part Number 660-0130)	

APPENDIX E

CLEANING A CASSETTE READ/WRITE HEAD

Tape cassettes and cassette read/write heads for the System 2200 should be kept as dust and dirt free as possible. The magnetic head in a tape cassette drive unit should be cleaned periodically, approximately every three weeks under normal conditions. More frequent cleaning is required if tape cassettes become contaminated with dust and dirt (or if possible electrostatic attraction of dust and dirt occurs because the operating environment humidity is below 20%).

Cleaning pads (Part number 660-0130) can be obtained from your Wang Serviceman. Head cleaning cassettes should not be used; they are too abrasive for Wang equipment.

The cassette read/write head is located in the upper central portion of a cassette drive unit. The head is barely visible when the cassette drive door is open (see Figure E-1).

To lower the cassette read/write head to the cleaning position, proceed as follows:

1) Turn ON the System 2200 and the tape cassette drive unit(s) to be cleaned.

2) Open the door of the unit to be cleaned by pressing the Door Release button.

3) Enter LOAD or LOAD/xxx (where xxx is the device address of the unit to be cleaned), and then touch the EXECUTE key.

4) Disregard the error message (Code 49) which appears on the CRT display. The cassette read/write head is lowered into the position shown in Figure E-2.

Tear open a foil packet containing a cleaning pad, and rub the magnetic head gently for a few moments (see Figure E-3). After cleaning the cassette read/write head, dispose of the cleaning pad in the foil packet. Exercise care to prevent the cleaning pad from touching any painted, shellacked, or plastic surface.

To return the cassette read/write head to its normal position, press the Rewind button. The cassette drive rewinding action restores the head to its normal position shown in Figure E-4.

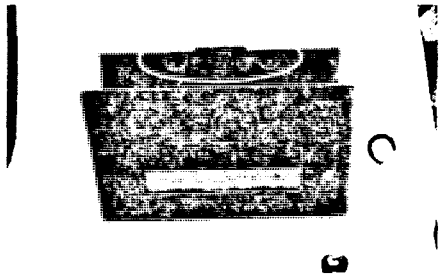


Figure E-1. Cassette Read/Write Head Assembly

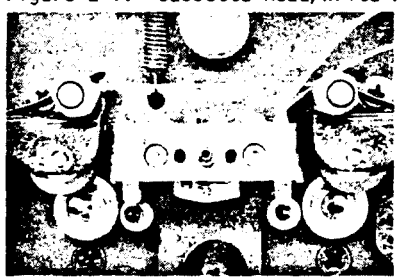


Figure E-2. Read/Write Head in Lowered Position

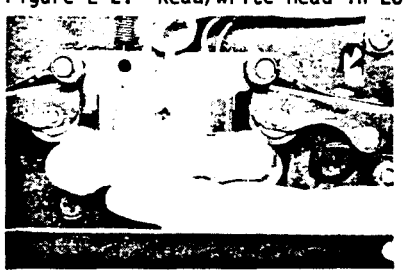


Figure E-3. Using the Cleaning Pad

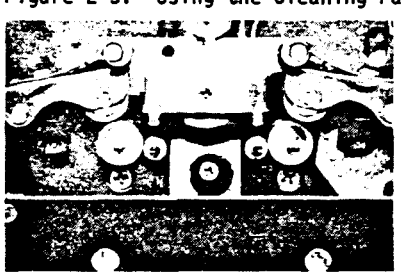


Figure E-4. Read/Write Head in Normal Position

INDEX

EQUIPMENT MAINTENANCE

Wang Laboratories recommends that the Model 2217 and Model 2218 cassette drives be serviced semi-annually. A Maintenance Agreement is available to assure this servicing automatically. If no Maintenance Agreement is acquired, any servicing must be arranged by the customer. A Maintenance Agreement protects your investment and offers the following benefits:

1. Preventive Maintenance: Semi-annually, your Model 2217 or Model 2218 cassette drive is inspected and updated with engineering changes, if any. Preventive maintenance minimizes "downtime" by anticipating repairs before they are necessary.

2. Fixed Annual Cost: When you buy a maintenance agreement, you issue only one purchase order for service for an entire year and receive one annual billing, or more frequent billing, if desired.

Further information regarding a Maintenance Agreement can be acquired from your local Sales Service Office.

> Note:
>
> Wang Laboratories, Inc. will not guarantee or honor maintenance agreements for any equipment modified by the user. Damage to equipment incurred as a result of user modifications will be the financial responsibility of the user.

To help us to provide you with the best manuals possible, please make your comments and suggestions concerning this publication on the form below. Then detach, fold, tape closed and mail to us. All comments and suggestions become the property of Wang Laboratories, Inc. For a reply, be sure to include your name and address. Your cooperation is appreciated.

TAPE CASSETTE DRIVE REFERENCE MANUAL Models 2217 & 2218 **700-3427**

TITLE OF MANUAL:

COMMENTS:

Fold

Fold

(Please tape. Postal regulations prohibit the use of staples.)

WANG

Fold

FIRST CLASS
PERMIT NO. 16
Tewksbury, Mass.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

– POSTAGE WILL BE PAID BY –

WANG LABORATORIES, INC.
836 NORTH STREET
TEWKSBURY, MASSACHUSETTS 01876

Attention: Marketing Department

Fold

Cut along dotted line.

Printed in U.S.A.

WANG LABORATORIES (CANADA) LTD.
49 Valleybrook Drive
Don Mills, Ontario M3B 2S6
TELEPHONE (416) 449-2175
Telex: 069-66546

WANG EUROPE, S.A.
Buurtweg 13
9412 Ottergem
Belgium
TELEPHONE 053/704514
Telex: 26077

WANG ELECTRONICS LTD.
1 Olympic Way, 4th Floor
Wembley Park,
Middlesex, England
TELEPHONE 01/903/6755
Telex: 923498

WANG FRANCE S.A.R.L.
Tour Galleni, 1
78/80 Ave. Galleni
93170, Bagnolet, France
TELEPHONE 8589007
Telex: 68958

WANG LABORATORIES GMBH
Moselstrasse 4
6000 Frankfurt AM Main
West Germany
TELEPHONE (0611) 252061
Telex: 04-16246

WANG SKANDINAVISKA AB
Fredsgatan 17, Box 122
S-172 23 Sundbyberg 1, Sweden
TELEPHONE 08-98-1245
Telex: 11498

WANG NEDERLAND B.V.
Damstraat 2
Utrecht, Netherlands
(030) 93-09-47
Telex: 47579

WANG PACIFIC LTD.
902-3, Wong House
26-30 Des Voeux Road, West
Hong Kong
TELEPHONE 5-435229
Telex: HX4879

WANG INDUSTRIAL CO., LTD.
110-118 Kuang-Fu N. Road
Taipei, Taiwan
Republic of China
TELEPHONE 784181-3
Telex: 21713

WANG GESELLSCHAFT M.B.H.
Formanekgasse 12-14
A-1190 Vienna, Austria
TELEPHONE 36.60.652
Telex: 74640

WANG COMPUTER PTY. LTD.
25 Bridge Street
Pymble, NSW 2073
Australia
TELEPHONE 449-6388

WANG DO BRAZIL COMPUTADORES LTDA.
Rua Barao de Lucena No. 32
Bota Fogo. ZC-01, 20,000
Rio de Janeiro, Brazil
TELEPHONE 246 7959

WANG COMPUTERS (SO. AFRICA) PTY. LTD.
Corner of Allen Rd. & Garden St.
Bordeaux, Transvaal
Republic of South Africa
TELEPHONE 486-123

WANG INTERNATIONAL TRADE, INC.
836 North Street
Tewksbury, Massachusetts 01876
TELEPHONE (617) 851-4111
TWX 710-343-6769
TELEX 94-7421

WANG COMPUTER SERVICES
836 North Street
Tewksbury, Massachusetts 01876
TELEPHONE (617) 851-4111
TWX 710-343-6769
TELEX 94-7421

24 Mill Street
Arlington, Massachusetts 02174
TELEPHONE (617) 648-8550

WANG LABORATORIES, INC.
836 NORTH STREET TEWKSBURY MASSACHUSETTS 01876 TEL (617) 851-4111 TWX 710 343-6769 TELEX 94-7421

Printed in U.S.A.
700-3427
3-75-5M
Price $10.00