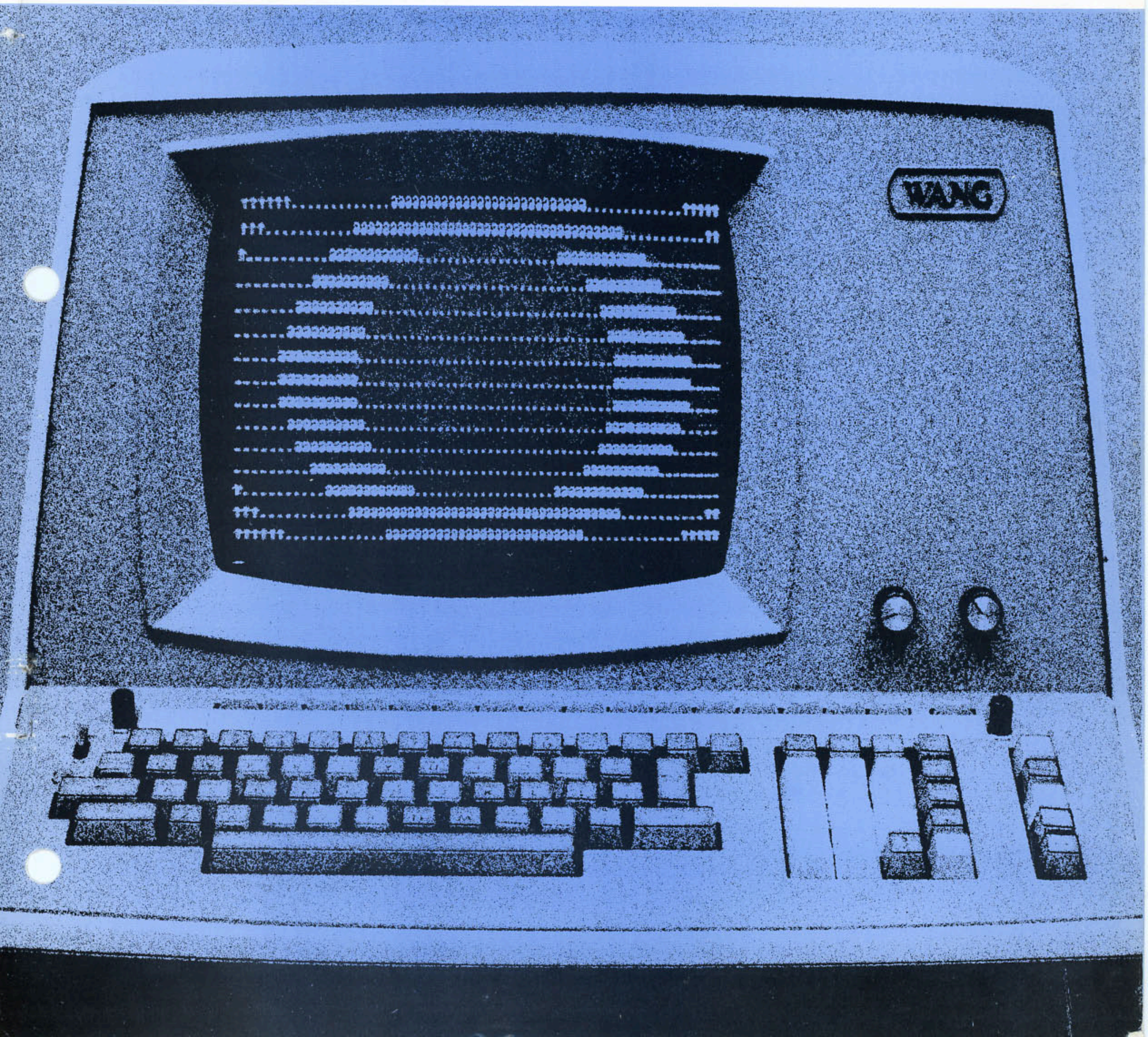
WANG

2234A/2244A HOPPER-FEED CARD READERS REFERENCE MANUAL

SYSTEM 2200



2234A/2244A

Hopper-Feed Punched Card Reader & Hopper-Feed Punched/Mark Sense Card Reader Reference Manual

©Wang Laboratories, Inc., 1974

WANG LABORATORIES, INC.
ONE INDUSTRIAL AVENUE, LOWELL, MASSACHUSETTS 01851, TEL. (617) 459-5000, TWX 710 343-6769, TELEX 94-7421

HOW TO USE THIS MANUAL

Wang's Model 2234A and Model 2244A Hopper Feed Card Readers present the user with a perhaps bewildering variety of card reading capabilities. The Model 2234A offers no fewer than eight separate card reading modes, while the Model 2244A, with its capability to read mark sense as well as punch cards, offers a total of 12 reading modes. (The characteristic features of the several reading modes are summarized in Chapters 1 and 3.)

Such wide-ranging versatility presents certain difficulties for the explanation of card reader operation, since most users are concerned only with one or two of the many available reading modes, and do not wish to be bothered with a discussion of capabilities for which they have no use. To provide the reader with ready access to material which is relevant to his particular needs, therefore, the Reference Manual is designed in a modular fashion.

Each reading mode is treated in a single, self-contained chapter, which exhaustively describes the procedures for formatting and reading cards in that mode. Chapters 4-9 deal with the several modes of reading Hollerith program and data cards, Chapters 10-14 with the reading of BASIC mark sense program/data cards, and Chapters 15 and 16 with binary card reading.

The manual's rather imposing size should not, therefore, be a cause for uneasiness. Apart from the first three chapters, which present general introductory and operational information of interest to all users, the reader should find that one or two chapters will provide him with everything he needs to know for his application.

An omnibus programming chapter (Chapter 17) also is included to provide assistance in programming the card reader. Chapter 17 discusses a variety of programming techniques for processing data from cards, and for handling error conditions under program control. Chapter 18, lastly, contains a complete list of mark sense and punch card specifications, for those who wish to design customized cards.

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.
ONE INDUSTRIAL AVENUE, LOWELL, MASSACHUSETTS 01851, TEL. (617) 459-5000, TWX 710 343-6769, TELEX 94-7421

TABLE OF CONTENTS

PART III
BASIC MARK SENSE PROGRAM/DATA CARDS

LIST OF EXAMPLES

LIST OF FIGURES

LIST OF TABLES

PART I

GENERAL INFORMATION

CHAPTER 1
INTRODUCTION TO CARD READER OPERATION

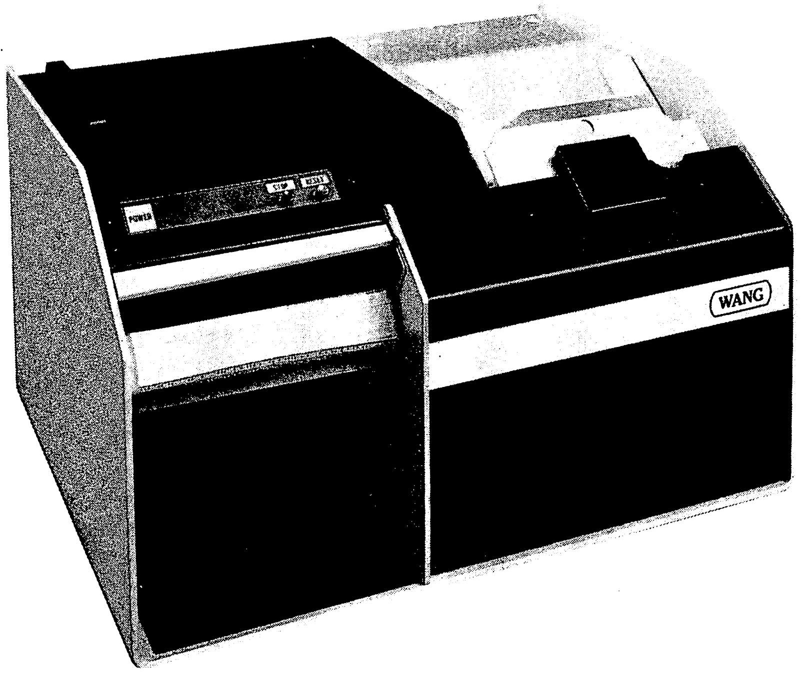


Figure 1-1. The Models 2234A/2244A (The Two Units Have Identical Front Panels)

1.1 INTRODUCTION

Wang's two hopper-feed card readers, the Model 2234A and the Model 2244A, offer a fast and efficient means of entering data from cards into the system. Both card readers can read standard punched cards. The Model 2244A offers the additional capability to read mark sense cards, a valuable feature for users who do not have access to a keypunch, or who want to design their own mark sense cards for data entry.

Hollerith and Binary Punched Data Cards

The keynote of both card readers is versatility. Both readers simplify and speed up the processing of standard 80-column Hollerith punched cards by automatically converting the Hollerith code into ASCII, the code used internally by the system, as the cards are read. The user is therefore spared the task of writing his own code conversion routine for Hollerith. For cards punched or marked in non-Hollerith codes, such as binary cards or custom-designed mark sense cards, the card readers can read straight binary data from a card (two binary bytes per card column), without performing a code conversion. In such cases, it is the programmer's job to write a code conversion routine which will translate the binary data into a meaningful form under software control.

NOTE:

The automatic Hollerith-to-ASCII conversion performed by the card readers assumes the expanded version of Hollerith utilized on the IBM Model 29 Keypunch. The earlier Model 26 Keypunch does not employ expanded Hollerith. There are, therefore, differences in the codes for certain special symbols between the Model 26 and the Model 29. Model 26 owners must identify the equivalent Model 29 codes for the characters in question, and perform their own code conversions for those characters.

BASIC Mark Sense Data Cards

With its capability to read mark sense cards, the Model 2244A offers an additional degree of versatility. In cases where no keypunch is available or where data collection must be carried out on-site, the use of punched cards may be impractical or impossible. The Model 2244A provides the capability to read several types of standard mark sense cards, as well as a wide range of custom-designed cards. Standard 80-column mark sense cards can, of course, be marked in Hollerith just as they would be punched. Because the 80-column format is somewhat inconvenient for marking data, however, two types of cards specially designed for marking programs or data in a simpler, non-code-oriented format are available. Standard educational format BASIC mark sense cards and Wang format BASIC mark sense cards both offer a card format and character code much simpler and more convenient for marking programs and data than the standard Hollerith format. The codes used on both BASIC mark sense cards are automatically converted into ASCII by the card reader as the cards are read. In these cases as with Hollerith, therefore, the user is not required to write his own code conversion routines.

Custom Designed Mark Sense Cards

Mark sense cards also offer an enormous range of flexibility in the design of customized cards. Special codes and card formats used on custom-designed cards can be read as pure binary data, and translated into meaningful data under program control. The data manipulation language features available on most Wang systems can support a wide variety of code conversion routines.

"Look-Ahead" Mode

In addition to their versatility in reading a variety of different kinds of cards, both the Model 2234A and the Model 2244A provide a special "look-ahead" feature which can speed up the processing of data cards in many applications. The "look-ahead" feature makes it possible for the card reader to access and read the next card in the hopper while data from the previous card is being processed in the CPU. Because card reading is overlapped with CPU processing, the total throughput time for processing cards can be reduced substantially in many applications.

Program Entry with Cards and "Batch Processing"

Off-line data collection and storage is, of course, only one facet of card reader operation. The off-line programming capability offered by the card reader may be equally as important in multiple-user situations. If a keypunch unit is available, BASIC programs can be punched off-line in Hollerith and loaded into the system with the Model 2234A or the Model 2244A. The Model 2244A offers the additional capability to read programs marked on either of the two types of BASIC Mark Sense program cards. Both punched and mark sense program decks can be loaded and run in batch processing mode. In this mode, a series of separate programs (and associated data decks) can be loaded, listed, and run automatically in sequence. To facilitate debugging of the individual programs, erroneous lines in each program are printed or displayed, along with the appropriate error codes.

In summary, the Models 2234A and 2244A serve the needs of the user who is utilizing standard cards and is concerned primarily with ease and simplicity of operation, while also providing the versatility to handle a wide range of non-standard card formats and codes. Techniques for reading Hollerith and non-Hollerith punched cards (available on both the Model 2234A and the Model 2244A) are discussed in Chapters 4 through 9, and in Chapters 15 and 16. Techniques for reading standard and non-standard mark sense cards (available only on the Model 2244A) are covered in Chapters 10 through 14.

1.2 THEORY OF OPERATION

For purposes of general description, the card reader may be thought to consist of four basic components:

1. The input hopper, in which the cards to be read are stacked. (Chapter 2, Section 2.4, describes the proper technique for loading card decks in the input hopper.) The input hopper holds a maximum of 550 cards.

2. The picking mechanism, which picks cards from the input hopper and passes them through to the reading station.

3. The reading station, which reads data from the cards.

4. The output stacker, in which the cards are stacked after they have been read. The output stacker, like the input hopper, holds a maximum of 550 cards.

The card reading mechanism is designed around an air-flow system which uses air pressure to separate cards in the hopper, and a vacuum to pick individual cards from the hopper. Pressurized air from a blower riffles the first half inch of cards in the input hopper, so that they stand apart, individually "air-cushioned" from the rest of the cards and from each other. The air cushion prevents the cards from sticking together because of static electricity, hole locking, or torn webs, and eliminates frictional forces between the cards. When the card reader is in AUTO SHUTDOWN mode (normal operating mode), the blower is activated by depressing the RESET button on the card reader front panel, and shuts down automatically when the hopper is emptied of cards, or when the card deck is momentarily lifted from the hopper. In MANUAL SHUTDOWN mode, the blower operates continuously while the card reader power is on.

At the outset of the picking operation, a vacuum picker in the bottom of the input hopper pulls the bottom card down and holds it against the picker's rubber surface. When a read command is received from the system (or from the card reader itself, if it is being operated off-line in LOCAL MODE), the picker mechanism rotates slightly, moving the card forward until its leading edge catches in a pair of rollers. A series of rollers then pass the card from the input hopper into and through the read station. As the card in the track clears the picker's surface, the next card is sucked down in preparation for the next read command. The vacuum picker offers significant advantages over most mechanical pickers because it subjects cards to a minimum of wear and tear, and can handle worn cards with much less difficulty than a mechanical pick mechanism.

The read station consists of two parts, a light station and, depending upon the card reader model, either a single or dual read head. The light station consists of 12 light emitting diodes (LEDs), while the read heads consist of 12 phototransistor sensors arranged in vertical columns. The Model 2234A has a single punched card read head, located directly opposite the light station. The Model 2244A has a dual read head, one located opposite the light station for reading punched cards, the second located next to the light station for reading reflective information from mark sense cards.

As a punched card passes between the light emitting diodes of the light station and the 12 photo transistors of the read head, light and dark conditions in the twelve rows of each column on the card are sensed and converted to digital form. A light condition is registered where light passes through a punched hole, and is read as a 1-bit. A dark condition is registered where no hole is punched, and is read as a 0-bit.

With mark sense cards, the technique is similar except that the mark sense read head is positioned to detect light reflecting from the card rather than light passing through it. Where the reflectivity in a row falls significantly below the background reflectivity of the card (indicating the presence of a data mark), a light condition (1-bit) is read. Where the reflectivity remains constant, a dark condition (no data mark, 0-bit) is read. This process is repeated for all valid data columns on the card.

An internal column counter automatically counts data columns as the card passes through the reading station, synchronizing the mechanical card movement with the reading electronics. At the same time, an internal timer ensures that each of the card's 80 columns is properly aligned in the reading station before an attempt is made to read it. Because the internal timer is set up to expect 80 data columns at specific intervals on the card, data cards which do not have their own index marks must conform to established punched card standards. (See Section 1.4 for a description of the applicable standards for 80-column punched and mark sense cards.)

In the Model 2234A, which cannot sense index marks, only 80-column punched cards which conform to the specified standards can be read. The Model 2244A, however, provides an additional capability to sense index marks on a card. When the INDEX MARKS switch on the rear panel of the Model 2244A is set to CLOCK, the card reader's internal timer is disabled, and index marks on the card itself are sensed and utilized to align data columns in the read station. Punched or mark sense cards which contain index marks (also called "timing marks" and "clock marks") may have fewer than 80 columns, since the width, spacing and number of data columns is determined by the spacing of the index marks. General specifications for custom designing a card with index marks are described briefly in Section 1.4, and in greater detail in Chapter 18.

When the last data column on the card has been read, the card is transferred to the output stacker. Cards are stacked in the output stacker in exactly the same order in which they were read from the input hopper. As the last card in the input hopper is read and placed in the output stacker, the machine signals a HOPPER CHECK and STOP condition, and automatically shuts down the vacuum/blower.

1.3 PUNCHED/MARK SENSE CARD SPECIFICATIONS

Punched Cards

Both the Model 2234A and the Model 2244A are equipped to read standard 80-column, 12-row punched cards. In general, data and program cards are punched on a keypunch, which is designed to align data columns and register punch marks according to a standard format. The standards which apply to punched cards are established by the American National Standards Institute (ANSI), and are detailed in their publication "Rectangular Holes in Twelve-Row Punched Cards" (ANSI specifications X3.21-1967). Punch cards read by the Model 2234A or 2244A must conform to these specifications. Figure 1-2 illustrates a typical 80-column punch card.

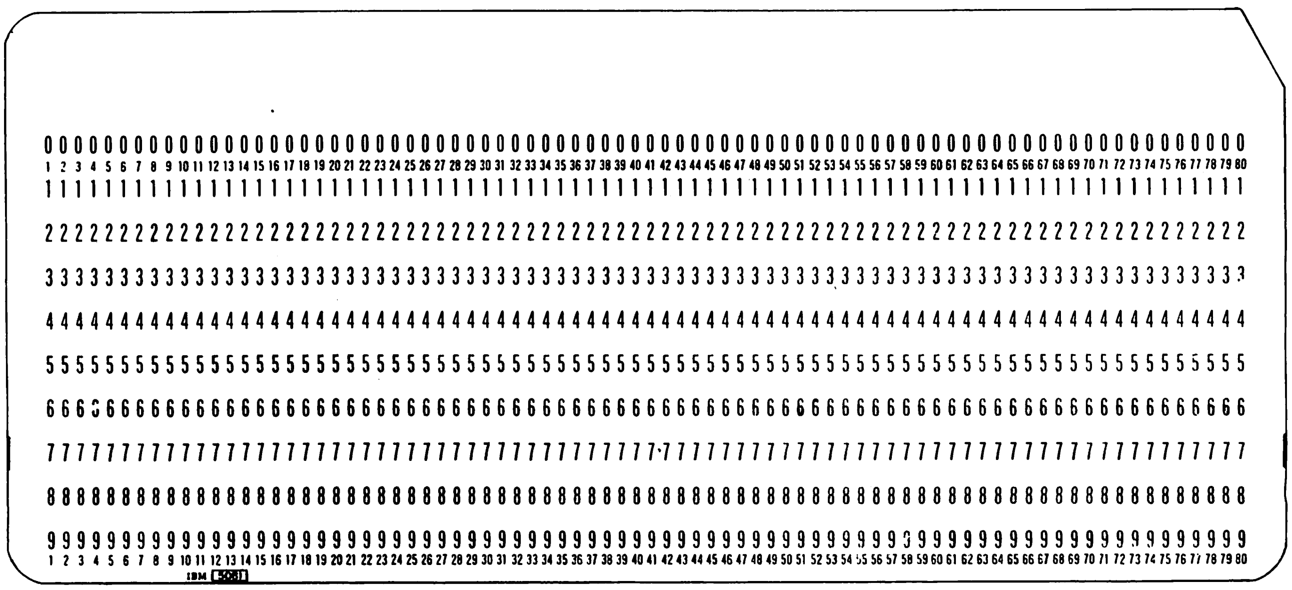


Figure 1-2. Typical 80-Column Punched Card

Mark Sense Cards

The Model 2244A has the capability to read mark sense cards as well as punched cards, and to sense index marks on a card and use them to align the data columns. These additional features make it possible to design special mark sense cards for processing through the 2244A. Two types of BASIC mark sense program/data cards, the Wang BASIC mark sense card (illustrated in Figure 1-3), and the standard BASIC mark sense card (illustrated in Figure 1-4), can be read and automatically decoded by the Model 2244A. Note that both types of cards have fewer than 80 columns. The number and size of data columns on cards which contain index marks are defined by the spacing of the index marks on the card. The use of index marks is not restricted to mark sense cards; an 80-column card which contains index marks, and which can be used either as a punched card or as a mark sense card, is illustrated in Figure 1-5. Similarly, a 40-column card with index marks which may be punched or marked is shown in Figure 1-6. In each case, it is possible to place a combination of punches and marks on the same card. Standards which prescribe the reflectivity of data marks and printed material on a mark sense card, as well as the location of data rows and the spacing of data columns and index marks, are described in ANSI specifications X3.21-1967, and in Chapter 18 of this manual. Users who wish to design their own special cards should refer to Chapter 18. The cards illustrated in Figures 1-3, 1-5, and 1-6 below may be purchased directly from Wang Laboratories.

NOTE:

Printed information on cards which are to be used interchangeably for punching and marking must meet the reflectance requirements for mark sense cards. Refer to Chapter 18 for a complete discussion of those requirements.

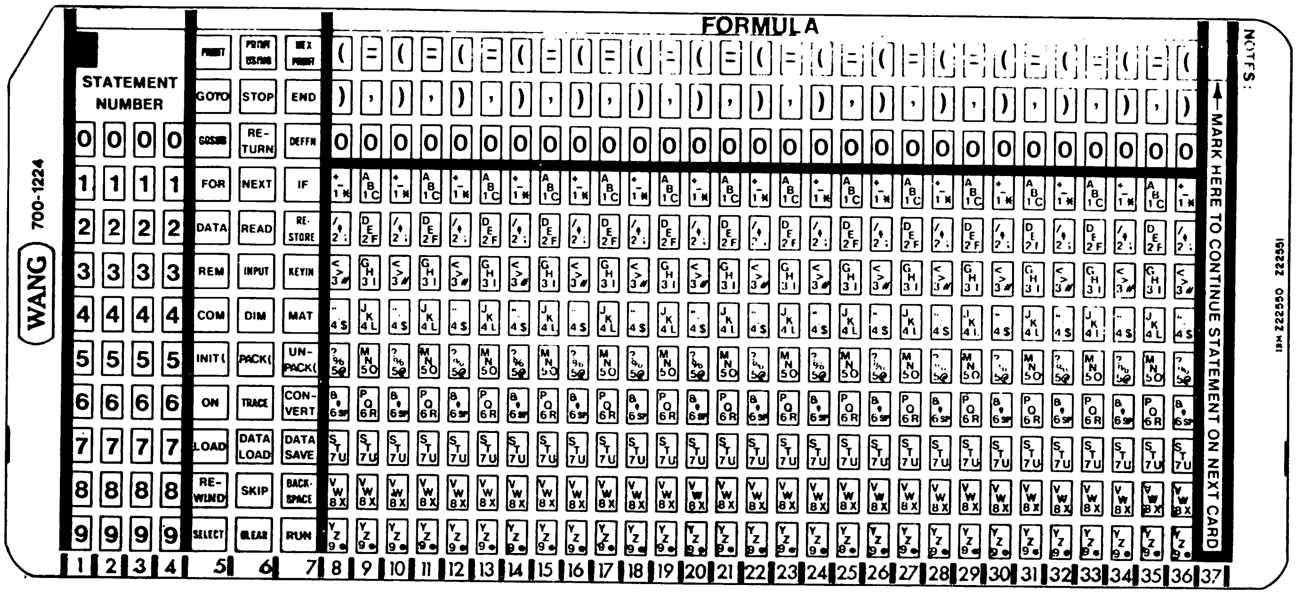


Figure 1-3 Wang BASIC Mark Sense Card (Wang Part #700-1224)

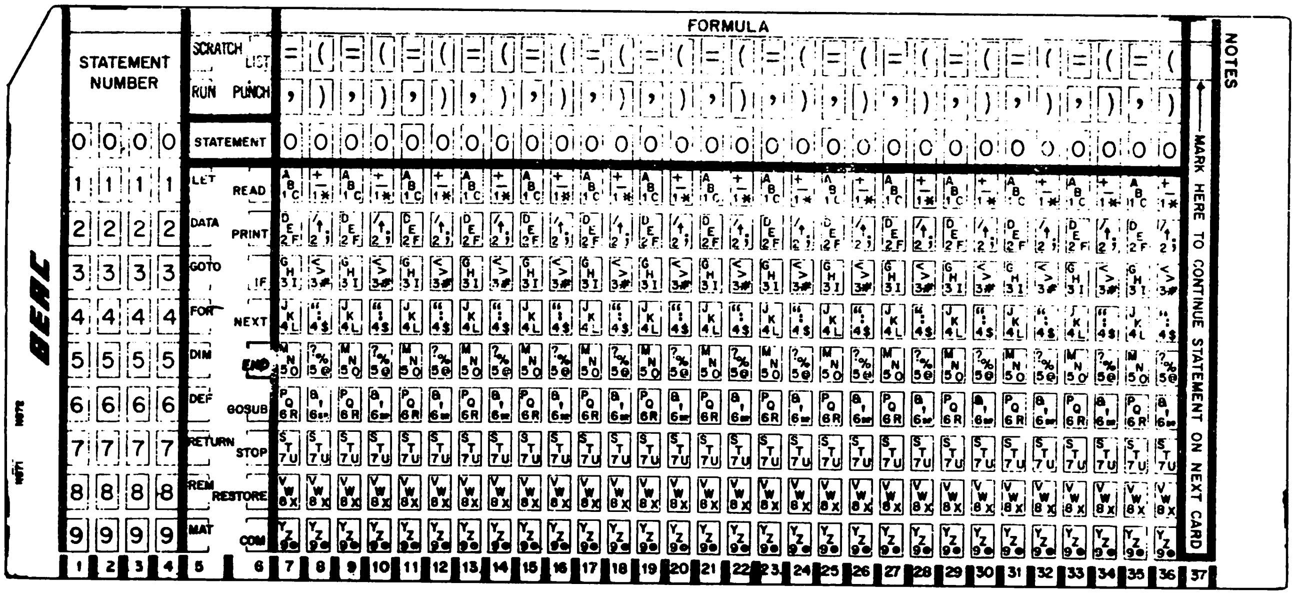


Figure 1-4 Standard BASIC Mark Sense Card
(Not Available through Wang Laboratories, Inc.)

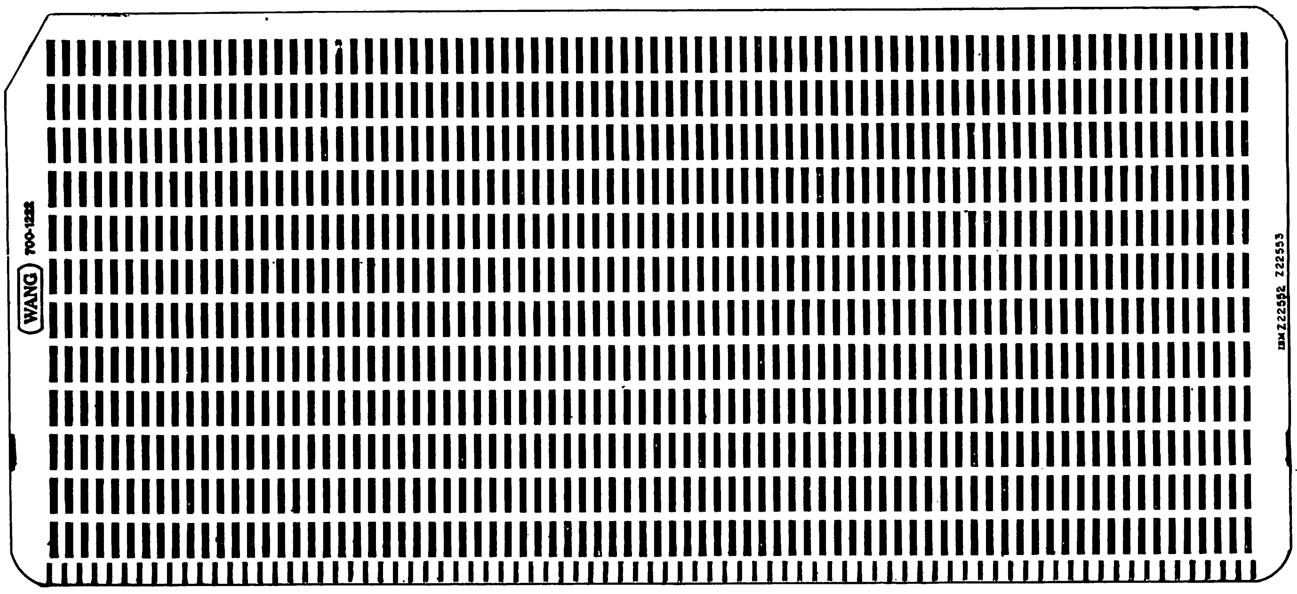


Figure 1-5 Typical 80-Column Card with Timing Marks, for Use Either as Punched Card or Mark Sense Card (Wang Part #700-1222)

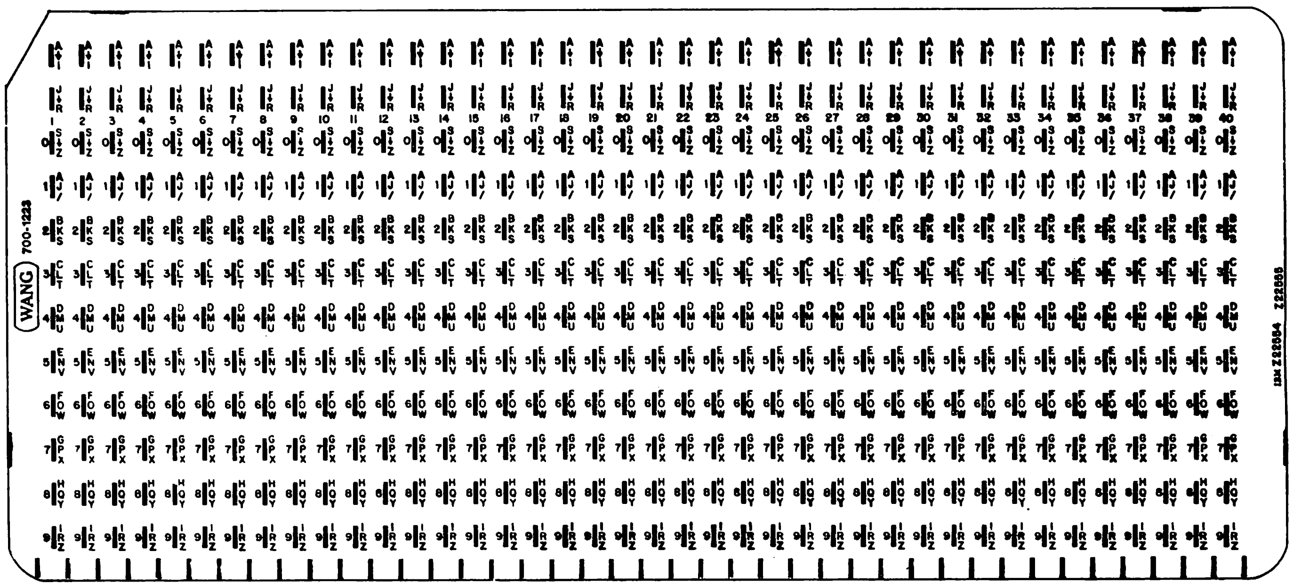


Figure 1-6 Typical 40-column Card with Timing Marks, for Use Either as Punched or Mark Sense Card. (Wang Part #700-1223.)

Card Stock

The paper stock used for all types of cards must meet certain requirements for strength, durability, thickness, etc. These requirements are described in the American National Standards Institute publication "Specifications for General Purpose Paper Cards for Information Processing" (ANSI X3.11-1969). Paper stock for mark sense cards (as well as preprinted information on the cards) must, in addition, meet certain requirements for reflectivity; these requirements are spelled out in Chapter 18.

1.4 SUMMARY OF DIFFERENCES BETWEEN THE MODEL 2234A AND THE MODEL 2244A

The Model 2234A reads only punched cards. It has a single optical reading head designed to interpret punched holes in a card as data, provided the punch registration and column spacing conform to punch card standards. The Model 2244A reads both punched cards and mark sense cards. The Model 2244A has a dual reading head; one reading head reads punched holes on a card, the other head reads #2 pencil marks on a card. Both reading heads may be activated in conjunction (so that a combination of punches and marks can be read), or the punched card reading head only may be activated (so that smudges or written comments on a punched card will not be read and interpreted as data). In addition, the Model 2244A offers the option to read cards with or without index marks. (The Model 2234A cannot read index marks.) The ability to read index marks is particularly valuable for reading mark sense program and data cards with fewer than 80 columns, and for reading custom-designed cards.

Despite differences in their internal designs, the Models 2234A and 2244A have the same front panel controls and indicators. In Section 1.5, which discusses the front panel controls, and 1.6, which discusses the front panel indicator lights, the two card reader models are treated together as one unit. Rear panel controls on the two models are significantly different, however, due to the additional features of the Model 2244A. For this reason, the rear panel controls of the two readers are treated in separate sections (Sections 1.7 and 1.8).

1.5 FRONT PANEL CONTROL BUTTONS (MODELS 2234A/2244A)

Two control buttons are located on the front panel of the Model 2234A and 2244A: STOP and RESET. Their functions are described below.

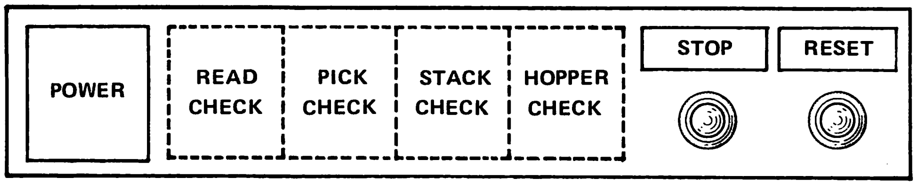


Figure 1-7. Front Panel Control Buttons and Indicator Lamps (2234A/2244A)

STOP - Depressing the STOP button halts the reading operation after the card currently being processed is read completely. STOP does not cut off power to the card reader, nor does it shut down the blower. The red STOP lamp over the STOP button illuminates during a STOP condition.

RESET - Depressing the RESET button clears and resets all error indicators on the card reader, and activates the blower if it is not already operational. The card reader is then ready to be accessed by the system. The RESET button must be depressed for initial start-up of the card reader, or to restart the reader after it has been halted by a stop condition. When RESET is depressed, the RESET indicator lamp glows greenly. (Note that the card reader RESET button should not be confused with the RESET button on the System Keyboard.)

1.6 FRONT PANEL INDICATOR LAMPS (2234A/2244A)

In addition to the control buttons, the card reader front panel presents a row of five indicator lamps: POWER, READ CHECK, PICK CHECK, STACK CHECK, and HOPPER CHECK. With the exception of POWER, these lamps are illuminated automatically by the card reader to indicate specific error condition(s). The meaning of each indicator lamp is explained below.

POWER - Illuminates when the unit's AC Power Switch, located on the rear panel of the card reader, is switched ON.

READ CHECK - Indicates that the card just read is torn on the leading edge, or contains punches or marks before the first data column. Replace the bad card, and depress the card reader RESET button to restart the reader.

STACK CHECK - Indicates that the last card read has not been properly seated in the output stacker. Check the card track to be sure it is clear, and check the stacker for a badly warped or mutilated card. Clear and stack the card manually. Depress the card reader RESET button to restart the reader.

PICK CHECK - Indicates that a card has failed to reach the read station during a read operation. Inspect the cards in the input hopper for excessive leading edge damage, torn webs, or cards stapled together. Remove faulty cards. For stapled cards, remove the staple, straighten the card, and reinsert. If no faulty cards can be found, check for excessive warpage in the card deck (in excess of one inch), and/or ink glaze buildup on the picker face. If necessary, clean the picker face with denatured alcohol. When corrective action has been taken, restart the unit with the card reader RESET button.

HOPPER CHECK- Indicates either that the input hopper is empty or that the output stacker is full. In either case, a normal operational occurrence. Empty the stacker or refill the hopper, and restart the card reader with RESET.

In certain modes of card reader operation, the error check conditions are signalled to the controlling program via error codes generated by the card reader. In such cases, restart procedures may be determined by the controlling program.

1.7 MODEL 2234A REAR PANEL CONTROLS

Three control switches are grouped in the upper right-hand corner of the Model 2234A rear panel: LAMP TEST, SHUTDOWN, and MODE. The AC Power ON/OFF switch is isolated in the upper left-hand corner of the rear panel. (See Figure 1-8.)

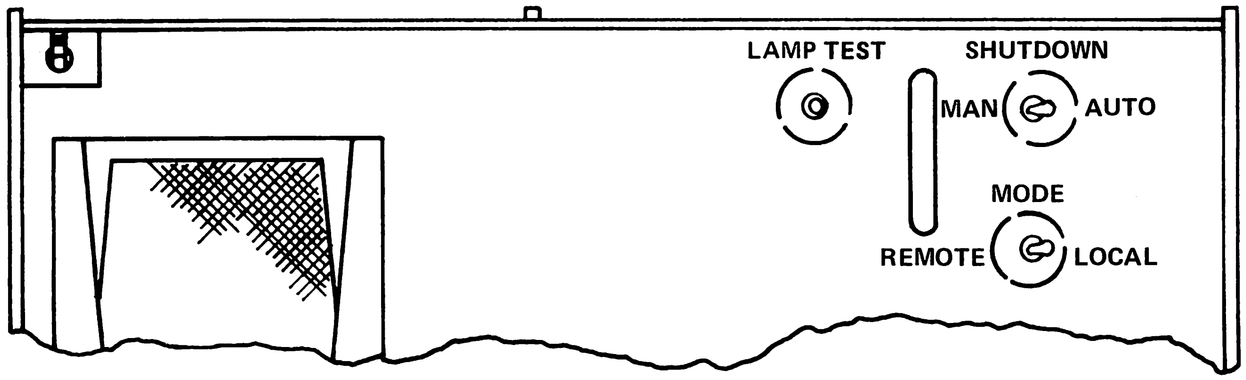


Figure 1-8 Model 2234A Rear Panel Controls

AC POWER SWITCH - Used to power the card reader on and off. The toggle switch has two positions:

UP - Power on.
DOWN - Power off.

When the power switch is up (ON), the POWER lamp on the front panel illuminates.

LAMP TEST - This button illuminates all indicator lamps on the front panel. It should be used periodically to check for burned out lamps.

SHUTDOWN - The SHUTDOWN toggle switch controls the blower, and has two positions, MAN (manual) and AUTO. In MANUAL mode, the blower operates continuously while power is ON, whether or not there are cards in the input hopper. This mode is used to operate the card reader off-line for maintenance purposes. Normally, the SHUTDOWN switch should be in the AUTO position. In AUTO mode, the blower is started by depressing the RESET button, and automatically shuts down when the input hopper is emptied, or when the card deck is lifted momentarily out of the hopper. Note that the blower typically requires about three seconds to reach normal operating speed after it is activated.

MODE - The MODE toggle switch has two settings, LOCAL and REMOTE. In LOCAL mode, the card reader can be operated off-line by depressing the RESET button on the front panel. Off-line operation is useful during certain maintenance procedures. For normal operation, the MODE switch must be in the REMOTE position. In REMOTE mode, the card reader is operated under system control.

1.8 MODEL 2244A REAR PANEL CONTROLS

Five control switches are grouped in the upper right-hand corner of the Model 2244A rear panel: LAMP TEST, INDEX MARKS, SHUTDOWN, DATA MODE, and CONTROL MODE. The AC Power ON/OFF switch is isolated in the upper left-hand corner of the rear panel (see Figure 1-9).

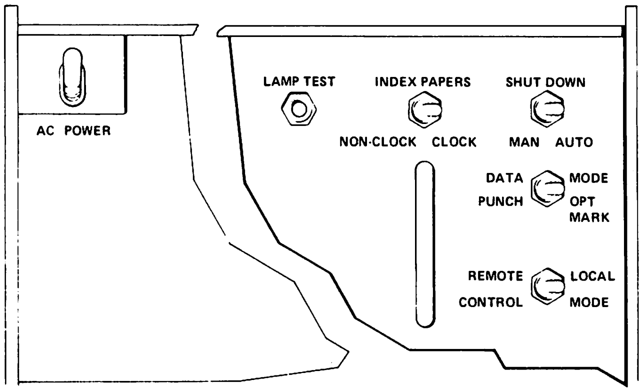


Fig. 1-9 Model 2244A Rear Panel Controls

AC POWER SWITCH - Used to power the card reader on and off. The toggle switch has two positions:

UP - Power on.
DOWN - Power off.

The POWER lamp on the front panel illuminates when the unit is powered on, and remains lit until power is turned off.

LAMP TEST - This push button illuminates all indicator lamps on the card reader front panel. It should be used periodically to check for burned out lamps.

SHUTDOWN - The SHUTDOWN switch controls the blower, and has two positions, MAN (MANUAL), and AUTO. In MANUAL mode, the hopper blower operates continuously while power remains on, whether or not there are cards in the hopper. This mode is used to operate the blower off-line during certain maintenance procedures. Normally, SHUTDOWN should be left in the AUTO position. In that mode, the blower is activated only by depressing the RESET button on the front panel, and shuts down automatically when the input hopper is emptied, or when the card deck is lifted momentarily from the hopper. Note that the blower generally takes about three seconds to reach normal operating speed after it is activated.

CONTROL MODE - The CONTROL MODE switch has two positions, LOCAL and REMOTE. In LOCAL mode, the card reader can be operated off-line independently of the system by depressing the RESET button on the front panel. Off-line operation is often useful during maintenance procedures. For normal operation, the CONTROL MODE switch should be set to REMOTE. In REMOTE mode, cards are read only under system control.

DATA MODE - DATA MODE has two settings, PUNCH and OPT MARK. In PUNCH mode, the card reader's punched card reading head only is activated, and punched cards only can be read. In OPT MARK mode, both the punched card reading head and the mark sense card reading head are activated; therefore both punched and mark sense cards can be read. Note that the reader should always be left in PUNCH mode when reading punched cards only, since there is then no possibility of reading printed material, smudges, or written comments on the cards as data. (This distinction is significant because punch cards typically are printed with a non-reflective ink which will be picked up as data if the card is read in OPT MARK mode.)

INDEX MARKS - The Model 2244A is capable of reading cards with or without index marks. Index marks, also called "clock marks" and "timing marks", are heavy black lines sequenced along the bottom edge of the card. They are used by the card reader to delimit valid data columns on the card. With the INDEX MARKS switch in NON-CLOCK mode, the card reader does not look for index marks; instead, it utilizes an internal timer to align data columns in the reading station. The timer is designed to expect 80 columns on each card, spaced according to ANSI specifications X3.21-1967. Cards with or without index marks may be read in NON-CLOCK mode, as long as the spacing of data columns and punch or mark registration conforms to the designated specifications. In CLOCK mode, the card reader's internal timer is disabled, and the reader utilizes index marks on the card itself to delineate data columns. Marks or punches on the card located between two consecutive index marks are read as data in CLOCK mode. Specifications governing the design of custom mark sense cards with index marks are detailed in Chapter 18.

DATA MODE/INDEX MARKS Summary

In summary, four types of cards can be read by the Model 2244A Card Reader:

1. Punched cards without index marks (typically, standard Hollerith cards and binary punched cards).

2. Mark sense cards without index marks. (Data columns must be aligned in standard 80-column spacing.)

3. Punched cards with index marks.

4. Mark sense cards with index marks. The 2244A automatically decodes two types of such cards - standard BASIC mark sense cards and Wang BASIC mark sense cards. A variety of other formats can be used for custom-designed cards with index marks. Customized cards may also combine punches and marks on the same card, provided the punched columns conform to standard 80-column spacing (most keypunch equipment punches at 80-column intervals).

The table below summarizes the four possible modes of card reader operation.

Table 1-1

DATA MODE and INDEX MARKS Settings for Different Types of Cards

TYPE OF CARD	DATA MODE Setting	INDEX MARKS Setting
Punched cards without index marks (e.g., normal Hollerith punched and binary punched cards)*	PUNCH	NON-CLOCK
Mark sense cards without index marks (may be marked only, or both punched and marked)**	OPT MARK	NON-CLOCK
Punched cards with index marks	PUNCH	CLOCK
Mark sense cards with index marks (e.g., Standard BASIC or Wang BASIC Mark Sense Cards, or custom-designed cards. Custom-designed cards may combine marks and punches, as long as punched columns conform to 80-column spacing per ANSI x 3.21-1967.)	OPT MARK	CLOCK

*Punched cards without index marks must conform to ANSI standards X3.21-1967.

**Mark sense cards without index marks must conform to ANSI X3.21-1968.

In all cases, card stock must conform to ANSI standards X3.11-1967.

NOTE:

The Model 2244A can read a combination of punched and mark sense cards in the same job stream, provided all cards in the job stream meet the requirements specified in Chapter 18 for reflectance of printed information on cards. It is important to recognize that many standard types of punched cards are printed in a non-reflective ink which will register as data when read in OPT MARK mode. Such punch cards cannot be combined with mark sense cards in the same job stream. Additionally, it is not possible to read cards which do not have index marks in the same job stream with cards which do have index marks, since the reader must be set either to expect index marks on all cards (CLOCK mode), or to ignore them on all cards (NON-CLOCK mode). Many standard types of punch cards do not have index marks, while most mark sense cards do. The 80- and 40- column punch/mark sense cards sold by Wang (see Figures 1-5 and 1-6) are printed in reflective ink, and contain index marks. These cards may be read interchangeably in PUNCH or OPT MARK mode. The 80-column card may be read in CLOCK or NON-CLOCK mode. Both cards therefore enable the user to combine punches and marks on the same card, or to combine punch cards and mark sense cards in the same deck.

CHAPTER 2
INSTALLING AND OPERATING THE CARD READER

2.1 UNPACKING AND INSPECTION

The Model 2234A/2244A is packed in a cardboard shipping crate, with cushioning and padding to protect it from damage during shipment. Inspect the outside of the shipping container, and report any sign of physical damage to the shipping agency at once.

Packed with the card reader are the following three items:

- Card Reader Power Cord
- 2234A or 2244A Controller Board (be sure you have the right board for your model)
- Card Reader - CPU Controller Cable

After removing the above material, lift the card reader straight up and out of the carton, and place it on a flat, sturdy support area. Carefully inspect all items for damage; if there is any indication of damage, notify the shipping agency at once. Check all equipment received against the purchase order (decals specifying model numbers are affixed to all Wang equipment, usually on the back of the unit).

Tilt the card reader back on its side, and use a phillips screwdriver to remove the two red shipping screws in the bottom plate. Save these screws; they lock the blower motor plate in a solid position to prevent damage to the motor during shipment, and must be reinstalled if the reader is reshipped.

2.2 INSTALLATION

Observe the following steps to install the card reader:

1. Be sure that all AC Power for the card reader and the system Power Supply Unit is OFF.
2. Insert the card reader controller board in an available slot in the system CPU.
3. Connect the Amphenol plug on the controller cable to the receptacle on the controller board, and fasten it in place with the two lock clips provided.

4. Connect the other end of the connector cable to the receptacle on the rear panel of the card reader. Fit the connector over the receptacle, and screw it in tightly with the thumbscrew provided. The cable connector cannot be pushed into its receptacle.

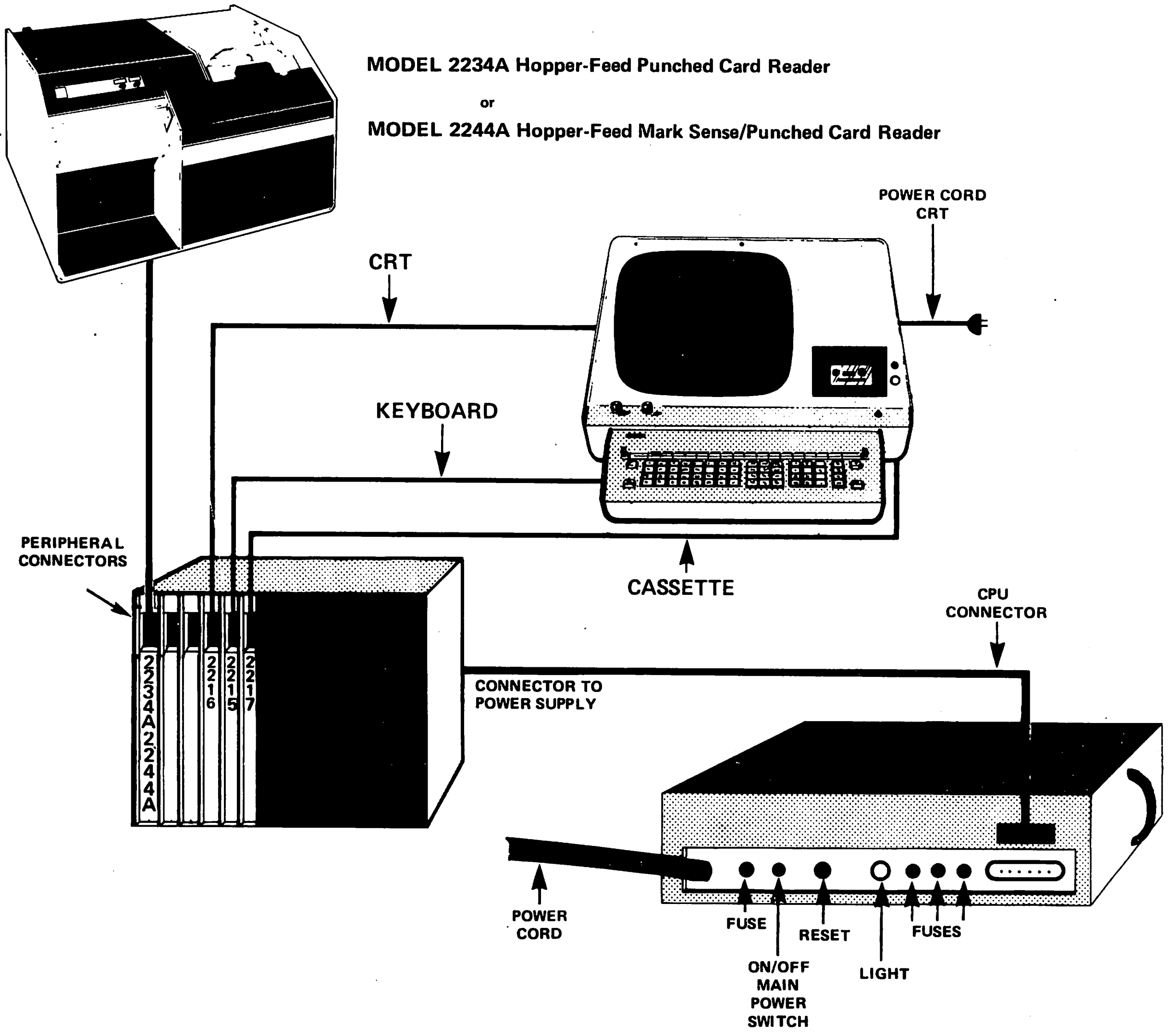


Figure 2-1. Typical System Configuration

2.3 SYSTEM POWER-ON PROCEDURES

1. Set the MODE switch (Model 2234A) or CONTROL MODE switch (Model 2244A) to REMOTE.

2. Set the SHUTDOWN switch to AUTO.

3. On the Model 2244A, be sure that the DATA MODE and INDEX MARKS switches are appropriately set for the type of cards to be read. (See Section 1.8 for an explanation of how these switches should be set.)

4. Turn ON the AC power switch on the card reader and all peripherals, and turn ON the master power switch on the system Power Supply Unit. Turning on the master power switch automatically Master Initializes the system.

2.4 LOADING CARDS IN THE INPUT HOPPER

The following procedures should be observed when loading and unloading the input hopper and output stacker.

1. Remove the hopper follower, and load the card deck into the hopper. The first card to be read should be at the bottom of the deck. All cards must be face down, with the "9" row toward the back of the hopper, and the "1" column to the left. For cards with index marks, the index marks must be at the back of the hopper. The hopper may be loosely filled with about 550 cards. Avoid packing the input hopper so full that the bottom half inch of cards cannot be riffled by the vacuum/blower unit. Replace the hopper follower.

2. The hopper may be loaded while cards are being read if the operator is careful to keep tension on the bottom portion of the deck while loading additional cards on top. Reloading should be done while the hopper is one-half to one-third full. Keep just enough pressure on the deck to maintain the riffle action.

3. Normally all cards in the hopper are processed through the reader. If, however, it becomes necessary to unload the hopper, simply remove the follower and extract the card deck. The action of lifting the card deck out of the hopper automatically causes the vacuum/blower to shut down. If the cards are arranged in a particular order, exercise care in repacking them to maintain the order.

4. The stacker is unloaded by pulling the stacker plate down and removing the card deck. Be careful that deck order is maintained.

5. The stacker can be unloaded during operation simply by pulling the stacker plate down and removing some or all of the deck (if the stacker plate is pushed all the way down, however, a STACK CHECK error is signalled, and the card reader halts). Take care to allow the stacker plate to return to its normal position gradually.

CHAPTER 3
THE MODES OF CARD READER OPERATION

3.1 READING PUNCHED AND MARK SENSE CARDS

The character code used internally by all Wang systems is ASCII. Because data and program text are typically recorded on cards in codes other than ASCII, however (Hollerith and modified versions of Hollerith are the most common), it is necessary to convert these codes into ASCII before they can be regarded meaningfully by the system. In the cases of standard Hollerith code (as used in the IBM Model 29 Keypunch), and the somewhat modified versions of Hollerith used on the BASIC mark sense cards, the card reader can convert these codes automatically into ASCII as the cards are read. For other codes, the data must be read in pure binary form and converted into a meaningful format under software control. In this case, the programmer must write his own code conversion routine.

In all cases, the various types of code conversions to be performed are indicated by specifying different device addresses when accessing the card reader. Unique device addresses identify built-in code conversion routines for Hollerith-to-ASCII and BASIC mark sense card code-to-ASCII conversion, and for straight binary reading. Other addresses enable the card reader to perform "look-ahead" operations, in which the next card is read by the card reader while data from the previous card is still being processed in the system CPU.

When combined with an appropriate BASIC statement or command (LOAD, DATALOAD, INPUT, DATALOAD BT or DATASAVE BT), individual device addresses make possible the loading of programs or data from cards, and the automatic conversion of the card code into ASCII or binary format. Each combination of a BASIC statement and a device address enables the system to load program text or data in specific code (for example, the statement LOAD/62B loads program text in Hollerith code, while the statement LOAD/62C loads program text in special mark sense card code). The BASIC statement determines the type of reading operation to be performed (load programs, data card images, or discrete data values from cards). The device address determines the type of code conversion which is to be performed, and/or the type of card format which is to be read. For this reason, each combination BASIC statement and device address is referred to as a "reading mode". The several reading modes available on the Model 2234A are listed in Section 3.4. The more extensive range of reading modes available on the Model 2244A are listed in Section 3.5. Note that the limited BASIC language sets of the System 2200A and of the 2200S and WCS/10 systems if they do not have Option -22 or -23, restrict the card reading modes available on these systems. The limitations of the System 2200A and System 2200S and the WCS/10 in card reader operations are explained in Section 3.3.

3.2 DATA VALUES VS. CARD "IMAGES"

Both card readers provide the user with the option of reading either individual data values from a card, or complete data card "images". Data values must be punched or marked in a specified system format. (The required format for Hollerith cards is explained in Chapter 4, and for the BASIC mark sense cards, in Chapter 12.)

The device addresses 628, 62B, 62C, and 62D are used with the statements INPUT or DATALOAD to read BASIC data values in a prescribed free-format, and to interpret commas as data separators. These modes of reading are convenient ones, since they automatically separate discrete data values and perform data validity checks, functions which would otherwise have to be carried out by the programmer in his software. At the same time, however, the programmer is deprived of complete control over his data. A greater degree of control is obtained by reading the entire card, not as a series of discrete data values but as a single continuous data card "image". The chief virtue of this approach lies in the fact that the programmer has greater control over the response to certain error conditions. Certain data format errors produce an automatic system error condition and program termination when encountered in data reading mode. When a card image is read, however, all data validity checks are made under software control. The program can therefore be designed to ignore an invalid value, and continue processing the remainder of the data deck. Two types of card images can be read: Hollerith images and binary images. Device address 629 is used to indicate Hollerith data card images. In this case, the complete 80 columns are read from the card, and each column is converted into ASCII. Collectively, the 80 characters read in this way represent an exact ASCII "image" of the data punched or marked on the card. In addition, the card reader sends a pair of extra bytes which specify the total number of columns read from the card, and identify reading errors and other types of card reader malfunctions which may have occurred. The programmer is then in a position to examine all of his data under program control, and conduct his own validity checks. For non-Hollerith punched or mark sense cards, device address 62A is used to read a binary card image. In this case, 80 columns of data are read as two-byte binary values which can be interpreted under program control. The techniques used for reading Hollerith data values are discussed in Chapters 4 and 7. The reading of Hollerith card images is covered in Chapter 5, and binary card images are covered in Chapter 15.

3.3 LIMITATIONS OF THE SYSTEM 2200A, AND THE SYSTEM 2200S AND WCS/10 WITHOUT OPTION -22 OR -23 IN CARD READER OPERATIONS

The Models 2234A and 2244A are designed to interface with all versions of the System 2200, as well as all three versions of the Wang Computer Systems (WCS/10, WCS/20, and WCS/30). However, the more limited BASIC language features of the System 2200A, and of the System 2200S and WCS/10, if they do not have Option-22 or -23, preclude the use of a number of card reader operations. (A System 2200S or WCS/10 equipped with Option 22, the Advanced Programming ROM, or Option-23, General I/O ROM, has a language set nearly as extensive as those of the larger systems, and is not subject to these restrictions.)

Standard card reader operations involve the use of the BASIC statements LOAD, DATALOAD, DATALOAD BT, and INPUT, as well as the use of Console Input operations. The statements LOAD, DATALOAD, DATALOAD BT, and DATASAVE BT are unavailable for card reader operations on a System 2200A, or a 2200S or WCS/10 which does not have Option -22 or -23. The DATALOAD BT statement does not appear at all in the BASIC language repertoire of a 2200A, nor in a 2200S or WCS/10 without Option-22 or -23. Both LOAD and DATALOAD are used in the 2200A, 2200S, and WCS/10 in tape cassette operations for loading programs and data. Despite the fact that these statements are available for tape cassette operations, however, they cannot be used with the card reader because they do not include certain additional logic needed to support card reader operations. (The additional logic required is provided for the System 2200S and WCS/10 in Option-22 and Option-23.) System 2200A owners, and owners of a System 2200S or WCS/10 which does not have Option-22 or -23, can operate the card readers with INPUT and Console Input only. INPUT is covered in Chapters 7 and 12; Console Input, in Chapters 9 and 14.

3.4 MODEL 2234A READING MODES

The Model 2234A Hopper Feed Punched Card Reader utilizes the BASIC statements LOAD, DATALOAD, INPUT, DATALOAD BT, and DATASAVE BT for card reader operations. Alternatively, cards can be read by selecting the Model 2234A for Console Input operations. (Note that the BASIC statements LOAD, DATALOAD, DATASAVE BT, and DATALOAD BT are not available on the System 2200A and on standard versions of the 2200S and WCS/10; hence cards can be read only via INPUT or Console Input in those Systems.) A total of six reading modes, each identified with a unique device address, are available on the Model 2234A:

Reading Mode	Device Address	BASIC Statement	Operation
1	62B	LOAD	Load or overlay Hollerith program text from cards, convert automatically into ASCII and Wang program format.
2	628	DATALOAD	Read Hollerith data values in free format separated by commas from cards, convert automatically to ASCII.
3	629	DATALOAD BT	Read 80-column Hollerith card image, convert each character into ASCII.
4	62A	DATALOAD BT	Read 80-column binary card image, store each character as a two-byte binary value (6/6 format).

5	42E	DATASAVE BT	Hollerith Look-Ahead Mode: Initiate the reading and conversion of next card in the input hopper. The card is read as a Hollerith card image, converted to ASCII, and held in the card reader output buffer while CPU processing (possibly processing data from the previous card) continues. Subsequently, the preprocessed card can be read with DATALOAD or DATALOAD BT.
6	42F	DATASAVE BT	Binary Look-Ahead mode: Initiate the reading and conversion of the next card in the input hopper. The card is read as an 80-column binary card image, and is held in the card reader output buffer while CPU processing (typically processing data from the previous card) continues. Subsequently, the preprocessed data can be read with a DATALOAD BT statement. Each column is stored as two binary bytes.

In addition to the six standard card reading modes, the card reader can be used to read Hollerith BASIC program cards if it is selected for Console Input with a special device address of 02B (see below). Hollerith Data values in free-format can be read with an input statement if the Card Reader is first selected for input operations with a special address of 62B (see below).

SELECT Statement/Address	Operation
SELECT CI 02B	Load BASIC program text in Hollerith code from cards, convert to ASCII. Each card is echoed onto the Console Output device (normally, the CRT) as it is loaded. Program cards are automatically read, converted, and loaded in a manner similar to the entry of program text and commands from the keyboard. Console Input is used in batch processing operations (see Chapter 9).
SELECT INPUT 62B	When the card reader is selected for INPUT operations, an INPUT statement reads Hollerith data values in free format from punched cards, and converts them automatically into ASCII. Each data value is echoed onto the Console Output device (normally the CRT) as it is read.

3.5 MODEL 2244A CARD READING MODES

The Model 2244A Hopper Feed Punched/Mark Sense Card Reader utilizes the statements LOAD, DATALOAD, INPUT, DATALOAD BT, and DATASAVE BT for standard card reader operations. Alternatively, cards can be read by selecting the Model 2244A for Console Input operations. (Note that the BASIC statements LOAD, DATALOAD, DATASAVE BT, and DATALOAD BT are not available for card reader operations on the System 2200A or on standard versions of the System 2200S or WCS/10; hence, cards can be read only via INPUT or Console Input in those systems.) A total of eight standard reading modes are available on the Model 2244A, each identified with a unique device address. The eight standard modes and their device addresses are listed below:

Reading Mode	Device Address	BASIC Statement	Operation
1	62B	LOAD	Load Hollerith program text from punched or mark sense cards, convert automatically into ASCII and Wang program format.
2	628	DATALOAD	Read Hollerith data values in free format and separated by commas from punched or mark sense cards, convert to ASCII.
3	629	DATALOAD BT	Read 80-column Hollerith punched or mark sense card image, convert each character to ASCII. (Cards may have fewer than 80 columns if they contain index marks.)
4	62A	DATALOAD BT	Read 80-column punched or mark sense binary card image, store each card column as a two-byte binary value (6/6 format). (Cards may have fewer than 80 columns if they contain index marks.)
5	42E	DATASAVE BT	Hollerith Look-Ahead mode: Initiate the reading and conversion of the next card in the input hopper. Read this card as an 80-column Hollerith card image, and hold the data in the card reader output buffer while CPU processing (typically, processing the previous card) continues. Subsequently, the preprocessed data can be read with DATALOAD or DATALOAD BT.

6	42F	DATASAVE BT	Binary Look-Ahead mode: Initiate the reading and conversion of an 80-column binary card image, and hold the data in the card reader output buffer while CPU processing (typically, processing data from the previous card) continues. Subsequently, the preprocessed data can be read with DATALOAD BT. Each card column is stored as two binary bytes.
7	62C	LOAD	Load BASIC program text from BASIC mark sense program cards (standard format or Wang format), convert into ASCII and Wang program format.
8	62D	DATALOAD	Read data values in free format and separated by commas from BASIC mark sense program cards (standard format or Wang format), convert into ASCII.

In addition to the eight standard reading modes, the card reader can be used to read Hollerith or BASIC mark sense program cards if it is selected for Console Input with an appropriate device address (02B or 02C). Data values in free format on Hollerith or BASIC mark sense cards can be read with INPUT if the card reader is first selected for INPUT operations with an appropriate address (62B or 62C).

SELECT Statement/Address	Operation
SELECT CI 02B	Load BASIC program text from Hollerith punched or mark sense cards, convert to ASCII and Wang program format. Each statement line is echoed on the Console Output device (normally, the CRT) as it is loaded. Program cards are automatically read, converted to ASCII, and loaded in a manner analagous to keyboard entry. Command cards are executed immediately when read. Console Input has an important use in batch processing operations (see Chapter 9).
SELECT CI 02C	Load BASIC mark sense cards program text from BASIC mark sense cards convert into ASCII and Wang program format. Each card is echoed on the Console Output device (normally, the CRT) as it is loaded. Program cards are automatically read, converted to ASCII, and loaded in a manner analagous to keyboard entry. Command cards are executed immediately upon being read. Console Input has an important use in batch processing (see Chapter 14).

SELECT INPUT 62B	When the card reader is selected for INPUT operations with device address 62B, a subsequent INPUT statement instructs the card reader to read Hollerith data values in free format from punched or mark sense cards, and convert each valid data character into ASCII. Each value is echoed onto the Console Output device (normally, the CRT) as it is read.
SELECT INPUT 62C	When the card reader is selected for INPUT operations with device address 62C, a subsequent INPUT statement instructs the card reader to read data values from BASIC mark sense program cards and convert each valid data character into ASCII. Each data value is echoed onto the Console Output device (normally, the CRT) when read.

3.6 THE USE OF FILE NUMBERS AND THE 'SELECT TAPE' STATEMENT

Although in most cases it is most convenient to include a particular device address directly in the appropriate BASIC statement or command (for example, 'LOAD/62B' for loading Hollerith program cards) there are two alternative methods of specifying a device address indirectly. A device address may be assigned to a file number, or it can be designated as the default address for Console Tape operations in a SELECT TAPE statement.

File Numbers

File numbers provide one technique for specifying a device address indirectly. The device address is first assigned to one of the six available file numbers, #1 - #6, in a SELECT statement. Subsequently, the file number can be used in place of its assigned device address in a BASIC statement. For example, the device address 628 is used in a DATALOAD statement to initiate the reading of free-format data values in Hollerith code. Instead of including '628' directly in the DATALOAD statement, a file number could be used:

```
10 SELECT #3 628
20 DATALOAD #3, A$, B$, N
```

The SELECT TAPE Statement

For each class of I/O peripherals on the system, one device is designated as the default device in that class. The device address of the default peripheral becomes the default address, and is used whenever no other address is specified. For tape-class peripherals (of which the card reader is one), the default device (referred to as the "Console Tape" device) is the console tape cassette drive in the CRT; its address is 10A. When a tape statement (such as LOAD, DATALOAD, DATALOAD BT, or DATASAVE BT) is executed with no device address specified (directly or indirectly), the system automatically utilizes address 10A, and attempts to access the tape cassette drive.

The SELECT TAPE statement can be used to designate any one of the card reader device addresses as the Console Tape address instead of 10A. For example, the following statement designates address 62C (BASIC mark sense program cards) as Console Tape address:

```
100 SELECT TAPE 62C
```

Once this statement is executed, any tape-class operation which does not specify a different address automatically defaults to address 62C. Thus, the statement

```
150 LOAD 100, 200
```

initiates the loading of BASIC mark sense program cards, even though no device address is specified.

PART II

HOLLERITH PROGRAM AND DATA CARDS

CHAPTER 4
READING HOLLERITH DATA VALUES (DATALOAD, ADDRESS 628)

4.1 INTRODUCTION

Hollerith data cards may be read in two ways. A complete 80-character card "image" can be read with the DATALOAD BT statement and a device address of 629. In this case, the card reader reads the entire 80 columns from each card into a single alpha array, without testing for invalid data formats; individual data values must be separated and verified by the programmer under software control. This mode of reading data (Mode #3) is discussed in Chapter 6. Alternatively, the card reader can be instructed to read discrete Hollerith data values from each card. In this case, the DATALOAD statement with a device address of 628 is used. Individual data values on a card must be separated by commas. Numeric data values in a non-legal format, or containing illegal characters, generate an automatic error message, and halt program execution. This chapter discusses the use of DATALOAD to read Hollerith data values (Mode #2).

Note that Mode #2 is not recommended for data processing operations involving significant quantities of data. For such applications, the preferred reading mode is Mode #3 (DATALOAD BT, address 629), in which a complete Hollerith card image is read for each card. In Mode #3, error conditions (both card reader errors and data format errors) can be detected and dealt with under program control, without affecting continued program execution. Unacceptable data values can be printed or displayed for the operator's benefit, and processing of the remaining cards can resume. In Mode #2, however, invalid numeric data produces an automatic error condition, and causes program execution to terminate.

For reading with the Model 2234A in Mode #2, data cards must be standard 80-column cards punched in Hollerith code. For the Model 2244A, the cards may have fewer than 80 columns (provided they also have timing marks), and they may be either punched or marked; however, the data must be in Hollerith code.

4.2 READING HOLLERITH DATA VALUES

General Form: DATALOAD $\begin{bmatrix}\text{/628,}\\ \text{\#n,}\end{bmatrix}$ argument list

Where:

/628 =	The device address which designates the card reader as the device from which data is to be read, and also determines the type of code conversion routine which is to be performed, and the data format to be expected. Address 628 causes the reader to expect discrete data values, and to perform an automatic Hollerith-to-ASCII conversion for each data character read.
#n =	A file number to which the device address has been assigned in a SELECT statement ('n' is an integer from 1 to 6).
	If neither a device address nor a file number is specified, the address of the default tape device (normally the console cassette drive, address 10A) is used. Address 628 can be designated as the default address in a SELECT TAPE 628 statement. In that case, the system defaults to address 628 if no address or file number is specified.
argument list =	The list of receiving variables, array elements, and/or array designators, separated by commas. (An array designator consists of an array name followed by closed parentheses, e.g., A$(), N().)

Purpose:

The DATALOAD statement with a device address of 628 initiates the reading of discrete data values in Hollerith code from one or more data cards, converts each character to ASCII, and assigns the values read sequentially to receiving variables in the DATALOAD argument list. (Arrays are filled row by row.) Numeric values must be in free-format (see below). Multiple values on a single card must be separated by commas. If the argument list is not filled by a single card, additional cards are read until all receiving variables are filled. Unread data on the last card is lost. Both alphanumeric and numeric values may be stored in alphanumeric variables, but only legitimate BASIC numbers can be stored in numeric variables (otherwise, an error results and program execution is terminated).

Example 4-1: Loading Hollerith Data Values from Cards (DATALOAD, Address 628)

```
10 DATALOAD/628, A, B$, N$
```

Statement 10 reads Hollerith data values from cards. Because address 628 is specified, individual data values in free-format are expected. Three values, a numeric (A) and two alphanumerics (B$, N$) are read. The values may be on three separate cards, or on the same card (separated by commas). Note that the first value read must be a numeric, and the second and third may be numeric or alphanumeric. Any attempt to read an alpha value into a numeric variable produces an Error 43 (Wrong Variable Type).

It is possible to designate entire arrays as arguments in a DATALOAD argument list. In that case, each element of the array receives a separate value from the card. Arrays are filled row by row.

Example 4-2: Loading Hollerith Data Values from Punched Cards into Arrays (DATALOAD, Address 628)

```
100 DIM A$(10,10)10, B(25)
.
.
.
250 SELECT #3 628
260 DATALOAD #3, A$(), B()
```

In this example, 100 alphanumeric values are read from cards and stored in alpha array A$(). (Each value is a maximum of 10 characters in length.) Then, 25 numeric values are read and stored in numeric array B(). Note that the specified device address, 628, is selected to file number #3, and #3 then is used to designate 628 in the DATALOAD statement. Address 628 indicates that individual data values punched in Hollerith code in free-format are to be read. The DATALOAD statement continues reading cards until both arrays are filled. If there is not sufficient data in the card deck to satisfy both arrays, the system hangs up awaiting more data.

4.3 HOLLERITH DATA CARD FORMAT

Data values read in Mode #2 must be punched (or marked) on cards in a free format identical to the format in which data is entered from the keyboard. Each data character must be punched (or marked) in Hollerith code. The following rules apply to the preparation and reading of data values in Mode #2:

1. A varying number of data values can be punched on a single card. To separate individual values, a comma (',') character is punched on the card following each value; the comma separator must not be punched following the last value on the card, however.

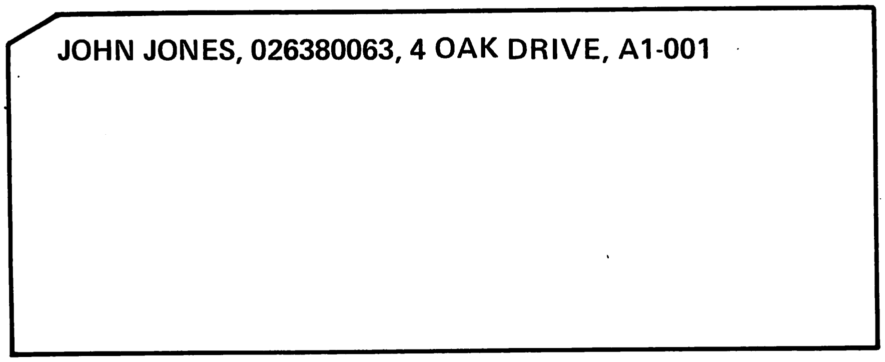


Figure 4-1.
Typical Hollerith Data Card With Multiple Data Values Separated by Commas.

2. Each DATALOAD statement can read one or more cards. If one card does not contain sufficient data values to fill all receiving variables in the DATALOAD argument list, the next card is automatically read. When, however, the last receiving variable in the DATALOAD argument list has received a value from a card, any additional data values on that card are ignored. A new DATALOAD statement begins reading with the next card, even if there are unread values on the previous card.

3. A combination of numeric and alphanumeric data values may be recorded on the same card (but individual values must be separated by commas). The receiving variables in the DATALOAD argument list must correspond sequentially to the values on the card. Alpha variables may receive either numeric or alphanumeric data, but numeric variables may contain only legitimate BASIC numbers.

4. Alphanumeric values may be recorded with or without enclosing quotation marks:

   (a) Without Quotes - If the first character of an alphanumeric value is not a quote (") character, the entire value, consisting of all characters up to a comma separator or the end of card, is read. If the receiving alpha variable is dimensioned to contain fewer bytes than there are characters in the value, the variable is filled and additional characters in that value are ignored. If the receiving variable is larger than the value read, remaining bytes in the variable are filled with space characters. Leading and trailing spaces (unpunched, unmarked columns) are not read; spaces embedded within a value are read, however, and are stored as part of the value.

(b) With Quotes - If the first character of an alphanumeric value is a quote (") character, all characters following the quote character up to a second quote character are read and interpreted as belonging to the same value. In this case, commas within the quotes do not act as separators, but are read as data characters. Embedded spaces (unpunched, unmarked columns) as well as leading and trailing spaces inside the quotes are read as parts of the value. Leading and trailing spaces outside the quotes are ignored. The quotes are not considered part of the data.

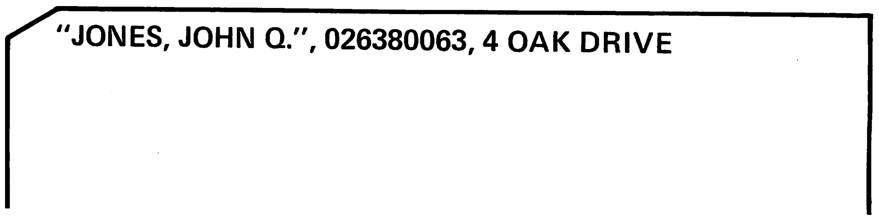


Figure 4-2.
Alphanumeric Values on a Hollerith Card With and Without Quotes. (Because the first value is enclosed in quotes, the embedded comma does not act as a data separator.)

5. Numeric data values can be entered in free format in any numeric representation legal in a Wang system (e.g., 2.4, -973, 21.2 E-07, etc.) The conventions governing numeric free format are as follows:

   (a) Optional plus (+) or minus (-) sign. If no sign is specified, the number is assumed positive. For negative values, a minus (-) sign must be specified.

   (b) From one to 13 digits to represent the value of a number (e.g., 400125), or the significant digits of a floating point number, with or without a decimal point (e.g., 1.45796E07, 240006E-11).

   (c) Optional two-digit exponent of number (E$\pm$XX).

   Leading, trailing, and embedded space characters in a numeric value are ignored.

6. Each valid data character is read and automatically converted from Hollerith to ASCII. Hollerith codes which cannot be converted into legal ASCII codes are automatically read as ASCII exclamation ('!') characters (HEX 21). For numeric values, an illegal character causes the value in which it appears, and all succeeding values, to be ignored. An error message is generated, and program execution is automatically terminated (unless the program contains an ON ERROR GOTO statement). Values preceding the faulty value in the DATALOAD argument list are read and stored. Illegal characters are stored into alphanumeric variables without incident.

7. Use of the Hollerith codes whose ASCII equivalents are Carriage Return (multi-punch rows 12, 9, 8, 5) and X-OFF (multi-punch rows 11, 9, 3) characters is illegal in this mode. (A Carriage Return is not required to terminate a card.)

8. The last (80th) column on each card is reserved in this mode for a special character, the ampersand ('&'), and should never be used for data. The special ampersand character ('&') is used to indicate continuation of an alpha or numeric value to a second card, and is discussed in Section 4.4, entitled "Continuation of a Data Value". Any character other than an '&' punched or marked in column 80 is read as an ASCII exclamation ('!') character. Except in the special case of the ampersand character, therefore, column 80 should be left blank.

9. Blank cards are ignored by the system in this reading mode.

4.4 CONTINUATION OF A DATA VALUE

In most cases, a single punched card contains one or more whole data values, appropriately separated by commas. Occasionally, however, it may be convenient to extend a single numeric or alphanumeric data value from one card onto a second card. An ampersand ('&') character marked or punched in the last (80th) column of the first card is used to indicate continuation. The maximum length of a numeric value is 19 characters (including maximum 13 digits, optional sign of number, optional exponent, and optional sign of exponent). The maximum length of an alphanumeric value is 64 characters.

1. Numeric values - To continue a numeric value from one card to another, simply punch an ampersand ('&') character in column 80 of the first card, and continue the data value in column 1 on the second card. For example, the numeric value 8001624E07 is continued from one card to a second in the following way:

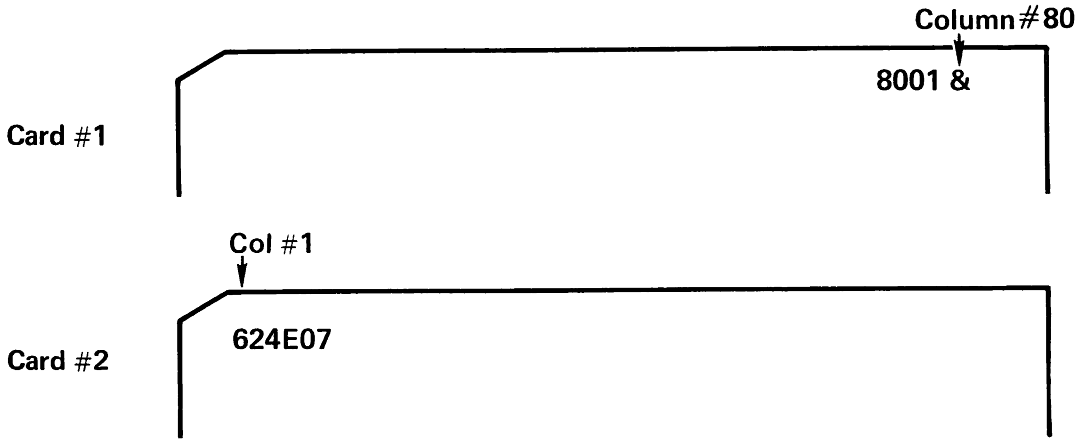


Figure 4-3.
Continuing a Numeric Value from One Hollerith Card to a Second.

2. Alphanumeric Values - To continue an alphanumeric value from one card to another, an ampersand ('&') character must be punched in column 80 of the first card. The following rules should be followed when continuing an alpha value:

   (a) The character string being continued should be enclosed in quotes.

   (b) The first column of the second card should contain a quote (") character. This is to ensure that spaces and commas embedded within the value are preserved when the second card is read.

   For example, the alphanumeric value "Jones, John" is continued from one card to another in the following way:

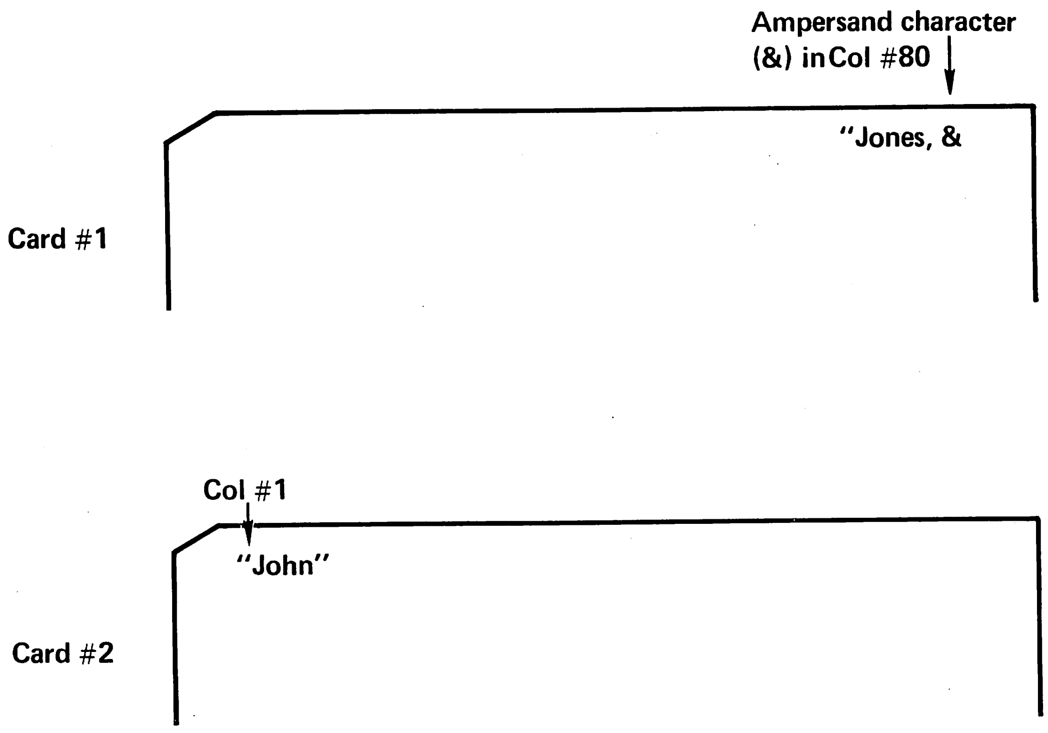


Figure 4-4.
Continuing an Alphanumeric Value from One Hollerith Data Card to a Second Card.

Note that if the quote (") character in column 1 of the second card is omitted, leading spaces on the second card (i.e., spaces actually embedded within the data value) are ignored, and commas within the data value on card #2 are interpreted as separators rather than data.

4.5 TESTING FOR THE END-OF-FILE

Typically, all data cards in a data deck have the same format - that is, the same number of data fields, in the same order. Each card (or a fixed number of sequential cards) is read with a DATALOAD statement (address 628), the data is processed, and the program loops back to read the next card or sequence of cards. It is convenient in this case to be able to test for the end-of-file (i.e., no more cards to be read), since it is not always possible to know beforehand exactly how many cards are to be read. For this purpose, the user may design his own end-of-file card containing in the first field a special data value which would never appear as normal data. This special value is tested for in the program each time a DATALOAD statement is executed. Note that all remaining fields on the End-Of-File Card must be filled with dummy data values to satisfy the DATALOAD argument list. See Figure 4-5.

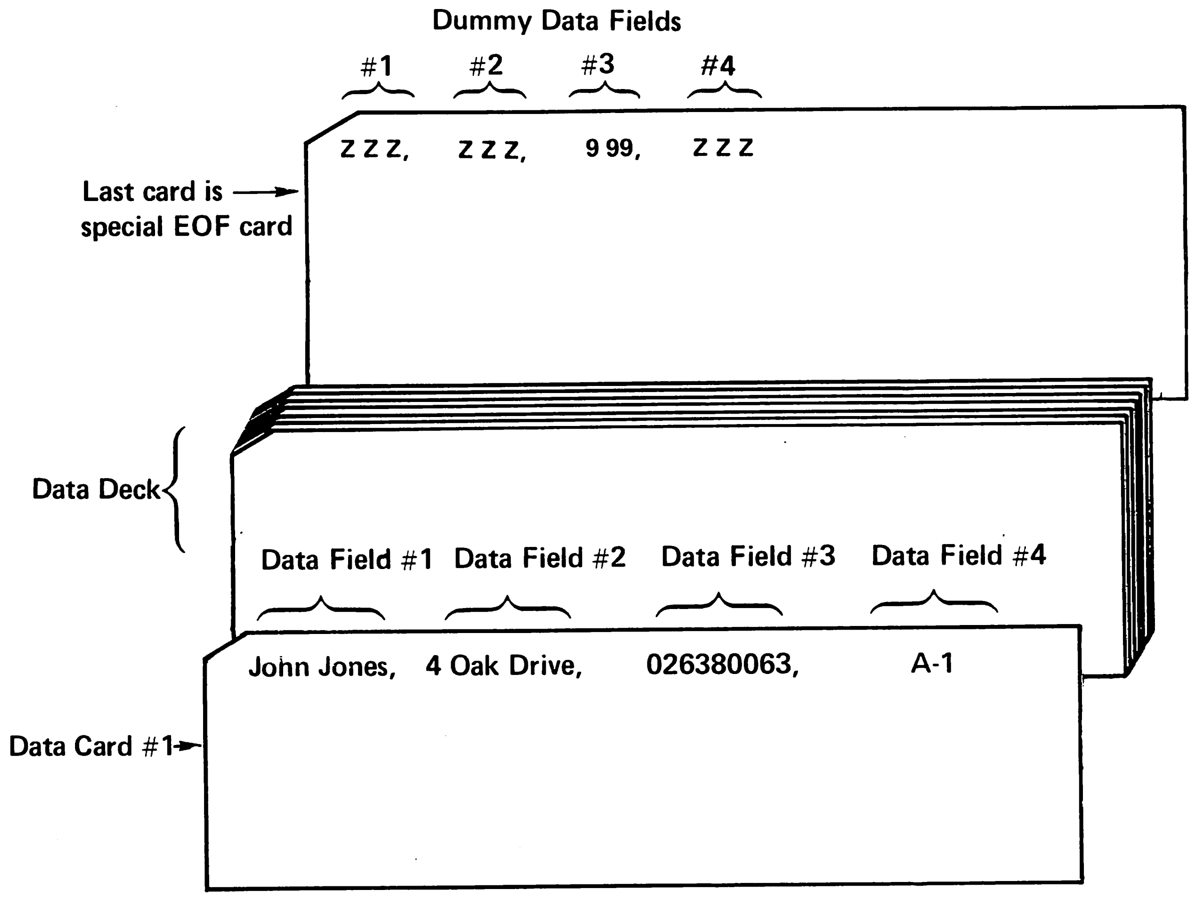

Dummy Data Fields
#1
#2
#3
#4
ZZZ, ZZZ, 999, ZZZ
Last card is special EOF card
Data Deck
Data Field #1 Data Field #2 Data Field #3 Data Field #4
John Jones, 4 Oak Drive, 026380063, A-1
Data Card #1

Figure 4-5 .
Typical Data Deck with EOF Card Containing Dummy Data in All Fields.

Note that the dummy data fields must correspond in number and type (alpha or numeric) to the legitimate data fields. As each card is read, the controlling program checks for the presence of the first dummy data value. If the dummy value is not found, the card contains legitimate data, and normal processing resumes. If the dummy value is detected, the DATALOAD operation is terminated, and the program branches to another routine.

Example 4-3: Reading Hollerith Data Values and Testing for the End-of-File (DATALOAD, Address 628)

```
50 DATALOAD/628, A$, B$, N, F$
60 IF A$ = "ZZZ" THEN 160
.
.
.   (Process data)
.
.
.
150 GOTO 50
160 STOP
```

This example might be used to read the data deck in Figure 4-5, and test for an end-of-file condition. Each data card holds four data fields: data fields #1 and #2 are alphanumeric, and are read into alpha variables A$ and B$, respectively; data field #3, a numeric field, is read into numeric variable N; the fourth and last field, an alpha field, is read into F$. As each card is read, the system tests for end-of-file by checking for the dummy field "ZZZ" in A$. If the dummy value is not detected, the data just read is processed until, at statement 150, the program is instructed to loop back and read in a new card. If the EOF card is read, the four dummy data fields are read into variables A$, B$, N, and F$, respectively, and the program skips down to statement 160 and stops.

4.6 <u>THE SPECIAL "RESET" CARD</u>

A number of possible error conditions, such as a logical error in the main program, a miscount of the number of data cards in the data deck, or a missing end-of-file card, might cause the DATALOAD routine to attempt to continue reading data cards beyond the last card in the data deck. If the data deck is followed by a program deck (as in a batched job stream or a series of program overlays), the system attempts to read the first program cards as data. If the data deck is the last or only deck in the input hopper, the system hangs up awaiting more cards. Both of these undesirable developments can be avoided by including a special RESET card at the end of the data deck.

The RESET card is created by multi-punching rows 2, 3, and 4 (i.e., multi-punching integers 2, 3, and 4) in the first card column. The remainder of the card should be left blank. When the RESET card is read, it produces a system RESET condition (identical to the condition generated by touching the RESET button on the keyboard). The DATALOAD operation is terminated, and program execution resumes at the next line in the controlling program. In this way, the possibility of reading into the next program deck, or hanging up with no more cards to read, is obviated. If there are no errors or other problems with the DATALOAD routine, the RESET condition does not adversely affect normal program execution.

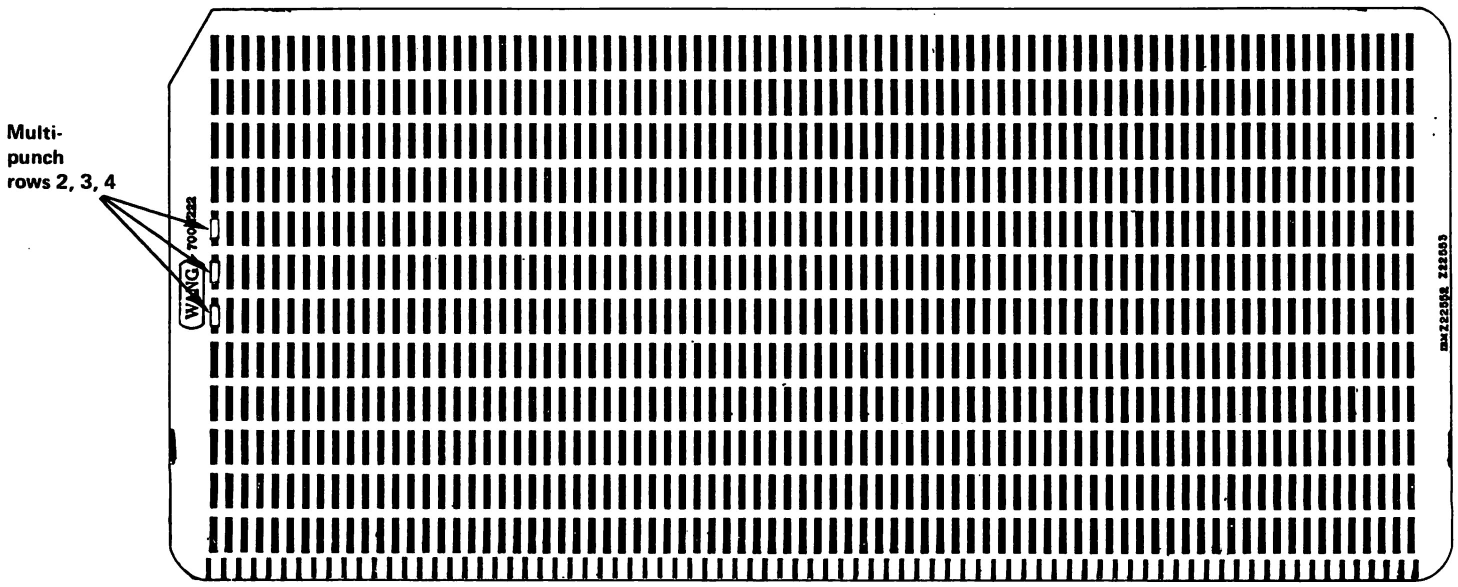


Figure 4-6. Special 'RESET' Card For DATALOAD Data Decks.

CHAPTER 5
READING HOLLERITH DATA VALUES (INPUT, ADDRESS 62B)

5.1 INTRODUCTION

Hollerith data values on punched or mark sense cards can be read with an INPUT statement, if INPUT operations have been selected to the appropriate card reader address, 62B. Once INPUT operations have been selected to the card reader with a SELECT INPUT statement, no input from the keyboard is possible. INPUT operations must be selected to the address for loading Hollerith data cards with the following SELECT INPUT statement:

```
SELECT INPUT 62B
```

Like DATALOAD, INPUT is not recommended for serious data processing operations, principally because it does not provide the capability for adequate data control and verification. In particular, the use of INPUT poses potential problems in the handling of erroneous numeric data, since INPUT merely ignores an invalid numeric, and reads the next data card. There are, however, several legitimate uses for INPUT:

1. For System 2200A and 2200S, and WCS/10 owners who do not have access to any other modes of entering data. If INPUT is the only available means of reading data from cards, it is recommended that all data, numeric as well as alphanumeric, be read initially into alphanumeric variables. Numeric data can be converted and stored in numeric variables after it has been checked for validity. See Section 5.3 below for an example of how this can be done. (This technique is not, however, possible on a System 2200A.)

2. For testing and debugging a program which normally requires a great deal of data entry from the keyboard. Programs designed to accept input from the keyboard can be altered with the addition of a SELECT INPUT 62B statement to read data from cards instead. A single data deck can be used repeatedly to check out the program, possibly saving considerable time in keyboard entry.

3. For educational purposes, as an introduction to card reader operations. Because of its similarity to keyboard entry operations, INPUT can serve as a relatively simple and familiar introductory method of entering data from cards.

5.2 DATA CARD FORMAT FOR 'INPUT'

The required format of Hollerith data values read from cards with INPUT is identical to the required format of data values entered from the keyboard in response to an INPUT request. The INPUT format is also identical to the format required by the DATALOAD statement, discussed in Chapter 4. See Section 4.3 for a detailed list of format requirements. In brief, the format requirements are:

1. Multiple values on the same card must be separated by commas. However, a comma must not be placed after the last value on a card.

2. Alphanumeric values may or may not be enclosed in quotation marks. Embedded commas in alpha values which are enclosed in quotes are interpreted as data rather than as data separators.

3. Numeric values may be punched in any free-format legal in a Wang system.

4. Numeric values can be read into alphanumeric receiving variables, but alphanumeric values cannot be stored in numeric variables.

5. The 80th card column is reserved in this mode for a special character, the ampersand ('&'), and cannot be used for any other purpose. The ampersand indicates continuation of a data value (see Section 5.6). If an ampersand is not used, the 80th column must be left blank.

6. Carriage Return and X-OFF characters are illegal. (All illegal characters are automatically decoded as exclamation ('!') characters.) Blank cards are ignored.

5.3 READING DATA VALUES FROM CARDS WITH 'INPUT'

```
GENERAL FORM:  INPUT ["character string",] variable [,variable...]
```

Purpose:

Once the card reader has been selected for INPUT operations with a SELECT INPUT 62B statement, the reader functions like a keyboard, reading one or more data cards in response to each INPUT request from the controlling program. As each input request is executed, the system displays a question mark ('?'), preceded by the optional character string. Data values are read from cards and sequentially assigned to the receiving variables in the INPUT argument list. Both numeric and alphanumeric values can be stored in alphanumeric variables. However, only legitimate BASIC numbers in free-format may be read into numeric variables. Otherwise, an ERROR 29 (Illegal Data Format) is generated, and the erroneous value is skipped. Each value may be marked or punched on a separate card, or multiple values may be placed on a single card, provided the values are separated by commas. If the argument list is not satisfied by a single card, additional cards are read until all receiving variables in the argument list have been assigned values. Note, however, that each time the INPUT statement is executed, it automatically begins reading with the next card in the input hopper, even if there are remaining unread data values on the previous card. It is not therefore possible to use two or more INPUT statements to read several data values from a single card. Unread data values on the last card read by an INPUT statement are lost.

The INPUT statement operates in a manner substantially similar to that of DATALOAD, with one important difference. The DATALOAD statement displays an error message and terminates program execution when it encounters a format error in a numeric data value. With INPUT, however, a numeric format error generates an error code but does not terminate program execution. Instead, the erroneous value is ignored (along with any remaining values on the same card), and the INPUT request is repeated. The next data card is automatically read, and values from that card are sequentially assigned to the remaining unfilled variables in the INPUT argument list. INPUT is therefore somewhat unpredictable in its handling of erroneous data values.

Note that a program which utilizes INPUT to read data from the card reader can be readily altered to accept data from the keyboard instead, simply by omitting the initial SELECT INPUT 62B statement (i.e., by not selecting the card reader for INPUT operations).

Example 5-1: Reading Hollerith Data Values with INPUT (Address 62B)

```
 10 DIM A$3
  .
  .
  .
 50 SELECT INPUT 62B
 60 INPUT A$
 70 IF A$ = "999" THEN 200
  .
  .  (Process data read at line 60)
  .
  .
190 GOTO 60
200 SELECT INPUT 001
210 STOP
```

In this example, the card reader is first selected for INPUT operations with the Hollerith data card address for INPUT (62B). A single value is then read into alpha variable A$. Immediately following the read, A$ is checked for the value "999". The dummy value "999" is used in this case to signal the end-of-file (that is, no more cards in the deck). A card containing "999" must be included as the last card in the data deck read by this routine. If a value other than "999" is read, the program drops through to process the data, and, at line 190, loops back to read in the next value. If "999" is read, the program branches to line 200, where INPUT is selected back to address 001 (the keyboard), and the program stops.

5.4 SPECIAL TECHNIQUES FOR USING 'INPUT' TO PROCESS NUMERIC DATA

Although the INPUT statement is not recommended for serious data processing work, it is the only statement available on the System 2200A and standard versions of the System 2200S and WCS/10. In cases where INPUT must be used for data processing on a System 2200S or WCS/10 which does not have Option-22 or -23, it is strongly suggested that alphanumeric variables be utilized to receive both numeric and alphanumeric values. (This technique is not practical for a System 2200A, which lacks the NUM and CONVERT functions used to validate and convert numeric data from alphanumeric to numeric format.) The use of alphanumeric receiving variables for numeric values makes the procedure for handling erroneous numeric values more predictable, since the system will not in that case automatically ignore an erroneous numeric and proceed to read in the next card. Because no special format is required for alphanumeric data (even non-ASCII characters are automatically converted to ASCII exclamation characters, and stored as valid data), it is not possible for the system to detect a data format error in an alphanumeric value.

Once the numeric value has been stored in an alpha variable, it can be tested for validity under program control using the NUM function. If it is found to be valid, it may be converted to numeric format with the CONVERT function, and processed in the usual way. If it is found to be invalid, the system can be instructed to print out an error message to the operator. In this way, the operator always knows which value is erroneous, and he is assured that any remaining data values in the data deck will not be affected by the detection of an erroneous value.

A numeric value punched or marked in Hollerith code in free-format may consist of a maximum of 19 characters, arranged in the following order:

Optional sign of number.
Maximum 13 digits in number.
Optional decimal point in number.
Optional 'E' character identifying exponent.
Optional sign of exponent.
Optional two-digit exponent.

For example, the following number consists of the maximum 19 characters:

```
-123456789.1234E-07
```

When this number is read into a numeric variable, it is automatically converted to the system's internal numeric format, and occupies only eight bytes. When it is read into an alphanumeric variable, however, each character occupies a single byte; a 19-byte alpha variable is therefore required. Once the numeric value is stored in an alpha variable, it is verified with a NUM function. NUM determines the number of legitimate numeric characters (the plus (+) and minus (-) signs, decimal point, digits, the 'E' character, and spaces all are considered legitimate numeric characters) in the alpha variable. If the numeric value read consists of fewer than 19 characters (as it would in most cases), INPUT automatically pads the remaining bytes of the alpha variable with trailing spaces, which are accepted as numeric characters by NUM. Thus, the number of numeric characters returned by NUM should be 19 in every case; if fewer than 19 numerics are detected, at least one character is non-numeric, and the value should be rejected. Example 5-2 below illustrates a routine to test and convert numeric data entered with INPUT.

Example 5-2: Testing and Converting Numeric Data Entered Via INPUT (System 2200S and WCS/10)

```
10 DIM A$19, B$24, C$9
20 SELECT INPUT 62B
30 INPUT A$, B$, C$
40 IF  NUM(A$)  <>  19 THEN 150
50 CONVERT A$ TO N
.
.  (Normal Processing)
.
140 GOTO 30
150 STOP "ERRONEOUS VALUE"
```

In this example, three Hollerith data values are read from cards with the INPUT statement at line 30. All three values may be on a single card (if they are separated by commas), or they may be on three separate cards. The first value read is numeric, the second and third are alphanumeric. All three, however, are read into alphanumeric variables. Alpha variable A$ is designated to receive the numeric value. It is dimensioned to 19 bytes in length so that it can contain the maximum possible number of characters in a single numeric value. Following the INPUT request, NUM is used to determine the number of valid numeric characters in A$. If there are fewer than 19 numeric characters (trailing spaces are counted as numeric characters), at least one of the characters is illegal, and the program branches to an error routine (in this case, an error message is displayed at line 150). If A$ contains 19 valid numeric characters, its value is converted into numeric format by the CONVERT statement at line 50, and normal processing continues.

5.5 CONTINUING DATA VALUES AND TESTING FOR THE END-OF-FILE

Continuation of a Data Value

A single data value may be continued onto one or more cards for reading with INPUT. In this case, an ampersand ('&') character is punched in the 80th column of each intermediate card containing the value (but not on the last card containing the value). For a more complete discussion of the continuation feature, refer to Chapter 4, Section 4.4, "Continuation of a Data Value".

Testing for the End-of-File

In cases where a series of data cards are read with an INPUT loop, it is generally desirable to test for a special end-of-file card indicating that all data cards in the deck have been read. Such a card is designed by the user to contain dummy data values in all fields. As each data card is read, the program checks to see whether one of the dummy data values was received. When the designated dummy value is read, the system knows that all data cards have been processed, and terminates the INPUT loop. The logic involved in creating and testing for an end-of-file card is the same for DATALOAD operations as for INPUT operations, and is described in detail in Chapter 4, Section 4.5, "Testing for the End-of-File".

5.6 THE SPECIAL 'RESET' CARD

In programs which utilize an INPUT routine to read data cards, there is always the possibility that a problem such as a logical error in the INPUT routine or a miscount of the number of data cards in the data deck may cause the system to continue attempting to read data cards after the data deck has been exhausted. If the data deck is part of a larger job stream for batch processing, or is inserted between overlays, the system will attempt to read program or command cards from the following program deck as data. If the data deck is the last or only deck in the input hopper, the system hangs up awaiting additional cards. Both of these situations can be prevented by inclusion of a special RESET card as the last card in the data deck.

The special RESET card is created by multi-punching rows 2, 3, and 4 (i.e., multi-punch integers 2, 3, and 4) in the first column. All remaining columns are left blank. When the RESET card is read, it generates a System RESET condition (that is, it has the same effect as touching the RESET key on the keyboard). The RESET condition returns control to the currently selected Console Input device. If the card reader is selected for Console Input, the next program deck is loaded and processed as if no error had occurred. In this way, the system is prevented from either hanging up or reading into a subsequent program deck in the event of a card miscount or logical error in the INPUT routine. Note that if no such error exists, the RESET card does not interfere with normal processing.

The RESET card is illustrated at the end of Chapter 4.

CHAPTER 6
READING HOLLERITH DATA CARD IMAGES (DATALOAD BT, ADDRESS 629)

6.1 INTRODUCTION

In Mode #3, the statement DATALOAD BT is used with a device address of 629 to read complete Hollerith data card images. For the Model 2234A, the cards read must be standard 80-column cards punched in Hollerith code. For the Model 2244A, the cards may be marked or punched in Hollerith code, and may contain fewer than 80 columns (if they also have index marks).

The principal value of Mode #3 is that it permits the programmer to read Hollerith cards in any data format, and provides him with total control over card format error conditions which may arise as the data is read. In Mode #3, the entire 80 columns of data are read from each card and automatically converted from Hollerith to ASCII. No check is made by the system for illegal characters or data formats, and no distinction is made between data characters and control characters; the entire card is simply read into a single alpha array. The 80 characters read into memory therefore constitute an exact ASCII "image" of the data punched (or marked) on the card. Once a card image has been read, the validity of individual data fields can be tested, and the data values transferred to individual variables under program control. Various BASIC instructions (such as the STR and NUM functions, and the CONVERT statement) are available on most systems for this purpose. Additionally, several General I/O statements, particularly $UNPACK, provide extremely efficient techniques for verifying and converting data from a data card image. Data conversion programming techniques are discussed in Chapter 17.

In addition to the 80 characters of data read from each card in Mode #3, the card reader itself generates two special control characters, a LENGTH code and an ERROR code. The 81st character transmitted is a LENGTH code, which contains a binary count of the number of data columns read from the card. The 82nd character transmitted is the ERROR code. Individual bits in the error code signify different types of error conditions.

6.2 READING HOLLERITH DATA CARD IMAGES

General Form: DATALOAD BT (N=82) $\begin{bmatrix}\text{/629,}\\ \text{\#n,}\end{bmatrix}$ alpha array designator

Where:

(N=82) =	The number of characters to be received for each card read. For Hollerith card images (address 629), a total of 82 characters are received for each card (80 data characters and two control characters). If the card has fewer than 80 columns, the reader automatically "pads" the remaining unread characters up to 80 with HEX(FF) characters.
/629 =	The device address which designates the card reader as the device from which data is to be read, and also indicates the type of code conversion which is to be performed and the data format which is to be expected. Address 629 initiates the reading of a complete 80-character card image, and causes an automatic Hollerith-to-ASCII conversion of all data read.
#n =	A file number to which the device address has been assigned in a SELECT statement (n is an integer from 1 to 6). If neither a device address nor a file number is specified, the address of the default tape device is used. Ordinarily, this is the console tape cassette drive, address 10A. However, address 629 can be designated as the Console Tape default address with a SELECT TAPE 629 statement. Following execution of such a statement, address 629 is used if no address or file number is specified in the DATALOAD BT statement.
alpha array designator =	An alphanumeric array name followed by closed parentheses (e.g., A$(), F$()).

Purpose:

The DATALOAD BT statement with a device address of 629 initiates the reading and conversion of a complete 80-character Hollerith card image. Each column is translated from Hollerith code into its ASCII equivalent, and stored in the receiving alphanumeric array. In this mode, the system always expects to receive exactly 80 data characters for each card read (blank columns on a card are read as ASCII space characters). A card which contains timing marks may have fewer than 80 columns, however. In this case, the card reader itself generates HEX(FF) characters for all unread characters up to 80, so that exactly 80 data characters are transmitted. In addition, the card reader generates a LENGTH code and an ERROR code, and transmits them as the 81st and 82nd characters for each card read. The receiving alpha array must therefore be dimensioned to hold at least 82 characters, and the number of characters read in every case must be 82 (N=82). If an entire card or any portion of a card cannot be read (due to a card jam or other reader malfunction), 80 HEX(FF) codes are transmitted, and the ERROR Code identifies the source of the difficulty.

Example 6-1: Reading Hollerith Data Card Images
(DATALOAD BT, Address 629)

```
50 DIM A$(3)40
.
.
.
500 DATALOAD BT (N=82)/629,A$()
```

In this example, the receiving array is dimensioned to contain three elements, each 40 bytes in length. This arrangement is convenient because it effectively segregates the two control bytes in A$(3), while the 80 characters of data are stored in A$(1) - A$(2). The first two characters in A$(3) are the 81st and 82nd characters received; the remaining 28 bytes of A$(3) retain their original values (that is, they are unaffected by the DATALOAD BT operation). Note that DATALOAD BT continues reading only until the specified 82 characters (N=82) have been received, regardless of whether the receiving array has been completely filled.

6.3 HOLLERITH CARD IMAGE CONVENTIONS

The following conventions apply to Mode #3 reading operations:

1. For each card read, 82 characters are transmitted to the system (80 characters of data from the card, plus two special control characters). Special mark sense Hollerith cards read by the Model 2244A may have fewer than 80 columns, if the cards contain index marks.

2. The receiving alphanumeric array in the DATALOAD BT statement should be dimensioned to hold at least 82 characters of data.

3. The 629 address specifies that 80 characters of data be read and converted from Hollerith to ASCII. Illegal Hollerith characters are converted to ASCII exclamation ('!') characters (HEX(21)), and stored in the receiving array. (In addition, an error bit is set in the ERROR code character indicating that an illegal character was read.)

4. The 81st character read for each card is the LENGTH code, a binary count of the number of columns read from the card.

5. The 82nd character read for each card is the ERROR code, an 8-bit character whose individual bits represent different types of card reader error conditions.

6. With the Model 2234A, only standard 80-column punched cards can be read. A correct read, therefore, always results in the reading of 80 data characters. Thus the LENGTH code should always be HEX(50) (decimal equivalent, 80).

7. With the Model 2244A, cards having fewer than 80 columns may be read if they contain index marks. A valid read in this case results in fewer than 80 characters being read, and the remaining bytes of the receiving alpha array are filled with HEX(FF) characters up to 80. The LENGTH code then is set equal to the number of columns actually read.

8. If, because of a card reader error condition, the reader cannot read the next card, 80 HEX(FF) data characters are returned, with an appropriate ERROR code, and a LENGTH code of HEX(00). For this reason, it is necessary that the card reader be in a RESET condition before an attempt is made to read data in Mode #3. The programmer can, however, design an input loop which continually attempts to read a card, checking for the 'NOT READY' error code after each attempt. (If the reader is not in a RESET condition when an attempt is made to read a card image, the 'NOT READY' bit is automatically set in the ERROR code.) If the card reader is not ready, suitable action can be taken by the program (e.g., display a message to the operator). This is one example of the total control provided in the card image reading mode over all types of error conditions. A complete discussion of programming techniques for card image reading is found in Chapter 17.

9. Unlike Modes #1 (loading Hollerith programs) and #2 (reading Hollerith data values), Mode #3 attaches no special significance to the 80th card column. Data may therefore be recorded in all 80 columns on cards read under Mode #3.

10. Blank cards are not ignored by the System in Mode #3; instead, 80 space characters (HEX(20)) are read for each blank card.

6.4 THE LENGTH CODE

The 81st character transmitted for each card in Mode #3 is the LENGTH code. In the normal case, 80 columns are read, and the LENGTH code is equal to HEX(50), the hexadecimal equivalence of decimal 80. In special cases, fewer than 80 columns are read. The special cases include:

1. Fewer than 80 columns on the card. This situation cannot occur on cards read by the Model 2234A, since only standard 80-column punched cards can be read by the 2234A. Cards read by the Model 2244A may, however, have fewer than 80 columns if they also contain index marks. In that case, the LENGTH code indicates the number of columns actually read from each card.

2. Short Card. The data card may be physically shorter than a standard 80-column card. The deck should be examined for mutilated or non-standard cards. The card reader cannot process non-standard cards.

3. Reader Failure. The reading sensors may experience a failure before an entire card has been read or a card may become jammed in the reading station. Check the card track and reading station for jammed cards. If the problem appears to be a sensor failure, call your Wang Service Representative.

When cards are read in Mode #3, the LENGTH code should be checked for each card image to ensure that all expected data columns were read. In all cases, if fewer than 80 columns are read, the remaining unread columns up to 80 are received as HEX(FF) characters.

6.5 THE ERROR CODE

The 82nd character received for Hollerith card images is the ERROR code. The ERROR code is an 8-bit code, each individual bit of which represents a unique card reader error condition. The eight bit-positions are identified by the following eight HEX codes:

Bit Position	8	7	6	5	4	3	2	1
HEX Code	80	40	20	10	08	04	02	01
Error Indicator	1	1	1	1	1	1	1	1

Figure 6-1

The table below lists the eight bit-positions and the error condition associated with each:

Table 6-1. Card Reader Error Conditions.

BIT	VALUE	MEANING
80	1	Machine Not Ready
40	1	Hopper Empty
20	1	Stacker Full
10	1	Pick Check (A motion check: the picker failed to engage the card after six consecutive attempts.)
08	1	Read Alert (Generally, a card has a tear or marking on the leading edge, a photoelectric sensor has failed, or the read head is dirty.)
04	1	Less than expected data received (Short card, jam, read alert, or special card with fewer than 80 columns.)
02	1	Invalid ASCII conversion converted to ASCII exclamation ('!') character.
01	1	Invalid Look-Ahead operation (Attempted to perform Hollerith card read following Binary Look-Ahead, or Binary card read following Hollerith Look-Ahead.)

Note that the presence of a particular error bit may be anticipated under certain conditions. Custom-designed mark sense cards having fewer than 80 columns can be read in Mode #4 (Binary Card image), for example; in such cases, the 04-bit ('Less than Expected Data') is set in the ERROR code for each card. Because the card is known to contain fewer than 80 columns in this case, however, no corrective action is required. The LENGTH code should be checked to ensure that the expected number of columns actually were read.

In general, both the LENGTH and ERROR codes should be checked following every card read operation in Mode #3. To perform such checks most efficiently, and to ensure that the two control characters are never mistakenly interpreted as data, it is good practice to isolate the LENGTH and ERROR codes in the last element of the receiving array.

Example 6-2: Checking the ERROR Code (Hollerith Card Image)

```
60 DIM A$(3)40, N$1
.
.
.
.
100 DATALOAD BT (N=82)/629, A$()
110 IF STR(A$(3),2,1) <> HEX(00) THEN 350
.
.
. (Normal processing)
.
.
340 GOTO 100
350 N$ = STR(A$(3),2,1)
360 AND (N$,80):REM CHECK FOR 'MACHINE NOT READY'
370 IF N$ <> HEX(00) THEN 500
.
. (Test for other error conditions)
.
500 STOP "MACHINE NOT READY"
```

This example illustrates a typical processing routine which consists of two sections, a normal processing routine (lines 100-340) and an error processing routine (lines 350-500). At line 100, the 82-character image is read into array A$(). The 80 data characters are stored in the first two array elements, A$(1) - A$(2), while the two control characters are stored in the last element, A$(3). Line 110 performs a test on the ERROR code, which is the second character in A$(3). If the code is zero, no errors were detected, and the program proceeds to normal processing. If, however, the code is other than HEX(00), then at least one error bit must be on, and the program branches down to the error routine beginning at line 350. At line 350, the ERROR code is stored into N$, a one-byte working variable. Next, the AND function is used to check for the presence of an 80 bit (indicating 'Machine Not Ready'). If the 80 bit is on (line 370), the program branches to line 400, stops, and prints the appropriate error message. If the 80 bit is off, the routine drops through to line 380, where it may test for other possible error conditions.

NOTE:

A more detailed discussion of programming techniques for checking the LENGTH and ERROR codes following a card image read is presented in Chapter 17.

CHAPTER 7
HOLLERITH LOOK-AHEAD MODE (DATASAVE BT, ADDRESS 42E)

7.1 INTRODUCTION

Mode #5 is the Hollerith "Look-Ahead" mode. Look-Ahead mode enables the card reader to feed a card through the reading station, convert the data to ASCII, and store it in the card reader buffer. During this entire procedure, the CPU may be involved with other processing such as operating on data read in from the previous card. By thus overlapping the card reading/code conversion operations with CPU internal processing operations, the total throughput time for processing Hollerith data decks can be significantly reduced in many data processing applications.

7.2 OPERATION OF THE LOOK-AHEAD MODE

Mode #5, the Hollerith Look-Ahead mode, utilizes the BASIC statement DATASAVE BT and the special device address '42E'. In this mode, the card reader reads an 80-character Hollerith card image, converts each character into its ASCII equivalent, and holds the data in the card reader output buffer, where it waits to be transferred into memory. Illegal characters are read as ASCII exclamation ('!') characters. If fewer than 80 characters are read, the output buffer is padded with HEX(FF) characters up to 80. Following the read, two special characters, a LENGTH code and an ERROR code, are generated by the card reader and stored as the 81st and 82nd characters in the buffer.

Note that the Look-Ahead mode is used only to read a card and temporarily store the data in the card reader output buffer. This mode cannot be used to transmit the data into memory for processing. Data is transmitted from the card reader output buffer into memory only in one of the two legitimate Hollerith data reading modes, Mode #2 (DATALOAD with address 628), or Mode #3 (DATALOAD BT with address 629). Although Hollerith Look-Ahead Mode is employed for both types of Hollerith data reading (Hollerith data values and Hollerith card images), the reading mode selected always determines the amount and type of data actually read into memory. If the data is read in Mode #3 (Hollerith card images), all 82 characters are transferred from the card reader's output buffer into the receiving array in memory. If, on the other hand, Mode #2 (Hollerith data values) is utilized to read the data, only legitimate data values (separated by ASCII comma characters) are read from the output buffer into the receiving argument list in memory. The 80th character is interpreted as a control character, and the 81st and 82nd characters are ignored completely. In short, data read in Mode #2 (DATALOAD, address 628) must conform to the conventions spelled out in Section 4.3 for Hollerith data values, regardless of whether the Look-Ahead mode is or is not utilized.

7.3 CARD READING WITH LOOK-AHEAD

GENERAL FORM: DATASAVE BT $\begin{bmatrix}\text{/42E,}\\ \text{\#n,}\end{bmatrix}$ alpha variable

Where:

/42E =	The special card reader device address which specifies Hollerith Look-Ahead operations. Address '42E' causes the card reader to feed in the next card from the input hopper, convert the data from Hollerith to ASCII, and hold the converted data in the card reader output buffer awaiting transmission to the system.
#n =	A file number to which address 42E has been assigned in a SELECT statement ('n' must be an integer from 1 to 6).
	If neither a device address nor a file number is specified, the address of the device currently designated as Console Tape device is used. Normally, the Console Tape device is the console tape cassette drive in the CRT (address 10A). However, address 42E could be designed as the default address with a SELECT TAPE 42E statement. In this case, all subsequent card reader statements which do not specify a device address or file number automatically default to address 42E.
alpha variable =	A "dummy" alphanumeric variable included to satisfy general format requirements for the DATASAVE BT statement, but not used in the Look-Ahead operation. (Note that the operation is somewhat more efficient if the dummy alpha variable is dimensioned to the minimum length of one byte.)

Purpose:

The DATASAVE BT statement with a device address of 42E initiates the reading of one card into the card reader buffer, and converts the data from Hollerith to ASCII. Illegal characters are translated as ASCII '!' characters. LENGTH and ERROR codes are also generated, and can be obtained if a Hollerith card image (DATALOAD BT, address 629) is subsequently read. The Look-Ahead operation in effect constitutes the first stage of a reading operation in Modes #2 and #3. The data cannot actually be transmitted from the card reader buffer into memory, however, until a Mode #2 (DATALOAD, address 628) or Mode #3 (DATALOAD BT, address 629) statement is executed. There are no timing restrictions governing when a DATALOAD or DATALOAD BT statement may be executed following a Look-Ahead operation.

Example 7-1: Processing Hollerith Data Values with Look-Ahead (DATASAVE BT, Address 42E)

```
10 DIM F$1                     (Dimension dummy variable to one byte
                                for efficient operation)
.
.

200 DATALOAD/628, A$, B$, N    (Read a card)

210 IF A$ = "ZZZ" THEN 460     (Test for last card)

220 DATASAVE BT/42E, F$        (Initiate reading the next card)
.
.
.   (Process data read at line 200; simultaneously, the next card is
.   read and converted to ASCII by the card reader.)
.
450 GOTO 200                   (Loop back to get converted data from
                                card reader buffer)
460 STOP                       (Stop if last card is read)
```

In this example, the Hollerith Look-Ahead mode is used to speed up processing of Hollerith data cards. Data values from the first card are read at line 200. At line 220, the card reader is instructed to feed in the next card, convert from Hollerith to ASCII, and store in the card reader buffer. While this procedure takes place, data processing continues in the CPU at line 230. When processing is completed at line 450, the program loops back to line 200 to read in the next set of values, already converted to ASCII and waiting in the output buffer of the card reader. Note that F$ in line 220 is a dummy variable which is dimensioned to the minimum length of one byte for more efficient processing. Data read at line 200 must conform to the conventions for Hollerith data values listed in Section 4.3.

Example 7-2: Processing Hollerith Card Images With Look-Ahead
(DATASAVE BT, Address 42E)

```
70 DIM A$(3)40, F$1                          (Dimension receiving alpha array
                                              and dummy variable)
.
.
.
200 DATALOAD BT (N=82)/629, A$()              (Read a card)

210 IF STR (A$(1), 1, 3) = "ZZZ" THEN 630     (Check for last card)

220 DATASAVE BT/42E, F$                       (Initiate reading the next card)
.
.
.   (Process data read at line 200; simultaneously, the next card is
.   read and converted to ASCII by the card reader.)
.
.
.
620 GOTO 2100                                 (Loop back to get converted data
                                               from card reader buffer)

630 STOP                                      (Stop if last card read)
```

In this example, the Hollerith Look-Ahead mode (address '42E') is used to speed up the processing of Hollerith data card images. A card image (82 characters) is read at line 200. At line 210, the card reader is instructed to feed in the next card, convert from Hollerith to ASCII, and store in the output buffer. While this procedure takes place, the data read at line 200 is processed by the system. When processing is completed at line 620, the program loops back to line 200 to read in the next card image, already converted to ASCII and waiting in the card reader's output buffer. Note that F$ in line 210 is a dummy variable whose value is not affected by the Look-Ahead operation.

CHAPTER 8
LOADING HOLLERITH BASIC PROGRAMS AND PROGRAM OVERLAYS
(LOAD, ADDRESS 62B)

8.1 INTRODUCTION

This chapter covers the loading and overlaying of BASIC programs from standard 80-column Hollerith punched cards. Although the term "punched cards" is used throughout, cards read by the Model 2244A may be either punched or marked. For reading with a Model 2234A, of course, only punched cards are legal. The LOAD statement and the LOAD command are used with a device address of 62B to load and overlay Hollerith program decks. Batch processing of Hollerith program decks involves a somewhat different loading technique, and is discussed in Chapter 9.

The BASIC program cards read by the Model 2234A in Mode #1 must be standard 80-column cards punched in Hollerith. BASIC program cards read by the Model 2244A in this mode may contain fewer than 80 columns (if the cards have timing marks), and they may be either marked or punched, as long as they are in Hollerith Code.

8.2 LOADING HOLLERITH PROGRAM DECKS

A "program deck" is a deck of BASIC program cards which conform to the format conventions given in Section 8.4, and containing a series of program lines which collectively constitute a complete BASIC program. A "program overlay deck" is identical to a program deck, except that it typically constitutes only a segment or portion of a complete BASIC program, and is loaded in or "overlayed" under program control with a LOAD statement. The last card in a program or program overlay deck must always be an End card, as described in the section entitled "End of Program".

BASIC program and program overlay decks are loaded into memory from cards with the LOAD instruction. The LOAD instruction has two forms, the LOAD command and the LOAD statement, distinguished by their mode of execution. When LOAD is executed in immediate mode (no line number), it is always interpreted as the LOAD command. When LOAD is executed in a program (in a numbered statement line), it is always interpreted as the LOAD statement. Typically, the LOAD command is used to load program decks, while the LOAD statement is used to load program overlay decks. The LOAD command is described in this section, the LOAD statement in the following section.

General Form: LOAD $\begin{bmatrix} \text{/62B} \\ \text{\#n} \end{bmatrix}$

Where:

/62B = The device address which designates the card reader as the device from which programs are loaded, and also determines the type of code conversion which is to be performed. Address 62B causes program text to be converted from Hollerith to ASCII.

#n = A file number to which the device address 62B has been assigned in a SELECT statement ('n' is an integer from 1 to 6).

If neither a device address nor a file number is specified, the address of the Console Tape device (normally, the Console Tape cassette drive, address 10A) is used. Address 62B can be designated as the Console Tape address with a SELECT TAPE 62B statement. In that case, any subsequent card reader statements which do not specify a device address or file number automatically default to 62B.

Purpose:

The LOAD command with a device address of 62B initiates the reading of BASIC program cards, and automatically converts the program text from Hollerith to ASCII. Newly-loaded program text is appended to the current program in memory, with new program lines which have the same line numbers as existing lines replacing the old lines in memory. Otherwise, existing program text in memory is unaffected by the LOAD operation. For example, if the old program in memory has line numbers 10,20,30, etc., and the newly loaded program has line numbers 15,25,35, etc., the resultant program in memory following the LOAD is numbered, 10,15,20,25,30, etc. Lines which contain syntax errors are loaded and displayed with an appropriate error code. The LOAD operation is not terminated by syntax errors, but the program cannot be run until all syntax errors are corrected. The last card in the program deck must be an END card (with an 'E' in column 80); otherwise, the system continues attempting to load program lines.

To avoid the problem of accidentally including old program text in the newly loaded program, the old program should be cleared from memory prior to the LOAD. After the new program has been loaded from cards, the operator must enter RUN, EXECUTE to run the program.

Example 8-1: Loading a Hollerith Program Deck (LOAD Command, Address 62B)

```
CLEAR
LOAD/62B
```

In this example, old program text and variables are cleared from memory, and a new program is loaded from Hollerith cards. After the new program is loaded, the operator must enter RUN, EXEC to run the program. The program must be terminated by an 'E' in the last (80th) column of the last program card, or by the inclusion of an END card as the last card in the deck.

If a syntax error is detected in a program line as the card is read, the faulty line is displayed or printed, along with an appropriate error code, on the Console Output device. The system continues loading program lines until an END card is read. The program cannot, however, be run until all syntax errors have been corrected. If the system does not encounter an END card (or an 'E' in the last column of a program card), it continues reading cards, and hangs up when no more cards are in the hopper.

8.3 LOADING HOLLERITH PROGRAM OVERLAY DECKS

General Form: LOAD $\begin{bmatrix}\text{/62B,}\\ \text{\#n,}\end{bmatrix}$ [L1, L2]

Where:

/62B =	The device address which designates the card reader as the device from which programs are to be loaded, and also determines the type of code conversion which is to be performed. Address 62B causes program text to be converted from Hollerith into ASCII.
#n =	A file number to which the device address 62B has been assigned in a SELECT statement ('n' is an integer from 1 to 6). If neither a device address nor a file number is specified, the address of the device currently selected as Console Tape device (normally, the console tape cassette drive, address 10A) is used. Address 62B can be designated as the Console Tape address with a SELECT TAPE 62B statement. In this case, any subsequent card reader statements which do not specify a device address or file number automatically default to address 62B.
L1 =	The line number of the first line of resident program text to be cleared from memory prior to loading in the new program, and the first line to be executed in the overlayed program.
L2 =	The line number of the last line of resident program text to be cleared from memory prior to loading in the new program.

Purpose:

The LOAD statement with address 62B initiates the reading and conversion of BASIC programs from Hollerith cards. The LOAD statement must be executed on a numbered statement line (otherwise, it is interpreted as a LOAD command). When the LOAD statement is executed, it produces an automatic combination of the following operations:

STOP - Stop current program execution.

CLEAR P - Clear all resident program text, or that portion specified by lines L1 and L2. If one line number is specified, (L1), all resident program text beginning with that line is cleared. If two line numbers are specified (L1, L2) all lines between and including those lines are cleared. If no line number is specified, all resident program text is cleared.

CLEAR N - Clear all non-common variables. Common variables are unaffected.

LOAD - Load BASIC program from cards, convert from Hollerith to ASCII. Stop loading when an END card ('E' in column 80) is read.

RUN - Run the program, beginning at line L1, if specified, or at the lowest line number in memory, if L1 and L2 are not specified.

The LOAD statement is useful in loading program overlays because of its ability to automatically clear a specified portion of resident program text prior to loading, and to automatically execute the newly loaded program segment after loading. Non-common variables are cleared along with the program text, but common variables, which may be needed by successive overlays, are unaffected. The LOAD statement also may be used to load complete programs. In that case, no line numbers are included in the LOAD statement, so that all program text in memory is cleared before the new program is loaded.

Example 8-2: Loading a Hollerith Program Deck
(LOAD Statement, Address 62B)

```
100 LOAD/62B
```

Statement 100 automatically clears all program text and non-common variables from memory, and loads in a new program from cards. Program text is automatically converted from Hollerith to ASCII. Loading stops when an END card is read. After loading, the new program is automatically run from the lowest statement line.

Example 8-3: Loading a Hollerith Program Overlay Deck
(LOAD Statement, Address 62B)

```
500 LOAD/62B, 100, 500
```

When it is executed, statement 500 clears statement lines 100 through 500 inclusive from the resident program, along with all non-common variables. The new program is then read from cards, converted to ASCII, and stored. Loading stops when an END card is read. After loading, the new program is automatically run starting at line 100. (If there is no line 100 in the new program, an error is signalled.)

Note that if the LOAD statement includes one or two line numbers (L1,L2), the program to be overlayed must have a line number identical to the first line number specified in the LOAD statement (L1). Otherwise, an error is signalled when the system attempts to run the program from that line. Note, too, that the overlay deck must be terminated with an END card. If no END card is read, the system continues reading cards until it encounters an END card or hangs up with no more cards to read.

If a program and several program overlays and data decks are to be read from cards, the main program deck should be loaded in the input hopper first (and followed by an END card). Program overlay and data decks should then be loaded into the hopper in the order in which they will be read or called in by the main program. Each overlay deck also must have an END card as the last card in the deck (the End card is discussed below in Section 8.5, "End of Program").

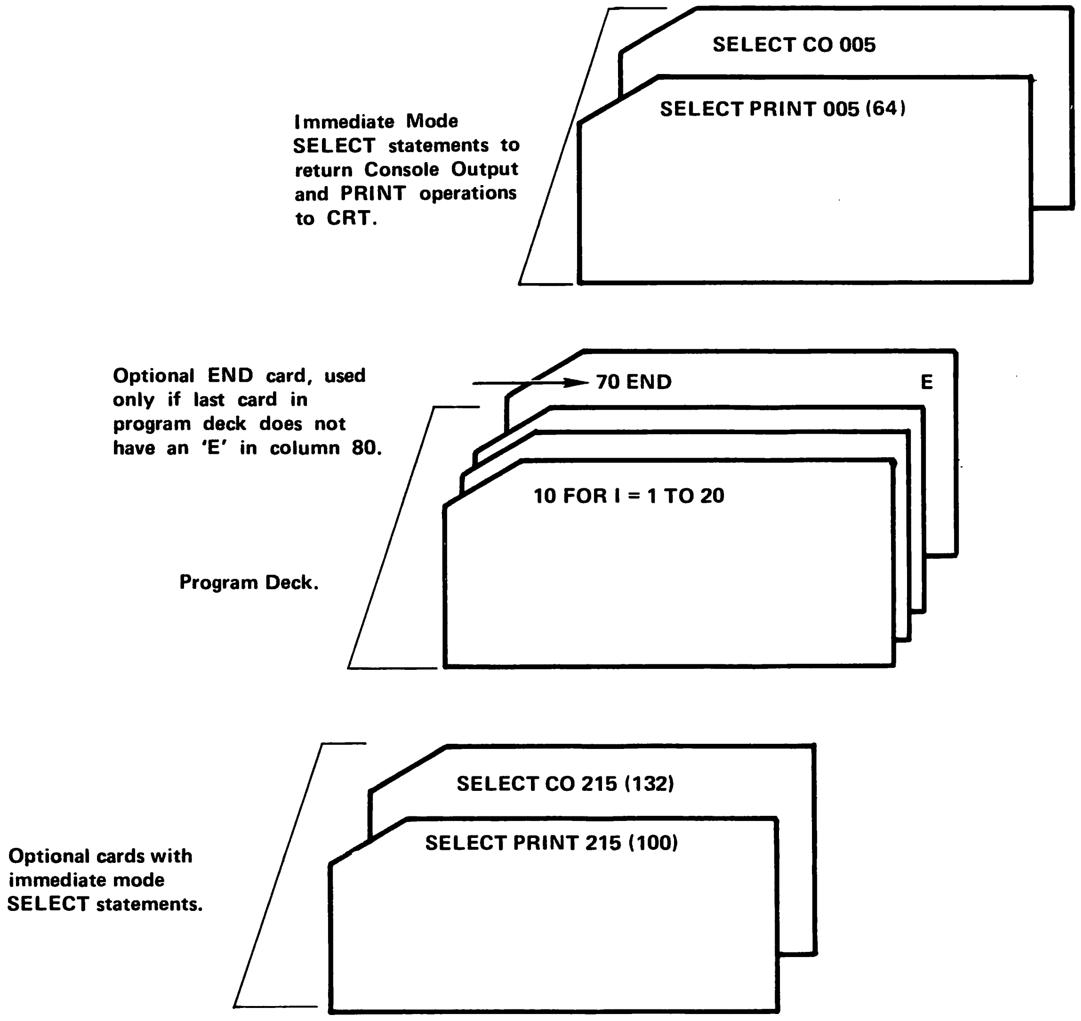

SELECT CO 005
SELECT PRINT 005 (64)
Immediate Mode SELECT statements to return Console Output and PRINT operations to CRT.
Optional END card, used only if last card in program deck does not have an 'E' in column 80.
70 END
E
10 FOR I = 1 TO 20
Program Deck.
SELECT CO 215 (132)
SELECT PRINT 215 (100)
Optional cards with immediate mode SELECT statements.

Figure 8-1.
Typical Card Deck Arrangement for Single Hollerith Program Deck Read with LOAD.

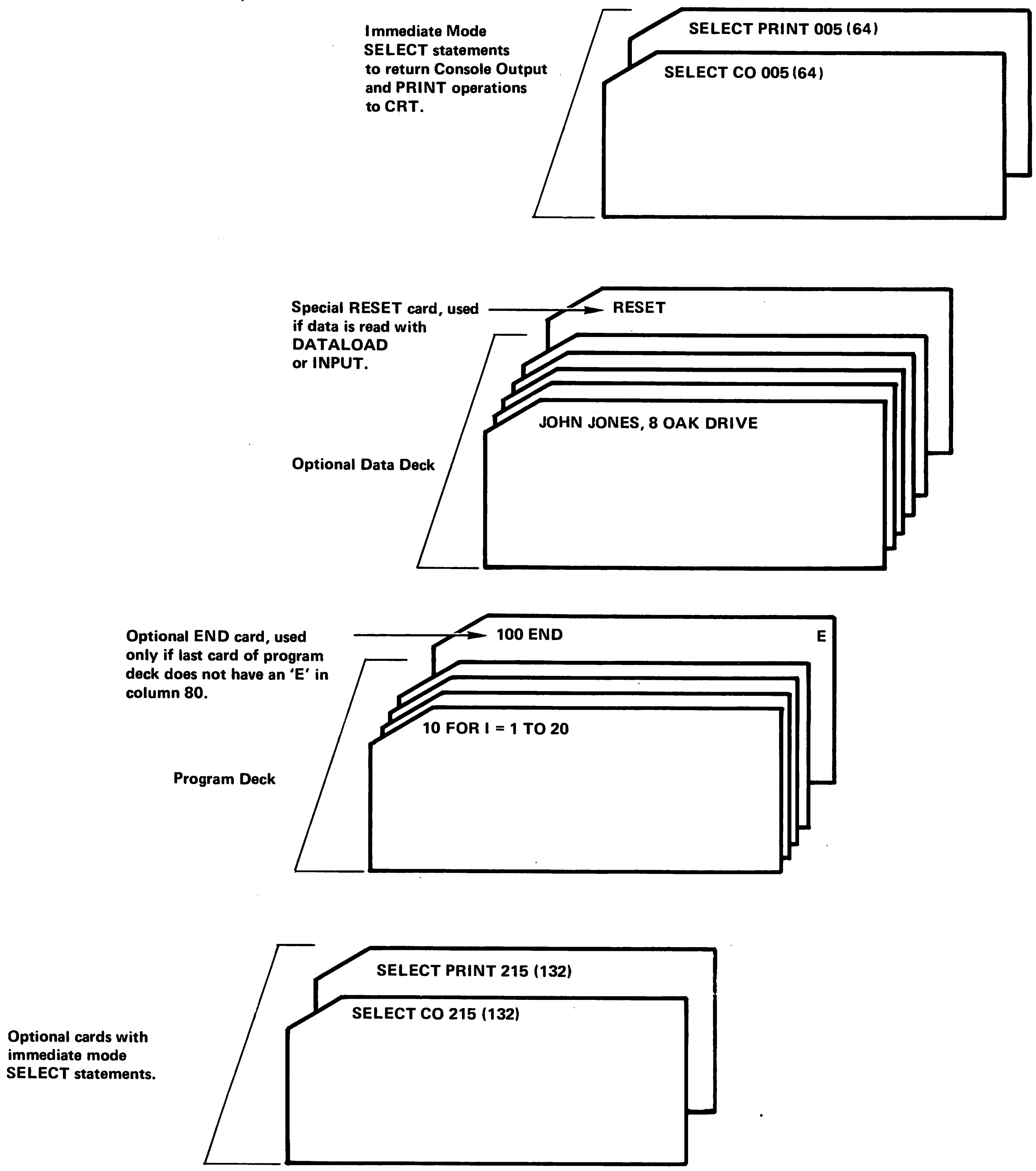
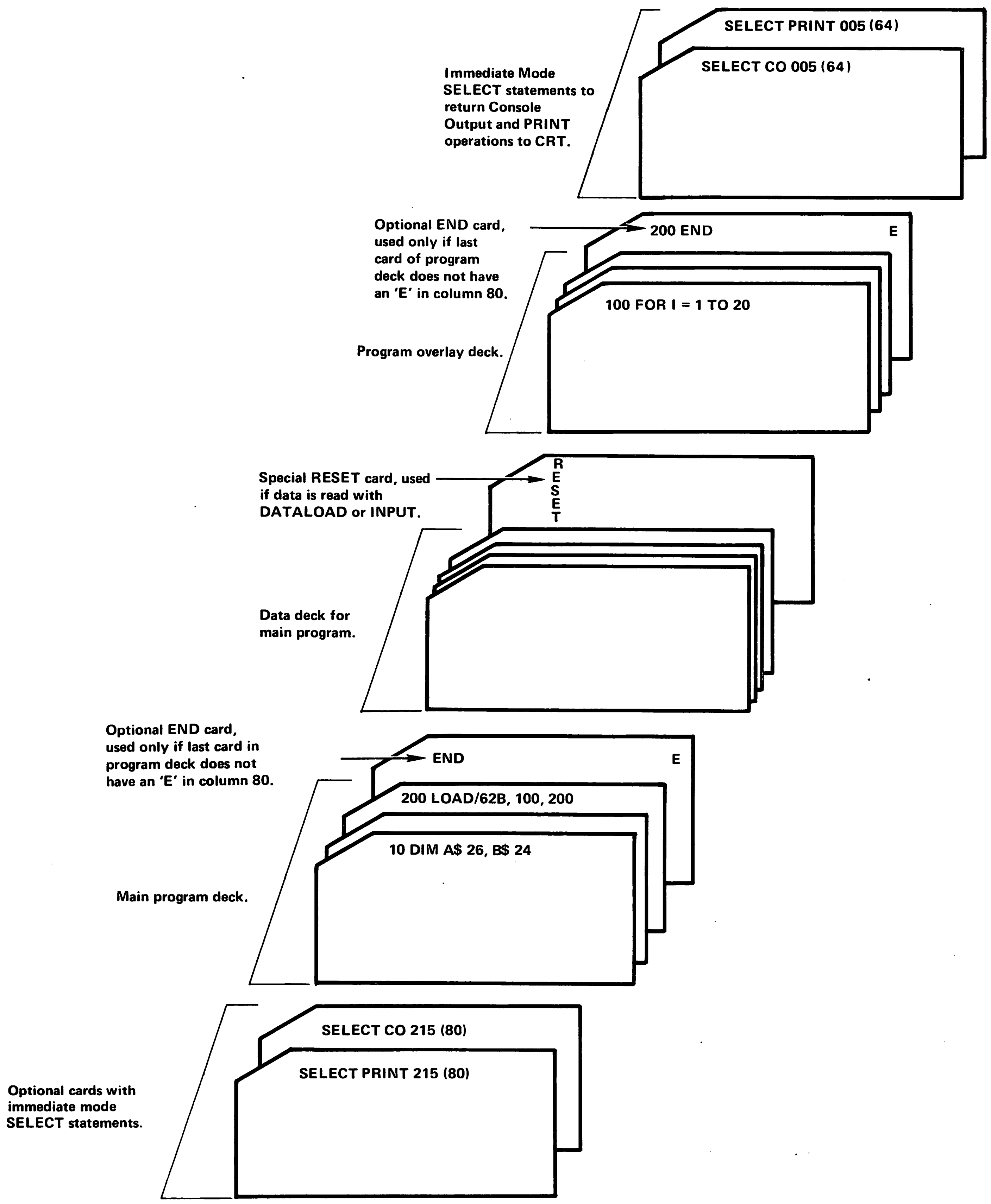

Immediate Mode SELECT statements to return Console Output and PRINT operations to CRT.
SELECT PRINT 005 (64)
SELECT CO 005 (64)
Special RESET card, used if data is read with DATALOAD or INPUT.
RESET
JOHN JONES, 8 OAK DRIVE
Optional Data Deck
Optional END card, used only if last card of program deck does not have an 'E' in column 80.
100 END
E
10 FOR I = 1 TO 20
Program Deck
SELECT PRINT 215 (132)
SELECT CO 215 (132)
Optional cards with immediate mode SELECT statements.

Figure 8-2.
Typical Card Deck Arrangement for Single Hollerith Program Deck and Associated Data Deck Read with LOAD.



Immediate Mode SELECT statements to return Console Output and PRINT operations to CRT.
SELECT PRINT 005 (64)
SELECT CO 005 (64)
Optional END card, used only if last card of program deck does not have an 'E' in column 80.
200 END
E
100 FOR I = 1 TO 20
Program overlay deck.
Special RESET card, used if data is read with DATALOAD or INPUT.
RESET
Data deck for main program.
Optional END card, used only if last card in program deck does not have an 'E' in column 80.
END
E
200 LOAD/62B, 100, 200
10 DIM A$ 26, B$ 24
Main program deck.
SELECT CO 215 (80)
SELECT PRINT 215 (80)
Optional cards with immediate mode SELECT statements.

Figure 8-3.
Typical Card Deck Arrangement for Hollerith Program Deck with Program Overlay Deck and Associated Data Decks Read with LOAD.

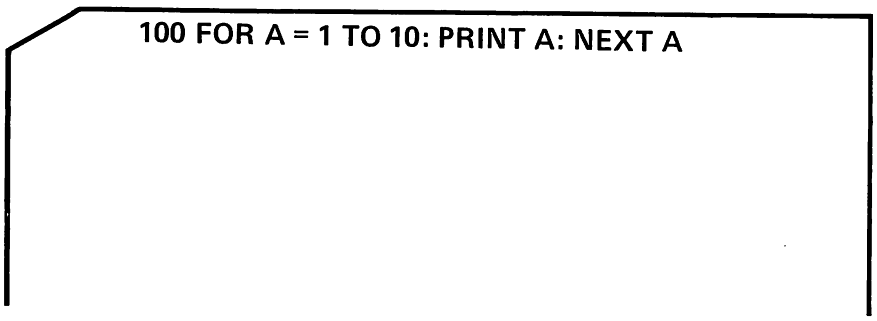
8.4 HOLLERITH BASIC PROGRAM CARD FORMAT

Program text loaded from cards in mode #1 must conform to the same conventions which govern program text entry from the keyboard. Additionally, certain conventions concerned exclusively with the reading and recording of program text on Hollerith cards must be observed. Briefly, the conventions are as follows:

1. The line number must occupy the first column(s) on the card. No data may be marked or punched before the line number. Leading spaces (i.e., blank columns preceding the line number) are read as part of the line number. The use of leading spaces is not recommended.

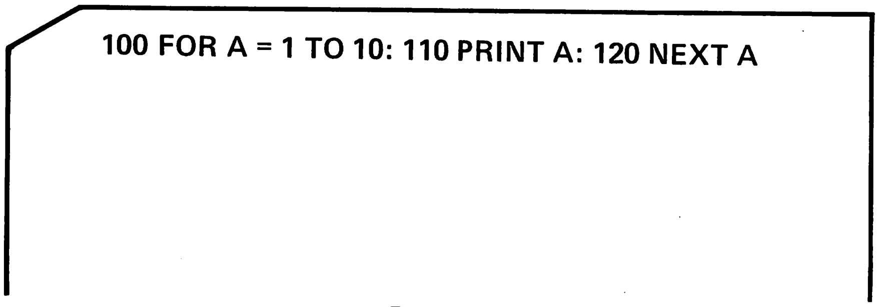
2. Multiple-statement lines on a card are legal, provided the individual statements are separated by colons (:). However, only one numbered statement line may be entered on a card. A single numbered line may consist of one or more statements (appropriately separated by colons), but attempts to record more than one line number on the same card will produce a reading error:

Right



```
100 FOR A = 1 TO 10: PRINT A: NEXT A
```

Wrong



```
100 FOR A = 1 TO 10: 110 PRINT A: 120 NEXT A
```

Figure 8-4.
Right and Wrong Methods of Recording Multiple-Statement Lines on a Single Hollerith Program Card. (Multiple-statement lines separated by colons are permitted; multiple line numbers are not. The second card produces an ERROR 31 [Illegal Line Number] when read.)

It is possible to continue the same program line onto a second card. See Section 8.6, "Continuation of a Program Line".

3. In general, cards containing program lines without line numbers produce unpredictable results, and should not be used in this reading mode. There are, however, two exceptions to this rule:

   a. The special END card (discussed under convention #8 and in Section 8.5, "End of Program") requires no line number.

   b. SELECT statements can be recorded on cards without line numbers. The availability of SELECT statements in immediate mode makes possible the dynamic selection of device addresses.

4. The use of Hollerith characters whose ASCII equivalents are Carriage Return (multi-punch rows 12, 9, 8, 5) or X-OFF (multi-punch rows 11, 9, 3) characters is illegal in this mode. (Note that a Carriage Return is not required to terminate a line on a card.)

5. Every column on the card from the first digit of the line number to the last character in the line is read and converted from Hollerith into ASCII. Unpunched or unmarked columns are treated as space characters. Space characters embedded within the program text are read; space characters which follow the last character on the line (trailing spaces) are ignored, however. This is important, because space characters occupy memory when a program is stored in the system.

6. Any Hollerith code which does not convert into a legal ASCII character is automatically read as an ASCII exclamation character ('!'). Such illegal characters normally cause the program line in which they appear to be displayed on the CRT (or the selected output device) with an appropriate syntax error code, thus facilitating easy identification and correction of the erroneous character(s). One or more lines with syntax errors do not prevent the program from loading (loading continues until an END card is read), but the program cannot be run until all syntax errors have been corrected.

7. The last (80th) column on a program card is reserved for two special characters, indicating continuation (an '&' character), or end of program (an 'E' character). Any other character punched or marked in the 80th column is read as an exclamation character ('!'), and generally will produce a syntax error when the card is read. Therefore, if the special characters for continuation or end of program are not used, the 80th column should always be left blank.

8. Each program or program overlay must be ended with an end character ('E') in the 80th column of the last program card, or with a special End card. (See Section 8.5, "End of Program".)

9. Blank cards are ignored by the card reader.

8.5 END OF PROGRAM

The end of a program deck or program overlay deck is indicated in one of two ways:

1. Punching or marking an 'E' in the last (80th) column of the last program card in the deck, or

2. Including a special END card, which has an 'E' in column 80, as the last card in the deck.

If the 'E' character is included on the last program card in the deck, that card must contain some program text. A card containing an 'E' character in the 80th column, but with no other columns punched or marked, is ignored by the system.

A special END card can be created by punching or marking the characters 'END' in the first three columns on the card, and punching or marking an 'E' in the last column. This card should then follow the last program card in the deck. In this case, the 80th column on the last program card should be left blank.

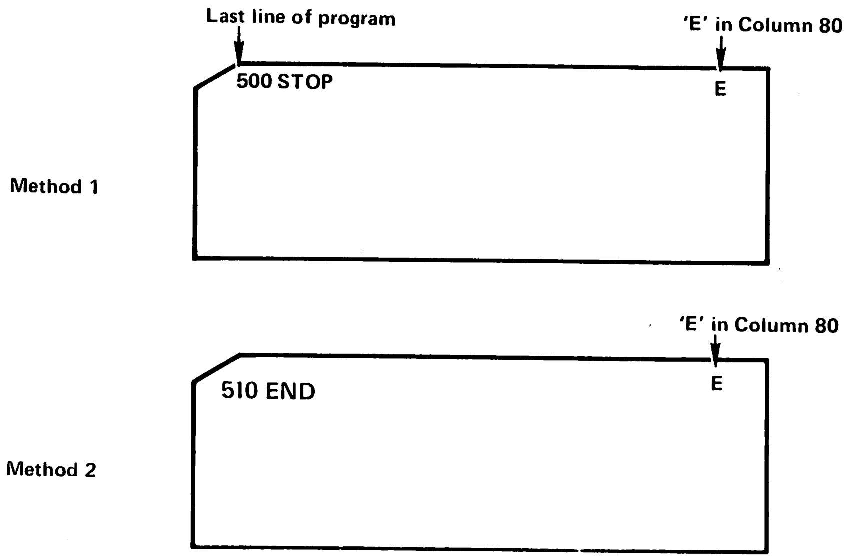


Figure 8-5.
Two Methods of Signalling End-of-Program. In the First Case, an "E" Is Punched or Marked in the 80th Column of the Last Program Card. In the Second Case, a Special Card Is Created with the Word 'END' and an 'E' in Column 80. This Card Is Added at the End of the Deck.

It is imperative that every program deck end with one of the two types of END cards described above, since it is the END card which terminates the program loading operation. If no end-of-program condition is detected, the system continues loading any remaining cards in the input hopper (with unpredictable results, if they happen to be data cards), or hangs up if the hopper is empty.

8.6 CONTINUATION OF A PROGRAM LINE

In most cases, each program line occupies a single card. An unusually long program line may, however, extend onto two or more cards. To indicate that a single program line is to be continued onto a second card, an ampersand character ('&') must be marked or punched in the last (80th) column of the first card. The maximum legal length of a single program line is 192 characters (about 2-1/2 80-column cards), including embedded space columns (unpunched, unmarked), but not including trailing spaces on the last card. If a line is continued onto more than one card, every card except the last must contain an ampersand ('&') in column 80. The last card must not have an ampersand in the 80th column.

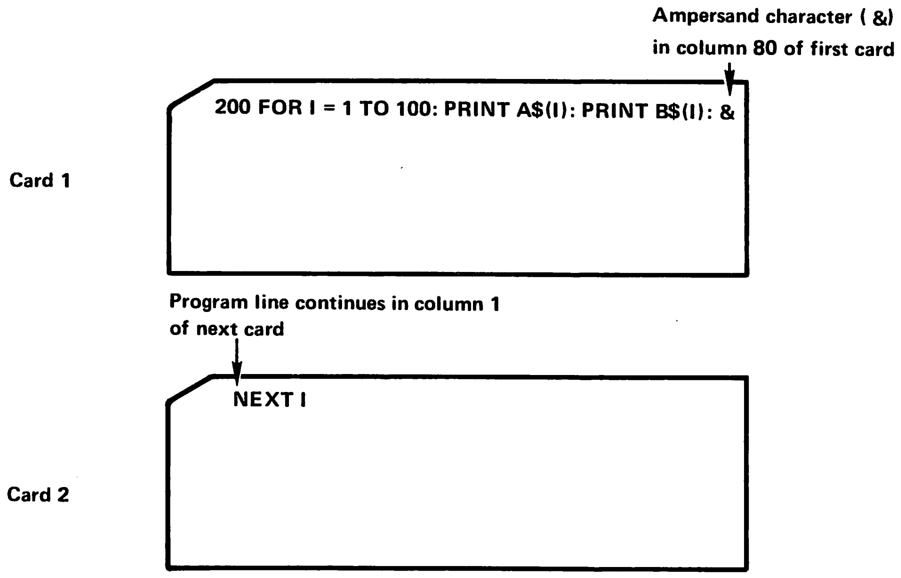

Ampersand character (&)
in column 80 of first card
200 FOR I = 1 TO 100: PRINT A$(I): PRINT B$(I): &
Card 1
Program line continues in column 1
of next card
NEXT I
Card 2

Figure 8-6.
Continuing a Single Program Line from One Hollerith Card to a Second.

CHAPTER 9
BATCH PROCESSING HOLLERITH BASIC PROGRAMS

9.1 INTRODUCTION TO "BATCH PROCESSING" ON THE MODELS 2234A AND 2244A

The term "batch processing" denotes an operation in which a series of discrete program decks and associated data decks are automatically loaded and run in sequence without normal user intervention. A "batch" of individual program and data decks are loaded into the input hopper, separated by system command cards such as CLEAR, LOAD, LIST, and RUN. The card reader is then selected for Console Input operations, and automatically begins loading in the first program deck. If the program requires data to be entered from cards, an accompanying data deck is automatically read. Thus, in effect, the card reader assumes the role normally taken by the keyboard. At any point where the system would normally expect a program line or system command to be entered from the keyboard, the card reader is accessed instead to automatically read the next card. Program lines are therefore loaded from cards just as if they had been keyed in from the keyboard, and system commands and immediate mode statements (such as LIST, LOAD, RUN, CLEAR, SELECT, etc.) are executed immediately upon being read from cards. When the first program has completed execution, the system command cards are read and executed, and the next sequential program is automatically loaded and run. Hardcopy output and listings for each program can be generated on a line printer or output writer by dynamically changing the LIST, PRINT and CO parameters with immediate mode SELECT cards. The batch operation continues until all cards have been read and processed.

9.2 LOADING HOLLERITH BASIC PROGRAMS WITH CONSOLE INPUT

The principal value of Console Input mode derives from its usefulness in batch processing operations. On the System 2200A, however, and on a System 2200S or WCS/10 system without Option-22 or -23, Console Input provides the only method of loading BASIC programs from cards. Hollerith cards are read under Console Input control by selecting the card reader for Console Input operations with a device address of 02B:

```
SELECT CI 02B
```

Immediately after this statement is keyed in and executed, the card reader assumes all program input operations normally associated with the keyboard. The following sequence of events then takes place:

1. If the card reader is in a ready condition, cards are read and automatically converted from Hollerith to ASCII.

2. Numbered program lines are read from cards, displayed on the currently selected Console Output device (normally, the CRT), and entered into memory, just as if they had been entered from the keyboard.

3. Numbered program lines which contain syntax errors are printed or displayed with an appropriate error code, and entered in memory. Erroneous program lines do not terminate the program loading operation, but the program cannot be run until all errors have been corrected.

4. System commands (such as CLEAR, LOAD, LIST, RUN, etc.) are read from cards, printed on the currently selected Console Output device, and automatically executed, just as if they had been entered from the keyboard.

5. Immediate mode statements without line numbers (such as SELECT, PRINT, etc.) are read from cards, printed on the currently selected Console Output device, and automatically executed, just as if they had been entered from the keyboard.

The SELECT CI 02B statement is, of course, entered from the keyboard. Once it is executed, the keyboard is locked out, and all subsequent commands and statements must be entered from cards. Typically, control cards with immediate mode SELECT statements directing the LIST, PRINT, and/or Console Output operations to a printer are included at the beginning of a card deck. Corresponding SELECT cards generally are included at the end of the deck to reselect output back to the CRT. Additionally, each program deck should be preceded by a CLEAR command card. (Note that if the CLEAR card follows the SELECT CO card, all PRINT and LIST operations, as well as Console Output operations, are defaulted to the printer whose address is specified in the SELECT CO statement.) If Console Output is selected to a printer, each card is automatically printed out as it is read, thus providing a hardcopy listing of the entire program deck. In addition, all syntax error messages are printed as they occur, thereby producing a useful "audit trail" for debugging purposes.

Each program deck must be followed by certain control cards. A RUN command card must follow the last program card. After the last program card has been loaded, the RUN card is read, initiating program execution. The very last card in the batch job stream must be a SELECT CI 001 card which returns Console Input operations to the keyboard. Once this card is read, any remaining cards in the input hopper are ignored.

9.3 HOLLERITH PROGRAM CARD FORMAT FOR CONSOLE INPUT

The format of program lines read from cards with Console Input is identical to that of programs entered from the keyboard, and is also quite similar to the required format of programs read from cards with LOAD. Chapter 8, Section 8.4 ("Hollerith BASIC Program Card Format") lists the applicable format conventions in some detail. In brief, they are:

1. The line number must begin in the first card column; leading spaces are read as part of the line number.

2. Multiple-statement lines are allowed, provided the statements are separated by colons.

3. Hollerith characters which translate to Carriage Return (multi-punch rows 12, 9, 8, 5) and X-OFF (multi-punch rows 11, 9, 3) characters are illegal. A Carriage Return is not required to terminate a line on a card.

4. Hollerith codes which do not convert to legal ASCII codes are automatically converted to ASCII exclamation ('!') characters.

5. The last (80th) card column is reserved for two special characters, indicating continuation of a program line (an '&' character) or end of program (an 'E' character). No other characters may be punched or marked in the 80th column.

6. Blank cards are ignored by the card reader.

9.4 DIFFERENCES BETWEEN 'CONSOLE INPUT' AND 'LOAD'

Although program cards read with the LOAD command or program statement must conform to the same format conventions as programs read with Console Input (and therefore programs read with LOAD can in general be read with Console Input, and vice-versa), there are certain important differences between the two modes:

1. Each card read via Console Input is automatically "echoed" onto the currently selected Console Output device (typically the CRT display) as it is entered. This does not occur with cards read by LOAD. Thus the process of program loading is somewhat faster with LOAD than with Console Input.

2. Console Input reads and executes system commands cards and immediate mode statement cards (i.e., cards which do not contain line numbers) such as CLEAR, RUN, LIST, SELECT, etc. In program decks read with LOAD, only the END card and SELECT statements can be read and executed in immediate mode. In general, other types of cards which lack line numbers produce unpredictable results when read under LOAD control.

3. Programs read with LOAD must be ended with an END card. Programs loaded via Console Input do not require an END card; it is recommended, however, that all programs be terminated with an END card.

9.5 BATCH PROCESSING WITH CONSOLE INPUT

Because of its ability to read and execute immediate mode statements and commands, Console Input is an extremely valuable reading mode for batch processing operations. A batched job stream consisting of several programs and associated data decks separated by appropriate system command cards can be loaded and run under Console Input control. Because commands such as CLEAR, LOAD, and RUN for each program are read from cards rather than entered from the keyboard, the entire job stream can be processed without user intervention (provided the programs are free of syntax errors). Each program is automatically loaded and run; when it has completed execution, it is cleared from memory, and the next program is loaded. This process continues until a card is read which reselects Console Input back to the keyboard.

Within the job stream, each program should be preceded by a CLEAR card and immediately followed by a RUN card. Additional SELECT cards may be inserted at appropriate points to dynamically control the selection of output devices. Figures 9-2 and 9-3 below illustrate typical batched job streams with control cards.

9.6 THE USE OF 'LOAD' IN CONJUNCTION WITH 'CONSOLE INPUT' FOR MORE EFFICIENT BATCH PROCESSING

Although overall control of a batched job stream must lie with Console Input, it is possible to speed up batch processing time by loading individual programs with the LOAD command. This can be done by adding a LOAD command card at the beginning of each program deck (the LOAD card must not have a line number). When the LOAD card is read and executed, it causes the subsequent program to be loaded in under LOAD control. In this case, program lines are not automatically echoed to the Console Output device as the cards are read; the program therefore loads in more rapidly than would be the case under direct Console Input control. Program loading continues until an End card is read. If a listing is desired, an immediate mode LIST card must be inserted at the end of the program deck (but before the RUN card). A typical batch job stream illustrating the use of LOAD command cards for batch processing is shown in Figure 9-4.

9.7 PRINTING BATCHED PROGRAM OUTPUT

In batch mode operation, it is generally good policy to maintain hardcopy listings of the programs loaded from cards, along with the printed output of each program. This is most conveniently done by selecting Console Output operations to a line printer or output writer. With Console Output continuously selected to a printer-type device, each card is automatically printed as it is read and processed, along with all other types of Console Output (including error codes). The hardcopy record thereby produced can be a useful debugging tool, since the type and location of all syntax errors are clearly shown. Console Output is selected to the line printer by including a card containing one of the following immediate mode statements at the beginning of the first program deck:

```
SELECT CO 215 (131) - 2221
SELECT CO 215 (80) - 2231
SELECT CO 211 - 2201
```

Note that the SELECT CO card should not have a program line number.

Console Output does not include the printed output of the PRINT and PRINTUSING statements. In order to produce hardcopy output from these statements, a SELECT PRINT 215 card should follow the SELECT CO card. Alternatively, a CLEAR card may be used following the SELECT CO card. CLEAR automatically resets the PRINT and LIST default addresses to the address currently selected for Console Output. The line lengths for PRINT and LIST also default to the line length specified in the SELECT CO statement. (If the CLEAR card precedes the SELECT CO card, however, the PRINT and LIST addresses are not changed.)

In order to cleanly separate individual program listings from one another, a PRINT HEX (OC) card should be used to eject the printer to the top of the next page for each new job if a line printer is used for output. If an output writer is used, a PRINT HEX (OAOAOA) card should be substituted. In this case, each 'OA' causes the typewriter to skip one line. An immediate mode PRINT HEX (OC) or PRINT HEX (OAOAOA...) card should be inserted at the beginning of each program deck to ensure that the program listing begins on a new page. Similarly, a programmed PRINT HEX (OC) or PRINT HEX (OAOAOA...) may be included in the program prior to each PRINT or PRINTUSING statement, to ensure that all printed program output begins on a new page. (Refer to Figures 9-3, 9-4, and 9-5.)

9.8 READING DATA CARDS IN BATCH PROCESSING MODE

A program processed in batch mode may contain one or more INPUT, DATALOAD, or DATALOAD BT statements used to read in and process data cards from a subsequent data deck. The data deck accompanying each program should immediately follow the program (and the RUN command) in the input hopper (see Figure 9-5). When the program deck has been loaded into memory and the RUN card is read, control reverts from the card reader to the system CPU, where program execution begins. If an INPUT, DATALOAD, or DATALOAD BT statement is encountered in the course of program execution, the card reader is accessed and the data deck is read. When the specified number of data cards or an end-of-file card have been read, the program proceeds with any further processing; upon completion of program execution, control reverts to Console Input, which proceeds to read the next card in the input hopper.

Batch Deck Protection

In programs which read data cards within a batched job stream, there is always the possibility that a logical error in the program, a miscount of the number of data cards in the deck, or a missing EOF card will cause the program to read beyond the data cards and into the control cards which follow. Such an oversight would destroy the first program (which bombs out with invalid data values) and the second program (which would lose its initial control cards). To protect against such an occurrence, a special RESET control card should be inserted as the last card in each data deck. The special RESET card is produced by multi-punching rows 2, 3, and 4 (i.e., multi-punch integers 2, 3, and 4) in the first column of a Hollerith card. The remainder of the card should be left blank.

When a special RESET card is read with INPUT or DATALOAD, it generates a system RESET condition (that is, it has the same effect as touching the RESET button on the keyboard). Thus a program which has erroneously attempted to read data beyond the last valid data card is automatically terminated when the special card is read, and control reverts to Console Input, which reads in the next card . If the RESET control card is read following normal processing of the data deck, the RESET operation does not affect the batch processing operation. It is recommended that a RESET control card be inserted at the end of all data decks read with INPUT or DATALOAD in batch processing mode.

Note that the RESET control card does not function as a special card when read with DATALOAD BT (addresses 629 or 62A). In these reading modes (Modes #3 and #4), a special data termination card can be easily designed and tested for, since an entire card image is read.

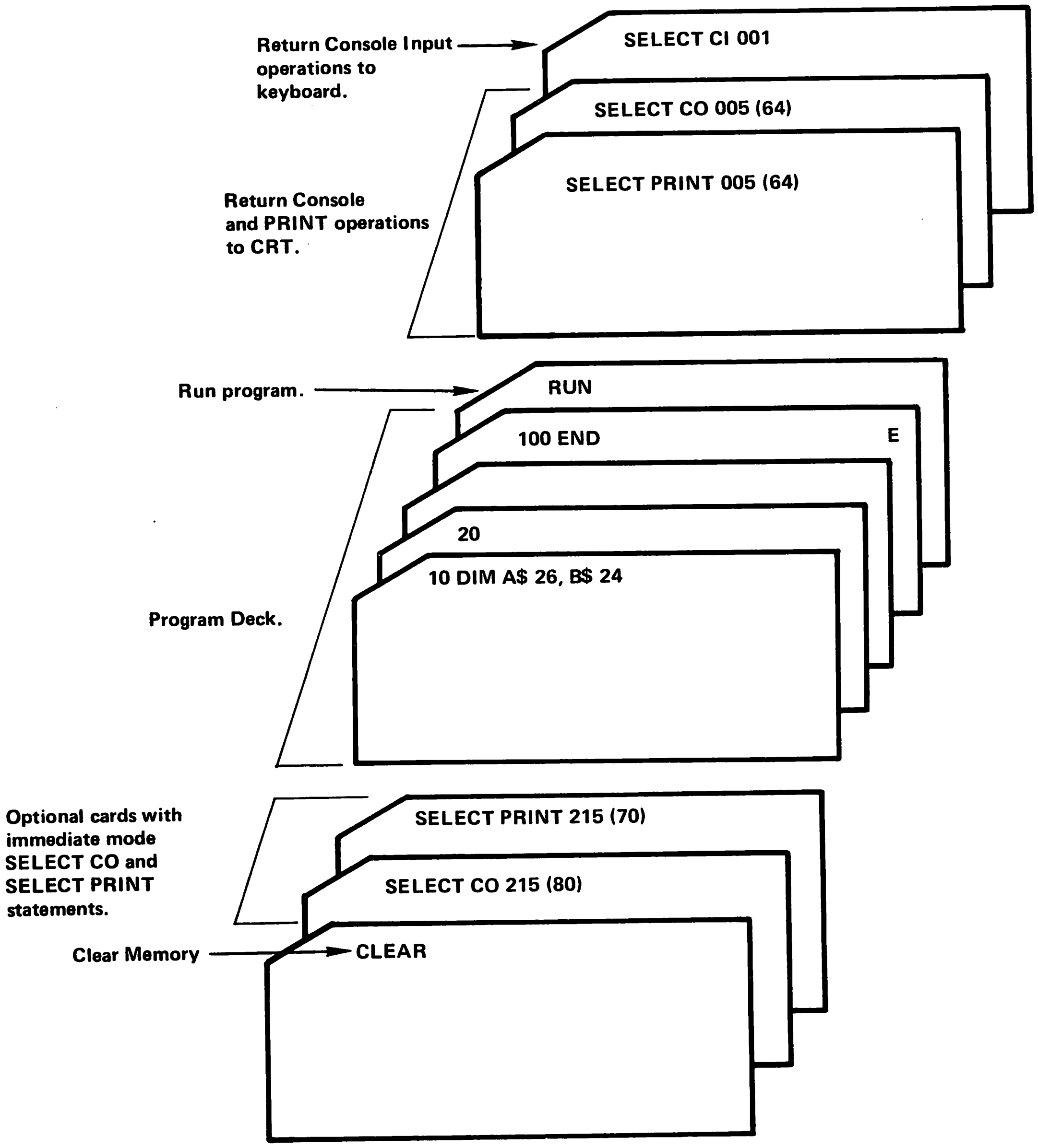


Figure 9-1.
Typical Card Deck Arrangement for Single Hollerith Program Deck Read Via Console Input.

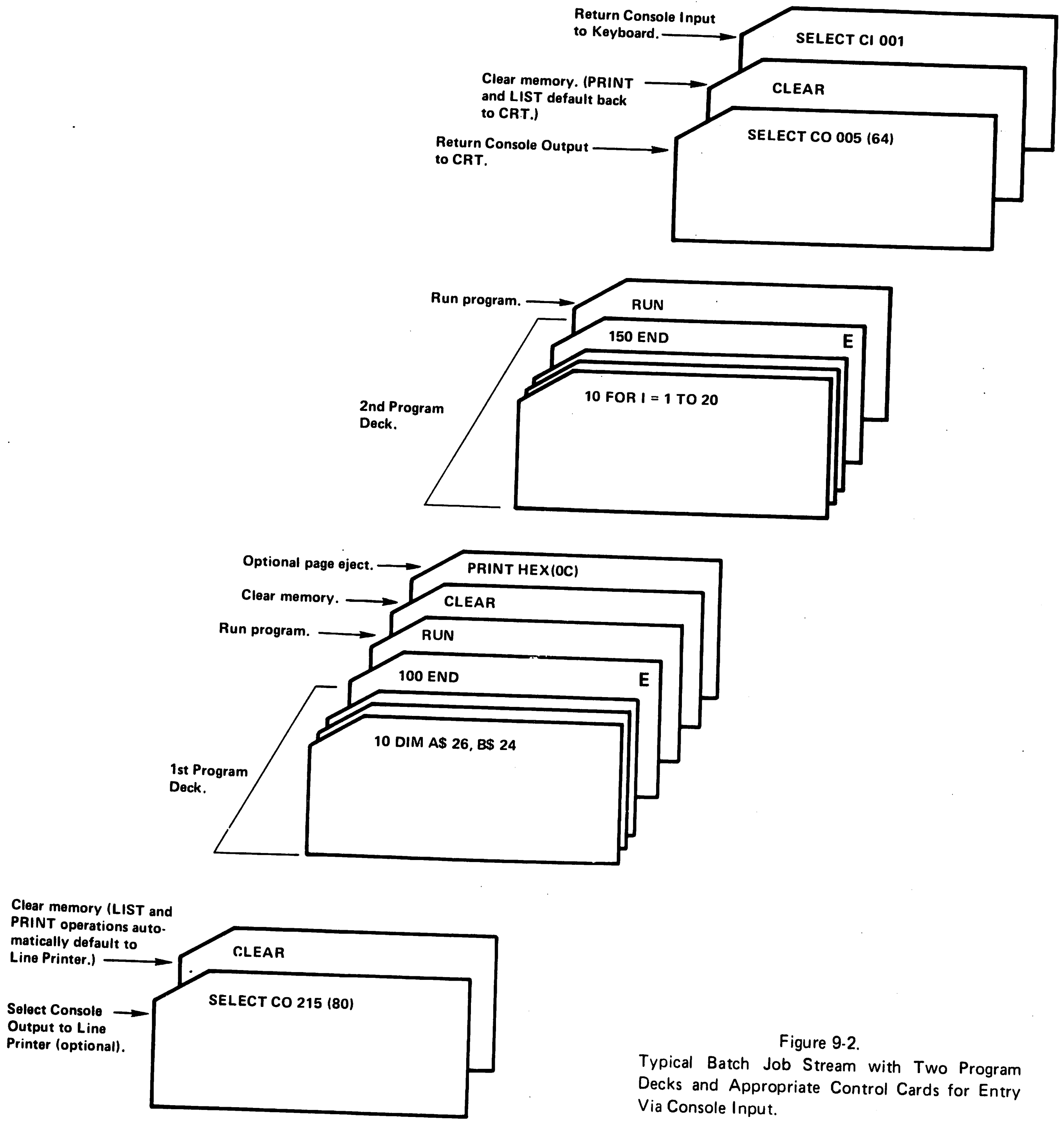

Return Console Input to Keyboard.
SELECT CI 001
Clear memory. (PRINT and LIST default back to CRT.)
CLEAR
Return Console Output to CRT.
SELECT CO 005 (64)
Run program.
RUN
150 END
E
2nd Program Deck.
10 FOR I = 1 TO 20
Optional page eject.
PRINT HEX(0C)
Clear memory.
CLEAR
Run program.
RUN
100 END
E
1st Program Deck.
10 DIM A$ 26, B$ 24
Clear memory (LIST and PRINT operations automatically default to Line Printer.)
CLEAR
Select Console Output to Line Printer (optional).
SELECT CO 215 (80)

Figure 9-2.
Typical Batch Job Stream with Two Program Decks and Appropriate Control Cards for Entry Via Console Input.

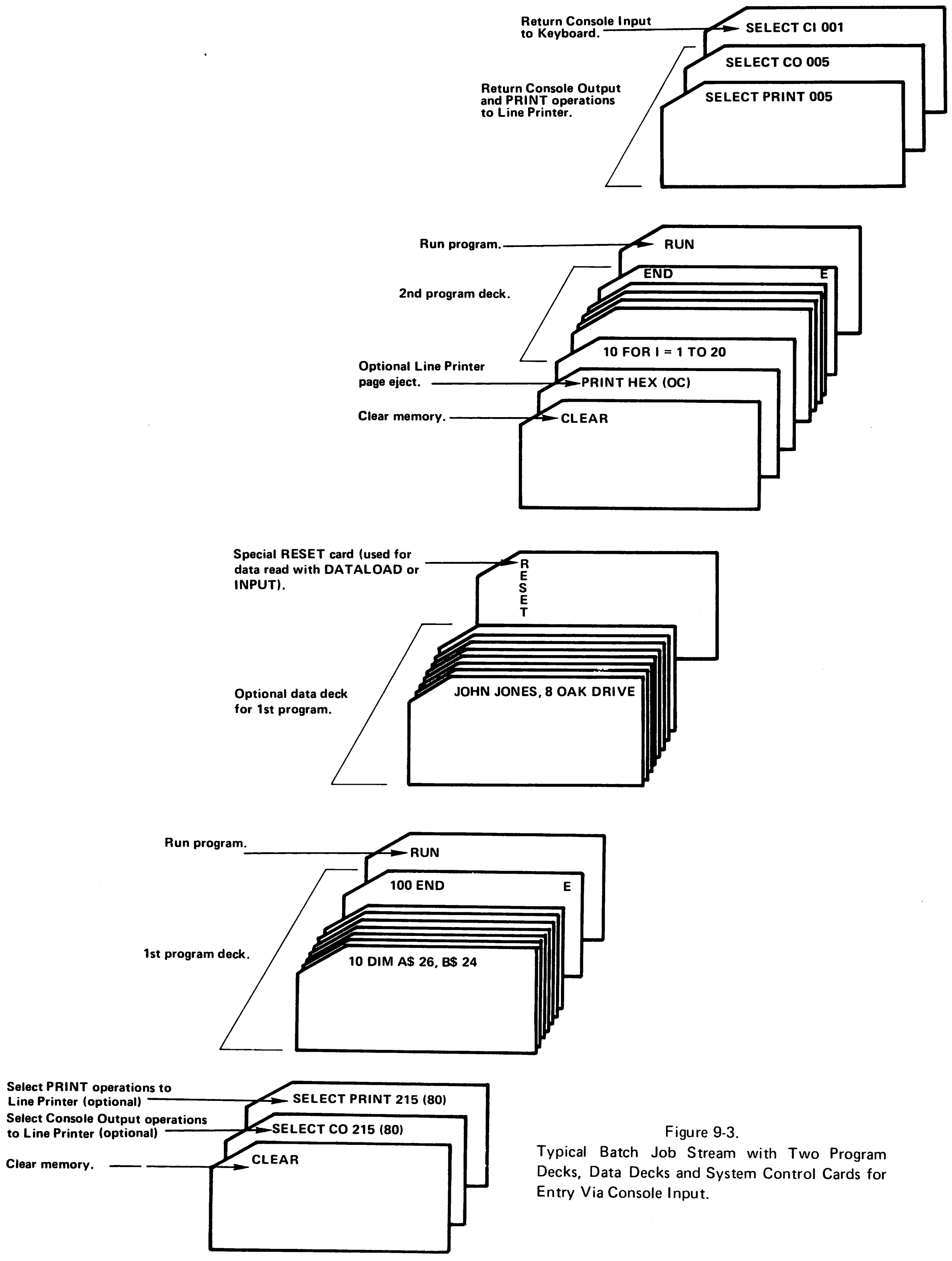

Return Console Input to Keyboard.
SELECT CI 001
SELECT CO 005
Return Console Output and PRINT operations to Line Printer.
SELECT PRINT 005
Run program.
RUN
END
E
2nd program deck.
10 FOR I = 1 TO 20
Optional Line Printer page eject.
PRINT HEX (0C)
Clear memory.
CLEAR
Special RESET card (used for data read with DATALOAD or INPUT).
RESET
Optional data deck for 1st program.
JOHN JONES, 8 OAK DRIVE
Run program.
RUN
100 END
E
1st program deck.
10 DIM A$ 26, B$ 24
Select PRINT operations to Line Printer (optional)
SELECT PRINT 215 (80)
Select Console Output operations to Line Printer (optional)
SELECT CO 215 (80)
Clear memory.
CLEAR

Figure 9-3.
Typical Batch Job Stream with Two Program Decks, Data Decks and System Control Cards for Entry Via Console Input.

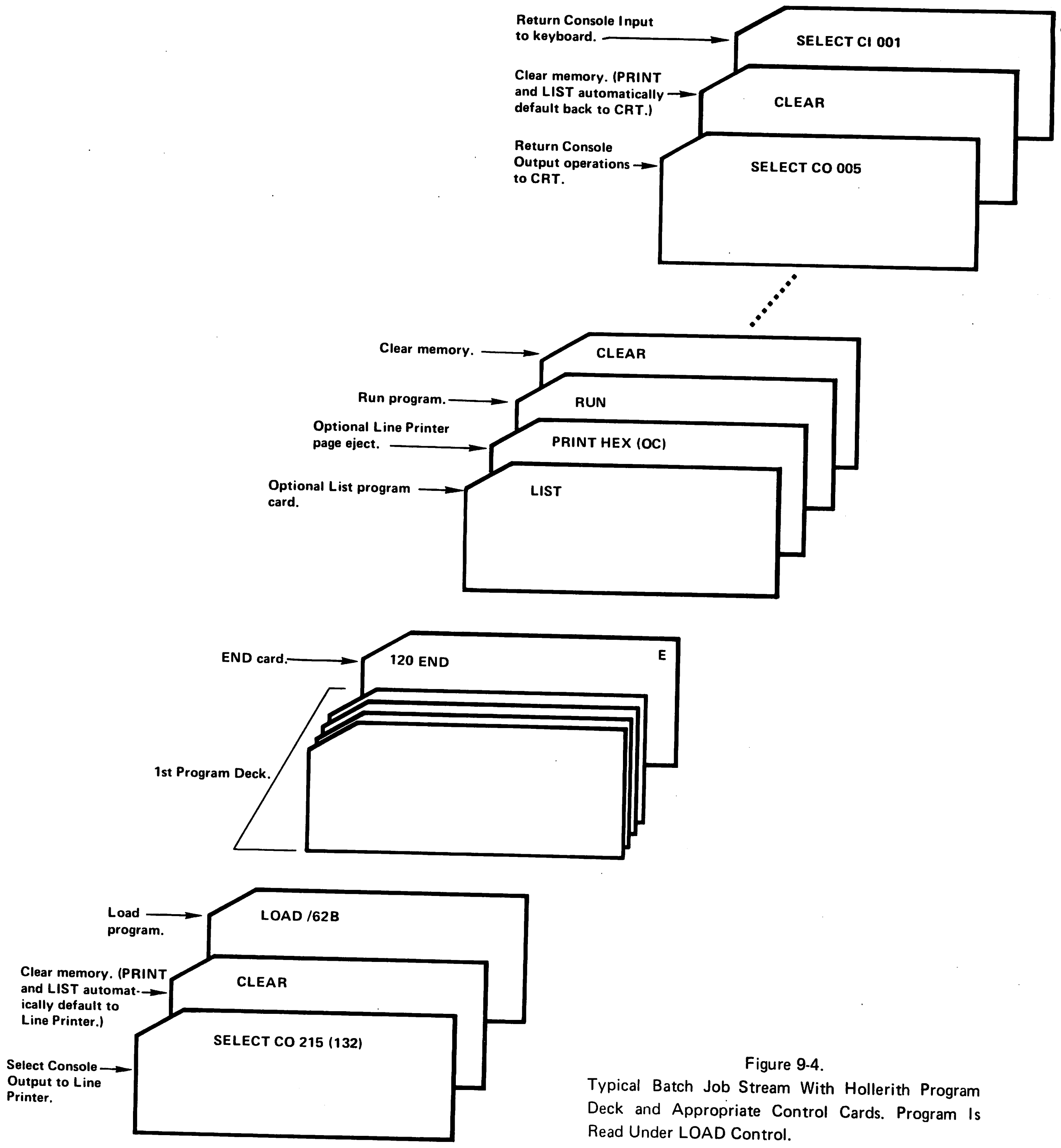
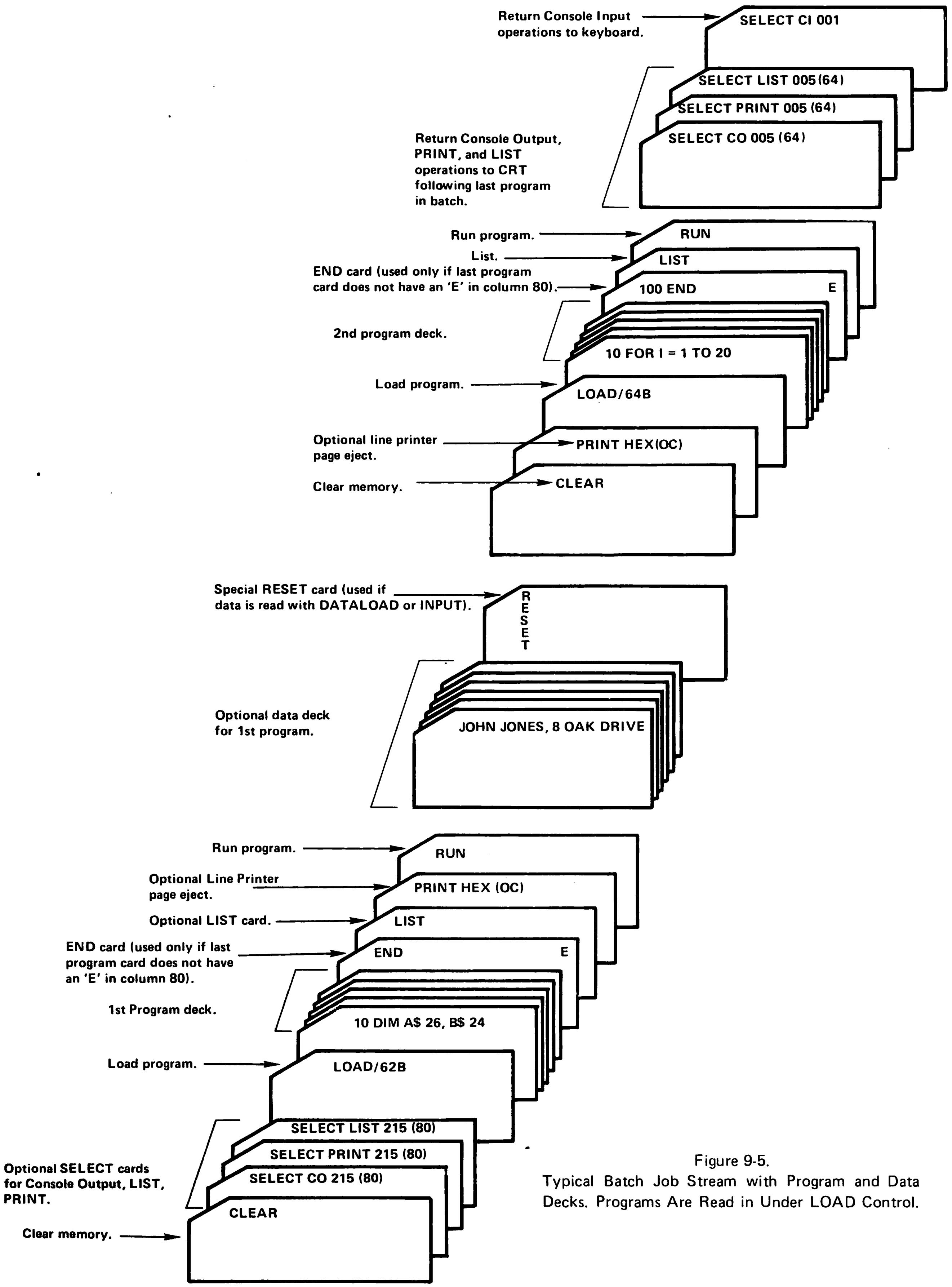

Return Console Input to keyboard.
SELECT CI 001
Clear memory. (PRINT and LIST automatically default back to CRT.)
CLEAR
Return Console Output operations to CRT.
SELECT CO 005
Clear memory.
CLEAR
Run program.
RUN
Optional Line Printer page eject.
PRINT HEX (OC)
Optional List program card.
LIST
END card.
120 END
E
1st Program Deck.
Load program.
LOAD /62B
Clear memory. (PRINT and LIST automatically default to Line Printer.)
CLEAR
Select Console Output to Line Printer.
SELECT CO 215 (132)

Figure 9-4.
Typical Batch Job Stream With Hollerith Program Deck and Appropriate Control Cards. Program Is Read Under LOAD Control.



Return Console Input operations to keyboard.
SELECT CI 001
SELECT LIST 005(64)
SELECT PRINT 005 (64)
SELECT CO 005 (64)
Return Console Output, PRINT, and LIST operations to CRT following last program in batch.
Run program.
RUN
List.
LIST
END card (used only if last program card does not have an 'E' in column 80).
100 END
E
2nd program deck.
10 FOR I = 1 TO 20
Load program.
LOAD/64B
Optional line printer page eject.
PRINT HEX(OC)
Clear memory.
CLEAR
Special RESET card (used if data is read with DATALOAD or INPUT).
RESET
Optional data deck for 1st program.
JOHN JONES, 8 OAK DRIVE
Run program.
RUN
Optional Line Printer page eject.
PRINT HEX (OC)
Optional LIST card.
LIST
END card (used only if last program card does not have an 'E' in column 80).
END
E
1st Program deck.
10 DIM A$ 26, B$ 24
Load program.
LOAD/62B
SELECT LIST 215 (80)
SELECT PRINT 215 (80)
SELECT CO 215 (80)
Optional SELECT cards for Console Output, LIST, PRINT.
CLEAR
Clear memory.

Figure 9-5.
Typical Batch Job Stream with Program and Data Decks. Programs Are Read in Under LOAD Control.

PART III
BASIC MARK SENSE PROGRAM/DATA CARDS
CHAPTER 10
MARKING THE BASIC MARK SENSE CARDS

10.1 THE TWO BASIC MARK SENSE PROGRAM CARDS

The Model 2244A card reader is capable of reading a variety of custom-designed mark sense cards. The special codes used on such cards must be converted to a meaningful form by a user-generated program. Two special mark sense cards are available, however, whose character codes can be read and converted automatically into ASCII by the card reader itself. The two special cards are the standard format BASIC mark sense card (Figure 10-1) and the Wang format BASIC mark sense card (Figure 10-2). Wang BASIC mark sense cards may be purchased directly from Wang Laboratories. Standard BASIC mark sense cards are not stocked by Wang.

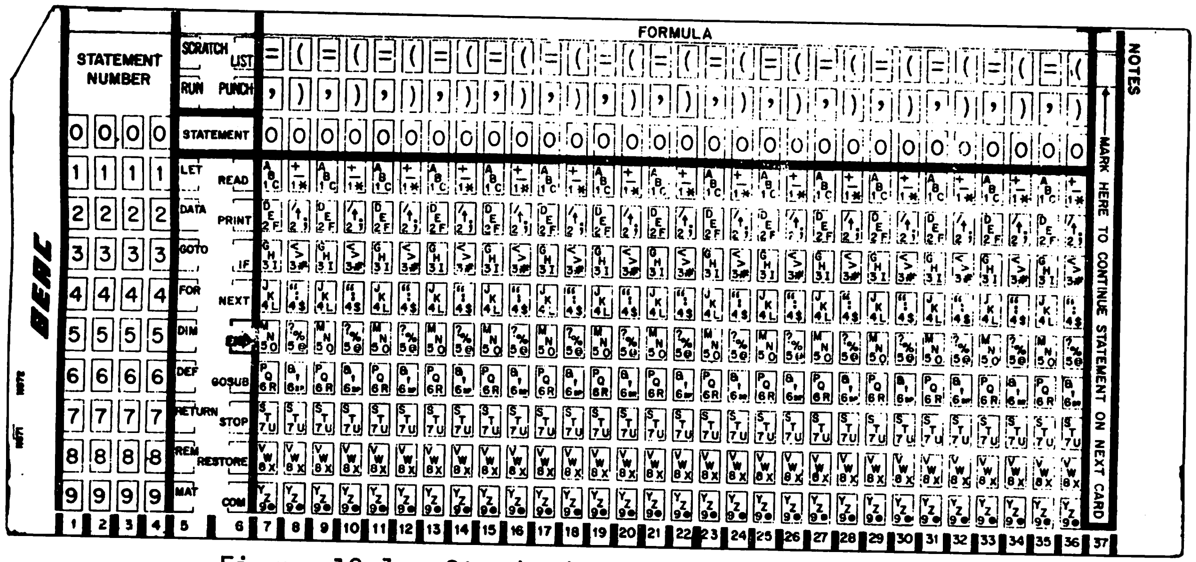


Figure 10-1. Standard BASIC Mark Sense Card

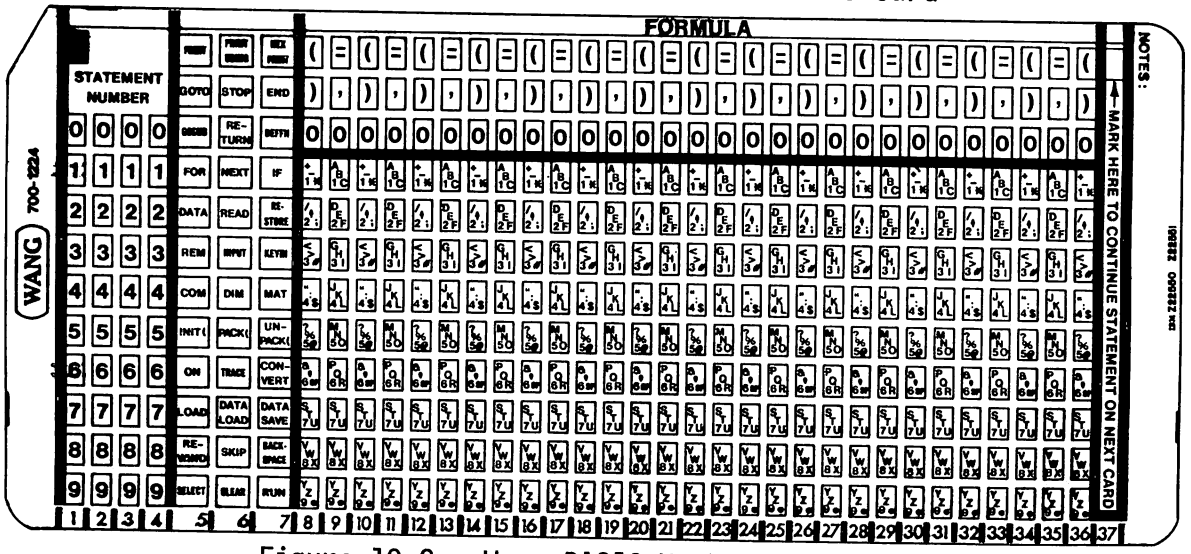


Figure 10-2. Wang BASIC Mark Sense Card

Both types of BASIC mark sense cards are designed to facilitate the marking of programs and data. Both cards have only 37 columns, fewer than half as many as a standard 80-column card. Each column is therefore wide enough to be easily identified and marked. Similarly, both cards contain lists of BASIC verbs which permit the user to indicate a complete verb with a single mark, rather than spelling it out in a series of columns.

Standard Format BASIC Cards

The standard format BASIC card is commonly used in educational institutions. As Figure 10-1 illustrates, it consists of 37 columns divided into four zones. Zone 1, consisting of columns 1-4, is used to indicate a program line number. Zone 2, comprising columns 5 and 6, contains a list of BASIC statements and commands. Zone 3, the FORMULA zone, comprises columns 7-36, and is used for text. Zone 4, finally, consists of the single column 37, and is marked to indicate continuation of a program line or data value onto a second card.

Zone 2 (BASIC commands and statements) contains a list of the following four commonly used BASIC commands:

SCRATCH*	LIST
RUN	PUNCH*

Zone 2 also contains the following commonly used program statements:

LET	READ
DATA	PRINT
GOTO	IF
FOR	NEXT
DIM	END
DEF	GOSUB
RETURN	STOP
REM	RESTORE
MAT*	COM

*SCRATCH is interpreted as CLEAR when read by the system, and PUNCH is interpreted as DATASAVE. The MAT statement can be used only in systems which support matrix operations.

NOTE:

Standard BASIC cards are not sold by Wang Laboratories.

Wang BASIC cards

The Wang BASIC mark sense card is essentially identical to the standard format card discussed above, with the exception that Zone 2 on the Wang card contains an additional column of BASIC verbs to include verbs from the expanded BASIC language available on most Wang systems. Zone 2 on the Wang card includes the following list of 36 commonly used statements and commands:

PRINT	PRINTUSING	HEXPRINT
GOTO	STOP	END
GOSUB	RETURN	DEFFN
FOR	NEXT	IF
DATA	READ	RESTORE
REM	INPUT	KEYIN
COM	DIM	MAT
INIT(	PACK(	UNPACK
ON	TRACE	CONVERT
LOAD	DATALOAD	DATASAVE
REWIND	SKIP	BACKSPACE
SELECT	CLEAR	RUN

NOTE:

Use only those statements and commands which are legal in your system. MAT, for example, cannot be used in a system which does not support matrix operations. INIT, PACK, CONVERT, etc. are illegal in a System 2200A. Consult your reference manual for a list of the BASIC statements and commands legal in your system.

10.2 MARKING PROGRAMS AND DATA ON BASIC MARK SENSE CARDS

The BASIC mark sense cards can be used to record either programs or data. Observe the following procedure when marking the cards:

(a) Zone 1 (Line Number) - Digits in the program line number are indicated by marking the appropriate boxes with a #2 pencil or equivalent. Only marked columns are read and decoded; unmarked columns are ignored. Only one box may be marked in each column. The mark should be a heavy vertical line through the center of the box (see Figure 10-3).

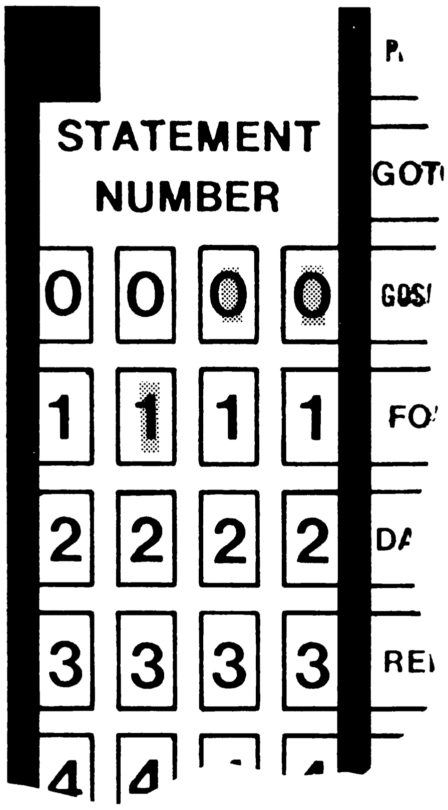


Figure 10-3.
Enlarged Portion of Zone 1 (STATEMENT NUMBER Zone) on a BASIC Mark Sense Card, Showing Line Number '100' Marked.

(b) Zone 2 (BASIC Verbs) - A BASIC verb is indicated by marking the appropriate box in Zone 2 with a vertical line through the center of the box. A #2 pencil should be used. In general, only one verb on each card can be marked from the list of BASIC verbs in Zone 2. Additional verbs on the same card must be spelled out in Zone 3. There is a single exception to this rule. On the Wang BASIC card, a SELECT INPUT Statement can be indicated by marking the SELECT box in column 5 and the INPUT box in column 6.

(c) Zone 3 (Formula Zone - Letters, Numbers, and Special Symbols) - Zone 3 comprises columns 7-36 on the standard format card, and columns 8-36 on the Wang Format card. Zone 3 is divided longitudinally by a heavy dark line into two sub-zones. In the three rows above the line, each box contains a single character. In the rows below the line, each box contains four characters. The characters and symbols in Zone 3 are marked as follows:

- To mark one of the characters in the upper three rows, mark only the box containing the desired character.

- Each box in the bottom nine rows contains a single digit printed in the lower left-hand corner. To mark one of the digits 1-9 in the bottom nine rows, mark only the box containing the specified digit.

- To mark one of the three letters or special symbols which run diagonally from the upper left-hand corner to the lower right-hand corner of each box in the bottom nine rows, mark the appropriate box plus a corresponding box in the top three rows of the same column. For the upper left character, mark the box in the top row; for the middle character, mark the second row from the top; for the lower right character, mark the third row from the top. (See Figure 10-4.) For example, to mark the character 'A', mark a box containing "A" in the lower nine rows, and the top box in the same column (the box containing '='). Similarly, for "B", mark the "B" box and the second box from the top in that column (the ',' box). (See Figure 10-5.)

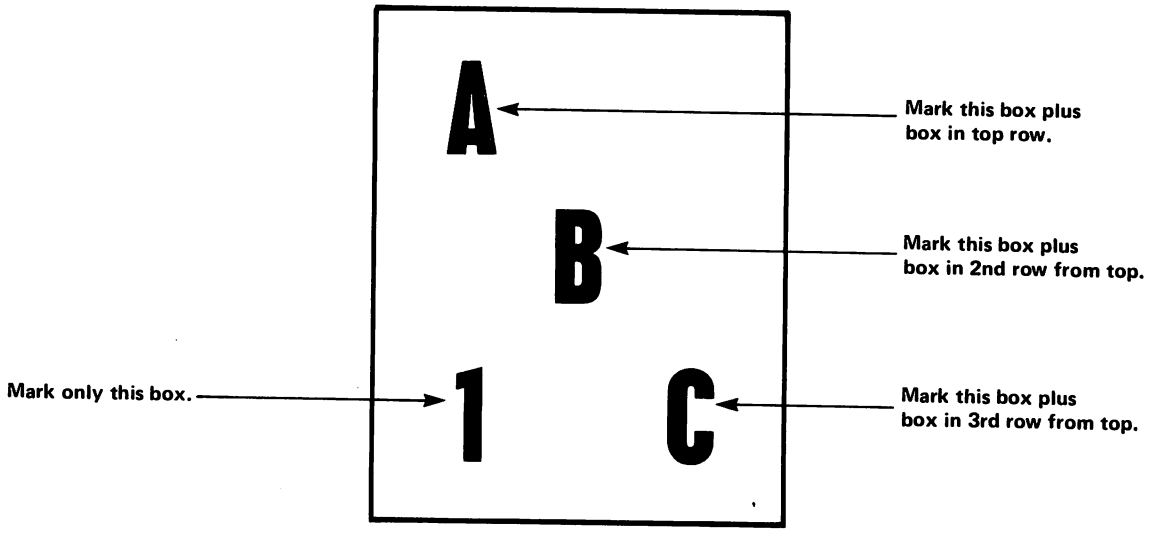

A
B
1
C
Mark this box plus box in top row.
Mark this box plus box in 2nd row from top.
Mark only this box.
Mark this box plus box in 3rd row from top.

Figure 10-4.
Enlarged view of Box in Row 1 of FORMULA Zone, showing which boxes must be marked to indicate specified characters.

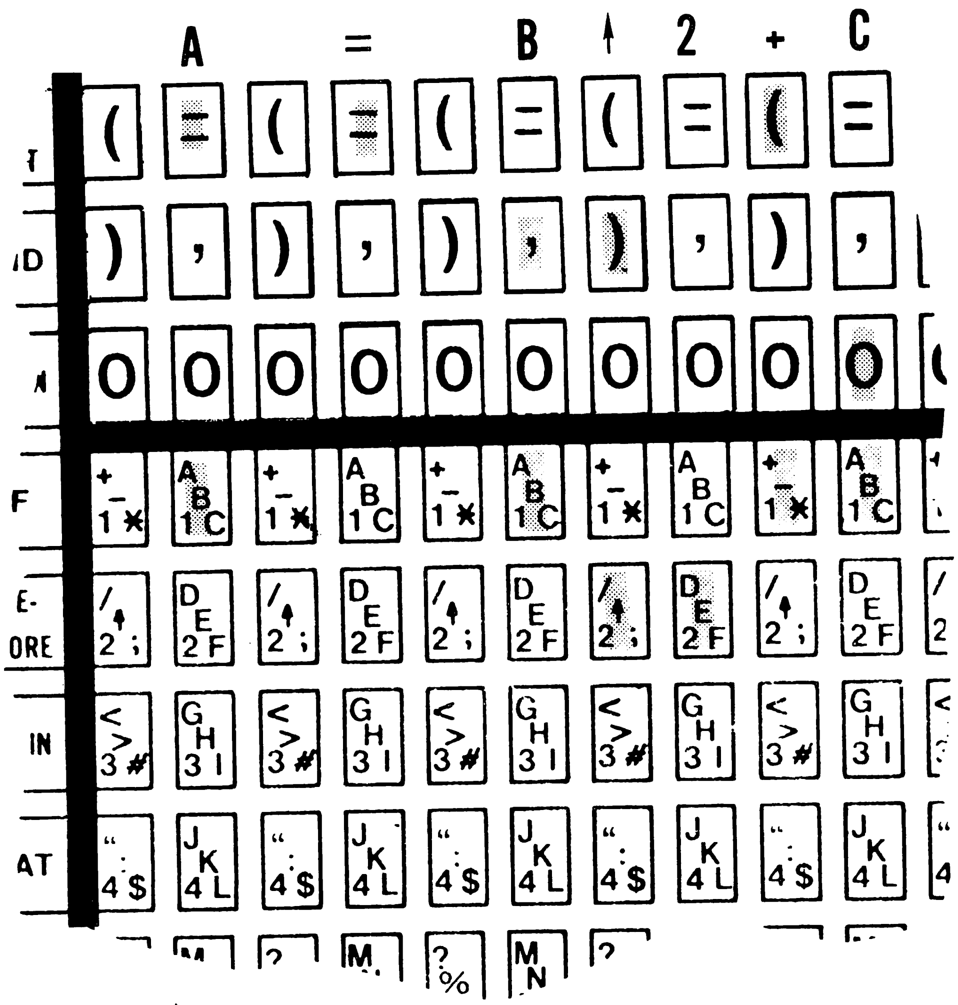


Figure 10-5.
Enlarged Portion of Zone 3 (FORMULA Zone) on a BASIC Mark Sense Card, Showing the Statement "A = B ↑ 2 + C" Marked.

NOTE:

The letters 'SP' in the lower right-hand corner of boxes in row 6 indicate an ASCII space character (HEX(20)). Since unmarked columns are ignored, spaces must be indicated by marking 'SP' and '0' in the desired column.

(d) NOTES SECTION - written comments may be placed in the NOTES section to identify the card for the programmer's convenience. Written material in this section of the card is ignored by the card reader.

CHAPTER 11
LOADING MARK SENSE BASIC PROGRAMS AND PROGRAM OVERLAYS
(LOAD, ADDRESS 62C)

11.1 INTRODUCTION

In Mode #7, BASIC programs can be loaded from both types of BASIC mark sense program cards with the LOAD instruction and a specified or selected device address of 62C. The same device address, 62C, is used with both Wang and standard BASIC mark sense cards. As each card is read, the Model 2244A automatically determines which type of card is being read, and utilizes the appropriate code conversion routine. (Wang BASIC cards have a special black box printed in column 1 to identify them.) The user is not therefore required to differentiate between the two types of cards; in fact, the two card formats may be intermixed in the same deck.

11.2 LOADING MARK SENSE BASIC PROGRAMS

A "program deck" is a deck of BASIC program cards which conform to the format conventions given in Section 11.4, and containing a series of program lines which collectively constitute a complete BASIC program. A "program overlay deck" is identical to a program deck, except that it typically constitutes only a segment or portion of a complete BASIC program, and is loaded in or "overlayed" under program control with a LOAD statement. The last card in a program or overlay deck must always be an END card, as described in Section 11.6, "End of Program".

BASIC mark sense program and program overlay decks are loaded into memory from cards with the LOAD instruction. The LOAD instruction has two forms, the LOAD command and the LOAD statement, distinguished by their mode of execution. When LOAD is executed in immediate mode (no line number), it is always interpreted as the LOAD command. When LOAD is executed in a program (on a numbered statement line), it is always interpreted as the LOAD statement. Typically, the LOAD command is used to load program decks, while the LOAD statement is used to load program overlay decks. The LOAD command is described in this section, the LOAD statement in the following section.

General Form: LOAD $\begin{bmatrix}/62C\\ \#n\end{bmatrix}$

Where:

/62C = The device address which designates the card reader as the device from which programs are to be loaded, and also determines the type of code conversion which is to be performed. Address 62C causes program text to be converted from the BASIC mark sense card code to ASCII.

#n = A file number to which the device address 62C has been assigned in a SELECT statement ('n' is an integer from 1 to 6).

If neither a device address nor a file number is specified, the address of the Console Tape device (normally, the address of the console tape cassette drive, 10A) is used. However, address 62C can, if it is convenient, be designated as the Console Tape address with a SELECT TAPE 62C statement. Following execution of this statement, any card reader statement with no specified device address or file number automatically uses address 62C.

Purpose:

The LOAD command with a device address of 62C initiates the reading of BASIC program cards, and automatically converts each character to ASCII. Newly-loaded program text is appended to the current program in memory, with new program lines which have the same line numbers as existing lines replacing the old lines in memory. Otherwise, existing program text in memory is unaffected by the LOAD operation. For example, if the old program in memory has line numbers 10,20,30, etc., and the newly loaded program has line numbers 15,25,35, etc., the resultant program in memory following the LOAD is numbered 10,15,20,25,30, etc. Lines which contain syntax errors are loaded and displayed with an appropriate error code. The LOAD operation is not terminated, but the program cannot be run until all syntax errors are corrected. The last card in the program deck must be an END card; otherwise, the system continues attempting to load program lines.

To avoid the problem of including statement lines from the old program in the newly loaded program, the old program should be cleared from memory prior to the LOAD. After the new program has been loaded from cards, the operator must enter RUN, EXECUTE to run the program.

Example 11-1: Loading a BASIC Program Deck from Mark Sense Cards (LOAD Command, Address 62C)

```
CLEAR
LOAD/62C
```

In this example, old program text and variables are cleared from memory, and a new program is loaded from either type of BASIC mark sense program cards. After the new program is loaded, the operator must enter RUN, EXEC to run the program. The program must be terminated by an END card as the last card in the deck.

11.3 LOADING BASIC PROGRAM OVERLAY DECKS

General Form: LOAD $\begin{bmatrix}/62C,\\ \#n,\end{bmatrix}$ [L1, L2]

Where:

/62C = The device address which designates the card reader as the device from which programs are to be loaded, and also determines the type of code conversion which is to be performed. Address 62C causes program text to be converted from the BASIC mark sense card code to ASCII.

#n = A file number to which the device address 62C has been assigned in a SELECT statement ('n' is an integer from 1 to 6).

If neither a device address nor a file number is specified, the address of the device currently selected as Console Tape device (normally, the console tape cassette drive, address 10A) is used. If convenient, however, address 62C can be designated as the Console Tape address with a SELECT TAPE 62C statement. Following execution of this statement, any card reader statement which does not specify a device address or file number automatically uses address 62C.

L1 = The line number of the first line of resident program text to be cleared from memory prior to loading in the new program. When program loading is complete, the new program is automatically run from this line.

L2 = The line number of the last line of resident program text to be cleared from memory prior to loading in the new program.

Purpose:

The LOAD statement with address 62C initiates the reading and conversion of BASIC programs from BASIC mark sense cards. The LOAD statement must be executed on a numbered statement line (otherwise, it is interpreted as a LOAD command). When the LOAD statement is executed, it produces an automatic combination of the following operations:

STOP	-	Stop current program execution.
CLEAR P	-	Clear all resident program text, or that portion specified by lines L1 and L2. If one line number is specified (L1), all resident program text beginning with that line is cleared. If two line numbers are specified (L1, L2), all lines between and including those lines are cleared. If no line number is specified, all resident program text is cleared.

CLEAR N	-	Clear all non-common variables. Common variables are unaffected.
LOAD	-	Load BASIC program from cards, convert each character to ASCII. Stop loading when an END card is read.
RUN	-	Run the program, beginning at line L1, if specified, or at the lowest line number in memory, if L1 and L2 are not specified.

The LOAD statement is useful in loading program overlays because of its ability to automatically clear a specified portion of resident program text prior to loading, and to automatically execute the newly loaded program segment after loading. Non-common variables are cleared along with the resident program text, but common variables, which may be needed by successive overlays, are unaffected. The LOAD statement also may be used to load complete programs. In that case, no line numbers are included in the LOAD statement, so that all program text in memory is cleared before the new program is loaded.

Example 11-2: Loading a BASIC Program from BASIC Mark Sense Cards (LOAD Statement, Address 62C)

```
100 LOAD/62C
```

Statement 100 automatically clears all program text and non-common variables from memory, and loads in a new program from cards. Program text is automatically converted from the BASIC mark sense card code to ASCII. Loading stops when an END card is read. After loading, the new program is automatically run from the lowest statement line.

Example 11-3: Loading a BASIC Program Overlay from BASIC Mark Sense Cards (LOAD Statement, Address 62C)

```
500 LOAD/62C, 100, 500
```

When it is executed, statement 500 clears statement lines 100 through 500 inclusive from the resident program, along with all non-common variables. The new program is then read from cards, converted to ASCII, and stored. Loading stops when an END card is read. After loading, the new program is automatically run starting at line 100. (If there is no line 100 in the new program, an error is signalled.)

Note that if the LOAD statement includes one or two line numbers (L1, L2), the program to be overlayed must have a line number identical to the first line number specified in the LOAD statement (L1). Otherwise, an error is signalled when the system attempts to run the program from that line. Note, too, that the overlay deck must be terminated with an END card. Otherwise, the system continues reading cards until it encounters an END card, or hangs up with no more cards to read.

11.4 BASIC MARK SENSE PROGRAM CARD FORMAT

In general, program text entered from mark sense cards must conform to the conventions governing program entry from the keyboard. Additionally, the BASIC mark sense card formats impose certain restrictions on the marking of program text. The applicable conventions are listed below:

1. All blank (unmarked) columns are ignored; spaces must be indicated by a space character ('SP' and '0').

2. Columns 1-4 (Zone 1) are reserved for the line number. Since only marked columns are decoded (unmarked columns are ignored), the only requirement is that at least one column in Zone 1 be marked with a digit. The line number need not begin in column #1 (as it must on punched cards). Line number 5, for example, may be marked in any one of the four columns in Zone 1. In general, cards without line numbers produce unpredictable results, and should be avoided in this reading mode. There are, however, two exceptions to this rule:

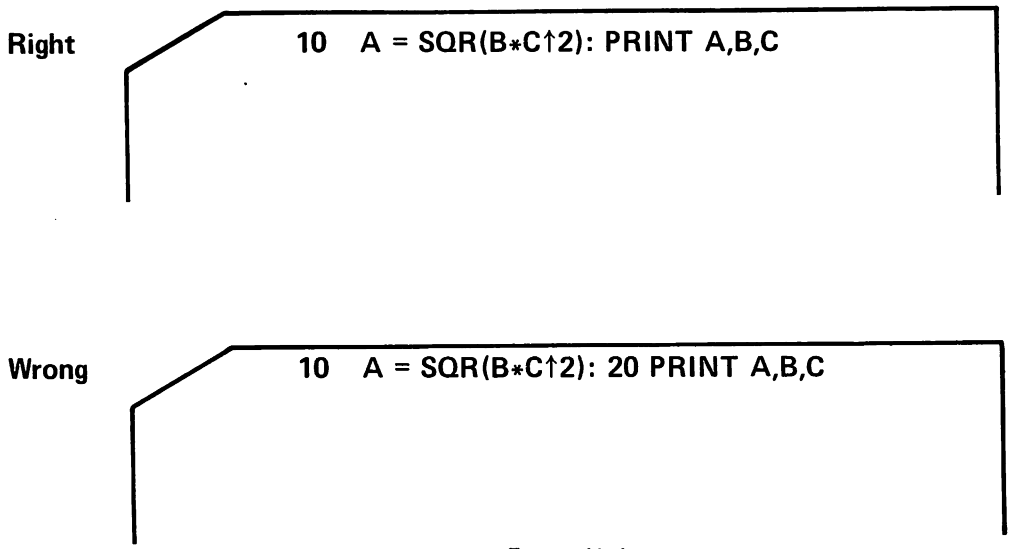
   (a) Immediate mode SELECT statements may be marked on a card without line numbers. The SELECT statements are executed immediately upon being read, thus permitting dynamic selection of device addresses during program loading.

   (b) The END statement may be marked on a card without an accompanying line number. The END card is always the last card in the program deck, and brings program loading to a halt immediately upon being read. It is generally good policy, however, to give the END statement a line number, so that it forms a part of the program stored in memory. Note that the END card must have the END box marked in Zone 2. It is not permissable to spell out the word "END" in the Formula Zone.

3. Columns 5-6 on the standard BASIC card, and 5-7 on the Wang BASIC card (Zone 2), are the BASIC verb columns. Only one box in each column may be marked. Unmarked columns are ignored. In general, only one BASIC verb in Zone 2 can be marked on each card. Additional verbs on the same card must be spelled out in the FORMULA columns (Zone 3). There is a single exception to this rule. On the Wang BASIC card, it is possible to mark a SELECT statement by marking the 'SELECT' box in column 5 and the 'INPUT' box in column 6. The selected device address must, of course, be spelled out in Zone 3.

4. Alphanumeric data, special symbols, etc., as well as additional BASIC verbs, are marked in the FORMULA section (Zone 3). Unmarked columns are ignored.

5. Multi-statement lines on a card are legal, provided the individual statements are separated by colons (:). However, only one line number may appear on each card. A single numbered line may consist of several statements (appropriately separated by colons), but an attempt to record more than one line number on the same card results in a reading error.



```
10   A = SQR(B*C↑2): PRINT A,B,C
```

```
10   A = SQR(B*C↑2): 20 PRINT A,B,C
```

Figure 11-1.
Right and Wrong Methods of Marking Multiple Statement Lines on a Single Card. (Multiple statements are legal if separated by colons; multiple line numbers are not. The second card produces an ERROR 31 [Illegal Line Number] when read. Note that BASIC verbs 'PRINT' and 'NEXT' must be spelled out in the alphanumeric field of Zone 3.)

6. The last (37th) column contains the 'CONTINUE' box. This box is marked to indicate continuation of a program line onto a second card. Refer to Section 11.5, "Continuation of a Program Line".

7. Each program or program overlay must be ended with an END card. The END card must have the "END" box marked. It may also have a line number, but all other columns should be left blank.

8. Blank (unmarked) cards are ignored in Mode #7.

11.5 CONTINUATION OF A PROGRAM LINE

In the normal case, each program line occupies a single card. An unusually long line may, however, be continued onto one or more additional cards. In order to indicate that a line is to be continued onto a subsequent card, the CONTINUE box on the first card (column 37) must be marked. If the program line occupies more than two cards, the CONTINUE box must be marked on every card except the last. The CONTINUE must not be marked on the last card. Nothing may be marked in the line number rows (Zone 1) of any card except the first. BASIC verbs (Zone 2) may, however, be marked on any continued card. The maximum legal length of a single program line is 192 characters (about 6 cards).

11.6 END OF PROGRAM

The last card in a program deck must be the END card. On this card, the BASIC verb END must be marked, along with a line number, if desired; any marked columns in the FORMULA Zone are ignored. An END card must be included as the last card in each program deck, since it is the only legal way to terminate the program loading operation. If the END card is omitted, the system continues loading program text beyond the last valid program card, with results which cannot be foreseen.

If a program and several program overlays and data decks are to be read from cards, the main program deck should be loaded in the input hopper first (and followed by an END card). Program overlay and data decks should then be loaded into the hopper in the order in which they will be read or called in by the main program. Each overlay deck also should have an END card as the last card in the deck.

Note that in Wang BASIC, it is possible to place an END statement anywhere in a program prior to the actual end of the program. When the END statement is encountered during program execution, it stops program execution and causes the END PROGRAM and FREE SPACE messages to be printed or displayed. END can be inserted at any point in a BASIC program loaded from cards; it will not terminate the LOAD operation if it is assigned a line number and is spelled out in the FORMULA zone rather than marked in Zone 2. However, whenever the the reader encounters a card on which the END box has been marked in Zone 2, it automatically terminates program loading (whether or not the END card has a line number). In general, it is recommended that only a single END statement be employed at the end of the program deck.

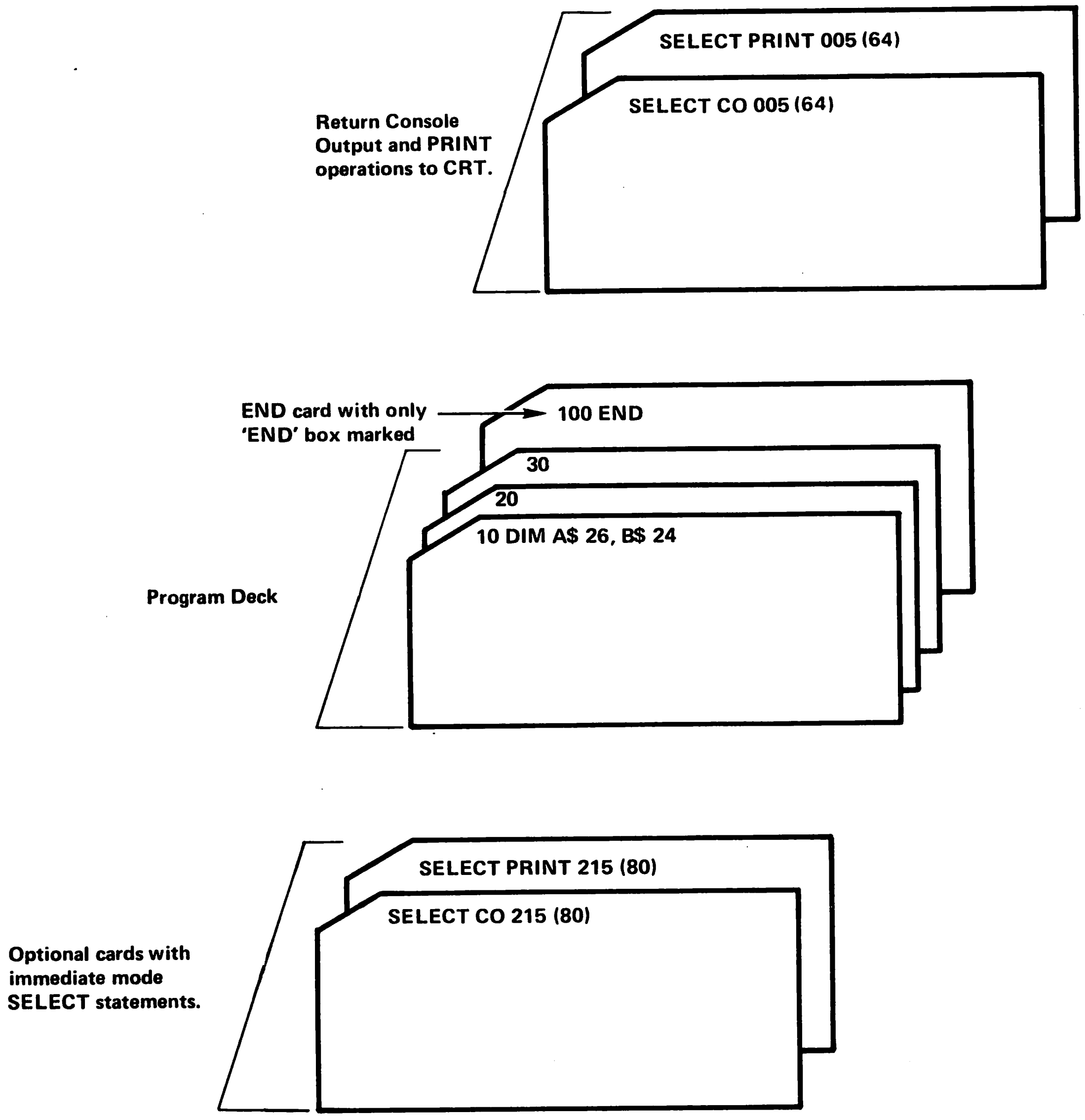

SELECT PRINT 005 (64)
SELECT CO 005 (64)
Return Console Output and PRINT operations to CRT.
END card with only 'END' box marked
100 END
30
20
10 DIM A$ 26, B$ 24
Program Deck
SELECT PRINT 215 (80)
SELECT CO 215 (80)
Optional cards with immediate mode SELECT statements.

Figure 11-2.
Typical Card Deck Arrangement for Single Mark Sense Program Deck Read with LOAD.

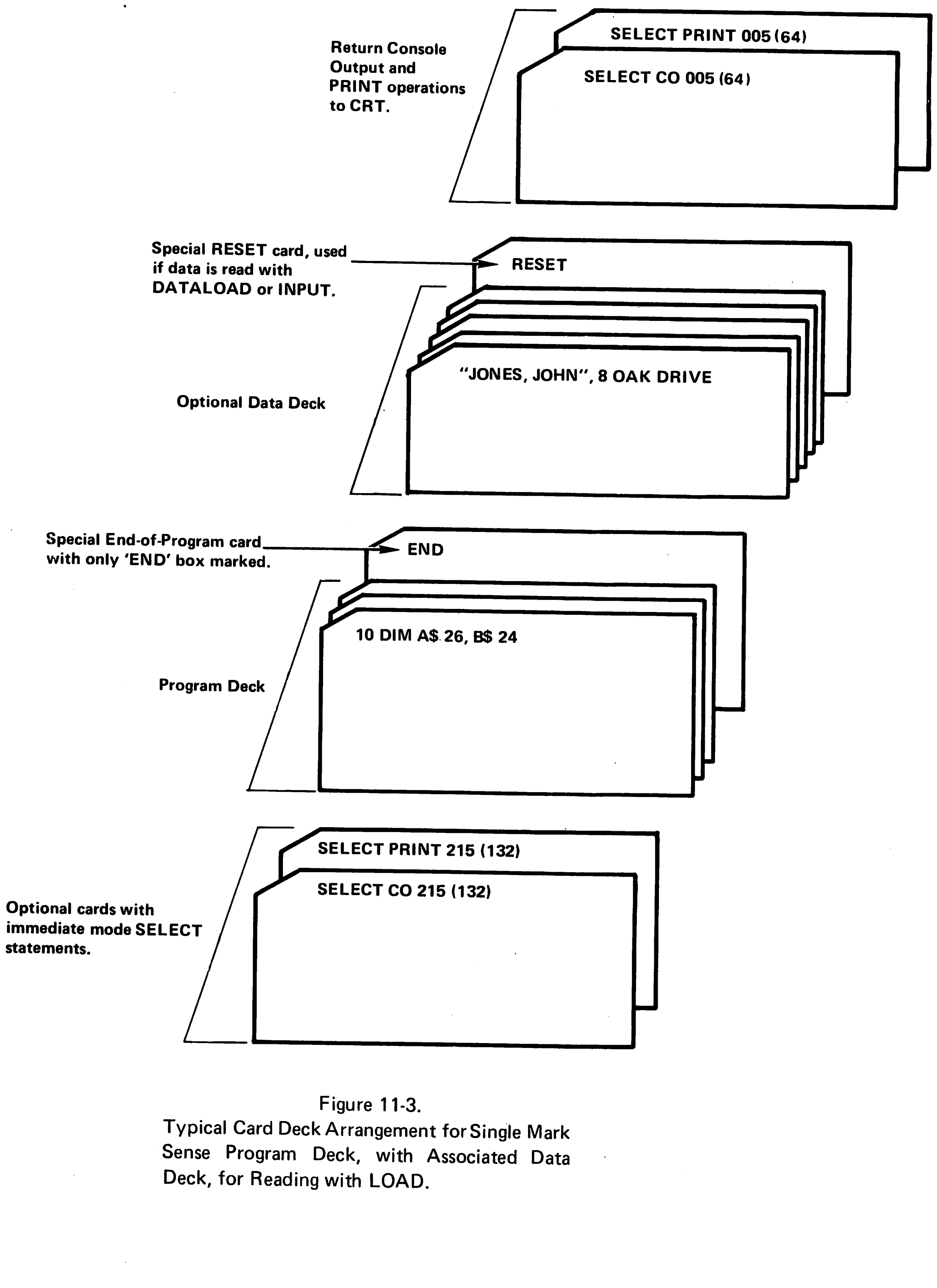

Return Console Output and PRINT operations to CRT.
SELECT PRINT 005 (64)
SELECT CO 005 (64)
Special RESET card, used if data is read with DATALOAD or INPUT.
RESET
"JONES, JOHN", 8 OAK DRIVE
Optional Data Deck
Special End-of-Program card with only 'END' box marked.
END
10 DIM A$ 26, B$ 24
Program Deck
SELECT PRINT 215 (132)
SELECT CO 215 (132)
Optional cards with immediate mode SELECT statements.

Figure 11-3.
Typical Card Deck Arrangement for Single Mark Sense Program Deck, with Associated Data Deck, for Reading with LOAD.

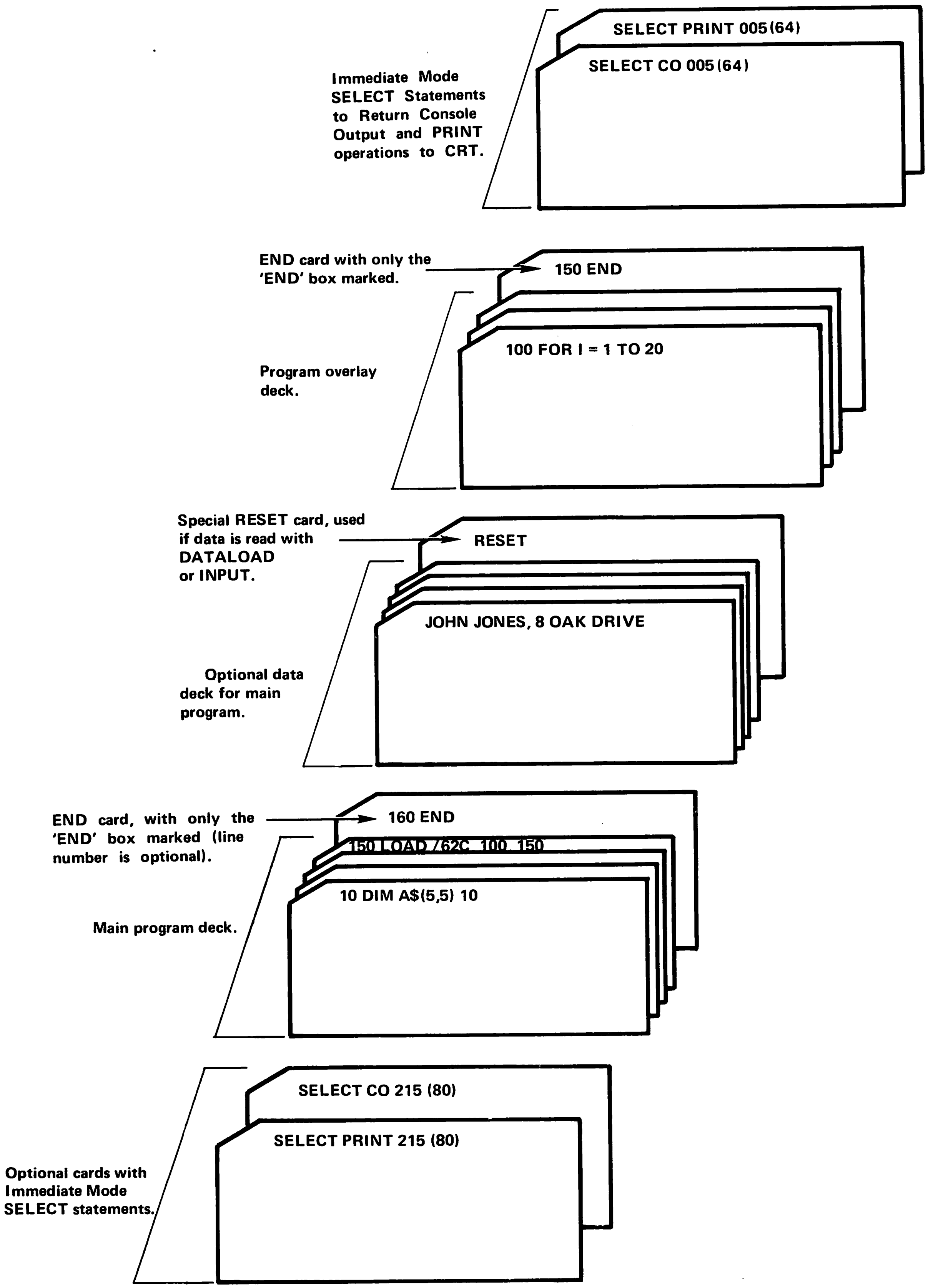

SELECT PRINT 005(64)
SELECT CO 005(64)
Immediate Mode SELECT Statements to Return Console Output and PRINT operations to CRT.
END card with only the 'END' box marked.
150 END
100 FOR I = 1 TO 20
Program overlay deck.
Special RESET card, used if data is read with DATALOAD or INPUT.
RESET
JOHN JONES, 8 OAK DRIVE
Optional data deck for main program.
END card, with only the 'END' box marked (line number is optional).
160 END
150 LOAD /62C, 100, 150
10 DIM A$(5,5) 10
Main program deck.
SELECT CO 215 (80)
SELECT PRINT 215 (80)
Optional cards with Immediate Mode SELECT statements.

Figure 11-4.
Typical Card Deck Arrangement for Mark Sense Program Deck with Program Overlay Deck and Associated Data Decks for Reading with LOAD.

CHAPTER 12
READING DATA VALUES FROM BASIC MARK SENSE CARDS (DATALOAD, ADDRESS 62D)

12.1 INTRODUCTION

Two statements are available for reading data values from BASIC mark sense cards: DATALOAD and INPUT. The required data card format is identical for both statements; thus, the same data decks may be read with either statement. DATALOAD is the preferred statement in most cases, however, because it is somewhat more predictable than INPUT in its handling of erroneous data values. DATALOAD immediately halts program execution and displays an error message when it encounters an illegal data format. (Note that in systems with the ON ERROR GOTO statement, it is possible to program a response to such error conditions without terminating program execution.) INPUT displays an error message, then ignores the erroneous value and proceeds to read the next card. There are, however, certain applications in which INPUT may be quite useful. INPUT is discussed in the following chapter, Chapter 13. The present chapter is devoted to an examination of the DATALOAD statement.

NOTE:

Two types of BASIC mark sense cards, standard format cards and Wang format cards, can be read with DATALOAD. An identifying black box printed in the upper left-hand corner of each Wang BASIC mark sense card enables the card reader to distinguish between a Wang card and a standard BASIC mark sense card as the card is read, and to automatically initiate the appropriate code conversion routine. For this reason, the user need not differentiate the two types of cards; he may, in fact, intermix both types of cards in the same deck.

12.2 READING DATA VALUES FROM BASIC MARK SENSE CARDS

General Form: DATALOAD $\begin{bmatrix}/62D,\\ \#n,\end{bmatrix}$ argument list

Where:

/62D =	The device address which designates the card reader as the device from which data is to be read, and also determines the type of code conversion routine which is to be performed. Address 62D causes a conversion from the BASIC mark sense card code into ASCII.
#n =	A file number to which the device address 62D has been assigned in a SELECT statement ('n' is an integer from 1 to 6). If neither a device address nor a file number is specified, the address of the default tape device (normally the console cassette drive, address 10A) is used. If convenient, it is possible to designate address 62D as the Console Tape address with a SELECT TAPE 62D statement. Following execution of such a statement, a DATALOAD statement which specifies neither a device address nor a file number automatically uses address 62D.
argument list =	The list of receiving variables, array elements, and/or array designators, separated by commas. (Array designators are array names followed by closed parentheses, e.g., A$(), N().)

Purpose:

The DATALOAD statement with a device address of 62D initiates the reading of discrete data values from one or more BASIC mark sense data cards, converts each character to ASCII, and assigns the values read sequentially to receiving variables in the DATALOAD argument list. (Arrays are filled row by row.) Numeric values must be in free-format (see below). Multiple values on a single card must be separated by commas. If the argument list is not filled by a single card, additional cards are read until all receiving variables are filled. Unread data on the last card is lost. Both alphanumeric and numeric values may be stored in alphanumeric variables, but only legitimate BASIC numbers can be stored in numeric variables (otherwise, an error results and program execution is terminated).

Example 12-1: Reading Data Values from BASIC Mark Sense Cards (DATALOAD, Address 62D)

```
10 DATALOAD/62D, A, B$, C$
```

Statement 10 reads data values from BASIC mark sense cards and assigns the values read sequentially to receiving variables in the argument list. Since the argument list specifies three variables, a numeric (A) and two alphanumerics (B$ and N$), the reader attempts to read one numeric and two alphanumeric values from one or more cards. The values may be on three separate cards, or on a single card (separated by commas). Note that an alphanumeric value cannot be read into a numeric variable; any attempt to do so results in an Error 43 (Wrong Variable Type). A numeric value may, however, be assigned to an alphanumeric variable.

It is possible to designate entire arrays as arguments in a DATALOAD argument list. In that case, each element of the array receives a separate value from the card. Arrays are filled row by row.

Example 12-2: Reading Data Values from BASIC Mark Sense Cards into Arrays (DATALOAD, Address 62D)

```
100 DIM A$(4,4)10, B(5)
.
.
.
200 SELECT #3 62D
210 DATALOAD #3, N$, A$(), B()
```

In this example, a single alphanumeric value is first read into alpha variable N$. Next, 16 alphanumeric values (each 10 characters or less in length) are read into alpha array A$(). Finally, five numeric values are read into array B(). As many cards as are necessary to satisfy the argument list are read. Note that the device address 62D, which indicates that data values are to be read from BASIC mark sense cards, is selected to file number #3 at line 200, and #3 subsequently is referenced in the DATALOAD statement at line 210.

12.3 BASIC MARK SENSE DATA CARD FORMAT

Data values read from BASIC mark sense cards in Mode #8 must be marked in the alphanumeric data columns (the FORMULA section, Zone 3) of each card, in a free format identical to the format in which data is entered from the keyboard. The following rules apply to the preparation of data on mark sense cards:

1. On BASIC mark sense cards used to store data values, the line number columns (Zone 1, columns 1-4) should be left blank (unmarked).

2. On cards used for data, the BASIC verb section (Zone 2) also should generally be left blank. If, however, a number of program and data decks are combined in a single job stream, the REM box on each data

card can be marked. This is a safety precaution which will prevent the occurrence of unforeseen errors if the data cards are inadvertently read as program cards under LOAD control. REM statements without line numbers are ignored by LOAD. The REM box does not affect legitimate data reading, since it is also ignored by DATALOAD.

3. Blank (unmarked) columns are always ignored by the reader and are never interpreted as space characters in Mode #8. Space characters must be specified by marking the 'SP' box in row 6, and the corresponding '0' box in the same column.

4. A varying number of data values can be marked on a single card. To separate individual values, a comma (',') character is marked on the card following each value; the comma separator must not be marked following the last value on a card, however.

5. Each DATALOAD statement can read one or more cards. If one card does not contain enough data values to fill all receiving variables and arrays in the DATALOAD argument list, the next card is automatically read. When, however, the last receiving variable in the DATALOAD argument list has received a value from a card, any additional data values on that card are ignored. A new DATALOAD statement begins by reading in the next card, even if there are remaining unread values on the previous card.

6. A combination of numeric and alphanumeric values can be marked on the same card (but individual values must be separated by commas). Alphanumeric values on a card must correspond in type to the receiving variables or arrays in the DATALOAD argument list. Although numeric values can be read into alphanumeric as well as numeric variables, alphanumeric values cannot be stored in numeric variables or arrays. Any attempt to do so results in an Error 43 (Wrong Variable Type), and program execution is halted. Note that this condition can be handled under program control in systems which provide an ON ERROR GOTO statement. Note, too, that the problem of illegal numeric format can be avoided by reading all values into alphanumeric variables, and subsequently testing and converting the numeric values. See Section 12.7 below for a discussion of this procedure.

7. Alphanumeric values may be recorded on cards with or without enclosing quotation marks:

   a) Without Quotes - If the first character of an alphanumeric value is not a quote (") character, all marked columns are read starting at the first character, up to a comma separator or the end of card. All unmarked columns are ignored. Leading and trailing space characters (so marked) also are ignored, but appropriately marked space characters embedded within the value are read. If a receiving variable or array element in the DATALOAD argument list contains insufficient bytes to store all characters of a value, the remaining unread characters of that value on the card are ignored. If the receiving argument contains more bytes than there are characters in the value read, the unfilled bytes in the variable are filled with space characters (HEX(20)).

b) With Quotes - If the first character of an alphanumeric value is a quote (") character, all marked columns are read beginning with the first character following the quote character, up to a second quote character. If the value is begun with a quote character, it must be ended with a quote character; otherwise, an Error 07 (Missing Quotation Marks) is indicated. All unmarked columns inside or outside the quotes are ignored. Commas enclosed within quotes are interpreted as data rather than separators. Legitimate space characters within the quotes, whether leading, trailing, or embedded, are read. Leading and trailing space characters outside the quotes are ignored, however. The quotes themselves are not read as part of the data. As with alphanumeric values marked without quotes, values in quotes which exceed the size of the receiving variable are truncated, and values which are smaller than the receiving variable are padded with trailing spaces up to the full length of the receiving variable.

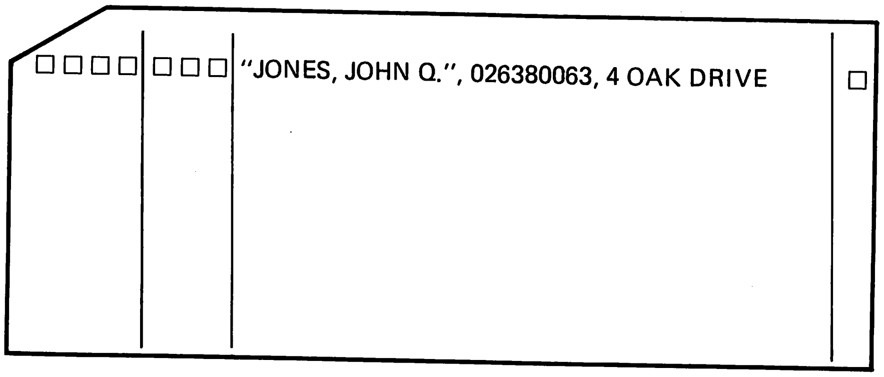


Figure 12-1.
Alphanumeric Values on a Wang Mark Sense Card With and Without Quotes. (Because the first value is enclosed in quotes, the embedded comma does not act as a data separator.)

8. Numeric values can be recorded in free format, in any numeric representation legal in a Wang system (e.g., 2.5, -973, 21.2 E-97, etc.) The conventions governing numeric free format are as follows:

   a) Optional plus (+) or minus (-) sign of number. If no sign is specified, the number is assumed positive. For negative values, a minus (-) sign must be specified.

   b) From one to 13 digits to represent the value of a number (e.g., 400125, -500), or the significant digits of a floating point number, with or without a decimal point (e.g., 1.45796E07, 24006E-11).

c) Optional two-digit exponent of floating point number, and optional sign of exponent (E ± XX). If no sign is specified, the exponent is assumed positive.

All unmarked columns and space characters (whether leading, trailing, or embedded) are ignored.

9. Each valid data character is read and automatically converted into ASCII. Codes which cannot be converted into legal ASCII characters are decoded as ASCII exclamation ('!') characters (HEX(21)). An exclamation character does not produce an error if read into an alphanumeric variable; it will, however, result in an error if an attempt is made to read it into a numeric variable.

10. The last (37th) column on the card is reserved exclusively for the special 'CONTINUE' box. The 'CONTINUE' box is used to indicate continuation of a numeric or alphanumeric value onto a subsequent card.

11. Blank cards are ignored by the system in Mode #8.

12.4 CONTINUATION OF A DATA VALUE

In most cases, a single mark sense card contains one or more whole data values, appropriately separated by commas. Occasionally, however, it may be necessary to extend a single value from one card onto a second card. In such cases, the special CONTINUE box in column 37 should be marked to indicate that the last value on the card will overlap onto a second card. The following conventions apply:

1. Numeric values - To continue a numeric value from one card to another, simply mark the CONTINUE box in column 37 of the first card, and continue the value in the FORMULA section (Zone 3) on the second card. For example, the numeric value '80001624E07' is continued from one card to a second in the following way:

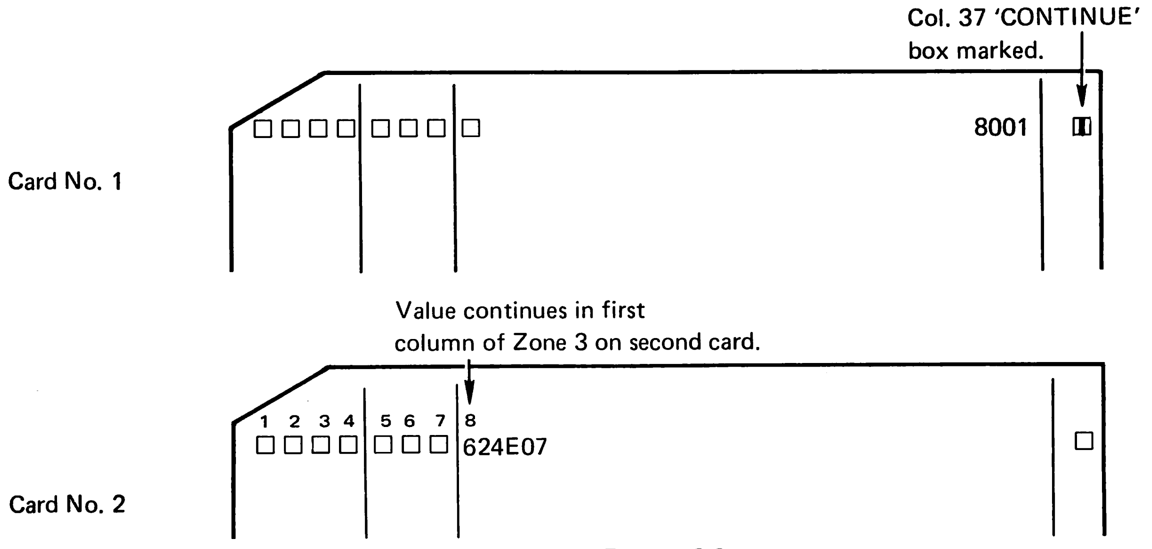

Col. 37 'CONTINUE' box marked.
8001
Card No. 1
Value continues in first column of Zone 3 on second card.
1 2 3 4 5 6 7 8
624E07
Card No. 2

Figure 12-2.
Continuing the Numeric Value '8001624E07' onto a Second Mark Sense Card.

2. Alphanumeric values - To continue an alphanumeric value from one card to another, mark the CONTINUE box (column 37) on the first card. The following rules should be observed when continuing an alpha value:

   a) The character string being continued should be enclosed in quotes.

   b) The first column in the FORMULA section (Zone 3) of the second card should contain a quote (") character. This is to ensure that spaces and commas embedded within the character string are preserved when the second card is read.

For example, the alphanumeric value "Jones, John" is continued in the following way:

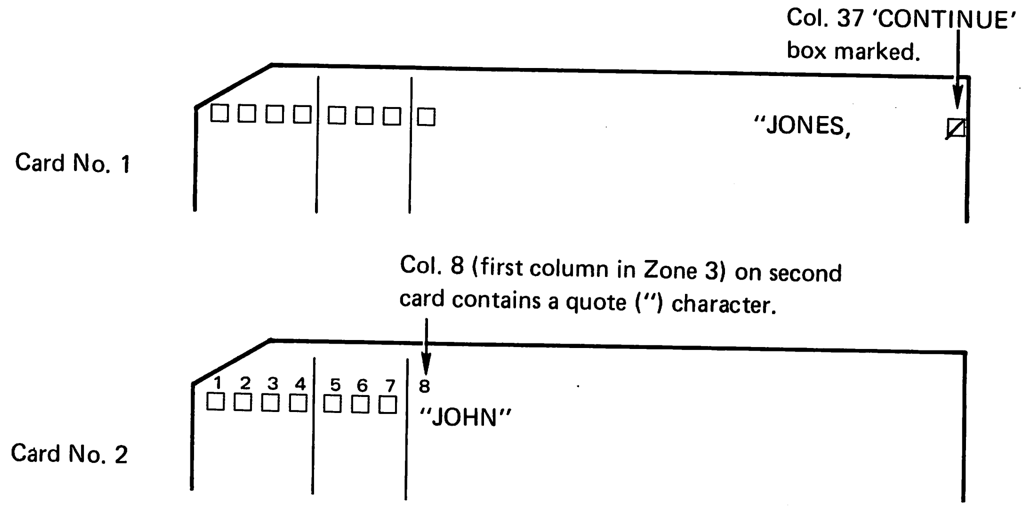

Col. 37 'CONTINUE' box marked.
Card No. 1
"JONES,
Col. 8 (first column in Zone 3) on second card contains a quote (") character.
1 2 3 4 5 6 7 8
"JOHN"
Card No. 2

Figure 12-3.
Continuing an Alphanumeric Value from One Mark Sense Card to a Second Card.

12.5 TESTING FOR THE END-OF-FILE

Typically, all data cards in a data deck have the same format - that is, the same number of data fields, in the same order. Each card (or a fixed number of sequential cards) is read with a DATALOAD statement (address 62D), the data is processed, and the program loops back to read the next card or sequence of cards. It is convenient in this case to be able to test for the end-of-file (i.e., no more cards to be read), since it is not always possible to know beforehand exactly how many cards are to be read. For this purpose, the user may design his own end-of-file card containing in the first field a special data value which would never appear as normal data. This special value is tested for in the program following each DATALOAD operation. Note that all other fields on the end-of-file card must be filled with dummy data values to satisfy the DATALOAD argument list. (See Figure 12-4.)

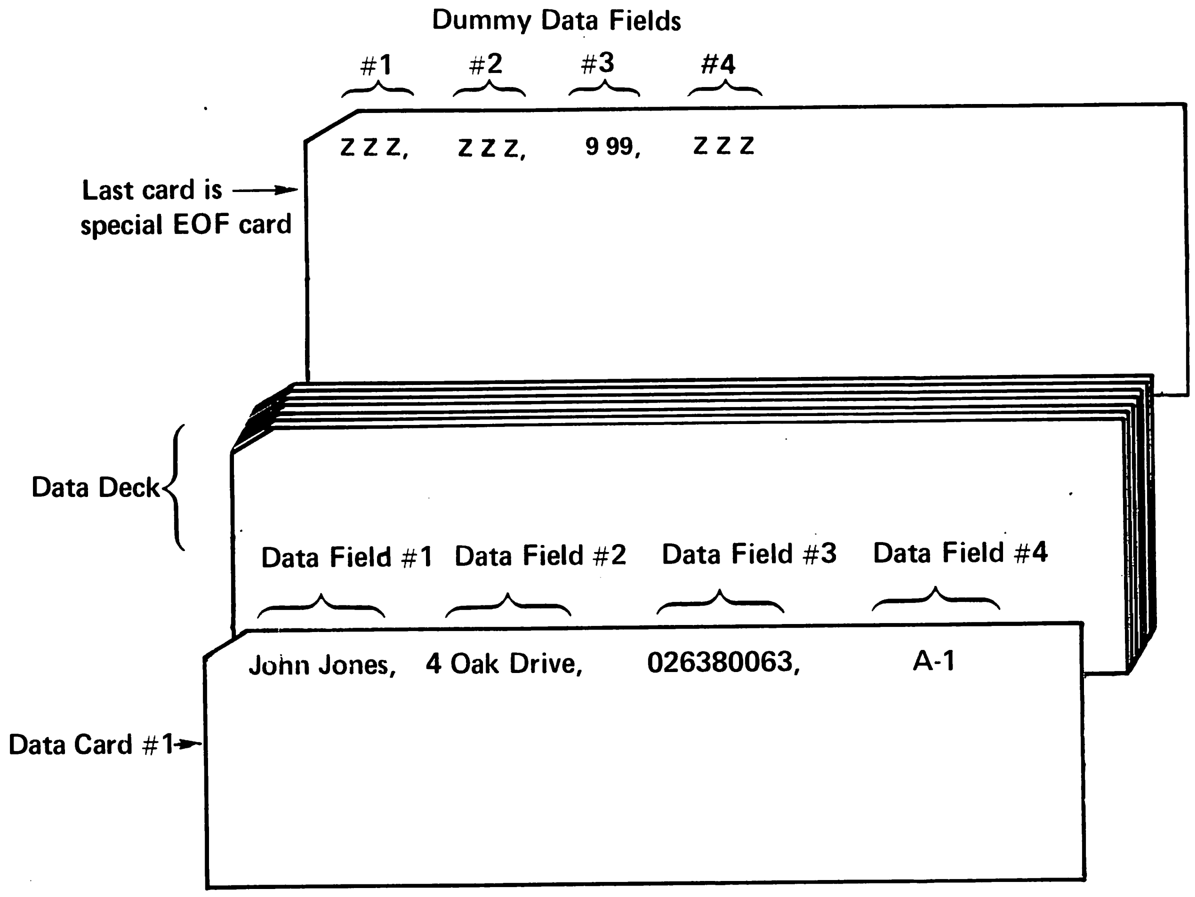


Figure 12-4.
Typical Data Deck with EOF Card Containing Dummy Data in All Fields.

Note that the dummy data fields must correspond in number and type (alpha or numeric) to the legitimate data fields. As each card is read, the controlling program checks for the presence of the first dummy data value. If the dummy value is not found, the card contains legitimate data, and normal processing resumes. If the dummy value is detected, the DATALOAD operation is terminated, and the program branches to another routine.

Example 12-3: Reading BASIC Mark Sense Values and Testing for the End-of-File (DATALOAD, Address 62D)

```
50 DATALOAD/628, A$, B$, N, F$
60 IF A$ = "ZZZ" THEN 160
.
.
.   (Process data)
.
.
.
150 GOTO 50
160 STOP
```

This example might be used to read the data deck in Figure 12-4, and test for an end-of-file condition. Each data card holds four data fields: data fields #1 and #2 are alphanumeric, and are read into alpha variables A$ and B$, respectively; data field #3, a numeric field, is read into numeric variable N; the fourth and last field, an alpha field, is read into F$. As each card is read, the system tests for end-of-file by checking for the dummy field "ZZZ" in A$. If the dummy EOF card is not detected, the data just read is processed until, at statement 150, the program is instructed to loop back and read in a new card. If the EOF card is read, the four dummy data fields are read into variables A$, B$, N, and F$, respectively, and the program skips down to statement 160 and stops.

12.6 THE SPECIAL "RESET" CARD

In programs which utilize a DATALOAD routine to read data cards, there is always the possibility that a logical error in the DATALOAD routine or a miscount of the number of data cards in the data deck may cause the system to continue attempting to read data cards after the data deck has been exhausted. If other cards follow the data deck, the program attempts to read these cards, an attempt which in most cases produces an error. If the data deck is the last or only deck in the input hopper, the system hangs up awaiting additional cards. Both of these situations can be avoided by inclusion of a special RESET card as the last card in the data deck.

The special RESET card is created by marking digits 2,3, and 4 in the first column of the STATEMENT NUMBER zone (Zone 1). All remaining columns on the card are then ignored by the reader. When the RESET card is read, it generates a system RESET condition (that is, it has the same effect as touching the RESET key on the keyboard). The RESET condition returns control to the currently designated Console Input device. If the card reader is currently selected for Console Input (with a SELECT CI 02D statement), it resumes processing cards under CI control. In this way, the system is prevented from either hanging up or reading into a subsequent program deck in the event of a card miscount or logical error in the DATALOAD routine. Note that if no such error exists, the RESET card does not interfere with normal processing.

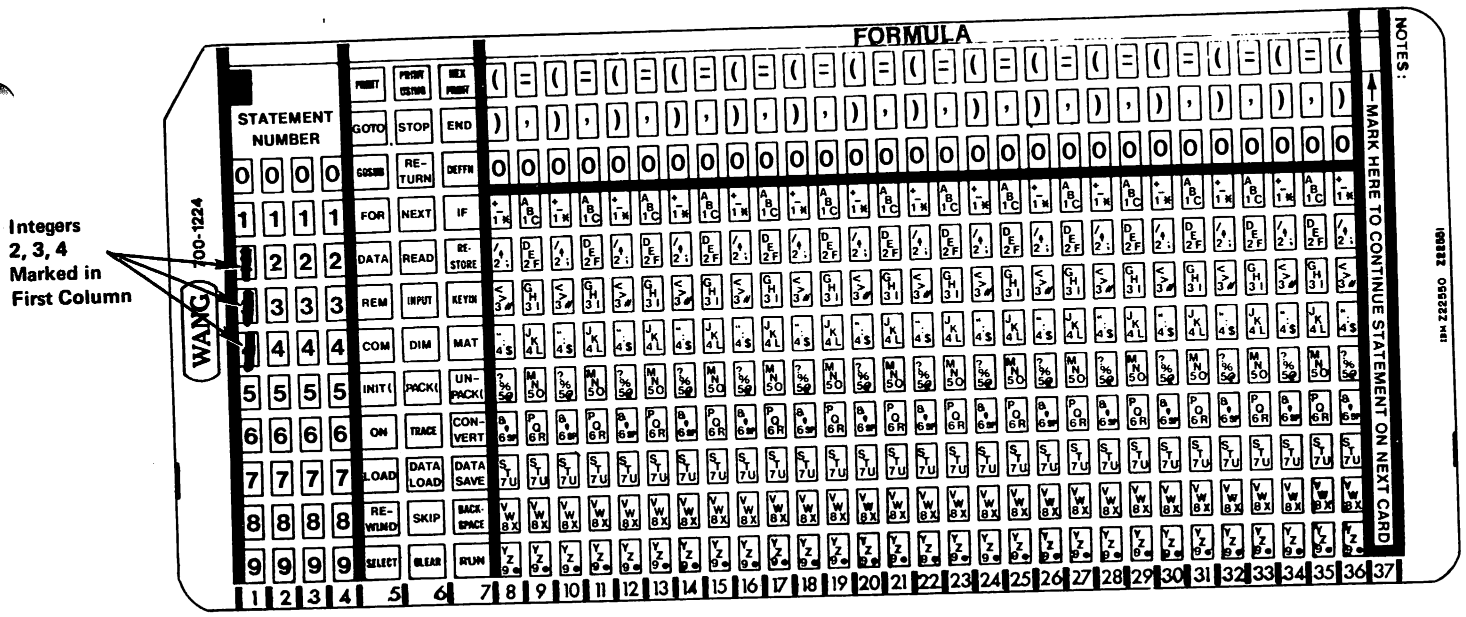

Integers 2, 3, 4 Marked in First Column
STATEMENT NUMBER
FORMULA
WANG
700-1224
MARK HERE TO CONTINUE STATEMENT ON NEXT CARD
NOTES:

Figure 12-5. Special RESET Card for BASIC Mark Sense Data Decks Read with DATALOAD

12.7 SPECIAL TECHNIQUES FOR PROCESSING NUMERIC DATA WITH 'DATALOAD'

The use of DATALOAD to read numeric data from cards presents special problems, because the presence of an invalid numeric value automatically causes the system to display an error message and hang up. In systems equipped with the ON ERROR GOTO statement, this problem can be handled under program control without terminating program execution. Where the ON ERROR GOTO statement is not available, however, it is recommended that numeric data be read into alphanumeric receiving variables, and tested for validity under software control with the NUM function. In this way, an invalid numeric will not automatically terminate program execution, and the programmer is free to respond to erroneous values in a manner most appropriate for his application. The general technique for testing numeric data stored in an alphanumeric variable, and converting it to numeric format, applies equally to data read via INPUT, and is described in detail in Section 13.4 of the INPUT chapter.

CHAPTER 13
READING DATA VALUES FROM BASIC MARK SENSE CARDS
(INPUT, ADDRESS 62C)

13.1 INTRODUCTION

Data values on BASIC mark sense cards can be read with an INPUT statement, if INPUT operations have been selected to the appropriate card reader address. Once INPUT operations have been selected to the card reader with a SELECT INPUT statement, data can no longer be entered from the keyboard in response to an INPUT request. For reading data values from BASIC Mark Sense cards, INPUT operations must be selected to address 62C with the following statement:

```
SELECT INPUT 62C
```

Like DATALOAD, INPUT is not recommended for serious data processing operations, principally because it does not provide the capability for adequate data control and verification. In particular, the use of INPUT poses potential problems in the handling of erroneous numeric data, since INPUT merely ignores an invalid numeric, and processes the next sequential data card. There are, however, several legitimate uses for INPUT:

1. On the System 2200A and on a System 2200S or WCS/10 System without Option-22, or -23, since these systems do not provide any other modes of entering data from cards. For a System 2200S or WCS/10 in which INPUT is the only available means of reading data from cards, it is recommended that all data, numeric as well as alphanumeric, be read initially into alphanumeric variables. Numeric data can be converted and stored into numeric variables after it has been checked for validity. (This technique is not practical on a 2200A, which does not provide the necessary statements for data verification and conversion.) See Section 13.4 below for an example of how this can be done.

2. For testing and debugging a program which normally requires a great deal of data entry from the keyboard. Programs designed to accept input from the keyboard can be altered with the addition of a SELECT INPUT 62C statement to read data from cards instead. A single data deck can be used repeatedly to check out the program, possibly saving considerable time in keyboard entry.

3. For educational purposes, as an introduction to card reader operations. Because of its similarity to keyboard entry operations, INPUT can serve as a relatively simple and familiar introductory method of entering data from cards.

13.2 DATA CARD FORMAT FOR 'INPUT'

The required format of data values read from mark sense cards with INPUT is identical to the required format of data values entered from the keyboard in response to an INPUT request. The INPUT format is also identical to the format required for cards read with the DATALOAD statement, discussed in Chapter 12. See Section 12.3 for a detailed list of format requirements. In brief, the format requirements are:

1. Multiple values on the same card must be separated by commas.
2. Alphanumeric values may or may not be enclosed in quotation marks. Embedded commas in alpha values which are enclosed in quotes are interpreted as data rather than as data separators.
3. Numeric values may be marked in any free-format legal in a Wang System.
4. Numeric values can be read into alphanumeric receiving variables, but alphanumeric values cannot be stored in numeric variables.
5. Numeric and alphanumeric values can be continued from one card to another if the 'CONTINUE' box is marked on the first card.
6. Carriage Return and X-OFF characters are illegal. (All illegal characters are automatically converted to exclamation ('!') characters.) Blank cards are ignored.

13.3 READING DATA VALUES FROM CARDS WITH 'INPUT'

```
GENERAL FORM:  INPUT ["character string",] variable [,variable...]
```

PURPOSE:

Once the card reader has been selected for INPUT operations with a SELECT INPUT 62C statement, the reader functions like a keyboard, reading one or more data cards in response to each INPUT request from the controlling program. Data values are read and sequentially assigned to the receiving variables in the INPUT argument list. Each value may be marked on a separate card, or multiple values may be marked on a single card, provided the values are separated by commas. Note, however, that each time the INPUT statement is executed, it automatically begins reading with the next card in the input hopper, even if there are remaining unread data values on the previous card. It is not therefore possible to use two or more INPUT statements to read several data values from a single card.

The INPUT statement operates in a manner substantially similar to that of DATALOAD, with one important difference. The DATALOAD statement displays an error message and terminates program execution when it encounters a format error in a numeric data value. With INPUT, however, a numeric format error generates an error code but does not terminate program execution. Instead, the erroneous value is ignored (along with all remaining values on the same card), and the INPUT request is repeated. The next data card is automatically read, and values from that card are sequentially assigned to the remaining unfilled variables in the INPUT argument list. INPUT is therefore somewhat unpredictable in its handling of erroneous data values.

There is one additional difference between INPUT and DATALOAD. Values entered via INPUT are automatically printed or displayed on the currently selected Console Output device (usually, the CRT) as they are read. This "echo" does not occur when values are read with DATALOAD; in that case, the value is printed or displayed only as the result of a PRINT statement in the program.

Note that a program which utilizes INPUT to read data from the card reader can be readily altered to accept data from the keyboard instead, simply by omitting the initial SELECT INPUT 62C statement (i.e., by not selecting the card reader for INPUT operations).

Example 13-1: Reading BASIC Mark Sense Data Values with INPUT (Address 62C)

```
 50 SELECT INPUT 62C
 60 INPUT A$
 70 IF A$ = "999" THEN 200
 .
 .  (Process data read at line 60)
 .
 .
190 GOTO 60
200 SELECT INPUT 001
210 STOP
```

In this example, the card reader is first selected for INPUT operations with the BASIC mark sense data card address for INPUT (62C). A single value is then read into alpha variable A$. Immediately following the read, A$ is checked for the value "999". The dummy value "999" is used in this case to signal the end-of-file (that is, no more cards in the deck). A card containing "999" must be included as the last card in the data deck read by this routine. If a value other than "999" is read, the program drops through to process the data, and, at line 190, loops back to read in the next value. If "999" is read, the program branches to line 200, where INPUT is selected back to address 001 (the keyboard), and the program stops.

13.4 SPECIAL TECHNIQUES FOR USING 'INPUT' TO PROCESS NUMERIC DATA

Although the INPUT statement is not recommended for serious data processing work, it is the only statement available on a System 2200A, and on a 2200S or WCS/10 system which does not have Option-22 or -23. In cases where INPUT must be used for data processing on a System 2200S or WCS/10, it is strongly suggested that alphanumeric variables be utilized to receive both numeric and alphanumeric values. In the normal case, when a format error is detected in a numeric value read into a numeric variable with INPUT, the system displays an error message, ignores the erroneous value and all subsequent values on the same card, and reads in the next card to satisfy the remaining variables in the INPUT argument list. If the cards contain a mixture of numeric and alphanumeric data, the values read no longer correspond to the appropriate receiving variables in the argument list, and the entire data deck may be read with errors generated for every card. A RESET card at the end of the data deck will prevent the program from reading into the next deck in the hopper, but the output of the current program will obviously be affected. If, however, all values on each card are read into alphanumeric variables, no format errors can arise. The program itself can be designed to check numeric values for validity before they are processed. If a format error is uncovered, appropriate action can be taken under program control. Note that this technique is not practical on a System 2200A, which does not provide the necessary language features for testing and converting numeric data.

Once the numeric value has been stored in an alpha variable, it can be tested for validity under program control using the NUM function. If it is found to be valid, it may be converted to numeric format with the CONVERT function, and processed in the usual way. If it is found to be invalid, the system can be instructed to print out an error message to the operator. In this way, the operator always knows which value is erroneous, and he is assured that any remaining data values in the data deck will not be affected by the detection of an erroneous value.

A numeric value punched or marked in Hollerith code in free-format may consist of a maximum of 19 characters, arranged in the following order:

Optional sign of number.
Maximum 13 digits in number.
Optional decimal point in number.
Optional 'E' character identifying exponent.
Optional sign of exponent.
Optional two-digit exponent.

For example, the following number consists of the maximum 19 characters:

```
-123456789.1234E-07
```

When this number is read into a numeric variable, it is automatically converted to Wang numeric format, and occupies only eight bytes. When it is read into an alphanumeric variable, however, each character occupies a single byte; a 19-byte alpha variable is therefore required. Once the numeric value is stored in an alpha variable, it is verified with a NUM function. NUM determines the number of legitimate numeric characters in the alpha variable. The plus (+) and minus (-) signs, decimal point, digits, the 'E' character, and spaces all are considered legitimate numeric characters. If the numeric value read consists of fewer than 19 characters (as it would in most cases), INPUT automatically pads the remaining bytes of the alpha variable with trailing spaces, which are accepted as numeric characters by NUM. Thus, the number of numeric characters returned by NUM should be 19 in every case; if fewer than 19 numerics are detected, at least one character is non-numeric, and the value should be rejected. Example 13-2 below illustrates a routine to test and convert numeric data entered with INPUT.

Example 13-2: Testing and Converting Numeric Data Entered Via INPUT
(All Systems Except 2200A)

```
10 DIM A$19, B$24, C$9
20 SELECT INPUT 62C
30 INPUT A$, B$, C$
40 IF NUM(A$)<19 THEN 150
50 CONVERT A$ TO N
.
.  (Normal Processing)
.
140 GOTO 30
150 STOP "ERRONEOUS VALUE"
```

In this example, three values are read from cards with each INPUT statement. The first value read is numeric, the second and third are alphanumeric. All three, however, are read into alphanumeric variables. Alpha variable A$ is dimensioned to 19 bytes in length so that it can contain the maximum possible number of characters in a single numeric value. Following the INPUT request, NUM is used to determine the number of valid numeric characters in A$. If there are fewer than 19 numeric characters (leading, trailing and embedded spaces are counted as numeric characters), the program branches down to line 150 and displays an error message. If A$ contains 19 valid numeric characters, its value is converted to numeric format and stored in numeric variable N, and normal processing continues.

13.5 CONTINUING DATA VALUES AND TESTING FOR THE END-OF-FILE

Continuation of a Data Value

A single data value may be continued onto one or more cards for reading with INPUT. In this case, the CONTINUE box is marked in the 37th column of each intermediate card containing the value (but not on the last card containing the value). For a more complete discussion of the continuation feature, refer to Chapter 12, Section 12.4, "Continuation of a Data Value."

Testing for the End-of-File

In cases where a series of data cards are read with an INPUT loop, it is generally desirable to test for a special end-of-file card indicating that all data cards in the deck have been read. Such a card is designed by the user to contain dummy data values in all fields. As each data card is read, the program checks to see whether one of the dummy data values was received. When the designated dummy value is read, the system knows that all data cards have been processed, and terminates the INPUT loop. The logic involved in creating and testing for an end-of-file card is the same for DATALOAD operations as for INPUT operations, and is described in detail in Chapter 12, Section 12.5, "Testing for the End-of-File."

13.6 THE SPECIAL 'RESET' CARD

In programs which utilize an INPUT routine to read data cards, there is always the possibility that a logical error in the INPUT routine or a miscount of the number of data cards in the data deck may cause the system to continue attempting to read data cards after the data deck has been exhausted. This problem can be avoided by inclusion of a special reset card as the last card in the data deck.

The special RESET card is created by marking digits 1,2, and 3 in column #1 of the STATEMENT NUMBER zone (Zone 1). All remaining columns on the card are ignored by the reader. When the RESET card is read, it generates a system RESET condition (that is, it has the same effect as touching the RESET key on the keyboard). The RESET condition returns control to the currently selected Console Input device, thus preventing the system from either hanging up or reading into a subsequent deck in the event of a card miscount or logical error in the INPUT routine. Note that if no such error exists, the RESET card does not interfere with normal processing.

The RESET card is illustrated at the end of Chapter 12.

CHAPTER 14
BATCH PROCESSING BASIC MARK SENSE PROGRAMS

14.1 INTRODUCTION TO "BATCH PROCESSING" ON THE MODEL 2244A

The term "batch processing" denotes an operation in which a series of discrete program decks and associated data decks are automatically loaded and run in sequence without normal user intervention. A "batch" of individual program and data decks are loaded into the input hopper, separated by system command cards such as CLEAR, LOAD, LIST, and RUN. The card reader is then selected for Console Input operations, and automatically begins loading in the first program deck. If the program requires data to be entered from cards, an accompanying data deck is automatically read. Thus, in effect, the card reader assumes the role normally taken by the keyboard. At any point where the system would normally expect a program line or system command to be entered from the keyboard, the card reader is accessed instead to automatically read the next card. Program lines are therefore loaded from cards just as if they had been keyed in from the keyboard, and system commands and immediate mode statements (such as LIST, LOAD, RUN, CLEAR, SELECT, etc.) are executed immediately upon being read from cards. When the first program has completed execution, the system command cards are read and executed, and the next sequential program is automatically loaded and run. Hardcopy output and listings for each program can be generated on a line printer or output writer by dynamically changing the LIST, PRINT and CO parameters with immediate mode SELECT cards. The process continues until all cards have been read and processed, at which point Console Input operations are reselected to the keyboard.

14.2 LOADING BASIC MARK SENSE PROGRAMS WITH CONSOLE INPUT

The principal value of Console Input mode derives from its usefulness in batch processing operations. On a System 2200A, however, and on a System 2200S or WCS/10 without Option-22 or -23, Console Input provides the only method of loading BASIC programs from cards. BASIC mark sense cards are read under Console Input control by selecting the card reader for Console Input operations with a device address of 02C:

```
SELECT CI 02C
```

Immediately after this statement is keyed in and executed, the card reader assumes all program entry operations normally associated with the keyboard. The following sequence of events then takes place:

1. If the card reader is in a ready condition, cards are read and automatically converted from the special mark sense card code to ASCII.

2. Numbered program lines are read from cards, displayed on the currently selected Console Output device (normally, the CRT), and entered into memory, just as if they had been entered from the keyboard.

3. Numbered program lines which contain syntax errors are printed or displayed with an appropriate error code, and entered in memory. Erroneous program lines do not terminate the program loading operation, but the program cannot be run until all errors have been corrected.

4. System commands (such as CLEAR, LOAD, LIST, RUN, etc.) are read from cards, printed on the currently selected Console Output device, and automatically executed, just as if they had been entered from the keyboard.

5. Immediate mode statements without line numbers (such as SELECT, PRINT, etc.) are read from cards, printed on the currently selected Console Output device, and automatically executed, just as if they had been entered from the keyboard.

The SELECT CI 02C statement is, of course, entered from the keyboard. Once it is executed, the keyboard is locked out, and all subsequent commands and statements must be entered from cards. Thus each program deck must be preceded and followed by control cards which perform operations normally initiated manually from the keyboard. A CLEAR command card and optional immediate mode SELECT statements generally are placed at the beginning of the program deck to clear memory, and to select specified output operations (such as PRINT, LIST, and/or Console Output)to a printer before loading in the program. (Note that if a CLEAR card follows a SELECT CO card, all PRINT and LIST operations, as well as Console Output operations, are defaulted to the printer whose address is specified in the SELECT CO statement. If, however, CLEAR precedes the SELECT CO card, the PRINT and LIST addresses are not altered.) If Console Output is selected to a printer, each card is automatically printed out as it is read, thus providing a hardcopy listing of the entire program deck. Program lines containing syntax errors are printed along with an identifying error code, thereby producing a useful "audit trail" for debugging purposes.

A program deck read under Console Input control must be followed by a RUN command card. After the last program card has been loaded, the RUN card is read, and automatically initiates program execution. The deck typically is followed by additional immediate mode SELECT cards which return output operations to the CRT. Finally, the last card in the job stream must be a SELECT CI 001 card which returns Console Input operations to the keyboard. Once this card is read, any remaining cards in the input hopper are ignored.

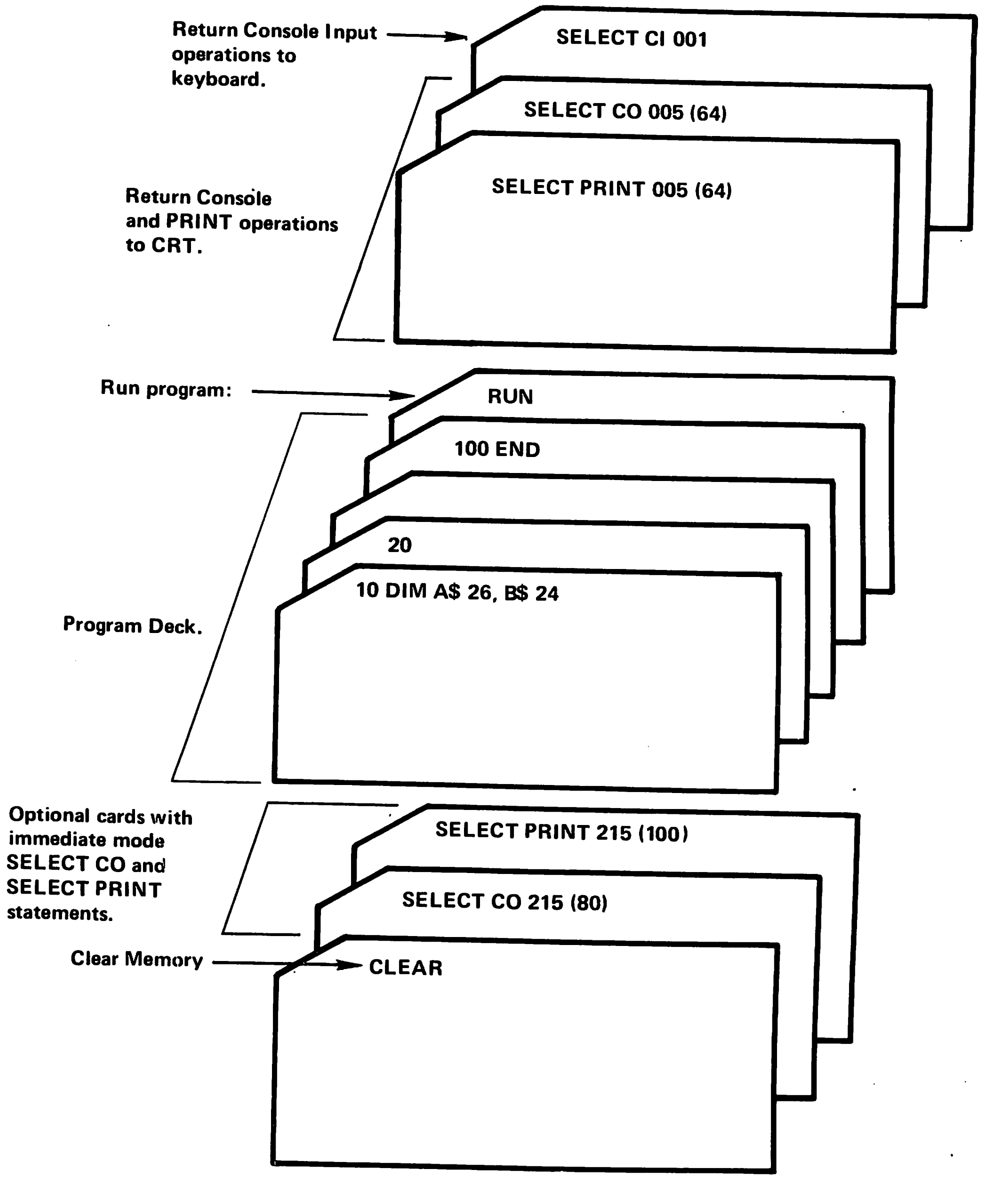


Figure 14-1.
Typical Card Deck Arrangement for Single Mark Sense Program Deck Read Via Console Input.

14.3 BASIC MARK SENSE PROGRAM CARD FORMAT FOR CONSOLE INPUT

The format of program lines read from cards with Console Input is identical to that of programs entered from the keyboard, and is also quite similar to the required format of programs read from cards with LOAD. Chapter 11, Section 11.4 ("BASIC Mark Sense Program Card Format") lists the applicable format conventions in some detail. In brief, they are:

1. The line number must be marked in Zone 1. Unmarked columns are ignored.

2. Multiple-statement lines are allowed, provided the statements are separated by colons. Multiple line numbers on the same card are not allowed, however.

3. A Carriage Return is not required to terminate a line on a card.

4. Codes which do not convert to legal ASCII codes are automatically converted to ASCII exclamation ('!') characters.

5. The last (37th) card column contains the CONTINUE box, used to continue a single statement line onto two or more cards.

6. Program lines containing syntax errors are read into memory, and the erroneous line is printed or displayed with an appropriate error code. Syntax errors do not terminate the program loading operation, but the program cannot be run until all errors are corrected.

7. Blank cards are ignored by the card reader.

14.4 DIFFERENCES BETWEEN 'CONSOLE INPUT' AND 'LOAD'

Although program cards read with both LOAD and Console Input must conform to the same format conventions (and therefore programs read with LOAD can in general be read with Console Input, and vice-versa), there are certain important differences between the two modes:

1. Each card read via Console Input is automatically "echoed" onto the currently selected Console Output device (typically the CRT display) as it is entered. This does not occur with cards read by LOAD. Thus the process of program loading is somewhat faster with LOAD than with Console Input.

2. Console Input will read and execute system command cards and immediate mode statement cards (i.e., cards which do not contain line numbers) such as CLEAR, RUN, LIST, SELECT, etc. In program decks read with LOAD, only the END card and immediate mode SELECT statements can be read and executed in immediate mode. Any other card which lacks a program line number may produce unpredictable results when read with LOAD.

3. Programs read with LOAD must be ended with an END card. Programs loaded via Console Input do not require an END card (although an END card is recommended as a general rule).

14.5 BATCH PROCESSING WITH CONSOLE INPUT

Because of its ability to read and execute system commands and immediate mode statements, Console Input is an extremely valuable reading mode for batch processing operations. A batched job stream consisting of several programs and associated data decks separated by appropriate system command cards can be loaded and run under Console Input control. Because commands such as CLEAR, LOAD, and RUN for each program are read from cards rather than entered from the keyboard, the entire job stream can be processed without user intervention (provided the programs are free of syntax errors). Each program is automatically loaded and run; when it has completed execution, it is cleared from memory, and the next program is loaded. This process continues until a card is read which reselects Console Input back to the keyboard. Figure 14-2 below illustrates a typical batched job stream with control cards.

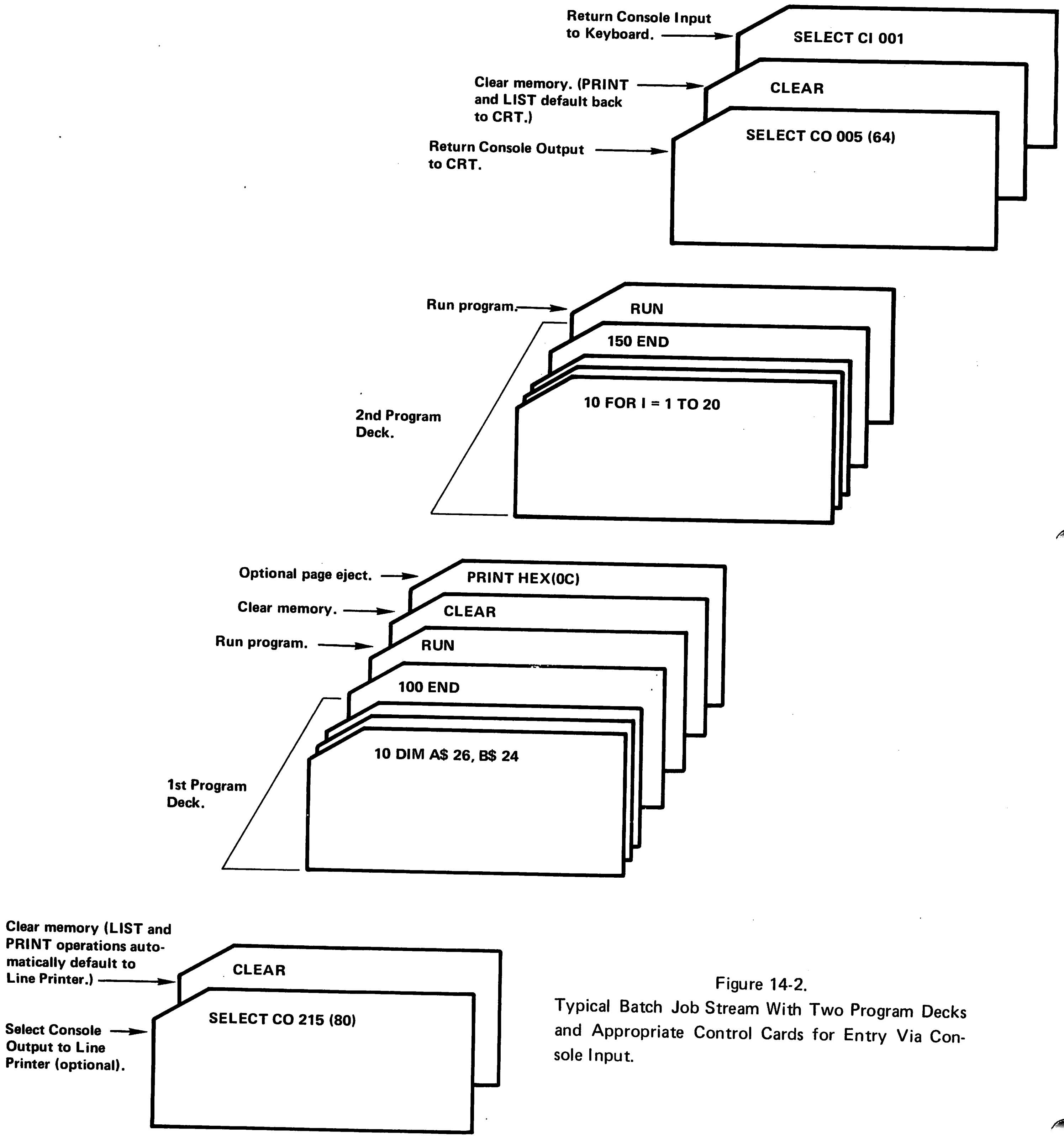

Return Console Input to Keyboard.
SELECT CI 001
Clear memory. (PRINT and LIST default back to CRT.)
CLEAR
Return Console Output to CRT.
SELECT CO 005 (64)
Run program.
RUN
150 END
2nd Program Deck.
10 FOR I = 1 TO 20
Optional page eject.
PRINT HEX(0C)
Clear memory.
CLEAR
Run program.
RUN
100 END
1st Program Deck.
10 DIM A$ 26, B$ 24
Clear memory (LIST and PRINT operations automatically default to Line Printer.)
CLEAR
Select Console Output to Line Printer (optional).
SELECT CO 215 (80)

Figure 14-2.
Typical Batch Job Stream With Two Program Decks and Appropriate Control Cards for Entry Via Console Input.

14.6 THE USE OF 'LOAD' IN CONJUNCTION WITH 'CONSOLE INPUT' FOR MORE EFFICIENT BATCH PROCESSING

Although overall control of a batched job stream must lie with Console Input, it is possible to speed up batch processing time by loading individual programs with the LOAD command. This can be done by adding a LOAD command card at the beginning of each program deck (the LOAD card must not have a line number). When the LOAD card is read and executed, it causes the succeeding program to be loaded in under LOAD control. In this case, program lines are not automatically echoed to the Console Output device as the cards are read; the program therefore loads in more rapidly than would be the case under direct Console Input control. Program loading continues until an END card is read, at which point control reverts to Console Input. If a listing is desired, an immediate mode LIST card must be inserted immediately after the END card at the end of the program deck (but before the RUN card).

The appropriate BASIC Mark Sense program address, 62C, must be specified for the LOAD operation. The address may be included directly in the LOAD command (e.g., LOAD/62C); alternatively, all tape-class operations (of which card reading is one) can be selected to the card reader with a SELECT TAPE 62C command. If this statement is executed prior to the LOAD command card, no address is needed in the LOAD command, since all tape-class operations automatically default to address 62C. The normal default address for tape-class operations is the console tape cassette drive in the CRT unit (address 10A). It is generally good practice to reselect Console Tape operations to the tape cassette drive upon completion of the batch processing operation. A typical batch job stream illustrating the use of LOAD command cards for batch processing is shown in Figures 14-4 and 14-5.

14.7 PRINTING BATCHED PROGRAM OUTPUT

In batch mode operation, it is generally good policy to maintain hardcopy listings of the programs loaded from cards, along with the printed output of each program. This is most conveniently done by selecting Console Output operations to a line printer or output writer. With Console Output continuously selected to a printer-type device, each card is automatically printed as it is read and processed, along with all other types of Console Output (including error codes). The hardcopy record thereby produced can be a useful debugging tool, since the type and location of all syntax errors are clearly shown. Console Output is selected to the line printer by including a card containing one of the following immediate mode statements at the beginning of the first program deck:

```
SELECT CO 215 (131) - 2221
SELECT CO 215 (80) - 2231
SELECT CO 211 - 2201
```

Note that the SELECT CO card should not have a program line number.

Console Output does not include the printed output of the PRINT and PRINTUSING statements. In order to produce hardcopy output from these statements, a SELECT PRINT 215 or 211 card should follow the SELECT CO card. Alternatively, a CLEAR card may be used following the SELECT CO card. CLEAR automatically resets the PRINT and LIST default addresses to the address currently selected for Console Output.

In order to cleanly separate individual program listings from one another, a PRINT HEX (OC) card should be used to eject the printer to the top of the next page for each new job if a line printer is used for output. If an output writer is used, a PRINT HEX (OAOAOA) card should be substituted. In this case, each 'OA' causes the typewriter to skip one line. An immediate mode PRINT HEX (OC) or PRINT HEX (OAOAOA...) card should be inserted at the beginning of each program deck to ensure that the program listing begins on a new page. Similarly, a programmed PRINT HEX (OC) or PRINT HEX (OAOAOA...) may be included in the program prior to each PRINT or PRINTUSING statement, to ensure that all printed program output begins on a new page.

14.8 READING DATA CARDS IN BATCH PROCESSING MODE

A program processed in batch mode may contain one or more INPUT, DATALOAD, or DATALOAD BT statements used to read in and process data cards from a subsequent data deck. The data deck accompanying each program should immediately follow the program (and the RUN command) in the input hopper (see Figure 14-4). When the program deck has been loaded into memory and the RUN card is read, control reverts from the card reader to the CPU, where program execution begins. If an INPUT, DATALOAD, or DATALOAD BT statement referencing an appropriate card reader device address is encountered in the course of program execution, the card reader is accessed and the data deck is read. When the specified number of data cards or an end-of-file card have been read, the data entry operation is terminated. Upon completion of program execution, control reverts once again to the currently selected Console Input device (the card reader), which proceeds to read the next card in the input hopper.

Batch Deck Protection

In programs which read data cards within a batched job stream, there is always the possibility that a logical error in the program, a miscount of the number of data cards in the deck, or a missing EOF card will cause the program to read beyond the data cards and into the control cards which follow. Such an oversight would destroy the first program (which bombs out with invalid data values) and the second program (which loses its initial control cards). To protect against such an error, a special RESET control card should be inserted as the last card in each data deck. The special RESET card is produced by marking the digits 1,2, and 3 in the first column of the STATEMENT NUMBER Zone (Zone 1). All other columns on the card are ignored by the reader.

When a special RESET card is read with INPUT or DATALOAD, it generates a System RESET condition (that is, it has the same effect as touching the RESET button on the keyboard). Thus a program which has erroneously attempted to read data beyond the last valid data card is automatically terminated when the special RESET card is read, and control reverts to Console Input, which proceeds to read in the next card. If the RESET control card is read following normal processing of the data deck, the RESET operation does not affect the batch processing operation. It is recommended that a RESET control card be inserted at the end of all data decks read with INPUT or DATALOAD in batch processing mode. (Note that the RESET card has no special meaning when read with DATALOAD BT, and should not be included in data decks read with that statement.)

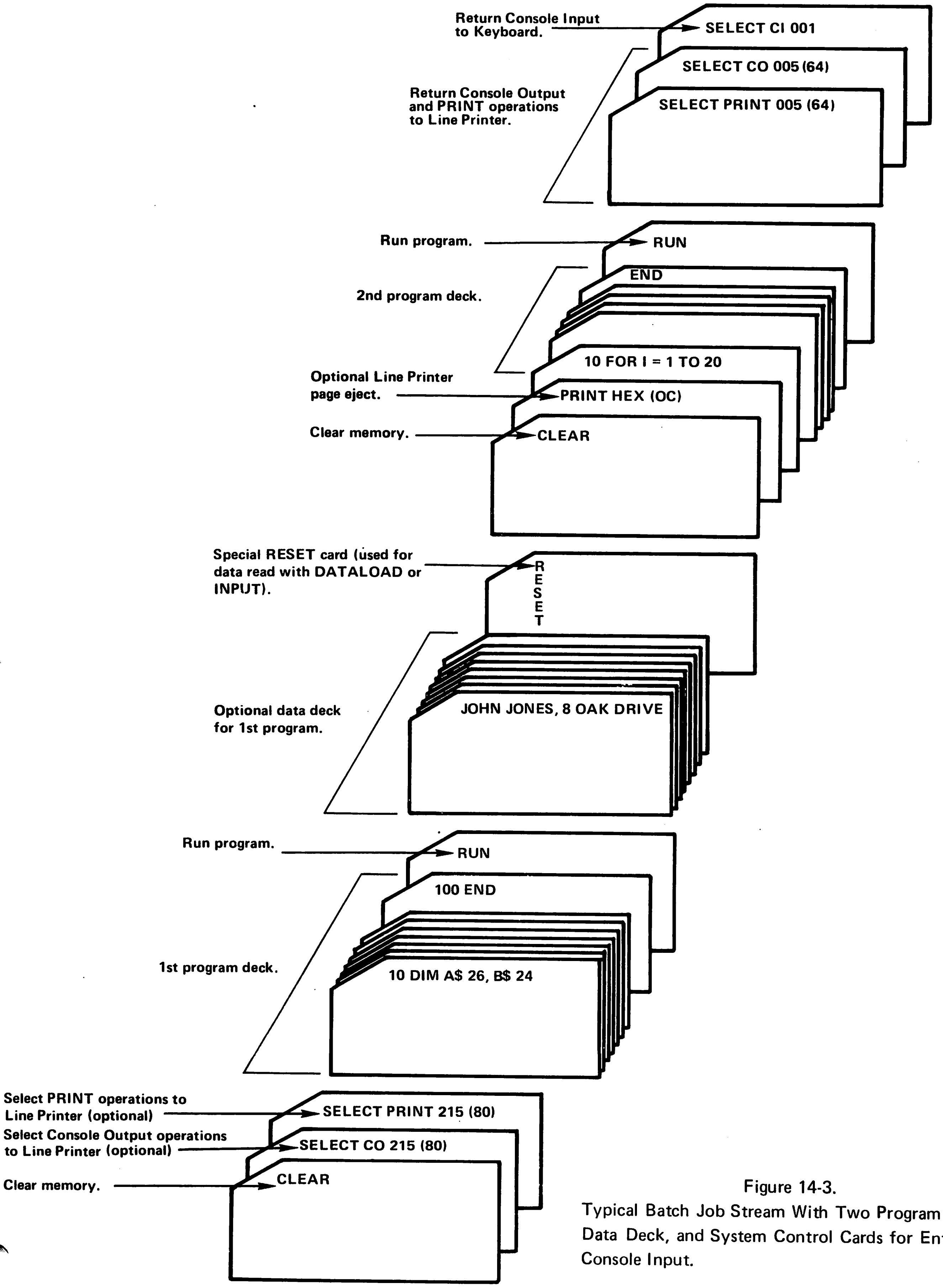

Return Console Input to Keyboard.
SELECT CI 001
SELECT CO 005 (64)
Return Console Output and PRINT operations to Line Printer.
SELECT PRINT 005 (64)
Run program.
RUN
END
2nd program deck.
10 FOR I = 1 TO 20
Optional Line Printer page eject.
PRINT HEX (OC)
Clear memory.
CLEAR
Special RESET card (used for data read with DATALOAD or INPUT).
RESET
Optional data deck for 1st program.
JOHN JONES, 8 OAK DRIVE
Run program.
RUN
100 END
1st program deck.
10 DIM A$ 26, B$ 24
Select PRINT operations to Line Printer (optional)
SELECT PRINT 215 (80)
Select Console Output operations to Line Printer (optional)
SELECT CO 215 (80)
Clear memory.
CLEAR

Figure 14-3.
Typical Batch Job Stream With Two Program Decks, Data Deck, and System Control Cards for Entry Via Console Input.

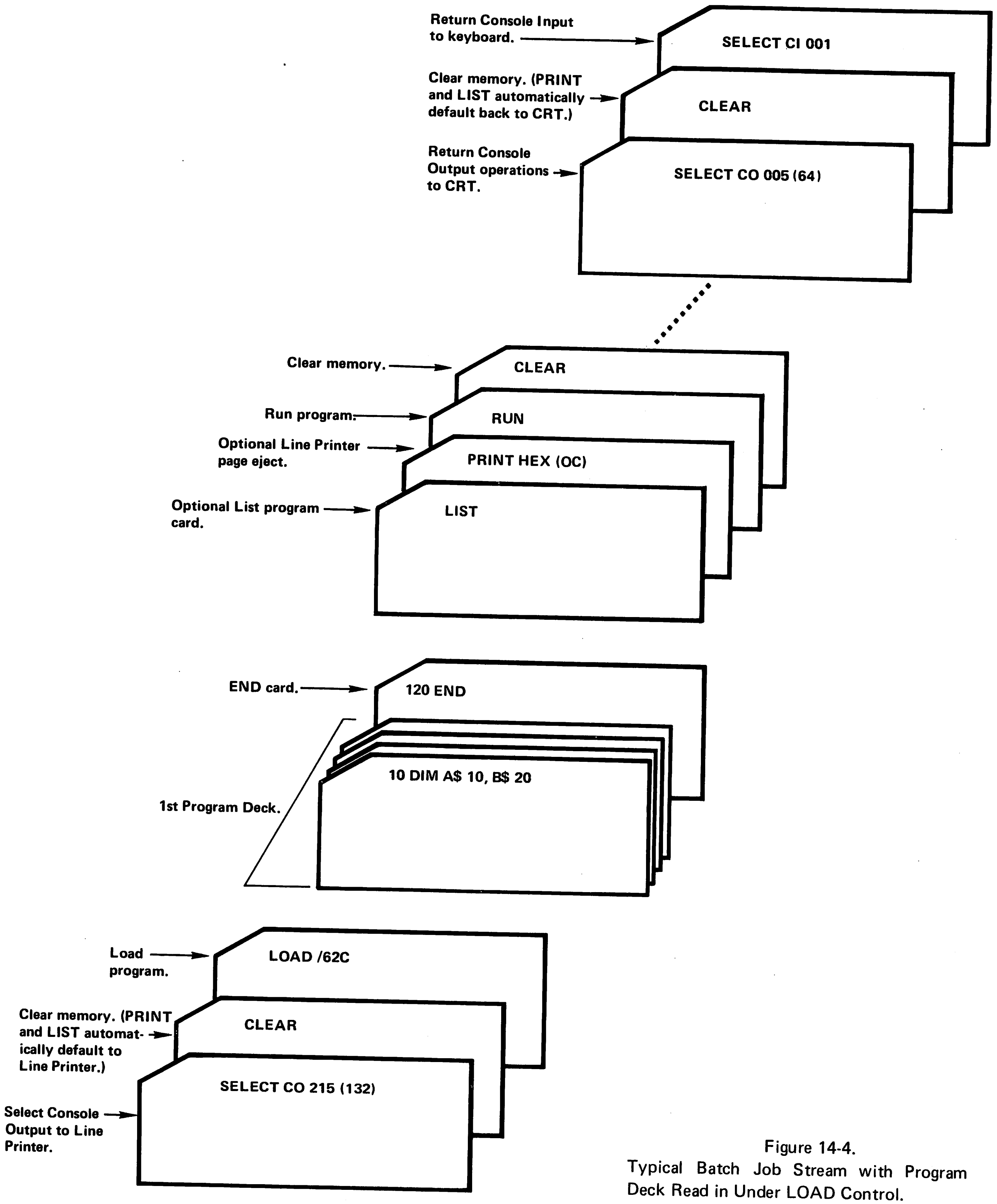

Return Console Input to keyboard.
SELECT CI 001
Clear memory. (PRINT and LIST automatically default back to CRT.)
CLEAR
Return Console Output operations to CRT.
SELECT CO 005 (64)
Clear memory.
CLEAR
Run program.
RUN
Optional Line Printer page eject.
PRINT HEX (OC)
Optional List program card.
LIST
END card.
120 END
1st Program Deck.
10 DIM A$ 10, B$ 20
Load program.
LOAD /62C
Clear memory. (PRINT and LIST automatically default to Line Printer.)
CLEAR
Select Console Output to Line Printer.
SELECT CO 215 (132)

Figure 14-4.
Typical Batch Job Stream with Program Deck Read in Under LOAD Control.

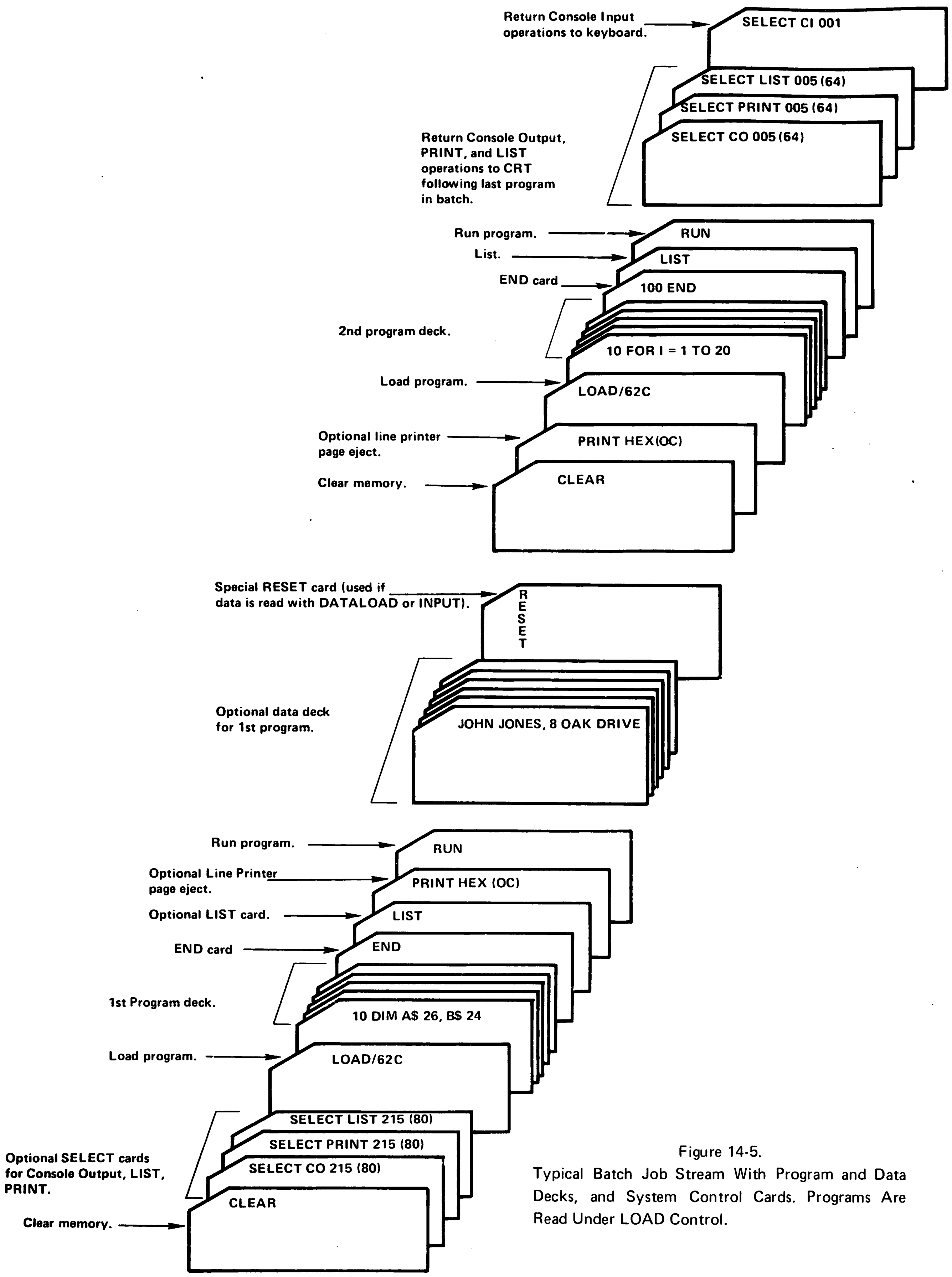


Figure 14-5.
Typical Batch Job Stream With Program and Data Decks, and System Control Cards. Programs Are Read Under LOAD Control.

PART IV
NON-STANDARD PUNCH AND MARK SENSE CARDS

CHAPTER 15
READING BINARY CARD IMAGES (DATALOAD BT, ADDRESS 62A)

15.1 INTRODUCTION

Data which is punched (or marked) in a code other than Hollerith can be read as straight binary data in Mode #4, and subsequently converted to a meaningful form in the system under program control. Binary data is read with a DATALOAD BT statement, and a specified or selected device address of 62A. With the Model 2234A, binary data can be read only from standard 80-column punched cards. With the Model 2244A, binary data can be read from punched or mark sense cards of 80 columns without timing marks, or of fewer than 80 columns with timing marks. Mode #4 reading operations are particularly valuable for reading custom-designed mark sense cards via the Model 2244A, since the special codes used on custom cards can be read as binary information, and converted to meaningful form with the system data manipulation features.

15.2 BINARY DATA FORMAT

In order to understand the format in which binary data is received from a card in Mode #4, it is helpful to consider the manner in which data is stored in memory. The smallest meaningful unit of storage in memory is a byte, which always consists of eight bits. The relative position of each bit is identified by the Hexadecimal value which the byte would have if only that bit were turned on (i.e., if only that bit equals "1", and all other bits equal "0"):

Bit Position	8	7	6	5	4	3	2	1
Hex Value	80	40	20	10	08	04	02	01
Binary Value	1	1	1	1	1	1	1	1

Figure 15-1.

Bit Positions and Hexadecimal Values

For example, if only the first (rightmost) bit is equal to "1", and the remaining seven bits equal "0" (00000001) the Hexadecimal value of the byte is HEX(01). If, on the other hand, only the fifth bit is turned on, and all others equal "0" (00010000), the byte's value is HEX(10), etc.

Because each card column consists of twelve rows, the card reader always reads twelve bits for each column. When the twelve bits are stored directly (without attempting to convert them into an equivalent 8-bit ASCII code), they must be divided up and stored in two separate bytes. The system therefore automatically divides the 12 bits received for each column into two groups of six bits, and stores each group of six bits into a separate byte. The two high-order bits of each byte (40 and 80) are set to zero. The format is illustrated in Figure 15-2:

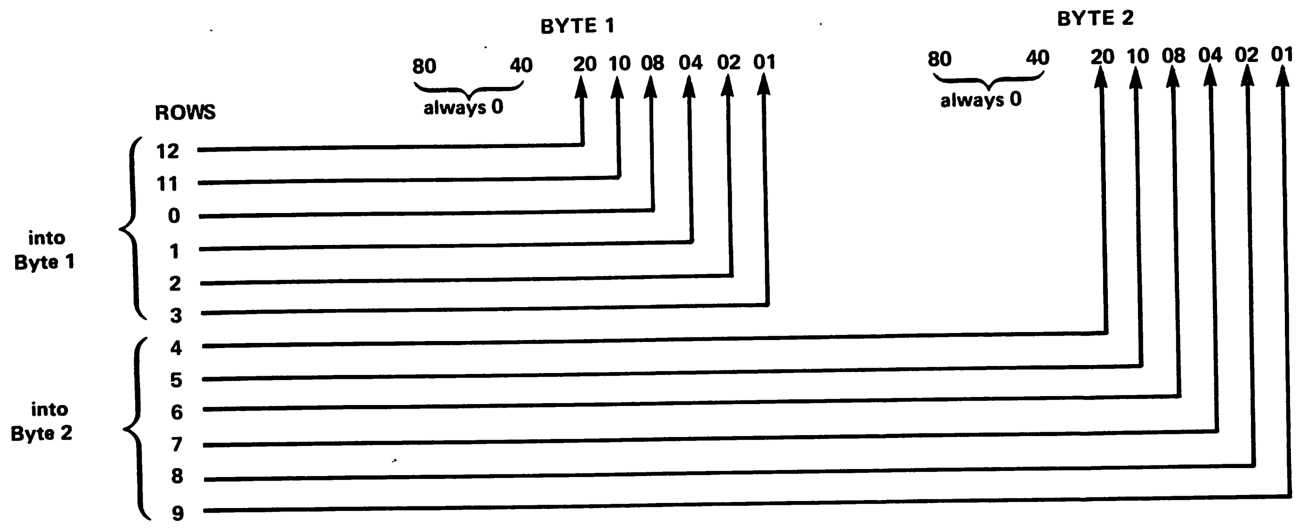


Figure 15-2.
Format of 12-Bit Binary Data Read from Cards in Mode No. 4. Bits from Row 12 Through Row 3 Are Read into Byte No. 1, Bits from Row 4 Through Row 9 Are Read into Byte No. 2.

Note that the two high-order bits (80 and 40) of Bytes #1 and #2 are set to zero. These bits function merely as "padding", to make up the needed eight bits for each byte. Consider, for example, the way in which the following card column is received in binary format:

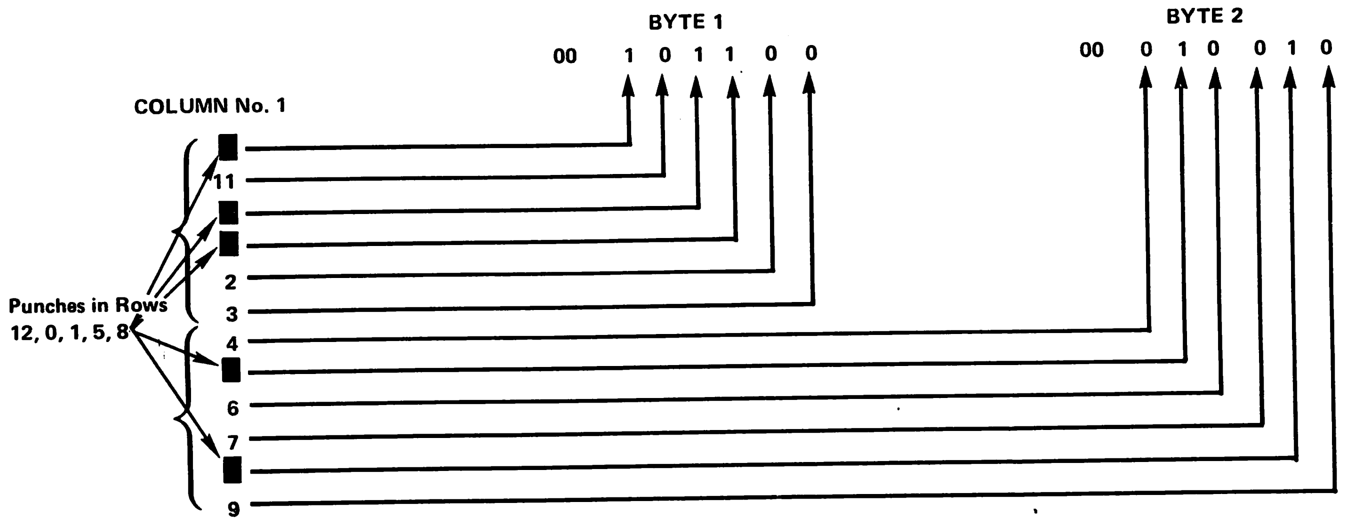


Figure 15-3.
Twelve Bits of Binary Data Read from a Single Card Column in Mode No. 4. The Top Six Rows Are Read into Byte No. 1, the Bottom Six Rows into Byte No. 2.

Since each column is converted into two binary bytes when a card is read in Mode #4, a total of 160 data characters are read for each card. In addition to the 160 data characters, the card reader itself automatically generates two special control characters, a LENGTH code character and an ERROR code character. The 161st character transmitted is the LENGTH code, which contains a binary count of the number of data characters read from the card. The 162nd character transmitted is the ERROR code, each bit of which signifies a unique card reader error condition. The LENGTH and ERROR codes should be checked following each read in Mode #4. If unexpected results are obtained, the program should alert the operator before reading another card.

15.3 READING BINARY CARD IMAGES

General Form: DATALOAD BT (N=162) $\begin{bmatrix}\text{/62A,}\\ \text{\#n,}\end{bmatrix}$ alpha array designator

Where:

(N=162) = The number of characters to be received for each card read. For binary card images, a total of 162 characters are received for each card (160 data characters for the card image (two per column) and two control characters). The number of characters to be read is always 162, irrespective of the number of actual data columns on the card (cards which contain index marks may have fewer than 80 columns). If fewer than 80 columns (i.e., fewer than 160 data characters) are read, the card reader automatically pads the remaining unread characters up to 160 with HEX(FF) characters.

/62A = The device address which designates the card reader as the device from which data is to be read, and also indicates the type of code conversion to be performed. Address 62A causes data to be read in binary format.

#n = A file number to which the device address 62A has been assigned in a SELECT statement ('n' is an integer from 1 to 6).

If neither a device address nor a file number is specified, the address of the Console Tape device is used. Ordinarily, this is the console tape cassette drive, address 10A. However, address 62A can be designated as the Console Tape device with a SELECT TAPE 62A statement. Following execution of this statement, a DATALOAD BT statement which specifies neither a device address nor a file number automatically uses address 62A.

alpha array designator = An alphanumeric array name followed by closed parentheses (e.g., A$(), F$()).

Purpose:

The DATALOAD BT statement with address 62A initiates the reading of an 80-column binary card image, and stores the data sequentially into the receiving alphanumeric array (the array is filled row by row). Each column is stored as a pair of binary bytes. A total of 162 bytes are received for each card. The first 160 bytes constitute the binary card image (two bytes per column); the 161st byte is the LENGTH code (a binary count of the number of data characters read), and the 162nd byte is the ERROR code. The receiving alphanumeric array must, therefore be dimensioned to hold at least 162 bytes. If a card contains fewer than 80 columns (i.e., if fewer than 160 data characters are read from the card), the card reader automatically pads the remaining unread characters with HEX(FF) characters, and the LENGTH code indicates the number of data characters actually read from the card. In cases where a card reader error condition prevents a card from being read, 160 HEX(FF) characters are transmitted, and the LENGTH code is binary zero (indicating no data characters read). An appropriate ERROR code is also indicated.

Example 15-1: Reading a Binary Card Image (DATALOAD BT, Address 62A)

```
50 DIM A$(5)40
.
.
.
.
500 DATALOAD BT (N=162)/62A, A$()
```

In this example, the receiving array A$() is dimensioned to consist of five elements, each 40 bytes in length. This arrangement is convenient because it effectively segregates the two control characters in the last element, A$(5), while the 160 characters of data are stored in the first four elements, A$(1) through A$(4). The first two characters of A$(5) are, therefore, the 161st and 162nd characters received; the remaining 38 bytes of A$(5) are unaffected by the DATALOAD BT operation. Note that DATALOAD BT continues reading only until the specified number of characters to be received (N=162) are read, regardless of whether the receiving array has been completely filled.

15.4 BINARY IMAGE CONVENTIONS

In summary, the following conventions apply to Mode #4 reading operations:

1. The '62A' address specifies that 80 columns be read for each card, and that each column be converted into two binary bytes.

2. For each card read with a '62A' address (Mode #4), 162 characters are transmitted to the system. Of these, 160 are data characters (two characters per column for each of the 80 columns), and the last two are control characters (the LENGTH and ERROR codes).

3. The receiving alphanumeric array in the DATALOAD BT statement should be dimensioned to contain at least 162 bytes.

4. The 161st byte received for each card is the LENGTH code, a binary count of the number of characters read (always twice the number of columns read).

5. With the Model 2234A, only standard 80-column punched cards are legal. A successful read, therefore, always results in 160 data characters being received (LENGTH code = HEX(A0); decimal equivalent, 160).

6. With the Model 2244A, cards having fewer than 80 columns may be read if the cards contain index marks. A valid read in this case may result in fewer than 160 data characters being read. The remaining unread characters up to 160 are received as HEX(FF) characters, and the LENGTH code indicates the number of characters (i.e., twice the number of columns) actually read. In addition, the ERROR code indicates that fewer than the expected number of columns were read.

7. The 162nd character received for each card is the ERROR code. Each bit of the error code represents a unique card reader error condition. (The ERROR code is discussed in Section 15.6, "Card Image ERROR Codes").

8. If, because of a reader error condition, the next card cannot be read, 160 HEX(FF) data characters are returned, along with a LENGTH code of HEX(00) and an appropriate ERROR code. For this reason, it is important that the card reader be in a RESET condition when the DATALOAD BT statement is executed. It is, however, possible to set up a read loop which regularly attempts to read a card, and checks for a "Machine Not Ready" error code following each attempt. If the error code is detected (indicating that the card was not read, and 160 HEX(FF) characters were received in lieu of valid data), the program can take appropriate action (such as alerting the operator). This is a typical example of the manner in which error conditions can be handled under program control in binary image reading mode. For a more extensive discussion of programming techniques associated with checking error codes under program control, refer to Chapter 17.

9. Blank (unpunched, unmarked) cards read in Mode #4 are not ignored, but are read in as 160 HEX(00) characters.

10. Unlike Modes #1 (loading Hollerith programs) and #2 (reading Hollerith data values), Mode #4 attaches no special significance to the 80th card column. Data may, therefore, be recorded in the entire 80 columns of cards read in Mode #4.

15.5 THE LENGTH CODE

The 161st character transmitted for each card is the LENGTH code, a binary count of the number of data characters read from the card. In Mode #4, the LENGTH code always equals twice the number of card columns read. Typically, 80-column cards are read; the LENGTH code in this case is HEX(A0), whose decimal equivalent is 160. In special cases, fewer than 80 columns may be read. The special cases include:

1. Fewer than 80 columns on the card. This situation cannot occur in normal reading on cards read by the Model 2234A, since only standard 80-column cards can be read by the 2234A. Cards read by the Model 2244A, may, however, have fewer than 80 columns if the cards also contain index marks. In such cases, the LENGTH code indicates the number of characters (twice the number of columns) actually read from the card.

2. Short card. The data card may be physically shorter than a standard 80-column card. The deck should be examined for mutilated or non-standard cards. Non-standard cards cannot be processed through the card reader.

3. Reader Failure. In rare cases, the reading sensors may experience a failure while reading a card, or a card may become jammed in the reading station as it is being read. Check the card track and reading station for jammed cards. If the problem appears to be a sensor failure, contact your Wang Service Representative.

NOTE:

In all cases, the number of characters to be received (specified by the 'N' parameter) must be 162 (N=162).

When cards are read in Mode #4, the LENGTH code should be checked after each card is read to ensure that all expected data was received. In all cases, if fewer than 160 data characters are read in Mode #4, the remaining unread characters are padded with HEX(FF) characters, and a bit in the ERROR code is set indicating that fewer than the expected number of data characters were received.

15.6 THE ERROR CODE

The 162nd character received for each card is the ERROR code. The ERROR code is an 8-bit code, each individual bit of which represents a unique card reader error condition. The table below lists the eight bit-positions and the error condition associated with each:

Table 15-1. Card Image Error Codes

BIT	VALUE	MEANING
80	1	Machine Not Ready
40	1	Hopper Empty
20	1	Stacker Full
10	1	Pick Check (A motion check: the picker failed to engage the card after six consecutive attempts.)
08	1	Read Alert (Generally, a card has a tear or marking on the leading edge, a photoelectric sensor has failed, or the read head is dirty.)
04	1	Less than expected data received (Short card, jam, read alert, or special card with fewer than 80 columns.)
02	1	Invalid ASCII conversion converted to ASCII exclamation ('!') character.
01	1	Invalid Look-Ahead operation (Attempted to perform Hollerith card read following Binary Look-Ahead, or Binary card read following Hollerith Look-Ahead.)

Note that the presence of a particular error bit may be anticipated under certain conditions. Custom-designed mark sense cards having fewer than 80 columns can be read in Mode #4 (Binary Card image), for example; in such cases, the 04-bit ('Less than Expected Data') is set in the ERROR code for each card. Because the card is known to contain fewer than 80 columns in this case, however, no corrective action is required. The LENGTH code should be checked to ensure that the expected number of columns actually were read.

In general, both the LENGTH and ERROR codes should be checked following every card read operation in Mode #4. To perform such checks most efficiently, and to ensure that the two control characters are never mistakenly interpreted as data, it is good practice to isolate the LENGTH and ERROR codes in the last element of the receiving array.

Example 15-2: Checking the LENGTH and ERROR Codes (Binary Card Image)

```
60 DIM A$(5)40, N$1
.
.
.
.
100 DATALOAD BT (N=162)/62A, A$()
110 IF STR(A$(5),1,2) <>  HEX (A000) THEN 350
.
.
. (Normal processing)
.
.
340 GOTO 100
350 N$ = STR(A$(5),2,1)
360 AND (N$,80):REM CHECK FOR 'MACHINE NOT READY'
370 IF N$ <> HEX(00) THEN 500
.
. (Test for other error conditions)
.
500 STOP "MACHINE NOT READY"
```

This example illustrates a typical processing routine which consists of two sections, a normal processing routine (lines 100-340) and an error processing routine (lines 350-500). At line 100, the 162-character image is read into array A$(). The 160 data characters are stored in the first four array elements, A$(1) - A$(4), while the two control characters are stored in the last element, A$(5). Line 110 checks both the LENGTH and ERROR codes, which are the first and second characters in A$(5). If the LENGTH code is A0(160) and the ERROR code is 00, all characters were read and no errors were detected, and the program proceeds to normal processing. If, however, the code is other than HEX(A000), then at least one error bit must be on, and the program branches down to the error routine beginning at line 350. At line 350, the ERROR code is stored into N$, a one-byte working variable. In this example, only one possible error condition, the 'Machine Not Ready' condition, is checked. The AND function is used to check for the presence of an 80 bit (indicating 'Machine Not Ready'). If the 80 bit is on (line 370), the program branches to line 400, stops, and prints the appropriate error message. If the 80 bit is off, the routine drops through to line 380, where it may test for other possible error conditions.

NOTE:

For a more extensive discussion of typical programming techniques utilized in testing the LENGTH and ERROR codes, see Chapter 17.

CHAPTER 16
BINARY LOOK-AHEAD MODE (DATASAVE BT, ADDRESS 42F)

16.1 INTRODUCTION

Binary Look-Ahead mode enables the card reader to feed a card through the reading station, convert the data to binary format, and store it in the card reader output buffer. During this entire procedure, the system's Central Processing Unit (CPU) may be occupied with other processing (perhaps operating on data read in from the previous card). By thus overlapping the card reading/code conversion operations with internal CPU processing operations, the total throughput time for processing binary data decks can be significantly reduced in many data processing applications.

16.2 OPERATION OF THE LOOK-AHEAD MODE

Mode #6, the Binary Look-Ahead mode, utilizes the BASIC statement DATASAVE BT with the special address '42F'. In Mode #6, the card reader reads a 160-character binary card image (two binary bytes for each card column) into the card reader output buffer, where the data is held awaiting transfer into memory. If fewer than 160 characters are read from a card, the output buffer is padded with HEX(FF) characters up to 160. Following the read, two special characters, a LENGTH code and an ERROR code, are generated by the card reader and stored as the 161st and 162nd characters in the buffer.

Note that the Look-Ahead mode is used only to read a card and temporarily hold the data in the card reader buffer. Look-Ahead mode cannot be used to transmit the data into memory for processing. Data is transmitted from the card reader output buffer into memory only in the legitimate binary card image reading mode, Mode #4 (DATALOAD BT with address 62A). Thus in every case the binary Look-Ahead operation must be followed by a binary read operation (DATALOAD BT, address 62A) which carries out the actual transfer of data from the card reader into the receiving array in memory.

16.3 CARD READING WITH LOOK-AHEAD

```
GENERAL FORM:  DATASAVE BT [/42F,] alpha variable
                           [#n,  ]
```

Where:

/42F =	The special card reader device address which specifies Binary Look-Ahead operations. Address '42F' causes the card reader to feed in the next card from the input hopper, convert the data into the 6/6 binary format, and hold the converted data in the card reader output buffer awaiting transmission to the system.
#n =	A file number to which address 42F has been assigned in a SELECT statement ('n' must be an integer from 1 to 6).
	If neither a device address nor a file number is specified, the address of the device currently designated as Console Tape device is used. Normally, the Console Tape device is the console tape cassette drive in the CRT (address 10A).
alpha variable =	A "dummy" alphanumeric variable included to satisfy general format requirements for the DATASAVE BT statement, but not used in the Look-Ahead operation. For optimum efficiency, this variable should be dimensioned to the minimum length, one byte.

Purpose:

The DATASAVE BT statement with a device address of 42F initiates the reading and conversion of an 80-column binary card image. Each column is converted into a pair of binary bytes, and the 160-character card image is stored in the card reader buffer, along with LENGTH and ERROR code characters. Subsequently, the binary card image can be read from the card reader buffer into a receiving alphnaumeric array in memory with a DATALOAD BT staement and a device address of 62A (mode #4, binary image reading). The dummy alphanumeric variable has no functional purpose, serving only to satisfy the format requirements of the DATASAVE BT statement. For maximum efficiency of the Look-Ahead operation, this variable should be dimensioned to the minimum possible length of one byte.

Example 16-1: Processing Binary Card Images With Look-Ahead
(DATASAVE BT, Address 42F)

```
70 DIM B$(5)40, D1$1                     (Dimension receiving array
.                                        for DATALOAD BT and dummy
.                                        variable for DATASAVE BT)
.
190 SELECT #2 62A, #3 42F                (Assign read address, 62A, to
                                         #2, and binary Look-Ahead
                                         address, 42F, to #3)
200 DATALOAD BT (N=162) #2, B$()         (Read a card)
210 IF STR(B$(1), 1, 4) = "ZZZZ" THEN 710 (Check for last card)
220 DATASAVE BT #3, D1$                  (Feed in the next card)
.
.
.   (Process data read at line 200; at the same time, the next card is
.   being read and converted to binary format by the card reader.)
.
.
700 GOTO 200
710 STOP
```

In this example, the Binary Look-Ahead mode (address '42F') is used to speed up the processing of binary card images. At line 190, the binary card reading address, 62A, is assigned to file number #2, and binary Look-Ahead address 42F is assigned to #3. A binary card image (160 data characters plus two control characters) is read at line 200. At line 210, the card just read is tested for End-of-File. If the dummy value "ZZZZ" is found in the first field, the end-of-file card has been read, and the program halts. Otherwise, the program drops through to line 220, where the card reader is instructed to feed in the next card, convert each column to binary format (two binary bytes), and hold the data in the card reader buffer. While this procedure takes place, the system continues normal processing of the data read at line 200. The card reading and conversion is thus carried out simultaneously with CPU internal processing. When processing is complete, the program loops back to line 200 to read in the next 162 characters, which are awaiting transmission in the card reader output buffer. Note that D1$ in line 210 is a dummy variable which has no functional use in the Look-Ahead operation.

CHAPTER 17
SUPPLEMENTAL PROGRAMMING TECHNIQUES FOR PROCESSING DATA CARD IMAGES

17.1 INTRODUCTION

Most Wang systems provide the programmer with a high degree of control over the validation of data read from cards, and the response to error conditions which arise in the course of reading data cards. Many types of errors which may arise in the reading process are detected by the card readers themselves, and signalled to the controlling program in the ERROR code (if the cards are read in one of the two image modes). Such situations as jammed or damaged cards, illegal ASCII conversions, or an empty input hopper generate unique ERROR codes which can be examined and acted upon under program control. Additionally, it is possible in most systems to check the validity of data values read from each card, and to take appropriate action in the event one or more invalid values are detected. For this purpose, the General I/O statements (especially $UNPACK) provide an added degree of efficiency.

Given the format restrictions discussed in previous chapters, it is possible to read data from cards with any one of the statements INPUT, DATALOAD, or DATALOAD BT. The first two statements require that data cards adhere to predetermined format conditions. Further, these statements provide minimal control over error conditions. Indeed, they provide no control over card reader-generated error conditions such as card jams, bad reads, etc. If a card jam occurs during a DATALOAD loop, for example, the card reader stops and the program simply hangs up. In image mode (DATALOAD BT), on the other hand, the programmer obtains complete program control over card reader error conditions and data validity. The DATALOAD BT statement provides a built-in means of storing the error indicators sent by the card reader with a special ERROR code character. And, because DATALOAD BT does not demand a predefined card format, the programmer is able to design his own format and verify each value read under program control. Its capability to provide extensive control over data validation and error detection makes DATALOAD BT the statement of choice for serious data processing applications.

This chapter discusses a number of program techniques for the validation and conversion of numeric data from Hollerith card images, and for the analysis and response to error conditions signalled by the card reader in image mode.

Section 17.2 deals with the detection of card reader errors, and presents several techniques for responding to error conditions; Section 17.3 contains a brief discussion of the effectiveness of Look-Ahead in speeding up card reader throughput; and Sections 17.4 through 17.8 cover a variety of methods of validating and converting numeric data values from Hollerith card images.

17.2 DETECTION OF CARD READER ERRORS

Each time a card is read in image mode with DATALOAD BT, two special control characters are transmitted to the CPU following the data. The first character is the LENGTH code, and the second is the ERROR code.

The LENGTH code is a byte of control information generated by the card reader itself following each card read in image mode. The LENGTH code is a binary count of the number of bytes of data actually read from a card. In the event of a reading error, the LENGTH code can be helpful in determining the source of the error by indicating the number of columns read before the error occurred. In a Hollerith card image, the LENGTH code always is the 81st byte transmitted by the reader; in a binary card image, the LENGTH code is the 161st byte transmitted.

In addition to a LENGTH code, the card reader automatically generates an ERROR code for each card image. The ERROR code is a byte of control information in which each of the eight bits represents a unique card reader error condition. In a Hollerith card image, the ERROR code is always the 82nd byte transmitted; in a binary card image, it is always the 162nd byte. The error conditions designated by each of the eight bits in the ERROR code are listed below:

TABLE 17-1
Card Reader ERROR Conditions

BIT	VALUE	MEANING
80	1	Machine Not Ready.
40	1	Hopper Empty.
20	1	Stacker Full.
10	1	Pick Check. (The picker has failed to engage the card after six consecutive tries.)
05	1	Read Alert. (Generally, a card has a tear or mark on the leading edge, a photoelectric sensor has failed, or the read head is dirty.)
04	1	Less Than Expected Data Received. (Short card, jam, read alert, or special card with fewer than 80 columns.)
02	1	Invalid ASCII Conversion (Illegal Hollerith code converted to ASCII exclamation ('!') character.)
01	1	Invalid Look-Ahead Operation. (Attempted to perform Hollerith card read following binary Look-Ahead, or binary card read following Hollerith Look-Ahead.)

NOTE:

On the Model 2244A, a LENGTH code of HEX(00) and an ERROR code of 04 may indicate that the card has been read upside down.

Three of the ERROR codes do not in themselves shut down the card reader. They are:

01	Invalid Look-Ahead Operation
02	Invalid ASCII Conversion
04	Less Than Expected Data

The remaining five codes always are accompanied by automatic shutdown of the card reader:

08	Read Alert
10	Pick Check
20	Stacker Full
40	Hopper Empty
80	Machine Not Ready

Following the occurrence of any one of these codes, the card reader must be manually restarted. It is not uncommon for more than one error condition to occur at the same time. For example, each of the error conditions "Read Alert", "Pick Check", "Stacker Full", and "Hopper Empty" is always accompanied by a "Machine Not Ready" code. A "Hopper Empty" condition is thus indicated by the presence of both an 80-bit and a 40-bit:

$$\begin{array}{cccccccc} 80 & 40 & 20 & 10 & 08 & 04 & 02 & 01 \end{array}$$
$$\underbrace{1 \quad 1 \quad 0 \quad 0}_{\text{HEX C}} \quad \underbrace{0 \quad 0 \quad 0 \quad 0}_{\text{HEX 0}}$$

Figure 17-1.
ERROR Code for "Hopper Empty", showing Both 80 and 40 Bits On.

As Figure 17-1 illustrates, the expected ERROR code for "HOPPER EMPTY" is HEX (C0) rather than HEX (40).

It is generally good practice to examine both the LENGTH code and the ERROR code after each card image is read. The LENGTH code always indicates the number of data characters read from a card. For an 80-column Hollerith card image, a normal LENGTH code is HEX(50). For an 80-column binary image, a normal LENGTH code is twice this value, or HEX (A0). Under certain conditions, the card reader will read fewer than 80 columns on a card. These conditions include a short card (physically shorter than a normal 80-column card), a card jam, or a special card containing index marks with fewer than 80 columns.

Even in cases where specially designed cards having fewer than 80 columns are to be read, the DATALOAD BT statement should always specify N=82 for Hollerith card images, and N=162 for binary card images. If a card contains fewer than 80 or 160 data characters, the reader automatically pads the remaining unread characters up to 80 or 160 with HEX(FF) characters. The LENGTH code then indicates the actual number of data characters read; the "dummy" HEX (FF) characters are not included in the character count. For a 40-column card, e.g., the LENGTH code equals HEX (28). Thus, the programmer can check the LENGTH code following each read to ensure that the expected number of characters have been read.

A normal ERROR code is HEX (00). In general, both the LENGTH and ERROR codes should be checked following each card image read. If either code has an unexpected value, the program can be directed to take appropriate action. There are several options open to the programmer following discovery of an error. The program can be directed to examine the LENGTH and ERROR Codes, print out a description of the error conditions that have been set, and halt program execution. Alternatively, it may be desirable to print a description of the error conditions and continue reading cards (if the reader has not already been halted). If an examination of the error conditions reveals that the card reader is not ready, an appropriate message to the operator can be displayed, and the audio signal on the printer or CRT can be sounded to indicate a condition that requires operator intervention. The program can then enter a loop in which it continually attempts to read a card, checking for a ready condition each time. When the operator has corrected the situation and reset the card reader, program execution continues as usual.

The examples below illustrate some techniques for reading data card images and analyzing the LENGTH and ERROR Codes. Several possible responses to the detection of error conditions under program control also are presented.

For Hollerith card images, the specified device address in a DATALOAD BT statement is 629. For binary card images, the device address is 62A.

Example 17-1: Checking the LENGTH and ERROR Codes in a Hollerith Card Image

```
10  DIM A$(3)40
20  DATALOAD BT (N=82) /629,A$()
30  IF STR(A$(3),1,2) = HEX(5000) THEN 40: GOSUB'30
```

In this example, the LENGTH and ERROR codes are checked for their normal values; the normal value of the LENGTH code in this case is HEX(50), and the normal value of the ERROR code is HEX (00). Thus, if the 81st and 82nd bits received equal HEX (5000), both codes are normal, and processing resumes at line 40. Otherwise, something is amiss, and the program branches to an error recovery routine (GOSUB '30 in this example).

Example 17-2: Checking the LENGTH and ERROR Codes in a Binary Card Image

```
10  DIM A$(5) 40
20  DATALOAD BT (N=162) /62A, A$()
30  IF STR(A$(5),1,2) = HEX (A000) THEN 40: GOSUB '30
.
.
.
```

In this example, the LENGTH and ERROR codes are checked following a binary image read. The normal value of the LENGTH code in this case is 160 (HEX (A0)). At line 30, the 161st and 162nd characters are checked. If their values equal HEX (A000), all is well, and processing resumes at line 40. Otherwise, the program branches to an error routine (GOSUB '30).

Since it is not uncommon for more than one error bit to be set in the ERROR code, it may be desirable to examine each of the meaningful bits, and list the errors before taking further action. This can be done by using a series of AND functions to mask out all but the one bit to be examined in each case.

Example 17-3: Use of the AND Function to Test for Individual Error Bits

```
100 DIM A$(3)40, T$1
110 DATALOAD BT (N=82) /629, A$()
120 IF STR(A$(3),1,2) = HEX(5000) THEN 40 : GOTO 500
.
.
.
500 T$ = STR(A$(3),2,1) : AND(T$,80)
510 IF T$ = HEX(00) THEN 530
520 PRINT "CARD READER NOT READY"
530 T$ = STR(A$(3),2,1) : AND(T$,40)
540 IF T$ = HEX(00) THEN 560
550 PRINT "HOPPER EMPTY"
```

In the input routine starting at line 110, a Hollerith card image is read in and, at line 120, the LENGTH and ERROR codes are tested for their normal values. If an unexpected value is found in either code, the program branches to line 500, where it begins an examination of individual bits in the ERROR code. At line 500, the ERROR code is stored in T$, and all bits except the 80 bit are masked out to zero. At line 510, T$ is tested for the presence of an 80 bit. If the 80 bit is on, the card reader is not ready to read a card, and the program prints an appropriate error message. If not, the program skips to line 530, where the 40 bit is checked. The process continues until all error bits have been checked, and the appropriate error message(s) printed out.

The fact that certain error codes imply automatic card reader shutdown, while others do not, may influence the program's response in each case. For the three errors which do not halt card reader operation (04, Less Than Expected Data; 02, Invalid ASCII Conversion; and 01, Invalid Look-Ahead Operation), it may be preferable simply to print out the erroneous card image(s) along with an error message, and continue regular processing.

Example 17-4: Response to Error Conditions Without Operator Intervention

```
500 T$ = STR(A$(3),2,1) : AND(T$, 04)
510 IF T$ = HEX (00) THEN 540
520 PRINT A$(1); A$(2) GOTO 110
530 PRINT "LESS THAN EXPECTED DATA ON ABOVE CARD"
540 T$ = STR(A$(3),2,1):AND(T$,02)
.
.
.
```

In this example, the AND function is used to test for Code 04 (Less than Expected Data). If the 04 bit is on, the card which was the source of the error is printed out, along with an appropriate error message, and the program proceeds to line 540 to check the next error bit.

Errors which shut down the card reader, and which therefore require operator intervention, cannot of course be treated in this way. In such cases, the program may be directed to signal the operator (perhaps by printing a HEX(07) to the printer or CRT if an audio signal option is available), after an appropriate error message has been printed or displayed. Note that the 'Machine Not Ready' bit (80) always accompanies a reader shutdown. It is only necessary, therefore, to test for an 80-bit in order to determine if operator intervention is required.

Example 17-5: Response to Error Conditions With Operator Intervention

```
500 T$ = STR(A$(3),2,1) : AND(T$,80)
510 IF T$ = HEX(00) THEN 700
520 PRINT "CARD READER NOT READY"; HEX(07)
530 T$ = STR(A$(3),2,1) : AND (T$,40)
540 IF T$ = HEX(00) THEN 560
550 PRINT "HOPPER EMPTY"; HEX(07)
.
.
.
```

Lines 500 and 510 check for the presence of an 80 bit in the ERROR code. If the 80 bit is found (indicating an error condition which has shut down the card reader), an error message is printed, the audio signal on the CRT or printer is activated, and the program proceeds to check for a specific error condition (40, 20, 10, or 08) beginning at line 530. If the 80 bit is not on, the error condition is not terminal, and the program branches to line 700 to check for codes 01, 02, and 04.

To avoid the necessity of having the operator rerun the program following a card reader shutdown, the program may enter a loop in which it continuously attempts to read a card, checking the ERROR code after each attempt. If the card reader is not in a ready condition, the system receives 80 or 160 HEX(FF) characters following each read, along with a LENGTH code of HEX(00) and an ERROR code in which the 80 bit is set. (Other error bits may, of course, also be set.) By checking for the 80 bit after each attempted read, the program can determine whether the operator has corrected the problem and restarted the card reader. Once the problem is corrected, normal program execution resumes automatically.

Example 17-6: Testing for Card Reader 'Ready' Condition

```
300 DATALOAD BT (N=82) /629, A$()
310 T$ = STR(A$(3),2,1) : AND(T$, 80)
320 IF T$ = HEX(00) THEN 340
330 PRINT HEX(07) : GOTO 300
340 T$ = STR(A$(3),2,1) : AND (T$,04)
.
.
.
```

In this example, the program loops continuously, attempting to read a card at line 300, and checking for an 80 bit at lines 310 and 320. If the 80 bit is on, the program drops through to line 330, where the audio signal is sounded to indicate that the problem remains uncorrected, and the system loops back to line 300 to attempt another read. If no 80 bit is found at line 320, the program branches to line 340 to check for error bits 04, 02, and 01 before processing the card image.

A more elaborate and systematic routine for checking error bits in the ERROR code is illustrated in Example 17-7. This example performs the following operations:

1. Reads an 80 column Hollerith card image into an alphanumeric array, A$() (line 70).

2. Checks the LENGTH and ERROR Codes. If these codes are normal, the program processes the card before returning to read another card (line 80). Since the example's only concern is to show a method of dealing with card reader errors, actual processing of data on the cards is ignored. If the LENGTH and ERROR codes have unexpected values, the example proceeds to anlayze the problem and take appropriate action (line 90).

3a. The first bit checked in the ERROR code is the 80 bit (Machine Not Ready) (lines 3000 - 3015). If this bit is ON, the routine proceeds to check only bits 40, 20, 10 and 08 since these are the only conditions that can generate a 'Not Ready' (lines 3020 - 3170).

3b. Once the reason for the 'Not Ready' condition is identified, an appropriate error message is displayed for the operator, and the audio signal is sounded.

3c. The routine then returns to read another card, and check the 'Not Ready' bit. As long as the reader remains not ready, the program remains in this loop, continuously sounding the audio signal.

3d. Once the reader becomes ready, and a good read is made, normal processing continues (line 100).

4. If the 80 bit is OFF when the ERROR code is first checked at line 3010, the only other bits checked are the 04 bit (Less than Expected Data), the 02 bit (Invalid ASCII Conversion), and the 01 bit (Invalid Lookahead Operation). In these cases, an image of the erroneous card is printed out along with an appropriate error message. Program execution then continues with the next card (lines 3180 - 3260).

Example 17-7: Testing for Card Reader Error Conditions

```
70  DATALOAD BT (N=82)/629, A$()
80  IF STR (A$(30, 1, 2) = HEX(5000) THEN 100
90 GOSUB 3000 : GOTO 70
.
.
.
500 GOTO 70
.
.
.
    .
    .
3000 T$ = STR(A$(3),2,1) : AND(T$, 80)
3010 IF T$  <> HEX(80) THEN 3180
3015 PRINT HEX(03) " CARD READER NOT READY "
3020 T$ = STR(A$(3),2,1) : AND(T$, 40)
3030 IF T$  <> HEX(40) THEN 3060
3040 PRINT HEX(0707); "     HOPPER EMPTY     "
3050 GOTO 3270
3060 T$ = STR(A$(3),2,1) : AND(T$,20)
3070 IF T$ <>  HEX(20) THEN 3100
3080 PRINT HEX(0707); "   STACKER FULL   "
3090 GOTO 3270
3100 T$ = STR (A$(3),2,1) : AND(T$, 10)
3110 IF T$ <>  HEX(10) THEN 3140
3120 PRINT HEX(0707); "  PICK CHECK, CAN NOT READ CARD  "
3130 GOTO 3270
3140 T$ = STR (A$(3),2,1) : AND(T$, 08)
3150 IF T$ <>  HEX(08) THEN 3180
3160 PRINT HEX(0707); "     READ ALERT-- DAMAGED CARD     "
3170 GOTO 3270
3180 T$ = STR(A$(3),2,1) : AND(T$, 04)
3190 IF T$ = HEX(04) THEN 3200 : GOTO 3225
3200 SELECT PRINT 215 (131)
3210 PRINT "LESS THAN EXPECTED DATA ON THE FOLLOWING CARD..."
3220 GOTO 3250
3225 T$ = STR(A$(3),2,1) : AND(T$, 02)
3226 IF T$ = HEX(02) THEN 3230 : GOTO 3246
3230 SELECT PRINT 215(131)
3240 PRINT "INVALID ASCII CONVERSION ON THE FOLLOWING CARD..."
3245 GOTO 3250
3246 SELECT PRINT 215 (131)
3247 PRINT "INVALID LOOKAHEAD ON THE FOLLOWING CARD..."
3250 PRINT A$(1); A$(2) : PRINT
3260 SELECT PRINT 005(64)
3270 RETURN
```

17.3 EFFICIENT USE OF THE LOOK-AHEAD MODE

The binary and Hollerith Look-Ahead modes can be used to reduce the total throughput time for processing data cards. Approximately 200 milliseconds are required to read a card, translate the codes to ASCII or binary format, and store the data in a receiving array in memory. In the normal case, no other processing can be carried out while this procedure takes place. With the Look-Ahead mode, however, it is possible to execute the card reading and code conversion operations concurrently with other processing in the CPU.

A Look-Ahead operation is indicated with the DATASAVE BT statement and a device address of 42E (Hollerith) or 42F (binary). When a DATASAVE BT statement with a Look-Ahead device address is executed, the card reader is signalled to read in the next card from the input hopper. At this point, the card reader is, in effect, "cut loose" from the system, and proceeds to read the card, convert the data to ASCII or binary format, and store the converted data in a buffer, independently of the CPU. Immediately after the Look-Ahead operation is initiated, program execution continues at the next line in the program; the system does not wait for the card reader to complete its read operation. Program execution therefore proceeds while the card is read and stored in the card reader buffer. Typically, the program processes data read from the previous card at this point. When the processing is complete, the program executes a DATALOAD BT statement to store the preprocessed data from the card reader buffer into a receiving array in memory.

Example 17-8: Processing Hollerith Card Images with Look-Ahead

```
10  DIM A$(3)40, F$1
.
.
.
100 DATALOAD BT (N=82)/629, A$()
110 DATASAVE BT/42E, F$
120 IF STR (A$(3), 1, 2) = HEX(5000) THEN 140
130 GOSUB 2500: GOTO 100
140 IF STR (A$(1), 1, 3) = "ZZZ" THEN 510
.
.
.
510 STOP
```

In this example, the DATALOAD BT statement initially causes a card to be read and the data stored into receiving array A$(). Immediately afterwards, the Look-Ahead command at line 110 signals the card reader to read the next card into its buffer, and convert each character to ASCII. While the card is being read, processing continues on the data already stored in A$() at line 120. When processing is complete, the program returns to line 100 to read in data from the next card. In most cases, this data is already waiting in the card reader buffer, and can be transferred very quickly into array A$().

Note that after the last data card is read in Example 17-8, the Look-Ahead operation attempts to read a card from an empty hopper. This attempt causes the buffer to be filled with 80 HEX(FF) characters, a LENGTH code of HEX(00) to be generated, and the 40 bit ("Hopper Empty") to be set in the ERROR code. None of this is meaningful to the program, which will never attempt to transfer this data from the buffer into memory, since it exits from the input loop after detecting the the dummy end-of-file value "ZZZ" at line 140. The data therefore remains in the card reader buffer, and is cleared only by Master Initializing the system or keying RESET from the keyboard.

In general, the amount of processing required for each card image is sufficient to produce an overall gain in throughput time with Look-Ahead. Only in cases where little or no processing takes place between reads is it likely that Look-Ahead will fail to speed up throughput time.

17.4 DATA VALIDATION AND CONVERSION PROCEDURES

Once a Hollerith card image has been read into a receiving alphanumeric array with DATALOAD BT, the first step in processing the data should normally be to check the LENGTH and ERROR codes. Techniques for checking these codes are described in preceding sections. When it has been established that no card reader errors have occurred, the program must proceed to isolate and extract individual data values from the card image, since typically several values are stored on each card. In the process of converting and processing data (particularly numeric data) from a card image, a variety of system errors may arise. Normally, such errors cause an error code to be displayed, and halt program execution. The programming techniques described in the following sections offer some suggestions on how to convert numeric values from a card image, and respond to any errors which may arise in this process under program control, without terminating program execution.

In order to extract an individual value from a card image, the programmer must be able to locate the value within the continuous image. Since DATALOAD BT, unlike INPUT and DATALOAD, does not recognize any special character as a data separator (but regards the entire card instead as one continuous 80-character value), the programmer is free to define individual data fields in any convenient way. Probably the simplest and most common technique for defining data fields is merely to alot a specific number of columns for each field. Because each column transfers as a single character when a Hollerith card image is read, it is an easy task to extract individual values from the receiving array with a sequence of STR functions.

On the card illustrated in Figure 17-2 below, for example, the first field consists of columns 1-24. When this card is read into an array, the first value is easily obtained by using the STR function to extract the first 24 characters from the first array element. For example, from a receiving array A$() the name "John Jones" can be extracted with a statement of the form:

```
B$ = STR (A$(1), 1, 24)
```

Following this operation, the first 24 characters of A$(1) (i.e., the first 24 card columns) are stored in B$.

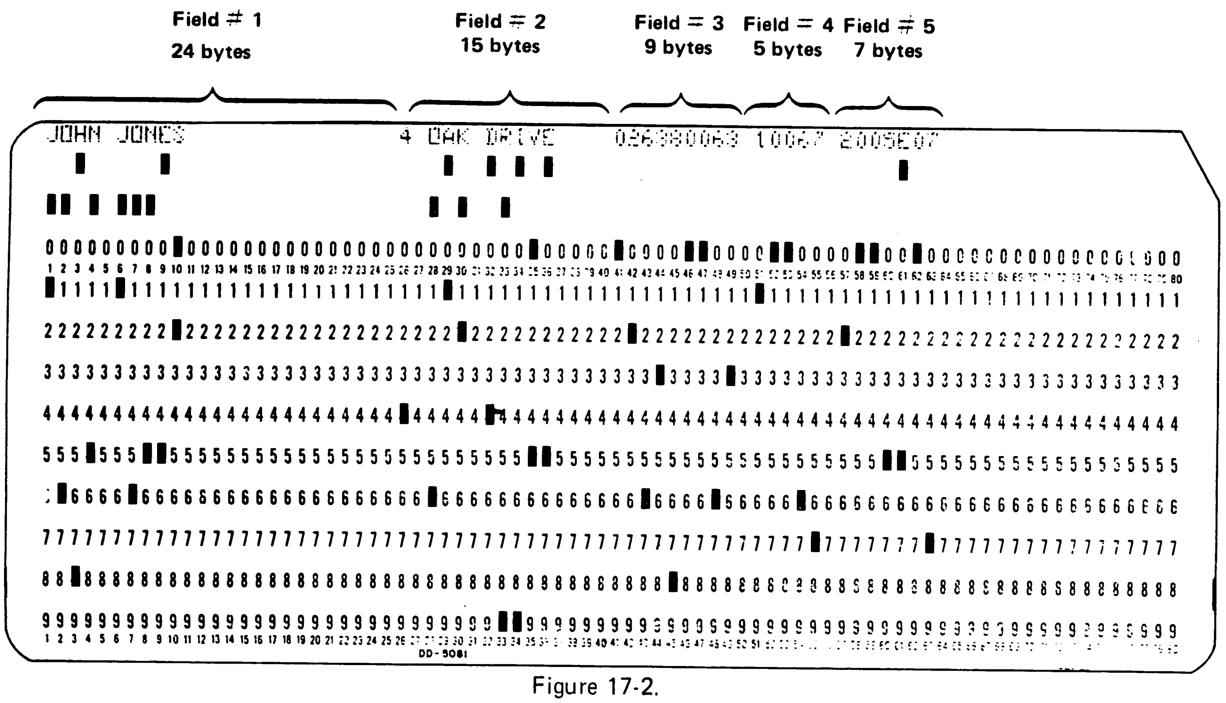


Figure 17-2.
Hollerith Card with Four Data Fields.

So long as alphanumeric fields only are treated, this procedure remains quite straightforward and essentially unproblematic. With the introduction of numeric values, however, the process assumes an added degree of complexity. Alphanumeric values are regarded by the system merely as strings of characters; the system assumes no predefined relationship between separate characters, and therefore demands no special format. The only requirement is that every character be a legal ASCII character; since the card reader automatically converts all illegal characters into ASCII exclamation ('!') characters, this requirement is always met. With numeric values, however, the situation is vastly different. When a numeric value is read from a card into an alphanumeric variable or array, it is regarded as an alphanumeric value - that is, as nothing more than a string of characters having no special significance from the system's point of view. When an attempt is made to transfer a numeric value into a numeric variable, however, the system must convert the value into the Wang internal numeric format. In this case, the value must conform to the system's format requirements for legal BASIC numbers. The presence of an illegal numeric character or an improper format generates an error message, and program execution is automatically halted.

Two major problems present themselves, then, in dealing with numeric data. First, numeric values must be converted from the alphanumeric array into which they are read to numeric format for storage in numeric variables. Second, adequate safeguards must be established to prevent program execution from halting with an error if an illegal numeric value is encountered. Section 17.5 discusses the use of the CONVERT statement to convert numeric data from a card image into internal numeric format. Section 17.6 covers the use of the NUM function in validating numeric data prior to converting it, and Section 17.7 explains how the ON ERROR GOTO statement can be used to recover from errors which may be generated by a CONVERT operation. Section 17.8, finally, discusses the General I/O statement $UNPACK in connection with data conversion and validation.

17.5 NUMERIC DATA CONVERSION WITH THE 'CONVERT' STATEMENT

Numeric values can be converted from the alphanumeric array argument of the DATALOAD BT statement to internal numeric format for storage in a numeric variable with the CONVERT statement. The CONVERT statement converts a number stored in a specified alphanumeric variable or array to numeric format, and stores it in a specified numeric variable. Consider, for example, the following pair of statements:

```
10 A$ = "1234"
20 CONVERT A$ TO N
```

The number "1234" is stored in A$ as a string of ASCII characters. CONVERT translates the character string into internal numeric format, and stores the converted numeric value in numeric variable N. Following execution of the CONVERT statement, therefore, the variable N contains the number 1234.

CONVERT also can be used in conjunction with the STR function to convert only a specified number of characters from an alpha variable or array element. For example, the statement

```
CONVERT STR (A$, 1, 3) TO N
```

converts only the first three characters of A$ ("123") to numeric format. In this case, therefore, N is set equal to 123.

Numeric data can be converted from the card image in the receiving alphanumeric array to numeric variables if the starting position and field length of each numeric field is known. For example, the three numeric fields on the card illustrated in Figure 17-2 could be processed with the following routine:

Example 17-9: Processing Numeric Data from a Card Image

```
10  DIM A$(3) 40
20  DATALOAD BT (N=82) /629, A$()
.
.
.
150 CONVERT STR (A$(2), 1, 9) TO N
160 CONVERT STR (A$(2), 11, 5) TO O
170 CONVERT STR (A$(2), 18, 5) TO P
```

A card image from the card illustrated in Figure 17-2 is read into array A$() at line 20. Since A$() is dimensioned to consist of three 40-byte elements, the card columns 41-80 are read into the second array element, A$(2). Thus column 41 (which contains the first character of the first numeric field) is read into byte #1 of A$(2). At line 150, a CONVERT statement utilizes the STR function to convert bytes 1-9 of A$(2) into numeric format, and store the resultant number in numeric variable N. A blank column separates the first and second numeric fields on the card, and this space character occupies byte #10 in A$(2). The second CONVERT statement therefore begins operating on the 11th byte of A$(2), and converts the five bytes 11-16 into numeric format for storage in numeric variable O. This process is repeated for the third and last numeric field at line 170.

Note that the character string operated upon by CONVERT must be a representation of a legal BASIC number. If the character string contains illegal numeric characters, or contains numeric characters in an illegal format, the CONVERT statement produces an error message, and program execution is terminated.

17.6 VALIDATING NUMERIC DATA WITH THE 'NUM' FUNCTION PRIOR TO CONVERSION

One way to ensure that a CONVERT statement will not encounter an illegal numeric and halt program execution is to validate each numeric value with the NUM function prior to converting it into numeric format. (If your system supports the ON ERROR GOTO statement, that statement may provide a more efficient technique for handling erroneous numeric data; see Section 17.7.)

The NUM function examines a specified string of ASCII characters in an alphanumeric variable or array element, and computes the number of sequential characters in the string which are legal numeric characters. Legal numeric characters include:

. Digits 0 - 9
. Plus (+) and Minus (-) signs
. Decimal Point (.)
. Letter 'E' (designating exponent)
. Space

Any other character is illegal in a BASIC number, and will produce an error if encountered by CONVERT. NUM counts sequential numeric characters starting with the first character of the specified variable or STR function. The character count is ended either by the detection of a non-numeric character, or when the sequence of numeric characters fails to conform to standard BASIC number format. Leading and trailing spaces are included in the count. Thus, NUM can be used to verify that an alphanumeric character string is a legal representation of a BASIC number, or to determine the length of a numeric portion of an alphanumeric value.

If a numeric field in a Hollerith card image is in correct format, the character count produced by NUM will equal the field length. Consider, for example, the following short routine:

```
10 DIM A$ 11
20 A$ = "+12.3E6"
30 N = NUM (A$)
```

In this case, A$ contains a seven-character numeric value, " + 12.3E6." Since A$ is dimensioned to 11 bytes, the remaining four bytes are filled with space characters, which also count as numerics. Thus, the value of N at line 30 is 11.

If, however, a non-numeric character appears in the field, the NUM character count will be less than the field length:

```
10  DIM A$ 11
20  A$ = "12.6P7"
30  N = NUM(A$)
```

In this case, N = 4, since the fifth character of A$ is the letter 'P', a non-legal numeric character.

The numeric fields read from the card in Figure 17-2 could be tested for validity prior to conversion by modifying the conversion routine in Example 17-9 in the following manner:

Example 17-10: Validating Numeric Data With NUM Prior to Conversion

```
10  DIM A$(3)40
20  DATALOAD BT (N=82) /629, A$()
.
.
.
150 X = NUM(STR(A$(2), 1, 9)): IF X = 9 THEN 160: GOTO 1000

160 Y = NUM (STR(A$(2), 11, 5)): IF Y = 5 THEN 170 : GOTO 1000
170 Z = NUM(STR(A$(2), 18, 5)) : IF Z = 5 THEN 180 : GOTO 1000
180 CONVERT STR (A$(2), 1, 9) TO N
190 CONVERT STR (A$(2), 11, 5) to O
200 CONVERT STR (A$(2), 18, 5) TO P
.
.
.
.
1000 PRINT "ERRONEOUS VALUE ON THIS CARD"
1010 PRINT A$(1); A$(2)
1020 GOTO 20
```

Line 150 utilizes the NUM function to compute the number of sequential numeric characters in the first numeric field. This field occupies bytes 1-9 in A$(2); thus, the character count stored in X should be 9. If X = 9, the program drops through to validate the second numeric field at line 160. If X does not equal 9, the field contains at least one illegal character, and a branch is made to the error routine at line 1000, where the card image is printed out along with an error message. The program then loops back to read the next card. If all three fields prove to contain valid data, the program proceeds to convert each field to numeric format at lines 180, 190, and 200.

17.7 ERROR RECOVERY WITH THE 'ON ERROR GOTO' STATEMENT

In systems which support it, the ON ERROR GOTO statement offers a most efficient means of handling error conditions under program control. If an error occurs during program execution, ON ERROR GOTO automatically directs execution to a specified line number, where the error can be analyzed and an appropriate response initiated under program control. Program execution is not automatically terminated by discovery of the error, as would be the case without ON ERROR GOTO.

Note that certain types of errors are not processed by ON ERROR GOTO. Errors which occur during program resolution (the phase immediately after a program is run, during which the system scans the program, checking the validity of line number references and allocating space for variables prior to beginning actual program execution) are not handled by ON ERROR GOTO. Such errors cause a normal error message to be displayed, and halt further program execution. All errors occurring in the course of program execution can, however, be processed by ON ERROR GOTO without terminating program execution or causing an error code to be displayed. Specifically, ON ERROR GOTO is useful for responding to errors which arise from the detection of an invalid number by CONVERT.

The ON ERROR GOTO statement must include two alphanumeric variables and a program line number. Upon detection of an error during program execution, the appropriate error code is stored in the first specified variable, while the line number of the line at which the error occurred is stored in the second variable. Program execution then branches to the specified line number. Consider, for example, the following statements:

```
10  DIM F$2, G$4
20  ON ERROR F$, G$ GOTO 1000
```

Upon detection of an error condition, variable F$ is set equal to the appropriate error code (two bytes), while G$ is set equal to the line number at which the error appeared (four bytes). Immediately upon detection of the error, program execution branches to line 1000, where an error analysis routine might be initiated.

It is mandatory that only one ON ERROR GOTO statement be included in a program. The statement should be placed near the beginning of the program.

In general, the error routine to which the program is directed by ON ERROR GOTO should analyze the error code to determine the cause of the error. Further action would be determined by the particular error discovered. Note that since only one ON ERROR GOTO can appear in a program, the error routine should cover all error possibilities in the entire program, and not just the numeric field verification discussed in this section.

Example 17-11: Recovering from Numeric Format Errors with ON ERROR GOTO

```
10  DIM F$2, G$4
20  ON ERROR F$, G$ GOTO 1000
.
.
.
90  DIM A$(3)40
100 DATALOAD BT (N=82)/629, A$()
.
.   (Check LENGTH and ERROR Codes)
.
150 CONVERT STR(A$(2),1,9) TO N
.
.
.
1000 IF E$ = "20" THEN 1010: GOTO 1040
1010 PRINT "ILLEGAL NUMBER FORMAT ON FOLLOWING CARD
     AT LINE #" ; G$
1020 PRINT A$(1); A$(2)
1030 GOTO 100
1040 IF E$ = "47" THEN 1050: GOTO 1080
.
.
.
etc.
```

In the above example, a Hollerith card image (perhaps the one in Figure 17-2) is read at line 100. At line 150, the program begins converting numeric fields to internal numeric format with CONVERT. If an illegal number is encountered, the system generates a CODE 20 (Illegal Number Format). In this event, the ON ERROR GOTO statement sets F$ equal to the error code (20), and G$ equal to the line number of the line at which the error occurred (line #150). Program execution is then directed to line 1000, where the error analysis routine begins checking the error code in F$. If error code 20 is indeed found in F$, the program displays an appropriate error message at line 1010, along with the line number from G$ and the contents of the erroneous card. In this way, identification and correction of the card are simplified. The program then loops back to read the next card. If code 20 is not found in F$, the routine continues to test for other possible error codes.

In most cases, ON ERROR GOTO simplifies the task of error detection and data verification, since it enables the programmer to test for a wide range of error conditions with a single statement. Without it, the programmer must adopt special procedures for identifying different types of errors and validating data values before attempting to convert the card image to a more useful form. (The NUM function, discussed in the preceding section, is one example of a special technique used to validate numeric data.)

17.8 DATA CONVERSION WITH '$UNPACK' (GENERAL I/O STATEMENT)

The $UNPACK statement is available as part of the General I/O instruction set. General I/O is optional on some Wang systems and standard on others; refer to the literature accompanying your system for further details.

For serious data processing applications, in which data is read in image mode from Hollerith cards, and subsequently distributed to individual variables and/or arrays, $UNPACK is an enormously useful tool. With a single $UNPACK statement, it is possible to "unpack" an entire card image (that is, to separate all data values from the image, and transfer each value to a specified receiving variable or array). $UNPACK automatically converts numeric values into internal numeric format prior to storing them in numeric variables. Thus, $UNPACK can perform in a single statement the work of many string functions and CONVERT operations.

$UNPACK offers the programmer two forms, a field form and a delimiter form. The particular form chosen is dictated by the data format of the cards to be read. In the field form of $UNPACK, the lengths and types of all fields in the card image can be specified. In the delimiter form, a special delimiter character separating data values in the image is specified. Whatever the card format, therefore, $UNPACK is, where available, the recommended method of converting data from a Hollerith card image.

```
General Form:

     $UNPACK  [F = alpha variable]  alpha array designator TO argument
              [D = alpha variable]                            list

where:

          F  =  field format parameter
          D  =  delimiter format parameter

alpha array  =  name of array containing card image, followed by
designator      closed parentheses (e.g., A$(), F$()).  This is
                the "buffer" from which the data is unpacked.

                numeric variable
                numeric array designator
argument        alphanumeric variable
list         =  alphanumeric array designator
                STR function
```

NOTE:

In the interest of simplicity, the general form of $UNPACK above includes only those parameters with a specific application for unpacking a data card image. Of course, the $UNPACK statement has a broad range of applications beyond those under consideration in this section. A comprehensive general form for $UNPACK, and a more general discussion of its capabilities, can be found in the General I/O Instruction Set Reference Manual.

Purpose:

$UNPACK causes values to be taken sequentially from the specified buffer (in this case, the buffer is the receiving alphanumeric array from the DATALOAD BT statement which contains the card image) and stored sequentially in the variables and arrays in the argument list following the word "TO". Arrays in the argument list are filled element by element, row by row; each array element receives a single data value.

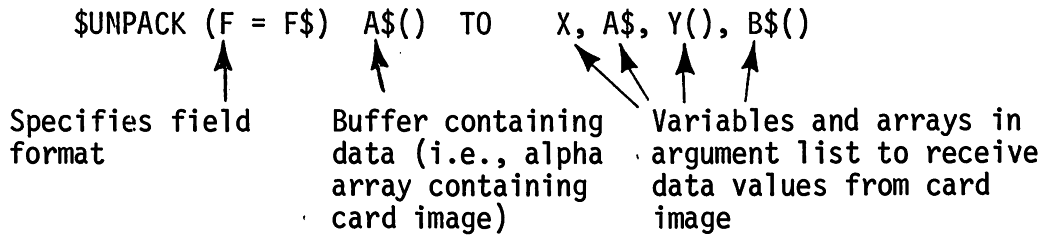

$UNPACK (F = F$) A$() TO X, A$, Y(), B$()
Specifies field format
Buffer containing data (i.e., alpha array containing card image)
Variables and arrays in argument list to receive data values from card image

Figure 17-3.
Typical $UNPACK Statement in Field Format Showing Typical Components

Note that $UNPACK has two formats, field format and delimiter format, specified by the letters "F" and "D" respectively. In general, the field format is used in cases where the programmer knows the length (number of bytes) of each data field. The delimiter parameter is used in cases where a predefined delimiter character (a space, comma, slash, etc.) is used to separate data fields on the card.

FIELD FORM OF $UNPACK

A "field" is a definite number of columns on a card (and, correspondingly, of bytes in the receiving alphanumeric array) occupied by a single value. It was pointed out in previous sections that a common method of defining values on a card is to assign each value a specific number of card columns (i.e., a specific field length). If this technique is employed, the field format for $UNPACK can be used to "Unpack" values from the fields in a card image.

Field format is specified with the letter 'F'. The alpha variable following 'F =' is called the "field specification variable", and contains the field specification codes for data fields in the buffer.

For example:

```
$UNPACK (F = F$)    B$()  TO  X , Y , Z

   Specifies   Field specification
   field       Variable
   format
```

Figure 17-4.
Field Specification Variable in a $UNPACK Statement

The field specification variable (F$ in Figure 17-4 above) contains one or more field specification codes. Field specification codes are two-byte hexadecimal codes, the first byte of which specifies the field type (alpha or numeric), and the second byte of which specifies the field length (the number of consecutive bytes in the field).

```
             F$   =   HEX (A008)

   Field specification     Field specification
   variable                code (two-byte code
                           specified in hexadecimal
                           notation.)
```

Figure 17-5
Field Specification Code Stored in the Field Specification Variable

$UNPACK sequentially transfers each field from the specified buffer (i.e., the alphanumeric array containing the 82-byte card image) to the corresponding variables and arrays in the $UNPACK argument list. Numeric fields (so identified by an appropriate field specification code) are automatically converted to numeric format prior to being transferred to the specified numeric variables or arrays. For this reason, numeric values can be stored only in numeric variables, and alphanumeric values can be stored only in alpha variables; otherwise, an error is signalled.

Each byte of the field specification code is expressed as a pair of hexadecimal characters. The first byte of the code (i.e., the first pair of hexadecimal digits) indicates the field type (alpha or numeric). For an alphanumeric field, the first byte must be HEX(A0). For a standard numeric field in free-format, the first byte of the field specification code must be HEX(10).

In addition to the standard alphanumeric and numeric codes, two special codes are provided for skipping a field and indicating a special numeric format. A specified field can be skipped (i.e., not transferred from the buffer to a variable or array in the argument list) by specifying HEX(00) as the first byte of the corresponding field specification code.

For numeric values which are punched in integer format with an implied decimal point, a second special code is used. Although the implied decimal point format is less versatile than standard free-format, it is used in some applications. In this case, the number is punched on a card without a decimal point or exponent. When it is unpacked, the position of the implied decimal point must be specified so that the decimal point can be inserted at the proper location. For this purpose, the two hexadecimal digits which comprise the first byte of the field specification code have discrete meanings. The first hex digit must be '2', indicating a numeric field in implied decimal format. The second hex digit specifies the position of the implied decimal point, measured from the right end of the field. In different terms, the second hex digit specifies the number of decimal digits (i.e., digits to the right of the decimal point) in the numeric field. For example, if the first byte of the field specification code is HEX(22), the system assumes that the corresponding data field is a numeric value in implied decimal format, and that the position of the implied decimal point is two digits from the right side of the field. If the number in this field is +12345, therefore, it is converted to +123.45.

Table 17-2
Valid Field Specification Codes (in Hexadecimal Notation)

00 XX	Skip this field
10 XX	Numeric field in free-format
2d XX	Numeric field in integer format
A0 XX	Alphanumeric field

where: XX = second byte of field specification code, indicating field length in binary

d = implied position of decimal point for integer format numbers

The second byte of the field specification code defines the field length (i.e., the number of sequential bytes in the field).

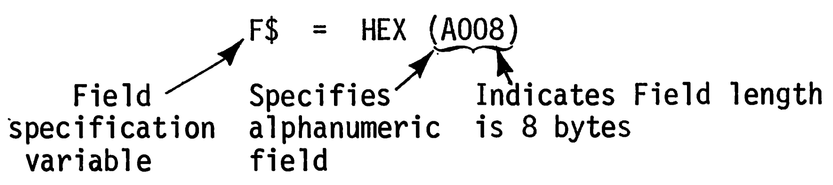

F$ = HEX (A008)
Field specification variable
Specifies alphanumeric field
Indicates Field length is 8 bytes

Figure 17-6.
Field Specification Code for Eight-Byte Alphanumeric Field

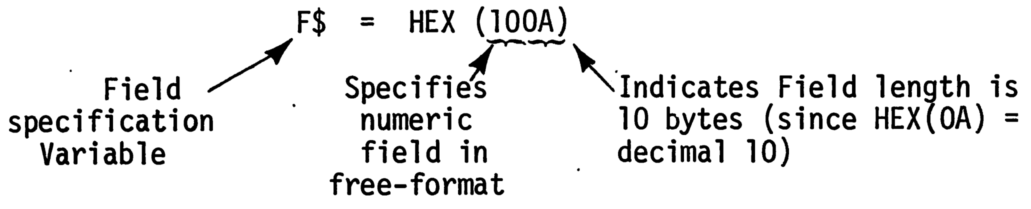

F$ = HEX (100A)
Field specification Variable
Specifies numeric field in free-format
Indicates Field length is 10 bytes (since HEX(0A) = decimal 10)

Figure 17-7.
Field Specification Code for 10-Byte Numeric Field in BASIC Free-Format

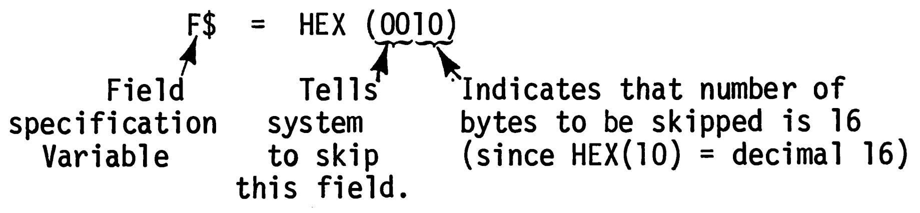

F$ = HEX (0010)
Field specification Variable
Tells system to skip this field.
Indicates that number of bytes to be skipped is 16 (since HEX(10) = decimal 16)

Figure 17-8.
Field Specification Code for Skipping a 16-Byte Field

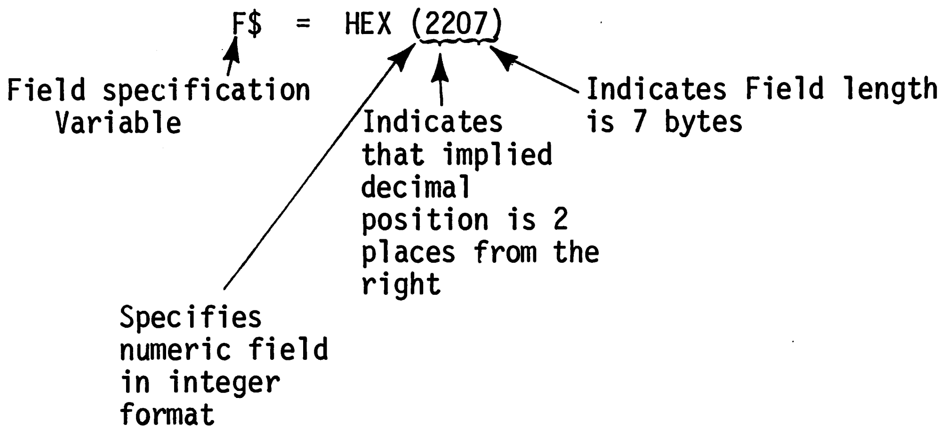

F$ = HEX (2207)
Field specification Variable
Indicates that implied decimal position is 2 places from the right
Indicates Field length is 7 bytes
Specifies numeric field in integer format

Figure 17-9.
Field Specification Code for Seven-Byte Numeric Field in Integer Format (Implied Decimal Position is Two Places to Right of First Digit)

A separate field specification code must be supplied for every variable or array in the $UNPACK argument list. For example, the following statement:

```
$UNPACK (F = F$)  B$()  TO  A$, B(), C$
```

requires three field specification codes. If

$$\text{F\$ = HEX (}\underbrace{\text{A008}}_{\text{A\$}}\underbrace{\text{1006}}_{\text{B()}}\underbrace{\text{A010}}_{\text{C\$}}\text{)}$$

then

- HEX(A008) is the field specification code for the field to be transferred to A\$. An alphanumeric field of eight bytes is specified.
- HEX(1006) is the field specification code for the fields to be transferred to B(). Assuming B() is dimensioned to 10 consecutive numeric fields in free format, each field is six bytes in length. Each element of B() receives the value from a single field. B() is filled row by row.
- HEX(A010) is the field specification code for C\$. An alphanumeric field of 16 bytes (since HEX(10) = decimal 16) is specified.

Note that all fields which are to be stored in an array are identified with a single field specification code. This is possible because the type and length of each field stored in an array must be the same as the type and length of all other fields stored in the same array. The system continues to unpack fields of the specified type and length from the buffer into the array until one of the following conditions is met: (a) the receiving array is full; or (b) an invalid numeric value is encountered; or (c) the data in the buffer is exhausted.

Suppose, for example, that a Hollerith card image is read into array A\$(). The card has the following format:

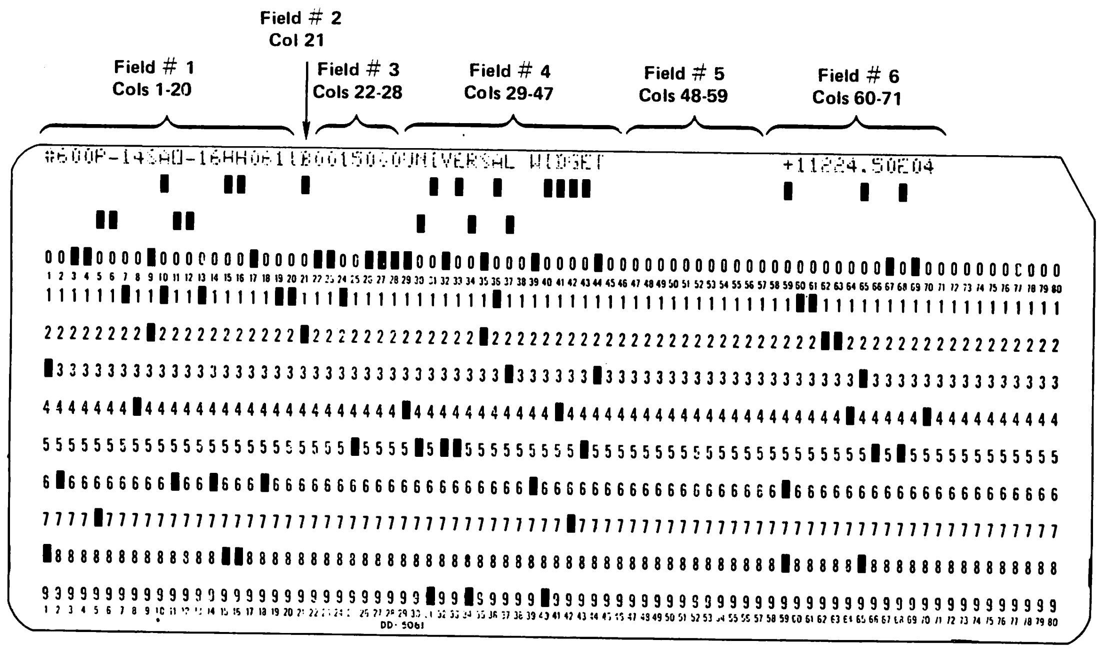


Figure 17-10.
Hollerith Card with Six Data Fields.

Columns 1 - 20 - an alphanumeric value
Column 21 - an alphanumeric value
Columns 22-28 - a numeric value in free-format
Columns 29-48 - an alphanumeric value
Columns 60-71 - a numeric value in free-format

Example 17-12 below might be used to read this card, separate the data fields, and store them in variables B$, C$, X, D$, and Y.

Example 17-12: Using Field Form of $UNPACK to Separate Data Values in a Hollerith Card Image

```
50  DIM A$(3)40, B$20, C$1, D$19
60  DATALOAD BT (N=82) /629, A$()
.
.
.
100 F$ = HEX(A014A0011007A013000C100C)
110 $UNPACK (F=F$) A$() TO B$, C$, X, D$, Y
.
.
.
```

In this example, the card from Figure 17-7 is read at line 60. At line 100, the field specification variable is assigned field specification codes for each variable in the argument list, according to the following scheme:

Field	Specification Code	Receiving Variable	Field Type & Length
1	A014	B$	Alpha field of 20 bytes
2	A001	C$	Alpha field of one byte
3	1007	X	Numeric field of seven bytes (free format)
4	A013	D$	Alpha field of 19 bytes
5	000C	-	Skip this field (12 bytes)
6	100C	Y	Numeric field of 12 bytes (free-format)

Notice that because the last alpha field ends at column 48, and the last numeric field begins at column 60, there is a range of 12 unused columns between the fields which are read as a blank field. In order to skip this field when transferring data from the buffer, the code 000C is inserted, instructing the system to skip the 12-byte field between the fourth and sixth fields. Following execution of line 110, the data values are separated and stored in the specified variables. Numeric values are converted automatically to internal numeric format.

A situation may arise in which the numeric fields on a card do not correspond to the numeric (or integer) field specifications in the $UNPACK statement. This might occur, for example, if an incorrect field length is specified in one of the field specification codes. Similarly, a numeric field may be in improper format or contain an illegal character. Illegal numeric characters might be produced, for example, if a card were punched on a keypunch other than the standard IBM 029 keypunch unit. (On keypunches which do not use the expanded version of Hollerith, such as the IBM 026 keypunch, Hollerith codes for certain numeric characters may decode into non-numeric ASCII characters.) In each of these cases, the system would normally signal an error and terminate program execution when the $UNPACK statement attempts to convert the illegal number into internal numeric format.

Errors such as these can be handled under program control with the ON ERROR GOTO statement discussed in Section 17.7. Alternatively, the occurrence of such errors can be prevented simply by unpacking all numeric fields into alphanumeric variables, and decoding the invalid numeric codes to valid ASCII equivalents prior to converting the values to numeric format.

DELIMITER FORM OF $UNPACK

In cases where a varying number of values of varying lengths are to be placed on cards, it may be inconvenient to assign each value a specified field length. A more convenient and efficient means of defining individual values in this case is to separate consecutive values with a specified delimiter character. The delimiter form of $UNPACK, specified with the 'D' parameter, is used to unpack data which are separated by a specified delimiter character. The data values are transferred from the buffer sequentially into the receiving variables and/or arrays in the argument list following the word 'TO' in the $UNPACK statement. A delimiter-form unpacking operation terminates when either the buffer is emptied or the argument list is filled .

The variable following "D=" in a $UNPACK statement is known as the delimiter specification variable. The first two bytes of this variable constitute the delimiter specification code, a two-byte code in hexadecimal notation which controls the unpacking operation. (The remaining bytes of the delimiter specification variable are ignored.)

```
$ UNPACK  (D = D$) B$()  TO  X, Y, Z
         ↗     ↑
Specifies   Delimiter specification
delimiter   variable
format
```

Figure 17-11.
Delimiter Specification Variable in a $UNPACK Statement.

The first byte of the delimiter specification code defines the particular unpacking procedure to be used, while the second byte is the ASCII code for the specified delimiter character.

Table 17-3 shows that the first byte of the delimiter specification code may assume one of four possible values: HEX(00),HEX(01), HEX(02), or HEX(03). These values dictate specific system responses to two special situations:

a) Insufficient data in the buffer to satisfy all variables in the argument list; and

b) More than one delimiter character between successive data values.

With regard to (a) above, if $UNPACK exhausts the data in the buffer before it has filled all variables in the argument list, the delimiter specification code offers two possible responses:

1. An error code (ERROR 97) can be generated and program execution terminated; or

2. The remaining unfilled variables in the argument list can be ignored (that is, left with their current values), and program execution allowed to continue with the next statement.

If the first response is selected, the error condition can be handled under program control with an ON ERROR GOTO statement. Note that the response selected will be determined by the format of the cards to be read. If all cards are expected to contain the same number of values, an insufficiency of data indicates an incorrect card or a bad read, and should signal an error. If, on the other hand, some cards contain fewer values than others, an insufficiency of data in some cases may be anticipated as normal procedure, and should not affect normal processing.

With respect to the presence of successive delimiter characters between data values, the delimiter specification code also offers two possible responses:

1. The successive delimiters can simply be ignored; or

2. Each successive delimiter character can be interpreted as an instruction to skip one argument (one variable or array) in the $UNPACK argument list. Skipped arguments retain their current values.

The capability to skip arguments in the argument list can be important if the card formats are not uniform. Assume, for example, that a deck of inventory cards contains some cards with a part number, model number, and assembly code, while other cards contain only the part number and assembly code. It is always assumed that the part number is stored in A$, the model number in B$, and the assembly code in C$. For cards which lack a model number, it is necessary to skip variable B$ when unpacking the card image, so that the assembly code can be stored in C$. In this case, the use of an additional delimiter character between the model number and assembly code can be used to cause the system to skip the intervening variable.

Table 17-3
Valid Delimiter Specification Codes in Hexadecimal Notation

Specification Code	Meaning
00 XX	1. Signal error if data is exhausted before all arguments filled; 2. Skip one argument in argument list for each successive delimiter between values.
01 XX	1. Ignore remaining arguments when data is exhausted before all arguments filled (no error); 2. Skip one argument in argument list for each successive delimiter between values.
02 XX	1. Signal error if data is exhausted before all arguments are filled; 2. Ignore successive delimiters between values.
03 XX	1. Ignore remaining arguments when data is exhausted before all arguments are filled (no error); 2. Ignore successive delimiters between values.

Where: XX = ASCII code in hex notation for specified delimiter character.

NOTE:

Insufficient data in the buffer is signalled with an ERROR 97.

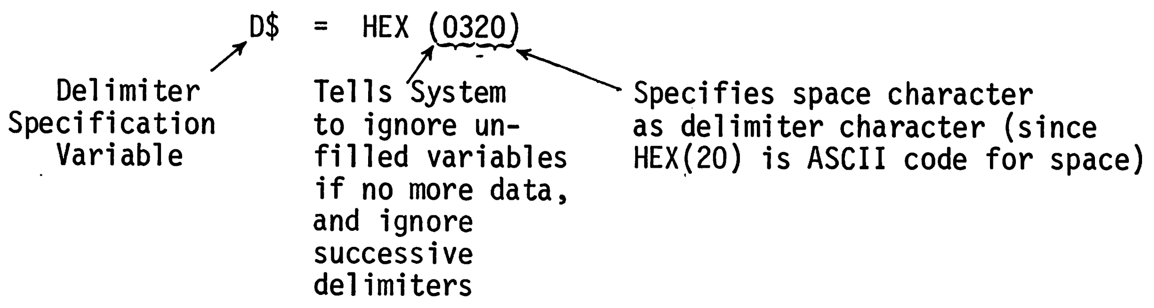

D$ = HEX (0320)
Delimiter Specification Variable
Tells System to ignore unfilled variables if no more data, and ignore successive delimiters
Specifies space character as delimiter character (since HEX(20) is ASCII code for space)

Figure 17-12
Typical Delimiter Specification Code

Assume, for example, that a Hollerith card image is read which consists of four values, an alphanumeric and three numerics (see Figure 17-13 below). The values are separated from each other with a slash ("/") character, HEX(2F).

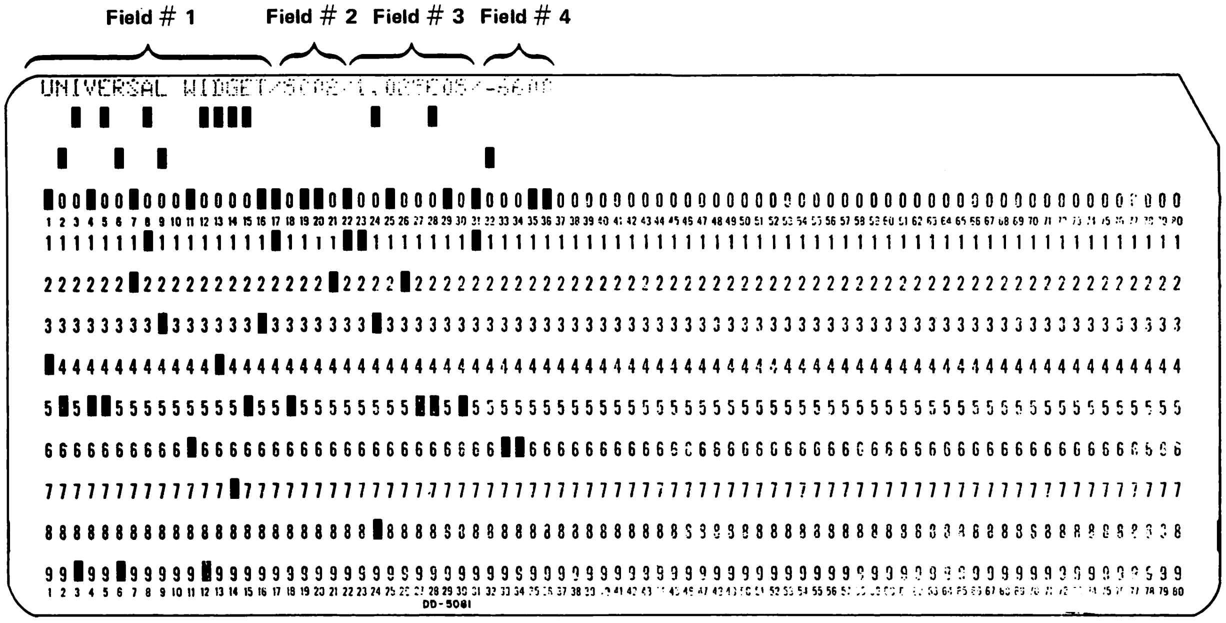


Figure 17-13.
Hollerith Card with Slash ("/") Characters
Serving as Delimiters Between Data Fields.

The values from this card image can be separated and stored in variables N$, S, A, and T, with the routine illustrated in Example 17-13 below.

Example 17-13: Using Delimiter Form of $UNPACK to Separate Data Values in a Hollerith Card Image

```
    10  ON ERROR E$, L$, GOTO 1000
    20  DIM A$(3)40, N$25
    30  DATALOAD BT (N=82) /629, A$()
    .
    .
    .
    .
    320 D$ = HEX (002F)
    330 $UNPACK (D=D$) A$() TO N$, S, A, T
    .
    .
    .
    .
    1000 (Error recovery routine)
```

The card image from Figure 17-13 is read into array A$() at line 30. At line 320, the delimiter specification code is stored in D$. The first byte of the code, HEX(00), specifies that an error 97 is to be generated if the data in the buffer is exhausted before all three variables in the argument list have been filled. HEX(00) also specifies that each successive delimiter between successive values on the card will be interpreted as a command to skip one variable in the argument list. The second byte of the specification code is the ASCII code for the designated delimiter character. In this case, HEX(2F) is the code in ASCII for a slash ("/") character. At line 320, four values (separated by the specified delimiter character) are unpacked sequentially from A$() into N$, S, A and T, in that order. The first value in A$() must be alphanumeric, and the next three values must be numeric; otherwise, an error is signalled. Note that if an error is produced by the $UNPACK statement, it is handled under program control by the ON ERROR GOTO statement, which directs program execution to line 1000 in this event.

CHAPTER 18
DESIGNING CUSTOMIZED CARDS (MODEL 2244A ONLY)

18.1 READING CUSTOMIZED CARDS WITH THE MODEL 2244A

With its ability to read both punched and mark sense cards, and to utilize printed index marks on a card to define data columns, the Model 2244A provides the flexibility to handle a wide range of custom-designed data cards. Punched and mark sense cards can be intermixed in the same deck, and, indeed, punches and marks may be combined on a single card. In many applications, specially designed data cards are prepunched with control information in certain columns, while the remaining columns are available to the user for marking data.

NOTE:

When punches are combined with marks on a single card, or punch cards are intermixed with mark sense cards in the same deck, all cards must be read with the DATA MODE switch set to OPT MARK. In that case, the ink used for printed material on punch cards must conform to the reflectance specifications for ink used on mark sense cards (see Section 18.3). Additionally, it is not in general possible to combine cards containing index marks with cards which do not have index marks in the same deck. Cards with index marks are read with the INDEX MARKS switch in the CLOCK position. Cards without index marks are read with INDEX MARKS set to NON-CLOCK.

Custom-designed cards which are marked and/or punched in Hollerith code can be read in Mode #3 (DATALOAD BT, Address 629); in this mode, the Hollerith is converted automatically to ASCII. Custom-designed cards which utilize any code other than Hollerith may be read in binary with Mode #4 (DATALOAD BT, Address 62A). In the latter case, the special code must be converted from binary into a meaningful form under software control. The bit and byte manipulation language features available on the Systems 2200B and 2200C, the WCS/20 and WCS/30, and in Option 22 for the System 2200S and WCS/10, are powerful tools for the manipulation and interpretation of binary data. To these statements, the General I/O statements (available in Option-2 for the 2200B & 2200C, in Option -23 for the 2200S and WCS/10, and as standard features on the WCS/20 and WCS/30) contribute an additional measure of power and versatility in certain code conversion applications.

Standard punched cards are divided into 80 vertical data columns. The data columns are, in turn, divided horizontally into 12 rows (see Figure 8-1). The number and width of data columns on specially designed cards are defined by the spacing of index marks along the card's bottom edge. The rows, however, must be uniformly spaced according to the specifications listed below, and are not subject to alteration.

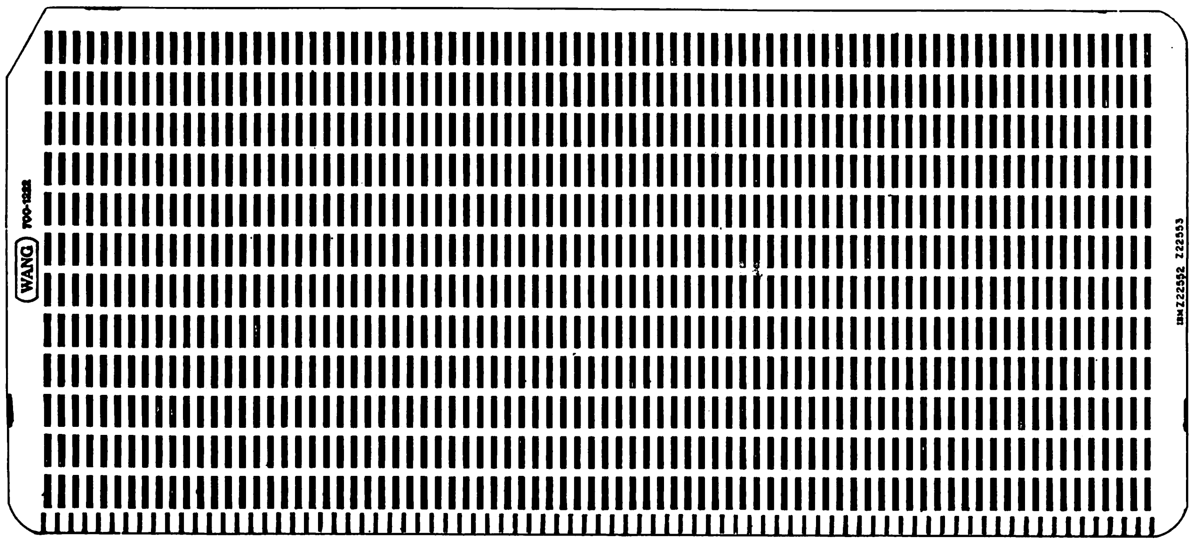


Figure 18-1. Standard 80-column, 12-Row Card with Index Marks for Marking or Punching (Wang Part No. 700-1222)

In general, punched cards need not contain index marks, since the spacing of punched data columns is set automatically by the keypunch, and conforms to punched card standards (as spelled out in ANSI specifications X3.21-1967, "Rectangular Holes in Twelve-Row Punched Cards"). If the punched card does have index marks, the index marks must be spaced according to the standards listed in Section 18.4, "Punch/Mark Sense Cards with Index Marks."

Custom-designed mark sense cards typically contain fewer than 80 columns, since the data columns on 80-column cards are too narrow for convenient labeling and marking of data. (The two special format mark sense BASIC program cards, for example, each have 37 columns.) In this case, index marks are used to delimit data columns. Punches and #2 pencil marks lying in the data field between consecutive index marks are read as data.

18.2 GENERAL CARD SPECIFICATIONS (MARK SENSE AND PUNCH CARDS)

Cards read by the Model 2234A or Model 2244A must conform to the specifications in "American National Standard Specifications for General Purpose Cards for Information Processing" (ANSI x3.11 - 1969). Card dimensions, paper requirements, and other detail requirements are set out concisely in the ANSI specifications. The major specifications are summarized below:

1. Card Dimensions. All cards must conform to the following dimensions:

   a) Height
      Min - 3.247 inches
      Max - 3.257 inches.

   b) Length
      Min - 7.370 inches
      Max - 7.380 inches

   c) Angles
      90 degrees ± 5 minutes

All edges must be straight, and opposite edges must be parallel within 0.003 inch.

A diagonal corner cut may be made in the upper left corner (preferred), or in the upper right corner of the card. The corner cut should remove 0.250 inch ± 0.016 inch from the long edge, and 0.433 ± 0.016 inch from the short edge of the card (at a reference angle of 60 degrees to the long edge of the card).

2. Paper Quality. The following standards apply to the paper stock used for cards.

   a) Paper Composition
      The paper should be composed of 100% chemical wood fibre.

   b) Grain
      The grain of the paper must be in the direction of the card length.

   c) Weight
      The paper must weigh 99 lbs ± 5% per ream of 500 sheets, 24 inches x 36 inches.

   d) Thickness
      The paper thickness must be 0.0070 inch ± 0.00040 inch.

   e) Bursting Strength
      The minimum bursting strength of the paper must be 55 lbs per square inch.

3. Reflectance.* The reflectance of the card stock, and of printed material on the card which is not meant to be read as data, must conform to the following standards:

   (a) Average Reflectance
       The average reflectance of the card must not fall below 80%. (Reflectance measurements taken using a Macbeth Standard Reflectance plaque as calibration standard.)

   (b) Printed Material
       Printed material and/or blemishes in the marking field of the card which are not to be read as data in OPT MARK mode must reflect at least 85% of the average reflectance of the card. (Thus, for example, a card whose average reflectance is 90% may not have a blemish or printed material whose reflectance is less than 0.85 x 0.90, or 76.5%.)

*Reflectance standards apply to all cards read in OPT MARK mode. This includes all mark sense cards, as well as punch cards which are to be intermixed with mark sense cards in the same deck and read in OPT MARK mode. The reflectance requirements do not apply to punch cards read in PUNCH mode only.

NOTE:

White and natural cards manufactured to card industry standards are, in general, satisfactory for the Model 2244A.

18.3 CUSTOM-DESIGNED MARK SENSE CARDS

The design specifications for mark sense cards with index marks allow for a great deal of flexibility in the design of customized cards. The relevant specifications are discussed in this section.

Definitions of Key Terms

Before embarking upon a discussion of custom card design, it may be helpful to define certain important terms.

a) Index Mark - One of the heavy black lines sequenced along the bottom edge of a card, and used to define data columns. Index marks are also commonly referred to as "timing marks" and "clock marks".

b) Index Column - An imaginary column created by extending an index mark vertically to the upper edge of the card (see Figure 18-2). No data marks can be placed in an index column.

c) Data Column - A vertical column bounded by two consecutive index columns (see Figure 18-2). The data column begins at the trailing edge of the first index mark, and ends at the leading edge of the second index mark. Data marks must be made within a data column. The width of a data column is determined by the spacing of index marks on the card.

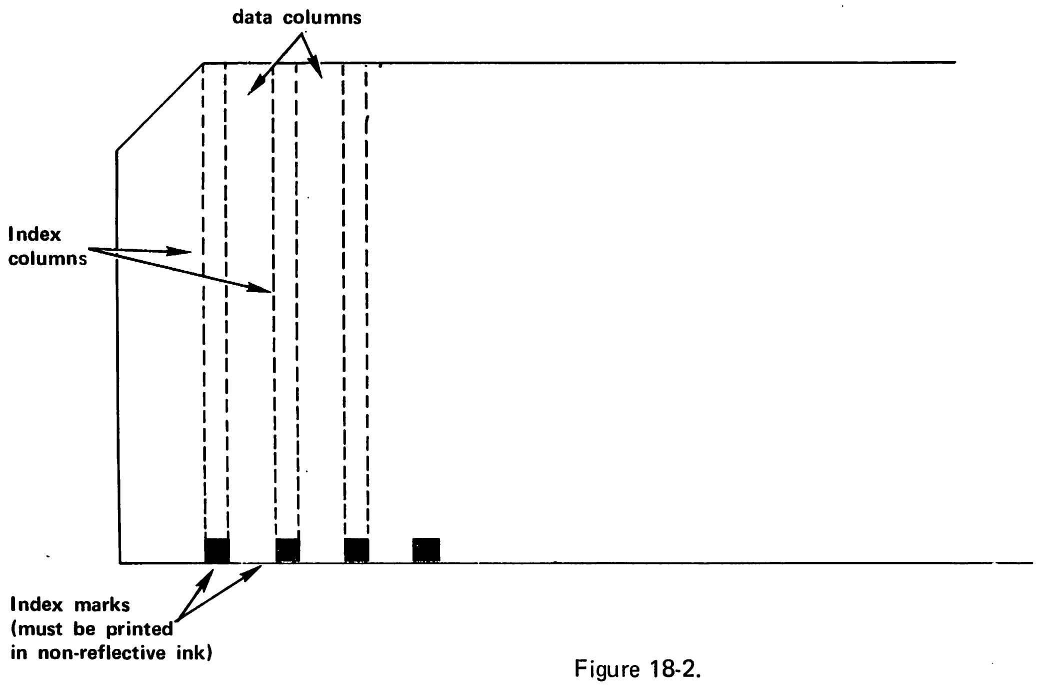


Figure 18-2.
Index Marks, Index Columns, and Data Columns.

d) Data Row - One of the twelve imaginary rows which run the length of the card horizontally, intersecting all data columns. The size and spacing of data rows is a constant which is determined by the alignment and sensitivity of the phototransistors in the reading head of the card reader. (See Figure 18-3.)

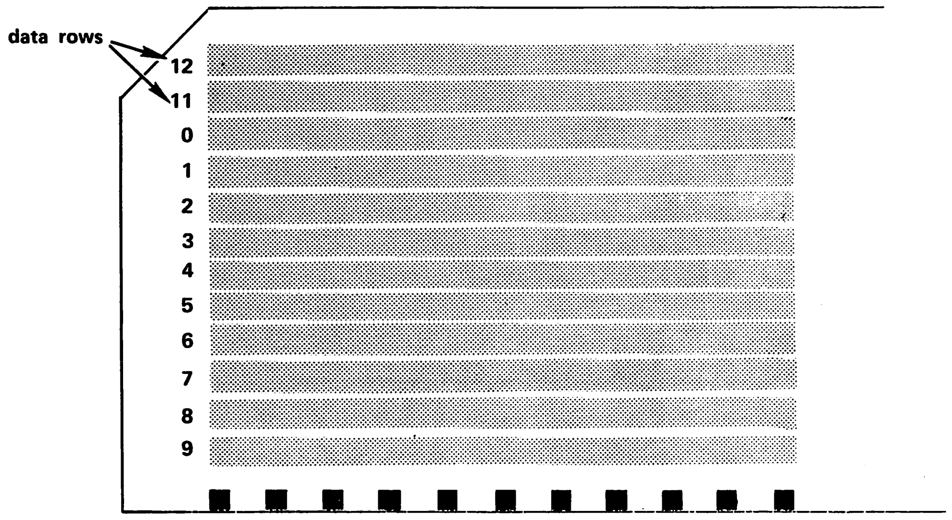


Figure 18-3.
Data Rows.

e) Marking Area - The area defined by the intersection of a data row with a data column (see Figure 18-4). Since there are 12 data rows, each column contains 12 marking areas. The marking areas in each column are the only areas on a card fully scanned by the 12 reading sensors as the card is read. In order to ensure accurate and consistent reading, therefore, each data mark must fall completely within the marking area.

f) Marking Constraint - A printed figure (such as a box, a circle, or a pair of parallel lines closed by dots at both ends) which locates and confines the data mark within a particular marking area. Often, marking constraints have identifying labels for marking data or program statements (see Figure 18-4).

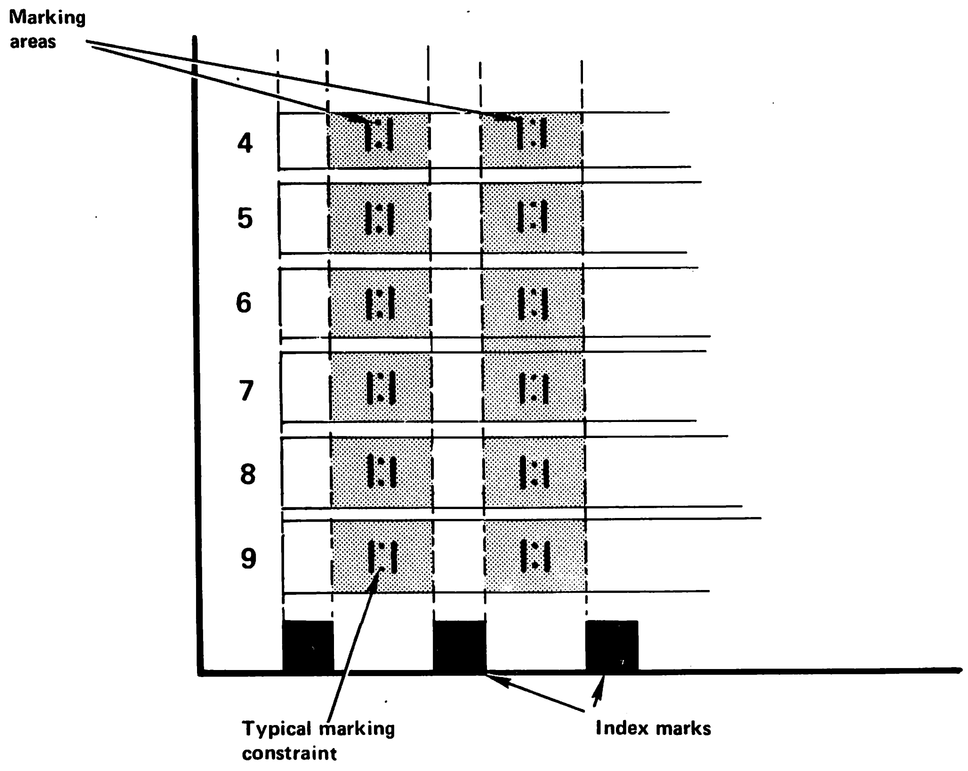


Figure 18-4.
Marking Areas and Typical Marking Constraints. (Marking areas are formed by overlap of rows and columns, and are indicated by heavily shaded areas. Marking constraints are used to confine the mark within a marking area.)

Detail Specifications for Mark Sense Cards

The specifications required to design a customized mark sense card include the dimensions and spacing of index marks (which define the vertical data columns), the width and spacing of horizontal data rows (which define the marking areas in each column), and the placement of data marks within a marking area.

1. Index Marks. The limiting dimensions and spacing restrictions for index marks are defined in the following specifications.

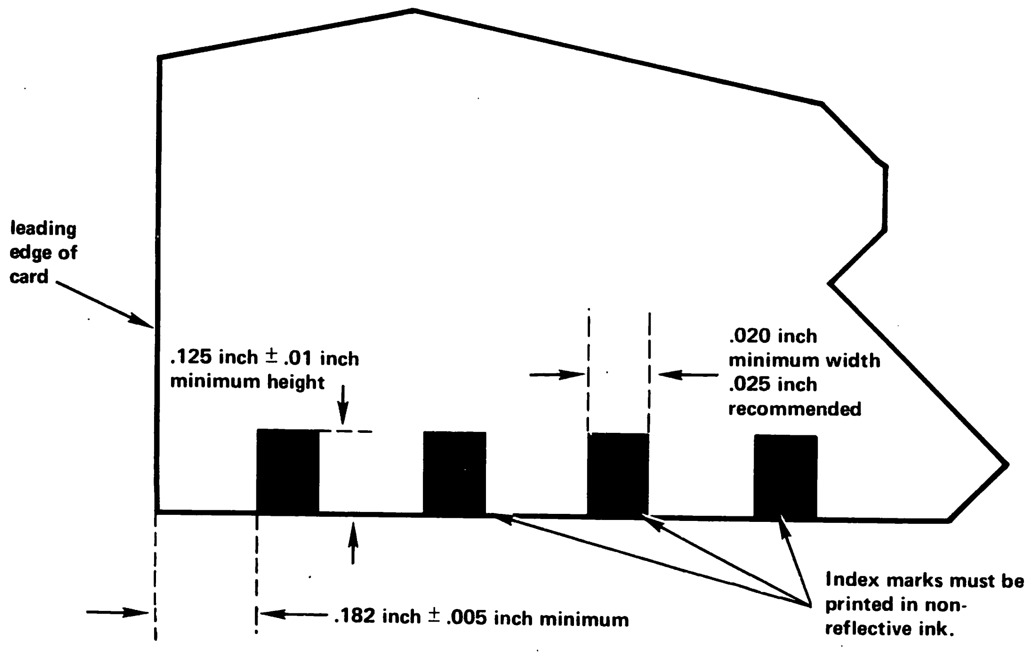

leading edge of card
.125 inch ± .01 inch minimum height
.020 inch minimum width .025 inch recommended
.182 inch ± .005 inch minimum
Index marks must be printed in non-reflective ink.

Figure 18-5.
Minimum Dimensions of Index Marks, and Minimum Distance Between First Index Mark and Leading Edge of Card.

a) Reflectance.
The index marks must be printed in a non-reflective ink which has an average reflectance less than or equal to 28% of the reflectance of that portion of the card immediately adjacent to the index marks.

b) Minimum distance from leading edge of card.
The minimum distance from the leading edge of the card to the leading edge of the first index mark is .182 inch ± .005 inch (see Figure 18-5).

c) Minimum width.
The minimum width of an index mark is .020 inch. The recommended width is .025 inch ± .005 inch. There is no maximum width. (See Figure 18-5.)

d) Minimum length.
The minimum length of an index mark is .125 inch ± .01 inch. This is also the recommended length. (See Figure 18-5.)

e) Minimum spacing between index marks.
The minimum distance between the trailing edge of a first and the leading edge of a second consecutive index mark (i.e., the minimum width of a data column) is .052 inch. (See Figure 18-6.)

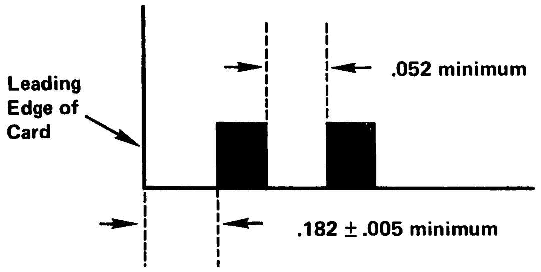


Figure 18-6.
Minimum Distance Between Consecutive Index Marks for Mark Sense Data Columns.

f) Maximum spacing between index marks.
The maximum distance between two consecutive index marks (i.e., the maximum width of a data column) is 2.175 inches. It is recommended, however, that the data columns not exceed .50 inch in width. (See Figure 18-7.)

g) Minimum distance from last index mark to trailing edge of card.
The minimum distance between the trailing edge of the last index mark and the trailing edge of the card is .190 inch. There is no maximum. (See Figure 18-7.)

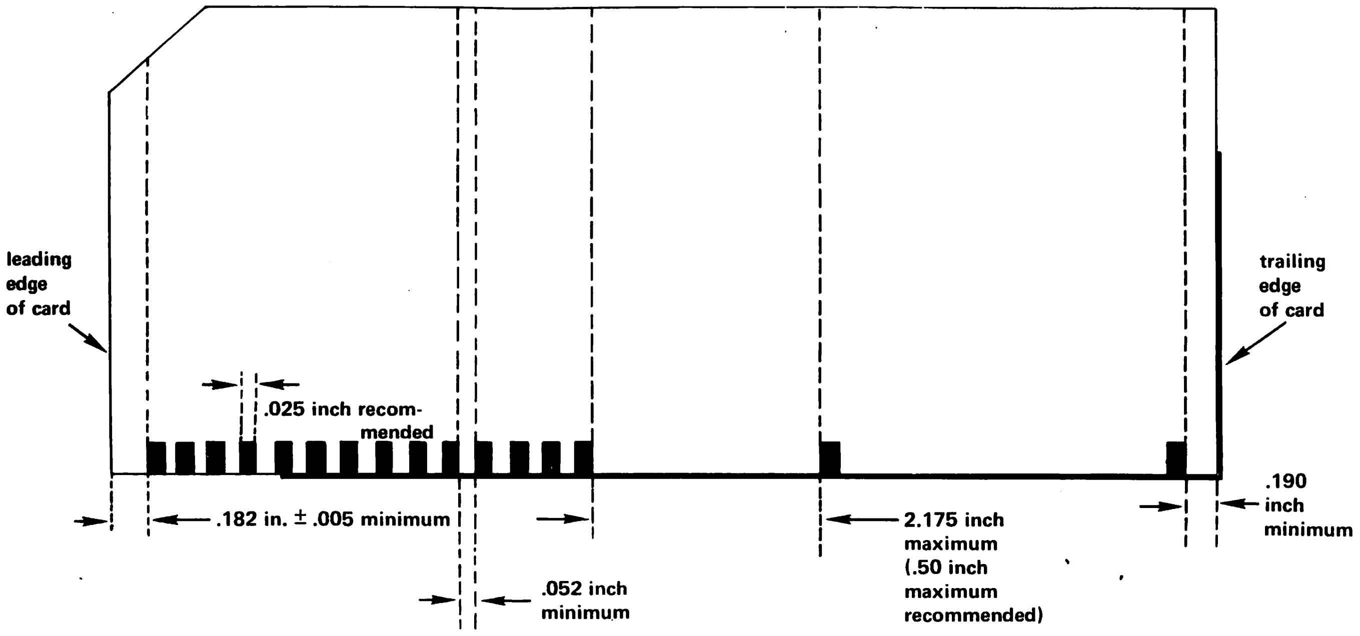


Figure 18-7.
Alignment of Index Marks on Card.

2. Data Rows and Marking Areas. The 12 marking areas in each column are defined by the intersection of the column with the 12 data rows. Data marks made in a data column outside a marking area may not be picked up by the card reader. Note that the spacing and size of the data rows are not defined by index marks or other printed material on a card, but are determined by the arrangement of the 12 photoelectric sensors in the card reader's reading head, and cannot be altered by the user.

   a) Distance from top edge of card to 12th row. The center line of the 12th data row is .250 inch from the top edge of the card. (See Figure 18-8.)

   b) Vertical spacing between data row centerlines. Data row centerlines are spaced .250 inch apart. (See Figure 18-8.)

   c) Width of data rows. Each data row has a maximum width of .240 inch (i.e., the marking area in a data row extends a maximum of .120 inch on each side of the centerline; see Figure 18-8).

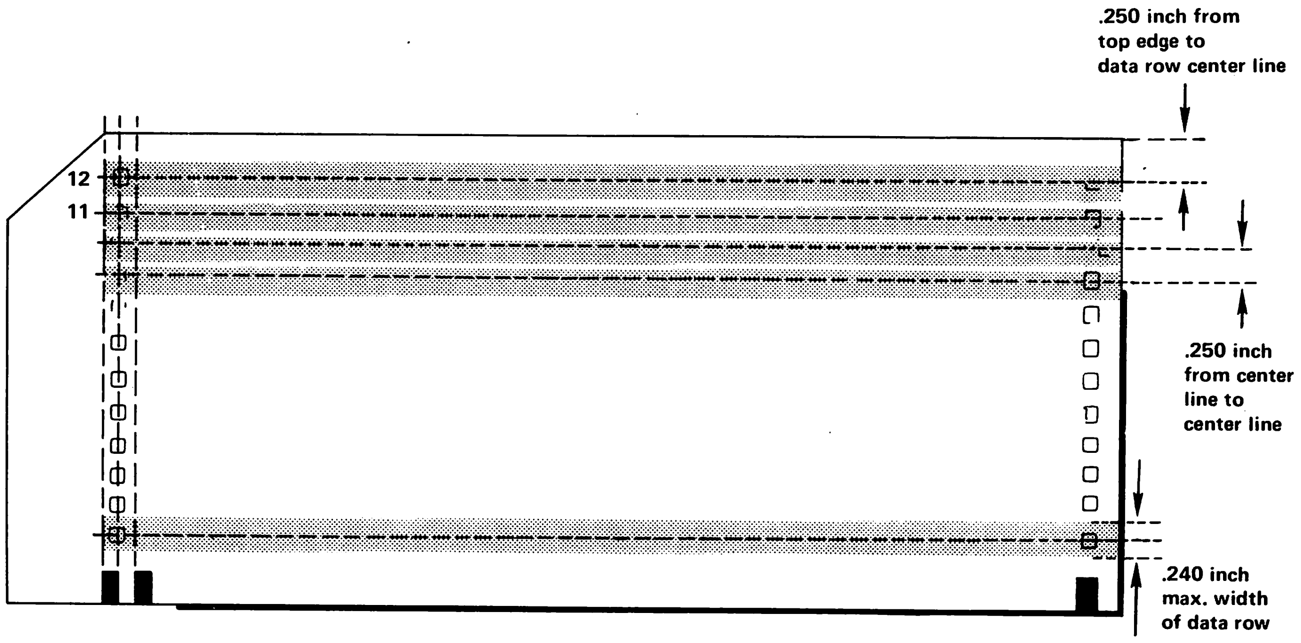

.250 inch from top edge to data row center line
12
11
.250 inch from center line to center line
.240 inch max. width of data row

Figure 18-8.
Maximum Width and Spacing of Data Rows.

3. Data Marks. A data mark must be a single-stroke vertical line made with a #2 pencil or equivalent.

   a) Minimum Width.
   The minimum width of a data mark is .015 inch. The maximum width is limited only by the width of the data column.

   b) Minimum length.
   The minimum length of a data mark is .125 inch, centered within the marking area. (See Figure 18-9.)

c) Maximum length.
The maximum length is .240 inch centered within the marking area. Data marks longer than .240 inch may extend into a neighboring data row, and cause a mark to be read in that row. (See Figure 18-9.)

4. Marking Constraints. Typically, marking constraints are used to locate and confine data marks within the marking areas. The minimum and maximum dimensions of a marking constraint must, therefore, conform to the dimensions of a valid data mark, as listed in paragraph 3 above. Recommended dimensions for marking constraints are listed below.

a) Reflectance.
Marking constraints and identifying labels must be printed in a reflective ink whose reflectance does not fall below 85% of the average background reflectance.

b) Vertical spacing of marking constraints. The centerline of the top (12th) row of marking constraints must be parallel to the top edge of the card, at a distance of .250 inch. Subsequent rows must be on .250 inch centers. (See Figure 18-9.)

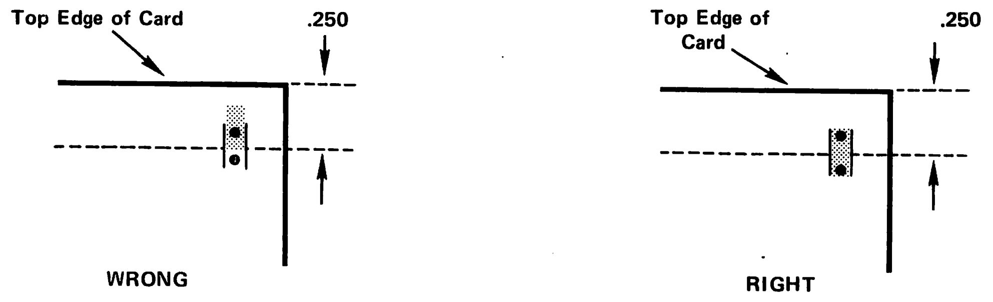


Figure 18-9.
Vertical Spacing of Data Marks and Data Constraints. Data Marks and Constraints Must Be Centered in Marking Areas. (Centerlines at .250 inch increments from top edge of card.)

c) Distance from index marks.
Marking constraints must be a minimum distance of .005 inch from the leading and trailing index columns. (See Figure 18-10.)

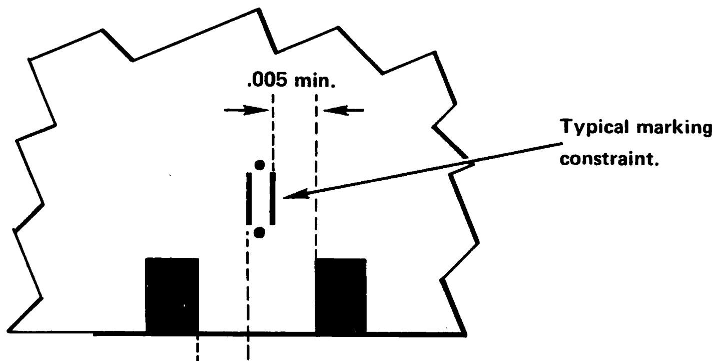


Figure 18-10.
Minimum Distance Between Marking Constraint and Index Marks.

d) Minimum recommended length.
   The recommended minimum length of a marking constraint is .150 inch ± .010.

e) Minimum recommended width.
   The recommended minimum width of a marking constraint is .030 inch ± .005.

18.4 STANDARD PUNCHED CARDS WITHOUT INDEX MARKS

Standard 80-column punched cards without index marks must conform to the specifications described in the USA Standard publication "Rectangular Holes in Twelve-Row Punched Cards" (ANSI x3.21-1967). Column spacing and punch registration on punched cards is determined automatically by the keypunch unit; in general, all keypunch units (including the IBM Models 26 and 29) are designed to conform to the ANSI x3.21-1967 standards.

18.5 PUNCH/MARK SENSE CARDS WITH INDEX MARKS

Combining Punches and Marks on Custom-Designed Cards

With the DATA MODE switch set to OPT MARK, the Model 2244A is capable of reading a combination of punched and mark sense cards in the same deck, as well as a mixture of punches and marks on the same card. In such cases, the width and spacing of data columns in the punch field must conform to the requirements of the keypunch unit. Wang Laboratories offers an 80-column card which can be used interchangeably as a mark sense or punch card (see Figure 18-1). This card (Wang Part No. 700-1222) contains index marks, and therefore can be read with the INDEX MARKS switch in CLOCK mode, the standard setting for mark sense cards. In addition, all data constraints on the card are printed in a reflective ink which is not picked up as data when the card is read in OPT MARK mode. Card #700-1222 provides the capability to intermix punch and mark sense cards in a single deck, or to intersperse punched and marked columns on the same card.

Card #700-1222 is a standard 80-column card which, though it can be marked, is designed primarily for use as a punch card. The narrowness of the data columns makes them somewhat inconvenient for marking, and there is no provision for the placement of identifying labels under specific boxes. As it stands, the card could be meaningfully interpreted only if it is marked in Hollerith code - a restriction which also may be inconvenient for certain applications. If the user wishes to custom-design his own card, he may utilize a number of columns for punch fields, while the remainder of the card is designed for marking. In this case, the placement of punch columns must conform to the exacting requirements of the keypunch unit, but the mark sense fields can be expanded to a size convenient for marking (subject to the very flexible restrictions spelled out in Section 18.3). Within the mark sense data columns, individual marking constraints can be tagged with identifying labels to clarify and simplify the marking process.

Detail Specifications for Punch Data Fields

The general specifications for mark sense data columns are covered in Section 18.3. The more rigorous specifications applying to punch data columns are listed below.

NOTE:

If punch fields and mark fields are to be combined on the same card, it is recommended that all punch columns be sequenced from the leading edge of the card, followed by the mark sense columns. The alternation of punch and mark sense data columns, or the placement of punch columns starting in the middle of a card, involves exceedingly complex design considerations (except in the case of a standard 80-column card in which all punch and mark sense columns are the same width), and is not recommended.

1. Card Stock and Dimensions. Cards used for punching only, or for both punching and marking, must conform to the specifications for card stock and card dimensions listed in Section 18.2, paragraphs 1 and 2.

2. Reflectance of Card Stock and Printed Material. The average reflectance of the card stock, and of the ink used in printed material on a card not meant to be read as data, must conform to the specifications in Section 18.2, paragraph 3.

3. Dimensions of Data Constraints. The recommended dimensions of data constraints for punch/mark cards are:

   Width - .030 $\pm$ .005 inch
   Height - .150 $\pm$ .010 inch

   The average size of a standard rectangular punch hole is 0.055 x 0.125. The recommended data constraint is therefore somewhat longer and narrower than the standard punch hole. In this way, it is less likely that a mark will accidentally be made wide enough to expand into a neighboring column if the data constraint is marked rather than punched.

4. Distance from Leading Edge of Card to First Punch Column. The distance from the leading edge of the card to the centerline of the first punch data column is .251 $\pm$ .005 inch. (See Figures 18-11, 18-13.)

5. Distance from First Index Mark to First Punch Column. The location of the leading index mark is determined with reference back from the first punch data column rather than to the leading edge of the card. The centerline of the first index mark must be .050 inch $\pm$ .005 from the centerline of the first punch data column. (Refer to Figures 18-11, 18-13.)

6. Dimensions of Index Marks.

   a) Height
      The recommended height of index marks is .125 inch ± .010.

   b) Width
      The recommended width is .025 inch ± .005.

7. Color and Reflectance of Index Marks. Index marks should be printed in a black, non-reflective ink which has an average reflectance less than or equal to 28% of the reflectance of that portion of the card immediately adjacent to the index marks.

8. Width of Punch Data Columns. The distance between the centerlines of consecutive index marks (i.e., the width of data columns) must be .087 inch ± .005 (refer to Figures 18-11,18-13.)

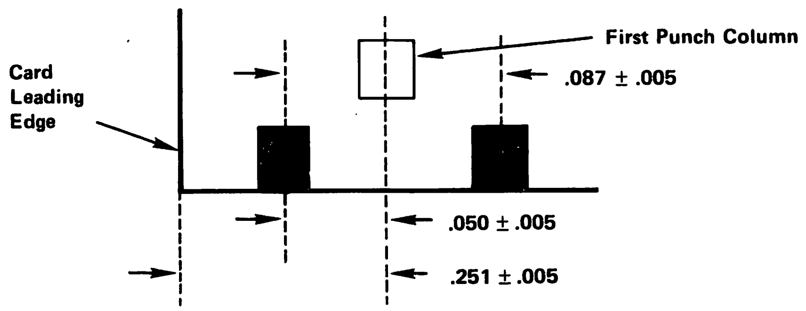


Figure 18-11.
Detail Specifications for First Punch Data Column and Spacing of First and Subsequent Index Marks.

9. Vertical Spacing of Data Constraints. The centerline of the top (12th) row of data constraints must run parallel to the top edge of the card at a distance of .250 inch ± .005. Successive rows of data constraints are on .250 inch centers. (See Figure 18-13.)

10. Punch Overlap into Index Column. Punch holes may overlap marginally from the data column into the leading or trailing index column. In such cases, the leading edge of a punch hole which overlaps the leading index mark must be a minimum of .015 inch from the leading edge of the index column (i.e., the punch hole may overlap a maximum of .010 inch into the index column). Similarly, the trailing edge of a punch mark must be a minimum distance of .015 inch from the trailing edge of the index mark. (See Figure 18-12.)

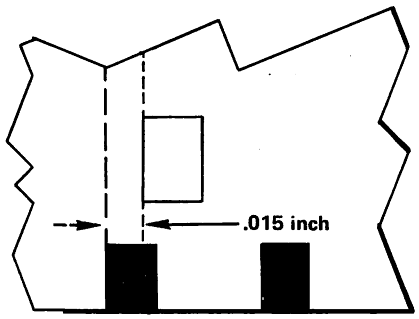


Figure 18-12.
Minimum Distance Between Leading Edge of Punch Hole and Leading Edge of Index Mark.

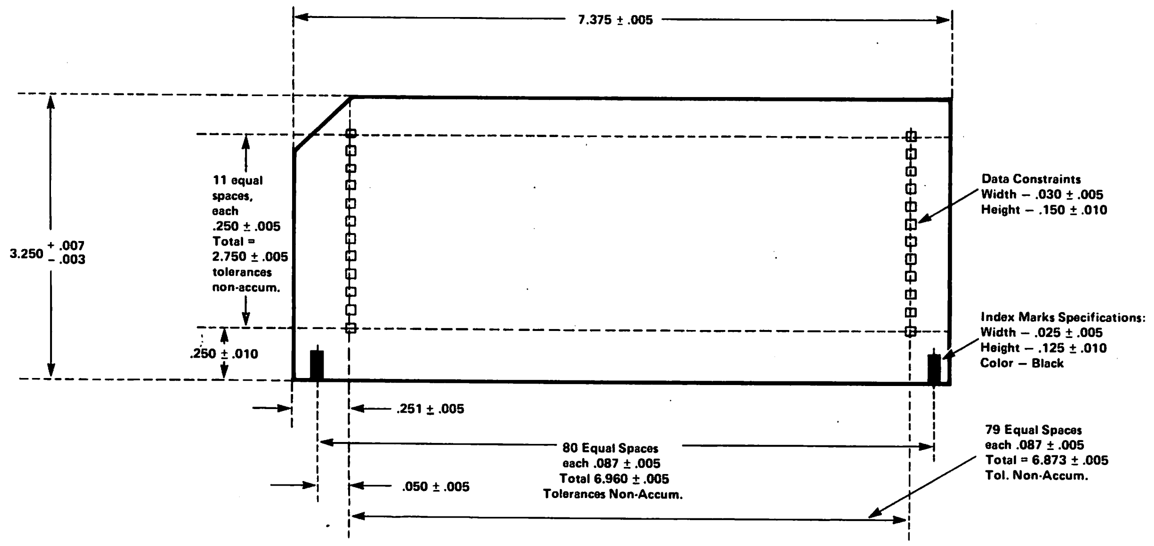

7.375 ± .005
3.250 + .007 − .003
11 equal spaces, each .250 ± .005 Total = 2.750 ± .005 tolerances non-accum.
.250 ± .010
Data Constraints Width – .030 ± .005 Height – .150 ± .010
Index Marks Specifications: Width – .025 ± .005 Height – .125 ± .010 Color – Black
.251 ± .005
80 Equal Spaces each .087 ± .005 Total 6.960 ± .005 Tolerances Non-Accum.
.050 ± .005
79 Equal Spaces each .087 ± .005 Total = 6.873 ± .005 Tol. Non-Accum.

Figure 18-13.
Card Specifications for 80-Column Punch/Mark Sense Card.
(Note: Data constraints and other printed matter must be printed in reflective ink [see paragraph #2, Section 18.5.]
Index marks must be printed in non-reflective ink [see paragraph 7, Section 18.5.]

APPENDIX A
GENERAL FORMS OF THE CARD READER STATEMENTS AND COMMANDS

All of the BASIC statements used with the card reader have applications for a number of other input and input/output devices. General discussions of these statements are found in the system Reference Manual. The general form and discussion of each statement and command in this Appendix focuses specifically upon the statement's applicability to the card reader and to card reading operations.

HOLLERITH STATEMENTS/COMMANDS

HOLLERITH DATA VALUES (DATALOAD)

DATALOAD $\begin{bmatrix} /628, \\ \#n, \end{bmatrix}$ argument list

Examples:

```
10 DATALOAD /628, A, B, C

10 SELECT #3 628
20 DATALOAD #3, N( ), B$( )

10 SELECT TAPE 628
20 DATALOAD A$, B$, N, O
```

Reference:
Chapter 4

HOLLERITH DATA VALUES (INPUT)

SELECT INPUT 62B
INPUT "character string", argument list

Examples:

```
10 SELECT INPUT 62B
20 INPUT A, B, N$

10 SELECT INPUT 62B
20 INPUT "READING DATA CARDS", A$, B$, N
```

Reference:
Chapter 5

HOLLERITH CARD IMAGE (DATALOAD BT)

DATALOAD BT (N=82) $\begin{bmatrix}\text{/629,}\\ \text{\#n,}\end{bmatrix}$ alpha array designator

Examples:

```
  5 DIM A$(3)40
 10 DATALOAD BT (N=82) /629, A$( )

  5 DIM A$(3)40
 10 SELECT #3 629
 20 DATALOAD BT (N=82) #3, A$( )

  5 DIM B1$ (3)40
 10 SELECT TAPE 629
 20 DATALOAD BT (N=82) B1$( )
```

Reference:
Chapter 6

HOLLERITH LOOK-AHEAD (DATASAVE BT)

DATASAVE BT $\begin{bmatrix}\text{/42E,}\\ \text{\#n,}\end{bmatrix}$ alpha variable

Examples:

```
 10 DIM F$1
100 DATASAVE BT /42E, F$

 90 DIM A$(3)40, F$1
100 DATASAVE BT /42E, F$
  .
  .
  .
200 DATALOAD BT (N=82)/629, A$( )
```

Reference:
Chapter 7

HOLLERITH PROGRAM LOADING (LOAD COMMAND)

LOAD $\begin{bmatrix}\text{/62B,}\\ \text{\#n}\end{bmatrix}$

Examples:

```
CLEAR
LOAD /62B

SELECT #5 62B
LOAD #5
```

Reference:
Chapter 8

HOLLERITH PROGRAM LOADING (LOAD STATEMENT)

```
      ⎡/62B,⎤
LOAD  ⎣#n,  ⎦    [L1]   [,L2]
```

Examples:

```
100 LOAD /62B, 100, 300

100 SELECT #3 62B
110 LOAD #3, 100

100 LOAD /62B
```

Reference:
Chapter 8

HOLLERITH PROGRAM LOADING (CONSOLE INPUT)

```
SELECT CI 02B         (All program text and system
     .                 commands entered from cards.)
     .
     .
SELECT CI 001         (After all cards are read, last card
                       should return CI to keyboard.)
```

Reference:
Chapter 9

BASIC MARK SENSE PROGRAM LOADING (LOAD COMMAND)

LOAD $\begin{bmatrix} \text{/62C} \\ \text{\#n} \end{bmatrix}$

Examples:

```
CLEAR
LOAD /62C

SELECT #4 62C
CLEAR
LOAD #4
```

Reference:
Chapter 11

BASIC MARK SENSE PROGRAM LOADING (LOAD STATEMENT)

LOAD $\begin{bmatrix} \text{/62C,} \\ \text{\#n,} \end{bmatrix}$ [L1] [,L2]

Examples:

```
100 LOAD /62C, 100, 300

 90 SELECT #2 62C
100 LOAD #2, 100

 90 SELECT TAPE 62C
100 LOAD
```

Reference:
Chapter 11

BASIC MARK SENSE DATA VALUES (DATALOAD)

DATALOAD $\begin{bmatrix} \text{/62D,} \\ \text{\#n,} \end{bmatrix}$ argument list

Examples:

```
10 DATALOAD /62D, A$, B, S$

10 SELECT #1 62D
20 DATALOAD #1, A$( ), B( )

10 SELECT TAPE 62D
20 DATALOAD N( ), B$( ), A$, N
```

Reference:
Chapter 12

Reference:
Chapter 13

BASIC MARK SENSE PROGRAM LOADING (CONSOLE INPUT)

```
SELECT CI 02C        (All program text and system commands
      .               entered from cards.)
      .
      .
SELECT CI 001        (After all cards are read, last card should
                      return CI to keyboard.)
```

Reference:
Chapter 14

BINARY CARD IMAGE (DATALOAD BT)

DATALOAD BT (N=162) $\begin{bmatrix}\text{/62A,}\\\text{\#n,}\end{bmatrix}$ alpha array designator

Examples:

```
10 DIM A$(5)40
20 DATALOAD BT (N=162) /62A, A$( )

10 DIM B$(5)40
20 SELECT #3 62A
30 DATALOAD BT (N=162) #3, B$( )

10 DIM A$(5)40
20 SELECT TAPE 62A
30 DATALOAD BT (N=162) A$( )
```

Reference:
Chapter 15

BINARY LOOK-AHEAD (DATASAVE BT)

DATASAVE BT $\begin{bmatrix}\text{/42F,}\\\text{\#n,}\end{bmatrix}$ alpha variable

Examples:

```
100 DIM F$1
110 DATASAVE BT /42F, F$

100 DIM A$(5)40, F$ 1
110 DATASAVE BT /42F, F$
 .
 .
 .
200 DATALOAD BT (N=82)/62A, A$( )
```

Reference:
Chapter 16

APPENDIX B
CARD PURCHASING INFORMATION

Wang Laboratories stocks and sells three types of cards for use on the Models 2234A and 2244A card readers. All three types of cards are printed in non-reflective ink, and contain index marks; all three are therefore suitable for reading in either PUNCH or OPT MARK mode. This fact is not significant for Model 2234A owners, since the Model 2234A reads only 80-column punch cards in any case. For Model 2244A owners who wish to intermix punch and mark sense cards in a single deck, however, it is important that the printed matter on punch cards meet the reflectance criteria imposed by the OPT MARK reading mode.

The three types of cards available are illustrated below. For detailed pricing information on these cards, write or call the Software Sales Department:

Wang laboratories, Inc.
Software Sales Department
836 North Street
Tewksbury, MA 01876
Tel. (617) 851-4111, extension 2617

When ordering or inquiring about cards, be sure to specify the correct part number:

80-Column Hollerith Punch/Mark Sense Card - #701-1222

40-Column Hollerith Punch/Mark Sense Card - #701-1223

Wang BASIC Mark Sense Card - #701-1224

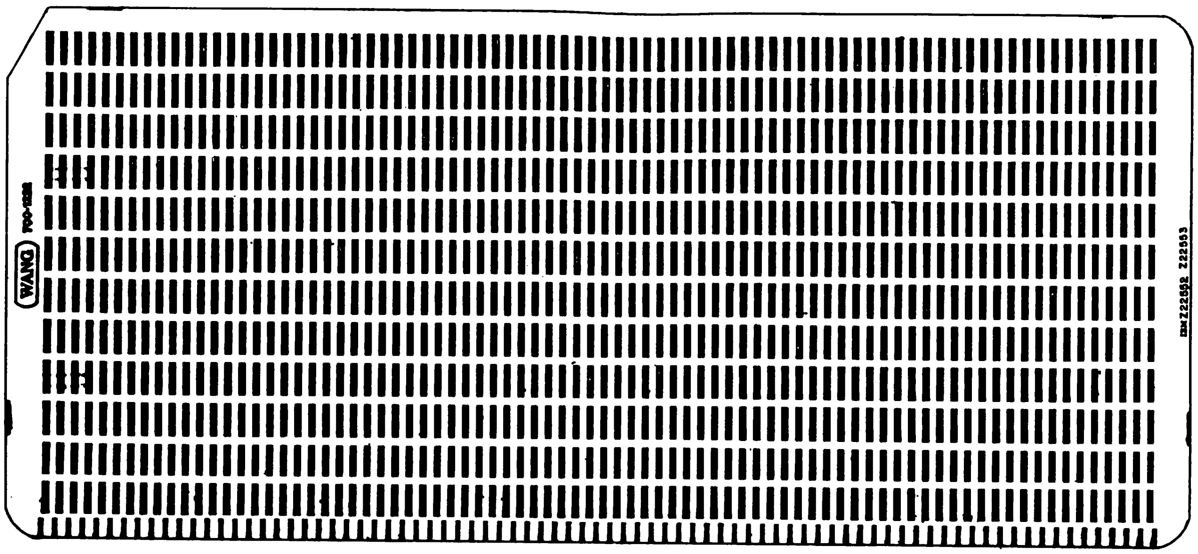


Figure B-1.
80-Column Hollerith Punch/Mark Sense Card
(Wang Part #701-1222)

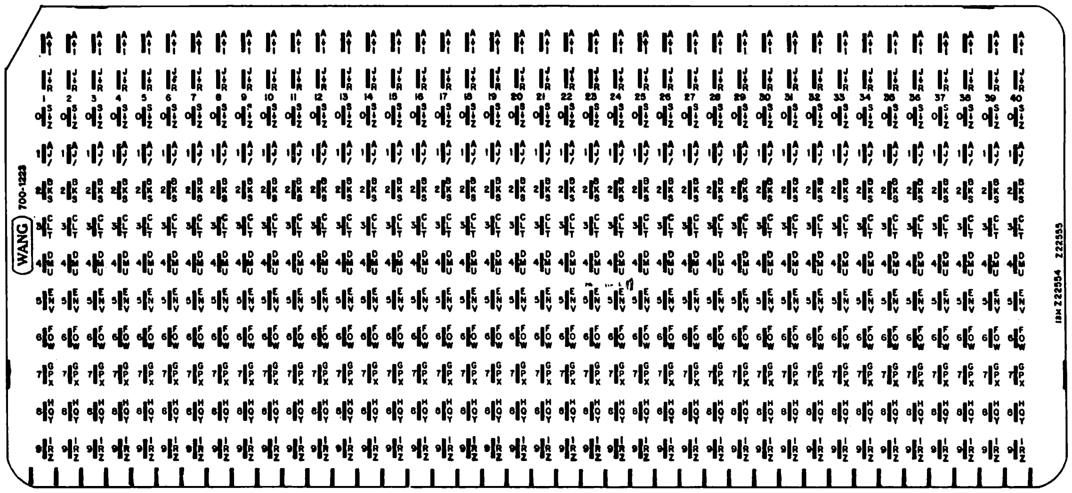


Figure B-2.
40-Column Hollerith Punch/Mark Sense Card
(Wang Part #701-1223)

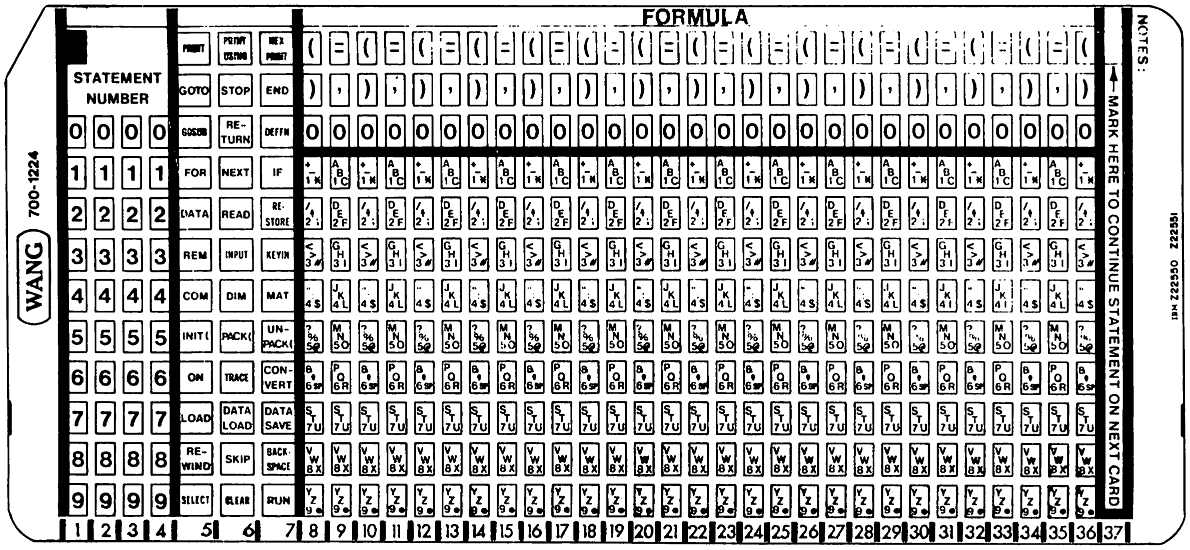


Figure B-3.
Wang BASIC Mark Sense Program/Data Card
(Wang Part #701-1224)

APPENDIX C
CARD READER MAINTENANCE INFORMATION

PREVENTIVE MAINTENANCE INFORMATION

It is recommended that your card reader be serviced semi-annually. A Wang Maintenance Agreement is available to assure this servicing automatically. If you do not purchase a Maintenance Agreement, all servicing must be arranged for by you. A Maintenance Agreement protects your investment and offers the following benefits:

Preventive Maintenance

Your equipment is inspected semi-annually for worn parts, lubricated, cleaned and updated with any engineering changes. Preventive maintenance minimizes "downtime" by anticipating repairs before they are necessary.

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. More frequent billing can be arranged, if desired.

Further information regarding Maintenance Agreements can be obtained from your local Sales-Service office.

NOTE:

Wang Laboratories, Inc. neither honors Maintenance Agreements for nor guarantees equipment modified by the user. Damage to equipment incurred as a result of such modification is the financial responsibility of the user.

OPERATOR MAINTENANCE

Although most preventive maintenance procedures for the card reader must be carried out by qualified service personnel, there are several elementary cleaning procedures which can be performed by the operator.

Exterior Cleaning

In a typical environment, the exterior of the card reader should be cleaned following each 40 hours of operation. If persistent dirt buildup is present, the exterior should be cleaned as frequently as possible. Simply wipe the housing thoroughly with a clean, lint-free cloth saturated with a mild solvent (such as denatured alcohol or household ammonia). Regular cleaning will keep the card reader's anodized finish bright and attractive, and will reduce the likelihood of operational problems resulting from excess dirt buildup.

Read Station Cleaning

The following procedure should be observed to clean the card reader's LED light source and reading sensors. In a typical environment, it is recommended that the read station be cleaned following each 40 hours of operation. In a particularly dusty environment, or in an application involving extensive processing of mark sense cards, the read station may require more frequent attention.

NOTE:

Frequent and recurrent reading errors may be indicative of excess dirt accumulation in the read station. If you encounter this problem, clean the reading station (use the method described below), and check for recurrence of the problem (allow three or four minutes for the solvent to evaporate). If the problem is not corrected, call your field service representative.

1. Saturate several unused (unmarked, unpunched) cards with alcohol. For the Model 2234A, use four or five cards; for the Model 2244A, use two or three cards. The gap between the light station and reading head varies marginally from unit to unit; experiment with your unit to determine the number of cards to be used.

2. Pass the cards <u>in a stack</u> (not individually) into the read station. Push the picker sector forward to start the cards through the rollers. (The picker sector is a perforated rubber insert on the bottom of the input hopper in front of the entrance to the read station; it moves on a pivot.) Push the cards through the rollers into the read station.

3. When enough of the stack emerges into the stacker to grip, pull the cards through and out of the read station. Repeat this procedure three or four times to ensure that everything is thoroughly cleaned.

APPENDIX D
HOLLERITH CODES AND ASCII EQUIVALENTS

TABLE I – Hollerith Codes
TABLE II – HEX, ASCII, and Hollerith Codes for BASIC Characters and Text Atoms

TABLE I. HOLLERITH CODES

	12 11 0 9	12 11 9	12 0 9	12 9	11 0 9	11 9	0 9	9	12 11 0	12 11	12 0	12	11 0	11	0	Blank
Blank	NP E1	r 72	i 69	I 49	z 7A	R 52	Z 5A	9 39	TA BA	| 7C	{ 7B	& 26	} 7D	- 2D	0 30	SP 20
1	TA BB	TA A9	TA A0	NP 01	TA 9F	NP 11	TA 81	TA 91	NP D9	j 6A	a 61	A 41	— 7E	J 4A	/ 2F	1 31
2	TA BC	TA AA	TA A1	NP 02	TA B2	NP 12	TA 82	NP 16	TA DA	k 6B	b 62	B 42	s 73	K 4B	S 53	2 32
3	TA BD	TA AB	TA A2	NP 03	TA B3	NP 13	TA 83	TA 93	NP DB	l 6C	c 63	C 43	t 74	L 4C	T 54	3 33
4	TA BE	TA AC	TA A3	TA 9C	TA B4	TA 9D	TA 84	TA 94	TA DC	m 6D	d 64	D 44	u 75	M 4D	U 55	4 34
5	TA BF	TA AD	TA A4	NP 09	TA B5	TA 85	LF 0A	TA 95	TA DD	n 6E	e 65	E 45	v 76	N 4E	V 56	5 35
6	TA C0	TA AE	TA A5	TA 86	NP B6	NP 08	NP 17	TA 96	TA DE	o 6F	f 66	F 46	w 77	O 4F	W 57	6 36
7	TA C1	TA AF	TA A6	NP 7F	NP B7	NP 87	NP 1B	NP 04	TA DF	p 70	g 67	G 47	x 78	P 50	X 58	7 37
8	TA C2	TA B0	TA A7	TA 97	TA B8	NP 18	TA 88	TA 98	NP E0	q 71	h 68	H 48	y 79	Q 51	Y 59	8 38
8-1	TA 90	NP 10	NP 00	TA 8D	TA 80	NP 19	TA 89	TA 99	NP D8	TA CA	TA C3	TA A8	NP D1	TA B1	NP B9	` 60
8-2	NP FA	NP EE	NP E8	TA 8E	NP F4	NP 92	TA 8A	TA 9A	NP E2	TA CB	TA C4	[5B	TA D2	] 5D	\ 5C	: 3A
8-3	NP FB	NP EF	NP E9	NP 0B	NP F5	TA 8F	TA 8B	TA 9B	NP E3	TA CC	TA C5	. 2E	TA D3	$ 24	, 2C	# 23
8-4	NP FC	NP F0	NP EA	NP 0C	NP F6	NP 1C	TA 8C	NP 14	NP E4	TA CD	TA C6	< 3C	TA D4	* 2A	% 25	@ 40
8-5	NP FD	NP F1	NP EB	CR 0D	NP F7	NP 1D	NP 05	NP 15	NP E5	TA CE	TA C7	(28	TA D5	) 29	_ 5F	' 27
8-6	NP FE	NP F2	NP EC	NP 0E	NP F8	NP 1E	NP 06	TA 9E	NP E6	NP CF	TA C8	+ 2B	TA D6	; 3B	> 3E	= 3D
8-7	NP FF	NP F3	NP ED	NP 0F	NP F9	NP 1F	NP 07	NP 1A	NP E7	NP D0	TA C9	! 21	NP D7	^ 5E	? 3F	" 22

NP = non-printable character
TA = Text atom (refer to Table II for text atom associated with listed HEX code).

TABLE II. HEX, ASCII, and HOLLERITH Codes for BASIC Characters and Text Atoms

BASIC SYMBOL (TEXT ATOM)	HEX CODE	ASCII CODE	HOLLERITH CODE (Rows Punched)
LINE FEED (LF)	0A	00001010	0-9-5
CARRIAGE RETURN (CR)	0D	00001101	12-9-8-5
X-ON	11	00010001	11-9-1
X-OFF	13	00010011	11-9-3
SPACE	20	00100000	BLANK
!	21	00100001	12-8-7
DOUBLE QUOTE	22	00100010	8-7
#	23	00100011	8-3
$	24	00100100	11-8-3
%	25	00100101	0-8-4
&	26	00100110	12
SING.QUOTE	27	00100111	8-5
(	28	00101000	12-8-5
)	29	00101001	11-8-5
*	2A	00101010	11-8-4
+	2B	00101011	12-8-6
, (comma)	2C	00101100	0-8-3
-	2D	00101101	11
. (decimal point)	2E	00101110	12-8-3
/ (slash)	2F	00101111	0-1
0	30	00110000	0
1	31	00110001	1
2	32	00110010	2

BASIC SYMBOL (TEXT ATOM)	HEX CODE	ASCII CODE	HOLLERITH CODE (Rows Punched)
3	33	00110011	3
4	34	00110100	4
5	35	00110101	5
6	36	00110110	6
7	37	00110111	7
8	38	00111000	8
9	39	00111001	9
:	3A	00111010	8-2
;	3B	00111011	11-8-6
<	3C	00111100	12-8-4
=	3D	00111101	8-6
>	3E	00111110	0-8-6
?	3F	00111111	0-8-7
@	40	01000000	8-4
A	41	01000001	12-1
B	42	01000010	12-2
C	43	01000011	12-3
D	44	01000100	12-4
E	45	01000101	12-5
F	46	01000110	12-6
G	47	01000111	12-7
H	48	01001000	12-8
I	49	01001001	12-9
J	4A	01001010	11-1
K	4B	01001011	11-2
L	4C	01001100	11-3

BASIC SYMBOL (TEXT ATOM)	HEX CODE	ASCII CODE	HOLLERITH CODE (Rows Punched)
M	4D	01001101	11-4
N	4E	01001110	11-5
O	4F	01001111	11-6
P	50	01010000	11-7
Q	51	01010001	11-8
R	52	01010010	11-9
S	53	01010011	0-2
T	54	01010100	0-3
U	55	01010101	0-4
V	56	01010110	0-5
W	57	01010111	0-6
X	58	01011000	0-7
Y	59	01011001	0-8
Z	5A	01011010	0-9
[	5B	01011011	12-8-2
]	5D	01011101	11-8-2
↑ (Up Arrow)	5E	01011110	11-8-7
_ (Underline)	5F	01011111	0-8-5
a	61	01100001	12-0-1
b	62	01100010	12-0-2
c	63	01100011	12-0-3
d	64	01100100	12-0-4
e	65	01100101	12-0-5
f	66	01100110	12-0-6
g	67	01100111	12-0-7
h	68	01101000	12-0-8
i	69	01101001	12-0-9
j	6A	01101010	12-11-1

BASIC SYMBOL (TEXT ATOM)	HEX CODE	ASCII CODE	HOLLERITH CODE (Rows Punched)
k	6B	01101011	12-11-2
l	6C	01101100	12-11-3
m	6D	01101101	12-11-4
n	6E	01101110	12-11-5
o	6F	01101111	12-11-6
p	70	01110000	12-11-7
q	71	01110001	12-11-8
r	72	01110010	12-11-9
s	73	01110011	11-0-2
t	74	01110100	11-0-3
u	75	01110101	11-0-4
v	76	01110110	11-0-5
w	77	01110111	11-0-6
x	78	01111000	11-0-7
y	79	01111001	11-0-8
z	7A	01111010	11-0-9
#PI	CC	11001100	12-11-8-3
ABS(	C1	11000001	12-11-0-9-7
AND	8A	10001010	0-9-8-2
ARC	CB	11001011	12-11-8-2
ATN(	D4	11010100	11-0-8-4
BA	BE	10111110	12-11-0-9-4
BACKSPACE	AB	10101011	12-11-9-3
BEG	B3	10110011	11-0-9-3
BIN(	DE	11011110	12-11-0-6
BT	DA	11011010	12-11-0-2

BASIC SYMBOL (TEXT ATOM)	HEX CODE	ASCII CODE	HOLLERITH CODE (Rows Punched)
C1	B5	10110101	11-0-9-5
CLEAR	81	10000001	0-9-1
CO	B8	10111000	11-0-9-8
COM	A6	10100110	12-0-9-7
CONTINUE	84	10000100	0-9-4
CONVERT	AE	10101110	12-11-9-6
COS(	C3	11000011	12-0-8-1
DA	BD	10111101	12-11-0-9-3
DATA	97	10010111	12-9-8
DBACKSPACE	BB	10111011	12-11-0-9-1
DC	BF	10111111	12-11-0-9-5
DEFFN	CE	11001110	12-11-8-5
DIM	93	10010011	9-3
DISK	8E	10001110	12-9-8-2
DSKIP	89	10001001	0-9-8-1
END	96	10010110	9-6
EXP(	C4	11000100	12-0-8-2
FN	C0	11000000	12-11-0-9-6
FOR	9E	10011110	9-8-6
GOSUB	9A	10011010	9-8-2
GOTO	9C	10011100	12-9-4
HEX(	D2	11010010	11-0-8-2
IF	9F	10011111	11-0-9-1
INPUT	99	10011001	9-8-1
INT(	C5	11000101	12-0-8-3

BASIC SYMBOL (TEXT ATOM)	HEX CODE	ASCII CODE	HOLLERITH CODE (Rows Punched)
KEYIN	88	10001000	0-9-8
LEN(	D5	11010101	11-0-8-5
LET	91	10010001	9-1
LIMITS	86	10000110	12-9-6
LIST	80	10000000	11-0-9-8-1
LOAD	A1	10100001	12-0-9-2
LOG(	C6	11000110	12-0-8-4
MAT	A8	10101000	12-8-1
MOVE	AD	10101101	12-11-9-5
NEXT	9D	10011101	11-9-4
NUM(	DD	11011101	12-11-0-5
OFF	BA	10111010	12-11-0
ON	94	10010100	9-4
OPEN	B4	10110100	11-0-9-4
OR	8B	10001011	0-9-8-3
PLOT	A4	10100100	12-0-9-5
PLOT (SEL)	AF	10101111	12-11-9-7
POS(	DF	11011111	12-11-0-7
PRINT	A0	1010000	12-0-9-1
PRINTUSING	A7	10100111	12-0-9-8
RE	D6	11010110	11-0-8-6
READ	98	10011000	9-8
REM	A2	10100010	12-0-9-3
RENUMBER	83	10000011	0-9-3
RESTORE	A3	10100011	12-0-9-4

BASIC SYMBOL (TEXT ATOM)	HEX CODE	ASCII CODE	HOLLERITH CODE (Rows Punched)
RETURN	9B	10011011	9-8-3
REWIND	A9	10101001	12-11-9-1
RND(	C9	11001001	12-0-8-7
RUN	82	10000010	0-9-2
SAVE	85	10000101	11-9-5
SCRATCH	AC	10101100	12-11-9-4
SELECT	A5	10100101	12-0-9-6
SGN(	C8	11001000	12-0-8-6
SIN(	C7	11000111	12-0-8-5
SKIP	AA	10101010	12-11-9-2
SQR(	C2	11000010	12-11-0-9-8
STEP	B0	10110000	12-11-9-8
STOP	95	10010101	9-5
STR(	D3	11010011	11-0-8-3
TAB(	CD	11001101	12-11-8-4
TAN(	CA	11001010	12-11-8-1
TAPE	8F	10001111	11-9-8-3
TEMP	8D	10001101	12-9-8-1
THEN	B1	10110001	11-0-1
TO	B2	10110010	11-0-9-2
TRACE	90	10010000	12-11-0-9-8-1
VAL(	DC	11011100	12-11-0-4
VERIFY	BC	10111100	12-11-0-9-2
MOR	8C	10001100	0-9-8-4

APPENDIX E
SYSTEM 2200 ERROR MESSAGES

CODE 01	TEXT OVERFLOW
CODE 02	TABLE OVERFLOW
CODE 03	MATH ERROR
CODE 04	MISSING LEFT PARENTHESIS
CODE 05	MISSING RIGHT PARENTHESIS
CODE 06	MISSING EQUALS SIGN
CODE 07	MISSING QUOTATION MARKS
CODE 08	UNDEFINED FN FUNCTION
CODE 09	ILLEGAL FN USAGE
CODE 10	INCOMPLETE STATEMENT
CODE 11	MISSING LINE NUMBER OR CONTINUE ILLEGAL
CODE 12	MISSING STATEMENT TEXT
CODE 13	MISSING OR ILLEGAL INTEGER
CODE 14	MISSING RELATION OPERATOR
CODE 15	MISSING EXPRESSION
CODE 16	MISSING SCALAR
CODE 17	MISSING ARRAY
CODE 18	ILLEGAL VALUE
CODE 19	MISSING NUMBER
CODE 20	ILLEGAL NUMBER FORMAT
CODE 21	MISSING LETTER OR DIGIT
CODE 22	UNDEFINED ARRAY VARIABLE
CODE 23	NO PROGRAM STATEMENTS
CODE 24	ILLEGAL IMMEDIATE MODE STATEMENT
CODE 25	ILLEGAL GOSUB/RETURN USAGE
CODE 26	ILLEGAL FOR/NEXT USAGE
CODE 27	INSUFFICIENT DATA
CODE 28	DATA REFERENCE BEYOND LIMITS
CODE 29	ILLEGAL DATA FORMAT
CODE 30	ILLEGAL COMMON ASSIGNMENT
CODE 31	ILLEGAL LINE NUMBER
CODE 33	MISSING HEX DIGIT
CODE 34	TAPE READ ERROR
CODE 35	MISSING COMMA OR SEMICOLON
CODE 36	ILLEGAL IMAGE STATEMENT
CODE 37	STATEMENT NOT IMAGE STATEMENT
CODE 38	ILLEGAL FLOATING POINT FORMAT
CODE 39	MISSING LITERAL STRING
CODE 40	MISSING ALPHANUMERIC VARIABLE
CODE 41	ILLEGAL STR(ARGUMENTS
CODE 42	FILE NAME TOO LONG
CODE 43	WRONG VARIABLE TYPE
CODE 44	PROGRAM PROTECTED
CODE 45	STATEMENT LINE TOO LONG
CODE 46	NEW STARTING STATEMENT NUMBER TOO LOW
CODE 47	ILLEGAL OR UNDEFINED DEVICE SPECIFICATION
CODE 48	UNDEFINED KEYBOARD FUNCTION
CODE 49	END OF TAPE
CODE 50	PROTECTED TAPE
CODE 51	ILLEGAL STATEMENT
CODE 52	EXPECTED DATA (NONHEADER) RECORD
CODE 53	ILLEGAL USE OF HEX FUNCTION
CODE 54	ILLEGAL PLOT ARGUMENT
CODE 55	ILLEGAL BT ARGUMENT
CODE 56	NUMBER EXCEEDS IMAGE FORMAT
CODE 57	ILLEGAL SECTOR ADDRESS
CODE 58	EXPECTED DATA RECORD
CODE 59	ILLEGAL ALPHA VARIABLE FOR SECTOR ADDRESS
CODE 60	ARRAY TOO SMALL
CODE 61	DISK HARDWARE ERROR
CODE 62	FILE FULL
CODE 63	MISSING ALPHA ARRAY DESIGNATOR
CODE 64	SECTOR NOT ON DISK
CODE 65	DISK HARDWARE MALFUNCTION
CODE 66	FORMAT KEY ENGAGED
CODE 67	DISK FORMAT ERROR
CODE 68	LRC ERROR
CODE 71	CANNOT FIND SECTOR
CODE 72	CYCLIC READ ERROR
CODE 73	ILLEGAL ALTERING OF A FILE
CODE 74	CATALOG END ERROR
CODE 75	COMMAND ONLY (NOT PROGRAMMABLE)
CODE 76	MISSING < OR > (PLOT ENCLOSURES)
CODE 77	STARTING SECTOR > ENDING SECTOR
CODE 78	FILE NOT SCRATCHED
CODE 79	FILE ALREADY CATALOGED
CODE 80	FILE NOT IN CATALOG
CODE 81	/XXX DEVICE SPECIFICATION ILLEGAL
CODE 82	NO END OF FILE
CODE 83	DISK HARDWARE FAILURE
CODE 84	NOT ENOUGH MEMORY FOR MOVE OR COPY
CODE 85	READ AFTER WRITE ERROR
CODE 86	FILE NOT OPEN
CODE 87	COMMON VARIABLE REQUIRED
CODE 88	LIBRARY INDEX FULL
CODE 89	MATRIX NOT SQUARE
CODE 90	MATRIX OPERANDS NOT COMPATIBLE
CODE 91	ILLEGAL MATRIX OPERAND
CODE 92	ILLEGAL REDIMENSIONING OF ARRAY
CODE 93	SINGULAR MATRIX
CODE 94	MISSING ASTERISK
CODE 95	ILLEGAL MICROCOMMAND OR FIELD/ DELIMITER SPECIFICATION
CODE 96	MISSING BUFFER
CODE 97	VARIABLE OR ARRAY TOO SMALL, OR INSUFFICIENT DATA IN BUFFER
CODE 98	ILLEGAL ARRAY MODIFIER ARGUMENTS

INDEX

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.

700-3330C

TITLE OF MANUAL: 2234A/2244A HOPPER FEED CARD READERS

COMMENTS:

Fold

Fold

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

WANG

Fold

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD

FIRST CLASS PERMIT NO. 16 LOWELL, MA

POSTAGE WILL BE PAID BY ADDRESSEE

WANG LABORATORIES, INC.
ONE INDUSTRIAL AVENUE
LOWELL, MASSACHUSETTS 01851

Attention: Technical Writing Department

Fold

Cut along dotted line.

Printed in U.S.A.
13-1019

International Representatives

American Samoa
Argentina
Bahrain
Bolivia
Botswana
Brazil
Canary Islands
Chile
Columbia
Costa Rica
Cyprus
Denmark
Dominican Republic
Ecuador
Egypt
El Salvador
Finland
Ghana
Greece
Guam
Guatemala
Haiti
Honduras
Iceland
India
Indonesia
Ireland
Israel
Italy
Ivory Coast
Jamaica
Japan
Jordan
Kenya
Korea
Kuwait
Lebanon
Liberia
Malaysia
Mexico
Morocco
Nigeria
Norway
Paraguay
Peru
Phillippines
Portugal
Qatar
Saudi Arabia
Senegal
South Africa
Spain
Sri Lanka
Sudan
Syria
Thailand
Turkey
United Arab Emirates
Uruguay
Venezuela
Yugoslavia

United States

Alabama
Birmingham
Mobile

Alaska
Anchorage

Arizona
Phoenix
Tucson

California
Culver City
Emeryville
Fountain Valley
Fresno
Inglewood
Sacramento
San Diego
San Francisco
Santa Clara
Ventura

Colorado
Englewood

Connecticut
New Haven
Stamford
Wethersfield

District of Columbia
Washington

Florida
Hialeah
Jacksonville
Orlando
Tampa

Georgia
Atlanta
Savannah

Hawaii
Honolulu

Idaho
Boise

Illinois
Chicago
Morton
Oak Brook
Park Ridge
Rock Island
Rosemont
Springfield

Indiana
Carmel
Indianapolis
South Bend

Iowa
Ankeny

Kansas
Overland Park
Wichita

Kentucky
Louisville

Louisiana
Baton Rouge
Metairie

Maryland
Rockville
Towson

Massachusetts
Boston
Burlington
N. Chelmsford
Lawrence
Littleton
Lowell
Tewksbury
Worcester

Michigan
Kalamazoo
Kentwood
Okemos
Southfield

Minnesota
Minneapolis

Missouri
Creve Coeur
St. Louis

Nebraska
Omaha

Nevada
Las Vegas

New Hampshire
Manchester

New Jersey
Bloomfield
Toms River

New Mexico
Albuquerque

New York
Albany
Fairport
Liverpool
New York City
Syosset
Tonawanda

North Carolina
Charlotte
Greensboro
Raleigh

Ohio
Akron
Cincinnati
Cleveland
Independence
Toledo
Worthington

Oklahoma
Oklahoma City
Tulsa

Oregon
Eugene
Portland

Pennsylvania
Allentown
Camp Hill
Erie
Philadelphia
Pittsburgh
State College
Wayne

Rhode Island
Providence

South Carolina
Charleston
Columbia

Tennessee
Chattanooga
Knoxville
Memphis
Nashville

Texas
Austin
Dallas
Houston
San Antonio

Utah
Salt Lake City

Vermont
Montpelier

Virginia
Newport News
Norfolk
Richmond

Washington
Richland
Seattle
Spokane

Wisconsin
Appleton
Brookfield
Green Bay
Madison
Wauwatosa

International Offices

Australia
Wang Computer Pty., Ltd.
Adelaide
Brisbane
Canberra
Milsons Point (Sydney)
South Melbourne
West Perth

Austria
Wang Gesellschaft, m.b.h.
Vienna

Belgium
Wang Europe, S.A.
Brussels
Erpe-Mere

Canada
Wang Laboratories (Canada) Ltd.
Burlington, Ontario
Burnaby, B.C.
Calgary, Alberta
Don Mills, Ontario
Edmonton, Alberta
Montreal, Quebec
Ottawa, Ontario
Toronto, Ontario
Victoria, B.C.
Winnipeg, Manitoba

France
Wang France, S.A.R.L.
Bagnolet, (Paris)
Discheim (Strassbourg)
Ecully (Lyon)
Nantes
Toulouse Cedex

Hong Kong
Wang Pacific Ltd.
Hong Kong

Japan
Wang Computer Ltd.
Tokyo

Netherlands
Wang Nederland B.V.
IJsselstein
Gronigen

New Zealand
Wang Computer Ltd.
Auckland
Wellington

Panama
Wang de Panama (CPEC) S.A.
Panama City

Puerto Rico
Wang Computadoras
San Juan

Singapore
Wang Computer (Pte) Ltd.
Singapore

Sweden
Wang Skandinaviska AB
Malmo
Stockholm (Solna)
Groteborg

Switzerland
Wang S.A./A.G.
Zurich
Bern
Geneva
Lausanne

Taiwan
Wang Industrial Co.
Taipei
Kaohsiong

United Kingdom
Wang (UK) Ltd.
Birmingham
London
Manchester
Richmond

West Germany
Wang Laboratories, GmbH
Frankfurt
Berlin
Dusseldorf
Essen
Freiburg
Hamburg
Hannover
Kassel
Koln
Munchen
Numberg
Saarbrucken
Stuttgart

WANG LABORATORIES, INC.

ONE INDUSTRIAL AVENUE, LOWELL, MASSACHUSETTS 01851, TEL. (617) 459-5000, TWX 710 343-6769, TELEX 94-7421

Printed in U.S.A.
700-3330C
11-80-3C
Price: see current list