2200
BASIC-2
Multiuser Operating System
Software Bulletin
Release 2.5

1st Edition — July 1983

700-8079

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

Disclaimer of Warranties and Limitation of Liabilities

The staff of Wang Laboratories, Inc., has taken due care in preparing this manual; however, nothing contained herein modifies or alters in any way the standard terms and conditions of the Wang purchase, lease, or license agreement by which this software package 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 the software package, the accompanying manual, or any related materials.

NOTICE:

All Wang Program Products are licensed to customers in accordance with the terms and conditions of the Wang Laboratories, Inc. Standard Program Products License; no ownership of Wang Software is transferred and any use beyond the terms of the aforesaid License, without the written authorization of Wang Laboratories, Inc., is prohibited.

WANG LABORATORIES, INC., ONE INDUSTRIAL AVENUE, LOWELL, MA 01851 • TEL: 617/459-5000, TWX 710-343-6769, TELEX 94-7421

PREFACE

The purpose of this Software Bulletin is to provide up-to-date documentation for software enhancements to the 2200 Series until the appropriate manuals are revised. Chapter 1 describes the changes and features of Release 2.5 of the 2200 BASIC-2 Multiuser Operating System. Chapter 2 summarizes changes in previous releases of the operating system. Chapter 3 describes the Initialize Date & Time Utility.

This documentation is intended to be used in conjunction with the following manuals:

- BASIC-2 Utilities Reference Manual (700-6855)
- Wang BASIC-2 Language Reference Manual (700-4080D)
- Wang BASIC-2 Disk Reference Manual (700-4081G)

CONTENTS

CONTENTS (continued)

FIGURES

CHAPTER 1
RELEASE 2.5

1.1 INTRODUCTION

Release 2.5 of the BASIC-2 Multiuser Operating System provides the following operating system enhancements:

- Support for the communications capabilities of the Model 2236MXE
- A new, more efficient disk index structure and access method
- A facility for tracing disk activity

1.2 RELEASE INFORMATION

Release 2.5 of the BASIC-2 Multiuser Operating System is available on the following media:

Part Number	Type	Comment
701-2294S	SSSD	2200 MVP, MVPC, LVP, LVPC (2270A Diskette Drive)
731-0058B	DSDD	2200 LVP, LVPC, SVP (DSDD Diskette Drive)

Hardware Requirements

Release 2.5 requires at least 24K of Control Memory.

1.3 CHANGES TO THE SYSTEM PLATTER

The system platter includes the MVP operating system and BASIC-2 language processor, the system diagnostics, and the system utilities. The following system files have been revised or deleted since Release 2.4:

@MVP - MVP Operating System and BASIC-2 Language Processor (Rel. 2.5)
@MXE0 - MXE Microcode
@MRTIAN - Martian Wars Game
@INSTALL - System Install Utility
@MOVEFIL - Move File Utility
@RECOVER - Recover from Backup Utility
@SYSMVPB - System Menu

- The MVP Operating System and BASIC-2 Language Processor file (@MVP) and the MXE Microcode file (@MXE0) have been updated to include the operating system enhancements.

- To accommodate the increased operating system code and the MXE asynchronous facility, the Martian Wars game (@MRTIAN) was deleted.

- The System File Installation utility and the System Menu file have been updated to reflect the deletion of the Martian Wars game.

- The Move File utility (@MOVEFIL) and the Recover from Backup utility (@RECOVER) have been updated to correct known problems with these utilities. Refer to Section 1.4 for more information.

1.4 CORRECTED ANOMALIES

Previous releases of the Move File utility (@MOVEFIL) destroyed files under the following conditions:

- The operator specified the same address for both source and destination platters.

- The operator did not specify a destination file name different from the source file name.

This utility now displays an error message if a user attempts to move a file to the disk on which it currently resides. Additionally, the @MOVEFIL utility now accepts lowercase characters for a disk address.

Previous releases of the Recover from Backup utility (@RECOVER) did not correctly recover 'all active files' if sectors in the catalogue index (i.e., @INDEX) contained duplicate file names. The utility has been modified to correct this problem; there has been no functional change to @RECOVER.

1.5 COMMAND MODE OF THE MODEL 2236MXE

When a 2200 system is configured with a Model 2236MXE or an SVP Option-W Terminal Processor, it is possible to perform certain system procedures by evoking the Model 2236MXE Command mode and entering MXE commands. These MXE Command mode procedures include:

- Setting the transmission rate of a port on the terminal processor
- Locking and unlocking the transmission rate of the current port
- Displaying the status of all ports on the terminal processor
- Changing the primary user (i.e., Port 1)

Running 2236MXE Command Mode

A user can run MXE commands at any time from any powered-on termimal currently connected to a port on the Model 2236MXE or an SVP Option-W. It is not necessary to configure the system before entering MXE Command mode. If the system is configured, the terminal from which the user issues the MXE command does not have to be in the current configuration.

Before configuration, the system is under the control of the Bootstrap. The Set Primary Port command can be run only when the system is under Bootstrap control or when the system is running with the VP operating system.

When the system is running under Bootstrap control or running the VP operating system, there is a special MXE password needed to execute most MXE commands. On power-up, this password defaults to MXEPSW. A user can change the password with the Change MXE Password command. When the system is running the MVP operating system, the MXE password is the same as the first six characters of the 2200 system configuration password.

Entering and Exiting 2236MXE Command Mode

Entering MXE Command mode alters the screen display. To enter MXE Command mode, press RESET, then press LOAD three times. After the third time, the prompt ENTER MXE COMMAND appears on the screen, followed by a percent character (%) on the next line. When the percent character appears, the user can enter the MXE command. Every MXE command begins with a one-character MXE command code.

After processing the command, the terminal again displays the percent character and the user can enter another MXE command. Pressing RETURN when the percent prompt appears, or pressing RESET exits MXE Command mode. Exiting MXE Command mode returns to the processing point from which MXE Command mode was entered.

NOTE

Although entering MXE Command mode halts processing on the current terminal, other terminals on the terminal processor operate normally. It is possible for more than one terminal to enter MXE Command mode simultaneously. Therefore, use caution when entering MXE commands that affect the performance of other terminals, such as Set Transmission Rate.

Status Command

General Form:

?

The Status command returns a chart indicating the status of all ports on the terminal processor to which the terminal issuing the command is connected. The first line of the chart uses the following format:

device R# (mode) option

where:

device = the model number of the terminal processor.

R# = the revision of terminal processor firmware. This number is the number of the general operating system release with the terminal processor release number appended to it. For example, Revision 2.51 or 2.53 of the terminal processor code runs with Operating System 2.5, while terminal processor Revision 2.92 runs with Operating System 2.9. Problems may occur if the number of the terminal processor firmware does not correspond to the operating system release number.

mode = the indication that the system has been configured with an MVP operating system, in which case the terminal displays MVP within the parentheses. If the system has not yet been configured or if the system has been configured with a VP operating system, the terminal displays BOOTSTRAP within the parentheses.

option = the indication of any additional options implemented by the MXE code.

The next four lines of the chart display the following information for each port on the terminal processor.

On/Off state -- If the terminal connected to the port is powered on (i.e., Data Set Ready on the RS-232 connection is active), the word ON appears on the screen. Otherwise, the word OFF appears.

Software selected transmission rate -- This is the transmission rate selected by MXE Command Code B, Set Transmission Rate. If MXE Command Code B has not been evoked, the transmission rate defaults to the rate set by the hardware switches.

Locked or unlocked state -- The Letter L indicates that MXE Command Code L, Lock, has been used to lock the transmission rate of the terminal. If the terminal is not locked, this field is left blank.

Hardware selected transmission rate -- This is the transmission rate set by the switches on the processor board.

Local/Remote state -- This field reflects the state of the Carrier Detect signal on the RS-232 line, indicating whether the terminal is a local terminal or is connected to the MXE through a modem. A remote terminal causes an active Carrier Detect signal.

Terminal type -- This field describes the format of the current communication protocol of the port. If the terminal on the indicated port is a standard Wang 2200 terminal or Wang 2200 terminal emulation (following Wang terminal protocol), the message WANG TERMINAL appears in this field. If the port is configured as a general asynchronous communications port, the message ASYNCH appears in this field.

An asterisk appears before the port to which the terminal issuing the status command is connected.

The following is an example of the chart the Status command returns:

```
2236 MXE R2.51  (BOOTSTRAP)
*PORT 1 ON  19200 L / (19200)  LOCAL   WANG TERMINAL
 PORT 2 OFF 19200   / (19200)  LOCAL   WANG TERMINAL
 PORT 3 OFF 19200   / (19200)  LOCAL   WANG TERMINAL
 PORT 4 OFF 19200   / (19200)  LOCAL   WANG TERMINAL
```

In this example, the terminal processor is a Model 2236MXE; the revision number is 2.51, indicating that this is the first bootstrap microcode revision that corresponds with Operating System Release 2.5; and the system has not yet been configured or has been configured with a VP operating system. The port issuing the status command is Port 1, which is the only port with a powered-on terminal. Port 1 on this 2236MXE terminal processor can be Port 1, Port 5, Port 9, or Port 13 on the system, depending on the number of terminal processors connected and the address of this MXE. The transmission rate at Port 1 is locked. All four ports are set to transmit at a rate of 19,200 bps and all terminals are local Wang terminals.

Change MXE Password Command

General Form:

C password1 password2

Where:

password1 = the current MXE password, consisting of six characters. The password does not appear on the screen; instead, the string "012345" appears.

password2 = the new MXE password.

Command Code C changes the MXE password used when the system is operating under control of the Bootstrap or with a VP operating system.

When the system is operating with an MVP operating system, the 2200 system reconfiguration password is required to execute MXE commands. The Change MXE Password command has no effect on the system reconfiguration password. The 2200 system password is eight bytes long. The MXE password is six bytes long. Therefore, upon configuration, the MXE password is set to the first six bytes of the system password.

After the system processes this command, the CRT displays the message OK.

Examples of valid syntax:

```
C OLPSWD NWPSWD
C secret Psword
```

Set Primary Port Command

General Form:

A password x

Where:

password = the current MXE password. The password does not appear on the screen; instead, the string "012345" appears.

x = the port designator, a number between 0 and 4 indicating which port to establish as the primary port of the 2236MXE. A port designator of 1, 2, 3, or 4 corresponds to the port number on the terminal processor. A port designator of 0 indicates the current terminal port.

Command Code A sets the primary port. The primary port supports the terminal that can perform master initialization. On a system with more than one terminal processor, the user can issue this command only from a terminal connected to the first terminal processor.

Additionally, the user can issue this command only when the system is under control of the Bootstrap. Unless a Change MXE Password command has been issued, the password is the default password, MXEPSW.

When this command is issued, Port 1 is logically swapped with the port indicated in the command. The MVP operating system will subsequently consider the swapped port to be Port 1 and Port 1 to be the swapped port. This command is used when Port 1 is not functioning or is not available.

After the system processes the Set Primary Port command, the CRT displays the message OK.

Examples of valid syntax:

```
A SYSTEM 3
A SECRET 0
A PSWORD 4
```

Note that the password itself does not appear on the screen.

Set Transmission Rate Command

General Form:

B password x rate

Where:

password = the current MXE password. The password does not appear on the screen; instead, the string "012345" appears.

x = the port designator, a number between 0 and 4 indicating the port to which the transmission rate is set. A port designator of 1, 2, 3, or 4 corresponds to the port number on the terminal processor. A port designator of 0 indicates the current terminal port.

rate = the transmission rate to set for the port indicated by the port designator. This command can set the transmission rate to 50, 75, 100, 110, 134.5, 150, 200, 300, 600, 1200, 2400, 4800, 9600, or 19200.

Command Code B sets the transmission rate of a particular port. Setting the transmission rate through software overrides any hardware switch settings. Master initializing or powering on the system resets the transmission rate to the hardware setting.

Command Code B can set the transmission rate of only those ports on the same terminal processor as the terminal issuing the command. The port must currently be a terminal port (i.e., the command will not work on a TC or other special type of port).

After a user enters this command, the terminal displays the 2236MXE Status chart and prompts the user to verify the entered information. The user enters a Y to verify or an N to cancel.

Examples of valid syntax:

```
B SYSTEM 3 9600
B secret 0 300
B Psword 4 50
```

Note that the password itself does not appear on the screen.

Lock Command

General Form:

L

Command Code L changes the lock state of the port issuing the command. When a port is locked, no port can change its transmission rate, including the locked port itself. The default condition, set when the system is powered on, is unlocked.

When this statement is issued from a port that is unlocked, the port changes to a locked state and the CRT displays the statement: BAUDRATE LOCKED. When this statement is issued from a port that is locked, the port changes to an unlocked state and the CRT displays the statement: BAUDRATE UNLOCKED.

1.6 COMMUNICATIONS CAPABILITIES OF THE 2236MXE

Release 2.5 provides support for the asynchronous communications capabilities of the 2236MXE. With the new parameters of the SELECT statement, the user can select any ports on the system (except Port 1) as telecommunications ports.

Asynchronous communications under the control of a user BASIC program or Wang ASYNC1 package is supported. 2227B asynchronous functionality is provided, with the following exceptions:

- Reverse channel is not supported.
- PRINT statements cannot be used for TC output; $GIO must be used.

A number of BASIC statements are used for controlling asynchronous communications. SELECT TC selects a port as a telecommunications port; SELECT TERMINAL converts a TC port to a terminal port. $OPEN and $CLOSE control access to a TC port. $GIO is used to issue TC commands, send output, and receive input. Refer to Section 1.8 for a description of each command.

1.7 NEW DISK INDEX

A new, more efficient disk index structure is available with Release 2.5. The new index method improves the performance of locating files within the index and is particularly effective on platters containing many files. The improved performance is reflected in the LOAD, SAVE, LIMITS, DATALOAD OPEN, and DATASAVE OPEN statements.

2200 Disk Index Structure

With Release 2.5, the 2200 operating system supports two distinct index structures. A platter has either the new index structure or the old index structure; however, both types can exist on the same system. The old index structure, created with the SCRATCH DISK statement, is fully supported with no change. A SCRATCH DISK ' statement creates the new index structure.

The first byte of sector 0 indicates which index structure exists on the platter. If Byte 1 of sector 0 equals hex 00, then the old index structure (i.e., SCRATCH DISK) exists on the platter. If Byte 1 of sector 0 equals hex 01, then the new index structure (i.e., SCRATCH DISK ') exists on the platter.

File entries in the new index structure are identical to those in the old index structure.

Access to the New Disk Index

The new index structure uses a more efficient hashing algorithm for locating files in the index, resulting in a more even distribution of file entries. Additionally, if an index sector is full, the system enters a new file into the next higher sector rather than the next lower, as is done with the old index. Entering the files into the next higher sector results in an improved performance on 2200 disks that provide a look-ahead read (e.g., the Winchester-style fixed drives of the 2200LVP and the Model 2280 Disk Drive).

The following BASIC-2 program simulates the new disk index hashing:

```
100 N$ = F$: REM F$ contains the name of the file to be entered
110 X$ = HEX(00)
120 FOR I = 1 TO 8
130  X1$ = STR(N$,I,1): REM X1$ contains the Ith byte of the filename
140  IF MOD(I,2) = 0 then 160
150  ROTATE (X1$,4):    REM Exchange upper/lower nibbles of odd bytes
160  X$ = X$ ADD X1$:   REM X$ contains the sum of all bytes
170 NEXT I
180 S = MOD(VAL(X$),C): REM C contains the number of index sectors
190 REM S contains the sector number into which the entry should go
```

If sector S is full, then S is set to MOD(S+1, C) and the entry goes in the new sector S. The system repeats this process until the entry is placed or until S equals its original value, thus indicating that the disk catalog is full.

WARNING

The new disk index structure is not appropriate for all users.

Certain BASIC-2 programs make direct use of knowledge of the old index method, accessing the index with DATALOAD BA and DATASAVE BA. These programs will not operate properly on platters with the new disk index since the hashing algorithm has changed. Careful analysis should be done before converting to the new index since there is a risk that some software will not operate properly after the conversion.

It is not a recommended practice to directly access the disk index with DATALOAD BA and DATASAVE BA. However, if such a program is written, it should support both the old and new index structures since it may not be possible or wise to convert all old platters to the new format.

Wang Laboratories, Inc. will update Integrated Support Systems (ISS) to support both the new and old index structures. However, Wang will not update several other packages (e.g., STAT). Therefore, such packages will only operate with the old index.

Conversion to the New Index

To convert an existing platter to the new index structure, perform the following:

- Establish a new disk index on a blank platter by executing a SCRATCH DISK' statement.
- Move files from the existing platter to the new platter with a MOVE file statement.

1.8 GENERAL FORMS

In summary, Release 2.5 includes the following new statements to the BASIC-2 language: SCRATCH DISK ', SELECT TC, SELECT TERMINAL, and TRACE DISK. The general forms for these statements are presented on the following pages.

SCRATCH DISK '

General Form:

SCRATCH DISK ' platter [file # / disk-address,] [LS = exp-1,] END = exp-2

Where:

' = an indication that the new disk index structure is used. The absence of the single quote indicates that the old disk index structure is used.

LS = a parameter specifying the number of sectors to be set aside for the Catalog Index.

exp-1 = an integer or expression whose value is from 1 to 255. If the LS and expression-1 parameters are not included, the system automatically sets the size of the Catalog Index to 24 sectors.

END = a parameter specifying the last (highest) sector address in the Catalog Area.

exp-2 = an expression whose value must be less than or equal to the last (highest) sector address on the disk.

The SCRATCH DISK ' statement creates a disk index with the new format. The new disk index provides faster access to files by providing a better distribution of file entries in the catalog than is provided with indexes created by the SCRATCH DISK statement.

Examples of valid syntax:

```
SCRATCH DISK ' T/D25, LS = 50, END = 52607
SCRATCH DISK ' T/D10, END = 3873
```

General Form:

TRACE DISK [OFF]

The TRACE DISK statement enables the disk tracing facility. The TRACE DISK OFF statement disables the disk tracing facility. If disk tracing is not enabled, the system ignores the TRACE DISK OFF statement.

TRACE DISK lists all disk activity performed by the system for the current partition. The system sends the trace output to the currently selected CO device (refer to the SELECT statement in the Wang BASIC-2 Language Reference Manual.) The trace remains in effect until a TRACE DISK OFF statement is executed or RESET is pressed.

The TRACE DISK statement is useful in analyzing the disk activity of application packages.

The trace disk output has one of the following forms:

/taa hh ddddd pp or /taa 20hh ddddd pp

where:

/taa = the address of the disk drive being accessed.

hh or 20hh = a disk command, where:

00 = Read sector
40 = Write sector
80 = Compare sector (i.e., read-after-write)
20hh = Special command to Disk Processing Unit (DPU)

Special commands to the DPU include service and diagnostics operations, as well as, the following operations:

2001 - Copy sectors
2002 - Format platter
2010 - Start of multisector write
2011 - End of multisector write
2012 - Verify sectors

Not all 2200 disk units use 20hh commands.

ddddd = the sector being accessed. The sector number is in decimal starting from 0.

pp = the partition issuing the disk command.

Example:

```
TRACE DISK

/D10 00   00000 01
/D10 00   00011 03
/D10 00   00012 03
/D12 40   00123 11
/D11 2004 01234 10
```

Examples of valid syntax:

```
TRACE DISK
TRACE DISK OFF
```

SELECT TC

General Form:

SELECT TC port-number

Where:

port-number = { /Add, where dd is a decimal value from 02 to 16. | <alpha-variable>, where alpha-variable contains the port-address Add. }

The SELECT TC statement selects the specified port as a telecommunications (TC) port instead of a terminal port. No partition, including the partition issuing the SELECT TC statement, should be attached or assigned to the specified port when the SELECT TC statement is executed. If a partition is assigned or attached to the port, the system generates an illegal device error (P48). Therefore, a programmer should release all partitions assigned to the requested port by issuing $RELEASE PART statements before issuing a SELECT TC statement or configure the system with no partitions assigned to the TC port.

The device address /Add or <alpha-variable> indicates which port to select. Port 01 cannot be selected as a TC port since Port 01 serves as the terminal port for the system console. If the specified port is selected as a TC port already, the system does not generate an error.

To avoid losing control of a port while selecting it as a TC port, perform the following. This example assumes that A$ is set to the device-address.

```
$BREAK
SELECT #1 <A$>
$RELEASE PART
SELECT TC <A$>
$OPEN #1
```

The port designated by A$ should be attached to the partition initially.

Examples of valid syntax:

```
SELECT TC /A03
SELECT TC /A12
SELECT TC <A$>
```

SELECT TERMINAL

General Form:

```
SELECT TERMINAL port-number
```

Where:

port-number = { /Add, where dd is a decimal value from 02 to 16. | <alpha-variable>, where alpha-variable contains the port-address Add. }

The SELECT TERMINAL statement selects a port as a terminal port or converts a TC port to a terminal port. The specified port should not be opened (by means of a $OPEN statement) as a TC port when executing a SELECT TERMINAL statement. If the specified port has been opened as a TC port by another partition, the system generates an illegal device error. When the system is powered on, the system selects all ports as terminal ports.

Examples of valid syntax:

```
SELECT TERMINAL /A04
SELECT TERMINAL <B$>
```

General Form:

$OPEN [line-number,] { device-address / file-number / port-number } [, { device-address / file-number / port-number }] ...

Where:

device-address = /taa, where t is the device-type and aa is the physical device-address.

file-number = #n, where n is integer or numeric variable with a value 0 to 15, inclusive.

port-number = /Add, where dd is decimal value from 02 to 16.

The $OPEN statement requests exclusive use of a device (i.e., peripherals) or TC port for the current partition. To request exclusive use of a device, the programmer specifies either the device-address or the file-number in the $OPEN statement. To request exclusive use of a TC port, the programmer specifies the port-number in the $OPEN statement.

The system signals an ERR P48 - Illegal Device Specification if one of the following conditions occurs:

- The specified device is not in the Master Device Table.
- The specified device is already opened for exclusive use by another partition.
- The specified port was not selected as a TC port before executing the $OPEN statement.

Once open, the current partition maintains exclusive control of the device or port until one of the following conditions occurs:

- Execution of a $CLOSE statement
- Execution of a program END statement
- Execution of a CLEAR or LOAD RUN command
- Execution of a SELECT TERMINAL statement
- Pressing of the RESET key

If another partition has already opened the specified device or TC port, the $OPEN statement branches to the specified line-number. If no line-number is specified, the $OPEN statement waits until the specified device or TC port becomes available for this partition.

If the programmer specifies multiple devices (including TC ports) in a $OPEN statement and the system cannot open one of the devices, then $OPEN does not open any of the devices.

Examples of valid syntax:

```
$OPEN /A13, /215, /A07
$OPEN /A02, #4
```

General Form:

```
$CLOSE [line-number,] { device-address } [, { device-address } ] ...
                      { file-number   }      { file-number   }
                      { port-number   }      { port-number   }
```

Where:

device-address = /taa, where t is the device-type and aa is the physical device-address.

file-number = #n, where n is integer or numeric variable with a value 0 to 15, inclusive.

port-number = /Add, where dd is decimal value from 02 to 16.

The $CLOSE statement releases the specified devices and TC ports that are currently restricted by the $OPEN statement. If no device-address or port-number is specified, the $CLOSE statement releases all devices and TC ports opened for the current partition.

The specified port must be selected as a TC port previously, otherwise the system generates an illegal device error. If the TC port has not been opened previously, the system does not generate an error.

Once open, the current partition maintains exclusive control of the port until one of the following conditions occurs:

- Execution of a $CLOSE statement
- Execution of a program END statement
- Execution of a CLEAR or LOAD RUN command
- Execution of a SELECT TERMINAL statement
- Pressing of the RESET key

Examples of valid syntax:

```
$CLOSE
$CLOSE /A15, #4
```

General Form:

```
$GIO [comment] [file-number,]  (arg-1 [,arg-2])[arg-3 [,arg-3 ...]]
               [port-number,]
```

Where:

file-number = #n, where n is integer or numeric variable with a value 0 to 15, inclusive.

port-number = /Add, where dd is decimal value from 02 to 16.

The specified port must be selected as a TC port previously, otherwise the system generates an illegal device error. If the specified port is opened by another partition, the $GIO statement waits until the specified port is available.

During $GIO through TC port, the device address change commands, 71hh and 73r0, are not allowed.

For a description of the $GIO commands for asynchronous communication, refer to the Asynchronous Communications User Guide for Model 2236MXE Terminal Processor and Option-W Terminal Processor (700-8098).

CHAPTER 2
RELEASE 2.4, RELEASE 2.3, AND RELEASE 2.2

2.1 INTRODUCTION

This chapter summarizes enhancements made to 2200 BASIC in previous releases that are not yet incorporated into 2200 system manuals.

2.2 RELEASE 2.4

Release 2.4 of the BASIC-2 Multiuser Operating System was issued primarily to provide support for the Model 2236MXE Terminal Processor and SVP Option-W. Additionally, Release 2.4 included a system date and time, a connect/disconnect detection facility, and a screen input facility.

MXE Support

Release 2.4 (or later) of the BASIC-2 Operating System is required for BASIC-2 systems with a Model 2236MXE Terminal Processor or SVP Option-W.

Disk Write

If a Data Error (Error 96) occurs during a disk write operation, the system will retry the operation up to three times. This is expected to reduce error rates for disk operations and should be particularly effective for multiplexed 2280 disk units.

DATE and TIME

The Model 2236MXE Terminal Processor, SVP Option-W, and the current version of the Model 22C32 Triple Controller include a time-of-day clock that is accessible through and maintained by BASIC-2.

Release 2.4 introduced the following BASIC-2 language enhancements to support these added features:

- The DATE statement sets or changes the system date; the DATE function returns the value of the current date.
- The TIME statement sets or changes the system time; the TIME function returns the value of the current date.

Refer to Section 2.5 for a description of the DATE and TIME functions and statements.

To set or adjust the time and date, Release 2.4 provided an Initialize Date & Time utility. This utility resides on the system platter; it is discussed in Chapter 3.

To support the DATE and TIME statements, Release 2.4 included the following new error message:

ERR X79 - Invalid Password

If an incorrect password is specified in either the DATE or TIME statement, the system returns an ERR X79. This error is recoverable (i.e., a programmer can intercept and respond to the error under program control). Refer to Chapter 9 of the BASIC-2 Language Reference Manual for more information on error control features.

Screen Input

Release 2.4, when used with a Model 2336DW or 2336DE Terminal and a Model 2236MXE Terminal Processor or SVP Option-W, allows the image displayed on the screen to be transmitted to a BASIC program for further processing. BASIC receives the screen data with an INPUT SCREEN statement.

The 2200 screen input facility enables a BASIC program to receive an image of a terminal screen display. This program would typically print the image on a system printer or save the image in a disk file for processing. A screen input can be requested by a BASIC program or by an operator. In the first case, a BASIC program executes an INPUT SCREEN statement that reads the screen of the terminal attached to the current partition. In the second case, the operator holds down SHIFT and EDIT, thereby attaching the terminal to a partition that has previously been made available for screen input. Refer to Section 2.5 for a description of the INPUT SCREEN statement and the procedure for operator-initiated screen inputs.

The screen input facility requires the following hardware and software:

- MVP BASIC-2 Release 2.4 (or later)
- 2200MVP, 2200LVP, or 2200SVP Central Processing Unit
- Model 2336DW or Model 2336DE Terminal
- Model 2236MXE Terminal Processor (for MVP and LVP systems)
- Option-W (for SVP systems)

Terminal Disconnect

The 2236MXE and SVP Option-W can alert the 2200 CPU when a local or remote terminal is connected or disconnected. The terminal connect/disconnect detection facility enables 2200 BASIC programs to be written to monitor system users. With this facility, the operating system can detect whether a terminal is connected or disconnected. Upon disconnection, the system can initiate bookkeeping procedures, force a terminal to disconnect automatically after a specified period of time, or initiate Logon and Logoff programs. The $DISCONNECT statement enables or disables disconnect detection. Refer to Section 2.5 for a description of the $DISCONNECT statement.

The following list summarizes the features of the terminal connect/disconnect detection facility:

- Connect detection for partition allocation and initiation of a user Logon program.
- Forced disconnection if the user does not log on within the program specified time. Thus, a user is prevented from tying up a port without completing the Logon procedure.
- Program control of the connect/disconnect facility through BASIC.
- Disconnect detection for initiation of user Logoff programs.
- BASIC TIME and DATE functions for logging system use.

The connect/disconnect facility requires the following hardware and software:

- MVP BASIC-2 Release 2.4 (or later)
- 2200MVP, 2200LVP, or 2200SVP Central Processing Unit
- Model 2236MXE Terminal Processor (for MVP and LVP systems)
- Option-W (for SVP systems)

NOTE

VP BASIC-2 will not be updated to support the Model 2236MXE and the SVP Option-W. Therefore, the DATE, TIME, INPUT SCREEN, and $DISCONNECT statements will generate syntax errors.

2.3 RELEASE 2.3

Release 2.3 of the BASIC-2 Multiuser Operating System was issued primarily to provide support for the additional memory banks available on the 2200MVPC and 2200LVPC. However, this operating system can also run on the 2200MVP, 2200LVP, and 2200SVP. Release 2.3 included the following features and enhancements:

- Revised operating system to support up to 512K bytes of user memory, provided by the 2200MVPC and 2200LVPC. (The maximum number of partitions remains 16; the maximum number of terminals remains 13.)
- Revised Partition Generator (@GENPART) and Partition Status (@PSTAT) System Utilities to support 512K bytes of user memory.
- Revised system diagnostics for testing control memory and user memory.

2.4 RELEASE 2.2

Release 2.2 of the BASIC-2 Multiuser Operating System was issued to provide enhancements to the System Utilities, as well as provide several new features. Refer to the BASIC-2 Utilities Reference Manual (700-6855) for further information about each utility.

Additionally, Release 2.2 introduced the #ID function and the $ALERT statement. The numeric function #ID returns the CPU identification number. With the $ALERT statement, partitions can interrupt each other's execution.

2.5 GENERAL FORMS

The general forms for the BASIC-2 functions and statements that accompanied the last three releases are presented on the following pages. Release 2.2 introduced the $ALERT statement and the #ID function. Release 2.4 introduced the DATE and TIME statements and functions, and the $DISCONNECT and INPUT SCREEN statements.

.ERT

General Form:

$ALERT partition

Where:

partition = a numeric-expression specifying a partition number.

The $ALERT statement generates an interrupt to the specified partition. In order for the interrupt to have any effect, the alerted partition must execute a SELECT ON ALERT GOSUB statement. The SELECT ON ALERT GOSUB statement defines that alert interrupts are to be fielded and indicates a subroutine to execute when an alert interrupt occurs. (Refer to SELECT ON in Chapter 8 of the BASIC-2 Language Reference Manual.)

When an alert interrupt is acknowledged, the programmer knows that at least one $ALERT statement has been executed by some partition since the last occurrence of a $ALERT interrupt or a LOAD, CLEAR, or RUN command. The programmer does not know which partition executed the $ALERT, or whether or not several $ALERTs have been executed since the last $ALERT interrupt was acknowledged.

Example:

```
500 $ALERT 5: REM ALERT Partition 5
```

If Partition 5 has enabled alert interrupts, the alert interrupt is fielded as soon as Partition 5 comes to the end of processing a BASIC statement that is not in the interrupt handling subroutine.

Examples of valid syntax:

```
$ALERT 5
$ALERT T(N)
```

DATE

General Form (as a statement):

DATE = { alpha-variable / literal-string } PASSWORD { alpha-variable / literal-string }

General Form (as a function):

DATE

DATE Statement

The DATE statement sets or changes the system date. The programmer can specify the new ASCII date in an alpha-variable or a literal-string. The date is specified as a six character alphanumeric value in the form YYMMDD (year, month, day).

The alpha-variable or literal-string following the keyword PASSWORD represents the system password. If the given password is correct (i.e., matches @GENPART password), the system date is updated. Otherwise, the system returns a recoverable error (ERR X79 - Invalid Password).

The system date can be set even if there is no system clock. This capability allows a clockless system to maintain the system date manually.

Examples of valid syntax:

```
DATE = "820801" PASSWORD "SYSTEM"
DATE = D$ PASSWORD P$
```

DATE Function

DATE is an alphanumeric function that returns a six character ASCII string containing the current date in the form YYMMDD (year, month, day). DATE is used as an operand in alphanumeric expressions.

Examples of valid syntax:

```
T$ = DATE & TIME
A$ = DATE
```

$DISCONNECT

General Form:

$$\$DISCONNECT \left\{ \begin{array}{l} ON \ [expression] \\ OFF \end{array} \right\}$$

Where:

$0 \leq$ value of expression ≤ 65534

The $DISCONNECT statement enables or disables terminal disconnection detection. Once enabled, the operating system can detect when a terminal is disconnected. A local terminal is considered to be disconnected when the terminal is powered off. A remote terminal is disconnected when telephone communication is terminated. The operating system cannot distinguish between disconnections caused by timeout, modem disconnection, or terminal power off.

A $DISCONNECT ON statement enables disconnect detection. A $DISCONNECT OFF statement disables disconnect detection; this is the default state of the system. Issuing a subsequent $DISCONNECT statement always overrides the previous command and sets a new state.

The optional expression in the $DISCONNECT ON statement represents the time in seconds after which the operating system forces a disconnection. If specified, the operating system decreases the value of the expression by one each second and forces a disconnection of the terminal when the value of the expression equals zero. The counter remains at zero until another $DISCONNECT ON statement sets the time expression. Therefore, even after the terminal is disconnected, the next terminal at the same port is not disconnected until the execution of another $DISCONNECT ON statement.

$DISCONNECT is a command to the port attached to the partition issuing the statement and not to the partition. Therefore, changing partitions or terminals does not affect the way $DISCONNECT affects a specific port (i.e., $DISCONNECT remains in effect after the execution of a $RELEASE PART or $RELEASE TERMINAL statement).

Terminal Connect Detection

A local terminal is considered to be connected when it is powered on. A remote terminal is considered to be connected when the phone line link between the terminal and the terminal processor is established.

When the operating system detects a terminal connection or a RESET from a terminal that does not control any partitions, it automatically assigns an available partition to that terminal. All partitions assigned to the null terminal (i.e., terminal 0) and waiting for a terminal (i.e., having accessed a statement that performs I/O to the terminal, such as PRINT or LINPUT) are available partitions. If the terminal already controls one or more partitions, or if no available partitions exist, the operating system performs no action.

Once a partition is assigned to a terminal, this partition can then execute a Logon program that issues a $DISCONNECT ON statement and sets a time limit for the user to complete the Logon sequence. (Refer to Example 1, "Sample Logon Program".) If the user does not complete the logon sequence within the program specified time, the operating system automatically disconnects the user. Since a user can initiate but cannot complete the logon procedure, forced disconnection prevents the user from tying up a terminal port. If, however, the user properly completes the logon procedure, the forced disconnection can be overridden by executing a $DISCONNECT ON statement with no specified disconnect time.

Example 1: Sample Logon Program

```
0010 REM @CONNECT -- Sample Logon Program
0020 REM To log system users, run this program (or similar
   : REM program) in all released partitions.  When a terminal is
   : REM connected, the operating system assigns a partition to the
   : REM terminal and begins execution of this program.
0025 REM For protection, the partition should be nonprogrammable.
0030 REM%

Disconnect terminal if not logged on within 2 minutes

   : $DISCONNECT ON 120
0040 REM%

Title

   : PRINT HEX(03); AT(0,30); "LOG ON"
0050 REM%

Obtain user's password

   : PRINT AT(10,20,8); "Enter password: ";
   : FOR I = 1 TO 8
   : KEYIN STR(P$,I,1)
   : PRINT HEX(8B);
   : NEXT I
0060 REM%

Verify user's password and log on user

   : REM Perform any required logon procedures

0070 REM%

Turn off the disconnect timeout

   : $DISCONNECT ON
0080 REM%

Load menu

   : LOAD RUN T "START"
   : REM START should check that user has logged on
```

Terminal Disconnect Detection

The operating system is informed of disconnections only if disconnect is enabled. Upon detection of a disconnect, the operating system forces all partitions assigned to that terminal to run a user-written BASIC program called @DISCNCT. (Refer to Example 2, "Sample Logoff Program".) The process is functionally equivalent to typing RESET followed by LOAD RUN "@DISCNCT". However, running the program occurs independently of the terminal.

The @DISCNCT program can perform any required Logoff or accounting procedures concerning system use. Typically, this program makes the partition available to other users by executing a $RELEASE PART statement. Thereafter, the released partition can execute a Logon program that interacts with the user assigned this partition when terminal connections are detected.

If disconnect detection is not enabled for the terminal port, the operating system does not perform any action following the disconnection.

Example 2: Sample Logoff Program

```
0010 REM%

@DISCNCT -- Sample Logoff Program

0020 REM If terminal disconnect detection is enabled and a terminal
   : REM disconnects, the operating system automatically runs the
   : REM @DISCNCT program in all partitions assigned to that terminal
0030 REM%

Release the partition

   : $RELEASE PART
0040 REM%

Log off the user

   : REM Perform any required Logoff procedures
0050 REM%

Load a Logon program

   : LOAD T "@CONNECT"
```

Examples of valid syntax:

```
$DISCONNECT ON
$DISCONNECT ON 60
$DISCONNECT OFF
```

#ID Function

General Form:

```
#ID
```

The #ID function returns the value of the CPU identification number. The value is a number from 1 to 65535. With the #ID function, a program can distinguish one CPU from another. This capability is useful in licensing software to specific installations.

INPUT SCREEN

General Form:

```
INPUT SCREEN alpha-variable
```

The INPUT SCREEN statement reads the screen of the terminal attached to the current partition and stores an image of the screen in the alpha-variable. Each character on the screen and its associated display attributes are represented in the screen image.

INPUT SCREEN can only be used with controllers (e.g., Model 2236MXE or SVP Option-W) and terminals (e.g., Model 2336DW or 2336DE) that support the screen input facility. Attempting to execute an INPUT SCREEN statement with other terminals or controllers results in an error.

During a screen input, the terminal sends a total of 4080 bytes (characters) to the alpha-variable. Therefore, the alpha-variable should be at least 4080 characters in length in order to receive a full screen. The screen image consists of the following items:

- Terminal self-identification message
- Current cursor position
- Characters currently displayed
- Display attributes for each character

The first 78 bytes contain the self-identification message. The message identifies the type of terminal and its character set. The next 2 bytes identify the cursor location. The first byte identifies the row position; the second byte identifies the column position. Rows are numbered 0 to 24; columns are numbered 0 to 79.

The following 2000 bytes (25 rows by 80 columns) represent the characters currently displayed. The terminal uses the character codes of the Alternate Character Set (refer to Figure 2-1). Codes are sent row by row, starting at the first character in the first row. The 25th row of characters is all zeroes.

Each character has an associated attribute byte describing how the character is displayed. The 2000 bytes representing the characters are followed by 2000 bytes representing the display attributes. The 25th row of display attributes is only used for the box graphics under the characters of the 24th row. A display attribute byte has the following format:

```
bit 80 = 1 if character graphic
bit 40 = 1 if reverse video
bit 20 = 1 if blink
bit 10 = 1 if high intensity
bit 08 = 1 if underline
bit 04 = 1 if left horizontal box graphic segment
bit 02 = 1 if right horizontal box graphic segment
bit 01 = 1 if vertical box graphic segment
```

Only a foreground partition (i.e., a partition with a terminal attached) can execute an INPUT SCREEN statement. If a background partition with a terminal assigned to it issues INPUT SCREEN, execution is suspended until the terminal is attached to this partition.

Operator-Initiated Screen Input

Executing INPUT SCREEN in a partition with no terminal assigned to it (i.e., a $RELEASE PART statement is executed) notifies the operating system that this partition is available to receive screen input initiated by a terminal operator. Execution is suspended until a request for screen input is received.

To initiate a request for screen input, the operator must hold down SHIFT and EDIT for at least two seconds. The operating system then temporarily attaches the terminal to a partition waiting to receive screen input. If no such partition is available, the screen input request remains pending until a partition becomes available, or until the operator aborts the request. The operator can abort the request by pressing RESET, or SHIFT and RESET. Pressing RESET terminates the screen input request and leaves the CRT screen intact. Pressing SHIFT and RESET terminates the request and clears the screen.

To handle more than one screen input concurrently, more than one partition can be set up to receive screen input. The operating system selects one of the available partitions for each screen input. Once the terminal is attached to a partition, the INPUT SCREEN statement resumes execution.

When INPUT SCREEN is finished, the partition remains assigned to the terminal, but becomes a background partition. The programmer can identify the terminal that executed the screen input request by using the #TERM function. As soon as the screen image is processed, the partition can be made available for another screen input by reexecuting the $RELEASE PART and INPUT SCREEN statements.

Example:

The following example outlines how to structure a program to receive screen input:

```
10 DIM A$(51)80
20 REM Release partition from any terminal
 : $RELEASE PART
30 REM Wait for screen input
 : INPUT SCREEN A$( )
40 REM Process screen input
50 GOTO 20
```

Example of valid syntax:

```
INPUT SCREEN A$( )
```

High-order HEX Digit

Low-order HEX Digit	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	â	Space	0	@	P	°	p	.	•						
1	ê	!	1	A	Q	a	q	♦	◇						
2	î	"	2	B	R	b	r	▶	▲						
3	ô	#	3	C	S	c	s	◀	▼						
4	û	$	4	D	T	d	t	→	↓						
5	ä	%	5	E	U	e	u	⌙	¬						
6	ë	&	6	F	V	f	v	|	√						
7	ï	'	7	G	W	g	w	··	∘						
8	ö	(	8	H	X	h	x	'	{						
9	ü	)	9	I	Y	i	y	`	}						
A	à	*	:	J	Z	j	z	^	Δ						
B	è	+	;	K	[	k	§	■	□						
C	ù	,	<	L	\	l	£	!!							
D	Ä	-	=	M	]	m	é	↕							
E	Ö	.	>	N	↑	n	ç	ß							
F	Ü	/	?	O	←	o	¢	¶							

Figure 2-1. Alternate Character Set

General Form (as a statement):

TIME = { alpha-variable / literal-string } PASSWORD { alpha-variable / literal-string }

General Form (as a function):

TIME

TIME Statement

The TIME statement sets or changes the system time. The programmer can specify the new ASCII time in an alpha-variable or a literal-string. The time is specified as a six character alphanumeric value in the form HHMMSS (hours, minutes, seconds).

The alpha-variable or literal-string following the keyword PASSWORD represents the system password. If the given password is correct (i.e., matches @GENPART password), the system time is updated. Otherwise, the system returns a recoverable error (ERR X79 - Invalid Password).

The system time can only be set if a system clock exists. If a clock does not exist and a user attempts to set the time, the system signals an error.

Examples of valid syntax:

```
TIME = "170001" PASSWORD "SYSTEM"
TIME = T$ PASSWORD P$
```

TIME Function

TIME is an alphanumeric function that returns an eight character ASCII string containing the current time in the form HHMMSSCC (hours, minutes, seconds, centiseconds). TIME is used as an operand in alphanumeric expressions.

The TIME function returns a value of 99999999 (i.e., invalid) if there is no system clock.

Examples of valid syntax:

```
A$ = TIME & " " & DATE
T$ = TIME
```

CHAPTER 3
INITIALIZE DATE & TIME UTILITY

3.1 INTRODUCTION

The operator can select the Initialize Date & Time utility from the System Utilities menu (refer to Figure 3-1). The Initialize Date & Time utility (@CLOC) allows the operator to edit the system date and, if a system clock exists, the system time. A system equipped with a clock automatically advances the date at midnight. If the system does not have a clock, the operator must reset the date each day.

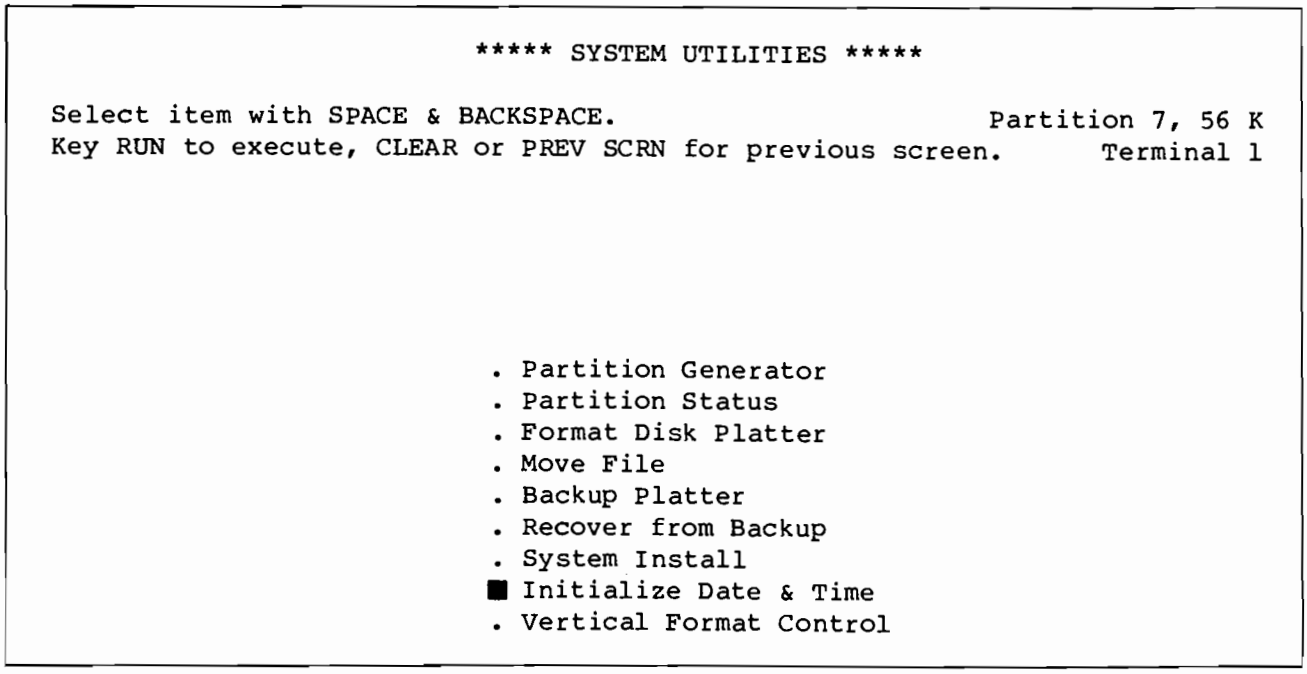

Figure 3-1. System Utilities Menu

3.2 OPERATION

Whenever the system password is not SYSTEM, the utility first requests the system password (refer to Figure 3-2). The operator enters the system password and presses RETURN to proceed editing the date and/or time.

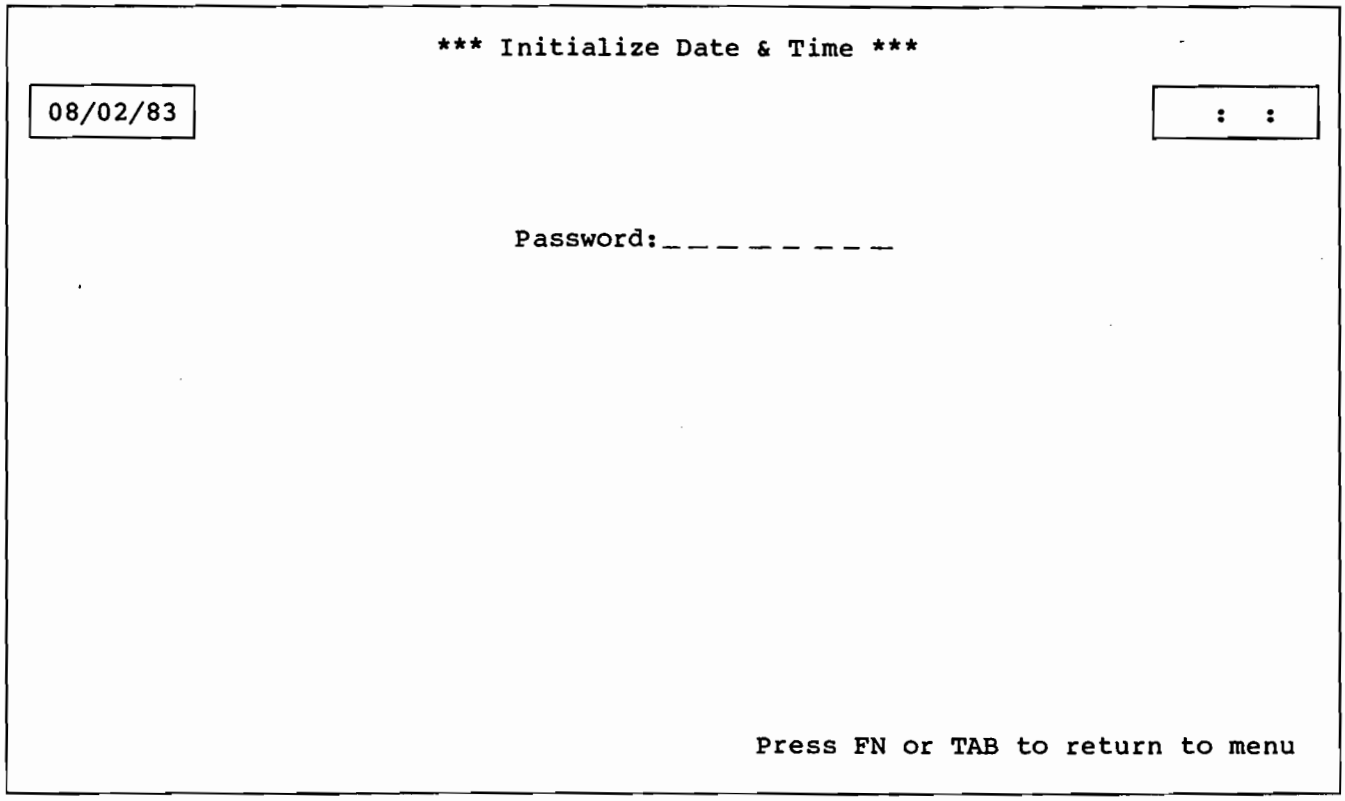

Figure 3-2. Password Screen

When the password requirement is satisfied, the utility displays the last entered date, the calendar page for the corresponding month, the calendar page for the succeeding month, and an enter date prompt (refer to Figure 3-3).

The operator can then enter any valid date (in the form MM/DD/YY). When the operator presses RETURN, the system updates the date, saves the date in a data file named @DATE, and displays the appropriate calendar pages.

If the system permits, the operator can enter any valid time (in the form HH:MM:SS). When the operator presses RETURN, the system updates the time.

At this point, the operator can re-edit the date and/or time. Pressing FN or TAB accepts the date and/or time and returns the System Utilities menu to the screen.

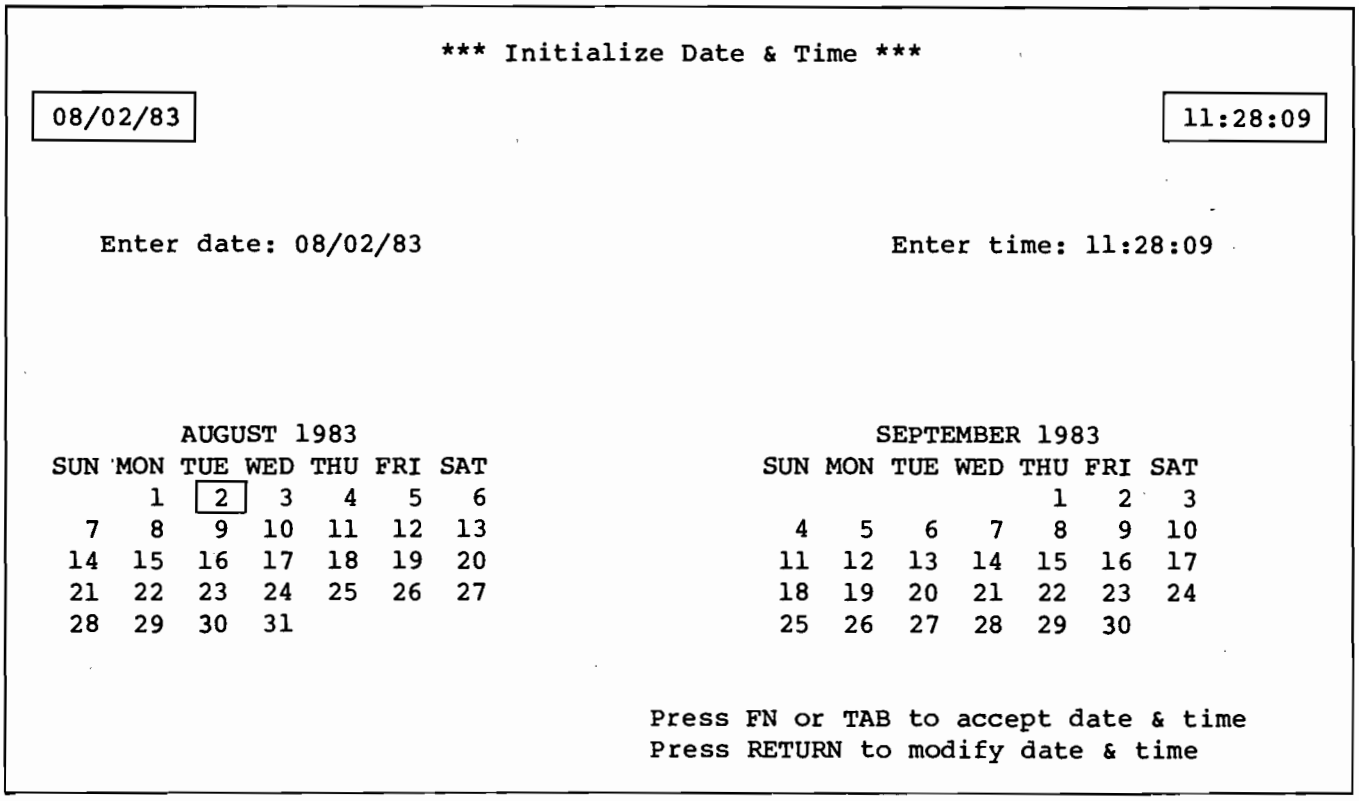

Figure 3-3. Edit Date & Time Screen