FMT1000 Series
Programmer's Guide

70-16674-01
Revision A
October 1995

Symbol Technologies, Inc.
116 Wilbur Place
Bohemia, N.Y. 11716

Symbol Support Center

For service information, warranty information, or technical assistance in the U.S.A., call:

SYMBOL SUPPORT CENTER
1-800-653-5350

If you purchased your Symbol product from a Symbol Business Partner, contact that Business Partner for service.

Canada

Mississauga, Ontario
Canadian Headquarters
(905) 629-7226

Europe

Wokingham, England
European Headquarters
734-771-222 (Inside UK)
011-44-734-771222 (Outside UK)

Asia

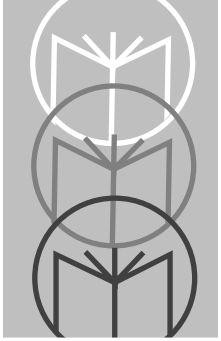
Singapore
Symbol Technologies Asia Inc.
337-6588 (Inside Singapore)
+65-337-6588 (Outside Singapore)

Contents

About This Guide

Chapter 1. FMT1000 Architecture

Chapter 2. I/O System Programming



About This Guide

Introduction

This manual is intended for programmers who are creating programs that will run on the FMT1000 Series of data collection computers. All of the FMT1000 Series computers have the same internal architecture and are covered in this manual. Differences such as keyboard layout are noted where necessary.

The most important thing to remember about programming the FMT1000 Series computers is that they are truly DOS PC compatibles and that most software will run without modification. Because FMT1000 Series computers are DOS compatible, they can be programmed using any DOS development system or language, from assembler to C to BASIC. Although the extended features described in this manual are unnecessary for many applications, intelligent use of these features will enhance the value and usefulness of FMT1000 series computers in every application.

This manual covers the following subject areas:

- FMT1000 Series architecture – what's in the product and how the pieces are connected
- Memory management
- Configuration and use of the I/O coprocessor.

Key Conventions for the FMT1000 Series Programmer's Guide

Keys and key sequences are printed in <lowercase> and enclosed in broken brackets. For example:

- <**enter**> the enter or return key.
- <**tab**> the tab key.
- <**ctrl**>-<***z***> the FMT1000 Series computer key sequence equivalent of the standard IBM PC <CTRL+Z> key combination.
- <**f6**> the F6 function key.
- <**udk1 up**> User Defined Key.

See page 2-101 for more information about user-defined keys.

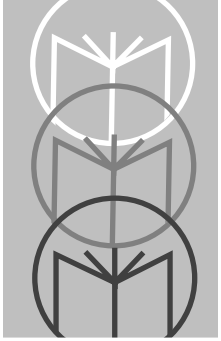
If you are using the FMT1000 Series computer as your workstation, you must press and release the <**ctrl**> key, then press and release the <***z***> key to achieve the same result as when using the standard IBM PC keyboard. No two or more keys are pressed at the same time in operating the FMT1000 Series computer.

Command lines to be entered from the keyboard will appear in **`BOLD UPPERCASE COURIER`**, e.g., **`FORMAT A:`**.

The prompt appears in regular courier, e.g., `C:\>` or a simple colon (:).

Commands entered in DOS are shown using a DOS style prompt with the correct drive and subdirectory, e.g., F:\LOGIN>.

Commands entered on the Novell NetWare server are shown using the Novell prompt, : (colon).



Chapter 1
FMT1000 Architecture

Introduction

The FMT1000 Series includes three models:

- the compact FMT1020
- the rugged FMT1040
- the FMT1060 for time clock applications.

The specific features of each model are covered in the installation guide provided with each unit. The architecture and command set are common to all models. Throughout this manual all models will be referred to as the *FMT1000 Series computer*.

FMT1000 Series computers contain two processors for operating and managing the operations of the unit:

- The *CPU* is the 8X86-compatible main processor that runs DOS and your applications.
- The *coprocessor* controls much of the FMT1000 Series computer's specialized I/O capability.

These two processors are connected in two ways:

- The CPU sends commands formatted as ASCII strings to the coprocessor, using COM2. This permits commands to be sent to the coprocessor by any program that can write to COM2, including redirected I/O from a batch file. (See Figure 1-1.) For example:

  ```
  echo SBL1 >com2
  ```

  turns the display backlight on.

- The coprocessor responds to requests for information and transmits

information coming from I/O devices, either by using the keyboard port, making the data appear as though it had been typed, or by sending it to COM2. Complete details of coprocessor operation and interfacing are contained in Chapter 2 of this manual.

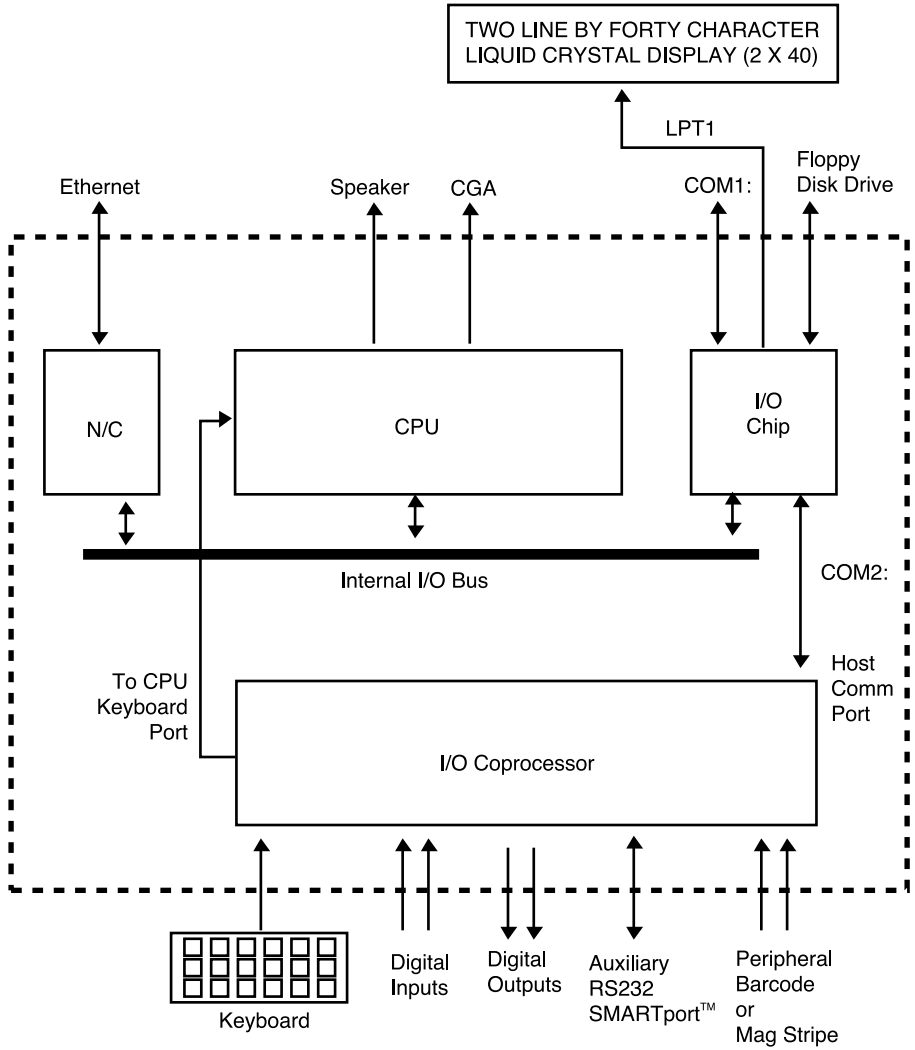


Figure 1-1. FMT1000 Series Computer Architecture

Please continue to refer to Figure 1-1 as the other architectural features of the FMT1000 Series computer are described below.

CPU

The FMT1000 Series CPU is the Chips & Technology F8680 PC / Chip processor. This highly integrated processor contains much of the basic I / O for the FMT1000 Series computer in addition to the CPU core. This processor, as used in FMT1000 Series computer, executes the 8086 instruction set, with an execution speed comparable to a 12-MHz 80286 processor. The processor does have extended capabilities, though not 100 % compatible with the 80286, that allow it to provide access to the high memory area and expanded memory (see memory management, below). For further information about the capabilities of the F8680 CPU, consult the Chips & Technology specifications.

Note: With the wide availability of 80286, 80386 and compatible processors, it is very common to compile programs to use the extended features of these processors. Programs compiled in this manner will not operate correctly on a FMT1000 Series computer. You must recompile your entire program (all modules, not just those that have changed), disabling the use of 80286 and 80386 instructions. Programs that autodetect the CPU should work correctly if they use instruction based tests to determine CPU type. Programs that make oversimplified assumptions (e.g., if a high-density floppy is attached, the CPU must be at least an 80286) may cause problems.

Memory

The FMT1000 Series computer contains 1 MB of RAM. This memory may be exploited in various ways to afford optimal usage in a variety of systems. The first usage choice is presented during system setup, where a selection is made from two possible memory maps, A and B, as shown in Figure 1-2.

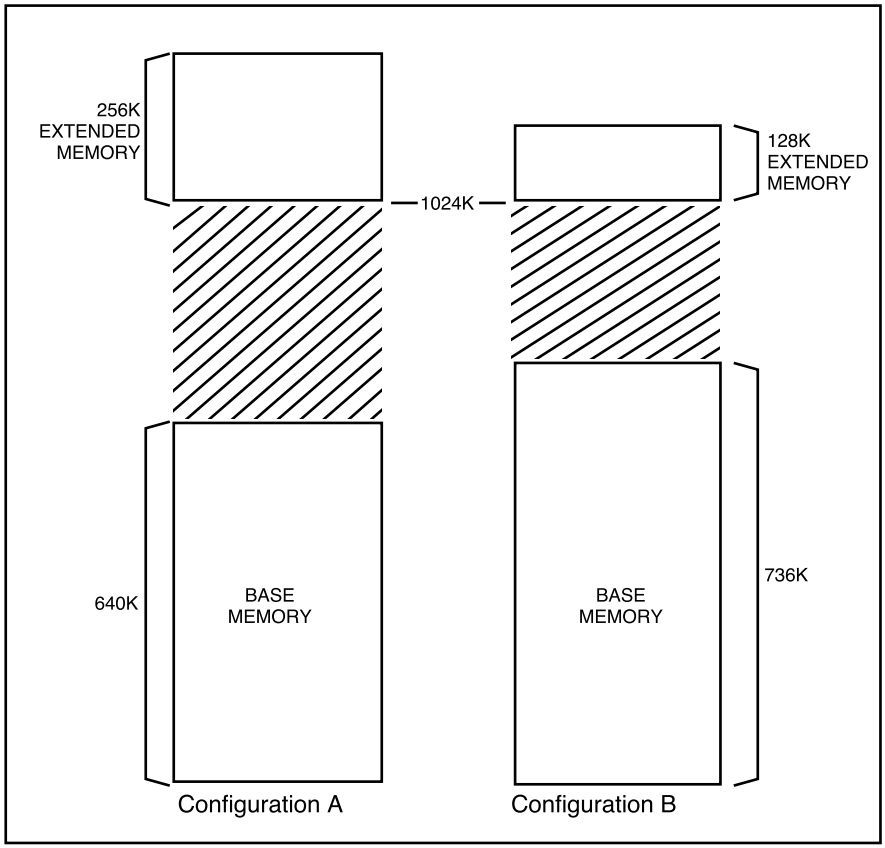


Figure 1-2. Memory Configuration Options

- Configuration A allows a base memory partition of 640 KB, which is the norm for PC-compatible computers.
- Configuration B extends the base memory partition by 96 KB to a total of 736 KB. This extra memory is directly usable by DOS without further complication of a memory management program. More space is available, therefore, for loadable device drivers, TSR programs, and / or larger application programs.

Configuration B is the simplest, most transparent way to obtain greater memory utilization. In some cases, some third-party programs may fail when presented with addresses above the 640 MB boundary. If you are using Configuration B, you should carefully test your application to ensure that it is compatible with the larger base memory configuration.

Careful inspection of the configuration illustrations will reveal that in the transformation from map A to map B, 128 KB are removed from the memory above 1 MB, while only 96 KB are added to the base memory partition. This loss of 32 KB is dictated by the limitations of the on-chip memory management hardware. In many cases, the added base partition memory, which is so easily used, offsets the loss of capacity.

Because the FMT1000 Series computer is a DOS-compatible computer, some of the same advanced memory utilization methods are available:

- High Memory Area (HMA)
- Extended Memory Manager (XMM)
- Expanded Memory Manager (EMM)
- Upper Memory Block (UMB).

The same concepts and methods with which the user is familiar on DOS machines may be used on the FMT1000 Series computer. However, the F8680 processor and its memory management hardware render the ordinary routines distributed with DOS for these purposes unusable here. Special routines distributed with FMT1000 Series computer are used in their stead.

XMM and HMA are both implemented by the Extended Memory Manager. FMT1000 Series computers offer two from which to choose.

- HIDOS.SYS, the simpler of the two, is the one which **must** be used if Expanded Memory is also required. HIDOS.SYS provides no UMB support.
- The second Extended Memory Manager is LPHIMEM.SYS. In addition to XMM and HMA, this driver implements UMB storage, allowing use of DEVICEHIGH and LOADHIGH commands from DOS. LPHIMEM.SYS is incompatible with Expanded Memory.

EMM is implemented by the driver PCCEMM.SYS. While fully functional on FMT1000 Series computer, the small area of RAM available for such use implies caution when implementing applications using EMM. Once again, PCCEMM.SYS is incompatible with the XMM LPHIMEM.SYS.

COM1

The COM1 port on the back of the FMT1000 Series computer is completely compatible with a standard PC COM1 port, including pinouts, base address, and interrupt.

CGA Output

The CGA output port on the back of the FMT1000 Series computer is completely compatible with a standard CGA and may be connected to any compatible monitor (including CGA, EGA, and digital multisync) using the available cable. Programming is also completely CGA compatible.

Network Adapter

The network adapter on the FMT1000 Series computer is implemented using a Fujitsu MB86965 network interface controller (NIC). Normally all network functions are performed by the supplied drivers, and there should be no reason to access the NIC registers directly. However, for reference, the I/O address of the NIC is 300H, and it is connected to interrupt 5.

Display Programming

As can be seen from Figure 1-1, the LCD display is connected to the LPT1 port of the CPU via the I/O chip. This considerably eases the use of the display, since any program capable of writing to LPT1 can write to the display. A modified version of the standard BIOS Int 17 function is provided for this service.

Operational Differences

There are several basic differences in operation of the display as compared to writing to a printer, and a number of extended function codes:

- Because the full PC character set cannot be displayed, undisplayable characters are intercepted and mapped to a unique display pattern.
- Carriage Return (0DH) returns the cursor to the leftmost position of the same line.
- Line Feed (0AH) moves the cursor down one position in the same column and scrolls if necessary.
- Form Feed (0CH) clears the display.

BIOS Functions

The BIOS functions respond as follows (numbers are in hexadecimal format unless otherwise noted):

The normal three Int 17 functions remain the same (for each function, DX=0):

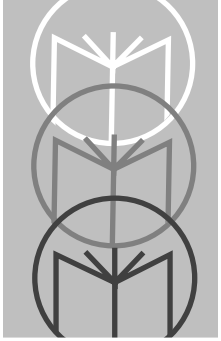
Function in Hexadecimal	**Description**
AH 00	Send byte (AL-char. DX-0). Return status in AH.
AH 01	Init printer (DX-0). Always OK: returns 0 in AH.
AH 02	Get status (no parameters). Always OK: returns 0 in AH.

Additional functions have been provided (for each function, DX=0 and all functions are in hex):

Function in Hexadecimal	Description
AH 10	Initialize port and LCD panel. This is an LCD specific init routine which takes the place of INT 17 FX 01.
AH 11	Clear display. The display is written with white space to clear it.
AH 12	Update entire display. Writes a complete 80 character buffer to the display. This pointer is in segment: offset style (ES:[DI]).
AH 13	Write null-terminated string to display. This pointer is in segment: offset style (ES:[DI]).
AH 14	Return cursor and shift state tracking modes. Returns in BH: 0: Underline cursor (not displayed) 1: Underline cursor (displayed) 2: Block cursor (not displayed) 3: Block cursor (displayed) Returns in BL: 0: Shift state not tracked 1: Shift state tracked
AH 15	Set cursor and shift status tracking modes. The subfunction codes set in BH are: 0: Underline cursor (not displayed) 1: Underline cursor (displayed) 2: Block cursor (not displayed) 3: Block cursor (displayed) Set in BL: 0: Shift state not tracked 1: Shift state tracked A value of 10H is added to the value to set to indicate that a change of state is desired. Thus, BH=1 becomes BH=11

Function in Hexadecimal	Description
AH 16	Return cursor position. Returns the position of the cursor in the screen: BH=Row (0 or 1) BL=Column (0-39)
AH 17	Set cursor position. Set the desired cursor position in BH and BL: BH=Row (0 or 1) BL=Column (0-39) Any value outside valid range will be ignored.
AH 18	Cursor up one row.
AH 19	Cursor down one row.
AH 1A	Cursor left one column.
AH 1B	Cursor right one column.
AH 1C	Return location of display buffer. Returns the address of the 80 character buffer that is periodically copied to the display. Returns in ES:[DI]
AH 1D	Report Display Tracking Items BH 00000abc where (abc are binary register bits): a = cursor source (0: BIOS, 1: CRTC) b = CGA snow (0: snow, 1: remove snow) c = display speed (0: fast, 1: slow) BL = Display Tracking (0: disable, 1: enable)
AH 1E	Set Display Tracking Items BH 00000abc where(abc are binary register bits): a = cursor source (0: BIOS, 1: CRTC) b = CGA snow (0: snow, 1: remove snow) c = display speed (0: fast, 1: slow) BL = Display Tracking (0: disable, 1: enable) A value of 10H is added to the value to set to indicate that a change of state is desired. Thus, BH=1 becomes BH=11
AH 1F	Report current tracking mode BL= 1,2,3, or 4

Function in Hexadecimal	Description
AH 20	Set new tracking mode in BL 1: 1x80 fixed 2: 1x80 follow cursor row 3: 2x40 fixed 4: 2x40 left justified, cursor in second row. A value of 10H is added to the value to set to indicate that a change of state is desired.
AH 21	Report tracking position BH: row position 0..23 for 2x40, 0..24 for 1x80 BL: column position 1..39 for 2x40, 0 for 1x80
AH 22	Set tracking box position BH: row position 0..23 for 2x40, 0..24 for 1x80 BL: column position 1..39 for 2x40, 0 for 1x80
AH 23	Force display update



Chapter 2
I/O System Programming

I/O system programming consists mainly of selecting the configuration options for the many features of the I/O coprocessor. To understand the programming, it is first necessary to understand the basic characteristics and features of the I/O.

Features

Much of the power of the FMT1000 Series computer, particularly for industrial and automation applications, lies in its extended I/O capability. This I/O is managed by the coprocessor, thus relieving the CPU of much of the work, and making the I/O much easier to work with from a DOS program. The I/O falls into four categories:

- ***Peripheral*** – FMT1000 Series computers can accept input from one or two bar code devices or a magnetic stripe reader. They support all commonly used bar code symbologies. The magnetic stripe reader is compatible with ABA Track 2 encoding.
- ***RS-232 Auxiliary Port*** – Connecting ordinary PC COM ports to real-world devices can require significant, difficult, real-time programming. The RS-232 auxiliary port automatically:
  - Handles flow control
  - Provides an interrupt-driven input buffer
  - Strips headers and trailers from a data stream (such as might come from a scale printer port) and forwards only the data to your application
  - Automatically adds headers and trailers to commands written to an attached device. An example of this is Hayes-compatible modem programming, in which every command must be preceded by an ***AT*** and terminated with a carriage return

  - Can transmit its data directly to the keyboard port of the CPU, eliminating the need for any special programming to receive the data.

- ***Digital Input*** – Two digital inputs can be configured to:
  - Sense whether they are "On" or "Off"
  - Detect a change in state and inform the CPU
  - Count transitions from "Off" to "On" for cycle-counting applications such as printing or machine monitoring.
- ***Digital Output*** – Two relay outputs can be set "On" or "Off" or pulsed for a period from 0.1 to 99.8 seconds.

In addition, several other FMT1000 Series computer internal housekeeping features are controlled by the coprocessor and can be manipulated using coprocessor commands. These include the internal beeper, display backlight and contrast angle adjustment, keyboard layout, and keyclick operation.

Basics of Coprocessor Communications

Communicating with the coprocessor has two parts:

- sending commands to the processor
- receiving replies.

As can be seen from Figure 1-1, the two processors can communicate using COM2 and the keyboard port.

Sending Commands to the Coprocessor

All coprocessor commands are ASCII strings, consisting of a three-character mnemonic followed by any necessary parameters. They are always transmitted via COM2. It is essential to ensure that both processors use the same baud rate and character format. The coprocessor host comm port defaults to 9600 baud, 8 data bits, no parity. In most situations it is unnecessary to change this.

> **Note:** The FMT1000 Series computer BIOS sets the COM2 communication parameters to the coprocessor default at system startup.

Coprocessor Command Format

All commands to the coprocessor have the following form:

```
<mnemonic>[<param>]<cr><lf>
```

where:

<mnemonic>	is the three-character mnemonic for the command
[<param>]	is an optional parameter
<cr><lf>	is a carriage return (hex 0D), line feed (hex 0A)

> **Note:** No spaces are allowed between the command and the parameter.

Many parameters accept string arguments. To embed nonprinting control characters in these strings, use the format **^<key>**, which is a two-character sequence. (**<key>** refers to the key that would be used in concert with a keyboard control key to generate the code from a keyboard.) Some commonly used values are:

Sequence	Meaning
^I	Tab
^J	LF
^M	CR
^L	FF
^[	Escape

Note: These sequences consume two characters in the specified maximum string length for a command.

Flow Control on the Coprocessor Port

The default flow control for the coprocessor is to use XON/XOFF handshaking. There is a 100-character input buffer, and the coprocessor will send XOFF when 88 characters have been received and XON when the buffer has been emptied. Remember, DOS provides no flow control on these ports.

You can use ECHO commands in BAT files to configure the coprocessor. For example,

```
echo SBL1 >COM2
```

turns the LCD display backlight on. However, sending too many commands this way can overflow the buffer, and there is no way to detect this condition. There are several solutions:

- Install an XON/XOFF handler for COM2. Several are available on CompuServe.
- Use a programming environment or a communications library that supports flow control on COM ports to write a program that sends commands to the coprocessor.

- Limit the total characters to fewer than 100. This is often adequate, as most coprocessor commands are short, and the default configuration is adequate for many applications.
- Synchronize with the coprocessor by including a command for which you expect a reply, and wait for the reply.

Receiving Replies from the Coprocessor

Replies can come from four sources:

- Reply strings generated internally by commands that request data from the coprocessor, such as returning the current setting of the baud rate for the RS-232 auxiliary port. The internal input path is set by the SRI command (*set internal response path*).
- Data coming from the auxiliary input. The auxiliary input path is set by the SRA (*set auxiliary response path*).
- Data coming from the peripheral input (bar code or magnetic stripe). The peripheral input path is set by the SRP command *(set peripheral path*).
- Strings from the digital inputs generated by the change-of-state or counter-match functions. The digital input path is set by the SRD command (set *digital response path*).

The replies from any of these sources can either be directed to the keyboard port and received as though typed or directed to COM2. The response path for each source is set by command. For example, the *set aux comm response path* command (SRA) will direct the auxiliary port data.

Coprocessor Reply Format

All replies coming from the coprocessor have a common format:

```
<tag><prefix><data><suffix><autoenter>
```

Where:

tag	is a single programmable character
prefix	is a programmable 16-character string
data	is a 1- to 80-character string
suffix	is a programmable-16 character string
autoenter	is a mode that determines how a reply is terminated.

Response tags, prefixes and suffixes can be used to make messages coming from the coprocessor to the CPU unique. For example, suppose your program were waiting for an input from either the auxiliary RS-232 port or a bar code wand. You could set the response prefix for peripheral input to BAR using the command:

```
SPPBAR
```

which stands for "***S***et ***P***eripheral ***P***refix," and set the response prefix to AUX for the RS-232 auxiliary port using the command:

```
SAPAUX
```

which stands for "***S***et ***A***uxiliary ***P***refix." Your program can then examine the first three characters of the data it receives to determine the source.

The tag characters behave very much like another prefix. Autoenter mode determines how data is terminated when it is sent to the CPU. There are four choices:

- ***none*** no terminator is sent
- ***CR*** a carriage return is sent
- ***CR/LF*** a carriage return / line feed is sent
- ***TAB*** a tab character is sent

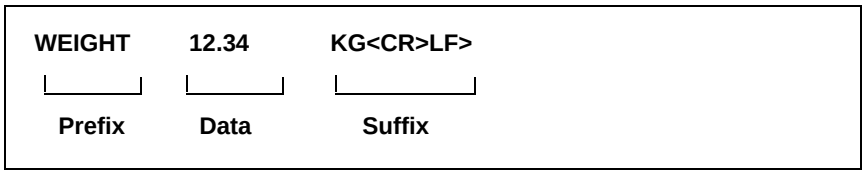
Note: You may program other terminators by turning autoenter off and embedding the termination string in the response suffix.

Prefixes and Suffixes in RS-232 Auxiliary Port Operations

The coprocessor manipulates strings for many operations to make the job of programming the CPU easier. When programming the RS-232 auxiliary port, consider strings to consist of three parts:

- prefix
- data
- suffix.

Auxiliary input prefixes and suffixes are used to find the actual data in a data stream and separate it for input to the CPU. For example, if the following data were coming from a scale printer port to the RS-232 auxiliary port



you could use the following commands to set the aux input prefix and suffix:

```
SPIWEIGHT
SSIKG^M^J
```

When you set the aux input prefix and suffix correctly for the RS-232 auxiliary port, the FMT1000 Series computer automatically captures just the numeric data from this string (12.34), and sends it to the CPU as though it had been typed at the keyboard.

Aux output prefixes and suffixes refer to strings that are added to each of the messages sent in the RS-232 auxiliary port output. They are often used when a device requires commands to begin always with the same characters. For example, Hayes-compatible modem commands begin with ***AT*** and terminate with a carriage return. The following commands could be used to set the auxiliary output prefix and suffix:

```
SPOAT
SSO^M
```

A dialing command could then be sent using:

```
SOWDT5551212
```

Creative use of auxiliary prefixes and suffixes can greatly ease the job of interfacing to external devices. In many cases, correctly setting the prefixes and suffixes for such devices as the scale above can make it possible to interface these devices as keyboard inputs with no special programming.

Use of the I/O Processor

A unique feature of all FMT1000 Series computers is an embedded I/O processor used to decode input from bar code wands, magstripe readers, and other I/O devices. Internally, this I/O processor is connected to both COM2: of the F8680 processor and the XT keyboard input. This design allows for bar code data to appear as "typed" input into application programs, yet permits configuration of decoder parameters without the need for specialized programs or drivers. For example, to set the I/O processor to append an apparent press of the ENTER key after reading a bar code, the following may be placed in the AUTOEXEC.BAT file:

```
echo SEP1>COM2
```

where the command SEP1 is explained in the following sections. Similarly, programs such as QBASIC may open COM2: and write the configuration strings and thereby change the behavior of the I/O processor dynamically.

Note: The default communication settings for the I/O processor are 9600 baud, no parity, 8 data bits, 1 stop bit (96,n,8,1). It may be necessary to preconfigure COM2: in the AUTOEXEC.BAT file prior to issuing configuration commands to the IO processor by including the command:

```
lpmode COM2:96,n,8,1
```

where *lpmode.exe* is found in the \UTL\ directory of the drivers and utilities disk. Alternatively, the DOS external command MODE.EXE may be used.

I/O Processor Commands

The subsequent pages contain a list of I/O controller mnemonics. Each page describes one command. Certain conventions are used for arguments:

- **<Broken brackets>** indicate the type of information to be passed; for example <string> indicates a string.
- Ranges of numbers are indicated as numbers in broken brackets, for example ***<0-80>***.
- **N/A** indicates that no arguments are passed or returned.

The basic format of a command description page is:

SIO, RIO *[mnemonic]* Set and Return Aux Input Timeout *[Title]*

Format

`SIO <0-255>`

[Format and parameters for the send command]

Summary

Set the length of time to wait for input, in seconds
[One sentence description of the command]
This setting is preserved in EEPROM.

Returns

Timeout value, integer 0-255.
[Type and value of returned data]

Description

[Explanation of the command and parameters]

Example

[Code fragment showing how the command can be used. Examples are offered when needed.]

Note: Factory defaults are indicated by an asterisk (*).

Command List

Command	Type	Pg.#	Command	Type	Pg.#
R1D	System	2-101	**RDA**	Auxiliary	2-27
R1M	Digital	2-65	**RDE**	Digital	2-67
R1U	System	2-101	**RDH**	Host	2-16
R25	Bar Code	2-77	**RDP**	Digital	2-55
R28	Bar Code	2-80	**RDS**	Digital	2-56
R2D	System	2-101	**RDT**	Digital	2-54
R2M	Digital	2-65	**RE1**	Digital	2-62
R2U	System	2-101	**RE2**	Digital	2-62
R39	Bar Code	2-76	**REA**	Auxiliary	2-24
R3D	System	2-101	**RED**	Digital	2-57
R3U	System	2-101	**REI**	System	2-90
R4D	System	2-101	**REP**	Bar Code	2-74
R4U	System	2-101	**RER**	System	2-106
RAD	System	2-94	**RES**	System	2-110
RAE	Auxiliary	2-50	**RHA**	Auxiliary	2-29
RAP	Auxiliary	2-22	**RHH**	Host	2-18
RAS	Auxiliary	2-23	**RI1**	Digital	2-59
RAT	Auxiliary	2-21	**RI2**	Digital	2-59
RBA	Auxiliary	2-26	**RIB**	Auxiliary	2-39
RBE	System	2-112	**RIM**	Auxiliary	2-37
RBH	Host	2-15	**RIN**	Auxiliary	2-41
RBL	System	2-92	**RIO**	Auxiliary	2-36
RBS	System	2-113	**RIP**	System	2-88
RCB	Bar Code	2-78	**RIR**	Auxiliary	2-40
RCD	System	2-100	**RIS**	System	2-89

Command	Type	Pg.#	Command	Type	Pg.#
RIT	System	2-87	**RS2**	Digital	2-60
RKC	System	2-99	**RSI**	Auxiliary	2-34
RKE	System	2-96	**RSA**	Auxiliary	2-28
RKH	System	2-95	**RSB**	Bar Code	2-83
RKR	System	2-98	**RSC**	Bar Code	2-81
RKT	System	2-97	**RSH**	Host	2-17
RM1	Digital	2-64	**RSI**	Auxiliary	2-34
RM2	Digital	2-64	**RSM**	Bar Code	2-82
ROB	System	2-111	**RSO**	Auxiliary	2-32
ROO	Auxiliary	2-45	**RSS**	System	2-109
ROU	Auxiliary	2-46	**RT1**	Digital	2-66
RPS	Bar Code	2-73	**RT2**	Digital	2-66
RP5	Bar Code	2-70	**RTC**	Auxiliary	2-49
RP9	Bar Code	2-70	**RUP**	Bar Code	2-79
RPA	Auxiliary	2-48	**RVA**	System	2-93
RPE	Bar Code	2-84	**RVR**	System	2-103
RPI	Auxiliary	2-33	**S1D**	System	2-101
RPO	Auxiliary	2-31	**S1M**	Digital	2-65
RPP	Bar Code	2-72	**S1U**	System	2-101
RPT	Bar Code	2-71	**S25**	Bar Code	2-77
RR1	Digital	2-63	**S28**	Bar Code	2-80
RR2	Digital	2-63	**S2D**	System	2-101
RRA	Auxiliary	2-25	**S2M**	Digital	2-65
RRD	Digital	2-58	**S2U**	System	121
RRI	System	2-91	**S39**	Bar Code	2-76
RRP	Bar Code	2-75	**S3D**	System	2-101
RS1	Digital	2-60	**S3U**	System	2-101

Command	Type	Pg.#	Command	Type	Pg.#
S4D	System	2-101	**SHA**	Auxiliary	2-29
S4U	System	2-101	**SHH**	Host	2-18
SAD	System	2-94	**SI1**	Digital	2-59
SAE	Auxiliary	2-50	**SI2**	Digital	2-59
SAP	Auxiliary	2-22	**SIB**	Auxiliary	2-39
SAS	Auxiliary	2-23	**SIC**	Auxiliary	2-35
SAT	Auxiliary	2-21	**SII**	Auxiliary	2-38
SBA	Auxiliary	2-24	**SIM**	Auxiliary	2-37
SBE	System	2-112	**SIO**	Auxiliary	2-36
SBH	Host	2-15	**SIP**	System	2-88
SBL	System	2-92	**SIR**	Auxiliary	2-40
SBP	System	2-108	**SIS**	System	2-89
SBS	System	2-113	**SIT**	System	2-87
SCB	Bar Code	2-78	**SKC**	System	2-99
SCD	System	2-100	**SKE**	System	2-96
SDA	Auxiliary	2-27	**SKH**	System	2-95
SDE	Digital	2-67	**SKR**	System	2-98
SDH	Host	2-16	**SKT**	System	2-97
SDP	Digital	2-55	**SM1**	Digital	2-64
SDS	Digital	2-56	**SM2**	Digital	2-64
SDT	Digital	2-54	**SOC**	Auxiliary	2-44
SEA	Auxiliary	2-24	**SOO**	Auxiliary	2-45
SED	Digital	2-57	**SOW**	Auxiliary	2-43
SEE	System	2-105	**SP5**	Bar Code	2-70
SEI	System	2-90	**SP9**	Bar Code	2-70
SEP	Bar Code	2-74	**SPA**	Auxiliary	2-48
SFD	System	2-104	**SPE**	Bar Code	2-84

Command	Type	Pg.#	Command	Type	Pg.#
SPI	Auxiliary	2-33	**SSB**	Bar Code	2-83
SPO	Auxiliary	2-31	**SSC**	Bar Code	2-81
SPP	Bar Code	2-72	**SSH**	Host	2-17
SPS	Bar Code	2-73	**SSI**	Auxiliary	2-34
SPT	Bar Code	2-71	**SSM**	Bar Code	2-82
SR1	Digital	2-61	**SSO**	Auxiliary	2-32
SR2	Digital	2-61	**SSS**	System	2-109
SRA	Auxiliary	2-25	**ST1**	Digital	2-66
SRD	Digital	2-58	**ST2**	Digital	2-66
SRI	System	2-91	**STC**	Auxiliary	2-49
SRP	Bar Code	2-75	**SUP**	Bar Code	2-79
SSA	Auxiliary	2-28	**SVA**	System	2-93

Host Comm Port Commands

The host comm port commands configure the coprocessor port that is connected to COM2 of the CPU. Ordinarily there is no reason to change them.

The commands are:

Command	Description	Pg.#
SBH, RBH	Set and Return Host Baud Rate	2-15
SDH, RDH	Set and Return Host Data Bits and Parity	2-16
SSH, RSH	Set and Return Host Stop Bits	2-17
SHH,RHH	Set and Return Host Handshake Type	2-18

SBH, RBH – Set and Return Host Baud Rate

Format

SBH *<0-6>*

0=300 baud
1=600
2=1200
3=2400
4=4800
5=9600 *
6=19200

Summary

Returns or sets the data speed for the coprocessor host communications port.

This setting is preserved in EEPROM.

Returns

Baud rate flag value.

Description

The factory default setting is 9600 baud.

Example

```
rem    Set host baud rate to 9600
echo   SBH5 > COM2
```

SDH, RDH – Set and Return Host Data Bits and Parity

Format

SDH <*0-6*>

0=7 bits, even parity
1=7 bits, odd parity
2=7 bits, space parity
3=7 bits, mark parity
4=8 bits, even parity
5=8 bits, odd parity
6=8 bits, no parity *

Summary

Sets the character length and parity check for the coprocessor host communications port.

This setting is preserved in EEPROM.

Returns

Host data bits and parity flag value

Description

The factory default is 8 data bits, no parity.

Example

```
rem  Set data bits and parity
rem  to 7 bits odd
echo SDH1 > COM2
```

SSH, RSH – Set and Return Host Stop Bits

Format

`SSH 0 | 1`

0 = 1 stop bit *
1 = 2 stop bits

Summary

Sets the number of stop bits for the coprocessor host communications port.

This setting is preserved in EEPROM.

Returns

Stop bit flag value, 0 or 1.

Description

The factory default is 1 stop bit.

Example

```
rem  Set Host Stopbits to 1
echo SSH0 > com2
```

SHH, RHH – Set and Return Host Handshake Type

Format

SHH 0 | 1

0=None
1=XON / XOFF *

Summary

Sets the type of hardware handshaking between the coprocessor host communications port and the host CPU communications port COM2.

This setting is preserved in EEPROM.

Returns

Handshake value flag, 0 or 1.

Description

Enables or disables the use of XON / XOFF handshaking for flow control between the coprocessor and CPU.

If no handshaking is established, data from the CPU can overrun the FMT1000 Series computers's buffer without any control or warning. Likewise, data from the FMT1000 Series computers can overrun the CPU's buffer.

Enabling XON / XOFF will send the XOFF code when the 100-character input buffer contains 88 characters and then XON when it is empty. DOS does not provide handshaking on COM2: by default. Either an XON / XOFF handler must be loaded or the port must be driven by other software that does support handshaking.

The factory default is ON (1).

Example

```
rem    Set Host Handshake type to none
Echo   SHH0 > com2
```

Auxiliary RS-232 Port Commands

There are many modes of auxiliary port operation. Using combinations of the features described below, you can readily manipulate almost any RS-232 device. The first group sets the reply format, as defined earlier:

Command	Description	Pg.#
SAT, RAT	Set and Return Aux Comm Response Tag	2-21
SAP, RAP	Set and Return Aux Response Prefix	2-22
SAS, RAS	Set and Return Aux Response Suffix	2-23
SEA, REA	Set and Return Aux Response Autoenter Mode	2-24
SRA, RRA	Set and Return Aux Comm Response Path	2-25

The next group of commands sets data formats and handshaking with the external device.

Command	Description	Pg.#
SBA, RBA	Set and Return Aux Baud Rate	2-26
SDA, RDA	Set and Return Aux Data Bits and Parity	2-27
SSA, RSA	Set and Return Aux Stop Bits	2-28
SHA, RHA	Set and Return Aux Handshake Type	2-29

The next group sets prefixes and suffixes for RS-232 auxiliary port operation:

Command	Description	Pg.#
SPO, RPO	Set and Return Output Prefix	2-31
SSO, RSO	Set and Return Aux Output Suffix	2-32
SPI, RPI	Set and Return Aux Input Prefix	2-33
SSI, RSI	Set and Return Aux Input Suffix	2-34

The RS-232 auxiliary port can perform I/O asynchronously while the CPU does other tasks. Input can occur continuously— desirable if a device that produces data only on demand, e.g., a large fixed station laser scanner, is connected. Or, if your device constantly streams data readings, you can set the port to capture a single reading and hold it. Further, data may be processed using the prefix and suffix capabilities described earlier, or just passed through without interpretation. Commands related to configuring the input are:

Command	**Description**	Pg.#
SIC	Aux Input Cancel	2-35
SIO, RIO	Set and Return Aux Input Timeout	2-36
SIM, RIM	Set and Return Input String Max Length	2-37
SII	Input Buffer Initialize	2-38
SIB, RIB	Input Beep Enable	2-39
SIR, RIR	Set and Return Input Read Mode	2-40
STC, RTC	Set and Return Passthrough Termination Characters	2-49
RIN	Return Input Status	2-41

Commands related to configuring the output mode are:

Command	**Description**	**Pg.#**
SOW	Set Output Write String	2-43
SOC	Aux Output Cancel	2-44
SOO, ROO	Set and Return Aux Output Timeout	2-45
ROU	Return Output Status	2-46

General configuration commands are:

Command	**Description**	**Pg.#**
SPA, RPA	Set and Return Aux Passthrough Mode	2-48
SAE, RAE	Set and Return Aux Communication Port Enable	2-50

SAT, RAT – Set and Return Aux Comm Response Tag

Format

SAT *<string>*
Length: 1

Summary

Sets a unique ID tag to prefix all input from the auxiliary communications port.

Returns

Aux tag string, 1 character.

Description

The I/O processor can route input via several devices. On occasion, input from several physical devices may be routed to the same system input port. For example, both the auxiliary serial communications port and the peripheral bar code port can send their data to the keyboard buffer.

This tag is a prefix which is added to the input from a given physical device to identify the source of the input.

Example

```
rem    Set aux comm response tag to 'A'
echo   SATA > com2
rem    The output from the RS-232 auxiliary port now has 'A'
rem    as a tag.
rem    If the data 'Hi There' is received,
rem    the string 'AHi There' will be forwarded
rem    to the CPU.
```

SAP, RAP – Set and Return Aux Response Prefix

Format

SAP *<string>*
Max length: 16

Summary

Sets the string to be transmitted before each response to an Aux Comm input operation.

Returns

Response Prefix, string of at most 16 bytes.

Description

This I/O processor command allows a standard prefix to be prepended to the response to a string of characters input to the auxiliary serial RS-232 port.

There is no factory default for this command, and it will not be reset using the SFD set factory defaults command.

Example

```
rem  Set aux response prefix to '1234'
echo SAP1234 > com2
rem  If the message: 'Hi There' is received,
rem  the string '1234Hi There' will be
rem  forwarded to the CPU.
```

SAS, RAS – Set and Return Aux Response Suffix

Format

SAS *<string>*
Max length: 16

Summary

Sets the string to be appended to the response to each auxiliary comm port input or command reply.

Returns

Response suffix, string of at most 16 bytes.

Description

This I/O processor command allows a standard suffix to be sent after the response to each line input to the auxiliary serial RS-232 port, but before the termination string specified by the autoenter mode, if any.

There is no factory default for this command, and it will not be reset using the SFD set factory defaults command.

Example

```
rem  Set aux response suffix to '5678'
echo SAS5678 > com2
rem  If the message: 'Hi There' is received,
rem  the string 'Hi There5678' will be
rem  forwarded to the CPU.
```

SEA, REA – Set and Return Aux Response AutoEnter Mode

Format

SEA *<0-3>*

0=Off *
1=CR (carriage return)
2=CR / LF (carriage return / line feed)
3=Tab

Summary

Sets the type of terminating characters for each response caused by data input via the RS-232 auxiliary port.

This setting is preserved in EEPROM.

Returns

AutoEnter mode flag, integer 0-3.

Description

This mode sets the character transmitted to the host CPU after the data input from the RS-232 auxiliary port.

For most ordinary I / O, carriage return would be appropriate, but Tab might be used if entering data into a multifield screen.

Example

```
rem  Set aux response autoenter mode to <CR>
echo SEA1 > com2
```

SRA, RRA – Set and Return Aux Comm Response Path

Format

SRA 0 | 1

0=Keyboard *
1=COM2 host port

Summary

Sets the RS-232 auxiliary port input path to the keyboard or the host serial port COM2.

This setting is preserved in EEPROM.

Returns

Response path flag, integer 0 or 1.

Description

The data from the RS-232 auxiliary port can be used to simulate input from the keyboard, or it can be transmitted to the host communications port COM2.

The advantage to the keyboard input is that no special programming is required to receive the data. However, keyboard input is slower due to the limitations of keyboard input speed.

The factory default is keyboard simulation input.

Example

```
rem  Set aux comm response path to com2
echo SRA1 > com2
```

SBA, RBA – Set and Return Aux Baud Rate

Format

SBA *<0-6>*

0=300 bits per second (bps)
1=600
2=1200
3=2400
4=4800
5=9600 *
6=19200

Summary

Returns or sets the data speed for the RS-232 auxiliary serial port.

This setting is preserved in EEPROM.

Returns

Baud rate flag value

Description

The factory default setting is 9600 bps

Example

```
rem  Set aux baud rate to 9600
echo SBA5 > com2
```

SDA, RDA – Set and Return Aux Data Bits and Parity

Format

SDA *<0-6>*

0=7 bits, even parity
1=7 bits, odd parity
2=7 bits, space parity
3=7 bits, mark parity
4=8 bits, even parity
5=8 bits, odd parity
6=8 bits, no parity *

Summary

Sets the character length and parity check for the RS-232 auxiliary serial port.

This setting is preserved in EEPROM.

Returns

Data bits and parity flag value.

Description

The factory default is 8 data bits, no parity.

Example

```
rem  Set aux data bits and parity to 7 bits odd
echo SDA1 > com2
```

SSA, RSA – Set and Return Aux Stop Bits

Format

SSA 0 | 1

0=1 stop bit *
1=2 stop bits

Summary

Sets the number of stop bits for RS-232 auxiliary serial port communications.

This setting is preserved in EEPROM.

Returns

Stop bit flag value, 0 or 1.

Description

The factory default is 1 stop bit.

Example

```
rem    Set aux stop bits to 1
echo   SSA0 > com2
```

SHA, RHA – Set and Return Aux Handshake Type

Format

`SHA 0 | 1 | 2`

0=None
1=XON / XOFF *
2=RTS / CTS hardware handshaking

Summary

Sets the type of handshaking between the RS-232 auxiliary port and the external device.

This setting is preserved in EEPROM.

Returns

Handshake value flag, 0, 1 or 2.

Description

Establishes the method by which a device connected to the RS-232 auxiliary port communications port on the FMT1000 Series computer can be directed to start or stop sending data.

If no handshaking is established, data from the device can overrun the FMT1000 Series computer's buffer without any control or warning. Likewise, data from the FMT1000 Series computer can overrun the remote device's buffer.

Handshaking XON / XOFF sends the XOFF code when the 100-character input buffer contains 88 characters and then XON when it is empty.

The remote device must be able to correctly interpret XON / XOFF signals for this handshaking to work correctly.

> Note: RTS / CTS handshaking applies only to output from the FMT1000 Series computer; it cannot be used to pace input. The FMT1000 Series computer asserts RTS whenever it wishes to transmit data. No data is transmitted unless CTS is asserted.

Note: If CTS is not connected, it does not appear to be asserted, and nothing is transmitted. You should not select this mode without correctly connecting RTS and CTS. You should either connect RTS and CTS to the appropriate signals on a remote device, or, if unused, wire RTS and CTS together.

The factory default is XON / XOFF (1).

Example

```
rem    Set aux handshake type to RTS/CTS
echo   SHA2 > com2
```

SPO, RPO – Set and Return Aux Output Prefix

Format

SPO *<string>*
Max length of the character string: 16

Summary

Sets the string to be sent before each message is output to the RS-232 auxiliary serial port.

Returns

Output Prefix, a character string of at most 16 bytes.

Description

This I/O processor command allows a standard prefix to be sent to a serial device connected to the RS-232 auxiliary serial port before each line of output. This string is not sent when passthrough mode (command SPA) is set to ON.

There is no factory default for this command, and it will not be reset using the SFD set factory defaults command.

Example

```
rem   Set output prefix to '123'.
echo  SPO123 > COM2
rem   The output from the RS-232 auxiliary port now has '123'
rem   as a prefix.
```

SSO, RSO – Set and Return Aux Output Suffix

Format

SSO *<string>*
Max length: 16

Summary

Sets the string to be sent after each line is output to the RS-232 auxiliary serial port.

Returns

Output suffix, a character string of at most 16 bytes.

Description

This I/O processor command allows a standard suffix to be sent to a serial device connected to the RS-232 auxiliary serial port after each line of output. This string is not sent when passthrough mode (command SPA) is set to ON.

There is no factory default for this command, and it will not be reset using the SFD set factory defaults command.

Example

```
rem  Set aux output suffix to '5678'
echo SSO5678 > com2
rem  If the string 'Hi There' is sent from the
rem  CPU with the suffix set to '5678', the
rem  message: 'Hi There5678' will be set to a
rem  serial device connected to the RS-232 auxiliary port.
```

SPI, RPI – Set and Return Aux Input Prefix

Format

SPI *<string>*
Max length: 16

Summary

Sets the string that will be removed from the beginning of each input message coming into the RS-232 auxiliary serial port before sending it to the CPU.

Returns

Input prefix, a character string of at most 16 bytes.

Description

This I/O processor command allows a standard prefix to be removed from a string of characters input from the RS-232 auxiliary serial port. This processing is not done when passthrough mode (command SPA) is set to ON.

There is no factory default for this command, and it is not reset using the SFD set factory defaults command.

Example

If the prefix is '1234' and the suffix is '5678' and the message:

```
1234Hi There5678
```

is received, only the string *"Hi There"* is forwarded to the CPU.

```
rem   Set input prefix to '1234'
echo  SPI1234 > COM2
rem   Set the input suffix to '5678'
echo  SSI5678 > COM2
rem   Now only data found between '1234'
rem   and '5678' will be passed.
```

SSI, RSI – Set and Return Aux Input Suffix

Format

SSI *<string>*
Max length: 16

Summary

Sets the string to be removed from the end of each message input to the RS-232 auxiliary serial port.

Returns

Input suffix, a character string of at most 16 bytes.

Description

This I/O processor command allows a standard suffix to be removed from a message from the RS-232 auxiliary serial port. This string is not removed when passthrough mode (command SPA) is set to ON.

There is no factory default for this command, and it will not be reset using the SFD set factory defaults command.

Example

```
rem    The prefix is '1234' and the suffix
rem    is '5678' and
rem    the message is '1234Hi There5678'.
echo   SPI1234 > COM2
echo   SSI5678 > COM2
rem    Only the string 'Hi There' is forwarded
rem    to the CPU.
```

See SPI example code, page 2-33.

SIC – Aux Input Cancel

Format

SIC

Summary

Cancels the current input request.

Returns

N / A

Description

Cancels an outstanding request for input set by the SIR set input read mode command and a read request.

This command is useful when RIN returns an error status on input, and can be used as part of an error recovery routine within the program.

Example

```
rem   Cancel aux input request
echo  SIC > COM2
```

SIO, RIO – Set and Return Aux Input Timeout

Format

SIO *<0-255>*

Summary

Set the length to wait for input, in seconds.

This setting is preserved in EEPROM.

Returns

Timeout value, integer 0-255.

Description

Sets the timeout delay before setting the error status 3 (timeout error) for the RIN return input status command.

If the delay is set to *0*, the device, by definition, will never time out. This is the factory default.

Example

```
rem    Set aux input timeout to 10 seconds
echo   SIO10 > COM2
```

SIM, RIM – Set and Return Input String Length

Format

SIM *<0-80>*
(Factory default is 80)

Summary

Sets the an exact size of the buffer allocated to receive input.

This setting is preserved in EEPROM.

Returns

Size of the string buffer, integer 0-80.

Description

This command sets an exact expected data size for messages coming into the RS-232 auxiliary serial port.

Example

```
rem   Set buffer size to 3.
echo  SIM3 > COM2
rem   Set for continuous input.
echo  SIR1 > COM2
rem   If the string '123456789' is received,
rem   the FMT1000 Series computers will transmit three
rem   separate messages to the CPU:
rem   '123'
rem   '456'
rem   '789'
```

See SIR example code, page 2-40.

SII – Input Buffer Initialize

Format

SII

Summary

Resets the auxiliary serial input buffer to empty.

Returns

N / A

Description

This command discards any data waiting in the RS-232 auxiliary serial port input buffer.

It is useful for being sure the buffer is empty before you expect a device to begin transmitting. Because the input buffer is interrupt-driven, other data may be left over. This obviates the need to continuously read the buffer until it is empty.

Example

```
rem    Initialize aux input buffer.
echo   SII > COM2
```

SIB, RIB – Input Beep Enable

Format

SIB 0 | 1

0=OFF *
1=ON

Summary

Enables or disables the beep when input is received.

This setting is preserved in EEPROM.

Returns

Beep Enable setting, integer 0 or 1.

Description

Allows an audible indication when input is received and accepted by the FMT1000 Series computers.

Example

```
rem    Set aux input beep enable to ON.
echo   SIB1 > COM2
```

SIR, RIR – Set and Return Input Read Mode

Format

SIR 0 | 1

0=Single Read
1=Continuous Reads

Summary

Sets the mode by which input will be accepted from the device attached to the RS-232 auxiliary serial port.

Returns

N / A

Description

Single input mode receives and forwards just one reading to the CPU. To enable reading another input record, the command must be reissued.

There is no factory default for this command, and it will not be reset using the SFD set factory defaults command.

Example

```
rem    Set aux input read mode to single.
echo   SIR0 > COM2
```

RIN – Return Input Status

Format

RIN

Summary

Returns the input status of the communications port.

Returns

Integer, 0-5

0=no error
1=command error
2=parameter error
3=timeout error
4=request was canceled
5=busy

Description

This command returns the current status for input from the serial port. It reflects the current status condition of the port, as opposed to the last error status, so it is important to retrieve status as needed.

Example

```
rem    Return aux input status.
echo RIN > COM2
rem    If the aux port status is OK, this command
rem    will return an '0' to the CPU.
```

Return Code	Error/Status	Description
1	(command error)	indicates that the last command mnemonic was incorrect or inappropriately sequenced.
2	(parameter error)	indicates that the parameter following the command was incorrect or inappropriate. This error can also occur if a required parameter is missing from a command.
3	(timeout error)	occurs if the timeout value is exceeded; see the SIO set input timeout command.
4	(request was canceled)	indicates that the SIC input cancel command was properly received and executed by the I/O processor.
5	(busy)	used when the operation is not complete (e.g., the buffer isn't full or the required prefix and/or suffix hasn't been received) but has not yet timed out.

SOW – Set Output Write

Format

SOW *<string>*
Length: 1-80 characters

Summary

Initiates an output write to the RS-232 auxiliary port.

Returns

N/A

Description

Initiates an output write to the RS-232 auxiliary port.

Example

```
rem    Write a string 'Hi There' terminated
rem    with a <CR><LF>.
rem    Set output suffix to <CR><LF>.
echo   SSO^M^J > COM2
echo   SOW Hi There > COM2
```

SOC – Aux Output Cancel

Format

SOC

Summary

Cancels the current output request.

Returns

N / A

Description

Cancels an outstanding request for output of the string defined by SOW.

This command is useful when ROU returns an error status on output, and can be used as part of an error recovery routine within the program.

Example

```
rem    Cancel aux output.
echo   SOC > COM2
```

SOO, ROO – Set and Return Aux Output Timeout

Format

SOO *<0-255>*

Summary

Set the length to wait for output, in seconds.

This setting is preserved in EEPROM.

Returns

Timeout value, integer 0-255.

Description

Sets the timeout delay before setting the error status 3 (timeout error) for the ROU return output status command.

If the delay is set to 0, the device will, by definition, never time out. This is the factory default.

Example

```
rem   Set aux output timeout to 10 seconds.
echo  SOO10 > COM2
```

ROU – Return Output Status

Format

ROU

Summary

Returns the output status of the RS-232 auxiliary port.

Returns

Integer, 0-5

0=no error
1=command error
2=parameter error
3=timeout error
4=request was canceled
5=busy

Description

This command returns the current status for output to the RS-232 auxiliary port. It reflects the current status condition of the port, as opposed to the last error status, so it is important to retrieve status as needed.

Example

```
rem    Return aux output status.
echo   ROU > COM2
```

Return Code	Error/Status	Description
1	(command error)	indicates that the last command mnemonic was incorrect or inappropriately sequenced.
2	(parameter error)	indicates that the parameter following the command was incorrect or inappropriate. This error can also occur if a required parameter is missing from a command.
3	(timeout error)	occurs if the timeout value is exceeded; see the SOO set output timeout command.
4	(request was canceled)	indicates that the SOC output cancel command was properly received and executed by the I/O processor.
5	(busy)	indicates that the operation is not complete (e.g., the prefix, data, and suffix have not yet been sent) but the timeout has not expired.

SPA, RPA – Set or Return Aux Passthrough Mode

Format

SPA 0 | 1

0=off *
1=on

Summary

Sets the I/O processor passthrough mode for the RS-232 auxiliary port.

Returns

N/A

Description

When passthrough is set to 0 (off), the I/O processor's prefix, suffix, and buffer length filters control the data flow from the serial port. These filters can be set using mnemonic commands.

When passthrough is set to 1 (on), the raw data received from the serial port is sent to the CPU application until the passthrough termination characters are received.

Example

```
rem   Set aux passthrough mode to ON.
echo  SPA1 > COM2
echo  This string will be terminated > COM2
echo  +++ > COM2
rem   Passthrough will terminate when the '+++'
rem   characters are received.
```

STC, RTC – Set and Return Passthrough Termination Characters

Format

STC *<string>*
Length: 3 characters

Summary

Sets the character string which the I/O processor will use to recognize the end of passthrough mode.

Returns

N/A

Description

When the RS-232 auxiliary port is in passthrough mode, the I/O processor still monitors the data input stream. When the passthrough termination characters are encountered, the passthrough mode is changed to 0 (off), and the I/O processor resumes active control over the data stream.

The characters chosen as terminators should be unique and not duplicated in the expected data stream.

The factory default string is +++ (three plus marks).

Example:

```
rem   Set aux passthrough termination characters
rem   to '123'.
echo  STC123 > COM2
```

SAE, RAE – Set and Return Aux Communications Port Enable

Format

SAE 0 | 1

0=Off
1=On *

Summary

Enables or disables transmission of RS-232 auxiliary port responses to the CPU.

This setting is preserved in EEPROM.

Returns

Aux comm port enable flag, integer 0 or 1.

Description

Enables or disables transmission of RS-232 auxiliary port responses to the CPU.

Example

```
rem    Set aux comm port enable to OFF.
echo   SAE0 > COM2
```

Digital I/O and Counter Commands

FMT1000 Series computers have two digital inputs and two digital outputs. These are often used for such applications as access control and machine monitoring.

The first set of commands configure the response format, defined earlier as the format of messages from the coprocessor to the CPU, in this case for digital I/O commands.

Command	Description	Pg.#
SDT, RDT	Set and Return Digital I/O Response Tag	2-54
SDP, RDP	Set and Return Digital I/O Response Prefix	2-55
SDS, RDS	Set and Return Digital I/O Response Suffix	2-56
SED, RED	Set and Return Digital I/O Response AutoEnter Mode	2-57
SRD, RRD	Set and Return Digital I/O Response Path	2-58

The digital inputs may be used in four ways:

- The state of an input may simply be sensed as active or not active.
- A change of state on an input can result in a user-defined string's being sent to the user's application.
- The input may be used as a counter that may be read on demand.
- A counter-match value may be set, resulting in a user-defined string's being sent to the user's application.

The following commands configure the input modes.

Note that *defining* some parameters activates the corresponding feature. For example, setting a Response String puts that input into state change detection mode. Also, note that the inputs are always counting and that reading and resetting the counters can be done at any time, without previously enabling count mode. The counters have a 32-bit capacity, with a 20-Hz bandwidth. No input conditioning is done, so the counters should be driven only with waveforms that have clean, sharp transitions.

Note: The terminology *on* and *off* is used in regard to digital inputs. The actual inputs are optical isolators. *On* means that current is flowing through the isolator, and *off* means that no current is flowing. Also, the input circuit inverts the signal so an input that is on will return a 0 (zero).

Command	Description	Pg.#
RS1, RS2	Return State of Input No. 1 or No. 2	2-60
SR1, SR2	Reset Counter No. 1 or No. 2	2-61
RE1, RE2	Read Counter No. 1 or No. 2	2-62
RR1, RR2	Read and Reset Counter No. 1 or No. 2	2-63
SI1, RI1, SI2, RI2	Set and Return Input No. 1 or No. 2 Response String	2-59
SM1, RM1, SM2, RM2	Set and Return Counter No. 1 or No. 2 Match String	2-64
S1M, R1M, S2M, R2M	Set and Return Counter No. 1 or No. 2 Response String	2-65

Similarly, the digital outputs can be used in two different ways, all controlled by values passed to the same command. They can be activated or not, or they can be set first to activate and then to deactivate a fixed time later. This latter mode is useful for door solenoid control and other such applications, where this capability eliminates the need for the user application to time the activation / deactivation (often difficult under DOS unless nothing else is going on). The digital output control command is:

Command	Description	Pg.#
ST1, RT1, ST2, RT2	Set and Return Momentary Timeout for Output No. 1 and No. 2	2-66

Note: *Activated* and *deactivated* are used in reference to the digital outputs, since they are connected to relays that have both normally open and normally closed contacts. This renders the conventional usage of ON and OFF somewhat confusing. In all cases *activated* means that the relay coil has power applied, and *deactivated* means the inverse.

One other digital I/O command is used:

Command	**Description**	**Pg.#**
SDE, RDE	Set and Return Digital I/O Line Enable	2-67

This command enables or disables the transmission of digital I/O response if the FMT1000 Series computer is programmed for change-of-state detection.

SDT, RDT – Set and Return Digital I/O Response Tag

Format

SDT <*string*>
Length: 1

Summary

Sets a unique ID tag to prefix input from the digital I/O lines.

Returns

Digital I/O tag, 1 character

Description

The I/O processor can route input via several devices. On occasion, input from several physical devices may be routed to the same system input port. For example, both the RS-232 auxiliary port and the digital I/O can send their data to the keyboard buffer.

This tag applies to input automatically generated by the change-of-state function (SI1, SI2) and the counter-match function (SM1, SM2).

This tag is a prefix which is added to the input from a given physical device to identify the source of the input.

Example

```
rem   Set digital I/O response tag to 'D'.
echo  SDTD > COM2
rem   Set input #1 Response String to
rem   'Input #1 Changed'
echo  SI1Input #1 Changed > COM2
rem   If input #1 changes state, the string
rem   'DInput #1 Changed' will be sent
rem   to the CPU.
```

SDP, RDP – Set and Return Digital I/O Response Prefix

Format

SDP *<string>*
Max length: 16

Summary

Sets the string to be prepended before each response containing data from the digital I/O.

Returns

Response Prefix, string of at most 16 bytes.

Description

This I/O processor command allows a standard prefix to be added to the response to a string of characters input from or output to the digital I/O lines.

This applies to input automatically generated by the change-of-state function (SI1, SI2) or the counter-match function (SM1, SM2).

There is no factory default for this command, and it will not be reset using the SFD set factory defaults command.

Example

```
rem    Set digital I/O response prefix
rem    to 'prefix_'.
echo   SDPprefix_ > COM2
rem    Set input #1 response string to
rem    'Input #1 Changed'.
echo   SI1Input #1 Changed > COM2
rem    If input #1 changes state, the string
rem    'prefix_Input #1 Changed' will be
rem    forwarded to the CPU.
```

SDS, RDS – Set and Return Digital I/O Response Suffix

Format

SDS <*string*>
Max length: 16

Summary

Sets the string to be appended to the response to each input from the digital I/O lines.

Returns

Response suffix, string of at most 16 bytes.

Description

This I/O processor command allows a standard suffix to be sent after the response to an input change of state, but before the termination string specified by the autoenter mode, if any.

This suffix applies to input automatically generated by the change-of-state function (SI1, SI2) and the counter-match function (SM1, SM2).

There is no factory default for this command, and it will not be reset using the SFD set factory defaults command.

Example

```
rem    Set digital I/O response suffix
rem    to '_suffix'.
echo   SDS_suffix > COM2
rem    Set input #1 response string to
rem    'Input #1 Changed'.
echo   SI1Input #1 Changed > COM2
rem    If input #1 changes state, the string
rem    'Input #1 Changed_suffix' will be
rem    forwarded to the CPU.
```

SED, RED – Set and Return Digital I/O Response AutoEnter Mode

Format

SED <*0-3*>

0=Off *
1=CR (carriage return)
2=CR/LF (carriage return/line feed)
3=Tab

Summary

Sets the type of terminating characters for data input via the digital I/O lines.

This setting is preserved in EEPROM.

Returns

AutoEnter mode flag, integer 0-3.

Description

This mode sets the character(s) transmitted to the host CPU after the data input from the digital I/O.

This mode applies to input automatically generated by the change-of-state function (SI1, SI2) and the counter-match function (SM1, SM2).

For most ordinary I/O, carriage return would be appropriate, but Tab might be used if entering data into a multifield screen.

Example

```
rem   Set digital I/O response autoenter mode
rem   to <CR>.
echo  SED1 > COM2
```

SRD, RRD – Set and Return Digital I/O Response Path

Format

SRD 0 | 1

0=Keyboard *
1=COM2 host port

Summary

Sets the digital I/O input path to the keyboard or the host serial port COM2.

This setting is preserved in EEPROM.

Returns

Response path flag, integer 0 or 1.

Description

The data from the digital I/O can be used to simulate input from the keyboard, or it can be transmitted to the host communications port COM2.

This path applies to input automatically generated by the change-of-state function (SI1, SI2) and the counter-match function (SM1, SM2).

The advantage to the keyboard input is that no special programming is required to receive the data. However, keyboard input is slower due to the limitations of keyboard input speed.

The factory default is keyboard simulation input.

Example

```
rem    Set digital I/O response path
rem    to COM2 host port.
echo   SRD1 > COM2
```

SI1, RI1, SI2, RI2 – Set and Return Input No. 1 or No. 2 Response String

Format

SI1 <*string*>
SI2 <*string*>
RI1
RI2
Max String Length: 80

Summary

Sets or returns the string sent to the CPU when an OFF to ON change of state occurs on a digital input.

Returns

The response string, if any.

Description

This command is used to set a response string that is sent to the CPU, with normal response format processing, in response to an OFF to ON change of state of the designated input. This may be used to detect such events as door openings without having to continuously monitor the state of the input.

> **Note:** Setting the response string enables the change-of-state detection mode; to disable it, define the response string as empty. Also, note that only a change from off to on is reported.

Example

```
rem   Set input #1 response string.
echo  SI1DOOR OPEN! > COM2
rem   If input #1 changes state, the string:
rem   'DOOR OPEN!' will be forwarded to the CPU.
```

RS1, RS2 – Return State of Input No. 1 or No. 2

Format

RS1
RS2

Summary

Returns the binary state of digital input No. 1 or No. 2.

Returns

String, 0 or 1

1=Off, no current flowing through opto-isolation.
0=On, current is flowing through opto-isolation.

Description

This command is used to read the current state of digital input No. 1 or No. 2.

Example

```
rem    Read input #1.
echo   RS1 > COM2
rem    The return string will be '0' or '1'.
```

SR1, SR2 – Reset Counter No. 1 or No. 2

Format:

SR1
SR2

Summary

Resets to 0 the value of the counter for input line No. 1 or No. 2.

Returns

N/A

Description

This command sets the value of the counter to 0. The counter is unsigned, 32 bit.

Example

```
rem    Reset counter #1 to 0.
echo   SR1 > COM2
```

RE1, RE2 – Read Counter No. 1 or No. 2

Format

RE1
RE2

Summary

Returns the value of the counter for input line No. 1 or No. 2.

Returns

An ASCII string representing the current value of the internal counter as decimal digits.

Description

This command retrieves the value of the counter for input lines No. 1 or No. 2. The counter is unsigned, 32 bit.

Example

```
rem    Read counter #1.
echo   RE1 > COM2
```

RR1, RR2 – Read and Reset Counter No. 1 or No. 2

Format

RR1
RR2

Summary

Returns the value of the input counter for line No. 1 or No. 2 and resets that counter to 0.

Returns

An ASCII string representing the current value of the internal counter as decimal digits.

Description

This command returns the current value of the counter for digital input lines No. 1 or No. 2, and then resets the value of the counter to 0. The counter is unsigned, 32 bit.

Example

```
rem    Read and reset counter #1.
echo   RR1 > COM2
```

SM1, SM2, RM1, RM2 – Set and Return Counter No. 1 or No. 2 Value Match

Format

SM1*<0-1,000,000>*
SM2*<0-1,000,000>*
RM1
RM2

0=Off
1-1,000,000=Count to match

Summary

Sets counter-match value.

Returns

Counter-match value, 0-1,000,000.

Description

This command set sa target value for a counter. When the value is reached, a response string is sent to the CPU via the digital I/O response path. Also see S1M and S2M, page 85.

Example

```
rem   Set counter #1 match value=3000.
echo  SM13000 > COM2
rem   Set counter #1 match response string
rem   to 'Count reached'.
echo  S1M count reached > COM2
rem   When counter #1 reaches 3000, the string
rem   'Count reached' will be sent to the CPU.
```

S1M, R1M, S2M, R2M – Set and Return Counter No. 1 or No. 2 Response String

Format

S1M<*string*>
S2M<*string*>
R1M
R2M
Max string length: 80

Summary

Sets counter response string.

Returns

Counter-match response string, string 0-80 characters.

Description

This command sets a response string that is sent to the CPU when a target counter-match value is reached. The response is sent via the digital I/O response path.

Also see SM1 and SM2, page 2-64.

Example

```
rem   Set counter-match value=50.
echo  SM150 > COM2
rem   Set counter #1 match response string
rem   to 'Count reached'.
echo  S1M count reached > COM2
rem   When counter #1 reaches 50, the string
rem   'Count reached' will be sent to the CPU.
```

ST1, RT1, ST2, RT2 – Set and Return Momentary Timeout for Output No. 1 and No. 2

Format

ST1 *<0-999>*

0=Off
999=On
1-998=Time value in tenths of a second

Summary

Activates, deactivates, or temporarily activates a digital output.

Returns

Timeout value, integer 0-999.

Description

This command can be used to activate, deactivate, or temporarily activate a digital output. Temporary activation, known as momentary mode, activates the digital output for from 0.1 to 99.8 seconds. This mode is useful for door solenoid control and other such applications where the output can be activated for a period of time to allow entry without the CPU having to provide timing.

Example

```
rem    Activate output #1 for 3 seconds.
echo   ST130 > COM2
```

SDE, RDE – Set and Return Digital I/O Response Enable

Format

SDE 0 | 1

0=Off
1=On *

Summary

Enables or disables the response to changes of state on digital inputs.

This setting is preserved in EEPROM.

Returns

Digital I/O line enable setting, integer 0 or 1.

Description

Enables or disables the response to changes of state on digital inputs. All other messages, such as the responses to commands to read the state of an input, or a counter, are unaffected.

Example

```
rem   Set change of state response to ON.
echo  SDE1 > COM2
```

Bar Code and Wand Control Commands

These commands control the peripheral inputs to the I/O coprocessor. These inputs can connect one or two bar code devices, or a bar code device and a magnetic stripe reader. Note that there are no specific commands to control the magnetic stripe reader.

The first set of commands configure the response format, defined earlier as the format of messages from the coprocessor to the CPU, in this case for peripheral input commands.

Command	Description	Pg.#
SP5, RP5	Set and Return Peripheral (5 pin connector) Tag	2-70
SP9, RP9	Set and Return Peripheral (9 pin connector) Tag	2-70
SPT, RPT	Set and Return Peripheral Response Tag	2-71
SPP, RPP	Set and Return Peripheral Response Prefix	2-72
SPS, RPS	Set and Return Peripheral Response Suffix	2-73
SEP, REP	Set and Return Peripheral Response AutoEnter Mode	2-74
SRP, RRP	Set and Return Peripheral Port Response Path	2-75

The next set of commands enables or disables the various bar code formats that the coprocessor can recognize. These formats are often referred to as symbologies. It should be noted that if more than one format is enabled, the coprocessor will attempt to autodiscriminate among them, trying each format in turn to see which one correctly decodes. Only enabled symbologies will be tried.

Command	Description	Pg.#
S39, R39	Set and Return Code 3 of 9 Mode	2-76
S25, R25	Set and Return Interleave 2 of 5 Mode	2-77
SCB, RCB	Set and Return Codabar Mode	2-78
SUP, RUP	Set and Return UPC Mode	2-79
S28, R28	Set and Return Code 128 Mode	2-80

The last set of commands sets various control parameters for peripheral input.

Command	**Description**	**Pg.#**
SSC, RSC	Set and Return Concatenate Mode	2-81
SSM, RSM	Set and Return String Match Length	2-82
SSB, RSB	Set and Return Read Beep Enable	2-83
SPE, RPE	Set and Return Peripheral Port Enable	2-84

SP5, RP5, SP9, RP9 – Set and Return Peripheral (5-Pin Connector) Tag and (9-Pin Connector) Tag

Format

SP5*<string>*
SP9*<string>*
RP5
RP9
Length: 1

Summary

Sets a unique ID tag to prefix bar code data from each connector.

Returns

Peripheral connector tag, 1 character

Description

This command set a tag which is added to the input from either bar code connector to identify the source of the input. When set, this tag will replace the tag set by the SPT (Set Peripheral Response Tag) command.

Example

```
rem   Set 5 pin connector tag to '5'.
echo  SP55 > COM2
rem   Set 8 pin connector tag to '9'.
echo  SP99 > COM2
rem   If the bar code “ABCDE” is read through
rem   the 5 pin connector, the string '5ABCDE'
rem   will be sent to the CPU.
```

SPT, RPT – Set and Return Peripheral Response Tag

Format

SPT *<string>*
Length: 1

Summary

Sets a unique ID tag to prefix all input from the peripheral communications port.

Returns

Peripheral tag string, 1 character.

Description

The I/O processor can route input via several devices. On occasion, input from several physical devices may be routed to the same system input port. For example, both the RS-232 auxiliary port and peripheral ports can send their data to the keyboard buffer.

This tag is a prefix which is added to the input from a given physical device to identify the source of the input.

Example

```
rem   Set peripheral response tag to 'P'.
echo  SPTP > COM2
```

SPP, RPP – Set and Return Peripheral Response Prefix

Format

SPP *<string>*
Max length: 16

Summary

Sets the string to be prepended before each response containing data from a peripheral port.

Returns

Response prefix, string of at most 16 bytes.

Description

This I/O processor command allows a standard prefix to be added to the response to a string of bar code or magstripe characters read at the peripheral port.

There is no factory default for this command, and it will not be reset using the SFD set factory defaults command.

Example

```
rem    Set peripheral response prefix to 'BAR'.
echo   SPPBAR > COM2
rem    If the bar code (or magstripe) symbol
rem    '13579' is scanned, the string 'BAR13579'
rem    will be forwarded to the CPU.
```

SPS, RPS – Set and Return Peripheral Response Suffix

Format

SPS *<string>*
Max length: 16

Summary

Sets the string to be appended to the response to each input from the peripheral port.

Returns

Response suffix, string of at most 16 bytes.

Description

This I/O processor command allows a standard suffix to be sent after the response to each line of input, but before the terminating character(s) set by the state of the autoenter mode.

There is no factory default for this command, and it will not be reset using the SFD set factory defaults command.

Example

```
rem   Set peripheral response suffix to 'BAR'.
echo  SPSBAR > COM2
rem   If the bar code (or magstripe) symbol
rem   '13579' is scanned, the string '13579BAR'
rem   will be forwarded to the CPU.
```

SEP, REP – Set and Return Peripheral Response AutoEnter Mode

Format

SEP *<0-3>*

0=Off *
1=CR (carriage return)
2=CR/LF (carriage return/line feed)
3=Tab

Summary

Sets the type of terminating characters for data input via the peripheral port.

This setting is preserved in EEPROM.

Returns

AutoEnter mode setting, integer 0-3.

Description

This mode sets the character(s) transmitted to the host CPU after the data input from the peripheral port.

For most ordinary I/O, carriage return is appropriate, but Tab may be used if entering data into a multifield screen.

Example

```
rem    Set peripheral response autoenter
rem    mode to <CR>.
echo   SEP1 > COM2
```

SRP, RRP – Set and Return Peripheral Port Response Path

Format

SRP 0 | 1

0=Keyboard *
1=COM2 host port

Summary

Sets the peripheral port input path to the keyboard or the host serial port COM2.

This setting is preserved in EEPROM.

Returns

Response path flag, integer 0 or 1.

Description

The data from the peripheral port can be used to simulate input from the keyboard, or it can be transmitted to the host communications port COM2.

The advantage to the keyboard input is that no special programming is required to receive the data. However, keyboard input is slower due to the limitations of keyboard input speed.

The factory default is keyboard simulation input.

Example

```
rem   Set peripheral response path to COM2.
echo  SRP1 > COM2
```

S39, R39 – Set and Return Code 3 of 9 Mode

Format

S39 *<0-3>*

0=Off
1=Code 3 of 9 *
2=Code 3 of 9 Mod 43
3=Full ASCII code 3 of 9

Summary

Enable and set the type of Code 3 of 9 bar code to decode.

This setting is preserved in EEPROM.

Returns

Code 3 of 9 setting, integer 0-3.

Description

Enables or disables the decoding of Code 39 and sets the variant of the symbology to be decoded if enabled.

> **Note:** The decoder must be informed in advance if checksums or full ASCII are to be used, because these variants are formed using the standard 3 of 9 character set.

The factory default is standard Code 3 of 9 (parameter 1).

Example

```
rem   Set code 3 of 9 mode to Code 3 of 9.
echo  S391 > COM2
```

S25, R25 – Set and Return Interleave 2 of 5 Mode

Format

S25 0 | 1

0=Off
1=On *

Summary

Enable or disable bar code type Interleave 2 of 5.

This setting is preserved in EEPROM.

Returns

Interleave 2 of 5 enable setting, 0 or 1.

Description

Enables or disables the decoding of bar code type Interleave 2 of 5.

The factory default is ON.

Example

```
rem   Set interleaved 2 of 5 code mode to ON.
echo  S251 > COM2
```

SCB, RCB – Set and Return Codabar Mode

Format

SCB 0 | 1

0=Off
1=On *

Summary

Enables or disables the decoding of bar code type CODABAR.

This setting is preserved in EEPROM.

Returns

Codabar enable setting, string 0 or 1.

Description

Enables or disables the decoding of bar code type CODABAR.

The factory default is ON.

Example

```
rem    Set Codabar Mode to OFF
echo   SCB0 > COM2
```

SUP, RUP – Set and Return UPC Mode

Format

SUP 0 | 1

0=Off
1=On *

Summary

Enables or disables the decoding of bar code type UPC.

This setting is preserved in EEPROM.

Returns

UPC enable setting, string 0 or 1.

Description

This command enables or disables the decoding of Universal Product Code (UPC) bar codes. This includes the decoding of EAN codes. All true UPC codes will be 13 digit numbers, prefixed by *0* (for U.S. country code). EAN codes will be prefixed by the appropriate country code.

Five-digit UPC-E codes will also be recognized and transmitted.

The factory default is ON.

Example

```
rem   Set UPC mode to ON.
echo  SUP1 > COM2
```

S28, R28 – Set and Return Code 128 Mode

Format

S28 0 | 1

0=Off
1=On *

Summary

Enables or disables the decoding of bar code type Code 128.

This setting is preserved in EEPROM.

Returns

Code 128 enable setting, string 0 or 1.

Description

Enables or disables the decoding of bar code type Code 128. All three code sets are supported.

The factory default is ON.

Example

```
rem    Set Code 128 mode to ON.
echo   S281 > COM2
```

SSC, RSC – Set and Return Concatenate Mode

Format

SSC 0 | 1

0=Off *
1=On

Summary

Selects whether to concatenate bar codes or not.

This setting is preserved in EEPROM.

Returns

Mode flag, string 0 or 1.

Description

When this mode is enabled, all bar codes which have a leading space are sent without the space and the peripheral response tag, prefix, suffix, and autoenter strings are not added to the response data.

This may be useful when using bar coded numeric entry keypads or similar constructs to emulate keypad input.

The factory default for this mode is OFF.

Example

```
rem    Set concatenate mode ON.
echo   SSC1 > COM2
```

SSM, RSM – Set and Return String Match Length

Format

SSM *<0-80>*

0=Off (no length matching)
Min length for matching: 1
Max length: 80

Summary

Sets the length against which input bar codes are matched.

This setting is preserved in EEPROM.

Returns

Match length, string 0-80.

Description

Sets a length against which the input bar code is matched. If it does not match, the bar code input is not accepted or processed. Certain codes, like Interleave 2 of 5, are more reliable when the expected input length can be predicted and checked for. However, this setting affects all bar code inputs and is thus inappropriate when more than one length of code is required, even if two different symbologies are used.

The factory default is *0*, for no length comparison.

Example

```
rem    Set string match length to 8.
echo   SSM8 > COM2
```

SSB, RSB – Set and Return Read Beep Enable

Format

SSB 0 | 1

0=Off
1=On*

Summary

Enables or disables a beep upon successful decoding of a bar code.

This setting is preserved in EEPROM.

Returns

Beep status setting, string 0 or 1.

Description

When a bar code has been successfully scanned, by default the coprocessor beeps indicating a successful read. However, if no beep is preferred, it can be disabled with this command.

Example

```
rem   Set read beep enable to ON.
echo  SSB1 > COM2
```

SPE, RPE – Set and Return Peripheral Port Enable

Format

SPE 0 | 1

0=Off
1=On *

Summary

Enables or disables transmission peripheral port responses.

This setting is preserved in EEPROM.

Returns

Peripheral port enable setting, string 0 or 1.

Description

Enables or disables transmission peripheral port responses from bar codes or magstripe input. All peripheral port commands that elicit a response are unaffected.

Example

```
rem   Set peripheral port enable to ON.
echo  SPE1 > COM2
```

System Commands

The system commands include commands for controlling the keyboard and display and various other internal functions.

The first set of commands configure the response format, defined earlier as the format of messages from the coprocessor to the CPU, in this case for system commands, also referred to as internal commands. Note that this format is used for all replies to CPU commands that come from the coprocessor, regardless of the internal subsystem to which they apply.

Command	Description	Pg.#
SIT, RIT	Set and Return Internal Command Response Tag	2-87
SIP, RIP	Set and Return Internal Command Response Prefix	2-88
SIS, RIS	Set and Return Internal Command Response Suffix	2-89
SEI, REI	Set and Return Internal Command Response AutoEnter Mode	2-90
SRI, RRI	Set and Return Internal Command Response Path	2-91

The following commands control display characteristics:

Command	Description	Pg.#
SBL, RBL	Set Backlight	2-92
SVA, RVA	Set and Return Viewing Angle	2-93

The following commands control keyboard operation and configuration for various versions of the FMT 1000 Series computer.

Command	Description	Pg.#
SAD, RAD	Set and Return Keyboard Auto Detect	2-94
SKH, RKH	Set and Return Keyboard Hardware Reset Enable	2-95
SKE, RKE	Set and Return Keyboard Enable Mode	2-96
SKT, RKT	Set and Return Keyboard Type	2-97

SKR, RKR	Set and Return Keyboard Repeat	2-98
SKC, RKC	Set and Return Keyboard Click	2-99
SCD, RCD	Set and Return Keyboard Intercharacter Delay	2-100
SnD, SnU RnD, RnU	Set and Return User Defined Key Up and Down Scan Codes (n=1, 2, 3, or 4)	2-101

The following are miscellaneous other commands:

Command	**Description**	**Pg.#**
RVR	Return Firmware Version	2-103
SFD	Reset to Factory Defaults	2-104
SEE	Write Setup to EEPROM	2-105
RER	Return Error Code	2-106
SBP	Set Beeper Tones	2-108
SSS, RSS	Set and Return Startup Message	2-109
RES	Echo String Via Current Internal Response Path	2-110
ROB	Return State of Battery Sense	2-111
SBE, RBE	Set and Return Battery Sense Enable	2-112
SBS, RBS	Set and Return Switch to Battery String	2-113

SIT, RIT – Set and Return Internal Command Response Tag

Format

SIT *<string>*
Length: 1

Summary

Sets a unique ID tag to prefix all input from the I/O processor in response to an internal command.

Returns

Aux tag string, 1 character.

Description

The I/O processor can route input via several devices. On occasion, input from several physical devices may be routed to the same system input port. For example, both the RS-232 auxiliary port and peripheral ports can send their data to the keyboard buffer.

This tag is a prefix which is added to the input from a given physical device or data source to identify the source of the input.

Example

```
rem   Set internal command response tag to 'I'.
echo  SITI > COM2
rem   If the data '10' is returned as a response
rem   to an internal command, the string 'I10'
rem   will be forwarded to the CPU.
```

SIP, RIP – Set and Return Internal Command Response Prefix

Format

SIP *<string>*
Max length: 16

Summary

Sets the string to be prepended before each response to an internal command

Returns

Response prefix, string of at most 16 bytes.

Description

This I/O processor command allows a standard prefix to be added to the response to internal command mnemonics.

There is no factory default for this command, and it will not be reset using the SFD set factory defaults command.

Example

```
rem    Set internal command response prefix
rem    to 'INT'.
echo   SIPINT > COM2
rem    If the data '10' is returned as a response
rem    to an internal command, the string 'INT10'
rem    will be forwarded to the CPU.
```

SIS, RIS – Set and Return Internal Command Response Suffix

Format

SIS *<string>*
Max length: 16

Summary

Sets the string to be appended to the response to each internal command.

Returns

Response suffix, string of at most 16 bytes.

Description

This I/O processor command allows a standard suffix to be sent after the response to a command mnemonic, but before the terminating character set by the state of the autoenter mode.

There is no factory default for this command, and it will not be reset using the SFD set factory defaults command.

Example

```
rem    Set internal command response suffix
rem    to '_INT'.
echo   SIS_INT > COM2
rem    If the data '10' is returned as a response
rem    to an internal command, the string '10_INT'
rem    will be forwarded to the CPU.
```

SEI, REI – Set Internal Command Response AutoEnter Mode

Format

SEI *<0-3>*

0=Off *
1=CR (carriage return)
2=CR / LF (carriage return / line feed)
3=Tab

Summary

Sets the type of terminating characters for internal commands.

This setting is preserved in EEPROM.

Returns

AutoEnter mode flag, string 0-3.

Description

This mode sets the character transmitted to the host CPU after the data input from internal commands.

For most ordinary I / O, carriage return is appropriate, but Tab might be used if entering data into a multifield screen.

Example

```
rem    Set internal command response autoenter
rem    mode to <CR>.
echo   SEI1 > COM2
```

SRI, RRI – Set and Return Internal Command Response Path

Format

SRI 0 | 1

0=Keyboard *
1=COM2 host port

Summary

Sets the internal command response path to the keyboard or the host serial port COM2.

This setting is preserved in EEPROM.

Returns

Response path flag, string 0 or 1.

Description

The data from internal commands can be used to simulate input from the keyboard, or it can be transmitted to the host communications port COM2.

The advantage to the keyboard input is that no special programming is required to receive the data. However, keyboard input is slower due to the limitations of keyboard input speed.

The factory default is keyboard simulation input.

Example

```
rem   Set internal command response path to COM2.
echo  SRI1 > COM2
```

SBL, RBL – Set Backlight

Format

SBL *<0-2>*

0 = Off
1 = On
2 = Automatic *

Summary

Sets the LCD display backlight mode.

This setting is preserved in EEPROM.

Returns

Backlight mode setting, string 0, 1, or 2.

Description

Sets the mode of backlighting for the LCD display. Automatic mode turns the backlight on in response to keyboard activity and off after 10 minutes of no keyboard activity. Because backlights have a finite lifetime, automatic and off are the usual choices.

Example

```
rem    Turn Backlight ON.
echo   SBL1 > COM2
```

SVA, RVA – Set and Return Viewing Angle

Format

SVA *<0-100>*

Summary

Sets the viewing angle for the LCD display.

This setting is preserved in EEPROM.

Returns

Viewing angle, integer 0-100.

Description

The SVA command allows customizing of the viewing angle for the LCD display on the FMT1000 Series computer. The defaults are set as a result of issuing a SKT set keyboard type command; however, SKT can be overridden by SVA.

The factory default angles are as follows:

FMT1020	30
FMT1060	30
FMT1040	50

Example

```
rem   Set viewing angle to '30'.
echo  SVA30 > COM2
```

SAD, RAD – Set and Return Keyboard Auto Detect

Format

SAD 0 | 1

0=disable auto detect
1=enable auto detect*

Summary

Enables or disables keyboard auto detection feature.

This setting is preserved in EEPROM.

Returns

Keyboard auto detect setting, integer 0 or 1.

Description

This feature, when enabled, allows the FMT1000 Series computer to detect which of the five keyboard configurations is being used.

To change keyboard type, install the new keyboard and apply power to the FMT1000 Series computer. Three beeps are issued during startup. Immediately after the three beeps, press the ***3*** key.

One more beep sounds to acknowledge the keyboard change and the new setting is preserved in EEPROM.

Example

```
rem    Set keyboard auto detect to enabled.
echo   SAD1 > COM2
rem    Save setting in EEPROM.
echo   SEE > COM2
rem    If the LANpoint is now reset, and
rem    the '3' key is pressed while the
rem    computer is initializing, the keyboard
rem    will be recognized and set properly.
```

SKH, RKH – Set and Return Keyboard Hardware Reset Enable

Format

SKH 0 | 1

0=Off
1=On *

Summary

This key combination enables or disables system hard resets **<ctrl+alt+R>** from the keyboard.

This setting is preserved in EEPROM.

Returns

Reset enable setting, integer 0 or 1

Description

While the well known **<ctrl+alt+delete>** key combination will software reset a DOS computer and cause it to reboot, at times a hard, hardware reset is required. If this mode is enabled, the key combination **<ctrl+alt+R>** (character **R**) will cause a hardware reset of both the main CPU and the coprocessor, very similar to a power-on.

For keyboards without the necessary keys, a user-defined key sequence may be used.

Example

```
rem    Enable keyboard hard resets
echo   SKH1 > COM2
```

SKE, RKE – Set and Return Keyboard Enable Mode

Format

SKE 0 | 1

0=Off
1=On *

Summary

Enables or disables the keyboard.

This setting is preserved in EEPROM.

Returns

Keyboard enable setting, string 0 or 1.

Description

Allows the keyboard to be disabled. Input from other devices will still be accepted. Note that the **<ctrl+alt+R>** (character **R**) hardware reset sequence will still cause a reset if enabled, regardless of the setting of the keyboard enable.

Example

```
rem    Set keyboard enable mode to ON.
echo   SKE1 > COM2
```

SKT, RKT – Set and Return Keyboard Type

Format

SKT *<0-4>*

0=FMT1020 QWERTY *
1=FMT1020 Alpha-numeric
2=FMT1020 Numeric
3=FMT1060
4=FMT1040

Summary

Sets the type of keyboard which is installed in the system.

This setting is preserved in EEPROM.

Returns

Keyboard type, string 0-4.

Description

Sets the type of keyboard that is installed in the FMT1000 Series computer. This command is used when the keyboard is changed.

Example

```
rem    Set keyboard type to FMT1040
echo   SKT4 > COM2
```

SKR, RKR – Set and Return Keyboard Repeat

Format

SKR 0 | 1

0=Off
1=On *

Summary

Enables or disables key repeat for the keyboard.

This setting is preserved in EEPROM.

Returns

Key repeat enable, string 0 or 1.

Description

When this setting is OFF, pressing a key will generate only one character or function. When it is on, the key starts repeating after it has been pressed and held for one second, and then continues repeating at a rate of five per second.

Example

```
rem    Set keyboard repeat ON.
echo   SKR1 > COM2
```

SKC, RKC – Set and Return Keyboard Click

Format

SKC 0 | 1

0=Disabled
1=Enabled*

Summary

Enables or disables the click speaker sound on keypress.

This setting is preserved in EEPROM.

Returns

Click setting, string 0 or 1.

Description

This command controls the sound generated when a key is pressed on the keyboard.

Example

```
rem    Set keyboard click ON.
echo   SKC1 > COM2
```

SCD, RCD – Set and Return Keyboard Intercharacter Delay

Format

SCD *<0-255>*

Summary

Sets the delay in tens of milliseconds between each key generated.

This setting is preserved in EEPROM.

Returns

Intercharacter delay, integer 0-255.

Description

This command sets the amount of time which will elapse between each key code output from the coprocessor to the CPU keyboard port, or between the characters sent to the keyboard port by those commands which simulate keyboard input.

The factory default is 2, or 20 milliseconds.

If an alternate keyboard handler is installed, or an application intercepts the keyboard, it may be necessary to lengthen this delay, in particular when automatic input is coming from a bar code, or from the RS-232 auxiliary port.

Example

```
rem    Set keyboard intercharacter delay
rem    to 100 msec.
echo   SCD10 > COM2
```

SnD, SnU – Set User Defined Key Up and Down Scan Codes
RnD, RnU – Return User Defined Key Up and Down Scan Codes

Note: *n* = 1, 2, 3, or 4

Format

SnD *<string>*
SnU *<string>*
RnD
RnU

Summary

Sets the scan codes sent in response to pressing and releasing the user-defined keys or UDKs.

This setting is preserved in EEPROM.

Returns

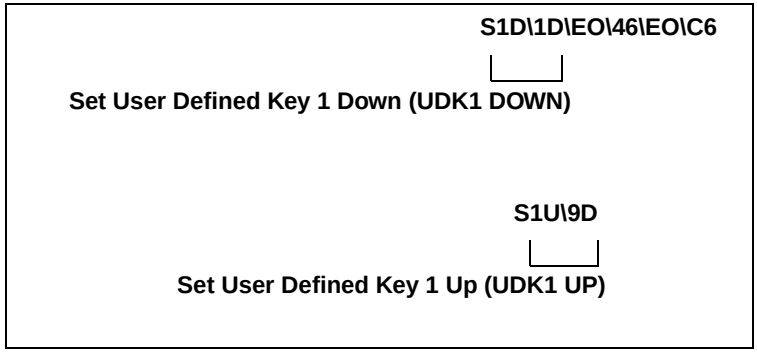
Description

These functions allow the user to define keys to do any special key sequence that cannot be generated by the keyboard. Due to space limitations, it is not possible to implement the full 83-key keyboard on FMT1000 Series computers; missing sequences that are required for an application can be generated using UDKs.

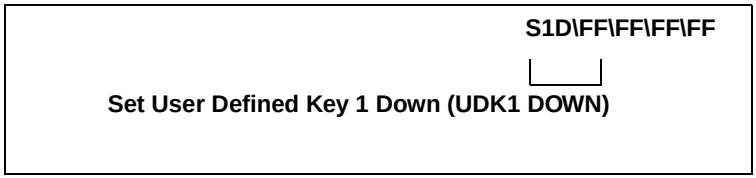
Note: Because keyboard scan codes are used, virtually any keyboard action can be duplicated. Each UDK can hold 8 up and 8 down scan codes. Scan codes are input as 2 hex digits, preceded by a backslash (\).

Example (SnD and SnU):

To implement **<control><break>** on UDK1, the following commands would be used:



Implementing hardware reset is a special case. To define a UDK to be hardware reset use:



RVR – Return Firmware Version

Format

RVR

Summary

Returns the current firmware revision level of the I/O coprocessor

Returns

A string of the format *I/O Coprocessor v XX.YY.*

Description

Returns information pertaining to the revision levels of the firmware in the I/O coprocessor board.

XX denotes the major version number, and *YY* designates the revision level within the version.

Example

At the DOS prompt, type:

```
C:\>echo rvr > COM2
```

The following will be returned:

```
C:\>I/O Coprocessor v 01.01
```

SFD – Reset to Factory Defaults

Format

SFD

Summary

Resets I/O configuration parameters to their factory defaults.

Returns

Single beep if successful.

Description

This command resets most configuration parameters to factory defaults. If a parameter does not have a factory default, it is not reset. These parameters are noted in this manual.

This command does not reset the keyboard type, as set by the SKT command, or the viewing angle.

Example

```
rem    Reset to factory defaults.
echo   SFD > COM2
```

SEE – Write Setup to EEPROM

Format

SEE

Summary

Writes the I/O configuration data to the EEPROM.

Returns

Single beep if successful.

Description

Once changes have been made and verified in the I/O processor configuration, they can be written to the EEPROM so that the FMT1000 Series computer will remember them every time it is powered up.

Example

```
rem   Write configuration data to EEPROM.
echo  SEE > COM2
```

RER – Return Error Code

Format

RER

Summary

Returns an error code relating to the last command.

Returns

Error code, integer 0-5

0=no error
1=command error
2=parameter error
3=timeout error
4=request was canceled
5=busy

Description

This command returns an error code relating to the last command executed. It allows sync with I/O processing. RER may be performed after each command.

Return Code	Error/Status	Description
1	(command error)	indicates that the last command mnemonic was incorrect or inappropriately sequenced.
2	(parameter error)	indicates that the parameter following the command was incorrect or inappropriate. This error can also occur if a required parameter is missing from a command.
3	(timeout error)	occurs if the timeout value is exceeded; see the SOO set output timeout or SIO set input commands.
4	(request canceled)	indicates that the SOC output cancel or SIC input cancel command was properly received and executed by the I/O processor.
5	(busy)	used when the auxiliary I/O device has not completed a transfer but has also not timed out.

SBP – Set Beeper Tones

Format

SBP *<0-5>*

0=high pitch
1=high, low pitches
2=high, high pitches
3=high, low, high pitches
4=high, high, high pitches
5=tick sound

Summary

Allows different beep tones to be generated by the internal beeper.

Returns

N / A

Description

This command allows the user to easily generate a variety of beep patterns for application specific use. One beep pattern sounds each time the command is executed.

Example

```
rem    Cause beep: high, low, high pitches.
echo   SBP3 > COM2
```

SSS, RSS – Set and Return Startup String

Format

SSS *<string>*
Max length: 16

Summary

Sets a string that may be recalled upon demand to execute a terminal specific command on startup. This setting is preserved in EEPROM.

Returns

Startup message, string no more than 16 characters.

Description

This string is provided to allow a simple method for each FMT1000 Series computer to take a custom action on booting. In general the technique is as follows:

1. For each different power on action, write a batch file or other program to invoke it, and store it in the logon directory on the network.
2. On power up, include commands that
   a. make the internal response path the keyboard
   b. recall the startup string.
3. Store the name of the batch file (or other program) as the startup string (remember to include a carriage return or set autoenter mode appropriately).

On boot, the startup string is recalled and executed as though typed, and the user-written program or batch file executes.

The factory default is null (no message).

Example

```
rem   Set startup string to 'INIT'.
echo  SSSINIT > COM2
```

RES – Echo String Via Current Internal Response Path

Format

RES *<string>*
Max length: 80

Summary

Echoes the string to current internal response path.

Returns

<string>

Description

Used for diagnostic purposes, to check out coprocessor communications.

Example

```
echo   RES This is a test. > COM2
rem    'This is a test.' will be sent to
rem    the CPU via the internal path.
```

ROB – Return State of Battery Sense

Format

ROB

Summary

Returns an integer that indicates on-battery operation.

Returns

Battery state, integer 0 or 1

0=Voltage input is 12VDC supply
1=Voltage input is battery pack

Description

This command is used to read the battery sense input. When the power input is lost, and the battery pack takes over, this input will switch from 0 to 1.

Example

```
rem    Read battery sense bit.
echo   ROB > COM2
rem    The return string will be '0' or '1'.
```

SBE, RBE – Set and Return Battery Sense Enable

Format

SBE 0 | 1

0=disable
1=enable

Summary

Enables or disables automatic battery power sense.

Returns

Battery sense enable, 0 or 1.

Description

This command enables the automatic battery power sense feature. This feature senses the switchover to battery power within 20 seconds of primary power loss. A string is sent to the CPU via the internal path to alert the application of the condition.

Also see the SBS command, page 2-113.

Example

```
rem   Enable battery sense.
echo  SBE1 > COM2
rem   Set switched to battery string.
echo  SBS Power_Failed! > COM2
rem   If primary power is lost and the battery
rem   pack is installed, the string
rem   'Power_Failed!' will be sent to the CPU.
```

SBS, RBS – Set and Return Switched to Battery String

Format

SBS<*string*>
Maximum string length: 16 characters

Summary

Set switched to battery string.

Returns

Switched to battery string 0-16 characters.

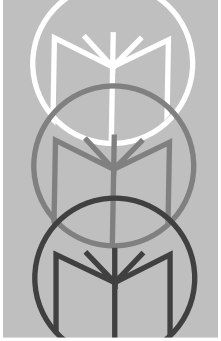
Description

This command sets the switched to battery string used by the automatic battery power sense feature. This feature senses the switchover to battery power within 20 seconds of primary power loss. The string is sent to the CPU via the internal path to alert the application of the condition.

Also see SBE command, page 2-112.

Example

```
rem   Enable battery sense.
echo  SBE1 > COM2
rem   Set switched to battery string.
echo  SBS Power_Failed! > COM2
rem   If primary power is lost and the battery
rem   pack is installed, the string
rem   'Power_Failed!' will be sent to the CPU.
```



Appendix A
Sample Applications

Sample applications are included on the Network Drivers and Utilities diskette provided with the FMT1000 Series computer. All the applications were written in QBasic. QBasic is included in MS DOS 5.0. These applications are meant to be samples. They provide some very simple methods of accessing the various features of FMT1000 Series computers.

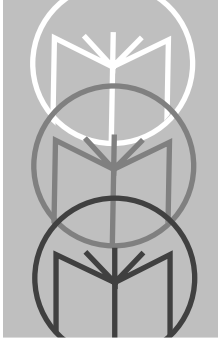
Some of the samples adjust the factory default settings. To reset the unit to factory defaults, send the command **`echo SFD > COM2`** at the DOS prompt or reboot the FMT1000 Series computer to remove the changes made.

Here are brief descriptions of some of the application samples:

1. *Setting the User-Definable Keys (UDK.BAS):* The FMT1000 Series computer has four user-definable keys, denoted on the keyboard as UDK1 through UDK4, which can be programmed with a specific key sequence. The scan codes, which are included in Appendix B, are the standard XT codes. This sample program takes you through the setting of the UDKs.
2. *Access Control and Time and Attendance (TIMEDEMO.BAS):* This application demonstrates a simple time and attendance application. The FMT1000 Series computer can read in numeric data, e.g., a badge number. This data can be on a bar code or keypunched. It then accesses a file to see if there is an employee name associated with the number. If it finds an employee name, it closes the access control relays and opens the door. If it does not, it asks if you want to add a name to the data file and then open the door. To demonstrate the relays, an LED (or other light source) may be turned on by the current supplied by the closing of the relays.
3. *Enable or Disable a Bar Code Type (BARDEMO.BAS):* Many applications require the reading of only one specific type of bar code. This application takes you through the process of enabling and disabling different bar code symbologies supported by the FMT1000 Series computer.

4. *Interfacing with RS-232 Devices through COM1 (COM1DEMO.BAS):* This application shows how to communicate with an RS-232 device connected to COMI. The device we selected is one of our OEM microterminals. For details about the RS-232 auxiliary port please refer to the Chapter 2 of this manual.

   To run this program, you will need a character mode RS-232 terminal and a null modem cable with CTS and DTR pulled high.



Appendix B
XT 101 Keyboard Scan Codes

For the sake of legibility, the keyboard scan codes are divided into two graphics:

- Section A Keyboard
- Section B Keyboard.

Figure B-1, Figure B-2, and Figure B-3 identify the sections of the keyboard and the scan codes specific to each.

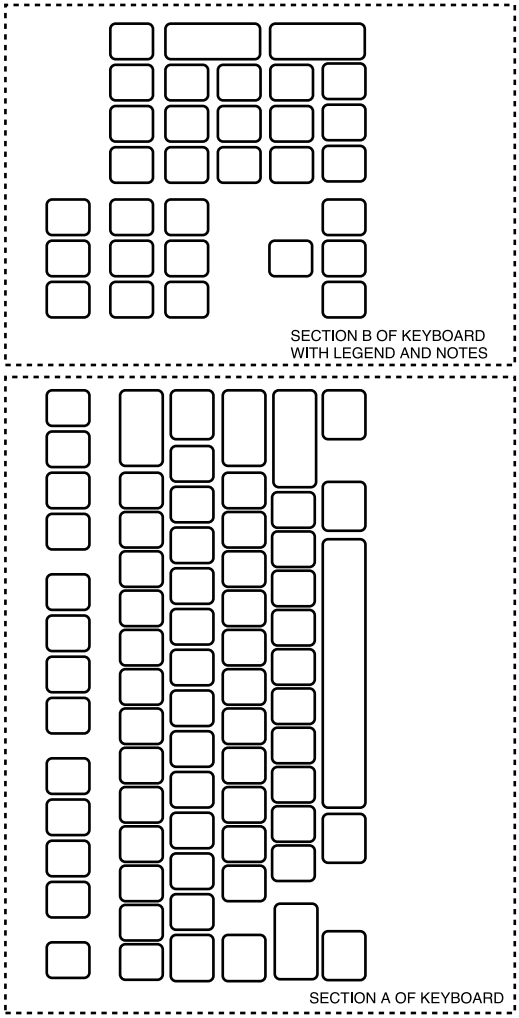


Figure B-1. Sections of the Keyboard

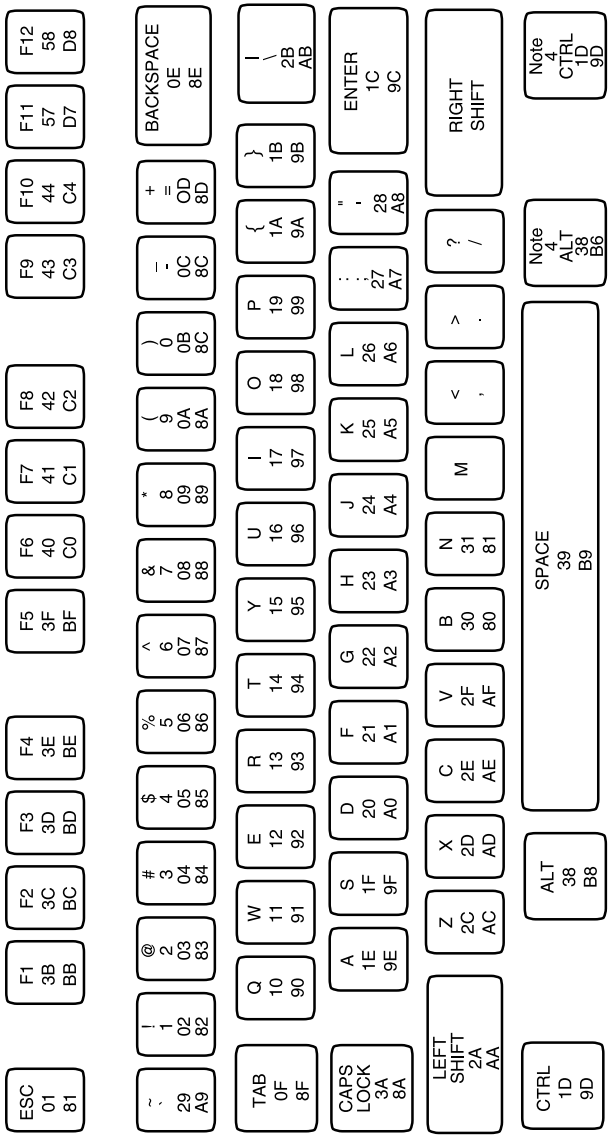


Figure B-2. Section A Keyboard Scan Codes

NOTES:

1. PRINT SCREEN key: DOWN code is E0 2A E0 7C. UP code is E0 B7 E0 AA.
   If CTRL KEY IS down, DOWN code is E0 37 and UP code is E0B7.
   If ALT key is down, DOWN code is 54 and UP code is D4.

2. PAUSE key: DOWN code is E1 1D 45 E1 9D C5. There is no UP CODE.
   BREAK is active when CTRL is down. DOWN code is E0 46 E0 C6. There is no UP code.

3. / key: DOWN code is E0 35.
   If LEFT SHIFT is down, the code is E0 AA D0 35 and the UP code is E0 B5 E0 2A.
   If RIGHT SHIFT is down, the code is E0 B6 E0 35 and the UP code is E0 B5 E0 35.

4. UP and DOWN code sequence is preceded by E0 hex.

5. If LEFT SHIFT is down, DOWN code is E0AA E) ## and UP code is E0 @@ E0 36, where ## is DOWN code and @@ is UP code for notated key.. See legend.

LEGEND FOR KEYBOARD SECTIONS A & B

XXXXX
YYYYY
##
@@

XXXXX = SHIFTED CHARACTER
YYYYY = CHARACTER
= DOWN CODE IN HEXADECIMAL
@@ = UP CODE IN HEXADECIMAL

Figure B-3. Section B Keyboard Scan Codes

Keyboard Redefinition Using ANSI.SYS

In DOS version 5.0 and higher, individual keys may be redefined using the ANSI.SYS driver. In particular, use the following invocation in CONFIG.SYS to enable full 101-key compatibility:

```
DEVICE=(drive:\path\)ANSI.SYS /X
```

Subsequently, individual keys may be redefined with sequences:

```
ESC[key;"string"p
```

where the letters ESC stand for the ASCII "Escape" character (Hex 1B), and the letters key stand for the particular keycode of the key being redefined. For example, to refine the key F1 to transmit "DIR /P" when pressed, issue the following:

```
echo ESC[0;59;"DIR /P"p
```

or to transmit a ^C when F2 is pressed:

```
echo ESC[0;60;03p
```

See pages B-3 and B-4 for keycode definitions.