TANDEM

T16/6520 and T16/6524 Video Display Units PROGRAMMING MANUAL

T16/8047-A00
82047

T16/6520 AND T16/6524

VIDEO DISPLAY UNITS

PROGRAMMING MANUAL

TANDEM COMPUTERS INCORPORATED
19333 Vallco Parkway
Cupertino, California 95014

Product No. T16/8047 A00
Part No. 82047

March 1980
Printed in U.S.A.

This publication is for programmers who are using the Tandem 16 Guardian Operating System to write application programs for the 6520 terminal. Additional information related to operating and programming the terminal can be found in the following publications.

- o T16/6520 and T16/6524 Video Display Units - Operating Guide 82048
- o Tandem 16 Guardian Operating System Programming Manual 82019

The information in this publication is organized to allow you to make a simple transition from learning about the terminal's functions to writing application programs for its use. It contains five chapters:

1. Introduction
2. Conversational Mode
3. Block Mode
4. Programming Considerations
5. Appendices

It is recommended that you use this publication by first reading the Introduction to familiarize yourself with some of the important terminal functions and features. Then, as you start writing application programs, use the remaining chapters to look up detailed reference information related to specific programming tasks.

Introduction - Contents

This chapter provides the information needed to understand the capabilities of the terminal. The organization is sequential; you can read it from front to back for general meaning and content. By using this chapter, you should be able to describe or define:

- o The basic functions provided by the terminal.
- o The differences between block and conversational modes.
- o How memory is organized in block and conversational modes.
- o The display control and attribute capabilities provided in block and conversational modes.
- o Control codes and how they are used.
- o Escape sequences and how they are used.
- o The file management procedures used in programming the terminal.
- o The buffer addressing and cursor addressing concepts required to read from and write to the terminal.

Conversational Mode - Contents

This chapter contains the reference information that you need to program the terminal in conversational mode. By using this chapter, you should be able to:

- o Understand the functions of all the control codes and escape sequences issued in conversational mode.
- o Set video attributes for conversational mode.

Block Mode - Contents

This chapter contains the reference information that you need to program the terminal in block mode. By using this chapter, you should be able to:

- o Understand the functions of all the control codes and escape sequences used in block mode.
- o Set video attributes for block mode.
- o Set data attributes for the protect submode of block mode.

Programming Considerations - Contents

This chapter contains general information related to programming the terminal in both block and conversational modes. By using this chapter and the information in the Guardian Operating System Programming manual, you should be able to:

- o Access the terminal.
- o Use the Write, Read, and Writeread procedures to write data to and read data from the terminal.
- o Switch modes.
- o Understand function key operations.
- o Understand cursor positioning and buffer addressing in block mode.
- o Write data to the 25th line.
- o Control the printer.

Appendices - Contents

This section contains two appendices.

1. Appendix A is a description of the data type table supplied with the 6520.
2. Appendix B is a summary of all the control codes and escape sequences that can be issued in block and conversational modes.

CONTENTS

INTRODUCTION 1-1
  Overview 1-1
  Conversational Mode Capabilities 1-3
    Memory Organization 1-3
    Submodes 1-4
    Display Control 1-4
    Cursor Movement 1-4
    Editing Functions 1-5
    Video Attributes 1-5
  Block Mode Capabilities 1-6
    Memory Organization 1-7
    Submodes 1-7
    Display Control 1-8
    Cursor Movement 1-9
    Editing Functions 1-10
    Video Attributes 1-10
    Data Attributes 1-11
  Programming Concepts 1-12
    Using File Management Procedures 1-12
    Control Codes and Escape Sequences 1-13
    Cursor Positioning and Buffer Addressing 1-14
    Selecting and Displaying Pages in Block Mode 1-14
    Conventions 1-15

CONVERSATIONAL MODE 2-1
  Function Overview 2-1
  Functions 2-3

BLOCK MODE 3-1
  Function Overview 3-1
  Functions 3-4

PROGRAMMING CONSIDERATIONS 4-1
  Description 4-1
  Accessing the Terminal 4-1
  Switching Modes 4-2
  Using File Management Procedures 4-3
  Block Mode Examples 4-3
  Conversational Mode Examples 4-6
  Function Key Operation 4-8
  Buffer Addressing In Block Mode 4-9
  Writing Data to the 25th Line 4-11
  Error Control 4-11
  Controlling the Printer 4-12

APPENDIX A - PREDEFINED DATA TYPE TABLE A-1
APPENDIX B - ESCAPE SEQUENCES AND CONTROL CODES B-1
INDEX

OVERVIEW

The 6520 terminal is an interactive terminal that provides a variety of functions and features including:

- Two modes of operation - conversational and block. Within each of these modes, display control, editing, and cursor movement functions along with a full range of video and data attributes (in block mode) are available.
- Multiple memory pages - increases throughput by keeping multiple pages of characters available for immediate access in the terminal's memory.
- A flexible communications interface - allows asynchronous operation; point-to-point attachment; and up to nine different asynchronous speeds.

The terminal keyboard is a typewriter style layout with 16 program function keys (providing 32 functions shifted and unshifted), an integral 10-key pad, local editing functions and an audible alarm.

Optionally, the terminal can support an external printer with the addition of an RS-232 output port that controls a serial printer. With this option, the terminal is designated as the 6524.

Figure 1-1 is an overview of the terminal and its operating environment. The key elements of the figure are:

- The application programs, that you write, which communicate through procedure calls with the Guardian file management system.
- The Guardian file management and terminal processes that provide the software interface between the application program and the terminal hardware.
- The terminal and its connections to the software system.

Your major concerns as an application programmer will be: how to assign attributes (video and data) to specific pieces of information on the terminal screen or in its memory and how to use the terminal functions (cursor movement, display control, etc.) to manipulate and display the information.

The rest of this chapter contains the general information needed to understand the functions provided by the terminal and to understand how to control the terminal from an application program. For detailed information on these topics see the Conversational Mode, Block Mode, and Programming Considerations chapters of this publication.

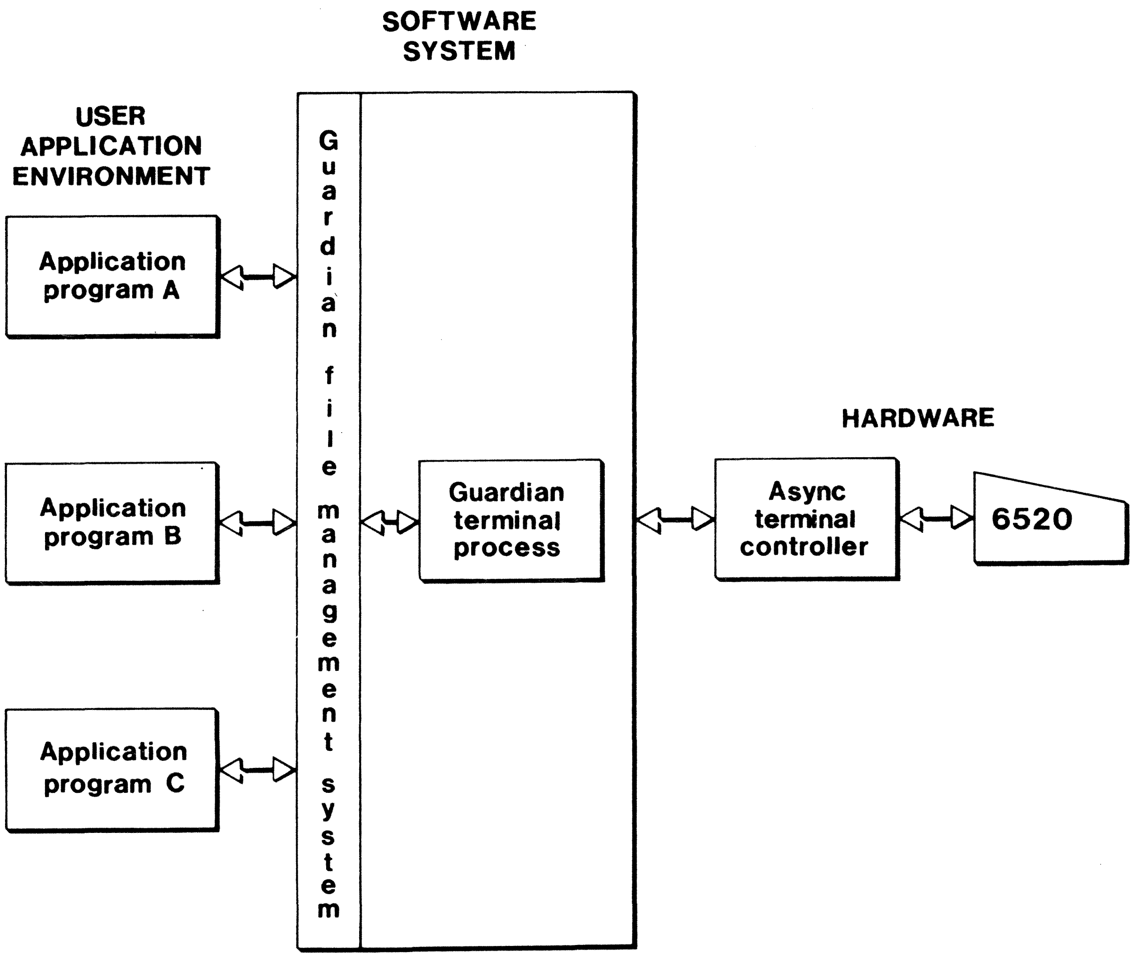

SOFTWARE SYSTEM
USER APPLICATION ENVIRONMENT
Application program A
Application program B
Application program C
Guardian file management system
Guardian terminal process
HARDWARE
Async terminal controller
6520

Figure 1-1. Overview of 6520 Operating Environment

CONVERSATIONAL MODE CAPABILITIES

In conversational mode, the terminal transmits characters as keystrokes are made. Selection of conversational mode is controlled in one of two ways:

1. If the switches inside or on the back of the terminal are set for conversational mode, the terminal will power up in conversational mode.
2. If the terminal is in block mode, it can be switched to conversational by issuing a mode switching procedure from your application program.

Conversational mode has two submodes: half duplex (HDX) and full duplex (FDX). Selection of submodes is controlled by a switch inside the terminal.

To help you understand how to use and control the capabilities provided in conversational mode, this section describes:

- How memory is organized
- The submodes
- Display control
- Editing functions
- Cursor movement
- Video attributes

Memory Organization - Conversational Mode

Memory is treated as a continuous area of 11,520 bytes, the equivalent of six 1920 character screens. Display memory consists of 144 lines, 24 of which can be displayed at one time. Through use of display control function, all the data in memory is available for display. The memory has an index mark and, when the screen is first displayed, the index mark is aligned with the top of the screen. Unless specified otherwise by cursor movement functions, data entry starts from the top of the screen and proceeds line-by-line to the 24th line. When the 24th line is filled, data continues to be entered on the 24th line and the previous data is automatically scrolled off the top; no data is lost. Data that is moved off the top of the screen continues to be stored until it is about to appear on the bottom of the screen; it is then erased to blanks.

In conversational mode, the 25th line is used as the status display line and since it resides in its own memory space, it is accessible to an application with up to 64 character spaces available for displaying information.

For an explanation of the functions available for viewing data that is not on the screen, see the Display Control section of this chapter.

Submodes - Conversational Mode

In the FDX submode, codes are transmitted, but not acted on, as they are entered from the keyboard. The codes are not stored or displayed. Incoming characters are always received and interpreted as displayable characters, control characters or escape sequences. All displayable characters are stored and displayed at the current cursor position and the cursor is incremented to the next position. If the cursor is at column 80, it is incremented to the first column of the next line. If the cursor is at column 80 of the 24th line, it is moved to the first column of the 24th line and the display is moved up one line.

In the HDX submode, all keystrokes are acted upon within the terminal and transmitted. For example, a typed character causes the terminal to transmit the character, store and display it at the current cursor position, and increment the cursor by one position. For incoming data, the HDX submode acts exactly the same as the FDX submode.

Display Control - Conversational Mode

Display control functions allow the operator or application program to control what appears on the display screen by moving lines of data from the memory onto the screen. In conversational mode, the following functions are available.

- o Roll up
- o Roll down
- o Next page
- o Previous page
- o Set line width to 40/80 characters

Cursor Movement - Conversational Mode

The cursor is identified as a blinking underline or an inverse block and cursor movement functions allow the operator or an application program to control the position of the cursor on the screen. In conversational mode, the cursor movement functions include:

- o Set/Clear tabs
- o Tab
- o Cursor Up/Down/Left/Right/Home/Home Down

Attempts to move the cursor off the right side of the screen result in the cursor moving to the first position of the next line. Attempts to move the cursor off the left side of the screen result in the cursor moving to the last position of the previous line. Attempts to move the cursor off the top or bottom of the screen result in existing lines being moved down onto the bottom of the screen or being moved up onto the top of the screen.

When the index mark inside the display memory aligns with the top of the screen, further movement down is inhibited. When the index mark is aligned with the 24th line, new lines moved up onto the bottom of the screen are blanked.

Editing Functions - Conversational Mode

Editing functions allow the operator or application program to change data after it appears on the display; the change is permanent and the new data will be placed into or removed from memory. Editing functions should be distinguished from display control functions that simply display different parts of the memory, but do not change the actual information in the memory. In conversational mode, the editing functions are:

- o Erase line
- o Erase page

Video Attributes - Conversational Mode

In conversational mode, video attributes can be turned on in one of two ways:

1. By issuing the set video escape sequence followed by one byte of video attributes (in ASCII character code form).
2. By issuing the set video condition prior register escape sequence followed by one byte of attributes (in ASCII character code form). When this escape sequence is issued, the selected attributes are stored in a register and the attribute remains in effect for the entire screen unless another attribute is encountered.

Regardless of how the attributes are defined, they perform the same functions on the screen. A video attribute is in effect until it is rolled off the screen or until another attribute is encountered. The available video attributes are:

Underscore	the character positions following the attribute bit definition are underscored. The character above the underscore is not reduced in size.
Blinking	the video attribute position and following character positions blink.
Reverse	the attribute position and following character positions are displayed as reverse video.
Dim	the attribute position and following character positions are displayed at half intensity.
Blank	the attribute position and following character positions will not be displayed. This attribute takes precedence over all other attributes.

Note that combinations of attributes (e.g, reverse dim) can be issued.

BLOCK MODE CAPABILITIES

In block mode, the terminal has the capability of transmitting and receiving blocks of data.

Selection of block mode occurs in one of two ways:

1. If the switches inside or on the back the terminal are set for block mode, the terminal will power-up in block mode.
2. If the terminal is not in block mode, it can be entered by issuing a procedure call from your application program.

The terminal is always ready to receive blocks of data, but it only transmits when a function key is used, or when an escape sequence requesting data is received.

Block mode has two submodes: protect and non-protect. Selection of submodes is controlled by escape sequences from the application program.

To help you understand how to control and use the capabilities provided in block mode, this section describes:

- o How memory is organized
- o The submodes
- o The display control functions available
- o Cursor control
- o Editing functions
- o Video attributes
- o The data attributes that can be defined for characters and fields in the protect submode.

Memory Organization - Block Mode

In block mode, memory is organized as:

1. Seven 1920-character pages where each page is displayed as 24 lines of 80 characters each

   or
2. Ten 960-character pages where each page is displayed as 24 lines of 40 double-width characters.

The default is 1920-character pages and it can be changed by use of an escape sequence from the application program.

Only one page can be displayed at a time and the operator cannot move from page to page without the aid of the application program. The program can perform I/O to or from the pages at any time. However, there is no I/O or editing across page boundaries.

Less than seven pages can be defined as useable for the display and the remaining memory can be used for additional data attribute space.

Submodes

In <u>non-protect submode</u>, all character positions are treated the same; the cursor can be positioned anywhere on the screen and data can be entered into any field.

In <u>protect submode</u>, individual character positions and an area of adjacent character positions on the screen (called a field) can be defined to have various attributes. The attributes specify:

- o How information is to be displayed (video attributes)
- o What the user can enter in the position and what specific characters are allowed (data attributes)

In protect submode, it is most convenient to think of all operations as taking place over the fields that you define. Fields can be any length and can span lines. When fields are defined as protected, they cannot be modified from the keyboard and the cursor cannot be positioned into their area. However, application programs can write into protected fields by using appropriate control codes and escape sequences.

In protect submode, the first position of the first row on the page is always protected; it is the start of the first field on the page. Subsequent fields can be defined on the page in any sequence. Each field definition starts with a buffer address followed by a start field control sequence. After issuing a start field control sequence, all character positions on the screen to the start of the next field have the same attribute. When another start field is encountered, the host processor program sets the buffer address to the start of the next field and defines the attributes for that field: terminating the previous field.

Each field on the screen has an associated entry in an off-screen table that defines its data attributes. This table is built as the page is defined and cleared as particular fields are deleted.

When a field is defined, its starting address is the address of the video attribute. However, the entry for the field in the attribute table points to the next position on the screen; this position is where data attributes take effect. The video/data attribute address pair will always remain intact with the video attribute being the controlling item. When field modification (insert line, delete line, etc.) is requested and the address range includes one but not both of the attributes, the requested change will occur to the screen. However, the requested change to the data attribute table will only occur if the address range includes the address of the video attribute.

Display Control - Block Mode

In block mode, particular pages can be selected for display through issuing an escape sequence from your application program.

Cursor Movement - Block Mode

The cursor is identified as a blinking underline or an inverse block; cursor movement functions allow the operator or an application program to control the position of the cursor on the screen. In block mode, the cursor movement functions are:

- o Set tab
- o Clear tab
- o Tab
- o Cursor Up/Down/Right/Left/Home/Home Down

Attempts to move the cursor off the right side of the screen result in the cursor moving to the first position of the next line. Attempts to move the cursor off the left side of the screen result in the cursor moving to the last position of the previous line. Attempts to move the cursor off the top or bottom of the screen result in the cursor wrapping around to the opposite edge of the screen.

Each page in memory has a separate cursor and when a new page is displayed, the cursor moves to the location stored for this page. This location can be: the same location as the last displayed page, a new location set by the host processor program, or the default home position if the cursor has not been set previously.

In protect submode, the cursor cannot be positioned into a protected character position on the display by the operator or by the application program. If an attempt is made to position the cursor into a protected field, the terminal will move the cursor from this position to the beginning of the next unprotected field--wrapping around the page boundaries if necessary. If there are no unprotected fields on the page, the cursor will not be displayed.

Editing Functions - Block Mode

Editing functions allow the operator or application program to change data on a page; the change will be permanent. In block mode, the editing functions available depend on whether protect or non-protect submode is selected.

In non-protect submode, the functions are:

- Insert line
- Delete line
- Erase line
- Erase page
- Insert character
- Delete character

In protect submode, the functions are:

- Erase field
- Erase page
- Insert character
- Delete character
- Insert line
- Delete line

Video Attributes - Block Mode

In block mode, video attributes can be turned on in one of three ways:

1. By issuing the set video escape sequence followed by one byte of video attributes (in ASCII character code form). This function can be used to store video attributes anywhere on the page, making it possible to define several video attributes within a field.

2. By issuing the set video condition prior register escape sequence followed by one byte of attributes (in ASCII character code form). When this escape sequence is issued, the selected attributes are stored in a register and the attribute remains in effect for the entire page unless another attribute is encountered. A value for the prior condition is maintained for each page and when a new page is displayed, this value is loaded into the hardware register.

3. By issuing the Start Field control code or Start Field Extended escape sequence followed by two bytes of attributes, with the first byte defining video attributes for fields in protect submode.

Regardless of how the attributes are defined, they perform the same functions on the screen. A video attribute is in effect until a new page is loaded or until another attribute is encountered. The available video attributes are:

Underscore	the character positions following the attribute bit definition are underscored. The character above the underscore is not reduced in size.
Blinking	the video attribute position and following character positions blink.
Reverse	the attribute position and following character positions are displayed as reverse video.
Dim	the attribute position and following character positions are displayed at half intensity.
Blank	the attribute position and following character positions will not be displayed. This attribute takes precedence over all other attributes.

Note that combinations of attributes (e.g., reverse dim) can be issued.

Data Attributes - Block Mode

In protected submode of block mode, data attributes can be specified for characters and fields on a page by issuing the start field control code or the start field extended escape sequence followed by two bytes of data in ASCII character code form; the first byte defines the video attributes as described previously and the second byte defines the following data attributes:

- o Modified data tag
- o Data type including: no restriction, alpha, numeric, alphanumeric, full numeric, full numeric with space, alpha with space, and alphanumeric with space.
- o Auto - Tab disable
- o Protected
- o Upshift

See the BLOCK MODE chapter of this publication for a detailed description of data attributes, data types and their implementation through escape sequences and control codes.

PROGRAMMING CONCEPTS

As illustrated previously, in Figure 1-1, control of the terminal is through the standard Guardian interface supplied with the system. However, your installation can write its own application programs to commmunicate with the terminal operator. These programs communicate through the Guardian file management system and in this publication, the word "application program" is used to denote the program communicating with the terminal through the standard interface. To effectively write application programs, you need to understand the following basic concepts.

- o Using file management procedures to read and write data from the 6520 and to write control codes and escape sequences to the terminal.
- o Using control codes and escape sequences to control terminal functions.
- o Positioning the cursor and writing to memory positions using buffer addressing.
- o Selecting and displaying pages in block mode.
- o The conventions used in this publication.

This section provides a brief overview of the preceding concepts. See the PROGRAMMING CONSIDERATIONS chapter of this publication for detailed information related to programming for the terminal.

Using File Management Procedures

To control the terminal from within an application program, you use the following procedures:

- o OPEN
- o READ
- o WRITE
- o WRITEREAD
- o SETMODE

These procedures allow you to: open the 6520 as a file and establish communications between your application program and the terminal (file); write escape sequences and control codes to the terminal; read from and write data to the terminal; and to change modes (block to conversational and vice versa). For detailed information about coding the procedures, see the file management section of the GUARDIAN OPERATING SYSTEM PROGRAMMING MANUAL. For detail related to using the procedures to control the 6520, see the PROGRAMMING CONSIDERATIONS chapter of this publication.

Control Codes and Escape Sequences

Control codes are not stored or displayed, but are acted on immediately by the 6520. These codes (ranging from %00 to %37) can be issued from an application program to control the functions shown in Table 1-1.

Table 1-1. Control Code Functions

ASCII Code	Keys used	Function	Mode(s)
NUL %00	CTRL @	Fill character - not stored in buffer	B & C
SOH %01	CTRL A	Function key header	B & C
ENQ %05	CTRL E	Send ACK	C only
ACK %06	CTRL F	Acknowledge ENQ	C only
BEL %07	CTRL G	Sound Alarm	B & C
BS %10	CTRL H	Cursor left one space	B & C
HT %11	CTRL I	Cursor to next tab stop	B & C
LF %12	CTRL J	Cursor down one line	B & C
CR %15	CTRL M	Cursor beginning of of current line	B & C
DC1 %21	CTRL Q	Set buffer address	B only
DC3 %23	CTRL S	Set cursor address	B only
ESC %33	CTRL [	Interpret next character for function	B & C
GS %35	CTRL]	Start field	B only

NOTE: B = block mode and C = conversational mode

All other codes within the range %00 through %37 are ignored.

NOTE: Problems can occur if you use control characters in passwords when logging onto the system. Since they are echoed back to the 6520 by the system, actions different from the ones that you anticipated can occur at the terminal. To avoid problems, all of the control codes in Table 1-1 should normally be avoided in passwords. For example, do not use CTRL Q as a password.

Escape sequences are a control code (ESC or CTRL [) followed by another character and, in some cases, data. The control code modifies the interpretation of the next character and you use escape sequences to perform a variety of functions in both block and conversational mode. See the BLOCK MODE and CONVERSATION MODE chapters of this publication for detailed information on control codes and escape sequences.

Cursor Positioning and Buffer Addressing

As you write application programs, you need to make a distinction between cursor position and buffer addressing. For the 6520, the cursor position is an offset from the home position on the screen. In both block and conversational modes all keyboard operations take place at the cursor position. After keyboard entry, the cursor will advance one position.

In block mode, buffer addressing is used to specify where the output from your application program will be written; the output can be written to any part of the screen regardless of cursor position or whether or not the screen contains protected fields. Each separate page in block mode has its own cursor. Therefore the cursor position and the buffer address can be the same or different, allowing you write to different areas while reading operator input from the cursor position. Additionally, buffer addressing can be used to read selected parts of a page.

Cursor position and buffer addresses are specified by issuing control sequences, in your program, followed by two ASCII characters that indicate the row and column positions over which operations start or take place.

Figure 1-2 illustrates the differences between memory management and addressing in block and conversational modes.

Selecting and Displaying Pages in Block Mode

In block mode, another important distinction is the difference between displayed pages and selected pages. A displayed page is the page on the terminal screen. A selected page is the page where your program's input and output occurs; again they can be the same of different. Initially, when you enter block mode, the selected page and the displayed page are both set to page 1 of memory. By issuing escape sequences from your program, you can control which page is selected and which page is displayed. This feature allows you to process operator input for one page while the operator is responding to another page of data on the screen.

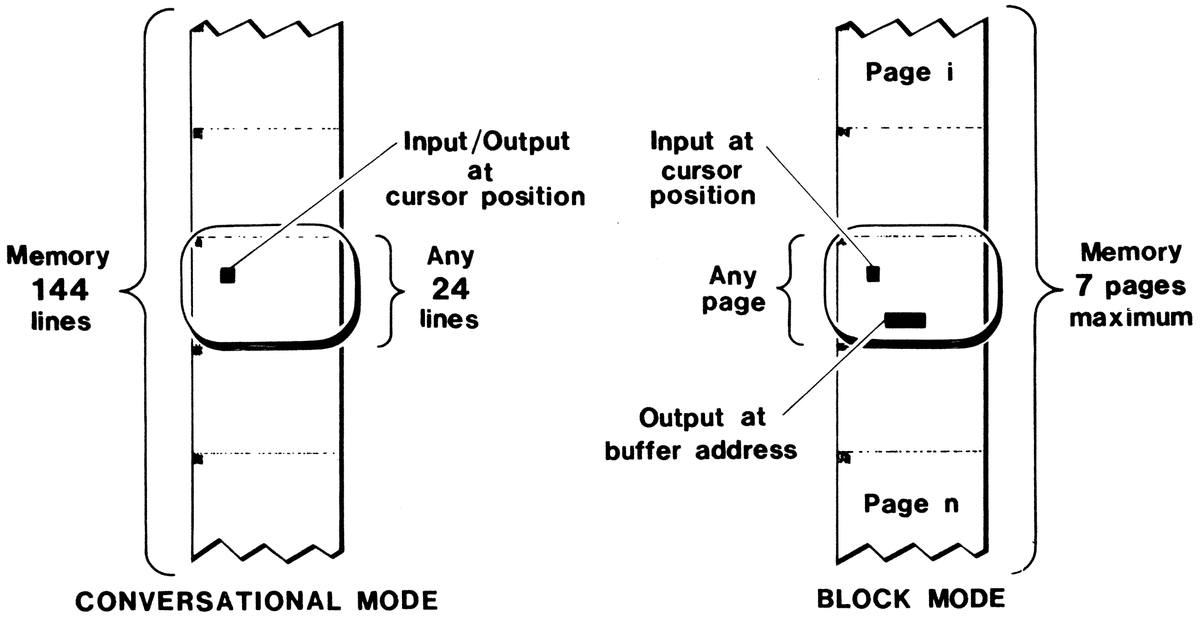


Figure 1-2. Memory Management and Addressing in Block and Conversational Modes

Conventions

Eight bit microprocessors traditionally number bits within a byte from the right using bit 0 as the LSB. Tandem normally stores a byte in bits 8-15 of a 16 bit word using bit 15 as the LSB. This publication uses the microprocessor designation for the 6520 shown below.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Tandem
								7	6	5	4	3	2	1	0	6520

FUNCTION OVERVIEW

There are five categories of functions in conversational mode: (1) cursor movement, (2) display control, (3) editing, (4) attribute control, and (5) miscellaneous control functions. The categories are described below and summmarized in Table 2-1. Each function can be controlled from an application program, communicating with the 6520, by issuing an appropriate escape sequence or control code. In the rest of this chapter, each function along with its associated escape sequence or control code is described in detail in the order that it appears in Table 2-1.

Cursor movement functions include:

- o Cursor left one space
- o Cursor down one line
- o Cursor to beginning of current line
- o Cursor home
- o Cursor home down
- o Cursor right
- o Clear tab
- o Cursor to next tab stop
- o Clear all tabs
- o Set tab

Display control functions include:

- o Next page
- o Previous page
- o Roll down
- o Roll up
- o Set line width to 40 characters
- o Set line width to 80 characters

Editing Functions include:

- o Erase to end of line.
- o Erase to end of page.

Attribute Control Functions include:

- o Set video attribute
- o Set video prior condition register

Miscellaneous control functions include:

- o Define RETURN key
- o Fill character
- o Function key header
- o Interpret next character for function
- o Modem disconnect
- o Print page
- o Read terminal status
- o Send ACK/Acknowledge ENQ
- o Sound audible alarm

Table 2-1. 6520 Programming Functions - Conversational Mode

Function	Controlled By
Clear all tabs	ESC 3
Clear memory to spaces	ESC I
Clear tab	ESC 2
Cursor down one line	code %12 (LF)
Cursor home	ESC H
Cursor home down	ESC F
Cursor left one space	code %10 (BS)
Cursor to beginning of line	code %15 (CR)
Cursor to next tab stop	code %11 (HT)
Cursor right	ESC C
Define RETURN key	ESC u
Display text on 25th line	ESC o
Erase to end of line	ESC K
Erase to end of page	ESC J
Execute self-test	ESC z
Fill character	code %00 (NUL)
Function key header	code %01 (SOH)
Interpret next character for function	code %33 (ESC)
Modem disconnect	ESC f
Next page	ESC U
Previous page	ESC V
Print page	ESC O
Read terminal status	ESC ^
Roll up	ESC S
Roll down	ESC T
Send ACK/Acknowlegde ENQ	codes %05 (ENQ) %06 (ACK)
Set line width to 40 chars	ESC 8
Set line width to 80 chars	ESC 9
Set tab	ESC 1
Set video attribute	ESC 6
Set video prior condition register	ESC 7
Sound Audible Alarm	code %07 (BEL)

FUNCTIONS

Each function in this section is described in the order that it appears in Table 2-1.

Clear All Tabs ... ESC 3.

All tabs will be cleared simultaneously.

Clear Memory to Spaces ... ESC I

All memory will be cleared to spaces. After the memory is cleared, the cursor is placed in the home position and all lines are set to display 80 characters.

Clear Tab ... ESC 2

The tab is cleared at the current cursor position.

Cursor Down One Line (LF) ... %12

The cursor is moved down one line. If the cursor is at the 24th line, the display is scrolled up one line.

Cursor Home ... ESC A

The display is scrolled down until the index mark is aligned with the top of the screen and the cursor is placed in the upper left-hand corner.

Cursor Home Down ...ESC F

The display is scrolled up until the index mark is at the 24th line and the cursor is placed at the 1st position of the line.

Cursor Left One Space (BS) ... %10

The cursor is moved left one space. If the cursor is at the home position on the screen, it will be moved to the last position of the previous line and the screen will be rolled up one line unless the index mark is aligned with the top of the screen.

Cursor Right ... ESC C

The cursor is moved right one column. If the cursor is at the last position of the screen, it is moved to the beginning of the next line and the display is scrolled up one line.

Cursor to Beginning of Current Line (CR) ... %15

The cursor is moved to the beginning of the current line.

Cursor to Next Tab Stop (HT) ... %11

The cursor is moved to the next specified tab stop.

Define RETURN Key ...ESC u

Upon power-up, the return key is set to generate a CR code in conversational mode. To modify this code when the RETURN key is pressed, issue:

```
     ESC u <count> <string>

where:

     count    is one character indiciating the length
              in bytes of <string> biased by %40;
              count therefore will equal length + %40.

     string   is an arbritrary character string that
              will be generated when the RETURN key
              is pressed.  Its maximum length is 8
              characters.
```

For example, to generate CR LF, issue the following.

```
ESC u <"> <CR> <LF>
```

Display Text on 25th Line ... ESC o

To display text on the 25th line, issue ESC o followed by a text string. The 6520 will display the text following the ESC o. To terminate the text string, follow it with a CR or any control sequence except ESC 6. A maximum of 64 characters can be displayed. The 25th line will be immediately blanked to display the characters. The characters can be erased by sending a null string: i.e. ESC o CR.

Erase to End-of-Line ... ESC K

All cursor positions, beginning with the current cursor position, are written to spaces. The cursor position does not change.

Erase to End-of-Page ... ESC J

All character positions in memory, beginning with the current cursor position, are written to spaces. The position of the cursor does not change.

Execute Self-Test ... ESC z

Self-test does a test of the RAM and ROM, and puts a pattern on the display. This pattern - containing all of the characters capable of being displayed as well as characters displayed with all of the video attributes - indicates that the self-test has passed. Symbols occupying the 32 control character positions in the ROM will not be displayed. Any error conditions are displayed on the 25th line.

After execution of self-test, the original contents of memory are lost and the screen will contain the test pattern. The terminal will be returned to the mode (block or conversational) that it was in when the self-test was issued.

Fill Character (NUL) ... %00

The character is not stored in the buffer.

Function Key Header (SOH) ... %01

In conversational mode, when a function key is depressed a predefined sequence starting with %01 and terminating with a CR is transmitted. The format of the sequence is

```
        SOH <char> <cursor> CR

  where:

       char     is an ASCII @ through O for F1 - F16 unshifted
                or an ASCII ` through o for F1 - F16 shifted.

       cursor   is a two-byte sequence specifying the current
                cursor position. The format of this sequence
                is two ASCII characters, representing the row
                and column respectively. The ASCII characters
                are derived from the respective absolute row and
                column numbers by adding %37. This results in
                characters space through 7 (%40 - %67) represent-
                ing rows 1 -24 and characters space through o
                (%40 - %157) representing columns 1-80.
```

Note that the preceding sequence can be echoed back by the host. Therefore, when the terminal receives a %01, it ignores it and all subsequent characters up to and including the next CR unless the character immediately following the %01 is a B or C which indicate that the sequence is a mode switching message. See the PROGRAMMING CONSIDERATIONS chapter for a discussion of mode switching.

Interpret Next Character For Function (ESC) ... %33

The terminal sets an internal flag that causes the character sequence following the %33 to be interpreted differently. This character sequence is known as an escape (ESC) sequence and it is used to control terminal functions.

Modem Disconnect ... ESC f

The terminal places the Data Terminal Ready line into a low state for three seconds which causes the terminal to "hang up."

Next Page ... ESC U

The display is moved up 24 lines and the next available 24 lines in memory are displayed.

Previous Page ... ESC V

The display is moved down 24 lines.

Print Page ... ESC O

The 24 lines on the display are sent to the printer port for printing. This function is available for printers that have an RS-232C signal interface. See the PROGRAMMING CONSIDERATIONS chapter of this publication for an explanation of controlling the printer.

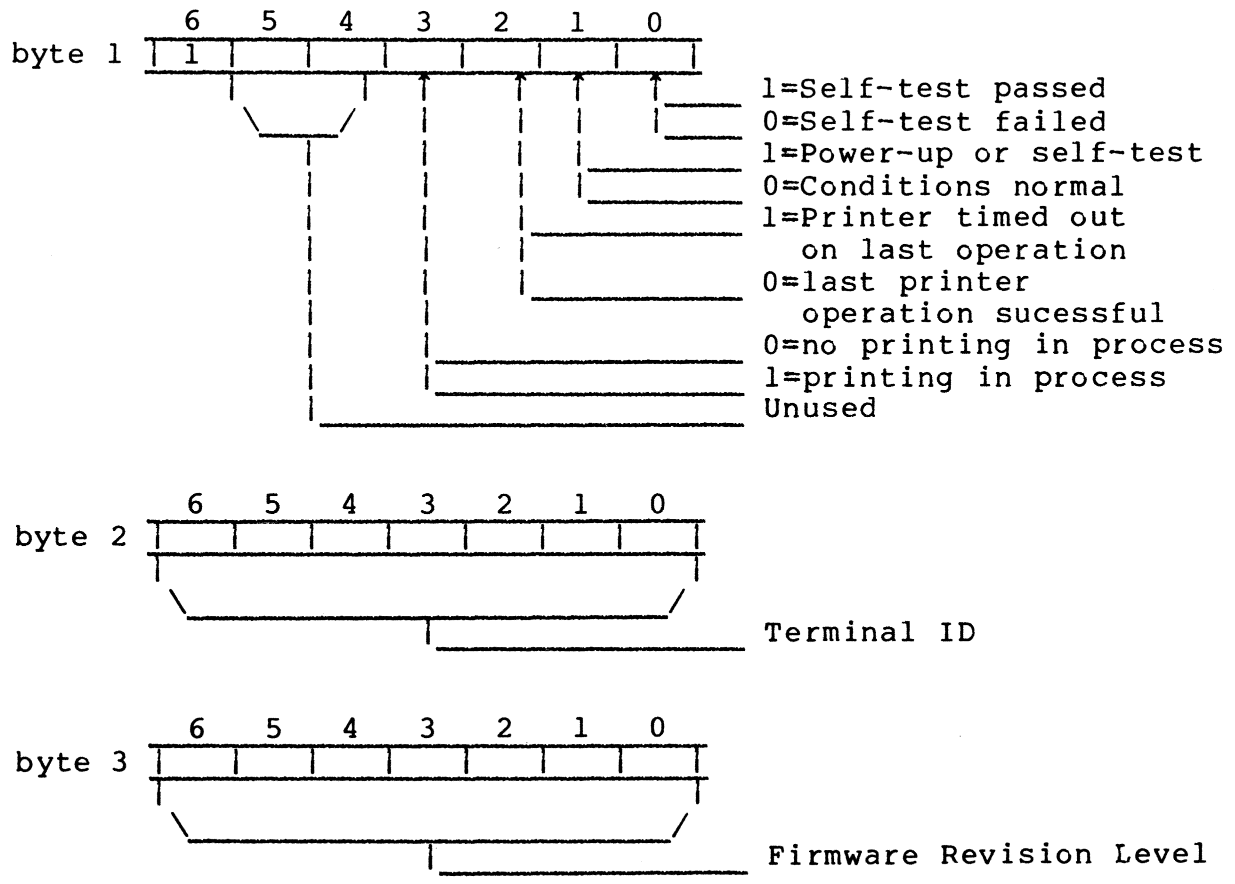
Read Terminal Status ... ESC ^

The 6520 transmits its status to the program. The format of this message is:

SOH char status CR

char is an ASCII ? (%77)

status is a three-byte sequence, with the following format:



The status bytes are described in Table 2-2.

Table 2-2. Terminal Status Bytes

Self-test passed = 1 when the last execution of self-test was successful. Self-test passed = 0 when either the last execution of self-test was unsuccessful or if self-test has not been run since the last power-up.
Printer status = 1 when the last printer operation timed out waiting for a data terminal ready from the printer to occur. Printer status = 0 when the last printer operation succeeded.
Print in process = 1 when a printer operation is in process and 0 when no printer operation is in process.
Power-up or self-test = 1 when either a power-up has occurred or self-test has been run. This bit is reset when status has been read successfully.
Terminal ID is a code which will be used to distinguish between various Tandem terminal products. For the initial product described in this publication, this code will be a "C".
Firmware Revision Level is a code which will be used to distinguish between various firmware revision levels of a given product. For the initial product described in this publication, this code will be a "C". The revision level will be changed each time the firmware is revised. This code is displayed in column 80 of the 25th line while self-test is executing.

<u>Roll Up ... ESC S</u>

The display will move up one line each time this sequence is issued. When the index mark is aligned with the 24th line, this sequence will be ignored.

<u>Send ACK/Acknowlege ENQ .... %05 (ENQ) or %06 (ACK)</u>

Since some of the control codes and escape sequences that are received by the terminal can take longer than one character sequence to process, the terminal has a software procedure that is used to buffer incoming characters until they can be processed. Under some circumstances, this procedure can be overrun. To control this problem, fill characters or time delay sequences can be used after the time-consuming sequences. An alternative solution is to send an %05 (ENQ) at any time. The terminal will respond with a %06 (ACK) only when the buffer is empty. To guarantee no overruns, the ENQ should be sent after every 256 characters. Under normal circumstances, the ACK will be returned immediately.

<u>Set Line Width to 40 Characters ... ESC 8</u>

The line containing the cursor is set to display 40 double-width characters and the remaining characters, on the line, are not displayed. This escape sequence affects only the line containing the cursor and all other lines are displayed as 80 characters unless an ESC 8 has been issued for them. As new lines are created (because of roll-up), the lines are set to 80 characters. If the cursor is beyond the 40th column when this sequence is issued, it is moved to the first position of the current line.

<u>Set Line Width to 80 Characters ... ESC 9</u>

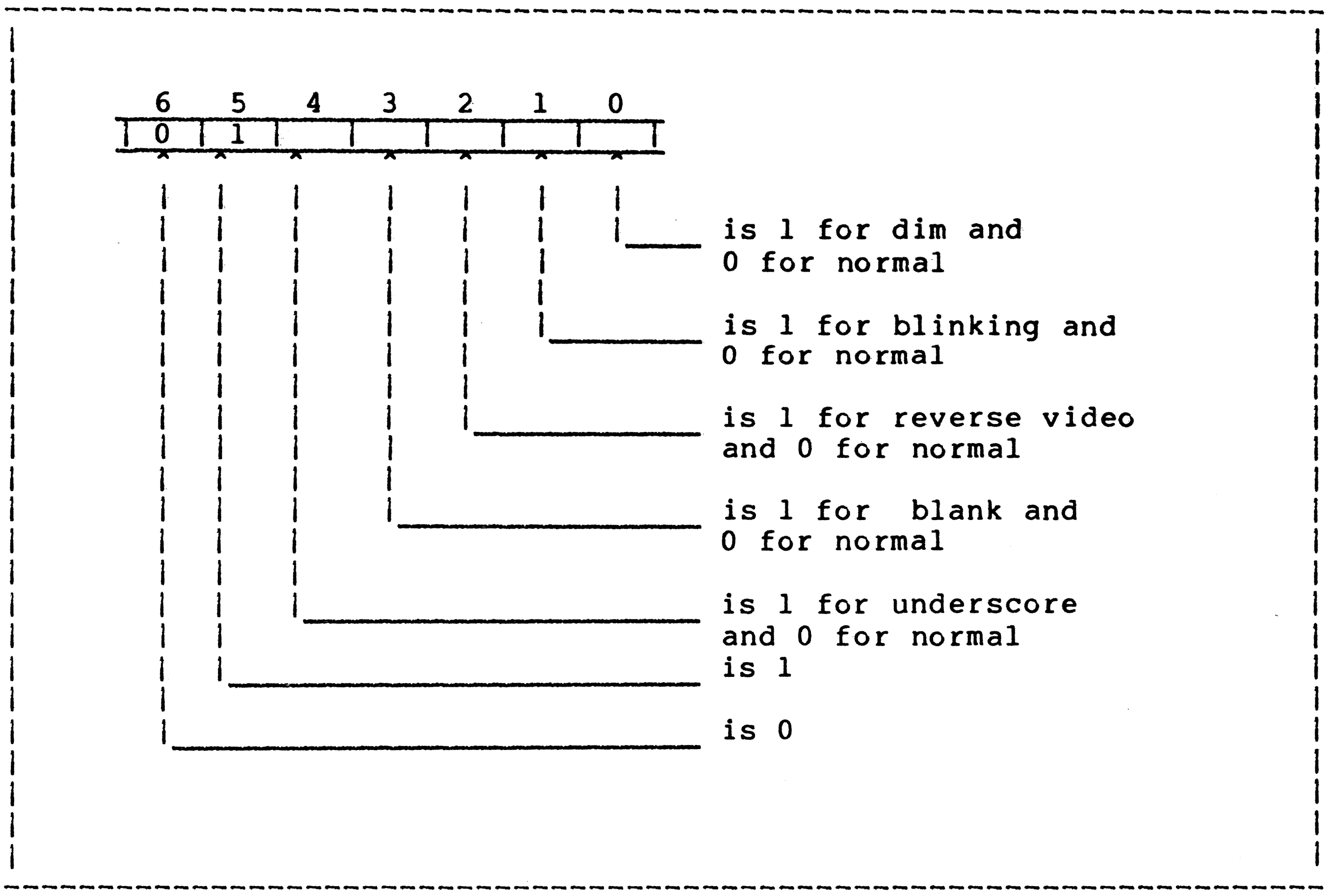
The line containing the cursor is set to display 80 characters.

<u>Set Tab ... ESC 1</u>

Tabs are set at the column of the current cursor position for all lines on the screen. The tabs do not take up space on the screen.

Set Video Attribute ... ESC 6

Video attributes are modified in conversational mode by issuing ESC 6 followed by an ASCII character corresponding to the binary values below.


6 5 4 3 2 1 0
0 1
is 1 for dim and 0 for normal
is 1 for blinking and 0 for normal
is 1 for reverse video and 0 for normal
is 1 for blank and 0 for normal
is 1 for underscore and 0 for normal
is 1
is 0

For example, to turn on a reverse video blinking field, issue ESC 6 & (& is the ASCII character code corresponding to binary 0100110). Issuing an ASCII character that is outside the range will cause all normal attributes to be assigned.

An attribute takes up a character position on the screen and is displayed as a blank with the same attribute as the field it is defining with one exception: the underscore is not turned on until the character following the attribute. When underscore is turned off, the effect is immediate.

The video attribute remains in effect until the next attribute character is encountered: left-to-right and top-to-bottom on the screen. When a video attribute rolls off the top of the screen, the rest of the characters return to normal until another video attribute is encountered.

Set Video Prior Condition ... ESC 7

ESC 7 should be followed by a video attribute character as described in the previous section. The terminal will load the video prior condition register with the specified attribute and it will remain in effect for the entire screen unless another attribute is encountered. The attribute does not occupy a position on the screen. When memory is cleared, the attribute remains in effect and can be changed only by issuing another ESC 7. Upon power-up, the video prior condition register is initialized to normal video.

Sound Audible Alarm (BEL) ... %07

When this code is received, the terminal will sound an audio tone.

FUNCTION OVERVIEW

There are five categories of functions in block mode: (1) cursor movement, (2) editing, (3) attribute control, (4) page control, and (5) miscellaneous control functions. The categories are described below and summarized in Table 3-1. Each function can be controlled from an application program, communicating with the 6520/6524, by issuing an appropriate escape sequence or control code. In the rest of this chapter, each function along with its associated escape sequence or control code is described in detail in the order that it appears in Table 3-1.

Cursor Movement Functions include:

- o Cursor left one space
- o Cursor down one line
- o Cursor to beginning of current line
- o Cursor to next tab stop
- o Cursor home
- o Cursor home down
- o Cursor right
- o Cursor up
- o Back tab
- o Clear all tabs
- o Clear tab
- o Set tab

Editing Functions include:

- o Clear to spaces
- o Delete character
- o Delete line
- o Disable line editing
- o Insert character
- o Insert line
- o Erase to end of line/field
- o Erase to end of page

Attribute Functions include:

- o Define data type table
- o Set video attribute
- o Set video prior condition register
- o Start field
- o Start field extended
- o Reset modified data tags

Page Control Functions include:

- o Display page
- o Select page
- o Set page size to 960
- o Set maximum page number

Miscellaneous Control Functions include:

- o Enter protect submode
- o Exit protect submode
- o Lock keyboard
- o Unlock keyboard
- o Read buffer
- o Read cursor address
- o Read with address
- o Read terminal status
- o Print page
- o Display text on 25th line
- o Modem disconnect
- o One second delay
- o Simulate function key
- o Execute self-test
- o Reinitialize
- o Fill character
- o Interpret next character for function
- o Set buffer address
- o Set cursor address
- o Sound audible alarm

Table 3-1. 6520 Programming Functions - Block Mode
Part 1 of 2

Function	Controlled By
Back tab	ESC i
Clear all tabs	ESC 3
Clear tab	ESC 2
Clear to spaces	ESC I
Cursor down one line	code %12 (LF)
Cursor home	ESC H
Cursor home down	ESC F
Cursor left one space	code %10 (BS)
Cursor right one space	ESC C
Cursor to beginning of current line	code %15 (CR)
Cursor to next tab stop	code %11 (HT)
Cursor up	ESC A
Define data type table	ESC r
Delete character	ESC P
Delete line	ESC M

Table 3-1. 6520/24 Programming Functions - Block Mode
Part 2 of 2

Function	Controlled By
Disable line editing	ESC N
Display page	ESC ;
Display text on 25th line	ESC o
Enter protect submode	ESC W
Erase to end of line/field	ESC K
Erase to end of page	ESC J
Execute self-test	ESC z
Exit protect submode	ESC X
Fill character	code %00 (NUL)
Insert character	ESC O
Insert line	ESC L
Interpret next character for function	code %33 (ESC)
Lock keyboard	ESC c
Modem disconnect	ESC f
One second delay	ESC @
Print page	ESC 0
Read buffer	ESC <
Read cursor address	ESC a
Read terminal status	ESC ^
Read with address	ESC =
Reinitialize	ESC q
Reset modified data tags	ESC >
Select page	ESC :
Set buffer address	code %21 (DC1)
Set cursor address	code %23 (DC3)
Set maximum page number	ESC p
Set page size to 960	ESC t
Set tab	ESC 1
Set video attribute	ESC 6
Set video prior condition register	ESC 7
Simulate function key	ESC d
Sound audible alarm	code %07 (BEL)
Start field	code %35 (GS)
Start field extended	ESC [
Unlock keyboard	ESC b

FUNCTIONS

Each function in this section is described in the order that it appears in Table 3-1.

Back Tab ... ESC i

In non-protect submode, the cursor is moved to the previous columnar tab stop or to the first column of the current line if there are no further tab stops on the current line. If the cursor is in the first colummn of the line, it is moved to the rightmost tab stop of the previous line.

In protect submode, the cursor is moved to the first unprotected position of the previous field in the page if it is at the beginning of a field when the back tab is issued. If there is no unprotected previous field, the cursor moves to the lower right-hand corner of the screen and the search continues. If the cursor is not at the beginning of a field when back tab is issued, it moves to the first position of the field that it is in.

Clear All Tabs ... ESC 3

All tabs will be cleared.

Clear Tab ... ESC 2

The tab is cleared at the current buffer address.

Clear to Spaces ... ESC I

This sequence is followed by a pair of buffer addresses specifying the starting row and column and the ending row and column over which the clearing occurs. The buffer addresses are transmitted as two encoded ASCII characters representing the row and column respectively. The ASCII charararacters are derived from the respective absolute row and column numbers by adding %37. This results in space through 7 (%40 through %67) representing rows 1-24 and space through o (%40 through %157) representing columns 1-80.

In non-protect submode, this escape sequence causes the page from the first specified address through the last address to be written to spaces. There is no modification of the buffer address or cursor address by this sequence.

In protect submode, this escape sequence causes the page from the first specified address through the last specified address to be written to spaces. Additionally, if any of the locations in the range are exactly equal to the start address of a field (this would normally contain the video specification for the field), the associated entry in the attribute table will be deleted. After this operation is complete, the cursor is in a protected position and it will tab forward to the next unprotected position. If there are no remaining unprotected positions on the page, the cursor will not be displayed. The current buffer address is not modified by this sequence.

<u>Cursor Down One Line (LF) ... %12</u>

The cursor is moved down one line. If the cursor is at the 24th line, the display is scrolled up one line.

<u>Cursor Home ... ESC H</u>

The 6520 puts the display in the home position, corresponding to the first column of the first row of the page in non-protect submode and the first position of the first unprotected field in protect submode.

<u>Cursor Home Down ... ESC F</u>

The 6520 puts the display in the home down position, corresponding to the first column of the 24th row of the page in non-protect submode and the first position of the last unprotected field in protect submmode.

<u>Cursor Left One Space (BS) ... %10</u>

The cursor is moved left one space. If the cursor is at the first position on the screen, it will be moved to the last position of the previous line.

<u>Cursor Right ... ESC C</u>

The cursor is moved right one column. If the cursor is at the last position of the line, it is moved to the beginning of the next line.

<u>Cursor to Beginning of Current Line (CR) .. %15</u>

In non-protect submode, the cursor will be positioned to the beginning of the next line. In protect submode, the cursor is moved to the next tab stop.

<u>Cursor to Next Tab Stop (HT) ... %11</u>

In non-protect submode, the cursor is moved to the next columnar tab stop or to the first column of the next line if there are no further tab stops on the line. In non-protect submode, there is only one set of columnar tab stops and they apply to all pages.

In protect submode, the cursor is moved to the first unprotected position of the next field in the page. If there are no unprotected fields, the cursor moves around the home position and the search continues.

<u>Cursor Up ... ESC A</u>

The cursor is moved up one line.

Define Data Type Table ... ESC r

The data type attribute defines which of several possible sets of characters can be entered into a field. The attribute is 3 bits long and it defines allowable characters to be from one of eight possible sets. The sets are defined by a 96-byte table that has one byte for each character from space through DEL (%40 through %177). For a given entry, if bit n is a "1", then the character is a member of set n and is allowable in a field of data type n. A character can be a member of more than one set.

An application program can redefine this table by issuing ESC r followed by a sequence of 96 hexadecimal-ASCII pairs (0-9, A-F). Each pair defines one entry in the table; all 96 pairs must be defined for each sequence. Each hexadecimal-ASCII pair is the ASCII equivalent of the hexadecimal values of the bits in the table. For example, the ASCII equivalent of X'A0' is A0. The new definition will be in effect until protect submode is reentered.

A predefined data type table is stored in ROM. See Appendix A for a description of the table, its contents and an example of how to change it.

Delete Character ... ESC P

In protect submode, the character at the current buffer address is deleted and all characters to the end of field are moved left one position. Additionally, a space is inserted at the end of the field. If the buffer address points to a protected area, no action occurs.

In non-protect submode, the character at the current buffer address is deleted and all characters to the end of the line are moved left one position.

Delete Line ... ESC M

All characters are deleted in the line at the current buffer address. In protect submode, all field attributes in the line will be deleted from the attribute table and no new attributes will created for the new 24th line.

Disable Line Editing ... ESC N

The INS/DEL LINE key is disabled from local operation and becomes an additional function key. The new function key operation will be in effect until:

- o Block mode is reentered
- o A reinitialize escape sequence is executed
- o A mode switch from protect to non-protect submode occurs

Display Page ... ESC ;

Issuing this sequence followed by a single character, specifying page number, allows the application program to control which page is selected for display. The character starts at ! (%41) representing page 1 and continues to the maximum number of pages. Specifying page 0 (space) causes the entire screen to be blanked and the keyboard to be locked. You should be aware of the following when using this escape sequence.

- o Writing to or reading from a non-displayed page does not cause the keyboard to be locked.
- o The selected page and the displayed page could be the same.
- o If the display page and the selected page are the same, any attempt to write to or read from the page will cause the keyboard to lock until the operation is complete.
- o The selected page continues to be displayed until a request to display a new page is received.
- o The default on entering block mode is page 1.

Display Text on 25th Line ... ESC o

Issuing ESC o followed by a character string will cause the 6520 to display the characters following the ESC o. To terminate the character string, follow it with a CR or any escape sequence except ESC 6. A maximum of 64 characters can be displayed. The 25th line will be immediately blanked to display the characters. The characters can be erased by sending a null string; i.e., ESC o CR.

Video attributes can be turned on and off in the 25th line by using the set video escape sequence. Since attributes require a character position, their use will restrict the text on the 25th line to less than 64 characters.

Enter Protect Submode ... ESC W

The 6520 is placed in protected submode of block mode. Upon entering protect submode,the following conditions are in effect.

- o Any printing in process is aborted.
- o Tabbing is done on a field base; columnar tabs are ignored.
- o Only unprotected characters can be modified from the keyboard.
- o All pages are cleared to protected space.
- o The set video prior condition register is set to normal video for all pages.
- o Page 1 is displayed.
- o Page 1 is selected.
- o The buffer address is set to row 1, column 1 for all pages.
- o The cursor address is set to row 1, column 1 for all pages.
- o The keyboard is locked.
- o The 25th line is cleared.
- o The data type attribute table is initialized to the ROM table.
- o Insert mode is reset.

Erase to End-of-Line/Field ... ESC K

In non-protect submode all character positions in the line, starting at the current buffer address, are erased to blanks.

In protect submode, all character positions in the field, starting at the current buffer address, are erased to blanks.

Erase to End-of-Page ... ESC J

In non-protect submode, all character positions in the page, beginning with the current buffer address, are written to spaces.

In protect submode, all unprotected character positions in the page, beginning with the current buffer address, are written to spaces.

Execute Self-Test ... ESC z

Self-test does a test of the RAM and ROM, and puts a pattern on the display. This pattern--containing all of the characters capable of being displayed as well as characters displayed with all of the video attributes--indicates that the self-test has passed. Symbols occupying the 32 control character positions in the ROM are not displayed. Any error conditions are displayed on the 25th line.

After execution of self-test, the original contents of memory are lost and the screen will contain the test pattern. The terminal will be returned to the mode (block or conversational) that it was in when the self-test was issued.

Exit Protect Submode ... ESC X

The 6520 is placed in non-protect submode within block mode. Upon entering non-protect submode, the following conditions are in effect.

- o Any printing in process is aborted .
- o All character positions are treated similarly; there is no concept of protected data.
- o All pages are cleared to spaces.
- o Page 1 is displayed.
- o Page 1 is selected.
- o The buffer address is set to row 1 and column 1 for all pages.
- o The cursor address is set to row 1 and column 1 for all pages.
- o The keyboard is locked.
- o The 25th line is cleared.
- o Insert mode is reset.
- o Local line editing is enabled.
- o All horizontal tab stops are cleared.

Fill Character (NUL) ... %00

The character is not stored in the buffer.

<u>Insert Character ... ESC O</u>

In non-protect submode, all characters in the line, starting at the current buffer address, are moved one position to the right.

In protect submode, all characters in the field, starting at the current buffer address, are moved one position to the right. If the address is in a protected field, no action is taken.

In both protect and non-protect submode, the rightmost character in the line or field is lost.

<u>Insert Line ... ESC L</u>

The line containing the buffer address and all following lines are pushed down one line. The 24th line will be lost and it is the responsibility of the application program to read and save the data on the 24th line. In protect submode, any field attribute for a field with a starting address in the 24th line is deleted from the attribute table and new entries will not be created in in the table.

<u>Interpret Next Character For Function (ESC) ... %33</u>

The terminal sets an internal flag that causes the character sequence following the %33 to be interpreted differently. This character sequence is known as an escape (ESC) sequence and you use it to control terminal functions.

<u>Lock Keyboard ... ESC c</u>

The keyboard is locked immediately; the cursor is not displayed on the screen; and all keys with the exception of RESET are disabled. The message KBD LOCK is displayed on the right-hand side of the 25th line. This sequence should be used whenever data is transmitted to or from the displayed page.

<u>Modem Disconnect ... ESC f</u>

The terminal places the Data Terminal Ready line into a low state for three seconds which causes the terminal to "hang up."

<u>One Second Delay ... ESC @</u>

The 6520 will stop processing the input stream for approximately one second; normal processing will continue after the delay.

<u>Print Page ... ESC 0</u>

The 24 lines on the display are sent to the printer. This function is available for printers that have an RS-232C signal interface. See the PROGRAMMING CONSIDERATIONS chapter of this publication for an explanation of controlling the printer.

Read Buffer (ESC <)

This escape sequence causes either the entire selected page to be transmitted to the application program (in non-protect submode) or all unprotected fields to be transmitted (in protect submode). The escape sequence must occupy the first two character positions in the "TEXT" buffer (see the PROGRAMMING CONSIDERATIONS chapter for an explanation of text transmission in block mode). The format of the data transmitted depends on the submode of the 6520.

In non-protect submode, the text consists of the displayable data, with any video attributes transmitted as their respective control sequences. Trailing spaces on a line are not transmitted. The individual lines are separated by CR characters.

```
-     Text for line 1
-
-
ESC}  Define video attribute
6  }
-  }
-     More text for line 1
-
CR    End of non-blank data for line 1
-     Text for line 2
-
      etc.
```

In protect submode, the text consists of the data stored in all unprotected fields, independent of whether these fields have been modified. The transmission starts with the first unprotected field on the selected page and continues through the last unprotected field on the page. The address transmitted for each field is the address of the first unprotected character of the respective field.

The format of the data transmitted is:

```
DC1}  Set buffer address
-  }  Address of 1st unprotected field
-  }
-     Text
-
-
DC1}  Set buffer address
-  }  Address of 2nd unprotected field
-  }
-     Text
-
      etc.
```

Trailing spaces in a field are not transmitted.

Read Cursor Address ... ESC a

The 6520 transmits the address of the cursor for the selected page to the application program. This escape sequence must be the only text in the TEXT buffer. The format of the transmission is:

char page cursor

where:

char	is an ASCII _ (%137)
page	is a byte that specifies the page number of the selected page
cursor	is a two-byte sequence specifying the current cursor position. The position is transmitted as two encoded ASCII characters representing the row and column respectively. The ASCII chararacters are derived from the respective absolute row and column numbers by adding %37. This results in space through 7 (%40 through %67) representing rows 1-24 and space through o (%40 through %157) representing columns 1-80.

Read with Address (ESC =)

This escape sequence is used to read selected areas of a page. The escape sequence must occupy the first two character positions in the 'TEXT' buffer and the sequence is immediately followed by two 2-byte buffer addresses which specify the starting and ending addresses (inclusive) of the area to be read. These six characters must be the only text in the buffer. The actual data transmitted and the format of the data depends on the submode of the terminal.

In non-protect submode, the format of the data is exactly the same as that in the Read Buffer case, except that instead of reading the entire page, only the specified data is read.

In protect submode, only data from fields with the Modified Data Tag set are transmitted, and then only fields within the range of the two buffer addresses specified. The transmission starts with the first field which begins at or after the starting buffer address. Hence, if a buffer address is given which is not at the beginning of a field, the transmission will not start until the next field. Note that both unprotected and protected fields may be transmitted if the particular fields have their respective MDT set. The format of the data transmitted is exactly the same as that in the Read Buffer case.

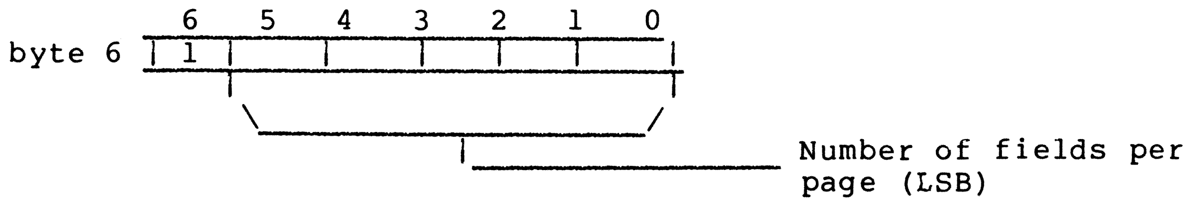
As in the Read Buffer case, in both submodes, the transmission of trailing spaces is suppressed.

Read Terminal Status ... ESC ^

The 6520 transmits its status to the host processor. The format of this message is:

```
-----------------------------------------------------------------------
|                                                                     |
|  SOH char status CR                                                 |
|  char is an ASCII ? (%77)                                           |
|  status is a six-byte sequence, with the following format:         |
|           6   5   4   3   2   1   0                                 |
|  byte 1 | 1 |   |   |   |   |   |   |                               |
|                                                                     |
|                       ^    ^  ^   ^                                 |
|               |     | |    |  |   |____ 1=Self-test passed          |
|                \___/  |    |  |   |____ 0=Self-test failed          |
|                  |    |    |  |________ 1=Power-up or self-test     |
|                  |    |    |  |________ 0=Conditions normal         |
|                  |    |    |___________ 1=Printer timed out         |
|                  |    |                   on last operation         |
|                  |    |    |___________ 0=last printer              |
|                  |    |                   operation sucessful       |
|                  |    |________________ 0=no printing in process    |
|                  |    |________________ 1=printing in process       |
|                  |_____________________ Unused                      |
|                                                                     |
|           6   5   4   3   2   1   0                                 |
|  byte 2 |   |   |   |   |   |   |   |                               |
|         |                           |                               |
|          \_________________________/                                |
|                      |_____________ Terminal Id                     |
|           6   5   4   3   2   1   0                                 |
|  byte 3 |   |   |   |   |   |   |   |                               |
|         |                         |                                 |
|          \_______________________/                                  |
|                    |______________Firmware revision level           |
|                                                                     |
|           6   5   4   3   2   1   0                                 |
|  byte 4 | 1 | 0 |   |   |   |   |   |                               |
|                 |                   |                               |
|                  \_________________/                                |
|                          |__________ Maximum page number            |
|                                                                     |
|           6   5   4   3   2   1   0                                 |
|  byte 5 | 1 | 0 |   |   |   |   |   |                               |
|                 |                 |                                 |
|                  \_______________/                                  |
|                          |__________ Number of fields per           |
|                                      page (MSB)                     |
|                                                                     |
-----------------------------------------------------------------------
```

Terminal Status Bytes - Part 1 of 2



Self-test passed = 1 when the last execution of self-test was successful.
Self-test passed = 0 when either the last execution of self-test was unsuccessful or if self-test has not been been run since the last power-up.

Power-up or self-test = 1 when either a power-up has occurred or self-test has been run. This bit is reset when status has been read successfully.

Printer status = 1 when a printer operation is in process and 0 when no printer operation is in process.

Terminal ID is a code which will be used to distinguish between various Tandem terminal products. For the initial product described in this specification, this code will be an "C".

Firmware Revision Level is a code which will be used to distinguish between various firmware revision levels of a given product. For the product described in this specification, this code will be a "C". The revision level will be changed each time the firmware is revised. This code is displayed in column 80 of the 25th line while self-test is executing.

Number of fields per page is a twelve-bit integer specifying the maximum number of fields that can be defined for each page in protect submode.

Terminal Status Bytes - Part 2 of 2

Reinitialize ... ESC q

The 6520 will initialize to the conditions set for entering block mode. They are:

- o Any printing in process is aborted
- o 1920 character pages are set
- o The non-protect submode is selected
- o The maximum page number is set to 7
- o Clear all pages to spaces; including page 1
- o Set video prior condition register for all pages to normal video
- o Display page 1
- o Select page 1
- o Set buffer address to row 1, column 1 for all pages
- o Set cursor address to row 1, column 1 for all pages
- o Lock keyboard
- o Clear 25th line
- o Reset insert mode
- o Enable local line editing
- o Initialize to ROM data-type table
- o Inhibit transmission from 6520 to CPU
- o Clear all horizontal tab stops

This sequence must be the only sequence in the TEXT buffer.

Reset Modified Data Tags ... ESC >

All modified data tags of all unprotected fields on the selected pages are reset. This escape sequence must be only text in the TEXT buffer. Modified data tags in protected fields are not changed.

Select Page ... ESC :

This sequence, followed by a single character, allows the application program to control which page is selected . Selected pages are where the input and output from the application program are directed to. The character starts at ! (page 1) and continues through the maximum number of pages available for display. After a page is selected, it remains selected until another select page sequence is received.

Set Buffer Address (DC1) ... %21

This code, followed by two bytes specifying the buffer address, sets the buffer address for the selected page. The buffer address specifies the area into which data will be stored and it is transmitted as two encoded ASCII characters representing the row and column respectively. The ASCII chararacters are derived from the respective absolute row and column numbers by adding %37. This results in space through 7 (%40 through %67) representing rows 1-24 and space through o (%40 through %157) representing columns 1-80.

Set Cursor Address (DC3)... %23

This code, followed by two bytes specifying the cursor address, sets the cursor address for the selected page. The cursor address is transmitted as two encoded ASCII characters representing the row and column respectively. The ASCII chararacters are derived from the respective absolute row and column numbers by adding %37. This results in space through 7 (%40 through %67) representing rows 1-24 and space through o (%40 through %157) representing columns 1-80. For example, to put the cursor on the beginning of the 24th row, issue DC3 followed by "7 ".

Set Maximum Page Number

This escape sequence is used to reduce the number of displayable pages below the maximum of seven. The format of the command is:

```
--------------------------------------------------------------------
|                                                                  |
|   ESC  p  n                                                      |
|                                                                  |
| where n is an ASCII number between 0 and 7.                      |
|                                                                  |
--------------------------------------------------------------------
```

You can use this sequence to increase the amount of data attribute space available in memory.

Because of the initialization process performed on entry into the protect submode, it is not possible to increase the maximum page number and the number specified must always be less than the previous value. Additionally, if the terminal is set to 960 character pages, the terminal will double the number, internally, before using it. Therefore, for example, setting the maximum page number to 5 in 960 character mode allows pages 1-10 to be used.

Set Page Size to 960 ... ESC t

The default upon entering block mode is 1920 character pages. This escape sequence sets the page size to 960 character pages; erases memory for all pages to spaces; and causes the terminal to default to the ROM data type attribute table.

Set Tab ... ESC 1

Tabs are set at the column of the current buffer address for all lines on the screen. The tabs do not take up space on the screen.

Set Video Attribute ... ESC 6

Video attributes are modified by issuing ESC 6 followed by an ASCII character corresponding to the binary values below.

```
----------------------------------------------------------------------
|                                                                    |
|                                                                    |
|        6   5   4   3   2   1   0                                   |
|       | 0 | 1 |   |   |   |   |   |                               |
|                  ^   ^   ^   ^   ^______ normal/dim                |
|                  |   |   |   |__________ normal/blinking           |
|                  |   |   |______________ normal/reverse            |
|                  |   |__________________ normal/blank              |
|                  |______________________ normal/underscore         |
|                                                                    |
|                                                                    |
|                                                                    |
----------------------------------------------------------------------
```

For example; to turn on a reverse video blinking field, issue ESC 6 & (& is the ASCII character code corresponding to binary 0100110).

An attribute takes up a character position on the screen and is displayed as a blank with the same attribute as the field it is defining with one exception;the underscore is not turned on until the character following the attribute is encountered.

The video attribute remains in effect until the next attribute is encountered: left-to-right and top-to-bottom on the screen. When video attribute rolls off the top of the screen, the rest of the characters return to normal until another video attribute is encountered.

This sequence can be used to store attributes anywhere on the page in both the protect and non-protect submodes. In protect submode, it is possible to define several video attributes within a field or to change the video attribute of a field without changing the data attributes.

You should note that if a video attribute is turned on within an unprotected field and the character position occupied by the attribute is not protected, the user can type over the position.

Set Video Prior Condition ... ESC 7

Issuing ESC 7 followed by a video attribute character, as described in the previous section, will cause the terminal to load the video prior condition register with the specified attribute and it will remain in effect for the entire page unless another attribute is encountered. The attribute does not occupy a position on the screen. When memory is cleared, the attribute remains in effect and can be changed only by issuing another ESC 7. Upon power-up, the video prior condition register is initialized to normal video.

A value for the prior condition register is maintained for each page, just as the cursor position is maintained for each page. When a new page is displayed, this value is loaded into the hardware register. Upon entering Block Mode, these save locations are all initialized to normal video.

Simulate Function Key ... ESC d

Issuing this escape sequence followed by a single character will cause the terminal to generate a function key sequence using the character as the key indicator. For example; issuing

```
   ESC d X
```

will cause the terminal to send

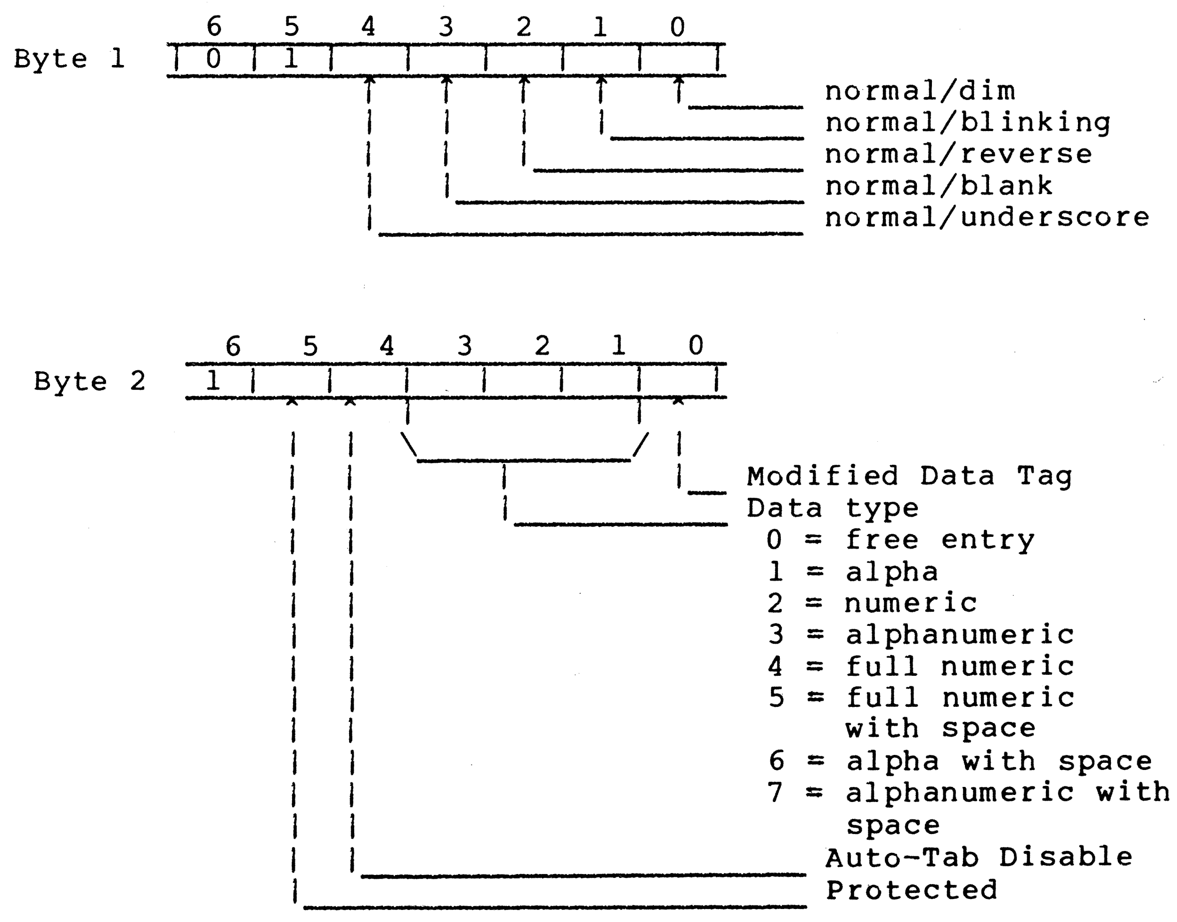
```
       X page cursor
```

Sound Audible Alarm (BEL) ... %07

When this code is received, the terminal will sound its audio tone.

Start Field (GS) ... %35

In protect submode, you use the start field control code, followed by two bytes of attributes, to define both video and data attributes for fields on the page. This code can also be used change attributes after the initial definition. The attributes that can be defined are illustrated below.



Protect

The protect attribute defines the field as a protected field. When fields are defined as protected, they cannot be modified from the keyboard and the cursor cannot be positioned into their area.

Auto-Tab Disable

Upon keyboard entry into the last character of an unprotected field, the cursor normally automatically tabs to the first position of the next unprotected field. If the Auto-Tab Disable is set for a field and data is entered into the last position of the field, the cursor advances to the next position on the screen: by definition a protected area. Although it is not necessary to have a protected field of non-zero length between adjacent unprotected fields, there will always be at least one space between fields to account for the video attribute stored for the following field. Under any other circumstance it is impossible for the cursor to be placed in a protected area. While the cursor is in this position, further data input is inhibited. As soon as the user takes some action to move the cursor (e.g., tab, cursor movement, etc.) the cursor returns to its normal behavior. Normal usage would be to fill a field, then strike TAB to move the cursor to the start of the next field. In this way the user can strike TAB after every field, regardless of whether the field is full.

Modified Data Tag

Every field, whether defined as protected or unprotected, has a Modified Data Tag (MDT). The MDT is set whenever the field is modified from the keyboard, and hence the operator can set the MDT only for unprotected fields. Note that there are several ways that the operator may modify the field, including simply typing into the field, executing an ERASE LINE, ERASE PAGE, INS CHAR, DEL CHAR, etc. The MDT may be set in protected fields from the host processor. The MDT for all unprotected fields may be reset by use of an escape sequence from the application program. When the page is later read with a Read Modified escape sequence, the terminal will suppress tranmission of those fields which have not been altered by the operator. Since the MDT can be set in protected fields, it is also possible for the application program to read fixed data from a page, such as a form ID (which might be displayed with a "blank" video attribute). The latter could be used to guarantee screen integrity.

-->

Data Type

The data type attribute defines which of several possible sets of characters may be entered into a field. The attribute is 3 bits long and defines the allowable characters to be from one of eight possible sets. The sets are defined by a 96-byte table which has one byte for each character from space through DEL. For a given entry, if bit n is a "1", then this character is a member of set n and is allowable in a field of data type n. Note that a character may be a member of several sets.

A data type table is stored in the ROM. See Appendix A for a description of the table contents.

For your convenience, the data types are summarized below.

Data Type	Use	Upper-Case Alpha	Lower-Case Alpha	Numeric	.,+-$	Space	Other
0	Free Entry	X	X	X	X	X	X
1	Alpha	X	X				
2	Numeric			X			
3	Alphanumeric	X	X	X			
4	Full Numeric			X	X		
5	Full Numeric with space			X	X	X	
6	Alpha with space	X	X			X	
7	Alphanumeric with space	X	X	X		X	

Start Field Extended ... ESC [

This escape sequence is used to perform exactly the same function as the Start Field sequence, with the exception that the upshift and reserved bits can also be set in the data attribute. The escape sequence is followed by three bytes, the first two of which have the same format as shown for Start Field. The third byte is

Table 3-4. Extended Field Attributes for Protect Mode

```
---------------------------------------------------------------------
|                                                                   |
|            6   5   4   3   2   1   0                              |
|   Byte 3  | 1 | 0 | 0 | 0 | 0 |   |   |                            |
|                                 x   x                              |
|                                 |   |_____ 1 = upshift             |
|                                 |_________ 1 = reserved            |
|                                                                   |
|   Upshift ... All alphabetic characters entered into              |
|               unprotected fields from the keyboard                |
|               to be upshifted.                                    |
|   Reserved .. Not used                                            |
|                                                                   |
|                                                                   |
---------------------------------------------------------------------
```

Unlock Keyboard ... ESC b

The terminal unlocks the keyboard immediately. Although the user can then enter data on the screen, any action that causes a transmission to the CPU is delayed until the terminal has received an ENQ.

DESCRIPTION

This chapter contains the information that you need to:

- o Access the terminal
- o Set device dependent functions, including mode switching, for the terminal
- o Use file management procedures to write data to and read data from the terminal.
- o Understand function key operations
- o Understand cursor positioning and buffer addressing in block mode
- o Write data to the 25th line
- o Understand the error control protocol followed for the terminal operation.
- o Control the printer

ACCESSING THE TERMINAL

Your application program communicates with the terminal through:

o $<device name>
or
o $<logical device number>

To obtain the device number of the home terminal where an application process is created, use the MYTERM procedure. To open the terminal for input and output, use the OPEN procedure. The following example illustrates the use of MYTERM and OPEN.

```
Int  .TERM^NAME[0:11],
      TERM^NUMBER;

Call MYTERM (TERM^NAME);            !returns $<device name>
Call OPEN (TERM^NAME,TERM^NUMBER);  !returns filenumber
```

SWITCHING MODES

To set device dependent functions, including mode switching, for the terminal, use the SETMODE procedure as described in the file management section of the GUARDIAN OPERATING SYSTEM PROGRAMMING MANUAL. The functions, in this procedure, applicable to the 6520 are numbered:

```
  6 - line printer or terminal, set system spacing control
  7 - terminal, set system autolinefeed
  8 - terminal, set system transfer mode

      <parameter 1> = 0 for conversational mode
                      1 for block mode
  9 - terminal, set interrupt characters (conversational mode)
 10 - terminal, set parity checking by system
 11 - terminal, set break ownership
 12 - terminal, set terminal access mode
 20 - terminal, set system echo mode
 22 - terminal, set baud rate
 24 - terminal, set parity generation by system
 27 - line printer or terminal, set system spacing mode
 28 - terminal, reset to configured values
```

For example, to set the terminal in block mode from your application program, issue:

```
    CALL SETMODE (TERM^NUMBER,8,1)
```

Note that the mode change will not occur until the first I/O operation to the terminal.

USING FILE MANAGEMENT PROCEDURES

To read data from the terminal and to write data to the terminal, from your application program, use the following file management procedures.

o READ	is used to return data to an array in the application program's data area. In conversational mode, this procedure is used to read data entered by the operator at the cursor position. In block mode, this procedure is only used to read the function key pressed by the operator to determine the next operation to be performed by the application. Note that the keyboard will be locked after the function key is pressed.
o WRITE	is used to write data from an array to the terminal. In block mode, this procedure is typically used to write a form or prompt to the terminal and to unlock the keyboard.
o WRITEREAD	is used to write data from an array in the application program's data area and then wait for data to be transferred back to the array. In conversational mode, this procedure is used to write a prompt to the screen and read the operator's reply. In block mode, this procedure is used to send control characters that the terminal has to respond to and then to read information returned by the terminal. The control characters allowed are:

- Read buffer (ESC <)
- Read cursor address (ESC a)
- Read terminal status (ESC ^)
- Read with address (ESC =)
- Simulate function key (ESC d)

Each of the preceding escape sequences will cause the terminal to immediately return the specified information to the application program. Typically, after the operator has entered information and pressed a function key signalling end of entry, READ would be used to determine when the operator has completed data entry and which function key was pressed, and WRITEREAD would be used to read all of the data entered by the operator. To summarize, think of using WRITEREAD when you expect the terminal to respond with information; use READ to read the function key and use WRITE to put your data on the screen for an operator response. Figure 4-1 illustrates this sequence.

```
-----------------------------------------------------------------------
|                                                                     |
|                                                                     |
|   WRITE ------------> PUT DATA ON SCREEN                            |
|   WRITE ------------> UNLOCK KEYBOARD                               |
|   READ -------------> READ FUNCTION KEY                             |
|   WRITEREAD --------> READ DATA ON SCREEN                           |
|                                                                     |
-----------------------------------------------------------------------
```

Figure 4-1. Block Mode Procedure Sequence

The information that your program passes to these procedures can include text, control codes and escape sequences. See the file management section of the GUARDIAN OPERATING SYSTEM for a complete description of the procedures.

Since the operating system protocols for block and conversational modes are different, the following examples illustrate use of the file management procedures in both modes.

BLOCK MODE EXAMPLES

This example illustrates the following sequence of operations in block mode:

1. Using WRITE to write information to the screen - two prompts.
2. Using WRITE to write a message to the 25th line.
3. Using WRITE to unlock the keyboard and allow the operator to reply.
4. Using READ to read the function key pressed by the operator.
5. Using WRITEREAD to read data entered by the operator at a specified position on the screen.

Each example uses the following declaration and assumes that the terminal is in the protect submode of block mode.

```
LITERAL ESC       = %033,      !ESCAPE FUNCTION
        READADRS  = "=",       !READ WITH ADDRESS FUNCTION
        UNLOCK    = "b",       !UNLOCK FUNCTION
        GS        = %035,      !START FIELD FUNCTION
        LINE25    = "o"        !WRITE TO 25TH LINE FUNCTION
        DC1       = %021,      !SET BUFFER ADDRESS FUNCTION
        REVERSE   = %044,      !REVERSE VIDEO ATTRIBUTE
        PROTECT   = %140,      !PROTECTED FIELD
        BLINK     = %042,      !BLINKING VIDEO ATTRIBUTE
        UNPROTECT= %104,       !UNPROTECTED NUMERIC FIELD
        UNBLINK = %040;        !STOP BLINKING FIELD VIDEO
DEFINE  ROW(A) = A  + %37#,
        COLUMN(A) = A + %37#;
INT     TERM^NUMBER,
         BUF[0:40];
STRING  SBUF = BUF;
```

<u>Writing Protected and Unprotected Fields to the Screen</u>

```
SBUF ':=' [GS,REVERSE,PROTECT,"ENTER ITEMNO:"];

!Start Reverse Video Protected Field

SBUF[16]':=' [GS,BLINK,UNPROTECT,"___",GS,UNBLINK,PROTECT];

!Start/end  Blinking/Unblinking Unprotected Field

CALL WRITE(TERM^NUMBER,BUF,24];

!Write First Prompt to Row 1 Column 1

SBUF ':=' [DC1,ROW(3),COLUMN(1),GS,REVERSE,PROTECT,"ENTER CODE:"];

!Start Second Reverse Video Protected Field

SBUF[17]':=' [GS,BLINK,UNPROTECT,"___",GS,UNBLINK,PROTECT];

!Start/end  Blinking/Unblinking Unprotected Field

CALL WRITE(TERM^NUMBER,BUF,26];

!Write Second Prompt to Row 3 Column 1
```

In this example, two reverse video prompts (ENTER ITEMNO: and ENTER CODE:) are written to rows 1 and 3 of the screen as protected fields. Each prompt is followed by a 3-byte blinking unprotected field (___); only numeric data will be allowed in these fields. After the writes, the cursor will be in the first position of the first unprotected field.

<u>Writing a Message to the 25th Line</u>

```
SBUF ':=' [ESC,LINE25,"PRESS ANY FUNCTION KEY FOR ENTRY",%12];
                                                         !  ^
!Message for 25th line.                                  !  |
                                                         !  Terminating
CALL WRITE (TERM^NUMBER,BUF,35);                         !  Seqeunce

!Write message to 25th line
```

In this example, the message (PRESS ANY FUNCTION KEY FOR ENTRY) is written to the 25th line.

Unlock the Keyboard

```
SBUF ':=' [ESC,UNLOCK];
CALL WRITE (TERM^NUMBER,BUF,2);                !Unlock Keyboard
```

In this example, the first two bytes of data in the application program's array are written to the terminal and the terminal is unlocked.

Read the Function Key

```
CALL READ (TERM^NUMBER,BUF,4);                 !Read Function Key
```

In this example, 4 bytes of data are read into the application program's array: the char, page and cursor as discussed in the function key section of this chapter.

Read Data Entered by Operator

```
SBUF ':=' [ESC,READADRS,ROW(1),COLUMN(15),ROW(1),COLUMN(17)];

          !Starting Address is row 1, column 15
          !Ending Address is row 1, column 17
          !Add Positions for Video Attributes

CALL WRITEREAD (TERM^NUMBER,BUF,6,6);
!First three characters returned are control code + addresses
```

In this example, the read with address function is written to the screen and six bytes of data are read back into the application program's array. The six bytes are: DC1, the address of the data, and the three characters entered by the operator.

CONVERSATIONAL MODE EXAMPLES

This example illustrates the following sequence of operations in conversational mode.

- o Issue a prompt and read the first line of data entered by the operator.
- o Write a line of data to the screen.
- o Read all of the data on the screen.

Each example uses the following declarations and assumes that the terminal is in conversational mode.

```
INT     BUF[0:40],
        TERM^NUMBER;
STRING  SBUF = BUF;
```

Issue a Prompt and Read the First Line of Data

```
SBUF := "?";
CALL WRITEREAD (TERM^NUMBER,BUF,1,80);
```

In this example, a one byte prompt (?) is written to the screen and the first line of the reply is written back to the application program's array.

Write a Line of Data to the Screen

```
SBUF ':=' "WHAT ARE YOU READING THIS FOR, THIS IS ONLY FILLER!"
CALL WRITE (TERM^NUMBER,BUF,50);
```

In this example, the first 50 bytes of data in the array are written to the screen.

Read all of the Data on the Screen

```
DO
 BEGIN
   CALL READ (TERM^NUMBER,BUF,80);
   continue to process input data
   END
UNTIL !END OF FILE!;
```

In this example, lines of data entered by the operator are read into the input array until the operator signals end of file.

FUNCTION KEY OPERATION

The terminal has 16 function keys (F1-F16) that can be operated shifted or unshifted, providing a total of 32 functions. The INSERT/DELETE LINE key can have its local function disabled and be used as a function key. In addition, in Block Mode the ROLL UP, ROLL DOWN, NEXT PAGE, and PREV PAGE keys have no local function but are treated as additional function keys. In a polling environment, the BREAK key acts as an additional function key.

Depression of a function key locks the keyboard and transmits the sequence:

```
char page  cursor
```

where: char is from Table 4-1

Table 4-1. Function Key Operations

Key	unshifted	shifted
F1-F16	@-O	`-o
ROLL UP	P	p
ROLL DOWN	Q	q
NEXT PAGE	R	r
PREV PAGE	S	s
INSERT/DELETE LINE	T	t
BREAK (polling only)	U	u

page is the page number of the displayed page

cursor is a two-byte sequence specifying the current cursor position. The format of these two bytes is described in the buffer addressing section.

BUFFER ADDRESSING IN BLOCK MODE

When functions are performed from the keyboard or remotely in Conversational Mode, any reference to screen location is via the cursor position.

In Block Mode, however, memory addressing is more complex as the concept of buffer addressing is introduced. Some of the terminal functions make use of the cursor address, while others make use of an independent buffer address. The rules which are followed are summarized below:

1. All keyboard operations take place at the cursor position.
2. All remote operations that are explicit cursor movement operations (such as cursor home, line feed, etc.) make use of the cursor position.
3. All remote operations that do not explicitly involve the cursor (such as set tab, erase page, etc.) make use of the buffer address.
4. All data input from the keyboard makes use of the cursor position, with the cursor advancing after each character.
5. All data input from the host makes use of the buffer address, with the buffer address advancing after each character.
6. A significant difference between cursor positioning and buffer addressing occurs in protect submode. The cursor may never be placed into a protected area, either from the keyboard or from a program. However, buffer addressing is always sequential, and programs can write into any area of the memory, protected or unprotected.

All cursor and buffer addresses are passed between the system and the terminal as two encoded ASCII characters, representing the row and column. The ASCII characters are derived from the respective absolute row and column numbers by adding %37. This results in space through 7 (%40 - %67) representing rows 1-24, and space through o (%40 - %157) representing columns 1-80. The first character always represents the row, and the second the column.

A summary of the action of each of the control sequences is shown in Table 4-2.

Table 4-2. Block Mode Addressing

Function	Keyboard	Host	
		Non-Protect	Protect
Set Tab	Cursor	Buffer Addr	N.A.
Clear Tab	Cursor	Buffer Addr	N.A.
Clear to Spaces	N.A.	Spec. Range	Spec. Range
Erase Line/Field	Cursor	Buffer Addr	} Buffer Addr.,
Erase Page	Cursor	Buffer Addr	} Unprotected
Insert Character	Cursor	Buffer Addr	} fields only,
Delete Character	Cursor	Buffer Addr	} ignored if Buffer
			} Addr. in
			} protected area
Start Field	N.A.	N.A.	Buffer Addr
Start Field Ext.	N.A.	N.A.	Buffer Addr
Set Video Attr.	N.A.	Buffer Addr	Buffer Addr
Insert Line	Cursor	Buffer Addr	N.A.
Delete Line	Cursor	Buffer Addr	N.A.
Backspace	Cursor	Cursor	Cursor}
Horizontal Tab	Cursor	Cursor	Cursor}
Line Feed	Cursor	Cursor	Cursor}
Carriage Return	Cursor	Cursor	Cursor}
Back-Tab	Cursor	Cursor	Cursor}-Unprotected
Cursor Up	Cursor	Cursor	Cursor} fields only
Cursor Right	Cursor	Cursor	Cursor}
Cursor Home	Cursor	Cursor	Cursor}
Cursor Home Down	Cursor	Cursor	Cursor}

WRITING DATA TO THE 25th LINE

The terminal is capable of displaying 25 lines on the screen. The normal text is displayed on the first 24 lines, and the 25th line is reserved for status information. There is only one 25th line, and it is stored in a separate area of memory independent of the other pages. The 25th line is formatted as follows.

```
                                         6 6 6              7 7 8
 1 2                                     5 6 7              8 9 0
| |AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA| |BBBBBBBBBBBBBBBBB| |C|
```

Text may appear in field A (columns 2-65) from two sources. If the terminal is operating in Block Mode, Protect submode, certain error messages may be displayed as a result of an attempt to enter invalid data into a field (e.g., numeric data in an alpha-only field, etc.). These messages appear and are erased automatically by the terminal

Application programs can write text into field A by sending ESC o, followed by the text to be displayed (up to 64 characters), and terminated by a CR or any sequence except ESC 6. The message is cleared by sending a null message. This escape sequence is described in the BLOCK and CONVERSATIONAL MODE chapters of this publication.

Field B (columns 67-78) is used for displaying terminal status information. Display of this information is initiated by depressing CTRL NEXT PAGE and disabled by depressing CTRL PREV PAGE.

Field C (column 80) is reserved for displaying the status of the Data Set Ready (CC) communications control line. An '*' is displayed if this line is high and a 'space' if it is low.

ERROR CONTROL

Error control depends on whether the terminal is in block or conversational mode.

In conversational mode, error control consists of parity checking if the parity switch is in the proper ODD or EVEN position.

In block mode, error control is in the form of LRC checking. It proceeds according to the following steps.

1. Each block is terminated with a single character LRC. On transmitting, the terminal generates the LRC and on receiving the terminal checks the LRC.

2. Both the host processor and the terminal acknowledge blocks with either an ACK or a NAK. Neither may send another block until the acknowledgement is received.

3. If the terminal receives a NAK it automatically retransmits the last block. It will do this continuously until an ACK is received,

or until receipt of an EOT.

4. If the LRC on a received message is in error, the VDU will NAK the block and wait for a retransmission.

During the receipt of a block or while waiting for an acknowledgement, the terminal sets up a time-out mechanism. If a received block terminates before the end, the terminal delays for 5 seconds. If no other character is received during this period, the terminal discards the partial block and waits for a completely new block. If an acknowledgement (ACK or NAK) is not received within 5 seconds, the terminal waits for a new message.

CONTROLLING THE PRINTER

Printing can be started in one of two ways.

1. The operator can press the print key. In conversational mode, the 24 lines on the display will be moved to a reserved page and the print process will start; the keyboard will not be locked. In block mode, printer pointers are set to the displayed page and the print process will start; the keyboard will be locked to prevent the operator to request an action that could modify the displayed page.

2. The application program can start printing a page by issuing the print page escape sequence. The process is the same as the one that occurs for pressing the print key with the exception of setting the print pointers to the selected page and not locking the keyboard.

You may encounter the following problems in attempting to control the printer from an application program.

Requesting a Print while a Print is in Process

When this situation occurs, the following actions are taken by the system.

- o The current print is aborted.
- o A CR is sent to the printer interface.
- o A form feed character is sent to the printer interface.
- o The audible alarm is sounded.
- o An error message (PTR ABORT) is displayed on the 25th line; it can be cleared by pressing any key.

To avoid this situation:

- o Lock the keyboard
- o Check the status bytes for a print in process.
- o Initiate a new print if there is no print in process.
- o Unlock the keyboard.

Modifying the Page being Printed while a Print is in Process

When this situation occurs, the following can occur. The printed page can contain some of the modified data. The modification to memory will be performed correctly and neither the operator nor the application program will be notified of the invalid print.

To avoid this situation:

- o Monitor your printing to avoid conflict by checking the status bytes before every print in block mode.

Unlocking the Keyboard While a Local Print is in Process

When this situation occurs, the operator can modify the page being printed.

To avoid this situation, check to determine if the selected page is not equal to the displayed page and avoid accepting any function key that changes pages until the printing is complete.

PREDEFINED DATA TYPE TABLE

Character		Data Type							
octal	Graphic	7	6	5	4	3	2	1	0
40	space	1	1						1
41	!			1					1
42	"			1					1
43	#			1					1
44	$			1	1				1
45	%			1					1
46	&			1					1
47	'			1					1
50	(			1					1
51	)			1					1
52	*			1					1
53	+			1	1				1
54	,			1	1				1
55	-			1	1				1
56	.			1	1				1
57	/			1					1
60	0	1		1	1	1	1		1
61	1	1		1	1	1	1		1
62	2	1		1	1	1	1		1
63	3	1		1	1	1	1		1
64	4	1		1	1	1	1		1
65	5	1		1	1	1	1		1
66	6	1		1	1	1	1		1
67	7	1		1	1	1	1		1
70	8	1		1	1	1	1		1
71	9	1		1	1	1	1		1
72	:			1					1
73	;			1					1
74	<			1					1
75	=			1					1
76	>			1					1
77	?			1					1
100	@			1					1
101	A	1	1			1		1	1
102	B	1	1			1		1	1
103	C	1	1			1		1	1
104	D	1	1			1		1	1
105	E	1	1			1		1	1
106	F	1	1			1		1	1
107	G	1	1			1		1	1
110	H	1	1			1		1	1
111	I	1	1			1		1	1
112	J	1	1			1		1	1
113	K	1	1			1		1	1
114	L	1	1			1		1	1
115	M	1	1			1		1	1
116	N	1	1			1		1	1
117	O	1	1			1		1	1

PREDEFINED DATA TYPE TABLE (Continued)

Character		Data Type							
Octal	Graphic	7	6	5	4	3	2	1	0
120	P	1	1			1		1	1
121	Q	1	1			1		1	1
122	R	1	1			1		1	1
123	S	1	1			1		1	1
124	T	1	1			1		1	1
125	U	1	1			1		1	1
126	V	1	1			1		1	1
127	W	1	1			1		1	1
130	X	1	1			1		1	1
131	Y	1	1			1		1	1
132	Z	1	1			1		1	1
133	[			1					1
134	\			1					1
135	]			1					1
136	^			1					1
137				1					1
140	`			1					1
141	a	1	1			1		1	1
142	b	1	1			1		1	1
143	c	1	1			1		1	1
144	d	1	1			1		1	1
145	e	1	1			1		1	1
146	f	1	1			1		1	1
147	g	1	1			1		1	1
150	h	1	1			1		1	1
151	i	1	1			1		1	1
152	j	1	1			1		1	1
153	k	1	1			1		1	1
154	l	1	1			1		1	1
155	m	1	1			1		1	1
156	n	1	1			1		1	1
157	o	1	1			1		1	1
160	p	1	1			1		1	1
161	q	1	1			1		1	1
162	r	1	1			1		1	1
163	s	1	1			1		1	1
164	t	1	1			1		1	1
165	u	1	1			1		1	1
166	v	1	1			1		1	1
167	w	1	1			1		1	1
170	x	1	1			1		1	1
171	y	1	1			1		1	1
172	z	1	1			1		1	1
173	{			1					1
174	|			1					1
175	}			1					1
176	~			1					1
177	DEL			1					1

Example

The character space is a member of sets 7, 6, and 0 (Hex-ASCII equivalent is Cl). To change it to be a member of sets 7, 6, 5, and 0 (Hex-ASCII equivalent is El), transmit ESC r El followed by 95 additional ASCII pairs for each of the characters in the table.

6520 Programming Functions - Part 1 of 2

Function	Controlled By	Mode
Back tab	ESC i	B
Clear all tabs	ESC 3	B
Clear tab	ESC 2	B & C
Clear to spaces	ESC I	B
Cursor down one line	code %12 (LF)	B & C
Cursor home	ESC H	B & C
Cursor home down	ESC F	B & C
Cursor left one space	code %10 (BS)	B & C
Cursor right one space	ESC C	B & C
Cursor to beginning of current line	code %15 (CR)	B & C
Cursor to next tab stop	code %11 (HT)	B & C
Cursor up	ESC A	B & C
Define data type table	ESC r	B
Define return key	ESC u	C
Delete character	ESC P	B
Delete line	ESC M	B
Disable line editing	ESC N	B
Display page	ESC ;	B
Display text on 25th line	ESC o	B & C
Enter protect submode	ESC W	B
Erase to end of line	ESC K	C
Erase to end of line/field	ESC K	B
Erase to end of page	ESC J	B & C
Execute self-test	ESC z	B & C
Exit protect submode	ESC X	B
Fill character	code %00 (NUL)	B & C
Insert character	ESC O	B
Insert line	ESC L	B
Interpret next character for funtion	code %33 (ESC)	B & C
Lock keyboard	ESC c	B
Modem disconnect	ESC f	B & C

NOTE: B = block mode; C = conversational mode

6520 Programming Functions - Part 2 of 2

Function	Controlled By	Mode
Next page	ESC U	C
One second delay	ESC @	B
Previous page	ESC V	C
Print page	ESC 0	B & C
Read buffer	ESC <	B
Read cursor address	ESC a	B
Read terminal status	ESC ^	B & C
Read with address	ESC =	B
Reinitialize	ESC q	B
Reset modified data tags	ESC >	B
Roll up	ESC S	C
Roll down	ESC T	C
Select page	ESC :	B
Set buffer address	code %21 (DC1)	B
Set cursor address	code %23 (DC3)	B
Set line width to 40	ESC 8	C
Set line width to 80	ESC 9	C
Set maximum page number	ESC p	B
Set page size to 960	ESC t	B
Set tab	ESC 1	B & C
Set video attribute	ESC 6	B & C
Set video prior condition register	ESC 7	B & C
Simulate function key	ESC d	B
Sound audible alarm	code %07 (BEL)	B & C
Start field	code %35 (GS)	B
Start field extended	ESC [	B
Unlock keyboard	ESC b	B

NOTE: B = block mode; C = conversational mode

Accessing the terminal 4-1
Audible alarm
   control codes
      in conversational mode 2-11
      in block mode 3-17

Back tab sequence 3-4
BEL control code
   in block mode 3-17
   in conversational mode 2-6
Block mode
   function overview 3-1
   list of functions 3-2/3
Block mode capabilities
   cursor movement 1-9
   data attributes 1-10/11
   display control 1-8
   editing functions 1-10
   memory organization 1-7
   submodes 1-7
   video attributes 1-10/11
Block mode function description
   clear to spaces 3-4
   cursor down one line 3-5
   cursor home 3-5
   cursor home down 3-5
   cursor left one space 3-5
   cursor right 3-5
   cursor to beginning of line 3-5
   cursor to next tab stop 3-5
   cursor up 3-5
   define data type table 3-6
   delete character 3-6
   delete line 3-6
   disable line editing 3-6
   display page 3-7
   display text on 25th line 3-7
   enter protect submode 3-7
   erase to end of line/field 3-8
   erase to end of page 3-8
   execute self-test 3-8
   exit protect submode 3-8
   fill character 3-8
   insert character 3-9
   insert line 3-9
   interpret next character for function 3-9
   lock keyboard 3-9
   modem disconnect 3-9
   one second delay 3-9

print page 3-9
read buffer 3-9, 3-10
read cursor address 3-11
read terminal status 3-12/13
reinitialize 3-12
reset modified data tags 3-14
select page 3-14
set buffer adddress 3-14
set cursor adddress 3-15
set page size to 960 3-15
set tab 3-15
set video attribute 3-15
set video prior condition 3-16
simulate function key 3-17
sound audible alarm 3-17
start field 3-17/20
start field extended 3-4, 3-21
unlock keyboard 3-21
BS control code
in block mode 3-5
in conversational mode 2-3
Buffer addresses
passed to system 4-10
Buffer addressing
concepts 1-14
example 4-5
in block mode 4-9/10

Clear all tabs sequence
in block mode 3-4
in conversational mode 2-3
Clear memory to spaces sequence
in block mode 3-4
in conversational mode 2-3
Control codes and escape sequences
in block mode 3-4/21
in conversational mode 2-3/11
in passwords 1-13
overview 1-13
Controlling the printer
concepts 4-12/13
problems 4-12/13
Conversational mode
function overview 2-1/2
list of functions 2-2
Conversational mode capabilities
cursor movement 1-4
display control 1-4
editing functions 1-5
memory organization 1-3
submodes 1-4
video attributes 1-5
Conversational mode function description
clear all tabs 2-3

clear memory to spaces 2-3
clear tab 2-3
cursor down one line 2-3
cursor home 2-3
cursor home down 2-3
cursor left one space 2-3
cursor right 2-3
cursor to beginning of line 2-3
cursor to next tab stop 2-4
display text on 25th line 2-4
erase to end of line/field 2-4
erase to end of page 2-4
execute self test 2-5
fill character 2-5
function key header 2-5
interpret next character for function 2-6
modem disconnect 2-6
next page 2-6
previous page 2-6
print page 2-6
read terminal status 2-7
roll up 2-9
send ACK/acknowledge ENQ 2-9
set line width to 40 characters 2-9
set line width to 80 characters 2-9
set tab 2-9
set video attribute 2-10
set video prior condition 2-10
sound audible alarm 2-11

CR control code
in conversational mode 2-3

Cursor addresses
passed to system 4-10

Cursor down one line code
in block mode 3-5
in conversational mode 2-3

Cursor home down sequence
in block mode 3-5
in conversational mode 2-3

Cursor home sequence
in block mode 3-5
in conversational mode 2-3

Cursor left one space code
in conversational mode 2-3

Cursor movement
in conversational mode 1-4

Cursor positioning and buffer addressing
concepts 1-14
descritpion 4-10

Cursor right sequence
in block mode 3-5
in conversational mode 2-3

Cursor to beginning of line code
in block mode 3-5

in conversational mode 2-3
Cursor to next tab stop
in block mode 3-5
Cursor to next tab stop code
in conversational mode 2-4
Cursor up
in block mode 3-5

Data attributes
defining in a program - example 4-4/5
defining using control codes 3-18/20
description 1-11
Data type table
changing 3-6
description 5-1/2
Data types in block mode 3-18/20
DC1 control code
in block mode 3-14
DC3 control code
in block mode 3-15
Define data type table sequence
in block mode 3-6
Define return key sequence
in conversational mode 2-4
Delete character sequence
in block mode 3-6
Delete line sequence
in block mode 3-6
Disable line editing sequence
in block mode 3-6
Display control
in block mode 1-8
in conversational mode 1-4
Display memory size
in block mode 1-7
in conversational mode 1-3
Display page sequence
in block mode 3-7
Display text on 25th line
in block mode 3-7
in conversational mode 2-4

Editing Functions
in block mode 1-10
in conversational mode 1-4
Enter protect submode sequence
in block mode 3-7
Erase to end of line sequence
in conversational mode 2-4
Erase to end of line/field sequence
in block mode 3-8
Erase to end of page sequence
in conversational mode 2-4
Error control 4-11

ESC escape sequence
  in block mode 3-9
  in conversational mode 2-6
Escape sequences
  in block mode 3-4/21
  in conversational mode 2-3/11
Examples
  block mode
    explanation 4-4
    reading data entered by operator 4-6
    reading the function key 4-6
    unlocking the keyboard 4-6
    writing a message to the 25th line 4-5
    writing fields to the screen 4-5
    writing information to the screen 4-5
  conversational mode
    issuing prompts and reading data 4-7
    read data on screen 4-7
    writing a line of data to the screen 4-7
Execute self test sequence
  in block mode 3-8
  in conversational mode 2-5
Exit protect submode sequence
  in block mode 3-8

Features
  multiple memory pages 1-1
  operational modes 1-1
Field definition
  in block mode 3-18/21
Fill character code
  in block mode 3-8
  in conversational mode 2-5
Full duplex 1-3
Function key header code
  in conversational mode 2-5
Function key operation 4-8

Half duplex 1-3
HT control code
  in block mode 3-5

Input and output to a page
  buffer and cursor positions 4-9/10
  concepts 1-14
Insert character sequence
  in block mode 3-9
Insert line sequence
  in block mode 3-9
Interpret character for function sequence
  in block mode 3-9
  in conversational mode 2-6
Introduction 1-1/15

LF control code
   in block mode 3-5
   in conversational mode 2-3
List - Control Codes / Escape Sequences 6-1/2
Lock keyboard sequence
   in block mode 3-9

Memory organization
   in block mode 1-7
   in conversational mode 1-3
Modem disconnect sequence
   in block mode 3-9
   in conversational mode 2-6
MYTERM procedure 4-1

Next page sequence
   in conversational mode 2-6
NUL control code
   in block mode 3-8
   in conversational mode 2-5

One second delay sequence
   in block mode 3-9
OPEN procedure 4-1
Overview of application environment 1-1

Predefined data type table 5-1/2
Previous page sequence
   in conversational mode 2-6
Print page sequence
   in block mode 3-9
   in conversational mode 2-6
Printer interface 1-1
Printer problems 4-12/13
Programming concepts
   control codes and escape sequences 1-13
   conventions 1-15
   cursor positioning and buffer addressing 1-14
   selecting and displaying pages 1-14
   using file management procedures 1-12
Programming Considerations 4-1/13
Programming Functions 6-1/2
Protected fields
   definition 1-7/8
   setting in block mode 3-18/20
Protect submode
   description 1-7/8
   entering 3-7
   exiting 3-8

Read buffer sequence
   in block mode 3-10
READ procedure
   description 4-3

example 4-4/7
Read terminal status sequence
in block mode 3-12
in conversational mode 2-7
Read with address sequence
in block mode 3-11
Reading the function key 4-3/4
Reinitialize sequence
in block mode 3-14
Reset modified data tags sequence
in block mode 3-14
Roll up escape sequence
in conversational mode 2-8

Selecting and displaying pages
overview 1-14
displaying a page in block mode 3-7
selecting a page in block mode 3-14
Selecting submodes
in block mode 1-6
in conversational mode 1-3
Send Ack/acknowledge Enq escape sequence
in conversational mode 2-9
Set cursor address code
in block mode 3-15
Set line width to 40 characters sequence
in conversational mode 2-9
Set line width to 80 characters sequence
in conversational mode 2-9
Set maximum page number sequence
in block mode 3-15
Set page size to 960 sequence
in block mode 3-15
Set tab sequence
in block mode 3-15
in conversational mode 2-9
Set video attribute sequence
in block mode 3-16
in conversational mode 2-10
Set video prior condition sequence
in block mode 3-17
in conversational mode 2-11
SETMODE procedure 4-2
SOH control code
in conversational mode 2-5
Sound audible tone code
in conversational mode 2-11
Start field control code
in block mode 3-18/20
Start field extended sequence
in block mode 3-21
Status bytes
terminal
in block mode 3-12/13

        in conversational mode 2-7/8

Submodes
    description
        in block mode 1-7/8
        in conversational mode 1-4

Switching modes
    using SETMODE procedure 4-2

Unlock keyboard sequence
    in block mode 3-21

Using file management procedures
    description 4-3
    overview 1-12

Video attributes
    defining in a program - example 4-4/5
    description for block mode 1-10/11
    description for conversational mode 1-4
    setting - in block mode 3-16/21
    setting - in conversational mode 2-10/11

WRITE procedure
    description 4-3
    example 4-4/5

WRITEREAD procedure
    description 4-3
    example 4-4/5

Writing data to the 25th line 4-11

READER'S COMMENTS

Tandem welcomes your feedback on the quality and usefulness of its publications. Please indicate a specific *section* and *page* number when commenting on any manual. Does this manual have the desired completeness and flow of organization? Are the examples clear and useful? Is it easily understood? Does it have obvious errors? Are helpful additions needed?

Title of manual(s): ______________________________

____

____

____

____

FOLD ➤

____

____

____

____

____

____

____

____

____

____

____

FOLD ➤

FROM:

Name ______________________________

Company ______________________________

Address ______________________________

City/State ______________________________ Zip ______________

A written response is requested, yes no ?

— FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 482 CUPERTINO, CA. U S A

POSTAGE WILL BE PAID BY ADDRESSEE

TANDEM
COMPUTERS, INC.

Attn: Technical Publications
19333 Vallco Parkway
Cupertino, CA, U.S.A. 95014

— FOLD

STAPLE HERE