ALPHA MICROSYSTEMS
RIGHT. FROM THE START.

AMOS â Terminal System User's Guide

DSS-10096-00 A01

REVISIONS INCORPORATED	
REVISION	DATE
A00	March 1987
A01	June 1996

AMOS Terminal System User's Guide.

To re-order this document, request part number DSS-10096-00.

ALPHA MICROSYSTEMS
2722 S. Fairview St.
P.O. Box 25059
Santa Ana, CA 92799

TABLE OF CONTENTS

PREFACE

Surprisingly, the subject of computer display terminals can often be a controversial one: the many variations present in available terminals, combined with the aesthetic judgments required to determine when a display looks "right," often lead to unproductive arguments.

The importance of the subject matter cannot be overlooked, however. The issue of terminal independence and how it is acheived, if it is acheived at all, can often "make or break" a software package, determining whether it can successfully coexist with software already present on your system.

This book is intended to explore the various problems and possible answers, and to present the set of solutions chosen by Alpha Micro as the best available, based both on looking at where we came from, as well as keeping an eye to the future.

Additionally, it provides information on how you can design screen displays that are easier to use and more attractive, based on sound human engineering principles.

While written primarily for software developers, interested users of the Alpha Micro computer system with a general understanding of computers can benefit from reading Part I of this book, and quickly skimming Parts II and III.

We wish to express our thanks to those members of the Alpha Micro community who gave their input and support to the effort culminating in this book. By making hard decisions, and sticking to those commitments, they have made the jobs of all software developers easier and made the Alpha Micro a better system for all of its users.

PART I

GENERAL OVERVIEW

CHAPTER 1

INTRODUCTION

This book attempts to present a solution to a problem. As evidenced by the size of the book, the problem is not a simple one. We hope, however, that as you read it you will gain an appreciation not only of the problem, but also of the many benefits associated with the solution.

1.1°A STATEMENT OF THE PROBLEM

The problem we wish to address in this book has ancient roots, at least as far as the computer industry goes. Back in the very early days of interactive computing there was the Teletype ASR-33 terminal. This was a hardcopy (printing) terminal that had a top speed of 10 characters per second, had only upper case characters, and could do nothing more complex than simply put characters down on a piece of paper. Its only redeeming quality was that it was available in quantity and was "inexpensive" (relative to the times) at a cost of several thousand dollars.

As the market for interactive computing grew and as the cost of integrated circuits fell, there came an opportunity to replace the venerable ASR-33 with something better. What became available next was a very simple CRT (video) terminal which did nothing more than emulate an ASR-33 at somewhat lower cost and somewhat higher speed. In fact, these early CRT terminals were often referred to as "glass teletypes." These terminals emulated the ASR-33 so well, in fact, that all of the old software written for the ASR-33 worked just fine on the new CRT terminals. No problem so far, right?

Well, as the market grew and the cost of electronics continued to plummet, some enterprising engineers started to see ways to increase the functionality of the new terminals by adding more sophisticated features, such as cursor addressing and special erase commands. Of course once someone had seen software which made use of these features, nothing else was satisfactory, so users were soon demanding these features. And once one company had a terminal with one of the new features, the competition needed to create a terminal with just one more feature to make theirs a bit better, all of which resulted in providing the end user with a better, more fully function-ed terminal at a lower cost. Everything sounds great, right?

Well not quite. Besides all of the benefits of competition in action we listed above, one of its disad-vantages was also in action. As each manufacturer improved on its competition’s terminal, it also made some minor changes. These minor changes were always made for the best of reasons (efficiency, cost reduction, etc.), but they led to one inevitable result: none of these new terminals were compatible with each other.

Users, once having been exposed to them, demand the advanced features present in these new terminals. We are therefore forced to make application software which uses these features. However, the software written for one terminal will not function properly when used on a different terminal. And terminals keep changing so often, as they constantly improve, we cannot afford to be stuck with only one type of terminal lest we be left behind our competition.

Thus we have our problem: how do we create software which takes advantage of the power of the very latest terminal without forcing ourselves to constantly rewrite our software at every turn.

As we will see throughout this book, this is not a simple problem to solve, although it does have a rather elegant solution.

1.2 SOME POSSIBLE SOLUTIONS

Given a problem of this size and importance, it is not surprising that many different groups have tried to devise a solution. For purposes of illustration, let's go through a few of the solutions that have been attempted.

1.2.1 One Application, One Terminal

Certainly the most straightforward of the possible solutions is simply to restrict a given software package to one terminal. This has been most popular with software vendors that also sell terminals, for obvious reasons.

It has not met with a great deal of success, however, because not only are the users of the software package denied the opportunity to take advantage of the latest in terminal technology, the vendor of the software package is also unable to take advantage of it. The software package, therefore, quickly becomes obsolete.

1.2.2 One Application, Several Terminals

The next approach is to create the software package such that it can take advantage of two or three (or up to 20 or more) different terminals. This is typically done by actually coding into the application package the various code sequences needed for each of the terminals.

This approach has been more successful than the first, but simply delays the same result as the first.

1.2.3 Many Terminals, One Code

What would seem to be an excellent solution is the idea of creating many different terminals which all accept the same code sequences. This has been accomplished through the establishment of the ANSI X3.64 terminal command standard. As a national standard, it has attracted many different terminal vendors that all produce terminals conforming to this code. It would therefore seem to be a simple matter for a software package to simply use only these code sequences and still have a wide variety of terminals to choose from.

However, while ostensibly designed to allow software to have access to many different terminals, because it was being promulgated by terminal manufacturers, the standard was actually intended to increase the sales for terminals by allowing the sale of terminals to many different, partly incompatible computer systems.

Because many of the computer systems being produced have very restrictive rules about what can and cannot be sent to a terminal, the ANSI standard had to specify commands that could be sent from any computer system. It therefore resulted in an extremely inefficient set of commands— so inefficient, in fact, that most of the terminal manufacturers that were

pushing hard for the ANSI standard now offer terminals that support both the standard and an "efficient" mode. (The differences between these two modes can be quite dramatic: a difference of 3 characters or 10 for exactly the same command.)

Also, all this standard did was partially delay the same old problems. Because ANSI X3.64 is now several years old, many terminals that claim to adhere to the standard also contain many "extensions" which, of course, are not standardized. The standard also never addressed characters going from the terminal to the host as in function key input.

1.3 THE PHILOSOPHY OF TERMINAL DEVICE INDEPENDENCE

After reviewing all of the possibilities outlined above, and several others, it became clear that a different approach had to be taken for the Alpha Micro computer system. Intending to be around for a long time, and requiring high performance from all parts of the system, the existing solutions simply were not acceptable for the Alpha Micro computer.

The most important item in this entire book is the concept of *terminal device independence*. Simply stated, this concept allows the same applications software to be used on any terminal, without requiring any modification to that software whatsoever.

This is accomplished by having the software perform all terminal functions through a common, standardized interface. While this interface will evolve as terminals themselves evolve, it is always upwardly compatible, allowing any software which properly uses the standard interface to continue to function regardless of the particular terminal and its capabilities.

After accepting a command from a program through this standardized interface, the command is passed to the terminal "driver" program. It is the responsibility of this driver program to translate the command into the specific command sequence required by a given terminal.

As we have said before, achieving this goal of terminal independence is not a simple task, even though the concept itself is simple enough. It requires a rigid adherence to a particular set of rules. However, because software is written only once, while terminals change far more rapidly, the initial investment of effort in adhering to the rules is much smaller than the continual modification necessary if the software is written to be terminal dependent.

In the discussion thus far we have concentrated on the commands that must be transmitted from the host to the terminal. Equally important to any application package is the information that goes from the terminal to the host, such as function key input. Both directions must be standardized in order to completely insulate the software from the terminal.

The important thing to remember is that by maintaining terminal independence you preserve your investment in application software and ensure you benefit from the improved price/performance supplied by future terminals.

1.4 THE PHILOSOPHY BEHIND ALPHA MICRO TERMINAL SOFTWARE

The concept of device independence as applied to terminals is certainly not unique to Alpha Micro. What is unique, however, is the depth to which this independence has been built into the system, and the variety of different methods provided for achieving this independence.

The demands on this interface are quite extensive. Typical of the types of items it must provide standardized access to are:

- Cursor positioning.
- Clear screen.
- Insert and delete lines, characters, and columns.
- "Special effects" like dim, reverse video, blinking, or underlining.
- Putting text on one of the status lines.
- Selecting foreground and background colors.
- Selecting 80 or 132 column mode.
- Accepting input from function keys.

1.4.1 Dealing with the Compromises

All of this cannot, of course, be achieved without some compromises. Whenever a standard method of doing something must be set, there will inevitably arise situations where the standard method is not the best method. To help minimize these compromises, the Alpha Micro terminal interface is considered to be an *evolving* standard. That is, the standard is intended to provide a "best fit" with the features found in the majority of the *current* terminals.

This means that over time the terminal interface standard changes, usually in the area of assuming that additional features will be present. However, it is intended that these changes are made in such a way as to be "upwardly compatible," not forcing modification to software created prior to the change in the interface standard. Of course, this requirement in turn forces additional compromises, forcing further changes, thus creating a constantly evolving terminal environment.

This is not to say, however, that the terminal interface is a contantly changing amorphous mass. Instead, it evolves in distinct steps or levels as terminal technology advances. Rather than changing constantly, there really have been only three different levels of terminal support during the history of the Alpha Micro computer.

The first level dates back to the "glass teletype" days. Here software had to assume the only features present were cursor addressing, clear screen, erase to end of line, and erase to end of screen. In actuality, many of the terminals of that era did not even possess the erase to end of line and screen commands, but instead required that these be simulated by the terminal interface software.

Next came the level where it could be safely assumed that, in addition to the features listed above, the typical terminal also contained insert and delete line capability, dim mode, and some "special effects" like reverse video or blinking.

With the introduction of the AM-60 terminal in 1983, we entered our third level of terminal interface. Here it is considered likely that the terminal possesses all of the above features, but also has function keys, status lines, split screen and video on/off. The subsequent introductions of the AM-62, AM-62A, and Workstation terminals have introduced additional features such as switchable 80/132 column modes, color, and printer ports.

Note that because the feature set is getting richer over time, software written for an earlier set of features continues to function, if not quite as efficiently as it might if written with the latest features.

1.4.2 Dealing with Feature Differences

While a standard interface is provided to give software access to all of the possible features that a terminal might have, we still must deal with terminals that do not have all of those features. Three methods are used to deal with this.

The first is emulation of the feature. On many terminals it is possible to emulate the operation of a particular feature by combining several other features. This is the first choice whenever a feature is missing from a terminal as it preserves the most functionality.

Because it is not always practical to emulate a particular feature, we must sometimes use our second method, which is to simply ignore the command. This is particularly successful with features that may improve the appearence of the screen display (such as the "video off" command), but that have no material effect on the actual operation of the software.

However, we need a third method of dealing with features that cannot be practically emulated, but affect the operation of the software if they are not present. To handle this situation it becomes important for software to be able to detect the presence or absence of a feature in the terminal it is currently being run on. The software can then tailor its operation based on the presence or absence of a given feature. For this reason, a method is provided for determining not only a list of the features supported by the terminal, but also other characteristics of the terminal, such as the number of rows and columns on the screen and the current color selections.

For these methods to work, note that it is the responsibility of the software to check for the presence of a feature before it attempts to use it. If a given feature is marked as absent, the results of using that feature are unpredictable. Note, however, that because of our concept of "levels" of terminals it is not necessary for the software to be constantly checking each and every feature, unless it is desired that the software be able to be run on earlier level terminals.

1.5 RELATED DOCUMENTATION

While this book is intended to be largely self-contained, the following documents may also prove to be useful:

- ***Introduction to AMOS***
- ***AMOS Monitor Calls Manual***
- ***AlphaBASIC User's Manual***

In addition when writing a terminal driver for a particular terminal, you will need a copy of the programming manual for that terminal, available from the terminal manufacturer. The terminal driver source code contained on the standard AMOS release (stored in account DSK0:[10,2]) will also provide examples of how various features are handled.

CHAPTER 2

HOW AMOS INTERFACES TO TERMINALS

To connect your software to your terminal, AMOS uses several different layers of software and hardware. All of these layers together are referred to as the terminal service system. While these layers are intended to be transparent to the user and programmer, some background on the various pieces, and a description of the functions of each, can lead to a clearer understanding of how the terminal service system functions. This chapter attempts to provide that information.

2.1 THE HARDWARE INTERFACE

The very first thing your terminal connects to is the hardware interface contained within your Alpha Micro computer. This *serial interface* converts the signal voltages from your terminal to data which the computer can more easily understand.

Several different types of serial interfaces are available for the Alpha Micro computer, each having different capabilities, although the basic function performed by each is essentially the same.

In the majority of cases, neither the programmer nor the user of the Alpha Micro need be concerned with the particulars of the serial interface in use.

2.2 INTERFACE DRIVERS

Because there are a variety of different serial interfaces which can be used on the Alpha Micro, and because the AMOS system itself wishes to only see one type of interface, a special piece of software called an *interface driver* is used to shield AMOS from the different hardware configurations.

Once again, the user and programmer can usually remain blissfully ignorant of the details of the interface driver. It is mentioned here simply to make you aware of the many layers which comprise the terminal service system.

2.3 TERMINAL DRIVERS

Now that AMOS has gotten data from the terminal, through the serial interface, and through the interface driver, there is one last piece of software used to further shield your program from the many differences which exist in the real world of terminals. This last piece is referred to as the *terminal driver*.

It is the terminal driver's job to translate your program's terminal commands to the special code sequences required by the terminal, as well as to do any special processing of your keyboard input.

While in most cases the user and programmer can ignore the details of how the terminal driver works, it is important to keep in mind that the entire terminal independence idea hinges on the use of a properly written terminal driver, specifically written for the type of terminal in use, used in concert with applications software written to observe the conventions and features of the terminal driver.

We hope such a terminal driver is already available and installed on your system. If not, later sections of this book describe how to create a terminal driver. This task is best left to an experienced systems programmer, however.

2.4 STANDARDIZED TERMINAL INTERFACE

Via the different layers we have described, AMOS creates a standardized terminal interface for your program, isolating your program from the many different terminals it may have to function with. This standardized interface has two main features your program will use: the TCRT calls, and the TRMCHR call.

2.4.1 The Monitor TCRT Calls

The individual functions the terminal driver supports are implemented via the TCRT call. This call passes the user's terminal command, whether a cursor position or a clear screen, to the terminal driver where it is translated into the special code sequence required by the particular terminal.

The TCRT call takes two arguments, row and column. The row argument has a value range of –127 to +128 and the column argument has a value range of 0 to +255. Whenever the row argument is positive, the row and column arguments are treated as a cursor position, and the cursor is moved to the position specified by the arguments.

When row is negative, the column argument is taken as a special function code and interpreted accordingly. The most common negative value of row is –1, which corresponds to the most commonly used set of terminal functions. Other negative values are used to flag different commands.

Therefore, when we speak of TCRT –1,12, we are describing a specific terminal function in the row –1 range and the specific command of 12.

2.4.2 The TRMCHR Call

The TRMCHR (terminal characteristics) call provides a standardized way for programs to determine the characterisitics of the terminal they are dealing with. TRMCHR returns such information as the current size of the screen in rows and columns, the features contained within the terminal, the current color selections, and several other descriptive items.

By providing this one method, information can be formatted and returned by the terminal driver in a consistent manner, regardless of what operations may be necessary to obtain the information.

2.4.3 Higher Level Language Interfaces to the Terminal

In addition to the assembly language monitor calls interface we have discussed, most of the higher-level languages provide ready access to the terminal interface system. AlphaBASIC™, for example, provides access to the TCRT calls themselves throughs its PRINT TAB statement, and provides access to the TRMCHR information via an XCALL subroutine (TRMCHR.SBR). Other languages provide similar interfaces.

It is important, regardless of the language, that all software have access to the standard terminal interface provided by TCRT and TRMCHR to allow us to achieve our goal of terminal independence.

CHAPTER 3

INTRODUCTION TO TERMINALS

3.1 CONVERSATIONAL VS. BLOCK MODE

Conversational mode is a method of terminal operation where information is sent to and received from the terminal on a character by character basis. In conversational mode all editing and display is controlled by the host computer with no intervention by the terminal itself. This is the mode in which the Alpha Micro computer and terminals were designed to work. This method of operation provides the most flexibility by placing the various editing functions in the host computer. This avoids the complications that arise when trying to handle a special editing task for which a terminal was not specifically designed.

Block mode is a method of handling terminals first used on large mainframe systems as a way of giving the appearance of interactive applications without actually disturbing the underlying batch nature of the large systems. Rather than sending individual characters as they are typed as in conversational mode, in block mode individual fields, and even whole screens, are stored up within the terminal and sent as a "block." This method may have been attractive on some older systems, but it suffers from some severe limitations in an interactive environment. Because all editing is performed within the terminal, software designers are limited to providing only that editing, regardless of the special needs of an application. In addition, because each terminal has been designed with a unique set of editing capabilities, over which the host has no control, it becomes exceedingly difficult to overcome these differences and provide a terminal independent environment.

For example, a given block mode terminal may only support the insert or delete character operations affecting an entire line and thus be unable to use them when several fields occur on the same line of the screen.

Applications software which uses block mode is also hard to integrate with the majority of other AMOS software, which is conversational mode based. For all of these reasons, block mode is not often used with AMOS, and its use with new applications is discouraged.

3.2 FIELD VS. MODE TERMINALS

Terminals handle screen display attributes, such as underscore, reverse video, or blinking, in one of two ways: field or mode.

In a field terminal, you begin an attribute "field" by issuing a special command. From that point on, progressing left to right, down the screen, all text on the screen will be displayed using the new attribute until either the end of the screen is hit, or another attribute is encountered. You can think of these attributes as being special characters which "turn on" and "turn off" the screen

attribute as the text is displayed on the screen. Writing additional characters to the screen between the begin and end attribute characters will cause the characters to be displayed with the screen attribute. Writing additional characters before the begin attribute character or after the end attribute character will cause them to be displayed normally.

In a mode terminal, all characters you write to the screen are written with the screen attribute corresponding to the current "mode" you are in. Upon entering "reverse video mode," for example, all characters you write will be displayed in reverse video, regardless of their location. This will continue until you select a different attribute mode for writing.

3.2.1 Compatibility Issues

Field terminals are still the most common as they were the first type of terminal to be widely used. Most of the very latest terminals, however, are being built as mode terminals. Because of the differences in the behavior of these two terminal types, special care must be taken when creating software which must run on both types, an important feature because terminals are likely to switch from field to mode terminals almost entirely over the next few years.

3.2.1.1 To Space or Not to Space

One of the major differences between mode and field terminals is how the attributes themselves appear on the screen. In a field terminal the attribute typically occupies a screen position, appearing as a space (a "non-hidden" attribute). In a mode terminal, however, the attribute is typically invisible and does not occupy a screen position (a "hidden" attribute). This presents an obvious problem when you wish to use either type of attribute and be able to line up text on the screen.

To solve this problem, the standard set of attribute commands implemented in the AMOS terminal system interface always occupies a screen position for all attribute changes. This screen position is taken regardless of whether the terminal hardware requires it, even if it means that the terminal driver must output a dummy space.

By always assuming that a space will be present, text can be assured of properly lining up on the screen, regardless of the number of attribute changes that may or may not be present.

For those applications which require the use of hidden attributes, such as underlining text in the middle of a word, special terminal system calls are available to access attributes without spaces on those terminals which support the feature. Because hidden attributes are not available on all terminals, however, use of them restricts your software to a smaller set of terminals.

3.2.1.2 Screen Flash

An annoying problem with field attribute terminals is the phenomena known as "screen flash." This problem arises when an attribute momentarily occupies a large area of the screen in the brief interval between issuing a "begin attribute" and an "end attribute" command.

To avoid this problem, which is purely an aesthetic (but very visible) one, the following procedure has been worked out. This procedure works equally well with mode terminals

which would not normally exhibit the screen flash problem.

1. Position the cursor at the end of the area to be written with text.

2. Issue an "end attribute" command for the attribute you are going to be using. (Mode terminals will ignore this.)

3. Position the cursor at the beginning of the area to be written with text.

4. Issue a "begin attribute" command for the attribute you wish to use. (This will start the attribute for both mode and field terminals.)

5. Write the text.

6. Issue an "end attribute" command, again. (This will end the attribute for mode terminals.)

PART II

DESIGNING EFFECTIVE SCREEN DISPLAYS

CHAPTER 4

PRINCIPLES OF EFFECTIVE SCREEN DESIGN

That the software we develop should be easy to use seems obvious; it goes without saying. Yet the computer industry has acquired a reputation for creating systems and products that are difficult, frustrating, and tedious to use. While undoubtedly deserved in some cases, this reputation does no one any good. Only by creating systems that are a pleasure to use will we be able to eradicate this image. This chapter will show some of the things we can do to try to acheive this goal.

4.1 WHAT WE ARE LOOKING FOR

Of course the ideal system has probably not yet been built. And yet, partly through experience with bad systems, we can agree on some characteristics of a good system. Below is a list of some of these characteristics.

Adaptable. A system must be adaptable to the physical, emotional, intellectual, and knowledge traits of the people whom it serves. All office workers should be permitted to interact with a computer in a manner and style which best suits their needs. In essence, the system should be responsive to individual differences in interaction manner, depth, and style.

Transparent. A system must permit one's attention to be focused entirely on the task being performed, without concern for the mechanics of the interface. One's thoughts must be directed to the application, not the communication. Any operations which remind a worker of their presence are distracting.

Comprehendible. A system should be understandable. A person should know what to look at, what to do, when to do it, why to do it, and how to do it. The flow of information, commands, responses, and visual presentations should be in a sensible order that is easy to recollect and place in context.

Natural. Operations should mimic the user's behavioral patterns. Dialogs should mimic the office worker's thought processes and vocabulary. Common human behavioral and perceptual traits should be taken into account in the design.

Predictable. System actions should be expected, within the context of other actions that are performed. All expectations should be fulfilled uniformly and completely.

Responsive. Every human request should be acknowledged, every system reaction clearly described. Feedback is the critical ingredient for shaping a person's performance.

Self-explanatory. Steps to complete a process should be obvious and, where not, supported and clarified by the system itself. Reading and digesting long explanations should never be necessary.

Forgiving. A system should be tolerant of the human capacity to make errors, at least up to the point where the integrity of the task or system are affected. The fear of making a mistake and not being able to recover from it is a primary contributor toward human fear of dealing with computers.

Efficient. Eye and hand movements should not be wasted. Attention should be directed to relevant controls and displays of information. Visual and manual transitions between various system components should proceed easily and freely.

Flexible. People should be able to structure or change a system to meet their particular needs. Inexperienced people may wish to confront and use only a small portion of a system's capabilities until they have become more familiar and comfortable with it. With experience, they may wish to use additional features.

Available. Like any tool, the computer must be available if it is to be effective. Any system unreliability, no matter how good normal system performance, will create dissatisfaction in the user.

4.2 THE REAL WORLD

Of course there is probably no system in existence that has all of these attributes. In addition, many of the features listed above can only be the result of good system level design, well beyond our ability to influence with simple screen design. But yet, with most systems the terminal screen is just about all of the system that anyone ever sees. As the only visible part of a computer system, it need to be beyond reproach. By giving the user a good first impression, a good screen design can actually help overcome deficiencies is other parts of the system.

As with everything else in the world, great screen design is both hard to create and rare. However, there are so many truly awful screen designs in the world, that to create a *good* screen design is not that hard. Simply by observing a few guidelines and thinking carefully about how the screen will be viewed and used, we can make screens that make the computer system easier to use, more productive, easier to sell, and a pleasure to use.

Better still, the principles of good screen design can be applied to existing applications, improving existing systems with only a small amount of effort.

4.3 THE WELL DESIGNED SCREEN

Two types of considerations drive the design of a good screen display. *Human considerations* are the needs and requirements of people and are oriented toward clarity, meaningfulness, and ease of use. *Hardware and software considerations* reflect the physical constraints of the terminal being used and the software which is driving it.

A well designed screen will be consistent within itself, within related screen formats, and with other screens in the same system.

4.4 THE HUMAN CONSIDERATIONS

Computers are only of use when they can be used by humans. We have not yet reached the point where computers are truly used by other computers, without humans in the chain somewhere. The fact that humans are involved is both a blessing and a curse. A blessing because there are certain behavior patterns we know and can take advantage of, and because, if all else fails, that person is capable of rational thought to determine what action is needed next. A curse because people are picky, hard to please, difficult to deal with, and demanding. We need to understand both sides of people in order to develop effective screens.

One large insurance company's MIS department asked their users what they looked for in a good screen design and they listed the following:

- Orderly, clean, clutter-free appearance
- Obvious indication of what is being shown and what should be done with it
- Expected information where it should be
- Clear indication of what relates to what (headings, field captions, etc.)
- Plain, simple English
- A simple way of finding out what is in the system and how to get it out
- A clear indication of when an action could make a permanent change in the data or system operation

The desired direction is toward simplicity, clarity, and understandablity.

4.4.1 Human Characteristics Important to Screen Design

A human being is a complex organism with a variety of attributes, many of which are important to screen design. Of particular importance are human perception, memory, and learning. These areas, unfortunately, are not as clear cut as we would like. We must turn to the science of cognitive psychology and human factors research to gain insight into these human characteristics.

Perception is our awareness and understanding of the elements of the environment through physical sensation. This comprehension is achieved in light of our experience; we classify stimuli based on models stored in our memories. Perception tends to put things into wholes, to establish order and meaning. The human sensing mechanisms are constantly bombarded with a wide range of stimuli, only some of which are important. These important stimuli we can call signals, the other stimuli noise. It is our perception that allows us to separate the signals from the noise. If, however, the noise level is too great, our perception is overwhelmed and we cannot determine what is important as a signal and what is meaningless.

The human perceptual mechanism seeks order and understanding when confronted with uncertainty. Given a screen display, humans will immediately try to make order out of the information, grouping things based on physcial position, like appearance, symmetrical balance, and any other hint that there might be structure to the display. A cluttered or unclear display requires extra effort to make sense of, beyond that of our basic perceptual mechanisms. The user who must deal with such a display is forced to take extra time and

effort to learn and understand. A user given a display which does not require this extra effort will not have to expend this effort. We should therefore capitalize on our perceptual desire for order when organizing screen displays.

Additionally, a user new to the system in question is probably already nervous, and easily overwhelmed by the data being presented on the screen. Presenting that data in an orderly, simple fashion can reduce the user's anxiety about using the new system.

In looking at displays of information, eyeball fixation studies indicate that initially one's eyes usually move to the upper left center of the display and then quickly move in a clockwise direction. During and following this movement people are influenced by the balance and weight of the titles, graphics, and text of the display. Significant changes in any of these can influence the pattern and order of eye movement.

Memory is not one of the great human virtues. Short-term memory is highly susceptible to interference and its contents are always in danger of being replaced by new information. Its capacity is about seven items. Remembering a telephone number long enough to dial it taxes the memory of many people. Clearly this memory should not be relied upon for screen design.

The human ability to learn is important— it is what differentiates people from machines. A screen design developed with the intent of minimizing human learning time can accelerate human performance. Given enough time, of course, people can improve their performance in almost any task. Most people can be taught to walk a tightrope, but a designer should not call for a tightrope if a walkway is feasible.

Most learning is a combination of trial and error and insight. Learning can be enhanced if it:

- allows skills acquired in one situation to be used in another somewhat similar situation. Design consistency accomplishes this.
- provides complete and prompt feedback.
- is phased, that is, requires a person to know only that information needed at that point in the learning process.

4.4.2 Screen Format and Content

Clarity, meaningfulness, and ease of use of a screen display is achieved through the format and content of the display itself and the information contained in the display. The format and content of the display will be determined by where information is placed, how information is structured, and what information is included. The guidelines that follow provide general rules addressing these questions.

Provide an obvious starting point in the upper-left corner of the screen.

This is where reading starts in most Western cultures, and where the eye usually first lands when presented with a screen display.

Reserve specific areas of the screen for certain kinds of information, such as commands, error messages, and input fields, and maintain these areas consistently on all screens.

People tend to have good location memory, so that once they have learned where to look for a specific information item, they will tend to look there again.

Provide cohesive groupings of screen elements by using blank spaces, surrounding lines, different intensity levels, varying display techniques, and so on.

Cohesive groupings allow people to perceive a full screen as having identifiable pieces; smaller chunks that may be more easily handled. Blank or white space is one way to achieve this. Use of contrasting display techniques such as differing intensity levels or reverse video are some other ways.

Provide symmetrical balance

Techniques for achieving balance include

- centering titles and illustrations in the vertical axis
- placing like elements on both sides of the axis
- placing lighter elements further from the vertical axis and larger elements closer to it

Provide only information that is essential to making a decision or performing an action. Do not overwhelm a person with information.

The more information, the more competition for the user's attention. Visual search times will be longer and meaningful patterns more difficult to create.

Provide all data related to one task on a single screen. One should not have to remember data from one screen to the next.

Short term memory is limited in capacity and unreliable. All necessary information should be available on the screen.

Present information in directly usable form. Do not require reference to documentation, translations, transpositions, interpolations, etc.

The computer is far better at performing such conversions than any person is. Let the computer do the work.

Guide a person through the screen with implicit or explicit lines formed by display elements.

By carefully arranging the screen display, the user's eye can be led to the next field to be used.

Visual appearance and usage should be consistent.

If the appearance is consistent, the user does not need to learn new information for each screen, reducing training time.

For text, use lower case with the initial sentence letter in upper case.

Studies have shown that the use of mixed upper and lower case greatly aids in comprehension and recognition of words, due to the larger number of symbol shapes available.

Use standard alphabetic characters to form words or captions

Specialized computer characters, especially when rendered in dot matrix form on a terminal screen, are not easily recognizable by many people.

Abbreviations, mnemonics, and acronyms should not include punctuation (e.g., "CPU" not "C.P.U.").

Better still, avoid the use of such shortcuts and jargon.

Make sure that the words in displayed messages are short, meaningful, common, and spelled out.

Also make sure that the words are not ambiguous or confusing. Jargon should be avoided whenever possible.

Make sure that the messages displayed exhibit all the characteristics of good writing.

Just because it on a computer does not mean that it can be illiterate or even inelegant. The sentences should be

- Brief, simple, and clear
- Complete
- Directly and immediately usable
- Affirmative
- In an active voice
- Nonauthoritarian
- Nonthreatening
- Nonpatronizing
- Nonpunishing
- Structured so that the main topic is near the beginning
- Cautious in the use of humor

4.5 HARDWARE CONSIDERATIONS

Once a screen has been designed, it must be implemented on the terminal hardware to be used with the system. Unlike those who deal with the printed word, we are somewhat limited in the ways in which we can display information. The printed page can display a variety of text styles in different sizes and orientations, as well as special characters, lines, charts, photographs, and just about anything else you can think of. Typically we are limited in display size (most often to something approximating 24 lines by 80 columns), in choice of characters (typically just the alphabet, numbers, and limited punctuation), and in character size and style (usually a single size and style).

Terminals do offer several ways of distinguishing text, such as reduced intensity, reverse video, blinking and underlining. However, each of these has its drawbacks. Use of these methods in designing a screen requires that we understand not only how they work (the subject of later chapters in this book), but also how they are perceived by the people viewing the screen display.

4.5.1 Blinking Text

Blinking text has excellent attention-getting ability. This is both a benefit and a liability. In situations where it is important to capture the operator's attention, blinking excels. However, blinking reduces the legibility of the message— you get to see it for less time since it keeps turning itself off— and can be quite distracting if it is left on the screen once attention has been gained. For this reason it should be limited to situations which require an immediate response, and not just used for informational purposes.

One interesting method of dealing with these problems is to turn off the blinking once the appropriate action has been taken, even if the text is otherwise left on the screen.

4.5.2 Different Brightness

Changing the intensity of the characters has good attention getting capability and is the least visually disturbing of the various display enhancements. It is best for attention getting when the text to be highlighted becomes brighter than the other text, although reversing this convention is also used.

4.5.3 Reverse Video

The use of reverse video— black characters on a light background— has good attention getting capability, but can reduce the legibility of the text so displayed. Particularly on terminals where the screen brightness control has been adjusted too bright, reverse video text may be hard to read.

It should not be used to excess on a screen or the display may suffer from the "crossword puzzle effect" where the areas of reverse video start to form patterns which distract from the information being presented.

4.5.4 Underlining

Underlining is not a very good attention getter on terminal screens since the underline is usually placed so close to the characters that it tends to blend into the character, reducing legibility. Underlining can serve a useful function, however, by seperating areas of text.

4.5.5 132-Column Display

Many modern terminals have the option of displaying the screen in a 132-column wide mode. While this gives the ability to display increased amounts of information, this is not usually an advantage. The reduced legibility of the smaller characters, combined with the danger of "information overload" make 132-column mode best suited for special applications, such as spreadsheets, or previewing printed reports.

4.5.6 Status Lines

Many terminals have additional screen display lines above and/or below the main display area. Typically known as *status lines* these areas remain even when the main screen area is cleared. For this reason, they are a convenient place to store information specific to the overall application, rather than to the specific screen being displayed. Relying on the presence of status lines, however, can cause problems if the application is run on a terminal without status lines.

4.5.7 Split Screen

Some terminals have the capability of dividing the display area into two or more independent screen areas which may be individually cleared and scrolled. This makes it convenient to implement screens formatted as multiple independent areas.

While split screen is a convenient implementation method for a specific type of screen design, don't let the availability of split screens drive the screen design process. Only use split screens if the information that must be displayed warrants it.

4.5.8 Color

Some terminals have the ability to display different colors. While this can be an excellent tool for effective screen design, it also has its pitfalls. Chapter 5 is devoted to the subject of color usage and how it can affect people's perceptions.

4.5.9 Graphics

The addition of graphics to a terminal opens up a wide variety of options. Even the simple line drawing character sets available on most terminals allow a variety of capabilities, including segmenting screen regions via boxes, lines, etc. Care must be taken, however, to avoid making things worse by having the graphic elements be what is focused on. Make sure that the graphic elements you add are in support of the information on the screen and do not overwhelm it.

A full graphics capability opens up possibilities which are too varied and numerous to be discussed here. As full graphics terminals become more widespread, additional attention to screen design for them will be needed.

4.6 IMPLEMENTING GOOD SCREEN DESIGNS

It is just about impossible to follow every one of these rules in a particular design. Likewise, the field of human factors study is rapidly evolving as more people begin to understand the importance of the field and more research is performed.

So rather than taking the information we have outlined as graven truth, it should instead be viewed as a series of guidelines that can help you to design your own set of rules for screen formatting. For example, during the development of the AlphaWRITE word processing system, an understanding of the principles we have outlined led to the following guidelines for screen displays being set up:

- All screens have a title, summarizing the purpose of the particular screen, displayed on the top line, centered, in reverse video.

- All information from the user is accepted from the central portion of the screen.

- Prompts are not displayed until the information is actually requested.

- Information supplied to a question is left on the screen to remind the user of previous responses, which may affect the choice of later responses.

- Any time that a response is requested, the range of valid responses is given to the user on the bottom portion of the screen in a standard, consistent format.

- Error messages are only generated when the user's input is ambiguous. Even if a typo is present, if the input cannot be mistaken for a different command, the command is executed with no error message.

- Error messages are displayed in blinking text at the bottom of the screen.

- All error messages display both the cause of the error and suggestions on how to rectify the problem.

- When a "fill in the blanks" type of screen display is presented, the user is free to move back-and-forth, from field to field, until satisfied with all answers.

- The use of the upper status line is limited to non-essential information in case it is not available or its use is preempted by other software.

- Command prompts, where the expected response is a keyword selected from a displayed list, display all possible responses. The keyword is displayed as "bright" text, while the explanatory text is displayed in reduced intensity. This focuses attention on the keywords, while providing additional information in a non-distracting manner.

- While primarily a menu driven product, the experienced user is accommodated by a "menu bypass" option, eliminating distracting information which a particular user may already have committed to memory. All help information remains available at each screen in case a particular detail is forgotten.

- Status information displayed during text editing is placed at the bottom of the screen in reduced intensity to reduce any distraction it might cause.

While these guidelines were specifically developed for AlphaWRITE, they have been successful enough to be used for later software development efforts such as AlphaCALC and AlphaMAIL.

The exact details of such guidelines are not nearly as important as the fact that you develop such guidelines to help you in screen design.

CHAPTER 5

USING COLOR EFFECTIVELY

Color can add another important dimension to screen displays. This chapter describes how we see colors and how that affects the way we should design our screens. While this chapter presents a wide variety of information, much of it seemingly unrelated to "how to change colors on the screen," it is intended to provide you with the information necessary to appreciate the complexities of color usage and to understand that color screen design is more than simply "picking a pretty color."

Information on how to make your program select and control the screen colors can be found in Chapter 10.

5.1 THE IMPACT OF COLOR

After many years of being an expensive option available on only very high-end computer systems, color terminals are declining in price to the point where they are readily affordable for a wide variety of applications.

Just as color film brought a new sense of life to photography, and greatly increased the number of applications to which it could be put, color can bring many important benefits to the user of a computer terminal. By coding data and prompts with color, the screen display can be made easier to comprehend and interpret. An important figure that might otherwise be lost in a screenful of numbers can be made to stand out and attract attention.

However, just as it can help the comprehension of a display, so can it confuse. Improper use of colors in combination can lead to eyestrain and reduced operator efficiency. Inappropriate choices of colors can cause an operator to jump to the wrong conclusion about a display, making the use of color misleading instead of helpful.

Additionally, the addition of color to our already confusing world of incompatible terminals creates a whole host of new problems relating to the number of available colors, how to describe them, and how to specify them within a program.

Color can be an important addition to software, helping in both its usability and saleability, but only if the proper attention is paid to the physical, physiological and psychological aspects of color. Unfortunately, no one set of guidelines can cover all applications.

Up to now, color has been used almost exclusively in a qualitative rather than a quatitative fashion, that is, showing that one item is "different from" another, rather than showing relationships of degree. A typical example might be in multi-layer printed circuit board design where each layer is assigned a different color. The colors serve to separate the layers visually on the screen, but say nothing about the relationships between the layers. By assigning the colors in spectral order (using the ever popular ROY G. BIV (red, orange, yellow, green, blue, indigo, violet) ordering), it is not clear that such an ordering would lead to an intuitive understanding of

the structure of the layers. As the application increases in complexity, the demands for the proper use of color also increase.

Effective color usage depends upon matching the physiological, perceptual, and cognitive aspects of the human visual system. This chapter will describe some well documented aspects of the human visual system and provide some basic principles that should allow for improved color displays. While this may seem like overkill when you simply wish to change the color of text on the screen, the principles laid out can help you devise more pleasing and more functional screen displays.

It is helpful to understand the ways in which colors can be combined to generate specific hues. This chapter will describe how both subtractive and additive color combinations work.

Finally, we will discuss how colors can be described so that you can precisely specify the color you wish to use and the computer can generate the appropriate response.

5.2 THE PHYSIOLOGY OF COLOR

In understanding how we see color, it is important to realize that color is not a physical entity, but is instead a sensation, like taste or smell, that is tied to the properties of our nervous system.

The light we see as colors is electromagnetic radiation with wavelengths ranging from approximately 400 nanometers to 700 nanometers, covering the violet to red spectrum. The color sensation results from the interaction of this light with a color sensitive nervous system. While the frequency of the radiation being viewed by two people may be the same, we must remember that those individuals can have vastly different color-discrimination capabilities because of differences in the eye's lens, its retina, and other parts of the visual system.

Figure 5-1 shows the basic arrangement of the components of the human eye. The lens focuses the image on the retina, which in turn converts the light to electrochemical impulses carried by the optic-nerve bundle to the brain. Each of these elements in the perceptual chain contributes to the sensation we call color.

Sorry, but this image could not be converted to Adobe Acrobat Format. Please refer to your printed manual for this picture.

Figure 5-1 *- Gross Structure of the Human Eye*

5.2.1 The Lens

Camera lens designers go to great lengths to color correct the lenses they design so that all colors are evenly focused on the film, yielding a sharp color photo. Unlike the lens in your camera, the lens of the human eye is not color corrected. Different colors of light are focused at different points within your eye, causing *chromostereopsis*, an effect that causes two pure colors at the same distance from the eye to appear to be at two different distances. For most people, reds appear closer and blues more distant. In fact, short wavelengths— pure blue— always focus in front of the retina and thus appear defocused. This is most noticeable at night when deep-blue signs seem fuzzy and out of focus while other colors appear sharp.

Lens transmissivity also has an effect on how we perceive color. While it may appear to be clear, the lens absorbs almost twice as much energy in the blue region as in the yellow or red region. Also, a pigment in the retina's center transmits yellow while absorbing blue. The net result is a relative insensitivity to shorter wavelengths (cyan to deep blue) and enhanced sensitivity to longer wavelengths (yellows and oranges).

As we grow older, lens yellowing increases, making us increasingly insensitive to blues. Similarly, aging reduces the transmittance of the eye's fluids, which makes colors appear less vivid and bright. Actually, age aside, there is normally a great deal of variation, with some people's eyes being very transparent and other's naturally yellowed. This variation alone contributes to differences in color sensitivities among individuals.

5.2.2 The Retina

The human retina consists of a dense collection of light-sensitive rods and cones. Rods are primarily responsible for night vision, while cones provide the initial element in color sensation.

Photopigments in the cones translate wavelength to color sensation. The range of sensation is determined by three photopigments— blue (445 nanometers or nm), green (535 nm) and red (575 nm). "Red" is really a misnomer, because maximum sensitivity at 575 nm actually invokes the sensation of yellow, not red.

Both photopigment and cone distribution vary over the retinal surface. Red pigment is found in 64% of the cones, green in 32%, and blue in just 2%. Additionally, the center of the retina, which provides detailed vision, is densely packed with cones but has no rods. Moving outward, the number of rods increases to eventually outnumber the cones. As a result, shapes appear unclear and colorless at the periphery of vision.

Because of the cone and photopigment distributions, we can detect yellows and blues farther into our peripheral vision than reds and greens. Also, the center of the retina, while capable of high acuity, has almost no cones with the blue photopigment. The result is a "blue blindness" that causes small blue objects to disappear when they are fixated upon in the center of our field of vision.

For the eye to detect any shape of a specific color, an edge must be created by focusing the image onto the mosaic of rods and cones. An edge is a basic element in perceiving form. It can be created by adjacent areas differing in brightness, color, or both. Edges guide the eye's accomodation mechanism, which brings images into focus on the retina. Recent research has shown, however, that edges formed by color difference alone with no brightness difference, such as a red circle centered on a large green square of equal brightness, are poor guides for accurate focusing. Such contours remain fuzzy. For sharply focused images, it is necessary to combine both brightness and color differences.

Also, for photopigments to respond, a minimum level of light is required. This accounts for the "fading" of colors as the sun sets. As the light level decreases, the cones eventually stop contributing to vision, shifting the burden entirely to the photopigmentless rods. Additionally, the response level of the photopigments depends on wavelength, with greatest sensitivity at the center of the spectrum with decreasing sensitivity at the spectral extremes. This means a blue or red must be of much greater intensity than a minimum-level green or yellow in order to be perceived.

5.2.3 After the Retina

The optic-nerve bundle leads from the photoreceptors at the back of the retina. Along the optic-nerve path, at the lateral geniculate body, the photoreceptor outputs recombine. Figure 5-2 diagrams how this combination takes place.

Notice in Figure 5-2 that the original retinal channels— red, green, and blue— form three new "opponent channels" which are transmitted to the brain. One channel signals the red-to-green ratio, another the yellow-to-blue, and the third indicates brightness. Again, we find a bias against blue, since the perception of brightness, and hence of edges and shapes, is signaled by the red and green photpigments. The exclusion of blue in brightness perception means that colors differing only in the amount of blue they contain will not produce sharp edges.

Neural organization into opponent channels has several other effects, too. The retinal color zones, which link opponent red with green and opponent yellow with blue, provide an example. Opponent-color linking prevents us from visually experiencing combinations of opposing colors. Even though we can create light of the appropriate spectral mix, we cannot experience reddish-green or yellowish-blue. When we create a spectral mix of red and green, the red-to-green ratio cancels out to zero, while the yellow-to-blue ratio, with its built-in bias away from blue, causes us to experience the color yellow.

5.2.4 Colorblindness

The term "colorblind" is often used to describe the color deficiencies affecting some individuals. A concern about colorblindness is often used as an objection to the wide use of color since 6-8% of the male population has some kind of color perception difficulty. Colorblindness is usually an inheirited defect which affects less than 1% of the female population.

Not all causes of color-deficient vision are known; however, some are related to the cones and their photopigments. A rare form occurs when the blue photopigment is missing. The most well known defect, however, is red-green deficiency, which occurs when either the red or green photopigment is missing. Lack of either photopigment causes the same color discrimination problem; however, for people lacking red photopigment, long-wave stimuli appear much darker.

More common than a missing photopigment is a photopigment where the spectral response curve differs significantly from normal. The so-called red-green color blind man may simply have very low sensitivity to differences in the red-green part of the spectrum. This will probably be most apparent at low light levels and with values of colors which are in reality only barely discernible as different by the rest of the population. In fact, there are enough color perception differences among the population in general to explain the common situation of two people differing on whether a given color is blue or green.

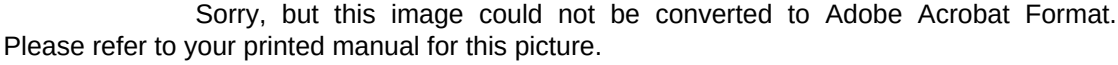

Figure 5-2 *- Processing of Color Input into Opponent Channels*

While serious forms of color deficiency exist, less than one-tenth of 1% of the male population actually has problems discerning differences in the types of colors used on terminal displays. While this can be a very important consideration in life-critical applications such as nuclear reactor control systems, aerospace navigation systems, or medical monitoring equipment, most of the environments we will be developing software for need not consider color blindness as a major constraint.

5.3 PRINCIPLES OF PERCEPTION

Perception refers to the process of sensory experience. Although perception is most certainly a product of our nervous system, adequate information about the "higher order" function does not exist to describe perception in physiological terms. As a result, psychological methods must be relied on, the most valuable discipline being psychophysics. Psychophysics is a discipline that seeks to describe objectively how we experience the physical world around us.

5.3.1 Perception is Nonlinear

Psychophysical research has shown that practically all perceptual experiences are nonlinearly related to the physical event. For example, the relationship between perceived intensity and physical intensity is nearly logarithmic, with perceived intensity increasing as the logarithm of stimulus intensity. We have all experienced this relationship when switching brightness of a three-way lamp— the brightness increase from 50 to 100 watts appears greater than that when switching from 100 to 150 watts.

5.3.2 Perception of Achromatic Color

White or achromatic light contains all the wavelengths to which the human eye responds. When such light strikes an object and all wavelengths are reflected equally, the color of the object is achromatic; that is, the object appears white, black, or some intermediate level of gray.

The lightness of the object depends on the amount of light reflected. An object reflecting 80% or more appears very light— we would call it white. Reflection of 3% or less results in objects appearing very dark— black. Various levels of gray appear in between, with lightness appearing to increase toward white as the logarithm of reflectance.

Consider, for example, black, white, and gray automobiles. Each car reflects different amounts of light and, therefore, takes on a specific achromatic color. If the total amount of light falling on the cars is increased, the lightness stays the same but the brightness increases. The white car stays white but becomes much brighter. Thus, lightness is a property of the object itself, but brightness depends on the amount of light illuminating the object.

A good illustration of all of this is black print on white paper. Changing illumination has little perceptual effect on the relative lightness of the paper or print, since the ratio of reflectance, or contrast, remains unchanged. Even in dim light, the white paper stays white and the black ink stays black.

5.3.3 Perception of Chromatic Color

Objects that reflect or emit unequal distributions of wavelengths are said to be chromatic, i.e., to have a color. The color we sense derives from the physical attributes of the dominant wavelengths, the intensity of the wavelengths, and the number and proportion of reflected wavelengths. Color identification also depends upon a multitude of learned variables, such as previous experiences with the object and association of specific sensations with color names. The sensation is also affected by the context in which the color occurs and the characteristics of the surrounding area or the colors of other objects. Because color perception is subjective, many of its aspects can be described only in the psychological dimensions of lightness, saturation, hue, and brightness, rather than in any absolute physical terms.

Hue is the sensation reported by observers exposed to wavelengths between approximately 380 and 700 nm, the range of light that the human eye is sensitive to. For the range between 450 and 480 nm, the predominant sensation reported is blue. Green is reported across a broader range of 500 to 550 nm, and yellow across a narrow band around 570 to 580 nm. Above 610 nm, most persons report the sensation red. The best or purest color— defined as those containing no trace of a second color— would indicate pure blue at 470 nm, pure green at 505, and pure yellow at 575 nm.

Hue, then, is the basic component of color. It is the primary determinant for a specific color sensation. Although hue is closely related to certain wavelengths, remember that hue is a psychological variable and wavelength a physical one. Although people with normal color vision will name a sector of the visual spectrum as red, they will disagree about the reddest red or where red becomes orange. Such disagreement reflects varying experiences with color as well as differences in each person's visual system.

Saturation is most closely related to the number of wavelengths contributing to a color sensation. As the band of wavelengths narrows, the resulting sensation becomes more saturated— the wider the band, the less saturated the color.

Conceptually, a scale of saturation can be envisioned as extending from a pure hue, such as red, through less distinct varients of the hue, such as shades of pink, to a neutral gray in which no trace of the original hue is noticed.

By taking a neutral color and determining the amount of a pure hue we must add to be able to perceive the color, we can see that different hues saturate at different points. For example, a yellow at 570 nm requires that largest amount of pure hue to be perceived. Such a yellow appears initially to be less saturated than any other pure hue and desaturates quickly as the distribution is broadened or as neutral colors are mixed with it.

Lightness, as mentioned previously, refers to those achromatic colors ranging from white through gray to black. By definition, achromatic colors are completely desaturated, since no trace of hue is present.

Just as with achromatic colors, the lightness of a mixed color also depends on the reflectance of the surface under consideration— the higher the reflectance, the lighter the color. As might be anticipated, monochromatic colors do not all appear equal in lightness. Some hues appear lighter than others even though their reflectances are the same. If, for example, observers are shown a series of monochromatic lights of equal brightness and are asked to rate them for lightness, we will see variations across the range of wavelengths. A monochromatic color at 570 nm (yellow) appears much lighter than all other wavelengths, even though the actual brightness is the same.

Brightness is another aspect of color perception. Increasing the illumination of both achromatic and chromatic colors produces a qualitative change in appearance that ranges from dark to bright. However, separation of brightness from lightness is often difficult, since brighter colors invariably appear lighter as well. And when dealing with a terminal screen which generates its own light, separation becomes impossible.

5.3.4 Colors in Context

Colors are also subject to contextual effects, in which adjacent colors influence one another. For example, a color on a dark background appears lighter and brighter than the same color on a light background. If a field is neutral (gray) or dark and displayed on a colored background, the background induces color into the gray. Red, for example, induces green into a neutral gray.

The size of a colored area also influences how we see it. In general, small areas become desaturated and can show a shift in hue. This creates problems when text is color coded, especially with blues and yellows, because they are susceptible to small-area color loss— yellow text is often hard to distinguish from white text, even though larger areas of the same colors are distinctly different. Also, small areas of color can mix; red and green in smaller areas are eventually integrated by the visual system into yellow. This is exactly the principle on which color video screens rely; they depend on the merging of the many small dots of color into larger areas of single colors. In other applications, however, the effect is not desirable.

5.3.5 Individual Characteristics

So far, perception has been described in general terms as it applies to the human visual system. Yet all of us have our own perceptual ideosyncrasies that affect how we use color. For example, some people prefer highly saturated colors and other prefer muted ones.

It is important, too, to remember that color perception changes over time. We adapt to color with prolonged viewing. This results in an apparent softening of colors. As a result, there is a tendency to use highly saturated colors to offset adaptation. The unadapted viewer, however, sees the colors as highly saturated. Additionally, some research indicates that pure colors are visually fatiguing.

Although we are still far from developing an aesthetics of color displays, some information has been compiled on color combinations that go well together and those that do not. Figures 5-3 and 5-4 present data from a study in which people were asked to pick the best and worst appearing colors on different backgrounds. Choices were made for both thin lines and for larger filled panels. The figures list those combinations preferred or rejected by at least 25% of the subjects participating in the study. While this was a small study, and may not reflect the choices that would have been made by the people who will use your software, it does provide a starting point for choosing color combinations.

Background	Thin Lines and Text	Thick Lines and Panels
White	Blue, Black, Red	Black, Blue Red
Black	White, Yellow	Yellow, White, Green
Red	Yellow, White, Black	Black, Yellow, White, Cyan
Green	Black, Blue, Red	Black, Red, Blue
Blue	White, Yellow, Cyan	Yellow, Magenta, Black, Cyan, White
Cyan	Blue, Black, Red	Red, Blue, Black, Magenta
Magenta	Black, White, Blue	Blue, Black, Yellow
Yellow	Red, Blue, Black	Red, Blue, Black

Figure 5-3 - *Best Color Combinations With Best First* (from G.M. Murch, "Physiological Principles for the Effective Use of Color")

Background	Thin Lines and Text	Thick Lines and Panels
White	Yellow, Cyan	Yellow, Cyan
Black	Blue, Red, Magenta	Blue, Magenta
Red	Magenta, Blue, Green, Cyan	Magenta, Blue, Green, Cyan
Green	Cyan, Magenta, Yellow	Cyan, Magenta, Yellow
Blue	Green, Red, Black	Green, Red, Black
Cyan	Green, Yellow, White	Yellow, Green, White
Magenta	Green, Red, Cyan	Cyan, Green, Red
Yellow	White, Cyan	White, Cyan, Green

Figure 5-4 - *Worst Color Combinations With Worst First* (from G.M. Murch, "Physiological Principles for the Effective Use of Color")

5.4 COGNITIVE PRINCIPLES

The least understood area of effective color usage is how to capitalize on our modes of thinking about, and associating with, color. This area of study falls into the realm of cognitive ergonomics.

Despite the infancy of this area of human-factors study, some initial observations prove useful in effective color usage. An example involves the functional use of color stereotypes: red for warning, green for go, and yellow for attention. Since we all have experience with these meanings, maintaining the relationship maps nicely into our expectations.

In color-coding graphed measurement data, variation of hue can quickly communicate important information. Portions of the data within a certain tolerance, or range limit, can be coded green; portions approaching a limit can be yellow; and excesses can be coded red. This procedure fits into the normal cognitive expectations.

For multiple graphs, where color is simply used to differentiate between data sets, contrast is a big consideration. As a result, it is tempting to use red for one line and green for another. While this makes the data readily distinguishable, which is the goal, it can also bias a viewer toward making some quality judgements about the data— the red data are bad or dangerous, the green are okay. Such biasing of the viewer might not be what you intended. Similarly, the perceived magnitude of different colors varies. A red square will be perceived as being larger than a green one of identical size. In a situation such as an area graph, this can be extremely misleading.

Combinations of colors may be perceived as defining groupings of data that you did not intend. For example, consider a graph which consists of 4 lines, one yellow, one red, one orange, and one green. An observer may tend to group the "warm" colors (yellow, red, and orange) together, and assume that the lone "cool" color (green) represents a different type of data.

Note that many of these cognitive principles can be used to your advantage. For example, using the tendency to group colors into "warm" and "cool" classifications can serve a useful grouping function in an appropriately designed graph. However, it is important that you be aware of the principles to avoid accidentally running afoul of them.

5.5 GUIDELINES FOR EFFECTIVE COLOR USE

As we have noted, color can be an extremely useful tool in helping to differentiate data. The human eye can differentiate between approximately 25 colors with no reference, and over 20,000 colors when they are placed next to each other. This provides for a very rich method of encoding data.

However, the use of color can also be very tiring to the operator by causing eyestrain. Such eyestrain can result from the physical aspects of the terminal, such as the highly reflective screen often found on color terminals, or it may be caused by a choice of colors which causes frequent re-focusing of the eye. In addition, the use of colors which violates our unconcious assumptions about the meanings of color can lead to increased error rates and even mental stress. Only through the careful application of color can we be sure that we gain the maximum advantage of its use, without incurring its penalties.

On the basis of the preceding discussions, some general guidelines for color usage can be stated. They are grouped according to the area of their derivation— physiological, perceptual, or cognitive.

5.5.1 Physiological Guidelines

Avoid the simultaneous display of highly saturated, spectrally extreme colors.

Reds, oranges, yellows, and greens can be viewed together without refocusing, but cyan and blues cannot be easily viewed with red. To avoid frequent refocusing and subsequent visual fatigue, extreme color pairs such as red and blue or yellow and purple should be avoided. However, desaturating spectrally extreme colors will reduce the need for refocusing.

Avoid pure blue for text, thin lines, and small shapes.

Our visual system is just not set up for detailed, sharp, short-wavelength stimuli. However, blue does make a good background color and is perceived clearly out of the periphery of our visual field.

Avoid adjacent colors differing only in the amount of blue.

Edges that differ only in the amount of blue will appear indistinct.

Older viewers need higher brightness levels.

Due to the effects of aging, higher brightness levels become necessary to distinguish colors.

Colors change appearance as ambient light level changes.

Displays change color under different kinds of ambient light—fluorescent, incandescent, or daylight. Appearance also changes as the light level is increased or decreased. On the one hand, a change occurs because of increased or decreased contrast, and on the other hand, because of the shift in the sensitivity of the eye.

The magnitude of a detectable change in color varies across the spectrum.

Small changes in extreme reds and purples are more difficult to detect than small changes in other colors such as yellow and blue-green. Also, our visual system does not readily perceive changes in green.

Difficulty in focusing results from edges created by color alone.

Our visual system depends on a brightness difference at an edge to effect clear focusing. Multicolored images, then, should be differentiated on the basis of brightness as well as of color.

Avoid red and green in the periphery of large-scale displays.

Because of the insensitivity of the retinal periphery to red and green, these colors in saturated form should be avoided, especially for small symbols and shapes. Yellow and blue are good peripheral colors. With the common terminal screen of 14 inches diagonal or so, peripheral vision is not usally relied upon to the extent where this would be a concern.

Opponent colors go well together

Red and green or yellow and blue are good combinations for simple displays. The opposite combinations— red with yellow or green with blue— produce poor images.

For color-deficient observers, avoid single color distinctions.

Colors which differ only in the amount of red or green added or subtracted from the two other primaries may prove difficult to distinguish for certain classes of color-deficient observers.

5.5.2 Perceptual Guidelines

Not all colors are equally discernible.

Perceptually, we need a large change in wavelength to perceive a color difference in some portions of the spectrum and a small one in other portions.

Luminance does not equal brightness.

Two equal-luminance but different hue colors will probably appear to have different brightnesses. The deviations are most extreme for colors towards the ends of the spectrum (red, magenta, blue).

Different hues have inherently different saturation levels.

Yellow in particular always appears to be less saturated than other hues.

Lightness and brightness are distinguishable on a printed copy, but not on a color display.

The nature of a color display does not allow lightness and brightness to be varied independently.

Not all colors are equally readable or legible.

Extreme care should be exercised with text color relative to background colors. Besides a loss of hue with reduced size, inadequate contrast frequently results when background and text colors are similar.

Hues change with intensity and background color.

When grouping elements on the basis of color, be sure that backgrounds or nearby colors do not change the hue in an element in the group. Limiting the number of colors and making sure they are widely separated in the spectrum will reduce confusion.

Avoid the need for color discrimination in small areas.

Hue information is lost for small areas. In general, two adjacent lines of a single-pixel width will merge to produce a mixture of the two. Also, the human visual system produces sharper images for achromatic colors. Thus, for fine detail, it is best to use black, white, and gray while reserving chromatic colors for larger panels or for attracting attention.

5.5.3 Cognitive Guidelines

Do not overuse color.

Perhaps the best rule is to use color sparingly. The benefits of color as an attention getter, information grouper, and value assigner are lost if too many colors are used. Cognitive scientists have shown that the human mind experiences great difficulty in maintaining more than five to seven elements simultaneously, so it is best to limit displays to about six clearly discriminable colors.

Group related elements by using a common background color.

Cognitive science has advanced the notion of set and preattentive processing. In this context, you can prepare or set the user for related events by using a common color code. A successive set of images can be shown to be related by using the same background color.

Similar colors connote similar meanings.

Elements related in some way can convey that message through the degree in similarity in hue. The color range from blue to green is experienced as more similar than the range from red to green. Along these same lines, saturation level can also be used to connote the strength of relationships.

Link the degree of color change to event magnitude.

As an alternative to bar charts or tic marks on amplitude scales, one can portray magnitude changes with progressive steps of changing color. A desaturated cyan can be increased in saturation as the graphed elements increase in value. Progressively switching from one hue to another can be used to indicate passing critical levels.

Order colors by their spectral position.

To increase the number of colors on a display requires imposing a meaningful order on the colors. The most obvious order is that provided by the spectrum with the mnemonic ROY G. BIV (red, orange, yellow, green, blue, indigo, violet).

Warm and cool colors should indicate action levels

Traditionally, the warm (long wavelength) colors are used to signify action or the requirement of a response. Cool colors, on the other hand, indicate status or background information. Most people also experience warm colors advancing toward them— thereby forcing attention—and cool colors receding or drawing away. This is apparently caused by the focus shift required by the different colors, and the everyday experience of atmospheric absorption where more distant object appear "bluer" than nearby objects.

5.6 COLOR MIXTURE

Having decided what colors we want on our displays and hardcopy, it becomes useful to understand how we can produce the different colors. The process by which inks and dyes are combined— whether for a magazine or computer output hardcopy— is called *subtractive color mixture*, while the generation of phosphor emmissions follows the process known as *additive color mixture*.

5.6.1 Subtractive Color Mixture

The perceived color of a surface, such as a sheet of paper, depends upon the capacity of the surface to reflect some wavelengths and absorb others. When a surface is dyed with a particular pigment, a new reflectance characteristic is created by the ability of that pigment to reflect some wavelengths and absorb others. A surface dyed yellow, for example, might reflect wavelengths above 570 to 580 nm while absorbing most of the longer and shorter wavelengths. Consider another surface dyed cyan (blue-green) such that wavelengths of 440 to 540 nm predominate. If we were to mix both pigments and deposit them on a surface, the resulting color would be green. The mixture of cyan and yellow produces green because the yellow pigment absorbs all of the short wavelengths (500 nm and below) and some of the middle band of wavelengths (500 to 550 nm). The cyan pigment absorbs all of the long wavelengths (560 nm and above) and some of the middle wavelengths (500 to 550 nm). Thus, the yellow absorbs those wavelengths evoking the sensation of blue while the cyan absorbs those wavelengths evoking yellow. Between these two extremes, a band of wavelengths is "left over" which evokes the sensation of green. This type of color mixture is called *subtractive color mixture* as bands of wavelengths are subtracted or cancelled by the combination of light absorbing materials.

If we add a third pigment to the mixture of yellow and cyan— one that absorbs the middle band of wavelengths such as magenta or violet— the surface would appear black since all of the light falling on it would be absorbed. By this process of eliminating parts of the reflectance distribution through varying the amount of each pigment, intermediate hues can be created. In the example given above, the resulting green would not be very light as much of the illumination falling on the surface is absorbed. The mixture of two pigments produces a reflectance surface which absorbs more light than either pigment alone.

With a palette of only two pigments, each with fairly broad reflectance distributions, consisting of a red and a blue, we can create all the intermediate hues by varying the density of each pigment. The problem, however, is that the resulting hues will not be very light. Adding a third, fourth, or fifth pigment to the palette helps a great deal by increasing the overlap in the reflectance distributions of each so that lighter mixtures can be produced. In fact, the minimal number of colors is three, which are often referred to as the *primary colors of subtractive color mixture*. An artist, of course, uses many other colors in order to increase the purity and lightness of the available colors.

Color hard copy, then, is produced by the subtractive combination of inks or dyes. In most applications, the three primaries used are yellow, cyan (blue-green), and magenta. These three in all possible combinations provide a minimum palette of eight colors as shown in the table below.

Color	*Subtractive Combination*
Red	Yellow + Magenta
Green	Cyan + Yellow
Blue	Magenta + Cyan
Yellow	Yellow
Cyan	Cyan
Magenta	Magenta
White	—
Black	Yellow + Cyan + Magenta

Pure black is often included as a fourth color simply because the three primaries which produce the best chromatic color usually do not produce the best black.

To extend the hardcopy palette beyond eight colors requires that different levels of the colors appear in each mixture. Two density levels increase the palette to 20 colors. An alternative method of increasing the number of available colors is known as *half toning*. This technique places color on the paper as tiny dots that can either vary in frequency or size. Pure red, for example, would require the highest dot frequency available, while a desaturated pink would result from halving that frequency. In this technique, the resulting hue occurs as a result of the *additive mixture* of red and the white of the paper, whereby the dot frequency defines the density of the color component.

5.6.2∞Additive Color Mixture

Colors can be mixed in another fashion in which bands of wavelengths are added to one another. In fact, this method of additive color mixture forms the underlying principle by which the visual system "mixes" colors. Additive color mixing is also the means by which color is produced on a color terminal screen.

The surface of a typical color terminal screen is made up of thousands of tiny dots of phosphor. The phosphors on the screen are grouped into threes— called *triads*— with one phosphor emitting long wavelength light (red), one emitting middle wavelength light (green), and the third emitting short wavelength light (blue). To display a red character on the screen, all the red phospors forming the outline and interior of the character are made to emit light. A green or blue character would be produced in the same manner.

Intermediate hues to red, green and blue are produced by simultaneously making two or more of the three phosphors in a triad emit light. Because the phosphor dots are very small, when viewed from a distance the output of the three members of the triad fuse together, forming a mixture of the phosphor outputs. The result is a homogenous appearing field of color.

Causing both the red and green phosphors to emit light will result in a yellow hue. The exact color of the yellow will depend on the relative intensities of the red and green phosphors. Increasing the amount of red while decreasing the green would move the color toward orange. Conversely, increasing the intensity of the green would move the color towards yellow-green.

All three phosphors together produce a very broad distribution of wavelengths which evokes the sensation of white. Varying the intensity or gray level of the three phosphors produces different levels of lightness. Because one is able to mix most hues, as well as achromatic colors, by additively combining red, green and blue, these are called the *primary colors of additive color mixture*.

By selectively combining the red, green and blue phosphors, we can produce eight colors, just as we did with the subtractive primaries:

Color	***Additive Combination***
Red	Red
Green	Green
Blue	Blue
Yellow	Red + Green
Cyan	Green + Blue
Magenta	Red + Blue
White	Red + Blue + Green
Black	—

If we can control the relative intensities of the three phosphors we can create a much larger palette of colors. If we can select among four different intensities— including "off"— for each phosphor, we get a palette of 64 colors. Encoding such a palette would require two bits per phosphor, or a total of six bits. Additional control over each phosphor gets us larger and larger color palettes. The largest color palette commonly used with color display terminals allows 8 bits per phosphor, allowing for a selection of 256 different intensities for each of the three phosphors. This yields a palette of over 16 million different colors.

5.7 DESCRIBING COLORS

One of the problems we must face when using colors with a computer is the task of being able to describe and name colors. While most of us could agree on a color we could all describe as "blue," we are less likely to agree on what exactly is meant by "light blue," "sky blue," "baby blue" or any of several hundred other slight variations on the same theme. As we have seen, this results not only from personal opinion and experience, but also from actual differences in the way different people experience color.

When we are dealing among ourselves, this does not present a major problem as it is possible to add additional adjectives to our description to reach agreement, or if all else fails, to compare color chips to reach a consensus.

This is not possible, however, when it comes to a program trying to specify a color to a display. Even if it were possible for the program to show a color chip to the display and ask for a color just like it, it is not likely that the display would respond in any meaningful fashion.

Because the same problem has been faced by color television engineers, and artists and printers before them, a number of systems for describing colors in a mathematical fashion have been devised. Most of these, while fairly accurate in terms of color specification and repeatability, are difficult to use and do not correspond to our intuitive sense of color.

The system most familiar to computer users is probably the RGB or Red-Green-Blue system. Based on various mixtures of these three primary colors, it is the system on which color television is based. Because most of the technology behind color terminals comes from the color

television world, RGB is the system employed internally in color computer terminals. Despite that fact that it attempts to mimic a portion of the human visual system (the three color receptor photopigments), it is not an intuitive system. Given a color described by its RGB components, most people would not be able to tell what should be changed to make the color "a bit yellower," a description that a human can readily understand.

It is important to note here that no color system can exactly describe all possible colors. All color description systems are at best a compromise with some being superior to others. Likewise, any mathematical color description system can be converted into another via a mathematical transformation. This fact is depended on in color television where the picture tube deals with colors in terms of Red-Green-Blue (RGB) descriptions but where the color signal is broadcast in terms of chrominance-luminance-intensity (IYQ). The electronics in the television receiver simply convert one representation to the other. If you have a VCR connected to your television, it even deals with colors in a third representation!

All of this discussion of television is simply to show that we can choose any representation for color that we wish, and leave it up to the terminal driver and the terminal itself to convert it to the representation which the terminal actually uses to physically display the colors.

5.7.1 The Hue-Lightness-Saturation Model of Color

Without trying to examine all of the possible color models that were considered, Alpha Micro has adopted the Hue-Lightness-Saturation (HLS) model as its standard for specifying color. As with most other color models it uses three components (hue, lightness, and saturation) for describing a given color.

We have described in some detail the three components of the HLS system that allow us to specify colors earlier in this chapter. As noted earlier, the fourth component, brightness, is not distinguishable from the lightness component when used with display terminals. For this reason we will ignore the brightness component and only deal with hue, lightness, and saturation.

To be able to conveniently specify colors we need to have an easy way of defining the value for each of our three components. Also, it is helpful to have a physical model in mind by which we can picture how the color system works. The HLS color system is usually depicted as a pair of cones, attached end-to-end, as shown in Figure 5-5.

Hue (*H*) is represented as an angle about the vertical axis, ranging from 0˚ to 360˚. The angle *H* = 0˚ corresponds to blue. The remaining colors are specified around the perimeter of the cone with magenta at 60˚, red at 120˚, yellow at 180˚, green at 240˚, and cyan at 300˚. Complementary colors are 180˚ apart.

Lightness (*L*) is represented as the vertical axis of the double cone. It is given a value varying between 0 and 1. At *L* = 0, we have black. At *L* = 1, we have white. The gray scale is along the *L* axis, with the "pure hues" in the *L* = 0.5 plane.

The saturation parameter *S* specifies the relative purity of a color. This parameter also varies between 0 and 1. Pure hues are those for which *S* = 1 and *L* = 0.5. As *S* decreases, the hues are said to be less pure. At S = 0, we have the gray scale.

This model allows you to think in terms of making a selected hue darker or lighter. A hue is selected with hue angle *H*, and the desired shade, tint, or tone is selected by adjusting *L* and *S*. Colors are made lighter by increasing *L* and made darker by decreasing *L*. When *S* is decreased, the colors move toward gray.

5.7.2∞The Alpha Micro Eight-Bit Color Map

Using the HLS color description system, we can describe a virtually unlimited number of different colors. Of course, the display equipment we will be using to display the specified color is far more limited. Likewise, some programs simply need to be able to specify a color, without regard to the subtleties of color mapping systems.

For this reason, we have defined a simple eight-bit subset of the HLS color space which conveniently allows a set of up to 256 colors to be displayed. By carefully choosing how this set is mapped, it has been possible to set it up such that colors will automatically map to the closest color to the one specified when the display device is not capable of displaying the full 256 color range.

For example, a program which specifies a color assuming that the full 256 color range is available will still work on a terminal which displays only 8 colors. Because of the way in which the colors are mapped, the nearest available color will be chosen automatically.

The Alpha Micro color map defines these 256 colors in terms of their HLS coordinates, a descriptive color definition, and their RGB values. This allows programs complete freedom in converting from one color space to another.

Appendix C lists the Alpha Micro color map in detail, as well as giving the equivalent color description, HLS coordinates, and RGB values. Chapter 10 describes how to use the color selection commands to display colors on your display.

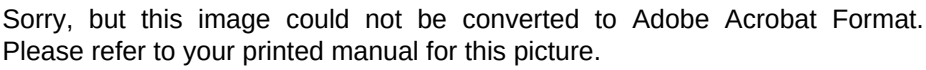

Figure 5-5 *- The HLS Color Description Model*

PART III

TERMINAL FEATURE OVERVIEW

CHAPTER 6

SCREEN DISPLAY ATTRIBUTES

6.1 DIM INTENSITY

The dim attribute causes text to be displayed in dim, or reduced intensity, mode. Unlike the other screen attributes, the dim attribute is always a mode attribute with hidden attributes. That is, it behaves like a mode terminal and does not require a screen position to switch from bright to dim or back again.

You start writing text in dim mode by issuing a TCRT –1,11 and end it by issuing a –1,12 command.

Dim mode text is an important part of protected fields, which are discussed in detail in Chapter 7.

Color terminals vary in the way that they handle the display of dim characters. Some actually choose a "dimmer" version of the currently selected foreground color while others substitute an entirely different color. Care should be taken in using the dim attribute on color displays unless the behavior of the terminal to be used is known.

6.2 UNDERLINE

Text written with the underline attribute is displayed with a line underneath it. Underline mode is selected by the TCRT –1,30 command, and disabled with the TCRT –1,31 command.

The screen position occupied by the begin attribute command is displayed as a normal space with no attributes. The underlining does not take affect until the next character. Likewise, the end-underline command is displayed as a normal space, with no underlining.

If your terminal supports hidden attributes (the T$NSP bit in the TRMCHR characteristics word is set), then you may also use the TCRT –1,106 and TCRT –1,107 commands to enable and disable underlining without occupying a screen position.

6.3 BLINK

Text written with the blink attribute is displayed as blinking or flashing text. The blink rate is determined by the terminal itself. Blink mode is selected by the TCRT –1,21 command, and disabled with the TCRT –1,22 command.

The screen position occupied by the begin attribute command is displayed as a normal space with no attributes. The blinking does not take affect until the next character. Likewise, the end-blinking command is displayed as a normal space, with no blinking. For the blinking attribute, this affect is usually visible only on color terminals.

If your terminal supports hidden attributes (the T$NSP bit in the TRMCHR characteristics word is set), then you may also use the TCRT –1,118 and TCRT –1,119 commands to enable and disable blinking without occupying a screen position.

6.4 REVERSE

Text written with the reverse video attribute is displayed with the foreground and background colors reversed. On monochrome terminals in which the normal display mode is light characters on a dark background, this will mean that reverse video is dark characters on a light background. On color terminals the foreground and background colors are simply exchanged.

Note that the reverse video attribute is not "additive." That is, setting reverse video mode once will reverse the foreground and background colors, but issuing it again will not reverse them again. The only way to return to "normal" video is to issue an "end reverse video" command.

Reverse video mode is selected by the TCRT –1,32 command, and disabled with the TCRT –1,33 command.

The screen position occupied by the begin attribute command is displayed as a normal space with no attributes. The reverse video does not take affect until the next character. Likewise, the end-reverse command is displayed as a normal space, with no reverse video.

If your terminal supports hidden attributes (the T$NSP bit in the TRMCHR characteristics word is set), then you may also use the TCRT –1,108 and TCRT –1,109 commands to enable and disable reverse video without occupying a screen position.

6.5 COMBINATIONS

In addition to the "pure" use of the attributes described above, the attributes can be combined to create many other combinations. The dim video attribute can be combined with any other attribute by issuing both the "begin dim" and "begin attribute" commands (where "attribute" is one of the other terminal attributes). The effects of these two commands are additive.

For combining the other attributes (reverse, blinking, and underscored), there are special terminal system commands defined to allow all of the possible permutations. Any of these permutations may be combined with the dim attribute.

The screen position occupied by the begin attribute command is displayed as a normal space with no attributes. The attribute does not take affect until the next character. Likewise, the end-attribute command is displayed as a normal space, with no attribute being displayed.

	Begin	End
Reverse and blink	–1,34	–1,35
Underscore and blink	–1,100	–1,101
Underscore and reverse	–1,102	–1,103
Underscore, reverse, and blink	–1,104	–1,104

And for attributes without embedded spaces:

	Begin	End
Reverse and blink	–1,110	–1,111
Underscore and blink	–1,112	–1,113
Underscore and reverse	–1,114	–1,115
Underscore, reverse, and blink	–1,116	–1,117

CHAPTER 7

PROTECTED FIELDS

Protected fields are simply areas of the screen display protected from erasure by the normal "clear screen" commands. Protected fields were originally designed for use with block mode terminals used on mainframe computers where there was no capability of providing a truly interactive environment because of the low data communications rates and the general "batch" or "transaction" orientation of such computers.

The majority of applications software consists of displaying a request for information, accepting a response, validating and storing the response, and then once again requesting the same item. In a block mode environment, then, it seems logical to move as much as possible of the responsibility of displaying the request, and accepting and validating the response, into the terminal, thus freeing up the host computer.

This was done by making the software create a "form" on the screen which would be filled in by the operator. Not until the entire form has been accepted and validated was anything sent to the host. Once everything was satisfactory, the entire form's worth of information would be sent at once, as in all block mode operation.

To further reduce the data communications requirements, it was attractive to only send the underlying form once, rather than for each request. The problem is once the form is filled out and transmitted, how do you erase the old information in preparation for the new request, without erasing the form itself? The answer, of course, is protected fields.

By designating the form as "protected," it became possible to clear only the operator's responses, without touching the form display.

This became a very popular method of dealing with block mode terminals. It was not long, however, before someone discovered that you could use the same technique with conversational mode terminals. Information would be transmitted on a character by character basis, but the form would not have to be repainted. This level of use of protected fields within a conversational, interactive environment is still valid. However, all of the fancy editing and validation capabilities that were built into some terminals simply aren't available in a conversational terminal.

Protected fields also are limited in their usefulness when dealing with any "form" that requires more than one screen to present. They are also being "squeezed out" by such features as color and graphics, where it is of limited use to be able to "clear everything but the red stuff" or "clear everything but the blue circles."

For this reason, care should be taken when designing new applications to make sure that by using protected fields you are not limiting your ability to use future terminals that will not contain protected field capabilities.

7.1 THE PROTECTED FIELD COMMANDS

Protected fields are implemented through the use of the TCRT –1,13 TCRT –1,14 and TCRT –1,11 TCRT –1,12 pairs of commands. Protected fields affect the operation of the commands TCRT –1,0, –1,9, and –1,10.

7.1.1 Enabling and Disabling Protected Fields

The terminal commands TCRT –1,13 and –1,14 are used to enable and disable protected fields, respectively. Once protected fields have been enabled, any text that has been written in the dim (protected) mode is protected from erasure via the Clear Screen (TCRT –1,0), Clear to End of Line (TCRT –1,9) and Clear to End of Screen (TCRT –1,10) commands.

Disabling protected fields returns the terminal to normal operation where the erase commands clear all text, dim or not, but does not change the appearance of the screen display.

7.1.2 Selecting Text as Protected

Text that is to be protected from erasure once protected fields are enabled must be written in dim (reduced intensity) mode. To begin writing text in dim mode, issue a TCRT –1,11 command. Once the desired text has been displayed, you may return to writing normal text by issuing a TCRT –1,12 command.

Note that some terminals require that you issue the Enable Protected Fields command (TCRT –1,13) prior to writing any protected text.

7.2 USING PROTECTED FIELDS

Because terminals vary widely in the precise interaction of the Dim commands and the Enable Protected Field commands, the following procedure should be used to eliminate any terminal dependent situations.

1. Plan on writing the entire protected area in one pass and the unprotected areas in a second pass.

2. Prepare the screen for the protected area by erasing the area first via the Clear Screen, Clear to End of Line, or Clear to End of Screen commands.

3. Turn on protected fields via the TCRT –1,13 command.

4. Turn on the Dim mode via the TCRT –1,11 command. When the protected fields mode is enabled, this command might be thought of as the "Write protected characters" command, rather than just the Dim command.

5. Position the cursor on the screen and display the characters that you wish to have protected from erasure.

6. Turn off Dim mode via a TCRT –1,12 command.

7. Position the cursor and accept input or output any characters you do not wish to have protected. This is the state that the program will be in for the majority of the time. As each query is responded to, the cursor position is updated and the next input is expected.

8. When a complete screenful of information has been accepted and a new "empty" screen is desired, position the cursor as necessary, and issue the appropriate erase command to clear out all characters except the protected text.

9. When the program is ready to erase the protected character (such as when it must display a different form), turn off protected fields via a TCRT –1,14 command, and erase the text via any one of the erase commands.

Because you cannot predict how any given terminal will deal with interaction between the cursor and protected fields, the applications software must perform all cursor movements between fields, not rely on the terminal to skip the cursor over any protected text.

For example, some terminals will not allow the cursor to be positioned within protected text and will force the cursor to be positioned on the next unprotected region of the screen, while others will allow the cursor to be placed anywhere, but will not allow any "write" of characters to that screen location. Still others regard the protected fields as only protected against erasure, and do no checking whatsoever when positioning the cursor or writing text to the screen.

For this reason, it is important the applications software not make any assumptions regarding the interaction of protected text with any command other than the three erase screen commands (TCRT –1,0, –1,9, and –1,10), lest the program become terminal dependent.

CHAPTER 8

PROGRAMMING A STATUS LINE

Most modern terminals now have one or more *status lines* which can be displayed on the screen in addition to the normal text. This chapter will describe what these status lines are and how to use them in a terminal independent fashion.

8.1 INTRODUCTION TO STATUS LINES

Status lines provide the programmer with one or more non-scrolling rows of text in addition to the normal text display. Because they are non-scrolling, they are often used for displaying fixed status information that should not be disturbed during normal program execution. Typical uses would be the name of the program being run, the file being edited, or the current operating mode of the program.

When status lines first became available they were often referred to as the "25th line" because of their position on the screen, following the standard display of 24 rows of text. This "25th line" often displayed the current operating mode of the *terminal*, displaying such items as whether the terminal was on- or off-line, what the state of the DTR control line was, etc. In addition, some terminals also had a small portion of the status line available for the display of arbitrary text by the host computer.

Because the terminal status information was mainly of use only to block mode users, terminals soon appeared with the entire "25th line" dedicated to programmable display, thus finally becoming useful to interactive users.

8.1.1 The Three Status Lines

Once the programmable "25th line" became popular, some terminal manufacturers started to include more than one status line on the screen, thus obsoleting the term "25th line."

Our standard terminal interface supports up to three status lines. The first, referred to as the "bottom status line" or "function key labeling line" appears at the bottom of the screen, in the same position as the old style "25th line." Most modern terminals have this capability.

In addition, a "shifted bottom status line" or "shifted function key labeling line" is supported. This line also appears at the bottom of the screen, but is visible only when the SHIFT key on the keyboard is pressed. This function is not available on all terminals, but is supported for terminals which have this capability.

A third status line, often referred to as the "top status line" or "host message field" is also supported. This status line appears at the top of the screen above the normal text display area. This function is not available on all terminals, but is supported for terminals which have this capability.

8.1.2 Status Line Addressability

The first programmable status lines available all required that the programmer redisplay the entire status line each time any information on the status line changed. This posed no problem for information such as the name of a program or the system date due to the infrequent updates required.

However, when used to display information such as the current time, or the current row and column position within an editor, requiring the entire status line to be constantly updated imposed a large amount of terminal communications overhead. In addition, because the status line typically was cleared each time it was selected, the status line appeared to "flash" each time it was updated.

For this reason, addressable status lines were developed. With an addressable status line it becomes possible to update any portion of the status line without disturbing the remaining areas. This is done by specifying a starting column number whenever the status line needs to be updated. By starting at a specific column, changing as few characters as possible, and then ending the update, a small portion of the status line can be modified quickly without annoying "screen flash."

All of the status lines supported by the standard terminal interface are assumed to be addressable. If a particular terminal does not support addressable status lines, such operation must be emulated by software contained within the terminal driver.

8.2 USING STATUS LINES

Before using any status line the very first thing that must be done is to check the TD$STS flag in the capabilities word returned by TRMCHR. If this bit is not set, no status line support is provided by the terminal in question and no further status line operations should be performed.

8.2.1 Displaying Text on a Status Line

Assuming that the TD$STS flag is set, the following procedure should be used when displaying text on a status line:

1. Select the bottom status line, shifted bottom status line, or top status line by issuing a TCRT –1,54, –1,55, or –1,63 command, respectively.

2. Select the column where you wish to begin updating the status line by sending one of the encoded column specifiers from Table B-1.

3. Display text and attributes as desired.

4. End the status line update by issuing a TCRT –1,129 command.

This procedure should be repeated for all updates you wish to make.

Some terminals will automatically exit the status line after all columns have been filled, without requiring a TCRT –1,129 call. Because not all terminals behave this way, this behavior should not be depended on. All status line calls should be terminated by a TCRT –1,129 call.

Note that only normal characters and attributes are allowed to be sent between the time a status line is selected (via the TCRT –1,54, –1,55, or –1,63 command) and it is terminated by the TCRT –1,129 call. Any other characters or TCRT commands will cause unpredictable results.

8.3 STATUS LINE ATTRIBUTES

Status lines use a different method of selecting display attributes than the rest of the screen. Rather than using TCRT calls to select the dim and bright attributes, special single character codes are used instead. Anytime that you write text to one of the status lines, you can select whether you want the text displayed in dim or bright mode.

To select dim mode text, precede the text with a hex 1F byte. To return to normal mode text, send a hex 1E byte. To avoid having to select dim or bright mode each time text is written to the status line, you can set the default mode for the status line by using the appropriate TCRT code to select the desired status line and then sending a 1E or 1F byte in place of the column code normally used. This will set the default display mode for the status line. A TCRT –1,129 should immediately follow this sequence. Any additional text sent to the status line will now be displayed in the selected mode, unless specifically overridden by sending a 1E or 1F code.

Other attributes (such as reverse video, underscore, etc.) are also implemented differently on status lines than they are in the other areas of the screen. Rather than using specific TCRT codes to turn these attributes on and off, special control characters must be sent to change the current attribute setting. These special control codes are shown in Table B-2.

Column one on all status lines contains the status line attribute. By reserving this position for the attribute, it becomes possible to select the entire status line as having a single attribute. For this reason, column one on all status lines must not be used to display normal text.

For example, column one of the top status line on an AM-62A terminal normally contains a "start underscore" attribute, resulting in the top status line being underlined. If text is written to column one the attribute will be lost, resulting in the top status line being displayed as normal text. If your program modifies the status line attributes in column one, be sure to restore the attribute before your program exits. By always restoring the attributes, all programs know the status line configuration upon start-up.

8.3.1 Choosing Where to Display Information

When choosing how to lay out your screen display it is important to keep in mind some important facts about status lines. While the terminal system command set supports both a top status line and a shifted and unshifted bottom status line, not all terminals have these capabilities. In addition, some system software may usurp one of these status lines for its own use.

Because most modern terminals have at least a single bottom status line, the unshifted bottom status line should be where your most important information is placed. That way, when the software is used on terminals which possess only the single bottom status line, the most important information will still be available.

Certain system software uses the top status line to display system status information and to notify you of system events (such as incoming mail). For this reason, only non-vital information should be placed on the top status line as the system may remove it from the screen.

8.4 STATUS LINE EMULATION

While most terminals have a bottom status line, not all of them have an addressable status line. Some terminal drivers, however, have been written to emulate an addressable status line. This emulation should be transparent to your software, although some performance degradation may be apparent to the user. If this type of performance is critical to your application, you should avoid the use of terminal drivers which emulate status lines.

CHAPTER 9

USING PROGRAMMED FUNCTION KEYS

While most of the work that the terminal service system must do to achieve terminal independence is on the output side, converting standard calls to the unique output required by a given terminal, the input side is also important. The use of function keys has become quite popular in software packages due to their inherent ease-of-use. As with everything else having to do with terminals, however, there is a multitude of different methods used for implementing them. To allow software to take advantage of function keys, but also to work on a variety of terminals, the AMOS terminal service system has been specially tailored to handle function keys in a terminal independent fashion.

9.1 GENERAL CONCEPTS

In order to allow programs to make use of function keys, a way had to be found to pass unique codes corresponding to the function keys to the applications program. On most terminals this unique code consists of a two- or three-character sequence, such as an ASCII escape character followed by another character. The problem with using such a method is that the two-character sequence is ambiguous: it can easily be confused with the same sequence generated by the operator pressing the two keys in sequence. The only difference between this and the function key is the timing relationship between characters. To require the applications program to monitor this timing relationship is not satisfactory.

To solve this problem, AMOS has been assigned the task of monitoring this timing relationship. It in turn translates the two or three character sequence into a truly unique, non-ambiguous code that is then passed to the applications software. This unique code is represented by a character with the high-order bit set to one.

This unique code will correspond to a given function key on a particular terminal's keyboard. It is important to note, however, that the code is unique to that particular type of terminal; that is, the F1 key on an AM-60 might generate an octal 301 code, while on some other terminal it might generate an octal 205. Thus the applications program must translate from this unique code to the code desired by the program based on the type of terminal being used.

9.2 FUNCTION KEY TRANSLATION TABLES

Because the function key codes returned by the terminal service system differ from terminal to terminal, a special translation table is required to convert the unique code output by a particular terminal to the character or characters that the program is looking for. This translation table is unique for each particular type of terminal that is to be supported.

For example, the AlphaWRITE and AlphaVUE programs use translation tables that are fetched from the disk each time the program is invoked. The choice of which translation table to use is based on the name of the terminal driver in use. To locate the function key translation table to use, the program simply looks up the name of the terminal driver in use, adds on a unique file extension, and loads that file into memory. For example, the translation tables for AlphaVUE all have the extension .VUX and are stored in DSK0:[7,0]. Thus if you are using an AM-60 terminal driver (AM60.TDV), AlphaVUE will use the translation table named AM60.VUX.

These translation tables consist of the unique code received from the function key, followed by the string of characters that the code is to be translated to. When a function key is pressed (indicated by the high-order bit being set), the application program simply scans the table for the function key value received and then uses the string of characters as its input.

By convention, all function key translation tables have a file extension ending in "X". Some common translation table extensions are:

AMX	General purpose table
CAX	AlphaCALC™ translation table
FIX	AlphaFIX translation table
MAX	AlphaMAIL™ translation table
MLX	MULTI translation table
VUX	AlphaVUE™ translation table
WRX	AlphaWRITE™ translation table

The internal format of these tables, as well as other function key translation tables used by AMOS, is described in Appendix D.

9.2.1 Loading a Function Key Translation Table

Your program should select the translation table to be used based on the name of the terminal driver in use. This can be located by first locating the job's terminal definition block (pointed to by JOBTRM within the job's JCB), then locating the terminal driver (pointed to by T.TDV within the terminal definition block). The terminal driver name is stored in the four bytes immediately preceding the terminal driver itself, packed RAD50.

For example, the following code will load the filename field of a DDB indexed by A4 with the name of the terminal driver in use. It then loads the DDB extension field with an extension of "AMX" and fetches the translation table from DSK0:[7,0].

```
JOBIDX  A0                  ; index job's JCB
MOV     JOBTRM(A0),A0       ; index job's TDB
MOV     T.TDV(A0),A0        ; index job's terminal driver
MOV     -(A0),D.FIL(A4)     ; get terminal driver name
MOVW    #[AMX],D.EXT(A4)    ; set extension
MOVW    #[DSK],D.DEV(A4)    ; set DSK0:
CLRW    D.DRV(A4)
MOVW    #<7_8.>+0,D.PPN(A4) ; set [7,0]
FETCH   @A4,A1              ; load table and index with A1
```

9.3 USING FUNCTION KEYS WITHIN YOUR APPLICATIONS SOFTWARE

In order for your applications program to receive the special function key codes, it must first perform two simple operations. The first is to place itself into "data" terminal input mode by setting the T$DAT flag (octal 10) in the terminal status word. It must then set the "function key translation" bit (T$XLT— octal 40) in the terminal status word to enable the function key routines in the terminal driver. Both of these status bits may be set in a single operation if desired.

Because use of the function keys requires that the program be in "data mode" it is best suited to those programs which receive and interpret keyboard input one character at a time. This type of input is best characterized by programs such as AlphaVUE or the INPUT.SBR portion of the AlphaACCOUNTING™ package.

When using function keys from higher-level languages, such as AlphaBASIC, it will be necessary to perform the manipulation of the terminal status word and the actual translation in an assembly language subroutine.

CHAPTER 10

PROGRAMMING COLOR TERMINALS

10.1 THE IMPACT OF COLOR

Color can add an important new dimension to your screen displays. In Chapter 5 we discussed how color is perceived and important aspects of its usage. This chapter describes how color control has been implemented in the AMOS terminal system and how you may use it in your programs.

10.2 SELECTING COLORS WITH SOFTWARE

Two different methods of selecting color on the screen are available. The first, which makes use of the terminal commands TCRT –1,132 through –1,147, is intended for use on the AM-70 and similar terminals only, and is considered to be obsolete for the purposes of developing new software. It is restricted to dealing with the eight additive primary colors as well as behaving as a "field" attribute, occupying a screen position and severely restricting how it can be used in formatted screen displays.

The second, and much more flexible, method of designating color makes use of the commands TCRT –2,n and TCRT –3,n. This method allows independent selection of foreground and background and provides for as many as 256 different foreground and background colors to be displayed on the screen at the same time. The use of these commands is recommended for all new software development.

10.3 USING FOREGROUND AND BACKGROUND COLOR SELECTION

The preferred method of color selection uses two ranges of TCRT calls— the –2 and –3 ranges— to allow independent specification of foreground and background colors using as many as 256 different colors. Not only does this method offer the most flexible selection of colors and color combinations, but it also does not require a screen position to change colors, allowing full utilization of the screen display space.

10.3.1 Selecting Foreground Color

To select the foreground display color, use the command TCRT –2,n, where the value "n" can range from 0 to 255 (decimal). This selects the color in which text will be displayed against the selected background color.

The value "n" is one of colors available from the color map discussed above. If you specify a color which is not available on the particular terminal in use, the closest available color will be used.

The TCRT –2,n command does not occupy a screen position. Further, color selected via this command should be treated as a "mode" attribute.

10.3.2 Selecting Background Color

To select the background display color, use the command TCRT –3,n, where the value "n" can range from 0 to 255 (decimal). This selects the color of the background against which the foreground color will be displayed.

The value "n" is one of colors available from the color map discussed above. If you specify a color which is not available on the particular terminal in use, the closest available color will be used.

The TCRT –3,n command does not occupy a screen position. Further, color selected via this command should be treated as a "mode" attribute.

10.3.3 Dealing with Color and Non-Color Terminals

The color selection TCRT commands are ignored on monochrome terminals. This allows you to place color selection commands in your software without requiring that you check to see if color is available on your terminal.

In cases where you wish the software to behave differently depending on whether color is available or not, your program can check the TD$CLR bit in the terminal capabilities flags returned by TRMCHR. This flag will be set for color terminals. In addition, TRMCHR returns the number of colors available.

10.3.4 Combining Color and Other Display Attributes

Color can be combined with other display attributes, such as underlining or reverse video, to create a wide variety of effects that enhance the appearance of the screen display. In most cases the combination of color and other attributes is straightforward. For example, selecting underline on a color terminal will simply display the underline in the foreground color. Selecting blinking text on a color terminal will blink the foreground color. Combining these attributes requires no special action on the programmer’s part.

Reverse video, however, is somewhat of a special case. When you select reverse video on a color terminal, you change the sense of foreground and background. When reverse video is in effect, changing the foreground color via the TCRT –2,n call will affect the background of the displayed text. Likewise, changing the background color will affect the foreground of the displayed text. This affect is undone by the end reverse video command.

Let's look at some examples. To take the simplest case first, assume that the current colors are yellow foreground text on a black background. Selecting the reverse video attribute will cause all subsequent text to be displayed as black characters on a yellow background. Ending the reverse video attribute will cause subsequent text to once again be displayed as yellow characters on a black background.

Lets see what happens when we start with yellow on black, select reverse video, and then change the foreground color to red (via a TCRT –2,n) call. In this case, subsequent text will be displayed as black characters on a red background. Furthermore, ending the reverse video attribute will cause later characters to be displayed as red on black.

10.3.5 Maintaining the Default Foreground and Background Colors

Programs which change the foreground and background colors must be careful to restore the user's selected foreground and background colors upon exiting. Because each user may have a particular choice of colors, which may be wildly different from your own, it is important that any program which manipulates the color settings first save the original color settings and restore them upon exit. The current color settings can be determined by using the TRMCHR call.

10.4 USING AM-70 COMPATIBLE COLOR SELECTION

The AM-70 color alphanumeric terminal uses a different method for selecting colors than the one described above. Based on an older technology, the AM-70 implements colors as field attributes. This means that changing colors occupies a screen position, reducing the amount of text that can be placed on the screen. In addition, since the colors are actually implemented in hardware as attributes, you lose the ability to use other terminal attributes such as blinking or underlining. Because there is a limited number of attributes available, the color selection is also limited: you can select eight different colors of text on a black background, or black text on a background of eight different colors.

While quite limited in its ability to display colors flexibly, the advantages of having any color at all can outweigh the rather severe limitations in the color selection. For those applications which can benefit from color, 16 TCRT calls have been defined through which you can select color on an AM-70 terminal.

The TCRT calls –1,132 through –1,139 are used to select different colors of text on a black background. The calls are assigned as follows:

–1,133	select white text on black background
–1,134	select blue text on black background
–1,135	select magenta text on black background
–1,136	select red text on black background
–1,137	select yellow text on black background
–1,138	select green text on black background
–1,139	select cyan text on black background

The TCRT calls –1,140 through –1,147 are used to select black text on different colored backgrounds. The calls are assigned as follows:

–1,140	select black text on black background
–1,141	select black text on white background
–1,142	select black text on blue background
–1,143	select black text on magenta background
–1,144	select black text on red background
–1,145	select black text on yellow background
–1,146	select black text on green background
–1,147	select black text on cyan background

These calls are considered to be obsolete and are provided primarily to support older applications with limited color selection capability.

Because the presence of colors removes the availability of some of the standardly available attributes— most notably the blinking and underline attributes— some application systems assign the TCRT calls which generate those attributes to color selection instead. This technique makes sure that programs which try to use the missing attributes still have some distinguishing attribute on the screen, but reliance on this technique for color selection will cause the programs to have unpredictable results on terminals which fully implement color selection and attributes.

CHAPTER 11

USING THE ALTERNATE CHARACTER SET

In addition to the standard 96 displayable ASCII characters, many terminals contain additional characters and symbols that can be displayed on the screen. These additional characters include line drawing characters, block graphics characters and special word processing symbols. These additional characters are said to belong to an "alternate character set."

This chapter describes these various symbols and how to display them on the terminal in a completely terminal independent method.

11.1 USING THE ALTERNATE CHARACTER SET

Once again we are in a situation where different terminals handle things with a wide variety of methods. This section will describe a terminal independent way of displaying alternate characters.

Typical methods employed by terminal manufacturers for displaying symbols from the alternate character set include "hiding" the alternate characters in the normally non-displayable control characters, preceding special characters with a special lead-in code, or having a special "alternate character set" mode where the actual meaning of all characters change.

To make software that works with all of these methods, the following procedure should be followed whenever alternate character set symbols are to be displayed:

1. Position the cursor to where the special character is to be displayed.

2. Enter the Alternate Character Set Mode via the TCRT –1,23 command.

3. Send the command for the character you wish to display (TCRT –1,38 through –1,53, or TCRT –1,64 through –1,78).

4. Continue sending alternate character commands until you are finished with the line or until you need to reposition the cursor. Repositioning the cursor includes carriage-return, line-feed, backspace, cursor addressing commands, and any other command which causes the cursor to move (except outputting the special character, or course).

5. Exit the Alternate Character Set Mode via the TCRT –1,24 command.

Repeat this sequence (steps 1 through 5) until you are finished.

11.2 THE LINE DRAWING CHARACTERS

The line drawing characters are used for drawing boxes, rules, and lines. The character assignments are as follows:

Command Code	Symbol Displayed
–1,38	Top left corner
–1,39	Top right corner
–1,40	Bottom left corner
–1,41	Bottom right corner
–1,42	Top intersection
–1,43	Right intersection
–1,44	Left intersection
–1,45	Bottom intersection
–1,46	Horizontal line
–1,47	Vertical line
–1,48	Center intersection
–1,52	Double horizontal line
–1,53	Double vertical line

11.3 THE BLOCK GRAPHICS CHARACTERS

The block graphics characters are typically used for filling in areas of the screen for bar charts and other special effects. The character assignments are as follows:

Command Code	Symbol Displayed
–1,49	Solid block
–1,50	Slanted line block
–1,51	Cross-hatch block

11.4 THE WORD PROCESSING CHARACTERS

The word processing characters contain special characters typically used in an office environment. The characters are as follows:

Command Code	Symbol Displayed
–1,64	Up arrow
–1,65	Down arrow
–1,66	Raised dot
–1,67	End of line marker
–1,68	Horizontal tab symbol
–1,69	Paragraph symbol
–1,70	Dagger symbol
–1,71	Section symbol
–1,72	Cent sign
–1,73	One-quarter symbol
–1,74	One-half symbol
–1,75	Degree symbol
–1,76	Trademark symbol
–1,77	Copyright symbol
–1,78	Registered symbol

11.5 DEALING WITH CHARACTERS WHICH ARE NOT PRESENT

Many terminals do not possess all (or even any) of the special characters described in this chapter. In this case the terminal driver must attempt to "emulate" the characters by substituting characters it does have for the missing characters.

This can be very successful in the case of characters such as the line drawing symbols where the use of standard ASCII characters (such as "!", "-" and "+") instead of true line drawing characters works very well. While the screen display may not be as attractive as when the line drawing characters are present, no information content is lost.

The same is not necessarily true for some of the other characters. In some cases where no existing character is even "close," a space may be substituted for the missing character. Clearly in some cases this will cause a loss of information content in the screen display. Be sure to check a given terminal for its level of alternate character set support if you wish to use it with software that makes extensive use of the alternate character set.

At all times, however, the display of an alternate character set symbol will occupy one screen position, even if all that is displayed is a space, thus preserving the vertical alignment of screen displays regardless of the available alternate character set symbols.

CHAPTER 12

USING THE PRINTER PORT

Some terminals contain the ability to be directly connected to a printer in addition to the host computer. This method of connection is referred to as a *printer port*.

This printer port can be used for a variety of purposes. The two primary uses are printing a copy of the current screen contents (Print Screen), and performing localized printing, bypassing the terminal (Transparent Print). In situations where the terminal is connected to the host computer via a modem, use of the printer port to connect a printer to the host may be far preferable to the cost of another set of modems and another phone line.

In addition to these two uses, which are described in more detail later in this chapter, the printer port can also be used to connect a variety of serial devices to the host. Examples of such devices might be alternate input devices such as mice, trackballs, and digitizers that would serve to augment the standard keyboard.

Because not all terminals contain a printer port, a TRMCHR flag has been defined to denote the presence of the feature. This flag must be checked prior to any attempt to use the printer port. If this flag is not set and you use the printer port commands, the results will be unpredictable.

12.1 SCREEN PRINT

While it might seem to be the simplest use of the printer port, the Screen Print operation (TCRT code –1,79) is actually one of the more complicated uses. In theory, all this command must do is cause an exact copy of the current screen display to be printed on the printer attached to the printer port.

However, each terminal will execute this command based on its own unique idea as to what is appropriate to be sent to the printer. Some terminals send every character, at every screen location, including attribute codes and nulls which may not be correctly interpreted by the local printer. Others will replace "non-printing" characters with spaces. Some terminate each line with a carriage-return, while others send no line terminator at all. In addition, characters that are sent by the printer to the terminal may or may not be sent back to the host. Some terminals do not accept keyboard input during the print operation, while others may send keyboard input either to the host or to the printer or both.

As can be seen, using the printer port is not necessarily a simple matter. However, most of the work in creating a working combination of host software, terminal, and printer is in choosing compatible items. To maintain compatibility between a wide variety of terminal/printer combinations, emphasis should be placed on acquiring the correct terminal and printer, rather than customizing the host software, since this defeats any device independence.

One area of concern is error recovery during a print operation. Because some terminals lock out the keyboard during a print operation, if the printer should fault (e.g., out of paper) there is no simple way to regain control of the application software from the keyboard. For this reason, timeouts or other escape mechanisms should be built into software which makes extensive use of the local printer capability.

12.2 TRANSPARENT PRINT

The second method of using the printer port is known as "transparent print." In this mode all characters the host sends to the terminal are immediately sent to the printer without any effect on the terminal screen.

A program causes the terminal to enter transparent print mode by issuing the Enter Transparent Print Mode call (TCRT –1,82). Once all desired data has been sent to the printer, the program sends the Exit Transparent Print Mode call (TCRT –1,83) to return the terminal to the normal conversational mode of operation.

As with the Screen Print function, some characteristics of this function are undefined as they depend on the particular combination of terminal and printer in use. In particular, input from the keyboard may or may not be acted on. In addition, any XON/XOFF protocol used by the printer may or may not be returned to the host.

Like Screen Print, attention must be paid to the error recovery aspects of using the Transparent Print mode.

CHAPTER 13

MISCELLANEOUS FEATURES

This chapter descibes a series of commands that perform special display functions on the terminal screen. Typically they do not do anything that could not be accomplished by using the simpler screen commands or by simply repainting the screen; instead, what they offer is better performance (by reducing the time to update the screen display) or a more pleasing appearance for the user.

13.1 BLOCK FILL

There are times when it is desirable to fill a rectangular area of the screen with a repeating character. Most commonly, the space character can be used to "clear" a rectangular region of the screen. This is often used for removing menus from the screen.

Two terminal commands allow you to "fill" a rectangular area of the screen with a specific character or attribute.

To fill an area of the screen with a character, first position the cursor at the upper left corner of the area to be filled. Issue a TCRT –1,91 call followed by height and width codes taken from Table B-5. Next output the single character you wish to have the area filled with. The screen display will then be updated.

To fill an area of the screen with an attribute, first position the cursor at the upper left corner of the area to be filled. Issue a TCRT –1,90 call followed by height and width codes taken from Table B-5. Next output the attribute code (taken from Table B-6) you wish to have the area filled with. The screen display will then be updated.

All validation of the size of the block to be filled is left to the application program. Using these commands with a block specified such that it will extend beyond the visible screen boundaries will have unpredictable results.

These functions are available only if the TD$BLF bit is set in the terminal capabilities flags.

13.2 SPLIT SCREENS

To make it easier to maintain and update screen displays, some terminals offer what is known as a "split screen" capability. This capability allows you to divide the screen into two seperate and independent segments. These two segments are completely independent of each other and behave as if they were actually seperate screens. By this we mean that a clear screen command sent to one screen segment will only clear that segment, leaving the other untouched. Likewise, scrolling text on the screen will only affect the text in the currently selected segment.

This capability can be very useful when you are displaying two seperate types of information. These two types may be two different documents, status information and data, or any other combination of text that you wish to be able to update independent of each other.

Most terminals that support split screen only allow the split to be done in the horizontal dimension; that is, you can divide the screen into upper and lower segments. A very few terminals allows you to divide the screen vertically: into left and right segments. So few terminals support this vertical split capability, however, that its use should be avoided except for very specialized applications where the reduction in terminal independence is not critical.

Commands are provided to split the screen into upper and lower segments at any line. A segment can be as small as a single line or as large as one line less than the height of the screen. The upper segment is referred to as segment 0 and the lower is referred to as segment 1. You select a horizontal split screen by issuing the TCRT –1,57 call and following it with the number of rows you wish segment 0 to contain, in a coded format. Details on using this call can be found in Chapter 15. Splitting the screen will clear both segments.

Once the screen has been split, you can select one segment or the other by using the TCRT –1,61 and TCRT –1,62 calls. The cursor will only appear in the currently selected segment. The cursor position is remembered for both segments. Thus positioning the cursor in one segment, then selecting the other will cause the displayed cursor to change positions on the screen. However, once the original segment is once again selected, the cursor will move back to its original position within that segment.

To remove the split screen, use the TCRT –1,56 call to reset the terminal to its original state.

The split screen commands are only available if the TD$SPL bit is set in the terminal capabilities word.

13.3∞ALTERNATE PAGE

Some terminals offer a feature known as an "alternate page." Normally your terminal screen displays what is known among terminal manufacturers as a "page" of memory. By installing additional memory inside the terminal, the terminal can contain another "page" in addition to the one being displayed on the terminal screen.

The advantage of this feature is that the terminal is capable of switching the screen display from one "page" to the other very quickly. This can provide the appearance of a screen update far in excess of the speed with which it can be updated by clearing and repainting it.

By sending text to the alternate page (via the TCRT –1,84 call) you can create an entire screen display without altering the one currently being displayed. By selecting the alternate page, all commands and text you send to the terminal will not be displayed, but instead will only affect the as yet invisible alternate page of memory. After exiting this alternate page mode (via the TCRT –1,85 call), you can quickly change to displaying the alternate page by issuing a TCRT –1,86 call.

When you select the "alternate page" it becomes the current page and the previously displayed page becomes the alternate page. Thus the TCRT –1,86 command simply toggles between the two pages, with the one being displayed being the "current page" and the one that is not being displayed being the "alternate page." By repeatedly sending this TCRT –1,86 call, you can switch rapidly back and forth between the two displays.

Typical uses for the alternate page include storing a "help" screen in the alternate page to make additional information available to the user very quickly. Additionally, by rapidly switching between pages, limited animation effects can be realized.

Since successive screen displays are often only slight modifications of the previous one, another helpful command is available on some terminals. This command creates a copy of the currently displayed screen in the alternate page of terminal memory. By being able to quickly make a copy, then having to send only the updates, you can reduce the screen update time by avoiding having to send the entire screen image again to update the alternate page.

The alternate page commands are available only if the TD$ALP bit is set in the terminal capabilities word.

13.4 CURSOR SHAPE CONTROL

The shape of the cursor is often a matter of personal choice for the user of the terminal. Typically, the cursor shape can be chosen from several different possibilities via switches on the terminal or through a terminal set up procedure.

There are times, however, when it is useful to be able to select the shape of the cursor via software control. For this reason, four terminal commands (TCRT –1,120 through –1,123) allow you to select the cursor shape.

In addition to controlling the shape of the cursor, commands are also available to turn the cursor on and off. When updating the screen, turning the cursor off during the update can make the operation much more pleasing to the eye by eliminating any "ghosting" of the cursor on the screen. Use TCRT –1,29 to turn the cursor off and TCRT –1,28 to turn it back on.

If a particular terminal does not support one or more of these commands, the command will simply be ignored.

13.5 VIDEO ENABLE/DISABLE

In addition to turning the cursor off while updating the screen, some terminals allow you to turn off the video display during updates, further reducing the visual "clutter" of updating the screen. When the entire screen is to be repainted, you can disable the video, blanking the screen display, by using a TCRT –1,36 command. Once the update is complete, turn the screen display back on via the TCRT –1,37 command.

13.6 INSERT/DELETE LINE, CHARACTERS AND COLUMNS

There are many times when a screen update consists simply of deleting an unwanted character or line from the screen, or inserting a new character or line. Instead of requiring that you redisplay large portions of the screen, most terminals offer the ability to insert and delete individual characters and lines. Text will automatically be moved to adjust for the insertion or deletion, without requiring special programming to update the screen. Some terminals also offer the ability to insert and delete columns of text.

Characters can be inserted and deleted via the TCRT –1,18 and TCRT –1,17 commands. By positioning the character and issuing one of these calls, text can be moved very quickly, with a minimum of distracting screen redisplay. Entire lines can be inserted and deleted via the TCRT –1,16 and TCRT –1,15 calls.

The character insert and delete functions are available only if the TD$CID bit is set in the terminal capabilities word. The line insert and delete functions are available only if the TD$LID bit is set in the terminal capabilities word.

In addition to character and line manipulation, some terminals offer the ability to insert and delete columns of text via the TCRT –1,88 and TCRT –1,89 calls. These operations are analgous to the the line insert and delete functions, but takes place in the vertical dimension. The column insert and delete functions are only available if the TD$KID bit is set in the terminal capabilities word.

13.7 BOX COMMANDS

When formatting a screen display for easy readability, it is often helpful to use the line drawing characters to help identify groups of similar items on the screen. Often boxes are drawn around items to group them and make it easier for the terminal operator to identify like items. While this can improve the readablity of the screen, repetitive drawing of boxes on the screen can consume much time, resulting in the application appearing "slow," irrespective of the actual speed of the software. For this reason, some terminals (such as the AM-62A) support a set of commands which can be used to draw and maniuplate "boxes" on the screen.

All validation of the size of the boxes used in these commands is left to the application program. Using these commands with a box specified such that it will extend beyond the visible screen boundaries will have unpredictable results.

These functions are available only if the TD$BOX bit is set in the terminal capabilities word.

13.7.1 Drawing Boxes

One command allows you to draw a "box" on the screen. This box encloses a rectangular area of the screen with line drawing graphics characters. Note that only the outside lines of the box are drawn on the screen; character inside and outside the box are unaffected by this command. If you wish to clear the inside of the box, the use of the Block Fill with Character command (TCRT –1,91) is recommended.

To draw a box, first position the cursor at the upper left corner of the box. Issue a TCRT –1,92 call followed by height and width codes taken from Table B-5. The screen display will then be updated.

13.7.2 Scrolling Boxes

Two commands allow you to "scroll" a rectangular area of the screen by copying that area up or down one line. This is useful when using a rectangular area of the screen as a "window" onto text.

To scroll an area of the screen up one line, first position the cursor at the upper left corner of the area to be scrolled. Issue a TCRT –1,93 call followed by height and width codes taken from Table B-5. The text within the rectangle will now be moved up one line. Note that the "last" line in the rectangle has been duplicated by the copy operation. If you wish the last line to be cleared, your program must perform that operation itself.

To scroll an area of the screen down one line, first position the cursor at the lower left corner of the area to be scrolled. Issue a TCRT –1,94 call followed by height and width codes taken from Table B-5. The text within the rectangle will now be moved down one line. Note that the "first" line in the rectangle has been duplicated by the copy operation. If you wish the first line to be cleared, your program must perform that operation itself.

13.8 VARIABLE SCROLL RATES

Video terminals have traditionally used a method of scrolling the screen display known as "jump scrolling." In this mode each line simply "disappears" off the top of the screen to be replaced with the new line. Newer terminals have implemented a method of scrolling known as "smooth scrolling," where the text of the line is rolled off the top of the screen one character dot at a time.

Smooth scrolling is generally considered to be more visually pleasing than jump scrolling, but restricts the rate at which information can be scrolled on the screen. Therefore it is usually used under special circumstances rather than as the normal mode of operation.

Terminal command calls (TCRT –1,96 through TCRT –1,99) are provided for up to four different smooth scrolling rates, although not all terminals provide all of these different scroll rates. The TCRT –1,95 command selects jump scrolling.

The smooth scrolling commands are available only if the TD$SMT bit is set in the terminal capabilities flags.

PART IV

THE TCRT COMMAND CODES

CHAPTER 14

AMOS MONITOR TCRT CODES

14.1 INDEX TO QUICK REFERENCE

This chapter provides two quick references to the available TCRT commands: one is arranged by TCRT command number and the other is arranged by function.

More detailed information on each command can be found in Chapter 15.

14.1.1 TCRT Codes by Number

TCRT –1, 0	clear screen
TCRT –1, 1	cursor home (move to column 1,1)
TCRT –1, 2	cursor return (move to column 1)
TCRT –1, 3	cursor up
TCRT –1, 4	cursor down
TCRT –1, 5	cursor left
TCRT –1, 6	cursor right
TCRT –1, 7	lock keyboard
TCRT –1, 8	unlock keyboard
TCRT –1, 9	erase to end of line
TCRT –1, 10	erase to end of screen
TCRT –1, 11	reduced intensity
TCRT –1, 12	normal intensity
TCRT –1, 13	enable protected fields
TCRT –1, 14	disable protected fields
TCRT –1, 15	delete line
TCRT –1, 16	insert line
TCRT –1, 17	delete character
TCRT –1, 18	insert character
TCRT –1, 19	read cursor address
TCRT –1, 20	read character at current cursor position
TCRT –1, 21	start blink field
TCRT –1, 22	end blink field
TCRT –1, 23	start graphics character mode
TCRT –1, 24	end graphics character mode
TCRT –1, 25	set horizontal position (obsolete)
TCRT –1, 26	set vertical position (obsolete)
TCRT –1, 27	set terminal attributes
TCRT –1, 28	cursor on
TCRT –1, 29	cursor off

TCRT –1, 30	start underscore
TCRT –1, 31	end underscore
TCRT –1, 32	start reverse video
TCRT –1, 33	end reverse video
TCRT –1, 34	start reverse blink
TCRT –1, 35	end reverse blink
TCRT –1, 36	turn off screen display
TCRT –1, 37	turn on screen display
TCRT –1, 38	top left corner character
TCRT –1, 39	top right corner character
TCRT –1, 40	bottom left corner character
TCRT –1, 41	bottom right corner character
TCRT –1, 42	top intersect character
TCRT –1, 43	right intersect character
TCRT –1, 44	left intersect character
TCRT –1, 45	bottom intersect character
TCRT –1, 46	horizontal line character
TCRT –1, 47	vertical line character
TCRT –1, 48	intersection character
TCRT –1, 49	solid block character
TCRT –1, 50	slant block character
TCRT –1, 51	cross–hatch block character
TCRT –1, 52	double line horizontal character
TCRT –1, 53	double line vertical character
TCRT –1, 54	send message to unshifted bottom status line
TCRT –1, 55	send message to shifted bottom status line
TCRT –1, 56	set normal display format
TCRT –1, 57	set horizontal split (follow with row code)
TCRT –1, 58	set vertical split (39 char columns)
TCRT –1, 59	set vertical split (40 char columns)
TCRT –1, 60	set vertical split column to next char
TCRT –1, 61	activate split segment 0
TCRT –1, 62	activate split segment 1
TCRT –1, 63	send message to top status line
TCRT –1, 64	up-arrow character
TCRT –1, 65	down-arrow character
TCRT –1, 66	raised dot character
TCRT –1, 67	end of line marker character
TCRT –1, 68	horizontal tab character
TCRT –1, 69	paragraph character
TCRT –1, 70	dagger character
TCRT –1, 71	section character
TCRT –1, 72	cent sign character
TCRT –1, 73	one-quarter character
TCRT –1, 74	one-half character
TCRT –1, 75	degree character
TCRT –1, 76	trademark character
TCRT –1, 77	copyright character
TCRT –1, 78	registered character
TCRT –1, 79	print screen
TCRT –1, 80	set to 132 column mode
TCRT –1, 81	set to 80 column mode
TCRT –1, 82	enter transparent print mode
TCRT –1, 83	exit transparent print mode

TCRT –1, 84	begin writing to alternate page
TCRT –1, 85	end writing to alternate page
TCRT –1, 86	toggle page
TCRT –1, 87	copy to alternate page
TCRT –1, 88	insert column
TCRT –1, 89	delete column
TCRT –1, 90	block fill with attribute
TCRT –1, 91	block fill with character
TCRT –1, 92	draw a box
TCRT –1, 93	scroll box up one line
TCRT –1, 94	scroll box down one line
TCRT –1, 95	select jump scroll
TCRT –1, 96	select fast smooth scroll
TCRT –1, 97	select med-fast smooth scroll
TCRT –1, 98	select med-slow smooth scroll
TCRT –1, 99	select slow smooth scroll
TCRT –1,100	start underscore/blink
TCRT –1,101	end underscore/blink
TCRT –1,102	start underscore/reverse
TCRT –1,103	end underscore/reverse
TCRT –1,104	start underscore/reverse/blink
TCRT –1,105	end underscore/reverse/blink
TCRT –1,106	start underscore w/o space
TCRT –1,107	end underscore w/o space
TCRT –1,108	start reverse w/o space
TCRT –1,109	end reverse w/o space
TCRT –1,110	start reverse/blinking w/o space
TCRT –1,111	end reverse/blinking w/o space
TCRT –1,112	start underscore/blinking w/o space
TCRT –1,113	end underscore/blinking w/o space
TCRT –1,114	start underscore/reverse w/o space
TCRT –1,115	end underscore/reverse w/o space
TCRT –1,116	start underscore/reverse/blink w/o space
TCRT –1,117	end underscore/reverse/blink w/o space
TCRT –1,118	start blink w/o space
TCRT –1,119	end blink w/o space
TCRT –1,120	set cursor to blinking block
TCRT –1,121	set cursor to steady block
TCRT –1,122	set cursor to blinking underline
TCRT –1,123	set cursor to steady underline
TCRT –1,124	RESERVED
TCRT –1,125	RESERVED
TCRT –1,126	RESERVED
TCRT –1,127	RESERVED
TCRT –1,128	select top status line w/o address
TCRT –1,129	end status line
TCRT –1,130	select unshifted status line w/o addr
TCRT –1,131	select shifted status line w/o addr
TCRT –1,132	select black text (obsolete)
TCRT –1,133	select white text (obsolete)
TCRT –1,134	select blue text (obsolete)
TCRT –1,135	select magenta text (obsolete)
TCRT –1,136	select red text (obsolete)
TCRT –1,137	select yellow text (obsolete)

TCRT –1,138	select green text (obsolete)
TCRT –1,139	select cyan text (obsolete)
TCRT –1,140	select black reverse text (obsolete)
TCRT –1,141	select white reverse text (obsolete)
TCRT –1,142	select blue reverse text (obsolete)
TCRT –1,143	select magenta reverse text (obsolete)
TCRT –1,144	select red reverse text (obsolete)
TCRT –1,145	select yellow reverse text (obsolete)
TCRT –1,146	select green reverse text (obsolete)
TCRT –1,147	select cyan reverse text (obsolete)
TCRT –1,148	save a rectangular area
TCRT –1,149	restore a rectangular screen area
TCRT –1,150	enter full graphics mode
TCRT –1,151	exit full graphics mode
TCRT –1,152	draw a box with rounded corners
TCRT –1,153	draw a window style box
TCRT –1,154	draw a box with double lines
TCRT –1,155	enable proportionally spaced text
TCRT –1,156	disable proportionally spaced text
TCRT –1,157	select color palette by RGB value
TCRT –1,158	enable graphics cursor
TCRT –1,159	disable graphics cursor
TCRT –1,160	select graphics cursor shape
TCRT –1,161	include graphics cursor location
TCRT –1,162	define graphics cursor regions
TCRT –1,163	output form feed character
TCRT –1,164	output line feed character
TCRT –1,165	output new line character
TCRT –1,166	output vertical tab character
TCRT –1,167	output plus-or-minus character
TCRT –1,168	output greater-than-or-equal character
TCRT –1,169	output less-than-or-equal character
TCRT –1,170	output not-equal character
TCRT –1,171	output British pound character
TCRT –1,172	output Pi character
TCRT –1,173	enter bidirectional print mode
TCRT –1,174	exit bidirectional print mode
TCRT –1,175	set terminal time
TCRT –1,176	set terminal date
TCRT –1,177	select color palette by HLS value
TCRT –1,178	select PC character set
TCRT –1,179	select default character set
TCRT –1,180	select PC terminal emulation
TCRT –1,181	select ASCII terminal operation
TCRT –1,182	select AUX port host
TCRT –1,183	select main port host
TCRT –1,184	select toggle host ports
TCRT –1,185	select 8-bit character display
TCRT –1,186	select 7-bit character display
TCRT –1,187	select 8-bit keyboard mode
TCRT –1,188	select 7-bit keyboard mode
TCRT –1,189	select primary printer port
TCRT –1,190	select secondary printer port
TCRT –1,191	select 161 column display mode

TCRT –1,192 thru 255	RESERVED
TCRT –2, n	select foreground color n
TCRT –3, n	select background color n
TCRT –4, n thru –5, n	RESERVED

14.1.2 TCRT Codes by Function

14.1.2.1 Cursor Positioning Commands

TCRT –1, 1	cursor home (move to column 1,1)
TCRT –1, 2	cursor return (move to column 1)
TCRT –1, 3	cursor up
TCRT –1, 4	cursor down
TCRT –1, 5	cursor left
TCRT –1, 6	cursor right

14.1.2.2 Screen Clearing Commands

TCRT –1, 0	clear screen
TCRT –1, 9	erase to end of line
TCRT –1, 10	erase to end of screen

14.1.2.3 Display Attribute Commands

TCRT –1, 11	reduced intensity
TCRT –1, 12	normal intensity
TCRT –1, 21	start blink field
TCRT –1, 22	end blink field
TCRT –1, 27	set terminal attributes
TCRT –1, 30	start underscore
TCRT –1, 31	end underscore
TCRT –1, 32	start reverse video
TCRT –1, 33	end reverse video
TCRT –1, 34	start reverse blink
TCRT –1, 35	end reverse blink
TCRT –1,100	start underscore/blink
TCRT –1,101	end underscore/blink
TCRT –1,102	start underscore/reverse
TCRT –1,103	end underscore/reverse
TCRT –1,104	start underscore/reverse/blink
TCRT –1,105	end underscore/reverse/blink
TCRT –1,106	start underscore w/o space
TCRT –1,107	end underscore w/o space
TCRT –1,108	start reverse w/o space
TCRT –1,109	end reverse w/o space

TCRT –1,110	start reverse/blinking w/o space
TCRT –1,111	end reverse/blinking w/o space
TCRT –1,112	start underscore/blinking w/o space
TCRT –1,113	end underscore/blinking w/o space
TCRT –1,114	start underscore/reverse w/o space
TCRT –1,115	end underscore/reverse w/o space
TCRT –1,116	start underscore/reverse/blink w/o space
TCRT –1,117	end underscore/reverse/blink w/o space
TCRT –1,118	start blink w/o space
TCRT –1,119	end blink w/o space

14.1.2.4 Status Line Commands

TCRT –1, 54	send message to unshifted bottom status line
TCRT –1, 55	send message to shifted bottom status line
TCRT –1, 63	send message to top status line
TCRT –1,128	select top status line w/o address
TCRT –1,129	end status line
TCRT –1,130	select unshifted status line w/o addr
TCRT –1,131	select shifted status line w/o addr

14.1.2.5 Graphics Character Set Commands and Characters

TCRT –1, 23	start graphics character mode
TCRT –1, 24	end graphics character mode
TCRT –1, 38	top left corner
TCRT –1, 39	top right corner
TCRT –1, 40	bottom left corner
TCRT –1, 41	bottom right corner
TCRT –1, 42	top intersect
TCRT –1, 43	right intersect
TCRT –1, 44	left intersect
TCRT –1, 45	bottom intersect
TCRT –1, 46	horizontal line
TCRT –1, 47	vertical line
TCRT –1, 48	intersection
TCRT –1, 49	solid block
TCRT –1, 50	slant block
TCRT –1, 51	cross-hatch block
TCRT –1, 52	double line horizontal
TCRT –1, 53	double line vertical
TCRT –1, 64	up-arrow
TCRT –1, 65	down-arrow
TCRT –1, 66	raised dot
TCRT –1, 67	end of line marker
TCRT –1, 68	horizontal tab symbol
TCRT –1, 69	paragraph
TCRT –1, 70	dagger
TCRT –1, 71	section
TCRT –1, 72	cent sign
TCRT –1, 73	one-quarter
TCRT –1, 74	one-half

TCRT –1, 75	degree
TCRT –1, 76	trademark
TCRT –1, 77	copyright
TCRT –1, 78	registered

14.1.2.6 Protected Field Commands

TCRT –1, 13	enable protected fields
TCRT –1, 14	disable protected fields

14.1.2.7 Local Printer Commands

TCRT –1, 79	print screen
TCRT –1, 82	enter transparent print mode
TCRT –1, 83	exit transparent print mode
TCRT –1, 173	enter bidirectional print mode
TCRT –1, 174	exit bidirectional print mode

14.1.2.8 Insertion and Deletion Commands

TCRT –1, 15	delete line
TCRT –1, 16	insert line
TCRT –1, 17	delete character
TCRT –1, 18	insert character
TCRT –1, 88	insert column
TCRT –1, 89	delete column

14.1.2.9 Variable Speed Scroll Commands

TCRT –1, 95	select jump scroll
TCRT –1, 96	select fast smooth scroll
TCRT –1, 97	select med-fast smooth scroll
TCRT –1, 98	select med-slow smooth scroll
TCRT –1, 99	select slow smooth scroll

14.1.2.10 Split Screen Commands

TCRT –1, 56	set normal display format
TCRT –1, 57	set horizontal split (follow with row code)
TCRT –1, 58	set vertical split (39 char columns)
TCRT –1, 59	set vertical split (40 char columns)
TCRT –1, 60	set vertical split column to next char
TCRT –1, 61	activate split segment 0
TCRT –1, 62	activate split segment 1

14.1.2.11 Block Fill Commands

TCRT –1, 90	block fill with attribute
TCRT –1, 91	block fill with character

14.1.2.12 Box Commands

TCRT –1, 92	draw a box
TCRT –1, 93	scroll box up one line
TCRT –1, 94	scroll box down one line
TCRT –1, 152	draw a box with rounded corner
TCRT –1, 153	draw a window style box
TCRT –1, 154	draw a box with double lines

14.1.2.13 Alternate Page Commands

TCRT –1, 84	begin writing to alternate page
TCRT –1, 85	end writing to alternate page
TCRT –1, 86	toggle page
TCRT –1, 87	copy to alternate page

14.1.2.14 Color Commands

TCRT –2, n	select foreground color n
TCRT –3, n	select background color n
TCRT –1,132	select black text (obsolete)
TCRT –1,133	select white text (obsolete)
TCRT –1,134	select blue text (obsolete)
TCRT –1,135	select magenta text (obsolete)
TCRT –1,136	select red text (obsolete)
TCRT –1,137	select yellow text (obsolete)
TCRT –1,138	select green text (obsolete)
TCRT –1,139	select cyan text (obsolete)
TCRT –1,140	select black reverse text (obsolete)
TCRT –1,141	select white reverse text (obsolete)
TCRT –1,142	select blue reverse text (obsolete)
TCRT –1,143	select magenta reverse text (obsolete)
TCRT –1,144	select red reverse text (obsolete)
TCRT –1,145	select yellow reverse text (obsolete)
TCRT –1,146	select green reverse text (obsolete)
TCRT –1,147	select cyan reverse text (obsolete)
TCRT –1,157	select color palette by RGB value
TCRT –1,177	select color palette by HLS value

14.1.2.15∞Miscellaneous Commands

TCRT –1, 7	lock keyboard
TCRT –1, 8	unlock keyboard
TCRT –1, 19	read cursor address
TCRT –1, 20	read character at current cursor position
TCRT –1, 28	cursor on
TCRT –1, 29	cursor off
TCRT –1, 36	turn off screen display
TCRT –1, 37	turn on screen display
TCRT –1, 80	set to 132 column mode
TCRT –1, 81	set to 80 column mode
TCRT –1,120	set cursor to blinking block
TCRT –1,121	set cursor to steady block
TCRT –1,122	set cursor to blinking underline
TCRT –1,123	set cursor to steady underline
TCRT –1,148	save a rectangular area
TCRT –1,149	restore a rectangular screen area
TCRT –1,150	enter full graphics mode
TCRT –1,151	exit full graphics mode
TCRT –1,155	enable proportionally spaced text
TCRT –1,156	disable proportionally spaced text
TCRT –1,158	enable graphics cursor
TCRT –1,159	disable graphics cursor
TCRT –1,160	select graphics cursor shape
TCRT –1,161	select graphics cursor location
TCRT –1,162	select graphics cursor regions
TCRT –1,176	set terminal date
TCRT –1,178	select PC character set
TCRT –1,179	select PC default character set
TCRT –1,180	select PC terminal emulation
TCRT –1,181	select ASCII terminal operation
TCRT –1,182	select AUX port host
TCRT –1,183	select main port host
TCRT –1,184	select toggle host ports
TCRT –1,185	select 8-bit character display
TCRT –1,186	select 7-bit character display
TCRT –1,187	select 8-bit keyboard mode
TCRT –1,188	select 7-bit keyboard mode
TCRT –1,189	select primary printer port
TCRT –1,190	select secondary printer port
TCRT –1,191	select 161-column display mode

14.2∞RESERVED CODES

In the interest of maximizing the compatibility between software packages developed to run on the Alpha Micro computer, and to minimize unwanted interactions between such packages, all unassigned TCRT codes are reserved for assignment by Alpha Micro. If you have a special application or terminal which requires additional TCRT codes, please contact Alpha Microsystems to request having a TCRT code assigned.

CHAPTER 15

TERMINAL CONTROL COMMANDS

The key to the terminal independence offered by AMOS is a set of terminal control functions which perform the same function, and have the same effect on the terminal screen, regardless of the characteristics of the terminal being used.

Because of the wide variety of terminals, and their broad range of capabilities, in addition to the standard terminal functions which all terminals are assumed to have, a number of the commands described below may or may not be present on a particular terminal. In these cases, the command is either ignored, where this will have no material effect on the screen display, or are enabled by a flag that tells a program that a particular feature is present. These flags may be obtained for the terminal that is currently in use via the TRMCHR assembly language monitor call (or the TRMCHR XCALL subroutine for AlphaBASIC users).

It is the responsibility of the program to check these flags and not use any feature that does not have its corresponding flag set.

Using a feature that does not have its flag set will result in unpredictable results ranging from visually unpleasant displays to locking up ("crashing") the terminal.

15.1 CLEAR SCREEN (TCRT –1,0)

This call clears the current terminal screen segment and positions the cursor at row 1, column 1.

Because some terminals require an extended period of time to clear the screen, the terminal driver may follow the clear screen code with a series of nulls to delay until the clear screen operation is finished. This should have no effect on any software using this call.

15.2 CURSOR HOME (TCRT –1,1)

This call moves the cursor to row 1, column 1, of the current screen segment, but does not otherwise affect the screen display.

15.3 CURSOR RETURN (TCRT –1,2)

This call moves the cursor to column 1 of the current row, but does not otherwise affect the screen display.

15.4 CURSOR UP (TCRT –1,3)

This call moves the cursor up one row. If the cursor is currently positioned on the top row of the current screen segment the cursor will "wrap around" and be positioned on the last row of the current segment. Otherwise this call does not affect the screen display.

15.5 CURSOR DOWN (TCRT –1,4)

This call moves the cursor down one row. If the cursor is positioned on the bottom row of the current screen segment when this call is made, it will cause the screen display to be scrolled up one line, and the cursor will remain positioned on the bottom row of the current screen segment. Otherwise this call does not affect the screen display.

If the screen scrolls then the top line is considered "lost." Although some terminals contain buffers larger than one screen and allow text scrolled off the top to be retreived, this is a terminal dependent feature and should not be relied upon.

15.6 CURSOR LEFT (TCRT –1,5)

This call moves the cursor one column to the left. If the cursor is positioned in column 1 when this call is made, the cursor will "wrap around" and be positioned in the rightmost column, one row above where it was before the call was made. If the cursor is in column 1, row 1 of the current screen segment when this call is made, the cursor will be positioned on the last column of the last row. Otherwise, this call does not affect the screen display.

15.7 CURSOR RIGHT (TCRT –1,6)

This call moves the cursor one column to the right. If the cursor is positioned in the last column of a row when this call is made, the cursor will "wrap around" and be positioned in the leftmost column, one row below where it was before the call was made. If the cursor is in the last column of the last row of the current screen segment when this call is made, the call will cause the screen to scroll up one line, and the cursor will be positioned in column 1 of the last screen line. Otherwise this call does not affect the screen display.

If the screen scrolls then the top line is considered "lost." Although some terminals contain buffers larger than one screen and allow text scrolled off the top to be retreived, this is a terminal dependent feature and should not be relied upon.

15.8 LOCK KEYBOARD (TCRT –1,7)

This call disables the keyboard, preventing the operator from sending further input to the computer. This call is normally used in conjunction with the TCRT –1,8 call, described below.

If a particular terminal does not have this function, this call will simply be ignored.

15.9 UNLOCK KEYBOARD (TCRT –1,8)

This call enables the keyboard, allowing the operator to send input to the computer. This call reverses the effect of TCRT –1,7, described above.

If a particular terminal does not have this function, this call will simply be ignored.

15.10 ERASE TO END OF LINE (TCRT –1,9)

This call causes the text to the right of the cursor to be cleared from the display. The text is cleared from the current cursor position (including the character the cursor is positioned on) to the end of the current screen line.

This function is available only if the TD$EOL bit is set in the terminal capabilities flags.

Some terminal drivers may simulate this function by overwriting the remainder of the line with spaces and repositioning the cursor. This accomplishes the same function, but is considerably slower.

15.11 ERASE TO END OF SCREEN (TCRT –1,10)

This call causes the text to the right of the cursor and on all following lines to be cleared from the display. The text is cleared from the current cursor position (including the character the cursor is positioned on) to the last row in the current screen segment.

This function is available only if the TD$EOS bit is set in the terminal capabilities flags.

Some terminal drivers may simulate this function by overwriting the remainder of the screen with spaces. This accomplishes the same function, but is considerably slower.

15.12 START REDUCED INTENSITY (TCRT –1,11)

All text written after issuing this command, up to the use of a TCRT –1,12 call (below), is displayed in reduced intensity. On terminals which support protected fields, this text will also be protected from erasure if a TCRT –1,13 call has been issued prior to this command.

Issuing this call does not alter the cursor position or occupy a screen position.

This function is available only if the TD$DIM bit is set in the terminal capabilities flags.

15.13 END REDUCED INTENSITY (TCRT –1,12)

All text written after issuing this command is displayed in normal intensity. This command is used to reverse the effects of the TCRT –1,11 call discussed above. On terminals which support protected fields, this text will be subject to erasure, even if a TCRT –1,13 call has been issued.

Issuing this call does not alter the cursor position or occupy a screen position.

This function is available only if the TD$DIM bit is set in the terminal capabilities flags.

15.14 ENABLE PROTECTED FIELDS (TCRT –1,13)

This call enables protected fields. Any text subsequently written on the screen in dim mode is protected from erasure.

Detailed information on the operation of protected fields may be found in Chapter 7 of this manual.

Protected fields may be turned off by using the TCRT –1,14 call, described below.

If a particular terminal does not support this function, this call is simply ignored.

15.15 DISABLE PROTECTED FIELDS (TCRT –1,14)

This call disables protected fields. After issuing this call, all information on the screen is subject to erasure by any one of the screen erase commands.

Detailed information on the operation of protected fields may be found in Chapter 7 of this manual.

If a particular terminal does not support this function, this call is simply ignored.

15.16 DELETE LINE (TCRT –1,15)

This call causes the line of text the cursor is currently positioned on to be deleted. All rows, starting from the row immediately below the cursor row to the last row of the current screen segment are moved up one row. A row of blanks appears as the new bottom line. The cursor position is unaffected.

This call is not available when protected fields are enabled.

This function is available only if the TD$LID bit is set in the terminal capabilities flags.

15.17 INSERT LINE (TCRT –1,16)

This call causes a blank line to be inserted at the current cursor row. It causes all rows, starting from the row the cursor is positioned on, to the last row in the current screen segment, to be moved down one row. A row of spaces is inserted at the original cursor row and the cursor is positioned at the start of that row.

This call is not available when protected fields are enabled.

This function is available only if the TD$LID bit is set in the terminal capabilities flags.

15.18 DELETE CHARACTER (TCRT –1,17)

This call deletes the character at the current cursor position. All characters, starting from the cursor position to the end of the row are moved one character position to the left. A space is placed at the last character position in the row.

If protect mode is on, this operation halts at the end of a protected field or the end of the row, whichever is encountered first.

This function is available only if the TD$CID bit is set in the terminal capabilities flags.

15.19 INSERT CHARACTER (TCRT –1,18)

This call inserts a single space into the row the cursor is positioned on. It moves all characters right one position starting from the cursor position to the end of the row. A space is placed at the cursor position. The last character on the row is lost.

If protect mode is on, this operation halts at the end of an unprotected field or the end of the row, whichever is encountered first.

This function is available only if the TD$CID bit is set in the terminal capabilities flags.

15.20 READ CURSOR ADDRESS (TCRT –1,19)

This call allows a program to query a terminal as to its current cursor position. After issuing this call the program must wait for the terminal to respond with the cursor row and column address. The address is encoded as described in Table B-1, and is followed by a carriage-return character.

If a particular terminal does not support this function, this call is simply ignored. Note that this may cause problems as the software issuing the call will be "stuck" waiting for a response to the command. A software "timeout" can be used to help alleviate this problem.

NOTE: Use of this call is not recommended due to the small number of terminals which support this function. While this command could be emulated by the terminal driver by having the driver keep track of the current cursor position, the complexity of the "rules" which any given terminal follows in positioning the cursor make this a very complex task. It is therefore likely that the effort (and code space) required to implement such an emulation make it more trouble than it is worth.

15.21 READ CHARACTER AT CURSOR POSITION (TCRT –1,20)

This call allows a program to request the character occupying the row and column where the cursor is currently located. The program issuing this call must wait for a single byte to be returned by the terminal.

If a particular terminal does not support this function, this call is simply ignored. Note that this may cause problems as the software issuing the call will be "stuck" waiting for a response to the command.

NOTE: Use of this call is not recommended due to the small number of terminals which support this function. Because emulation of this feature would require the terminal driver to keep a screen image in memory, it is not likely that this feature would ever be emulated in software.

15.22 START BLINKING TEXT (TCRT –1,21)

This call enables the blinking text attribute. As with all attribute commands, there are certain differences between field and mode oriented terminals which affect the use of this command. For further information on the differences which you must allow for, see Chapter 3.

Regardless of whether a terminal is field or mode oriented, this call will always occupy one screen position. This is done to assure text will properly line up irrespective of the terminal attribute style.

This function is available only if the TD$BLN bit is set in the terminal capabilities flags.

Because this attribute is present on the vast majority of modern terminals, many programs simply assume that it is present and do not bother to check for the terminal capabilities flag which says whether it is truly present or not. For this reason, all terminal drivers will support this call at least to the extent of outputting a space to ensure that text is properly aligned, even if the attribute is not present in any form.

15.23 END BLINKING TEXT (TCRT –1,22)

This call disables the blinking text attribute. As with all attribute commands, there are certain differences between field and mode oriented terminals which affect the use of this command. For further information on the differences which you must allow for, see Chapter 3.

Regardless of whether a terminal is field or mode oriented, this call will always occupy one screen position. This is done to assure text will properly line up irrespective of the terminal attribute style.

This function is available only if the TD$BLN bit is set in the terminal capabilities flags.

Because this attribute is present on the vast majority of modern terminals, many programs simply assume that it is present and do not bother to check for the terminal capabilities flag which says whether it is truly present or not. For this reason, all terminal drivers will support this call at least to the extent of outputting a space to ensure that text is properly aligned, even if the attribute is not present in any form.

15.24 ENABLE ALTERNATE CHARACTER SET (TCRT –1,23)

This call enables the display of the alternate character set. This call must be used prior to using the calls TCRT –1,38 through TCRT –1,53 and TCRT –1,64 through TCRT –1,78.

When the alternate character set is enabled, displayable characters may only be sent to the terminal via the TCRT calls listed above. Normal output of displayable characters will produce unpredictable results. You may, however, use other TCRT calls and output positioning codes including RETURN, LINE-FEED, and the cursor positioning call.

This call is used in conjuction with TCRT –1,24 to enable the alternate character set, display some characters, then return to the normal character set.

On terminals which do not have an alternate character set "mode," this call may perform no function, as the TCRT call which output the special characters are completely self-contained. However, to maintain compatibility with all terminals, you must always issue a TCRT –1,23 before using the special character calls, and always terminate such use with a TCRT –1,24.

15.25 DISABLE ALTERNATE CHARACTER SET (TCRT –1,24)

This call disables the display of the alternate character set which was enabled by a previous TCRT –1,23 call. Any characters displayed after this call will be displayed in the normal character set.

Like the TCRT –1,23 call, this call may perform no function on certain terminals. For compatibility, however, you must use this call after outputting special characters.

15.26 SET HORIZONTAL POSITION (TCRT –1,25)

This is an obsolete call, no longer supported or used by Alpha Micro provided software. It is only present for historical reasons.

15.27 SET VERTICAL POSITION (TCRT –1,26)

This is an obsolete call, no longer supported or used by Alpha Micro provided software. It is only present for historical reasons.

15.28 SET TERMINAL ATTRIBUTES (TCRT –1,27)

This call sets the default attributes for the top status line and the main display area of the screen. When used it is followed by two argument bytes, the first of which specifies the area for which you wish to set the attribute; and the second which specifies the attribute you wish to place there.

The first argument byte consists of a single ASCII character, either 0, 1, or 2. A 0 selects the data/text entry portion of the screen display. A 1 selects the "terminal message field" portion of the upper status line. (The "terminal message field" is where you see the "CAPS" indicator when you depress the CAPS LOCK key.) A 2 selects the "program message field" of the top status line.

The second argument is an attribute code taken from Table B-2.

This call is used to select attributes for the top status line and the text/data entry area only. To set the attribute for the bottom status line, position the cursor to the first position on the bottom status line and embed the appropriate control character to acheive the desired result, as described in the section on TCRT –1,54.

If a particular terminal does not have this function, this call and the following attribute characters will simply be ignored.

15.29 MAKE CURSOR VISIBLE (TCRT –1,28)

This call "turns on" the cursor and makes it visible once again. This call is used in conjunction with TCRT –1,29 (below) to "hide" the cursor during screen update. This produces a more pleasant screen display than one where the cursor is flashing around on the screen.

If a particular terminal does not support this function, this call is simply ignored.

15.30 MAKE CURSOR INVISIBLE (TCRT –1,29)

This call "turns off" the cursor and makes it invisible. It is used in conjunction with the TCRT –1,28 call (above) to "hide" the cursor during screen update. This produces a more pleasant screen display than one where the cursor is flashing around on the screen.

If a particular terminal does not support this function, this call is simply ignored.

15.31 START UNDERSCORED TEXT (TCRT –1,30)

This call enables the underscore text attribute. As with all attribute commands, there are certain differences between field and mode oriented terminals which affect the use of this command. For further information on the differences which you must allow for, see Chapter 3.

Regardless of whether a terminal is field or mode oriented, this call will always occupy one screen position. This is done to assure text will properly line up irrespective of the terminal attribute style.

This function is available only if the TD$UND bit is set in the terminal capabilities flags.

Because this attribute is present on the vast majority of modern terminals, many programs simply assume that it is present and do not bother to check for the terminal capabilities flag which says whether it is truly present or not. For this reason, all terminal drivers will support this call at least to the extent of outputting a space to ensure the text is properly aligned, even if the attribute is not present in any form.

15.32 END UNDERSCORED TEXT (TCRT –1,31)

This call disables the underscore text attribute. As with all attribute commands, there are certain differences between field and mode oriented terminals which affect the use of this command. For further information on the differences which you must allow for, see Chapter 3.

Regardless of whether a terminal is field or mode oriented, this call will always occupy one screen position. This is done to assure text will properly line up irrespective of the terminal attribute style.

This function is available only if the TD$UND bit is set in the terminal capabilities flags.

Because this attribute is present on the vast majority of modern terminals, many programs simply assume that it is present and do not bother to check for the terminal capabilities flag which says whether it is truly present or not. For this reason, all terminal drivers will support this call at least to the extent of outputting a space to ensure that text is properly aligned, even if the attribute is not present in any form.

15.33 START REVERSE VIDEO TEXT (TCRT –1,32)

This call enables the reverse video text attribute. As with all attribute commands, there are certain differences between field and mode oriented terminals which affect the use of this command. For further information on the differences which you must allow for, see Chapter 3.

Regardless of whether a terminal is field or mode oriented, this call will always occupy one screen position. This is done to assure text will properly line up irrespective of the terminal attribute style.

On color terminals the reverse video attribute reverses the foreground and background colors. For example, if the currently selected colors are a white foreground on a blue background, all reverse video text would be displayed as blue foreground characters on a white background.

This function is available only if the TD$RVA bit is set in the terminal capabilities flags.

Because this attribute is present on the vast majority of modern terminals, many programs simply assume that it is present and do not bother to check for the terminal capabilities flag which says whether it is truly present or not. For this reason, all terminal drivers will support this call at least to the extent of outputting a space to ensure that text is properly aligned, even if the attribute is not present in any form.

15.34 END REVERSE VIDEO TEXT (TCRT –1,33)

This call disables the reverse video text attribute. As with all attribute commands, there are certain differences between field and mode oriented terminals which affect the use of this command. For further information on the differences which you must allow for, see Chapter 3.

Regardless of whether a terminal is field or mode oriented, this call will always occupy one screen position. This is done to assure text will properly line up irrespective of the terminal attribute style.

This function is available only if the TD$RVA bit is set in the terminal capabilities flags.

Because this attribute is present on the vast majority of modern terminals, many programs simply assume that it is present and do not bother to check for the terminal capabilities flag which says whether it is truly present or not. For this reason, all terminal drivers will support this call at least to the extent of outputting a space to ensure that text is properly aligned, even if the attribute is not present in any form.

15.35 START REVERSE VIDEO, BLINKING TEXT (TCRT –1,34)

This call enables both the reverse video and the blinking text attributes. As with all attribute commands, there are certain differences between field and mode oriented terminals which affect the use of this command. For further information on the differences which you must allow for, see Chapter 3.

Regardless of whether a terminal is field or mode oriented, this call will always occupy one screen position. This is done to assure text will properly line up irrespective of the terminal attribute style.

On color terminals the reverse video attribute reverses the foreground and background colors.

For example, if the currently selected colors are a white foreground on a blue background, all reverse video text would be displayed as blue foreground characters on a white background.

This function is available only if the TD$RVA and TD$BLN bits are set in the terminal capabilities flags.

Because this attribute is present on the vast majority of modern terminals, many programs simply assume that it is present and do not bother to check for the terminal capabilities flag which says whether it is truly present or not. For this reason, all terminal drivers will support this call at least to the extent of outputting a space to ensure that text is properly aligned, even if the attribute is not present in any form.

15.36 END REVERSE VIDEO, BLINKING TEXT (TCRT –1,35)

This call disables both the reverse video and the blinking text attributes. As with all attribute commands, there are certain differences between field and mode oriented terminals which affect the use of this command. For further information on the differences which you must allow for, see Chapter 3.

Regardless of whether a terminal is field or mode oriented, this call will always occupy one screen position. This is done to assure text will properly line up irrespective of the terminal attribute style.

This function is available only if the TD$RVA and TD$BLN bits are set in the terminal capabilities flags.

Because this attribute is present on the vast majority of modern terminals, many programs simply assume that it is present and do not bother to check for the terminal capabilities flag which says whether it is truly present or not. For this reason, all terminal drivers will support this call at least to the extent of outputting a space to ensure that text is properly aligned, even if the attribute is not present in any form.

15.37 TURN OFF SCREEN DISPLAY (TCRT –1,36)

This call blanks the terminal screen until a TCRT –1,37 function is executed. This call is useful for temporarily "hiding" the screen from the operator during screen update. This can result in a more pleasing display.

If a particular terminal does not have this function, this call will simply be ignored.

15.38 TURN ON SCREEN DISPLAY (TCRT –1,37)

This call unblanks the terminal screen, reversing the effect of the TCRT –1,36 call. These two functions are useful for temporarily "hiding" the screen from the operator during screen update. This can result in a more pleasing display.

If a particular terminal does not have this function, this call will simply be ignored.

15.39∞TOP LEFT CORNER CHARACTER (TCRT –1,38)

This call is only valid when the alternate character set has been enabled (via a TCRT –1,23). The results of using this call when the normal character set is enabled are unpredictable.

If this character is not available on a particular terminal, either the closest approximation of the character or a space will be displayed. In all cases, however, this call will cause the display of some character occupying exactly one screen position.

On terminals which do not have this character available, a + character is normally substituted.

15.40∞TOP RIGHT CORNER CHARACTER (TCRT –1,39)

This call is only valid when the alternate character set has been enabled (via a TCRT –1,23). The results of using this call when the normal character set is enabled are unpredictable.

If this character is not available on a particular terminal, either the closest approximation of the character or a space will be displayed. In all cases, however, this call will cause the display of some character occupying exactly one screen position.

On terminals which do not have this character available, a + character is normally substituted.

15.41∞BOTTOM LEFT CORNER CHARACTER (TCRT –1,40)

This call is only valid when the alternate character set has been enabled (via a TCRT –1,23). The results of using this call when the normal character set is enabled are unpredictable.

If this character is not available on a particular terminal, either the closest approximation of the character or a space will be displayed. In all cases, however, this call will cause the display of some character occupying exactly one screen position.

On terminals which do not have this character available, a + character is normally substituted.

15.42∞BOTTOM RIGHT CORNER CHARACTER (TCRT –1,41)

This call is only valid when the alternate character set has been enabled (via a TCRT –1,23). The results of using this call when the normal character set is enabled are unpredictable.

If this character is not available on a particular terminal, either the closest approximation of the character or a space will be displayed. In all cases, however, this call will cause the display of some character occupying exactly one screen position.

On terminals which do not have this character available, a + character is normally substituted.

15.43 TOP INTERSECTION CHARACTER (TCRT –1,42)

This call is only valid when the alternate character set has been enabled (via a TCRT –1,23). The results of using this call when the normal character set is enabled are unpredictable.

If this character is not available on a particular terminal, either the closest approximation of the character or a space will be displayed. In all cases, however, this call will cause the display of some character occupying exactly one screen position.

On terminals which do not have this character available, a + character is normally substituted.

15.44 RIGHT INTERSECTION CHARACTER (TCRT –1,43)

This call is only valid when the alternate character set has been enabled (via a TCRT –1,23). The results of using this call when the normal character set is enabled are unpredictable.

If this character is not available on a particular terminal, either the closest approximation of the character or a space will be displayed. In all cases, however, this call will cause the display of some character occupying exactly one screen position.

On terminals which do not have this character available, a + character is normally substituted.

15.45 LEFT INTERSECTION CHARACTER (TCRT –1,44)

This call is only valid when the alternate character set has been enabled (via a TCRT –1,23). The results of using this call when the normal character set is enabled are unpredictable.

If this character is not available on a particular terminal, either the closest approximation of the character or a space will be displayed. In all cases, however, this call will cause the display of some character occupying exactly one screen position.

On terminals which do not have this character available, a + character is normally substituted.

15.46 BOTTOM INTERSECTION CHARACTER (TCRT –1,45)

This call is only valid when the alternate character set has been enabled (via a TCRT –1,23). The results of using this call when the normal character set is enabled are unpredictable.

If this character is not available on a particular terminal, either the closest approximation of the character or a space will be displayed. In all cases, however, this call will cause the display of some character occupying exactly one screen position.

On terminals which do not have this character available, a + character is normally substituted.

15.47 HORIZONTAL LINE CHARACTER (TCRT –1,46)

This call is only valid when the alternate character set has been enabled (via a TCRT –1,23). The results of using this call when the normal character set is enabled are unpredictable.

If this character is not available on a particular terminal, either the closest approximation of the character or a space will be displayed. In all cases, however, this call will cause the display of some character occupying exactly one screen position.

On terminals which do not have this character available, a - character is normally substituted.

15.48 VERTICAL LINE CHARACTER (TCRT –1,47)

This call is only valid when the alternate character set has been enabled (via a TCRT –1,23). The results of using this call when the normal character set is enabled are unpredictable.

If this character is not available on a particular terminal, either the closest approximation of the character or a space will be displayed. In all cases, however, this call will cause the display of some character occupying exactly one screen position.

On terminals which do not have this character available, a ! character is normally substituted.

15.49 CENTER INTERSECTION CHARACTER (TCRT –1,48)

This call is only valid when the alternate character set has been enabled (via a TCRT –1,23). The results of using this call when the normal character set is enabled are unpredictable.

If this character is not available on a particular terminal, either the closest approximation of the character or a space will be displayed. In all cases, however, this call will cause the display of some character occupying exactly one screen position.

On terminals which do not have this character available, a + character is normally substituted.

15.50 SOLID BLOCK CHARACTER (TCRT –1,49)

This call is only valid when the alternate character set has been enabled (via a TCRT –1,23). The results of using this call when the normal character set is enabled are unpredictable.

If this character is not available on a particular terminal, either the closest approximation of the character or a space will be displayed. In all cases, however, this call will cause the display of some character occupying exactly one screen position.

On terminals which do not have this character available, an O or # character is normally substituted.

15.51 SLANTED LINE BLOCK CHARACTER (TCRT –1,50)

This call is only valid when the alternate character set has been enabled (via a TCRT –1,23). The results of using this call when the normal character set is enabled are unpredictable.

If this character is not available on a particular terminal, either the closest approximation of the character or a space will be displayed. In all cases, however, this call will cause the display of some character occupying exactly one screen position.

On terminals which do not have this character available, an O or # character is normally substituted.

15.52 CROSS-HATCH BLOCK CHARACTER (TCRT –1,51)

This call is only valid when the alternate character set has been enabled (via a TCRT –1,23). The results of using this call when the normal character set is enabled are unpredictable.

If this character is not available on a particular terminal, either the closest approximation of the character or a space will be displayed. In all cases, however, this call will cause the display of some character occupying exactly one screen position.

On terminals which do not have this character available, an O or # character is normally substituted.

15.53 DOUBLE HORIZONTAL LINE CHARACTER (TCRT –1,52)

This call is only valid when the alternate character set has been enabled (via a TCRT –1,23). The results of using this call when the normal character set is enabled are unpredictable.

If this character is not available on a particular terminal, either the closest approximation of the character or a space will be displayed. In all cases, however, this call will cause the display of some character occupying exactly one screen position.

On terminals which do not have this character available, a = character is normally substituted.

15.54 DOUBLE VERTICAL LINE CHARACTER (TCRT –1,53)

This call is only valid when the alternate character set has been enabled (via a TCRT –1,23). The results of using this call when the normal character set is enabled are unpredictable.

If this character is not available on a particular terminal, either the closest approximation of the character or a space will be displayed. In all cases, however, this call will cause the display of some character occupying exactly one screen position.

On terminals which do not have this character available, a ! character is normally substituted.

15.55 SELECT UNSHIFTED BOTTOM STATUS LINE (TCRT –1,54)

This command selects the bottom unshifted status line as the place to receive subsequent text. After issuing this TCRT call, you must send a column position where you wish the text to begin. This column position can be selected from Table B-1. After sending the column position, the text to be displayed on the status line should then be sent.

Details on the use of status lines can be found in Chapter 8.

After issuing this command and sending the column address and the desired text, you must terminate the command by using the End Status Line functions (TCRT –1,129).

This function is available only if the TD$STS bit is set in the terminal capabilities flags.

15.56 SELECT SHIFTED BOTTOM STATUS LINE (TCRT –1,55)

This command selects the bottom shifted status line as the place to receive subsequent text. After issuing this TCRT call, you must send a column position where you wish the text to begin. This column position can be selected from Table B-1. After sending the column position, the text to be displayed on the status line should then be sent.

Details on the use of status lines can be found in Chapter 8.

After issuing this command and sending the column address and the desired text, you must terminate the command by using the End Status Line functions (TCRT –1,129).

This function is available only if the TD$STS bit is set in the terminal capabilities flags.

15.57 RESET TO DEFAULT DISPLAY FORMAT (TCRT –1,56)

This call resets the terminal to its normal power-up display configuration; in particular, it removes all split screen segments, rejoining the screen into a single segment and clears the screen.

If a particular terminal does not have this function, this call will simply be ignored.

15.58 SET HORIZONTAL SCREEN SPLIT (TCRT –1,57)

This command allows you to divide the screen into two independent areas. Each of these areas can be separately addressed, cleared, and otherwise worked with, just as it were an entire screen. You select which of these segments you wish to work with by using the TCRT –1,61 and –1,62 commands to select the active segment. The active segment behaves just a a normal screen, but does so without disturbing the other segment.

To select a horizontal split screen, determine how many rows of characters you wish to have in the upper segment. Use that number to determine the apporpriate height specification from Table B-5. Issue the TCRT –1,57 call and follow it with the character(s) from Table B-5. The screen will then clear and will be split. The upper segment (segment 0) is automatically selected.

The cursor will appear only in the active segment. However, whenever a segment is selected, the cursor will return to the position within the segment where it was last located.

The screen will stay split until a TCRT –1,56 is issued.

This function is available only if the TD$SPL bit is set in the terminal capabilities flags.

15.59 SET VERTICAL SCREEN SPLIT W/ 39 CHARACTERS (TCRT –1,58)

This command allows you to divide the screen into two independent areas. Each of these areas can be separately addressed, cleared, and otherwise worked with, just as it were an entire screen. You select which of these segments you wish to work with by using the TCRT –1,61 and –1,62 commands to select the active segment. The active segment behaves just a a normal screen, but does so without disturbing the other segment.

This command splits the screen into two vertical halves of 39 characters each. The 40th character position in each half is reserved for a dividing line, set via the TCRT –1,60 command.

The cursor will appear only in the active segment. However, whenever a segment is selected, the cursor will return to the position within the segment where it was last located.

The screen will stay split until a TCRT –1,56 is issued.

This function is available only if the TD$SPL bit is set in the terminal capabilities flags.

NOTE: Very few terminals support the use of a vertical split screen, and because of the complexity of the operation it is very unlikely that the operation will be supported via software emulation within the terminal driver. For these reasons any use of this command increases terminal dependence and is therefore not recommended.

15.60 SET VERTICAL SCREEN SPLIT W/ 40 CHARACTERS (TCRT –1,59)

This command allows you to divide the screen into two independent areas. Each of these areas can be separately addressed, cleared, and otherwise worked with, just as it were an entire screen. You select which of these segments you wish to work with by using the TCRT –1,61 and –1,62 commands to select the active segment. The active segment behaves just a a normal screen, but does so without disturbing the other segment.

This command splits the screen into two vertical halves of 40 characters each.

The cursor will appear only in the active segment. However, whenever a segment is selected, the cursor will return to the position within the segment where it was last located.

The screen will stay split until a TCRT –1,56 is issued.

This function is available only if the TD$SPL bit is set in the terminal capabilities flags.

NOTE: Very few terminals support the use of a vertical split screen, and because of the complexity of the operation it is very unlikely that the operation will be supported via software emulation within the terminal driver. For these reasons any use of this command increases terminal dependence and is therefore not recommended.

15.61 SET VERTICAL SPLIT DIVIDER CHARACTER (TCRT –1,60)

This command, when followed by a single character, places a vertical column of that character down the center of the screen (in column 40). This command is used to display a screen divider character when using vertical split screen mode.

This function is available only if the TD$SPL bit is set in the terminal capabilities flags.

NOTE: Very few terminals support the use of a vertical split screen, and because of the complexity of the operation it is very unlikely that the operation will be supported via software emulation within the terminal driver. For these reasons any use of this command increases terminal dependence and is therefore not recommended.

15.62 ACTIVATE SCREEN SEGMENT 0 (TCRT –1,61)

This command selects screen segment 0 (the upper part for horizontal split, the left part for vertical split) as the place to receive all subsequent commands and text. Segment 0 will remain the active segment until an Activate Screen Segment 1 command (TCRT –1,62) command is given.

This command is only valid after a split screen command (TCRT –1,57, –1,58, or –1,59) has been given. It is ignored if the terminal screen is not currently split.

This function is available only if the TD$SPL bit is set in the terminal capabilities flags.

15.63 ACTIVATE SCREEN SEGMENT 1 (TCRT –1,62)

This command selects screen segment 1 (the lower part for horizontal split, the right part for vertical split) as the place to receive all subsequent commands and text. Segment 1 will remain the active segment until an Activate Screen Segment 0 command (TCRT –1,61) command is given or the screen is unsplit via a TCRT –1,56 command.

This command is only valid after a split screen command (TCRT –1,57, –1,58, or –1,59) has been given. It is ignored if the terminal screen is not currently split.

This function is available only if the TD$SPL bit is set in the terminal capabilities flags.

15.64 SELECT TOP STATUS LINE (TCRT –1,63)

This command selects the top status line as the place to receive subsequent text. After issuing this TCRT call, you must send a column position where you wish the text to begin. This column position can be selected from Table B-1. After sending the column position, the text to be displayed on the status line should then be sent.

Details on the use of status lines can be found in Chapter 8.

After issuing this command, sending the column address and the desired text, you must terminate the command by using the End Status Line functions (TCRT –1,129).

This function is available only if the TD$STS bit is set in the terminal capabilities flags.

15.65 UP-ARROW CHARACTER (TCRT –1,64)

This call is only valid when the alternate character set has been enabled (via a TCRT –1,23). The results of using this call when the normal character set is enabled are unpredictable.

If this character is not available on a particular terminal, either the closest approximation of the character or a space will be displayed. In all cases, however, this call will cause the display of some character occupying exactly one screen position.

On terminals which do not have this character available, a space is normally substituted.

15.66 DOWN-ARROW CHARACTER (TCRT –1,65)

This call is only valid when the alternate character set has been enabled (via a TCRT –1,23). The results of using this call when the normal character set is enabled are unpredictable.

If this character is not available on a particular terminal, either the closest approximation of the character or a space will be displayed. In all cases, however, this call will cause the display of some character occupying exactly one screen position.

On terminals which do not have this character available, a space is normally substituted.

15.67 RAISED DOT CHARACTER (TCRT –1,66)

This call is only valid when the alternate character set has been enabled (via a TCRT –1,23). The results of using this call when the normal character set is enabled are unpredictable.

If this character is not available on a particular terminal, either the closest approximation of the character or a space will be displayed. In all cases, however, this call will cause the display of some character occupying exactly one screen position.

On terminals which do not have this character available, a space is normally substituted.

15.68 END OF LINE CHARACTER (TCRT –1,67)

This call is only valid when the alternate character set has been enabled (via a TCRT –1,23). The results of using this call when the normal character set is enabled are unpredictable.

If this character is not available on a particular terminal, either the closest approximation of the character or a space will be displayed. In all cases, however, this call will cause the display of some character occupying exactly one screen position.

On terminals which do not have this character available, a ~ (tilde) character is normally substituted.

15.69 HORIZONTAL TAB CHARACTER (TCRT –1,68)

This call is only valid when the alternate character set has been enabled (via a TCRT –1,23). The results of using this call when the normal character set is enabled are unpredictable.

If this character is not available on a particular terminal, either the closest approximation of the character or a space will be displayed. In all cases, however, this call will cause the display of some character occupying exactly one screen position.

On terminals which do not have this character available, a | (vertical bar) character is normally substituted.

15.70 PARAGRAPH CHARACTER (TCRT –1,69)

This call is only valid when the alternate character set has been enabled (via a TCRT –1,23). The results of using this call when the normal character set is enabled are unpredictable.

If this character is not available on a particular terminal, either the closest approximation of the character or a space will be displayed. In all cases, however, this call will cause the display of some character occupying exactly one screen position.

On terminals which do not have this character available, a space is normally substituted.

15.71 DAGGER CHARACTER (TCRT –1,70)

This call is only valid when the alternate character set has been enabled (via a TCRT –1,23). The results of using this call when the normal character set is enabled are unpredictable.

If this character is not available on a particular terminal, either the closest approximation of the character or a space will be displayed. In all cases, however, this call will cause the display of some character occupying exactly one screen position.

On terminals which do not have this character available, a space is normally substituted.

15.72 SECTION CHARACTER (TCRT –1,71)

This call is only valid when the alternate character set has been enabled (via a TCRT –1,23). The results of using this call when the normal character set is enabled are unpredictable.

If this character is not available on a particular terminal, either the closest approximation of the character or a space will be displayed. In all cases, however, this call will cause the display of some character occupying exactly one screen position.

On terminals which do not have this character available, a space is normally substituted.

15.73 CENT SIGN CHARACTER (TCRT –1,72)

This call is only valid when the alternate character set has been enabled (via a TCRT –1,23). The results of using this call when the normal character set is enabled are unpredictable.

If this character is not available on a particular terminal, either the closest approximation of the character or a space will be displayed. In all cases, however, this call will cause the display of some character occupying exactly one screen position.

On terminals which do not have this character available, a space is normally substituted.

15.74 ONE-QUARTER CHARACTER (TCRT –1,73)

This call is only valid when the alternate character set has been enabled (via a TCRT –1,23). The results of using this call when the normal character set is enabled are unpredictable.

If this character is not available on a particular terminal, either the closest approximation of the character or a space will be displayed. In all cases, however, this call will cause the display of some character occupying exactly one screen position.

On terminals which do not have this character available, a space is normally substituted.

15.75 ONE-HALF CHARACTER (TCRT –1,74)

This call is only valid when the alternate character set has been enabled (via a TCRT –1,23). The results of using this call when the normal character set is enabled are unpredictable.

If this character is not available on a particular terminal, either the closest approximation of the character or a space will be displayed. In all cases, however, this call will cause the display of some character occupying exactly one screen position.

On terminals which do not have this character available, a space is normally substituted.

15.76 DEGREE CHARACTER (TCRT –1,75)

This call is only valid when the alternate character set has been enabled (via a TCRT –1,23). The results of using this call when the normal character set is enabled are unpredictable.

If this character is not available on a particular terminal, either the closest approximation of the character or a space will be displayed. In all cases, however, this call will cause the display of some character occupying exactly one screen position.

On terminals which do not have this character available, a space is normally substituted.

15.77 TRADEMARK CHARACTER (TCRT –1,76)

This call is only valid when the alternate character set has been enabled (via a TCRT –1,23). The results of using this call when the normal character set is enabled are unpredictable.

If this character is not available on a particular terminal, either the closest approximation of the character or a space will be displayed. In all cases, however, this call will cause the display of some character occupying exactly one screen position.

On terminals which do not have this character available, a space is normally substituted.

15.78 COPYRIGHT CHARACTER (TCRT –1,77)

This call is only valid when the alternate character set has been enabled (via a TCRT –1,23). The results of using this call when the normal character set is enabled are unpredictable.

If this character is not available on a particular terminal, either the closest approximation of the character or a space will be displayed. In all cases, however, this call will cause the display of some character occupying exactly one screen position.

On terminals which do not have this character available, a space is normally substituted.

15.79 REGISTERED CHARACTER (TCRT –1,78)

This call is only valid when the alternate character set has been enabled (via a TCRT –1,23). The results of using this call when the normal character set is enabled are unpredictable.

If this character is not available on a particular terminal, either the closest approximation of the character or a space will be displayed. In all cases, however, this call will cause the display of some character occupying exactly one screen position.

On terminals which do not have this character available, a space is normally substituted.

15.80 PRINT SCREEN (TCRT –1,79)

This call causes the contents of the current screen display to be sent to the local printer port, if the terminal is so equipped. Note that the exact results of this operation will depend not only on the terminal being used, but also the printer that is attached to it. This is especially true in the area of characters in the alternate character set that may be displayed on the terminal screen, as not all printers have the capability of printing these characters.

This function is available only if the TD$PRT bit is set in the terminal capabilities flags.

15.81 SET TO 132 COLUMN MODE (TCRT –1,80)

This call selects the 132 column display mode on the terminal. If the terminal is already in 132 column mode, no operation is performed.

Programs using this call must assume that the screen display is cleared when this call is issued, even though not all terminals will clear the screen. To reduce the possiblity of an improper screen display appearing, the entire screen should be cleared prior to issuing this call.

In addition to changing the screen format this call also changes the value which will be returned for "number of columns" when a TRMCHR call is issued.

This function is available only if the TD$132 bit is set in the terminal capabilities flags.

15.82 SET TO 80 COLUMN MODE (TCRT –1,81)

This call selects the 80 column display mode on the terminal. If the terminal is already in 80 column mode, no operation is performed.

Programs using this call must assume that the screen display is cleared when this call is issued, even though not all terminals will clear the screen. To reduce the possiblity of an improper screen display appearing, the entire screen should be cleared prior to issuing this call.

In addition to changing the screen format this call also changes the value which will be returned for "number of columns" when a TRMCHR call is issued.

This function is available only if the TD$132 bit is set in the terminal capabilities flags.

15.83 ENTER TRANSPARENT PRINT MODE (TCRT –1,82)

This call enables the transparent print function on those terminals which have local printer support. Once this call has been issued, all further data sent to the terminal will be routed to the printer port and will be ignored by the terminal display.

When the terminal has been placed in transparent print mode the only command it will then recognize is the command to exit the transparent print mode (TCRT –1,83).

This function is available only if the TD$PRT bit is set in the terminal capabilities flags.

15.84 EXIT TRANSPARENT PRINT MODE (TCRT –1,83)

This call disables the transparent print function that was enabled by the TCRT –1,82 call. After issuing this call, all data sent to the terminal will once again be interpreted as commands or data to be displayed on the screen, as appropriate.

This function is available only if the TD$PRT bit is set in the terminal capabilities flags.

15.85 BEGIN WRITING TO ALTERNATE PAGE (TCRT –1,84)

This call causes all further data and commands to affect only the alternate page of memory in the terminal, not the current display. The data and commands you issue will not be visble until you select the alternate page of memory for display via a TCRT –1,86 call. Data and commands will continue to be sent to the alternate page until you issue a TCRT –1,85 call to resume writing to the currently displayed page.

This command is often used for maintaining two separate screen displays with the ability to very quickly alternate between the displays. A typical application would be the display of an application menu and explanatory text with the operator able to quickly flip from one to the other.

This function is available only if the TD$ALP bit is set in the terminal capabilities flags.

15.86 END WRITING TO ALTERNATE PAGE (TCRT –1,85)

This call cancels the effect of the TCRT –1,84 call and resumes the direction of all commands and data to the current screen display.

This function is available only if the TD$ALP bit is set in the terminal capabilities flags.

15.87 TOGGLE DISPLAYED PAGE (TCRT –1,86)

This call selects the alternate screen page for display, and causes the page being displayed at the time of the call to become the alternate page. In this manner, successive use of this call will cause the screen to alternate between the two pages of screen display.

This function is available only if the TD$ALP bit is set in the terminal capabilities flags.

15.88 COPY TO ALTERNATE PAGE (TCRT –1,87)

This call copies the contents of the currently displayed screen page to the alternate screen page. This is useful when you wish to make minor modifications to the current screen image, but wish to be able to present the results all at once via a Toggle Displayed Page command (TCRT –1,86).

This command can be used to effect limited animation on the terminal screen.

This function is available only if the TD$ALP bit is set in the terminal capabilities flags.

15.89 INSERT COLUMN (TCRT –1,88)

This call inserts a single space into all rows in the current screen segment. It moves all characters right one position starting from the cursor position to the end of the row. A space is placed in the column where the cursor is located. The last character on each row is lost.

This function is available only if the TD$KID bit is set in the terminal capabilities flags.

15.90 DELETE COLUMN (TCRT –1,89)

This call deletes a character from all rows in the current screen segment. All characters, starting from the current cursor column to the end of the row are moved one character position to the left. A space is placed at the last character position each of the rows.

This function is available only if the TD$KID bit is set in the terminal capabilities flags.

15.91 BLOCK FILL WITH ATTRIBUTE (TCRT –1,90)

This command allows you to "fill" a rectangular area of the screen with a specific attribute. This is useful for establishing a particular attribute for a section of the screen.

To fill an area of the screen, first position the cursor at the upper left corner of the area to be filled. Issue a TCRT –1,90 call followed by height and width codes taken from Table B-5, defining the lower right corner of the area. Next output the coded character corresponding to the attribute you wish to fill with. The screen display will then be updated.

All validation of the size of the block to be filled is left to the application program. Using this command with a block specified such that it will extend beyond the visible screen boundaries will have unpredictable results.

This function is available only if the TD$BLF bit is set in the terminal capabilities flags.

Because this command depends both on the availability of block fill commands in the terminal and on the terminal being a "field" oriented terminal, care should be taken in its use to avoid restricting the use of the software to a small number of terminals.

15.92 BLOCK FILL WITH CHARACTER (TCRT –1,91)

This command allows you to "fill" a rectangular area of the screen with a specific character. This is useful for clearing an area of the screen (by block filling with space) or to establish a background texture (by block filling with some other character).

To fill an area of the screen, first position the cursor at the upper left corner of the area to be filled. Issue a TCRT –1,91 call followed by height and width codes taken from Table B-5, defining the lower right corner of the area. Next output the single character you wish to have the area filled with. The screen display will then be updated.

All validation of the size of the block to be filled is left to the application program. Using this command with a block specified such that it will extend beyond the visible screen boundaries will have unpredictable results.

This function is available only if the TD$BLF bit is set in the terminal capabilities flags.

15.93 DRAW A BOX (TCRT –1,92)

This command allows you to draw a "box" on the screen. This box encloses a rectangular area of the screen with line drawing graphics characters. Note that only the outside lines of the box are drawn on the screen; characters inside and outside the box are unaffected by this command. If you wish to clear the inside of the box, the use of the Block Fill with Character command (TCRT –1,91) is recommended.

To draw a box, first position the cursor at the upper left corner of the box. Issue a TCRT –1,92 call followed by height and width codes taken from Table B-5, which define the lower right corner of the area. The screen display will then be updated.

All validation of the size of the box to be drawn is left to the application program. Using this command with a box specified such that it will extend beyond the visible screen boundaries will have unpredictable results.

This function is available only if the TD$BOX bit is set in the terminal capabilities flags.

15.94∞SCROLL BOX UP ONE LINE (TCRT –1,93)

This command allows you to "scroll" a rectangular area of the screen by copying that area up one line. This is useful when using a rectangular area of the screen as a "window" onto text.

To scroll an area of the screen up one line, first position the cursor at the upper left corner of the area to be scrolled. Issue a TCRT –1,93 call followed by height and width codes taken from Table B-5, defining the lower right corner of the area to be scrolled. The text within the rectangle will now be moved up one line. Note that the "last" line in the rectangle has been duplicated by the copy operation. If you wish the last line to be cleared, your program must perform that operation itself.

All validation of the size of the box to be scrolled is left to the application program. Using this command with a box specified such that it will extend beyond the visible screen boundaries will have unpredictable results.

This function is available only if the TD$BOX bit is set in the terminal capabilities flags. It has no other relationship to the draw box command: areas can be scrolled whether surrounded by a box or not.

15.95∞SCROLL BOX DOWN ONE LINE (TCRT –1,94)

This command allows you to "scroll" a rectangular area of the screen by copying that area down one line. This is useful when using a rectangular area of the screen as a "window" onto text.

To scroll an area of the screen down one line, first position the cursor at the lower left corner of the area to be scrolled. Issue a TCRT –1,94 call followed by height and width codes taken from Table B-5, defining the lower right corner of the area to be scrolled. The text within the rectangle will now be moved down one line. Note that the "first" line in the rectangle has been duplicated by the copy operation. If you wish the first line to be cleared, your program must perform that operation itself.

All validation of the size of the box to be scrolled is left to the application program. Using this command with a box specified such that it will extend beyond the visible screen boundaries will have unpredicatable results.

This function is available only if the TD$BOX bit is set in the terminal capabilities flags. It has no other relationship to the draw box command: areas can be scrolled whether surrounded by a box or not.

15.96∞SELECT JUMP SCROLL (TCRT –1,95)

This call selects the "jump" method of scrolling text off of the screen. In this mode each line simply "disappears" off the top of the screen, to be replaced with the new line. It is the opposite of "smooth" scrolling where the text of the line is rolled off the top of the screen one character dot at a time.

Smooth scrolling is generally considered to be more visually pleasing than jump scrolling, but restricts the rate at which information can be scrolled on the screen. Therefore it is usually used under special circumstances rather than as the normal mode of operation.

Calls are provided (TCRT –1,96 through TCRT –1,99) for up to four different smooth scrolling rates.

This function is available only if the TD$SMT bit is set in the terminal capabilities flags.

If a particular terminal does not have this function, this call will simply be ignored.

15.97 SELECT FAST SMOOTH SCROLL (TCRT –1,96)

This call selects the fastest of the available smooth scrolling modes.

See the information on TCRT –1,95 for more information on smooth scrolling.

This function is available only if the TD$SMT bit is set in the terminal capabilities flags.

If a particular terminal does not have different settings for smooth scroll, but instead offers only the choice between jump scroll and one speed of smooth scrolling, this call will select smooth scrolling.

15.98 SELECT MEDIUM-FAST SMOOTH SCROLL (TCRT –1,97)

This call selects the second fastest of the available smooth scrolling modes.

See the information on TCRT –1,95 for more information on smooth scrolling.

This function is available only if the TD$SMT bit is set in the terminal capabilities flags.

If a particular terminal does not have different settings for smooth scroll, but instead offers only the choice between jump scroll and one speed of smooth scrolling, this call will select smooth scrolling.

15.99 SELECT MEDIUM-SLOW SMOOTH SCROLL (TCRT –1,98)

This call selects the next-to-slowest of the available smooth scrolling modes. See the information on TCRT –1,95 for more information on smooth scrolling.

This function is available only if the TD$SMT bit is set in the terminal capabilities flags.

If a particular terminal does not have different settings for smooth scroll, but instead offers only the choice between jump scroll and one speed of smooth scrolling, this call will select smooth scrolling.

15.100 SELECT SLOW SMOOTH SCROLL (TCRT –1,99)

This call selects the slowest of the available smooth scrolling modes.

See the information on TCRT –1,95 for more information on smooth scrolling.

This function is available only if the TD$SMT bit is set in the terminal capabilities flags.

If a particular terminal does not have different settings for smooth scroll, but instead offers only the choice between jump scroll and one speed of smooth scrolling, this call will select smooth scrolling.

15.101 START UNDERSCORED AND BLINKING TEXT (TCRT –1,100)

This call enables both the underscore and the blinking text attributes. As with all attribute commands, there are certain differences between field and mode oriented terminals which affect the use of this command. For further information on the differences which you must allow for, see Chapter 3.

Regardless of whether a terminal is field or mode oriented, this call will always occupy one screen position. This is done to assure text will properly line up irrespective of the terminal attribute style.

This function is available only if the TD$UND and TD$BLN bits are set in the terminal capabilities flags.

Because this attribute is present on the vast majority of modern terminals, many programs simply assume that it is present and do not bother to check for the terminal capabilities flag which says whether it is truly present or not. For this reason, all terminal drivers will support this call at least to the extent of outputting a space to ensure that text is properly aligned, even if the attribute is not present in any form.

15.102 END UNDERSCORED AND BLINKING TEXT (TCRT –1,101)

This call disables both the underscore and the blinking text attributes. As with all attribute commands, there are certain differences between field and mode oriented terminals which affect the use of this command. For further information on the differences which you must allow for, see Chapter 3.

Regardless of whether a terminal is field or mode oriented, this call will always occupy one screen position. This is done to assure text will properly line up irrespective of the terminal attribute style.

This function is available only if the TD$UND and TD$BLN bits are set in the terminal capabilities flags.

Because this attribute is present on the vast majority of modern terminals, many programs simply assume that it is present and do not bother to check for the terminal capabilities flag which says whether it is truly present or not. For this reason, all terminal drivers will support this call at least to the extent of outputting a space to ensure that text is properly aligned, even if the attribute is not present in any form.

15.103 START UNDERSCORED AND REVERSE VIDEO TEXT (TCRT –1,102)

This call enables both the underscore and the reverse video text attributes. As with all attribute commands, there are certain differences between field and mode oriented terminals which affect the use of this command. For further information on the differences which you must allow for, see Chapter 3.

Regardless of whether a terminal is field or mode oriented, this call will always occupy one screen position. This is done to assure text will properly line up irrespective of the terminal attribute style.

On color terminals the reverse video attribute reverses the foreground and background colors. For example, if the currently selected colors are a white foreground on a blue background, all reverse video text would be displayed as blue foreground characters on a white background.

This function is available only if the TD$UND and TD$RVA bits are set in the terminal capabilities flags.

Because this attribute is present on the vast majority of modern terminals, many programs simply assume that it is present and do not bother to check for the terminal capabilities flag which says whether it is truly present or not. For this reason, all terminal drivers will support this call at least to the extent of outputting a space to ensure that text is properly aligned, even if the attribute is not present in any form.

15.104 END UNDERSCORED AND REVERSE VIDEO TEXT (TCRT –1,103)

This call disables both the underscore and the reverse video text attributes. As with all attribute commands, there are certain differences between field and mode oriented terminals which affect the use of this command. For further information on the differences which you must allow for, see Chapter 3.

Regardless of whether a terminal is field or mode oriented, this call will always occupy one screen position. This is done to assure text will properly line up irrespective of the terminal attribute style.

This function is available only if the TD$UND and TD$RVA bits are set in the terminal capabilities flags.

Because this attribute is present on the vast majority of modern terminals, many programs simply assume that it is present and do not bother to check for the terminal capabilities flag which says whether it is truly present or not. For this reason, all terminal drivers will support this call at least to the extent of outputting a space to ensure that text is properly aligned, even if the attribute is not present in any form.

15.105 START UNDERSCORED, REVERSE VIDEO, BLINKING TEXT (TCRT –1,104)

This call enables the underscore, reverse video and blinking text attributes. As with all attribute commands, there are certain differences between field and mode oriented terminals which affect the use of this command. For further information on the differences which you must allow for, see Chapter 3.

Regardless of whether a terminal is field or mode oriented, this call will always occupy one screen position. This is done to assure text will properly line up irrespective of the terminal attribute style.

On color terminals the reverse video attribute reverses the foreground and background colors. For example, if the currently selected colors are a white foreground on a blue background, all reverse video text would be displayed as blue foreground characters on a white background.

This function is available only if the TD$UND, TD$RVA and TD$BLN bits are set in the terminal capabilities flags.

Because this attribute is present on the vast majority of modern terminals, many programs simply assume that it is present and do not bother to check for the terminal capabilities flag which says whether it is truly present or not. For this reason, all terminal drivers will support this call at least to the extent of outputting a space to ensure that text is properly aligned, even if the attribute is not present in any form.

15.106 END UNDERSCORED, REVERSE VIDEO, BLINKING TEXT (TCRT –1,105)

This call disables the underscore, reverse video and blinking text attributes. As with all attribute commands, there are certain differences between field and mode oriented terminals which affect the use of this command. For further information on the differences which you must allow for, see Chapter 3.

Regardless of whether a terminal is field or mode oriented, this call will always occupy one screen position. This is done to assure text will properly line up irrespective of the terminal attribute style.

This function is available only if the TD$UND, TD$RVA and TD$BLN bits are set in the terminal capabilities flags.

Because this attribute is present on the vast majority of modern terminals, many programs simply assume that it is present and do not bother to check for the terminal capabilities flag which says whether it is truly present or not. For this reason, all terminal drivers will support this call at least to the extent of outputting a space to ensure that text is properly aligned, even if the attribute is not present in any form.

15.107 START UNDERSCORED TEXT W/O A SPACE (TCRT –1,106)

This call enables the underscore text attribute without occupying a screen position. As with all attribute commands, there are certain differences between field and mode oriented terminals which affect the use of this command. For further information on the differences which you must allow for, see Chapter 3.

This call is similar to the TCRT –1,30 function but does not occupy a screen position.

This function is available only if the TD$UND and TD$NSP bits are set in the terminal capabilities flags.

Because this command cannot be supported by many terminals presently installed, programs should avoid using this call except where absolutely necessary. In situations where the characteristic of not occupying a screen position is not vital, the corresponding attribute call which does occupy a space should be used, since that command is present in virtually all modern terminals.

15.108 END UNDERSCORED TEXT W/O A SPACE (TCRT –1,107)

This call disables the underscore text attribute without occupying a screen position. As with all attribute commands, there are certain differences between field and mode oriented terminals which affect the use of this command. For further information on the differences which you must allow for, see Chapter 3.

This call is similar to the TCRT –1,31 command, but does not occupy a screen position.

This function is available only if the TD$UND and TD$NSP bits are set in the terminal capabilities flags.

Because this command cannot be supported by many terminals presently installed, programs should avoid using this call except where absolutely necessary. In situations where the characteristic of not occupying a screen position is not vital, the corresponding attribute call which does occupy a space should be used, since that command is present in virtually all modern terminals.

15.109 START REVERSE VIDEO TEXT W/O A SPACE (TCRT –1,108)

This call enables the reverse video text attribute without occupying a screen position. As with all attribute commands, there are certain differences between field and mode oriented terminals which affect the use of this command. For further information on the differences which you must allow for, see Chapter 3.

This call is similar to the TCRT –1,32 function but does not occupy a screen position.

This function is available only if the TD$RVA and TD$NSP bits are set in the terminal capabilities flags.

Because this command cannot be supported by many terminals presently installed, programs should avoid using this call except where absolutely necessary. In situations where the characteristic of not occupying a screen position is not vital, the corresponding attribute call which does occupy a space should be used, since that command is present in virtually all modern terminals.

15.110 END REVERSE VIDEO TEXT W/O A SPACE (TCRT –1,109)

This call disables the reverse video text attribute without occupying a screen position. As with all attribute commands, there are certain differences between field and mode oriented terminals which affect the use of this command. For further information on the differences which you must allow for, see Chapter 3.

This call is similar to the TCRT –1,33 command, but does not occupy a screen position.

This function is available only if the TD$RVA and TD$NSP bits are set in the terminal capabilities flags.

Because this command cannot be supported by many terminals presently installed, programs should avoid using this call except where absolutely necessary. In situations where the characteristic of not occupying a screen position is not vital, the corresponding attribute call which does occupy a space should be used, since that command is present in virtually all modern terminals.

15.111∞START REVERSE VIDEO, BLINKING TEXT W/O A SPACE (TCRT –1,110)

This call enables the reverse video and blinking text attributes without occupying a screen position. As with all attribute commands, there are certain differences between field and mode oriented terminals which affect the use of this command. For further information on the differences which you must allow for, see Chapter 3.

This call is similar to the TCRT –1,34 function but does not occupy a screen position.

This function is available only if the TD$RVA, TD$BLK and TD$NSP bits are set in the terminal capabilities flags.

Because this command cannot be supported by many terminals presently installed, programs should avoid using this call except where absolutely necessary. In situations where the characteristic of not occupying a screen position is not vital, the corresponding attribute call which does occupy a space should be used, since that command is present in virtually all modern terminals.

15.112∞END REVERSE VIDEO, BLINKING TEXT W/O A SPACE (TCRT –1,111)

This call disables the reverse video and blinking text attributes without occupying a screen position. As with all attribute commands, there are certain differences between field and mode oriented terminals which affect the use of this command. For further information on the differences which you must allow for, see Chapter 3.

This call is similar to the TCRT –1,35 command, but does not occupy a screen position.

This function is available only if the TD$RVA, TD$BLK and TD$NSP bits are set in the terminal capabilities flags.

Because this command cannot be supported by many terminals presently installed, programs should avoid using this call except where absolutely necessary. In situations where the characteristic of not occupying a screen position is not vital, the corresponding attribute call which does occupy a space should be used, since that command is present in virtually all modern terminals.

15.113∞START UNDERSCORED, BLINKING TEXT W/O A SPACE (TCRT –1,112)

This call enables the underscore and blinking text attributes without occupying a screen position. As with all attribute commands, there are certain differences between field and mode oriented terminals which affect the use of this command. For further information on the differences which you must allow for, see Chapter 3.

This call is similar to the TCRT –1,100 function but does not occupy a screen position.

This function is available only if the TD$UND, TD$BLK and TD$NSP bits are set in the terminal capabilities flags.

Because this command cannot be supported by many terminals presently installed, programs should avoid using this call except where absolutely necessary. In situations where the characteristic of not occupying a screen position is not vital, the corresponding attribute call which does occupy a space should be used, since that command is present in virtually all modern terminals.

15.114∞END UNDERSCORED, BLINKING TEXT W/O A SPACE (TCRT –1,113)

This call disables the underscore and blinking attributes without occupying a screen position. As with all attribute commands, there are certain differences between field and mode oriented terminals which affect the use of this command. For further information on the differences which you must allow for, see Chapter 3.

This call is similar to the TCRT –1,101 command, but does not occupy a screen position.

This function is available only if the TD$UND, TD$BLK and TD$NSP bits are set in the terminal capabilities flags.

Because this command cannot be supported by many terminals presently installed, programs should avoid using this call except where absolutely necessary. In situations where the characteristic of not occupying a screen position is not vital, the corresponding attribute call which does occupy a space should be used, since that command is present in virtually all modern terminals.

15.115∞START UNDERSCORED, REVERSE VIDEO TEXT W/O A SPACE (TCRT –1,114)

This call enables the underscore text and reverse video attributes without occupying a screen position. As with all attribute commands, there are certain differences between field and mode oriented terminals which affect the use of this command. For further information on the differences which you must allow for, see Chapter 3.

This call is similar to the TCRT –1,102 function but does not occupy a screen position.

This function is available only if the TD$UND, TD$RVA and TD$NSP bits are set in the terminal capabilities flags.

Because this command cannot be supported by many terminals presently installed, programs should avoid using this call except where absolutely necessary. In situations where the characteristic of not occupying a screen position is not vital, the corresponding attribute call which does occupy a space should be used, since that command is present in virtually all

modern terminals.

15.116 END UNDERSCORED, REVERSE VIDEO TEXT W/O A SPACE (TCRT –1,115)

This call disables the underscore and reverse video text attributes without occupying a screen position. As with all attribute commands, there are certain differences between field and mode oriented terminals which affect the use of this command. For further information on the differences which you must allow for, see Chapter 3.

This call is similar to the TCRT –1,103 command, but does not occupy a screen position.

This function is available only if the TD$UND, TD$RVA and TD$NSP bits are set in the terminal capabilities flags.

Because this command cannot be supported by many terminals presently installed, programs should avoid using this call except where absolutely necessary. In situations where the characteristic of not occupying a screen position is not vital, the corresponding attribute call which does occupy a space should be used, since that command is present in virtually all modern terminals.

15.117 START UNDERSCORED, REVERSE, BLINKING TEXT W/O A SPACE (TCRT –1,116)

This call enables the underscore, reverse video and blinking text attributes without occupying a screen position. As with all attribute commands, there are certain differences between field and mode oriented terminals which affect the use of this command. For further information on the differences which you must allow for, see Chapter 3.

This call is similar to the TCRT –1,104 function but does not occupy a screen position.

This function is available only if the TD$UND, TD$RVA, TD$BLK and TD$NSP bits are set in the terminal capabilities flags.

Because this command cannot be supported by many terminals presently installed, programs should avoid using this call except where absolutely necessary. In situations where the characteristic of not occupying a screen position is not vital, the corresponding attribute call which does occupy a space should be used, since that command is present in virtually all modern terminals.

15.118 END UNDERSCORED, REVERSE, BLINKING TEXT W/O A SPACE (TCRT –1,117)

This call disables the underscore, reverse video, and blinking text attributes without occupying a screen position. As with all attribute commands, there are certain differences between field and mode oriented terminals which affect the use of this command. For further information on the differences which you must allow for, see Chapter 3.

This call is similar to the TCRT –1,105 command, but does not occupy a screen position.

This function is available only if the TD$UND, TD$RVA, TD$BLK and TD$NSP bits are set in the terminal capabilities flags.

Because this command cannot be supported by many terminals presently installed, programs should avoid using this call except where absolutely necessary. In situations where the characteristic of not occupying a screen position is not vital, the corresponding attribute call which does occupy a space should be used, since that command is present in virtually all modern terminals.

15.119∞START BLINKING TEXT W/O A SPACE (TCRT –1,118)

This call enables the blinking text attribute without occupying a screen position. As with all attribute commands, there are certain differences between field and mode oriented terminals which affect the use of this command. For further information on the differences which you must allow for, see Chapter 3.

This call is similar to the TCRT –1,21 function but does not occupy a screen position.

This function is available only if the TD$BLK and TD$NSP bits are set in the terminal capabilities flags.

Because this command cannot be supported by many terminals presently installed, programs should avoid using this call except where absolutely necessary. In situations where the characteristic of not occupying a screen position is not vital, the corresponding attribute call which does occupy a space should be used, since that command is present in virtually all modern terminals.

15.120∞END BLINKING TEXT W/O A SPACE (TCRT –1,119)

This call disables the blinking text attribute without occupying a screen position. As with all attribute commands, there are certain differences between field and mode oriented terminals which affect the use of this command. For further information on the differences which you must allow for, see Chapter 3.

This call is similar to the TCRT –1,22 command, but does not occupy a screen position.

This function is available only if the TD$BLK and TD$NSP bits are set in the terminal capabilities flags.

Because this command cannot be supported by many terminals presently installed, programs should avoid using this call except where absolutely necessary. In situations where the characteristic of not occupying a screen position is not vital, the corresponding attribute call which does occupy a space should be used, since that command is present in virtually all modern terminals.

15.121∞SET CURSOR TO BLINKING BLOCK (TCRT –1,120)

This command sets the text cursor "shape" to be that of a blinking block.

On terminals which do not support changing the cursor shape, this command will be ignored.

15.122 SET CURSOR TO STEADY BLOCK (TCRT –1,121)

This command sets the text cursor "shape" to be that of a steady (non-blinking) block.

On terminals which do not support changing the cursor shape, this command will be ignored.

15.123 SET CURSOR TO BLINKING UNDERLINE (TCRT –1,122)

This command sets the text cursor "shape" to be that of a blinking underline.

On terminals which do not support changing the cursor shape, this command will be ignored.

15.124 SET CURSOR TO STEADY UNDERLINE (TCRT –1,123)

This command sets the text cursor "shape" to be that of a steady (non-blinking) underline.

On terminals which do not support changing the cursor shape, this command will be ignored.

15.125 RESERVED FUNCTIONS (TCRT –1,124 THROUGH TCRT –1,127)

These functions are reserved for future expansion by Alpha Micro. If you have a specific need for a new TCRT function, you may contact Alpha Micro to reserve a TCRT call number. The reservation process avoids problems arising from incompatible extensions to the TCRT functions

15.126 SELECT TOP STATUS LINE W/O ADDRESS (TCRT –1,128)

This function allows you to send text to the top status line. Immediately after issuing this command, send the text you wish displayed on the status line, followed by an End Status Line (TCRT –1,129) command.

Details on the use of status lines can be found in Chapter 8.

This command is similar to the TCRT –1,63 command, but only allows writing to the entire status line, without individually addressable columns. This makes this command less efficient than the TCRT –1,63 command, but is supported by a wider variety of terminals.

The size of the status line referenced by this command may be found via the TRMCHR monitor call.

On terminals with only a single status line, this function will address that status line.

This function is available only if the TD$STS bit is set in the terminal capabilities flags.

15.127 END STATUS LINE (TCRT –1,129)

This command is used to terminate any of the status line functions. It must be used after sending the desired text to the status line to return the terminal to its normal text area display.

Details on the use of status lines can be found in Chapter 8.

This function is available only if the TD$STS bit is set in the terminal capabilities flags.

15.128 SELECT UNSHIFTED STATUS LINE W/O ADDRESS (TCRT –1,130)

This function allows you to send text to the unshifted bottom status line. Immediately after issuing this command, send the text you wish displayed on the status line, followed by an End Status Line (TCRT –1,129) command.

Details on the use of status lines can be found in Chapter 8.

This command is similar to the TCRT –1,54 command, but only allows writing to the entire status line, without individually addressable columns. This makes this command less efficient than the TCRT –1,54 command, but it is supported by a wider variety of terminals.

The size of the status line referenced by this command may be found via the TRMCHR monitor call.

This function is available only if the TD$STS bit is set in the terminal capabilities flags.

15.129 SELECT SHIFTED STATUS LINE W/O ADDRESS (TCRT –1,131)

This function allows you to send text to the shifted bottom status line. Immediately after issuing this command, send the text you wish displayed on the status line, followed by an End Status Line (TCRT –1,129) command.

Details on the use of status lines can be found in Chapter 8.

This command is similar to the TCRT –1,55 command, but only allows writing to the entire status line, without individually addressable columns. This makes this command less efficient than the TCRT –1,55 command, but is supported by a wider variety of terminals.

The size of the status line referenced by this command may be found via the TRMCHR monitor call.

This function is available only if the TD$STS bit is set in the terminal capabilities flags.

15.130 SELECT BLACK TEXT (TCRT –1,132)

This command selects the current text color to be black on the standard background. This function occupies one screen position and is therefore considered inferior to the general color selection commands TCRT –2,n and TCRT –3,n. It is provided for compatibility with certain terminals, such as the AM-70, only and is not recommended for use in new software development.

If the selected background color is black, as is normal, use of this function will produce invisible text.

This function is available only if the TD$CLR and TD$PHR bits are set in the terminal capabilities flags.

15.131 SELECT WHITE TEXT (TCRT –1,133)

This command selects the current text color to be white on the standard background. This function occupies one screen position and is therefore considered inferior to the general color selection commands TCRT –2,n and TCRT –3,n. It is provided for compatibility with certain terminals, such as the AM-70, only and is not recommended for use in new software development.

This function is available only if the TD$CLR and TD$PHR bits are set in the terminal capabilities flags.

15.132 SELECT BLUE TEXT (TCRT –1,134)

This command selects the current text color to be blue on the standard background. This function occupies one screen position and is therefore considered inferior to the general color selection commands TCRT –2,n and TCRT –3,n. It is provided for compatibility with certain terminals, such as the AM-70, only and is not recommended for use in new software development.

This function is available only if the TD$CLR and TD$PHR bits are set in the terminal capabilities flags.

15.133 SELECT MAGENTA TEXT (TCRT –1,135)

This command selects the current text color to be magenta on the standard background. This function occupies one screen position and is therefore considered inferior to the general color selection commands TCRT –2,n and TCRT –3,n. It is provided for compatibility with certain terminals, such as the AM-70, only and is not recommended for use in new software development.

This function is available only if the TD$CLR and TD$PHR bits are set in the terminal capabilities flags.

15.134 SELECT RED TEXT (TCRT –1,136)

This command selects the current text color to be red on the standard background. This function occupies one screen position and is therefore considered inferior to the general color selection commands TCRT –2,n and TCRT –3,n. It is provided for compatibility with certain terminals, such as the AM-70, only and is not recommended for use in new software development.

This function is available only if the TD$CLR and TD$PHR bits are set in the terminal capabilities flags.

15.135 SELECT YELLOW TEXT (TCRT –1,137)

This command selects the current text color to be yellow on the standard background. This function occupies one screen position and is therefore considered inferior to the general color selection commands TCRT –2,n and TCRT –3,n. It is provided for compatibility with certain terminals, such as the AM-70, only and is not recommended for use in new software development.

This function is available only if the TD$CLR and TD$PHR bits are set in the terminal capabilities flags.

15.136 SELECT GREEN TEXT (TCRT –1,138)

This command selects the current text color to be green on the standard background. This function occupies one screen position and is therefore considered inferior to the general color selection commands TCRT –2,n and TCRT –3,n. It is provided for compatibility with certain terminals, such as the AM-70, only and is not recommended for use in new software development.

This function is available only if the TD$CLR and TD$PHR bits are set in the terminal capabilities flags.

15.137 SELECT CYAN TEXT (TCRT –1,139)

This command selects the current text color to be cyan on the standard background. This function occupies one screen position and is therefore considered inferior to the general color selection commands TCRT –2,n and TCRT –3,n. It is provided for compatibility with certain terminals, such as the AM-70, only and is not recommended for use in new software development.

This function is available only if the TD$CLR and TD$PHR bits are set in the terminal capabilities flags.

15.138 SELECT BLACK REVERSE TEXT (TCRT –1,140)

This command selects the current text color to be the background color on a black background. This function occupies one screen position and is therefore considered inferior to the general color selection commands TCRT –2,n and TCRT –3,n. It is provided for compatibility with certain terminals, such as the AM-70, only and is not recommended for use in new software development.

If the selected background color is black, as is normal, use of this function will produce invisible text.

This function is available only if the TD$CLR and TD$PHR bits are set in the terminal capabilities flags.

15.139 SELECT WHITE REVERSE TEXT (TCRT –1,141)

This command selects the current text color to be the background color on a white background. This function occupies one screen position and is therefore considered inferior to the general color selection commands TCRT –2,n and TCRT –3,n. It is provided for compatibility with certain terminals, such as the AM-70, only and is not recommended for use in new software development.

This function is available only if the TD$CLR and TD$PHR bits are set in the terminal capabilities flags.

15.140 SELECT BLUE REVERSE TEXT (TCRT –1,142)

This command selects the current text color to be the background color on a blue background. This function occupies one screen position and is therefore considered inferior to the general color selection commands TCRT –2,n and TCRT –3,n. It is provided for compatibility with certain terminals, such as the AM-70, only and is not recommended for use in new software development.

This function is available only if the TD$CLR and TD$PHR bits are set in the terminal capabilities flags.

15.141 SELECT MAGENTA REVERSE TEXT (TCRT –1,143)

This command selects the current text color to be the background color on a magenta background. This function occupies one screen position and is therefore considered inferior to the general color selection commands TCRT –2,n and TCRT –3,n. It is provided for compatibility with certain terminals, such as the AM-70, only and is not recommended for use in new software development.

This function is available only if the TD$CLR and TD$PHR bits are set in the terminal capabilities flags.

15.142 SELECT RED REVERSE TEXT (TCRT –1,144)

This command selects the current text color to be the background color on a red background. This function occupies one screen position and is therefore considered inferior to the general color selection commands TCRT –2,n and TCRT –3,n. It is provided for compatibility with certain terminals, such as the AM-70, only and is not recommended for use in new software development.

This function is available only if the TD$CLR and TD$PHR bits are set in the terminal capabilities flags.

15.143∞SELECT YELLOW REVERSE TEXT (TCRT –1,145)

This command selects the current text color to be the background color on a yellow background. This function occupies one screen position and is therefore considered inferior to the general color selection commands TCRT –2,n and TCRT –3,n. It is provided for compatibility with certain terminals, such as the AM-70, only and is not recommended for use in new software development.

This function is available only if the TD$CLR and TD$PHR bits are set in the terminal capabilities flags.

15.144∞SELECT GREEN REVERSE TEXT (TCRT –1,146)

This command selects the current text color to be the background color on a green background. This function occupies one screen position and is therefore considered inferior to the general color selection commands TCRT –2,n and TCRT –3,n. It is provided for compatibility with certain terminals, such as the AM-70, only and is not recommended for use in new software development.

This function is available only if the TD$CLR and TD$PHR bits are set in the terminal capabilities flags.

15.145∞SELECT CYAN REVERSE TEXT (TCRT –1,147)

This command selects the current text color to be the background color on a cyan background. This function occupies one screen position and is therefore considered inferior to the general color selection commands TCRT –2,n and TCRT –3,n. It is provided for compatibility with certain terminals, such as the AM-70, only and is not recommended for use in new software development.

This function is available only if the TD$CLR and TD$PHR bits are set in the terminal capabilities flags.

15.146∞SAVE A RECTANGULAR AREA (TCRT –1,148)

This call saves a rectangular area of the screen in an off-screen buffer within the AM-75. It is intended to be used to save an area of the screen which is to be overlaid with a pop-up menu, and later restored when the menu is removed. Using this call removes the requirement for applications software to maintain its own screen image, greatly increasing the speed of display and reducing the communications overhead of using pop-up menus.

Screen areas are saved in a "stack" within the AM-75, in a last-in, first-out fashion. That is, the most recent area to be saved is that one that will be restored by the next Restore Area command (TCRT –1,149).

To use this call, issue a TCRT –1,148 call, followed by the dimensions of the area to be saved, encoded the same as the fill area and scroll area commands from Table B-5. The saved area extends from the current cursor location for the number of rows and columns specified.

The total number of characters that may be saved in all Save Area commands can be found by doing a TRMCHR call and looking at the TC.SVA field, a word value giving the total number of

bytes that can be saved at any one time. Note that this field contains a static value representing the total memory available for this purpose, and is not updated as saves and restores are performed. The amount of memory available for the stack is affected by the number of lines and columns being displayed, whether Dual Sessions is enabled, and whether the Graphics Cartridge is installed. It is fairly safe to assume that in any mode there will be sufficient memory to save one 80x24 page.

Areas that are saved remain valid until the screen scrolls, is cleared, or a Remove Split Screen command is executed.

Not available in Esprit III emulation.

15.147 RESTORE A RECTANGULAR SCREEN AREA (TCRT –1,149)

This call allows you to restore a rectangular area of the screen which has previously been saved by a TCRT –1,148 call.

To restore the most recently saved area, issue a TCRT –1,149 call, followed by the dimensions of the area to be restored. The area will be restored at the current cursor location.

Screen areas are saved in a "stack" within the terminal, in a last-in, first-out fashion. That is, the most recent area to be saved is the one that will be restored by the next Restore Area command.

You can restore a saved area to a screen position other than the one it was saved from, although extreme care must be taken to make sure that the desired results are achieved, particularly in field attribute emulations.

Restoring an area with a different size than the one it was saved with will yield unpredictable results.

Not available in Esprit III emulation.

15.148 ENTER FULL GRAPHICS MODE (TCRT –1,150)

This call is used to switch from text mode to graphics mode when using AM-62A or AM-70 emulations. It is not required, but is ignored by the AM-72 native modes.

This call clears the screen and prepares the terminal for graphics display instead of text display.

15.149 EXIT FULL GRAPHICS MODE (TCRT –1,151)

This call is used to switch from graphics mode to text mode when using AM-62 or AM-70 emulations. It is not required, but is ignored by the AM-72 native modes.

This call clears the screen and prepares the terminal for text display instead of graphics display.

15.150∞DRAW A BOX WITH ROUNDED CORNERS (TCRT –1,152)

This command allows you to draw a "box" on the screen. This box encloses a rectangular area of the screen with line drawing graphics characters. This command differs from TCRT -1,92 in that the box has rounded, rather than square corners. Note that only the outside lines of the box are drawn on the screen; characters inside and outside the box are unaffected by this command. If you wish to clear the inside of the box, the use of the Block Fill with Character command (TCRT –1,91) is recommended.

To draw a box, first position the cursor at the upper left corner of the box. Issue a TCRT –1,152 call followed by height and width codes taken from Table B-5, which define the lower right corner of the area. The screen display will then be updated.

All validation of the size of the box to be drawn is left to the application program. Using this command with a box specified such that it will extend beyond the visible screen boundaries will have unpredictable results.

15.151∞DRAW A WINDOW STYLE BOX (TCRT –1,153)

This command allows you to draw a "box" on the screen. This box encloses a rectangular area of the screen with line drawing graphics characters. This command differs from TCRT -1,92 in that the box shape has been specifically designed to be attractive when used as a pop-up window. Note that only the outside lines of the box are drawn on the screen—characters inside and outside the box are unaffected by this command. If you wish to clear the inside of the box, the use of the Block Fill with Character command (TCRT –1,91) is recommended.

To draw a box, first position the cursor at the upper left corner of the box. Issue a TCRT –1,153 call followed by height and width codes taken from Table B-5, which define the lower right corner of the area. The screen display will then be updated.

All validation of the size of the box to be drawn is left to the application program. Using this command with a box specified such that it will extend beyond the visible screen boundaries will have unpredictable results.

15.152∞DRAW A BOX WITH DOUBLE LINES (TCRT –1,154)

This command allows you to draw a "box" on the screen. This box encloses a rectangular area of the screen with line drawing graphics characters. This command differs from TCRT -1,92 in that the box has double lines surrounding the box area, rather than single lines. Note that only the outside lines of the box are drawn on the screen; characters inside and outside the box are unaffected by this command. If you wish to clear the inside of the box, the use of the Block Fill with Character command (TCRT –1,91) is recommended.

To draw a box, first position the cursor at the upper left corner of the box. Issue a TCRT –1,154 call followed by height and width codes taken from Table B-5, which define the lower right corner of the area. The screen display will then be updated.

All validation of the size of the box to be drawn is left to the application program. Using this command with a box specified such that it will extend beyond the visible screen boundaries will have unpredictable results.

15.153 ENABLE PROPORTIONALLY SPACED TEXT (TCRT –1,155)

This call enables the display of proportionally spaced text. All subsequent text output (until a TCRT -1,156 call) is displayed in the currently selected proportional font.

Available in AM-72 native modes only.

15.154 DISABLE PROPORTIONALLY SPACED TEXT (TCRT –1,156)

This call disables the display of proportionally spaced text.

Available in AM-72 native modes only.

15.155 SELECT COLOR PALETTE BY RGB VALUE (TCRT –1,157)

This call selects which of the 64 available colors you wish to select for the 16-color display palette. You can change any of the 16 palette indices individually or as part of a group, or you can reset the entire palette to its power-up configuration.

For any of the 16 palette indices, you can specify the desired color in terms of its RGB color value. (for selecting color palette indices based on HLS color specification, see TCRT -1,177)

To select a color index, start by issuing a TCRT -1,157 call. For each color index you wish to select, output a color index and the desired R (red), G (green), and B (blue) color values. You can repeat this process for as many color indices as you wish. End the color selection process by sending a 177 octal (7F hex) code.

The color index, R, G and B values are all output as eight bit integers using the packing method described in Appendix A of the AM-72 Programmer's Manual. The RGB values for the 64 available colors are shown in Appendix G of that manual.

To reset the color palette to its power-up configuration, issue a TCRT -1,157 call and immediately follow it with a 176 octal (7E hex) code. This resets the color palette and terminals the palette selection command.

It is best to reset as many palette as possible with a single TCRT call in order to eliminate display artifacts such as flashing or unpleasant color combinations.

15.156 ENABLE GRAPHICS CURSOR (TCRT –1,158)

This call enables the graphics cursor. The selected cursor shape (see TCRT -1,160) will be displayed on the screen and will track the movements of the mouse, a report will be sent to the host computer giving the current location and button status. The format of this report is described below.

To enable the graphics cursor, start by issuing a TCRT -1,158 call. Follow this with a "C" to select row/column coordinates. Next send it the string you wish to have sent to you at the start of each graphics cursor report. Specify this string by sending the number of characters in the string followed by the string itself. The number of characters is encoded as shown in table F-3 of the AM-72 Programmer's Manual.

Each time a mouse button is pressed or released, or you specifically request a report, the AM-72 will send a graphics cursor report consisting of your specified string, the row and column the mouse is currently positioned on (encoded as shown in table F-1 of the AM-72 Programmer's Manual) followed by the button status.

The button status can be decoded by subtracting 32. From the returned character and using the remianing value as a binary button mask, where the low-order bit corresponds to the status of button1, the next bit button 2, etc.

You can also select mouse reports to the returned with XY coordinates in a 32,767 by 32,767 world space. This is done by sending a "W" in place of the "C" described above. All subsequent position reports will return the current X and Y locations in place of the row and column positions described above. The X and Y values are packed as described in Appendix A.

15.157 DISABLE GRAPHICS CURSOR (TCRT –1,159)

This call disables the graphics cursor. The cursor itself is removed from the screen, and any subsequent button manipulation will not send position reports.

15.158 SELECT GRAPHICS CURSOR SHAPE (TCRT –1,160)

This call allows you to select the shape of the graphics cursor There are 16 different shapes to choose from.

To select a cursor shape, issue a TCRT -1,160 call, followed by the single character code for the desired cursor shape, from the table below.

space	White arrow
!	Black arrow
"	Hand
#	Pointing hand
$	Four-way arrow
%	Two-way arrow, horizontal
&	Two-way arrow, vertical
'	Pencil
(	Watch1
)	Watch2
*	Watch3
+	Watch4
,	Watch5
-	Watch6
.	Watch7
/	Watch8

15.159 INQUIRE GRAPHICS CURSOR LOCATION TCRT –1,161)

This call allows you to inquire the current state of the mouse, without waiting for a button press or release. Upon issuing a TCRT -1,161 call, the AM-72 will send a report to the host giving the current location and button status. The format of the report will be as described under TCRT -1,158, above.

15.160∞DEFINE GRAPHICS CURSOR REGIONS TCRT –1,162)

To make it easy to adapt existing applications to use the graphics cursor, the AM-72 allows you to specify up to 64 rectangular areas of the screen which will be sensitive to mouse button presses. Any time a mouse button is pressed in a particular region, a string which you have associated with that specific region will be sent to the host computer. This allows you to adapt an application to mouse menu selection without the application needing any knowledge of mouse reports or other details.

To define graphics cursor regions, start by issuing a TCRT -1,162 call. Follow this with the number of regions you are defining, encoded as defined in table F-3 of the AM-72 Programmer's Manual.

For each if the regions you wish to define, now output the string to be sent upon a mouse press within the region and the corners of the rectangle you are defining. The string is specified by sending in length of string as encoded by table F-3 of the AM-72 Programmer's Manual, followed by the characters of the string itself.

If you are using row/column mouse reports, specify the mouse sensitive rectangle by sending the row/column coordinates fo the upper left and lower right corners of the rectangle. The row and column are specified according to table F-1 of the AM-72 Programmer's Manual.

If you are using XY mouse reports, specify the mouse-sensitive rectangle by sending the XY coordinates of the upper left and lower right corners of the rectangle. The XY coordinates are packed as described in Appendix A.

Continue outputting strings and rectangle definitions until all regions have been defined.

15.161∞OUTPUT FORM FEED CHARACTER (TCRT –1,163)

This call outputs a form feed character (ASCII 012 decimal) at the current screen location. If the terminal driver does not support this call, nothing is output.

15.162∞OUTPUT LINE FEED CHARACTER (TCRT –1,164)

This call outputs a line feed character (ASCII 010 decimal) at the current screen location. If the terminal driver does not support this call, nothing is output.

15.163∞OUTPUT NEW LINE CHARACTER (TCRT –1,165)

This call outputs a new line character at the current screen location. If the terminal driver does not support this call, nothing is output.

15.164∞OUTPUT VERTICAL TAB CHARACTER (TCRT –1,166)

This call outputs a vertical tab character (ASCII 011 decimal) at the current screen location. If the terminal driver does not support this call, nothing is output.

15.165∞OUTPUT PLUS-OR-MINUS CHARACTER (TCRT –1,167)

This call outputs a plus-or-minus character at the current screen location. If the terminal driver does not support this call, nothing is output.

15.166∞OUTPUT GREATER-THAN-OR-EQUAL CHARACTER (TCRT –1,168)

This call outputs a greater-than-or-equal character at the current screen location. If the terminal driver does not support this call, nothing is output.

15.167∞OUTPUT LESS-THAN-OR-EQUAL CHARACTER (TCRT –1,169)

This call outputs a less-than-or-equal character at the current screen location. If the terminal driver does not support this call, nothing is output.

15.168∞OUTPUT NOT-EQUAL CHARACTER (TCRT –1,170)

This call outputs a not-equal character at the current screen location. If the terminal driver does not support this call, nothing is output.

15.169∞OUTPUT BRITISH POUND CHARACTER (TCRT –1,171)

This call outputs a British pound character at the current screen location. If the terminal driver does not support this call, nothing is output.

15.170∞OUTPUT PI CHARACTER (TCRT –1,172)

This call outputs a pi character at the current screen location. If the terminal driver does not support this call, nothing is output.

15.171∞ENTER BIDIRECTIONAL PRINT MODE (TCRT –1,173)

This call enables the bidirectional ("concurrent") print function. Upon issuance of this call, the terminal sends all subsequent data to the printer port designated in Setup Mode, as well as displaying it on the screen. While in this mode, the terminal also passes any data received from the serial printer port back to the host port.

The terminal indicates that it is in a print node by displaying "PRNT" in the upper right corner of the top status line. Bidirectional print mode is normally cleared by sending the exit command (–1,174), but entering Setup Mode on the terminal will also clear it.

15.172∞EXIT BIDIRECTIONAL PRINT MODE (TCRT –1,174)

This call disables the bidirectional print function which was invoked via the (–1,173) call. After issuing this call, all data will again be displayed only on the screen and not sent out the printer port.

15.173∞SET TERMINAL TIME (TCRT –1,175) (VERSION 2.00 AND LATER)

This call sets the AM-65's internal clock, as used by the Toolkit Alarm Clock. The Toolkit calendar has a fixed initial month and cannot be set by command.

15.174∞SET TERMINAL DATE (TCRT –1,176) (VERSION 2.00 AND LATER)

This call sets the date inside the AM-72's calendar, based on the current system date.

15.175∞SELECT COLOR PALETTE BY HLS VALUE (TCRT –1,177)

This call selects which of the 64 available colors you wish to select for the 16-color display palette. You can change any of the 16 palette indices individually or as part of a group, or you can reset the entire palette to its power-up configuration.

For any of the 16 palette indices you can specify the desired color in terms of its HLS color value. (For selecting color palette indices based on RGB color specification, see TCRT -1,157)

To select a color index, start by issuing a TCRT -1,177 call. For each color index you wish to select, output a color index and the desired H (Hue), L (Lightness), and S (Saturation) color values. You can repeat this process for as many color indices as you wish. End the color selection process by sending a 177 octal (7F hex) code.

The color index is an eight bit value while H,L,S are 12-bit values. All are output using the packing method described in Appendix A of the AM-72 Programmer's Manual. The HLS values for the 64 available colors are shown in Appendix G of that manual.

To reset the color palette to its power-up configuration, issue a TCRT -1,157 call and immediately follow it with a 176 octal (7E hex) code. This resets the color palette and terminates the palette selection commands.

It is best to reset as many palette indices as possible with a single TCRT call in order to eliminate display artifacts such as flashing or unpleasant color combinations.

15.176∞SELECT PC CHARACTER SET (TCRT –1,178) (VERSION 2.00 AND LATER)

Primarily used by the VPC Coprocessor for PC Terminal operation, this call selects the PC character set instead of the native character set.

15.177∞SELECT DEFAULT CHARACTER SET (TCRT –1,179) (VERSION 2.00 AND LATER)

Primarily used by the VPC Coprocessor for PC Terminal operation, this call returns the terminal to the character set selected in Setup Mode.

15.178∞SELECT PC TERMINAL EMULATION (TCRT –1,180) (VERSION 2.00 AND LATER

Primarily used by the VPC Coprocessor for PC Terminal operation, this call selects Scan Code keyboard, and 25 line display with no bottom status line.

15.179∞SELECT ASCII TERMINAL OPERATION (TCRT –1,181) (VERSION 2.00 AND LATER)

Primarily used by the VPC Coprocessor to exit PC Terminal operation, this call selects ASCII keyboard encoding, and 24 line display with top and bottom status lines.

15.180∞SELECT AUX PORT HOST (TCRT –1,182) (VERSION 2.00 AND LATER)

The Aux port on the terminal is enabled for host communication. This command has the same effect as using the "P" command in Setup Mode to set the Aux port. For more information on the AM-65's serial ports, see the *AM-65 Reference Guide.*

15.181∞SELECT MAIN PORT HOST (TCRT –1,183) (VERSION 2.00 AND LATER)

The Main port on the terminal is enabled for host communication. This command has the same effect as using the "P" command in Setup Mode to set the Main port. For more information on the AM-65's serial ports, see the *AM-65 Reference Guide.*

15.182∞TOGGLE HOST PORTS (TCRT –1,184) (VERSION 2.00 AND LATER)

This call toggles between Main and Aux ports for host communication. This command has the same effect as using the "P" command in Setup Mode to toggle ports. For more information on the AM-65's serial ports, see the *AM-65 Reference Guide.*

15.183∞SELECT 8-BIT CHARACTER DISPLAY (TCRT –1,185)

For terminals which support both 8-bit character sets (such as ISO Latin-1) and 7-bit National Replacement Character (NRC) sets, this call selects the 8-bit character set for all subsequent character displays.

15.184∞SELECT 7-BIT CHARACTER DISPLAY (TCRT –1,186)

For terminals which support both 8-bit character set (such as ISO Latin-1) and 7-bit National Replacement Character (NRC) sets, this call selects the 7-bit character NRC set for all subsequent character displays.

15.185 SELECT 8-BIT KEYBOARD MODE (TCRT –1,187)

For terminals which support both 8-bit character set (such as ISO Latin-1) and 7-bit National Replacement Character (NRC) sets, this call causes the keyboard to generate the 8-bit character set for all subsequent keystrokes.

15.186 SELECT 7-BIT KEYBOARD MODE (TCRT –1,188)

For terminals which support both 8-bit character set (such as ISO Latin-1) and 7-bit National Replacement Character (NRC) sets, this call causes the keyboard to generate the 7-bit character set for all subsequent keystrokes.

15.187 SELECT PRIMARY PRINTER PORT (TCRT –1,189)

When using a workstation as a terminal emulator, this call selects the first printer port (e.g., LPT1:) as the port to receive all output from terminal printer port commands.

15.188 SELECT SECONDARY PRINTER PORT (TCRT –1,190)

When using a workstation as a terminal emulator, this call selects the second printer port (e.g., LPT2:) as the port to receive all output from terminal printer port commands.

15.189 SELECT 161-COLUMN DISPLAY MODE (TCRT –1,191)

This call selects 161-column display mode. to determine if this call is available, you must first retrieve the terminal features bitmap (not TC.FLG) and test the appropriate bit for the presence of this specific TCRT call.

Please note that some versions of the terminal driver for the AM-75, which supports 161-column mode, erroneously assigned this function to a different TCRT number. This has been corrected in the version supplied with AMOS 2.2C.

15.190 RESERVED FUNCTIONS (TCRT –1,192 THROUGH TCRT –1,255)

These functions are reserved for future expansion by Alpha Micro. If you have a specific need for a new TCRT function, you may contact Alpha Micro to reserve a TCRT call number. The reservation process avoids problems arising from incompatible extensions to the TCRT functions.

15.191 SET FOREGROUND COLOR (TCRT –2,N)

This call selects the foreground color to be used for all further text display. The call takes an eight-bit argument "n" which specifies the color to be selected. The range of values for this argument can be obtained via the TRMCHR monitor call or the TRMCHR XCALL routine from AlphaBASIC. It is assumed that for all color terminals (i.e., number of colors is greater than zero), the minimum number of available colors will be eight. The argument values for the first eight colors can be found in Table B-3.

This call does not occupy a screen position.

This function is available only if the TD$CLR bit is set in the terminal capabilities flags.

15.192 SET BACKGROUND COLOR (TCRT –3,N)

This call selects the background color to be used for all further text display. The call takes an eight-bit argument "n" which specifies the color to be selected. The range of values for this argument can be obtained via the TRMCHR monitor call or the TRMCHR XCALL routine from AlphaBASIC. It is assumed that for all color terminals (i.e., number of colors is greater than zero), the minimum number of available colors will be eight. The argument values for the first eight colors can be found in Table B-3.

This call does not occupy a screen position.

This function is available only if the TD$CLR bit is set in the terminal capabilities flags.

15.193 RESERVED FUNCTIONS (TCRT –4,N THROUGH TCRT –127,N)

These functions are reserved for future expansion by Alpha Micro. If you have a specific need for a new TCRT function, you may contact Alpha Micro to reserve a TCRT call number. The reservation process avoids problems arising from incompatable extensions to the TCRT functions.

PART V

CREATING A TERMINAL DRIVER PROGRAM

CHAPTER 16

ASSEMBLY LANGUAGE TERMINAL DRIVERS

16.1 TERMINAL DRIVER STRUCTURE

The terminal driver is comprised of a header area containing descriptive information and subroutine dispatches, and a set of subroutines which implement special terminal driver functions.

Terminal drivers are stored in DSK0:[1,6]. The source to sample terminal drivers can be found on the standard AMOS release in account DSK0:[10,2].

16.1.1 The Terminal Driver Header

The terminal driver header contains both descriptive information, describing the particular terminal driver and its characteristics, and subroutine dispatches which allow the terminal service system to transfer control to the terminal driver support routines.

The terminal driver header is structured as follows:

TD.TYP	word	Type control bits
TD.INP	word	Branch to the input routine
TD.OTP	word	Branch to the output routine
TD.ECH	word	Branch to the echo routine
TD.CRT	word	Branch to the TCRT routine
TD.INI	word	Branch to the initialization routine
TD.IMP	word	Number of impure bytes to allocate
TD.ROW	byte	Obsolete
TD.COL	byte	Obsolete
TD.FLG	lword	Terminal capability flags
TD.TCH	word	Branch to TRMCHR routine

16.1.1.1 The Type Control Bits (TD.TYP)

This word contains a number of single-bit flags describing the overall characteristics of the terminal driver. These bits are:

TD$LCL This terminal has local echo

TD$NEW This is a "new" driver, in that it contains header entries and routines for initialization, impure size, and flags. There should be very few "old" drivers left anywhere.

TD$TCH This driver contains a TRMCHR routine.

TD$NTD This driver is not a "real" terminal driver. This flag is used by the AMOS Installation Program when initially configuring a system. It has no meaning outside of that process and can otherwise be ignored.

16.1.1.2 The Input Routine Transfer Vector (TD.INP)

This word consists of a branch to the input routine of the terminal driver. If a branch will not reach, it must be coded as a branch to a jump which will reach.

16.1.1.3 The Output Routine Transfer Vector (TD.OTP)

This word consists of a branch to the output routine of the terminal driver. If a branch will not reach, it must be coded as a branch to a jump which will reach.

16.1.1.4 The Echo Routine Transfer Vector (TD.ECH)

This word consists of a branch to the echo routine of the terminal driver. If a branch will not reach, it must be coded as a branch to a jump which will reach.

16.1.1.5 The TCRT Routine Transfer Vector (TD.CRT)

This word consists of a branch to the TCRT routine of the terminal driver. If a branch will not reach, it must be coded as a branch to a jump which will reach.

16.1.1.6 The Initialization Routine Transfer Vector (TD.INI)

This word consists of a branch to the initialization routine of the terminal driver. If a branch will not reach, it must be coded as a branch to a jump which will reach.

16.1.1.7∞The Impure Area Size (TD.IMP)

This word defines the number of bytes to be allocated as impure space for this terminal. A pointer to this impure space will be placed in the longword field T.IMP within the terminal definition block defining this terminal.

Such impure space is useful for storing the current values of such items as current color, or current screen width. Because terminal drivers are shared and must therefore always be coded as pure and re-entrant, this impure space is the only permanent storage available to the terminal driver routines.

16.1.1.8∞The Row and Columns Sizes (TD.ROW and TD.COL)

These fields are obsolete and should not be used. All software must use the TRMCHR call to determine the height and width of the terminal screen.

16.1.1.9∞The Terminal Capabilities Word (TD.FLG)

This longword field contains a number of single bit flags which describe the capabilities of the terminal. It is these flags that are returned by the TRMCHR call in the field TC.FLG.

TD$GRA	terminal supports full graphics capability
TD$NSP	terminal has "no space" attributes
TD$PHR	terminal has AM-70 color mode
TD$MOD	terminal has "mode" attributes (instead of field)
TD$BOX	terminal has draw and scroll box commands
TD$SMT	terminal has smooth scroll
TD$132	terminal has 80/132 column support
TD$PRT	terminal has printer support
TD$LID	terminal has line insert/delete
TD$CID	terminal has character insert/delete
TD$RVA	terminal has reverse video
TD$DIM	terminal has dim
TD$BLN	terminal has blink
TD$UND	terminal has underscore
TD$EOS	terminal has erase to end of screen
TD$EOL	terminal has erase to end of line
TD$SPL	terminal has split screen
TD$STS	terminal has status line
TD$AMS	terminal is an AlphaTERMINAL
TD$MLT	terminal has multi-key translation
TD$CLR	terminal has color capability
TD$KID	terminal has column insert/delete
TD$BLF	terminal has block fill
TD$ALP	terminal has alternate page

16.1.1.10 The TRMCHR Routine Transfer Vector (TD.TCH)

This word consists of a branch to the TRMCHR routine of the terminal driver. If a branch will not reach, it must be coded as a branch to a jump which will reach.

16.1.2 The Terminal Driver Input Routine

The input routine of the terminal driver receives all characters input from the terminal keyboard before they are passed on to the remainder of the terminal service system. This gives the terminal driver an opportunity to intercept specific characters, either translating them to one or more different characters or ignoring the keystroke altogether. It is within the terminal driver input routine that multi-keystroke function keys are detected and translated.

When the input routine is called the following registers will be set up:

D1 Contains the input character

A5 Indexes the terminal control block for this terminal

The N-bit will be set if the character in D1 is possibly in a multi-keystroke function key sequence.

The following registers are available for use by the input routine. All others must be saved and restored by the input routine if they are to be modified.

A0, A3, A6, D1, D6, D7

The character is returned to the terminal service system in D1. If a simple substitution is to take place, the new character should simply be placed in D1.

The input routine signals to the rest of the terminal service system what action is to be taken by setting the condition code flags prior to returning to the terminal service system. The following settings are valid:

Z-bit set	ignore this character (the contents of D1 are ignored)
V-bit set	force all eight bits of this character into the buffer regardless of input mode
N-bit set	the character in D1 could be the start of a multi keystroke function key
all bits reset	the character in D1 is to be processed normally (this is the normal case)

Because the input routine is called at interrupt level, the routine does not have a job context and is restricted as to the monitor calls it can issue. Specifically, you *cannot issue SLEEP, TTY, TTYI, or any other call requiring a job context. In addition, because this routine is being executed at interrupt level, it is vital that it be made as efficient as possible to avoid adverse impact on system performance.*

16.1.3∞The Terminal Driver Echo Routine

The echo routine of the terminal driver receives all characters which are to be echoed back to the user. The terminal driver is given control to allow different echoing styles for different terminals. For example, a video terminal may echo a rubout as a "backspace-space-backspace" sequence, removing the character from the screen. A printing terminal, however, may echo a rubout by backspacing and overstriking. It is up to the terminal driver echo routine to determine the behavior of echoed characters. While rubouts, tabs, and control-U are the characters most commonly echoed specially, there is nothing to prevent a terminal driver from treating either more or fewer characters as special cases.

When the echo routine is called the following registers will be set up:

D1 Contains the character to be echoed

A5 Indexes the terminal control block for this terminal

The following registers are available for use by the echo routine. All others must be saved and restored by the routine if they are to be modified.

A0, A2, A3, A6, D1, D2, D3, D6, D7

If no special action is taken by the terminal driver routine, the character to be echoed should be returned in D1 and the Z-bit reset. If the terminal driver does its own echoing, setting the Z-bit before returning will prevent the terminal service system from performing its default echo processing.

Because the echo routine is called at interrupt level, the routine does not have a job context and is restricted as to the monitor calls it can issue. Specifically, you cannot issue SLEEP, TTY, TTYI, or any other call requiring a job context. In addition, because this routine is being executed at interrupt level, it is vital that it be made as efficient as possible to avoid adverse impact on system performance.

16.1.4∞The Terminal Driver Output Routine

The output routine of the terminal driver receives all characters which are to be output to the terminal. This gives the terminal driver an opportunity to intercept specific characters, either translating them to one or more different characters or ignoring the character altogether.

When the output routine is called the following registers will be set up:

D1 Contains the character to be output

A5 Indexes the terminal control block for this terminal

The following registers are available for use by the output routine. All others must be saved and restored by the routine if they are to be modified.

A0, A2, A3, A6, D1, D2, D3, D6, D7

The output character is returned to the terminal service system in D1. If a simple substitution is to take place, the new character should simply be placed in D1.

The output routine signals to the rest of the terminal service system what action is to be taken by setting the condition code flags prior to returning to the terminal service system. The following settings are valid:

Z-bit set	ignore this character
N-bit set	output this character and adjust output position (this is the normal case)
all bits reset	output this character but do not adjust output position

Because the output routine is called at interrupt level, the routine does not have a job context and is restricted as to the monitor calls it can issue. Specifically, you cannot issue SLEEP, TTY, TTYI, or any other call requiring a job context. In addition, because this routine is being executed at interrupt level, it is vital that it be made as efficient as possible to avoid adverse impact on system performance.

16.1.5 The Terminal Driver TCRT Routine

The TCRT routine of the terminal driver receives all TCRT calls and is resposible for translating the specific TCRT call to the series of characters required to invoke the command on the terminal. While this action normally consists of looking up the function in a table of character sequences, the TCRT routine can perform almost any type of computation to detarmine the proper seqeuence.

When the TCRT routine is called the following registers will be set up:

D1	Contains the TCRT command exactly as specified by the user
A5	Indexes the terminal control block for this terminal

The following registers are available for use by the TCRT routine. All others must be saved and restored by the routine if they are to be modified.

A0, A6, D1, D6, D7

The TCRT routine returns whether or not the operation was successful by returning the Z-bit to the user. If the operation was successful, set the Z-bit before returning to the terminal service system. If the operation was not successful, reset the Z-bit.

You can output characters from within the TCRT routine via the standard terminal output calls such as TTY, TTYL, TTYI, etc. While the SLEEP call may be used, it is not usually effective as a method of introducting delays, since characters are buffered and therefore output at the full baud rate, regardless of what the outputting job is doing. The standard method of introducing delays is output a sequence of null bytes which are ignored by most terminals.

Remember that all output from the TCRT routine eventually ends up going out through the terminal driver output routine.

16.1.6 The Terminal Driver Initialization Routine

The initialization routine of the terminal driver is called when the terminal is first defined at system initialization time. Its purpose is to set the terminal into a known state so that the remainder of the system boot process may continue.

When the initialization routine is called the following registers will be set up:

A5 Indexes the terminal control block for this terminal

The following registers are available for use by the initialization routine. All others must be saved and restored by the routine if they are to be modified.

A6, D1, D3, D6

At the time that the initialization routine is called, the terminal is not attached to a job, thus preventing the use of normal terminal output calls such as TTY, TTYL, or TTYI. Any such output will appear on the booting terminal, rather than on the terminal being initialized.

To output to the terminal being initialized, use the TRMBFQ monitor call.

Remember that all output from the this routine eventually ends up going out through the terminal driver output routine.

Because the initialization routines of all terminals on the system are called in rapid succession during system bootup, care must be taken in this routine to not run the system out of queue blocks, since no extra queue blocks may have been allocated at this point in the boot process. For this reason, it is a good idea to test the system longword QFREE and skip over the initialization process if it contains a value less than 15 (decimal).

16.1.7 The Terminal Driver TRMCHR Routine

The TRMCHR routine of the terminal driver is called whenever the job attached to the terminal issues a TRMCHR monitor call. It is the responsibility of the TRMCHR routine to fill in the user's argument block with the current settings and values describing the terminal.

When the TRMCHR routine is called the following registers will be set up:

D2 Contains the flags specified on the TRMCHR call

A1 Indexes the user's argument block

A2 Indexes the current terminal driver

A5 Indexes the terminal control block for this terminal

The following registers are available for use by the TRMCHR routine. All others must be saved and restored by the routine if they are to be modified.

A1, A2, A6, D2, D6, D7

The TRMCHR routine must fill in all fields in the user's argument block. If the TC$BMP flag is set in D2, then the TRMCHR routine must also return the TCRT code availability bitmap.

The fields that must be returned by the TRMCHR routine in all cases are:

TC.FLG Longword specifying the terminal capabilities, as described in section 16.1.1.9 of this chapter.

TC.ROW Word specifying the number of rows currently available on the terminal screen. Note that the value returned in this field may vary depending on the current mode the terminal is in.

TC.COL Word specifying the number of columns currently available on the terminal screen. Note that the value returned in this field may vary depending on the current mode the terminal is in.

TC.CLR Word specifying the number of different colors available on the terminal. A value of zero in this field is assumed to mean a monochrome terminal.

TC.FGC Word specifying the currently selected foreground color. Zero is returned for monochrome terminals.

TC.BGC Word specifying the currently selected background color. Zero is returned for monochrome terminals.

TC.WNR Word specifying the number of rows in the currently selected window. (This field is only valid when the MULTI window based environment manager is in use.)

TC.WNC Word specifying the number of columns in the currently selected window. (This field is only valid when the MULTI window based environment manager is in use.)

If the TC$BMP flag is set in D2, indicating that the user has requested the TCRT feature availability bitmap, the TRMCHR routine must also return the following field:

TC.BMP A 32-byte table describing the TCRT codes in the –1,n set. For each of the 256 TCRT codes a corresponding bit is set if the TCRT call is available. If the TCRT call is not available, the bit will be zero. Bits are assigned starting at the first byte in the table, with the low-order bit (bit 0) corresponding to TCRT –1,0; the next bit (bit 1) corresponding to TCRT –1,1, etc.

16.2 EMULATING FEATURES NOT AVAILABLE ON YOUR TERMINAL

If a given terminal does not support a certain capability, the normal course of action is to simply omit the feature, adjusting the capabilities flags and TCRT bitmap accordingly. While this should be sufficient to alert applications software to the lack of the feature, there are times when the results acheived by this are not fully satisfactory.

An example of such a situation is where a given application program relies on drawing boxes to segment different types of data. A terminal which does not support the box drawing TCRT calls would most likely just cause the application program to draw the boxes with the alternate character set, making the application fully functional, albeit with slightly reduced display speed. A terminal with neither box drawing commands or an alternate character set would cause more

difficulty. In this case, there is no good way for the applications program to separate the different data elements.

In such a situation, it makes more sense for the terminal driver to offer a solution, rather than placing an additional burden on the application program. In this case, the problem is easily solved by substituting appropriate characters from the normal character set for the missing alternate character set. While using dashes and pluses instead of line drawing symbols may not be as attractive, the application program can be used without confusion.

This technique of substituting normal characters for missing alternate characters is one of several different techniques that can be used to emulate missing terminal features within a terminal driver. Other possible techniques include:

- Substituting one feature for another
- Emulation by combining features
- Full emulation of non-existent feature

16.2.1 Substituting One Feature for Another

Particularly in the area of terminal attributes, it may be possible to substitute an existing terminal feature for one that is missing. For example, if a specific terminal does not support the underscore attribute, it may be desirable to replace it with a different, but appropriate, attribute which the terminal does have. Care must be taken in this area, however, as the visual results with an arbritrary applications program are impossible to predict and may be neither visually pleasing or appropriate. For this reason, this type of substitution is not recommended except is special circumstances where the application software to be used is known in advance.

Specifically, since applications software may rely on the use of an attribute for a specific visual effect, attributes should not be substituted or changed if they are all present.

16.2.2 Emulation by Combining Features

In some circumstances it may be possible to combine two or more features a terminal does have to emulate one feature it does not have. For example, a "mode" terminal which does not require a screen position to select an attribute can emulate the attribute TCRT calls which do occupy a screen position by simply outputting a space along with the attribute selection.

Remember that the correct order for this emulation is to output the space and then the attribute start command, and the attribute end command and then the space. This will make the visual results match those of a "field" terminal.

16.2.3∞Full Emulation of Non-Existent Features

Beyond the simple emulations described above are a host of full blown emulations which attempt to simulate advanced terminal features as best they can. In most cases, there is little advantage to doing so, as the application software should be written to adapt itself to the missing features.

In some circumstances, such as where a terminal is missing some fundamental feature or where a particular application program does not properly adapt itself to a missing feature (or does not bother to check to see if the feature is available), it may be desirable to attempt emulation.

Such emulations range from the relatively simple (clear to end of line or clear to end of screen) to the more complex (box drawing and area fill commands) to the very complex (emulating an addressable status line on a terminal with only a non-addressable status line).

Because such emulations vary widely in their implementation due to differences in the particular terminal in use, this section will not detail the specific methods that can be used. Such emulations are typically more trouble than they are worth, and mention is made of them only for completeness and to offer them as a possible solution when all else fails.

Remember that characters output to terminal from within the TCRT routine are not immediately sent to the terminal, but are buffered along with all other output characters. These characters will eventually be passed through the terminal driver output routine and on to the terminal. This delay in the processing of output characters can be important when emulating certain features which must process both TCRT calls and subsequent output characters.

If a TCRT call requires following address characters, such as addressable status lines or the box and area fill commands, the following address characters will need to be pulled from the output stream within the terminal driver output routine.

PART VI

REFERENCE MATERIALS

APPENDIX A

ASCII CHARACTER CHART

The next few pages contain a chart listing the complete ASCII character set. We provide the octal, decimal and hexadecimal representations of the ASCII values. Note that the first 32 characters are non-printing control characters.

THE CONTROL CHARACTERS

CHARACTER	OCTAL	DECIMAL	HEX	MEANING
NULL	000	0	00	Null (fill character)
SOH	001	1	01	Start of Heading
STX	002	2	02	Start of Text
ETX	003	3	03	End of Text
ECT	004	4	04	End of Transmission
ENQ	005	5	05	Enquery
ACK	006	6	06	Acknowledge
BEL	007	7	07	Bell code
BS	010	8	08	Back Space
HT	011	9	09	Horizontal Tab
LF	012	10	0A	Line Feed
VT	013	11	0B	Vertical Tab
FF	014	12	0C	Form Feed
CR	015	13	0D	Carriage Return
SO	016	14	0E	Shift Out
SI	017	15	0F	Shift In
DLE	020	16	10	Data Link Escape
DC1	021	17	11	Device Control 1
DC2	022	18	12	Device Control 2
DC3	023	19	13	Device Control 3
DC4	024	20	14	Device Control 4
NAK	025	21	15	Negative Acknowledge
SYN	026	22	16	Synchronous Idle
ETB	027	23	17	End of Transmission Blocks
CAN	030	24	18	Cancel
EM	031	25	19	End of Medium

PRINTING CHARACTERS

CHARACTER	OCTAL	DECIMAL	HEX	MEANING
SS	032	26	1A	Special Sequence
ESC	033	27	1B	Escape
FS	034	28	1C	File Separator
GS	035	29	1D	Group Separator
RS	036	30	1E	Record Separator
US	037	31	1F	Unit Separator
SP	040	32	20	Space
!	041	33	21	Exclamation Mark
"	042	34	22	Quotation Mark
#	043	35	23	Number Sign
$	044	36	24	Dollar Sign
%	045	37	25	Percent Sign
&	046	38	26	Ampersand
’	047	39	27	Apostrophe
(	050	40	28	Opening Parenthesis
)	051	41	29	Closing Parenthesis
*	052	42	2A	Asterisk
+	053	43	2B	Plus
,	054	44	2C	Comma
-	055	45	2D	Hyphen or Minus
.	056	46	2E	Period
/	057	47	2F	Slash
0	060	48	30	Zero
1	061	49	31	One
2	062	50	32	Two
3	063	51	33	Three
4	064	52	34	Four
5	065	53	35	Five
6	066	54	36	Six
7	067	55	37	Seven
8	070	56	38	Eight
9	071	57	39	Nine
:	072	58	3A	Colon
;	073	59	3B	Semicolon
<	074	60	3C	Less Than
=	075	61	3D	Equal Sign
>	076	62	3E	Greater Than
?	077	63	3F	Question Mark
@	100	64	40	Commercial At
A	101	65	41	Upper Case Letter
B	102	66	42	Upper Case Letter
C	103	67	43	Upper Case Letter
D	104	68	44	Upper Case Letter
E	105	69	45	Upper Case Letter
F	106	70	46	Upper Case Letter
G	107	71	47	Upper Case Letter
H	110	72	48	Upper Case Letter
I	111	73	49	Upper Case Letter
J	112	74	4A	Upper Case Letter
K	113	75	4B	Upper Case Letter
L	114	76	4C	Upper Case Letter

CHARACTER	OCTAL	DECIMAL	HEX	MEANING
M	115	77	4D	Upper Case Letter
N	116	78	4E	Upper Case Letter
O	117	79	4F	Upper Case Letter
P	120	80	50	Upper Case Letter
Q	121	81	51	Upper Case Letter
R	122	82	52	Upper Case Letter
S	123	83	53	Upper Case Letter
T	124	84	54	Upper Case Letter
U	125	85	55	Upper Case Letter
V	126	86	56	Upper Case Letter
W	127	87	57	Upper Case Letter
X	130	88	58	Upper Case Letter
Y	131	89	59	Upper Case Letter
Z	132	90	5A	Upper Case Letter
[	133	91	5B	Opening Bracket
\	134	92	5C	Back Slash
]	135	93	5D	Closing Bracket
^	136	94	5E	Circumflex
_	137	95	5F	Underline
‘	140	96	60	Grave Accent
a	141	97	61	Lower Case Letter
b	142	98	62	Lower Case Letter
c	143	99	63	Lower Case Letter
d	144	100	64	Lower Case Letter
e	145	101	65	Lower Case Letter
f	146	102	66	Lower Case Letter
g	147	103	67	Lower Case Letter
h	150	104	68	Lower Case Letter
i	151	105	69	Lower Case Letter
j	152	106	6A	Lower Case Letter
k	153	107	6B	Lower Case Letter
l	154	108	6C	Lower Case Letter
m	155	109	6D	Lower Case Letter
n	156	110	6E	Lower Case Letter
o	157	111	6F	Lower Case Letter
p	160	112	70	Lower Case Letter
q	161	113	71	Lower Case Letter
r	162	114	72	Lower Case Letter
s	163	115	73	Lower Case Letter
t	164	116	74	Lower Case Letter
u	165	117	75	Lower Case Letter
v	166	118	76	Lower Case Letter
w	167	119	77	Lower Case Letter
x	170	120	78	Lower Case Letter
y	171	121	79	Lower Case Letter
z	172	122	7A	Lower Case Letter
{	173	123	7B	Opening Brace
|	174	124	7C	Vertical Line
}	175	125	7D	Closing Brace
~	176	126	7E	Tilde
DEL	177	127	7F	Delete

APPENDIX B

CODE SUMMARIES

This appendix contains various tables and charts summarizing the various code sequences required by some terminal commands.

Table B-1
Row and Column Codes

r = Row number c = Column number

r or c	ASCII Character	r or c	ASCII Character	r or c	ASCII Character
1	(space)	45	L	89	x
2	!	46	M	90	y
3	"	47	N	91	z
4	#	48	O	92	{
5	$	49	P	93	|
6	%	50	Q	94	}
7	&	51	R	95	~
8	’	52	S	96	DEL
9	(	53	T	97	^Y (space)
10	)	54	U	98	^Y !
11	*	55	V	99	^Y "
12	+	56	W	100	^Y #
13	,	57	X	101	^Y $
14	-	58	Y	102	^Y %
15	.	59	Z	103	^Y &
16	/	60	[	104	^Y ’
17	0	61	/	105	^Y (
18	1	62	]	106	^Y)
19	2	63	^	107	^Y *
20	3	64	_	108	^Y +
21	4	65	‘	109	^Y ,
22	5	66	a	110	^Y -
23	6	67	b	111	^Y .
24	7	68	c	112	^Y /
25	8	69	d	113	^Y 0
26	9	70	e	114	^Y 1
27	:	71	f	115	^Y 2
28	;	72	g	116	^Y 3
29	<	73	h	117	^Y 4
30	=	74	i	118	^Y 5
31	>	75	j	119	^Y 6
32	?	76	k	120	^Y 7
33	@	77	l	121	^Y 8
34	A	78	m	122	^Y 9
35	B	79	n	123	^Y :
36	C	80	o	124	^Y ;
37	D	81	p	125	^Y <
38	E	82	q	126	^Y =
39	F	83	r	127	^Y >
40	G	84	s	128	^Y ?
41	H	85	t	129	^Y @
42	I	86	u	130	^Y A
43	J	87	v	131	^Y B
44	K	88	w	132	^Y C

Table B-2
Status Line Attribute Codes

The codes shown below are embedded in the display text to achieve the indicated results.

NOTE: 1E (hex) turns off dim mode
1F (hex) turns on dim mode

Attribute Code	Screen Attribute Effect
Control-A	Normal
Control-C	Blink
Control-E	Reverse
Control-G	Reverse and Blink
Control-I	Underscore
Control-K	Underscore and Blink
Control-M	Underscore and Reverse
Control-O	Underscore, Reverse, and Blink

Table B-3
Color Selection Codes

The codes shown below are used with the foreground color and background color selection commands.

Color Code	Resulting Color
0	Black
1	White
2	Blue
3	Magenta
4	Red
5	Yellow
6	Green
7	Cyan

Only the first eight primary colors are shown here. Additional values are variations on these colors following a scheme allowing automatic mapping to the nearest color. Details on this scheme can be found in Chapter 8 and in the tables of Appendix C.

Table B-4
Alternate Character Set Commands

Note: These command are only valid after an Enter Alternate Character Set command (TCRT –1,23) has been issued.

Command Code	Symbol Displayed
–1,38	Top left corner
–1,39	Top right corner
–1,40	Bottom left corner
–1,41	Bottom right corner
–1,42	Top intersection
–1,43	Right intersection
–1,44	Left intersection
–1,45	Bottom intersection
–1,46	Horizontal line
–1,47	Vertical line
–1,48	Center intersection
–1,49	Solid block
–1,50	Slanted line block
–1,51	Cross-hatch block
–1,52	Double horizontal line
–1,53	Double vertical line
–1,49	Solid block
–1,50	Slanted line block
–1,51	Cross-hatch block
–1,64	Up arrow
–1,65	Down arrow
–1,66	Raised dot
–1,67	End of line marker
–1,68	Horizontal tab symbol
–1,69	Paragraph symbol
–1,70	Dagger symbol
–1,71	Section symbol
–1,72	Cent sign
–1,73	One-quarter symbol
–1,74	One-half symbol
–1,75	Degree symbol
–1,76	Trademark symbol
–1,77	Copyright symbol
–1,78	Registered symbol

Table B-5
Horizontal and Vertical Size Codes

v = vertical size in rows h = horizontal size in columns

v or h	ASCII Character	v or h	ASCII Character	v or h	ASCII Character
1	!	45	M	89	y
2	"	46	N	90	z
3	#	47	O	91	{
4	$	48	P	92	|
5	%	49	Q	93	}
6	&	50	R	94	~
7	’	51	S	95	DEL
8	(	52	T	96	^Y !
9	)	53	U	97	^Y "
10	*	54	V	98	^Y #
11	+	55	W	99	^Y $
12	,	56	X	100	^Y %
13	-	57	Y	101	^Y &
14	.	58	Z	102	^Y ’
15	/	59	[	103	^Y (
16	0	60	/	104	^Y)
17	1	61	]	105	^Y *
18	2	62	^	106	^Y +
19	3	63	_	107	^Y ,
20	4	64	‘	108	^Y -
21	5	65	a	109	^Y .
22	6	66	b	110	^Y /
23	7	67	c	111	^Y 0
24	8	68	d	112	^Y 1
25	9	69	e	113	^Y 2
26	:	70	f	114	^Y 3
27	;	71	g	115	^Y 4
28	<	72	h	116	^Y 5
29	=	73	i	117	^Y 6
30	>	74	j	118	^Y 7
31	?	75	k	119	^Y 8
32	@	76	l	120	^Y 9
33	A	77	m	121	^Y :
34	B	78	n	122	^Y ;
35	C	79	o	123	^Y <
36	D	80	p	124	^Y =
37	E	81	q	125	^Y >
38	F	82	r	126	^Y ?
39	G	83	s	127	^Y @
40	H	84	t	128	^Y A
41	I	85	u	129	^Y B
42	J	86	v	130	^Y C
43	K	87	w	131	^Y D
44	L	88	x	132	^Y E

Table B-6
Fill Area Attribute Codes

The codes shown below are used with the Fill Area with Attributes command to set the appropriate attribute.

Attribute Code	Screen Attribute Effect
0	Normal
1	Blank (no display)
2	Blink
4	Reverse
6	Reverse and blink
8	Underscore
:	Underscore and blink
<	Underscore and reverse
>	Underscore, reverse, and blink
p	Dim
r	Dim and blink
t	Dim and reverse
v	Dim, reverse, and blink
x	Dim and underscore
z	Dim, underscore, and blink
|	Dim, underscore, and reverse
~	Dim, underscore, reverse, and blink

APPENDIX C

THE ALPHA MICRO COLOR MAP

This appendix lists the first 64 colors contained in the Alpha Micro color map, along with the RGB and HLS representations of those colors. The RGB value is given as the relative value assigned to each of the three primaries, with 1.00 as the maximum value.

The HLS values are given in standard HLS notation. (See Chapter 5 for more information on the HLS system.)

The text color description is an attempt to describe the color in descriptive terms. As with any descriptive system, its fault lies in the many variables that may affect the reproduction of a particular color, such as display brightness, ambient light, etc. These descriptions are intended as a guide only.

Table C-1
The Alpha Micro Color Map

Color Number	R	G	B	H	L	S	Description
0	0.00	0.00	0.00	0	0.000	0.000	Black
1	1.00	1.00	1.00	0	1.000	0.000	White
2	0.00	0.00	1.00	0	0.500	1.000	Blue
3	1.00	0.00	1.00	60	0.500	1.000	Magenta
4	1.00	0.00	0.00	120	0.500	1.000	Red
5	1.00	1.00	0.00	180	0.500	1.000	Yellow
6	0.00	1.00	0.00	240	0.500	1.000	Green
7	0.00	1.00	1.00	300	0.500	1.000	Cyan
8	0.33	0.33	0.33	0	0.333	0.000	Charcoal gray
9	0.67	0.67	0.67	0	0.667	0.000	Light gray
10	0.00	0.00	0.67	0	0.334	1.000	Deep blue
11	0.67	0.00	0.67	60	0.334	1.000	Deep magenta
12	0.67	0.00	0.00	120	0.334	1.000	Deep red
13	0.67	0.67	0.00	180	0.334	1.000	Yellow brown
14	0.00	0.67	0.00	240	0.334	1.000	Deep green
15	0.00	0.67	0.67	300	0.334	1.000	Light blue
16	0.33	0.33	0.67	0	0.500	0.334	Violet blue
17	0.67	0.67	1.00	0	0.834	0.200	Light blue 2
18	0.00	0.00	0.33	0	0.167	1.000	Navy blue
19	0.33	0.00	0.33	60	0.167	1.000	Violet
20	0.33	0.00	0.00	120	0.167	1.000	Brick red
21	0.33	0.33	0.00	180	0.167	1.000	Yellow green
22	0.00	0.33	0.00	240	0.167	1.000	Forest green
23	0.00	0.33	0.33	300	0.167	1.000	Blue gray 1
24	0.33	0.67	0.33	240	0.500	0.334	Blue green
25	0.67	1.00	0.67	240	0.834	0.200	Very light blue
26	0.33	0.00	1.00	20	0.500	1.000	Medium blue
27	0.67	0.00	1.00	40	0.500	1.000	Purple
28	1.00	0.00	0.33	100	0.500	1.000	Hot pink
29	0.67	1.00	0.00	200	0.500	1.000	Chartreuse
30	0.00	1.00	0.33	260	0.500	1.000	Foam green
31	0.00	1.00	0.67	280	0.500	1.000	Pea green
32	0.67	0.33	0.33	120	0.500	0.334	Pale burgundy
33	0.33	0.67	0.67	300	0.500	0.334	Blue gray 2
34	0.00	0.33	1.00	340	0.500	1.000	Hot blue
35	1.00	0.00	0.67	80	0.500	1.000	Pink
36	1.00	0.33	0.00	140	0.500	1.000	Red orange
37	1.00	0.67	0.00	160	0.500	1.000	Amber
38	0.33	1.00	0.00	220	0.500	1.000	Light green 1
39	0.00	0.67	1.00	320	0.500	1.000	Sky blue
40	1.00	0.67	0.67	120	0.834	0.200	Purple
41	0.67	1.00	1.00	300	0.834	0.200	Light cyan 2
42	0.33	0.33	1.00	0	0.667	0.500	Faded blue
43	1.00	0.33	1.00	60	0.667	0.500	Pale purple
44	1.00	0.33	0.33	120	0.667	0.500	Pale hot pink
45	1.00	1.00	0.33	180	0.667	0.500	Pale yellow
46	0.33	1.00	0.33	240	0.667	0.500	Sea foam green

Table C-1 (cont'd)
The Alpha Micro Color Map

Color Number	R	G	B	H	L	S	Description
47	0.33	1.00	1.00	300	0.667	0.500	Light cyan 1
48	0.67	0.33	0.67	60	0.500	0.334	Purple gray
49	1.00	0.67	1.00	60	0.834	0.200	Light magenta
50	0.33	0.00	0.67	30	0.334	1.000	Indigo
51	0.67	0.33	1.00	30	0.667	0.500	Light purple gray
52	0.67	0.00	0.33	90	0.334	1.000	Burgundy
53	0.67	1.00	0.33	210	0.667	0.500	Gray green
54	0.00	0.67	0.33	270	0.334	1.000	Faded blue green
55	0.33	1.00	0.67	270	0.667	0.500	Pale cyan
56	0.67	0.67	0.33	180	0.500	0.334	Tan
57	1.00	1.00	0.67	180	0.834	0.200	Cream
58	0.00	0.33	0.67	330	0.334	1.000	Dull blue
59	1.00	0.33	0.67	90	0.667	0.500	Purple pink
60	0.67	0.33	0.00	150	0.334	1.000	Orange brown
61	1.00	0.67	0.33	150	0.667	0.500	Light brown
62	0.33	0.67	0.00	210	0.334	1.000	Khaki green
63	0.33	0.67	1.00	330	0.667	0.500	Light blue 1

APPENDIX D

FUNCTION KEY TRANSLATION TABLES

AMOS provides support for programmed function keys in a very generalized fashion, imposing no fixed structure or method of implementation onto the programmer; you are free to map a function key into its result in any fashion you wish.

However, AMOS does implement two different, standardized methods of performing function key translation. This appendix describes the format of the internal tables used by these two methods.

D.1 STANDARD FUNCTION KEY TRANSLATION TABLES

This section describes the internal format of the function key translation tables created by the FIXTRN program and used by such application programs as AlphaVUE, AlphaWRITE, AlphaCALC, and MULTI.

All offsets are from the base of the file, not including any file system links.

Bytes 0-3	Four bytes of zero
Bytes 4-515	A table of 256 word offsets. This table is indexed by the function key code (in the range 0-255) and yields an offset (from the base of the file) to the characters the function key is to be translated to.
Bytes 516-nnnnn	A series of null terminated strings which are pointed to by the offset table.

The sequence of events used to translate a function key is as follows:

1. Take the function key code to be translated and multiply it by two.
2. Add four to the result.
3. Use the result as an index into the translation file to retrieve an offset word.
4. Use this offset word as an index into the translation file. This index will point to the characters to be substituted for the function key.

Note that a function key can have no translation at all (if the byte it indexes is zero), a single character substitution, or an arbitrarily long substitution.

Any character can be translated via a function key translation table, not just those commonly referred to as "function keys."

D.2 SET PFK STYLE TRANSLATION TABLES

A somewhat different style of function key translation table is used to implement the SET PFK function of AMOS. Because these translation tables are loaded into memory at all times, the format of the table has been optimized to reduce memory requirements, at a slight increase in the amount of time required to perform the translation.

The format of a SET PFK type translation table is as follows:

Bytes 0-1	Two bytes containing the octal value 121212, which flag this as a valid translation table.
Bytes 2-nnnnn	A series of entries consisting of a single byte containing a function key to be translated, followed by a null terminated series of bytes to which the function key is to be translated.
Byte nnnnn+1	A zero byte used to terminate the whole file.

The sequence of events used to translate a function key is as follows:

1. Start by indexing the table at byte 2, just past the flag word.
2. If the byte being indexed contains a zero, you have reached the end of the table and there is no translation defined for this function key.
3. If the byte being indexed contains the function key you wish to translate, proceed to step 5.
4. Scan forward in the file until a zero byte is seen. Index the byte following the zero and return to step 2.
5. You are now indexing a null terminated string containing the characters to which the function key is to be translated.

Any character except null can be translated via a SET PFK type function key translation table, not just those commonly referred to as "function keys."

APPENDIX E

GLOSSARY

ASCII code - The acronym for American Standard Code for Information Interchange. This standardized code is used extensively in data transmission, especially to terminals. The code includes 128 upper and lower case letters, numerals, and special purpose symbols each encoded by a unique 7-bit binary number.

attribute - A particular display enhancement that can be applied to any character displayed on a terminal. Typical enhancements include underscoring, reverse video, dimmed video, and blinking.

baud rate - Synonymous with signal events (bits)-per-second and used as a measure of serial data flow between a computer and/or communications devices.

character attribute - A display enhancement that applies to individual characters. If the character is replaced with another, the attribute is not preserved. A character attribute is independent of terminal screen location but is instead associated with the character itself.

control character - A character whose occurance in a particular context initiates, modifies, or halts operation. Control codes are assigned ASCII character codes below the space character.

cursor - A small moving indicator on a CRT terminal which indicates the position where the next character will be output. The cursor can be displayed in a variety of shapes including small rectangular blocks or an underline. It may blink or not, or may be turned off entirely.

field attribute - A display enhancement that applies to a particular area of the terminal screen; in particular, the area before "begin attribute" and "end attribute" markers.

full duplex - Refers to a communications channel which can simultaneously and independently transmit and receive data. Most modern terminals are full duplex, in that data can be transmitted from the keyboard while it is being received by the screen display.

half duplex - Refers to a communications channel which can both transmit and receive, but not simultaneously.

hidden attribute - A display enhancement that does not occupy a terminal screen position. Hidden attributes are most often associated with character attributes.

interface board - A hardware device that does the actual data transfer from the computer to the *terminal*.

interface driver - A machine language program that transfers data back and forth between the *interface board* and the *terminal*.

job - The structure AMOS uses to allocate time a resources to a user. In order to make use of AMOS, you must have a job.

job control block (JCB) - A data structure allocated by AMOS for each *job*, maintaining specific information about that job.

monochrome terminal - A CRT *terminal* which displays text in a single color only. This color can be vary from green, to white, to amber.

reverse video - A special terminal display *attribute* that can be used to highlight text on a terminal screen. On a monochrome terminal that usually has a black background, text displayed in reverse video will be displayed as black text on a light background. On a terminal that usually has a light background, the effect is reversed. On a color terminal, reverse video will simply exchange the background and foreground colors.

serial IO - A method of data transfer between a computer and a peripheral device (such as a terminal) in which data is transmitted bit by bit over a single circuit.

terminal - The primary device used to communicate between the computer and yourself. Especially a CRT terminal or video display terminal with a keyboard and television-like screen, or a hard copy terminal with a keyboard and associated printer.

terminal control block (TCB) - The data structure within the *terminal service system* which is used to maintain the current status of a terminal, as well as its input and output buffers. Each terminal defined on a system has a terminal control block associated with it, whether it is attached to a *job* or not.

terminal driver - A machine language program which translates data from internal formats to a format understood by a particular terminal.

terminal service system - The part of AMOS responsible for coordinating and handling all serial IO for the system. Among its responsibilites are the buffering of both input and output characters, scheduling of jobs as they enter and leave terminal IO wait states, and servicing of all user requests for terminal input and output.

TRMSER - an acronym for the *terminal service system* contained within AMOS. The terminal service system is responsible for coordinating and handling all serial IO for the system.

INDEX