THE USE OF ALPHANUMERIC DISPLAY TERMINALS

A thesis presented in partial fulfilment of the requirements for the degree of Master of Arts in Computer Science at Massey University

Patrick John Barrer

1975

Acknowledgements

I should like to thank my supervisors, Bob Doran and Professor Tate, for their help and encouragement. Bob Doran originated the Genisys project, designed the General language and implemented Genisys's data structures.

I should also like to thank Chris Freyberg for his valuable help in interfacing the VT05 to the B6700 and Ted Drawneck for reading my thesis and for his helpful comments.

Abstract

A functional description of alphanumeric display terminals is given and then the four main classes of interactive systems that use alphanumeric display terminals; enquiry/response systems, interactive programming systems, data entry and computer-aided instruction are briefly described.

Some programming considerations where using alphanumeric display terminals with interactive systems are mentioned and then a "virtual" alphanumeric display terminal is introduced.

Finally as an example, the program entry phase of the Genisys system is described.

Preliminary Note

Throughout this thesis the term alphanumeric display terminal will from time to time be abbreviated to ADT.

TABLE OF CONTENTS

		Page
	Acknowledgements	i
	Abstract	ii
	Preliminary Note	iii
CHAPTER 1	A Functional Description of Alphanumeric Display Terminals	1
contents:	The Processor-Terminal Connection	1
	The Components of a typical ADT	3
	The Transmit and Receive Interface	3
	The Display Screen	4
	The Memory	6
	The Character Generating Subsystem	6
	The Keyboard	7
	The Control Logic	8
	Operating Modes	9
	Full-Duplex and Half-Duplex Modes	9
	Block and Conversational Modes	10
	Receive and Transmit Modes	11
	Additional Control Functions	12
	Scrolling and Paging	12
	Editing Capability	13
	Protected Fields	14
CHAPTER 2	Interactive Systems which use Alphanumeric Display Terminals	16
contents:	Enquiry/Response Systems	18
	Airline Reservation Systems	19
	The Medical Information System at El Camino Hospital, California	19
	Interactive Programming Systems	21
	CANDE, an example of a general purpose editor	22
	Single Language Systems	24
	The Interactive Programming Support System	25
	Emily	26

Page

The Automated Computer Science Education System 27

Data Entry 28

Conventional Data Entry Using ADTs 28

Form Filling 29

Computer Aided Instruction 30

The COPI system 31

The Socratic approach 32

CHAPTER 3 Some Programming Considerations when using ADTs with Interactive Systems 33

contents:

Dialogue Design 33

Passive Systems 34

Active Systems 35

Screen Management 37

Use limitations 37

Size limitations 38

Operating Modes 41

Conversational full-duplex mode 41

Half-duplex block mode 42

Conversational half-duplex mode 42

Hardware Requirements 42

CHAPTER 4 Virtual Alphanumeric display terminals 46

contents:

Some reasons for implementing virtual terminals 46

An example - a software interface for the DEC VT05 ADT 48

CHAPTER 5 An Example - Constructing Programs with Genisys 51

contents:

Screen Management 56

Dialogue Design 57

Line Entry 59

Using Genisys 59

Summing up Genisys 64

Page

APPENDIX I Summary of ADT characteristics 65

APPENDIX II Survey of ADTs available in New Zealand 67

APPENDIX III The Digital Equipment Corporations VT05 ADT 80

APPENDIX IV General's syntax 82

APPENDIX V Description of Buildprogram 83

Bibliography and References 93

List of Figures

		Between Pages
Figure 1.1	The ASCII 67 character code	2-3
Figure 1.2	The main components of a typical ADT	3-4
Figure 1.3	Some examples of dot matrix characters	5-6
Figure 3.1	A hypothetical menu	36-37
Figure 3.2		36-37
Figure 5.1		56-57
Figure 5.2		56-57
Figure 5.3		60-61
Figure 5.4		60-61
Figure 5.5		60-61
Figure 5.6		60-61
Figure 5.7		60-61
Figure 5.8		60-61
Figure 5.9		60-61
Figure 5.10		60-61
Figure 5.11		61-62
Figure 5.12		61-62
Figure 5.13		62-63
Figure 5.14		62-63
Figure 5.15		62-63
Figure 5.16		62-63
Figure 5.17		63-64
Figure 5.18		63-64
Figure 5.19		63-64
Figure 5.20		63-64

CHAPTER 1

A Functional Description of Alphanumeric Display Terminals

An alphanumeric display terminal is one of many devices that allow a person to communicate with a computing system. Typically such devices have a keyboard which the operator may use to enter messages and a screen where alphanumeric characters may be displayed.

More and more ADTs are being used with computing systems, and a large variety of different models is now available; indeed, Auerbach's "Guide to Alphanumeric Display Terminals" lists more than two hundred and fifty different terminals [Datamation May 1975, p30.]

This chapter described some of the basic features of these terminals. The discussion takes place under the following headings: the Processor-Terminal Connection; the Components of a typical ADT; Operating Modes; Additional Control Functions.

The Processor-Terminal Connection

Although most alphanumeric display terminals can be operated off-line, their basic use is to transfer information between the human operator and the computer. To accomplish this a terminal has to be physically linked to the computing system. This section is concerned with this link.

A terminal and processor communicate using characters. A character is a unique piece of information that can be represented in different ways. On the screen of the terminal the character is represented by a single graphic symbol but when held in the memory of the terminal and of the computer a character is represented as a set of bits. Because the physical lines which connect the terminal to the processor are, like the elements of memory, best limited to two states, characters transferred between processor and terminal are encoded as sets of bits. Since terminals are

$b_7 b_6 b_5$ \ $b_4 b_3 b_2 b_1$	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
0 0 0 0	NUL	DLE	SP	0	@	P	`	p
0 0 0 1	SOH	DC 1	!	1	A	Q	a	q
0 0 1 0	STX	DC2	"	2	B	R	b	r
0 0 1 1	ETX	DC3	#	3	C	S	c	s
0 1 0 0	EOT	DC4	$	4	D	T	d	t
0 1 0 1	ENQ	NAK	%	5	E	U	e	u
0 1 1 0	ACK	SYN	&	6	F	V	f	v
0 1 1 1	BEL	ETB	'	7	G	W	g	w
1 0 0 0	BS	CAN	(	8	H	X	h	x
1 0 0 1	HT	EM	)	9	I	Y	i	y
1 0 1 0	LF	SUB	×	:	J	Z	j	z
1 0 1 1	VT	ESC	+	;	K	[	k	{
1 1 0 0	FF	FS	,	<	L	\	l	|
1 1 0 1	CR	GS	−	=	M	]	m	}
1 1 1 0	SO	RS	.	>	N	^	n	~
1 1 1 1	SI	US	/	?	O	_	o	DEL

Fig1.1 ASCII 67 character code

ASCII Control characters

NUL	Null	DLE	Data Link Escape
SOH	Start of Heading	DC 1	Device Control 1
STX	Start of Text	DC 2	Device Control 2
ETX	End of Text	DC 3	Device Control 3
EOT	End of Transmission	DC 4	Device Control 4
ENQ	Enquiry	NAK	Negative Acknowledge
ACK	Acknowledge	SYN	Synchronous Idle
BEL	Bell (audible signal)	ETB	End of Transmission Block
BS	Backspace	CAN	Cancel
HT	Horizontal Tab	EM	End of Medium
LF	Line Feed	SUB	Substitute
VT	Vertifcal Tab	ESC	Escape
FF	Form Feed	FS	File Separator
CR	Carriage Return	GS	Group Separator
SO	Shift Out	RS	Record Separator
SI	Shift In	US	Unit Separator
		DEL	Rubout

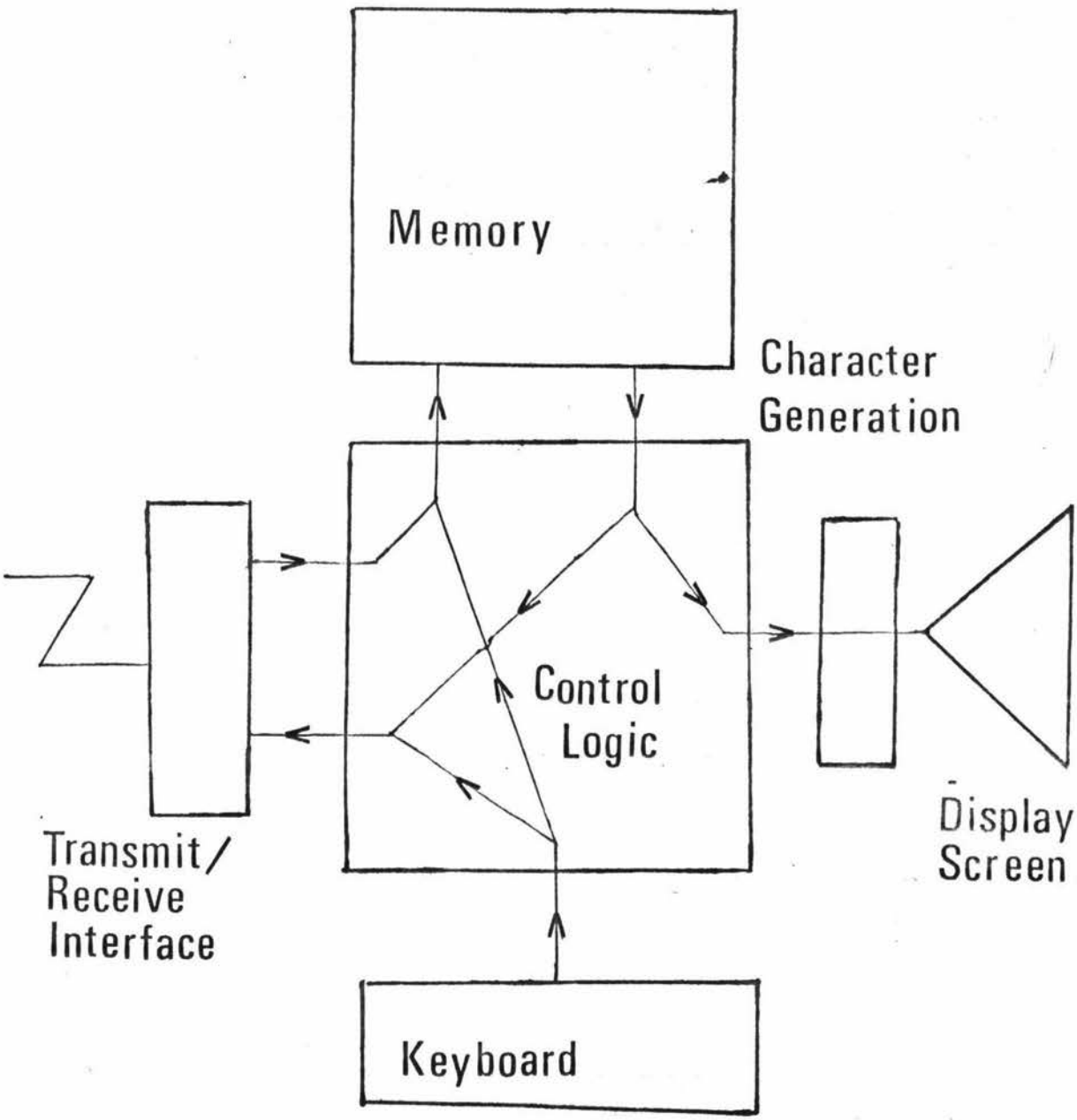
figure 1.1 (continued)

designed for use with more than one computer, a standard code, the American Standard Code for the Interchange of Information 1967 (ASCII 67) is frequently used to represent the characters. This is a seven bit code which therefore allows one hundred and twenty-eight different characters.

The transfer of information between terminal and processor is always in the form of a message sent one character at a time. The link on which the message is sent may take two forms. The first provides direct data paths for each bit in the character representation, each bit thereby being transmitted simultaneously. With such a link, data can be transferred at speeds that can approach two million characters per second. More commonly, however, only one data path is provided by the link. The characters must be transmitted serially; that is, bit by bit. This is done at switch-selectable rates generally ranging from one hundred and ten bits per second (or about ten characters per second) to a maximum of nine thousand six hundred bits per second (or about a thousand characters per second). Some newer terminals now offer twice this rate. The lower rates are provided only to give plug-in compatability with teletypes, that is, to use the lower grade teletype lines.

Serial characters are transmitted either synchronously or asynchronously. Synchronous transmission means that the characters comprising the message are transmitted at regular intervals. The message is preceded by a "header", a special set of characters which describe the following message, the last character of the message is a special end of text character. Asynchronous transmission means that the characters can be sent at either irregular intervals or at regular intervals, each character preceded by a special start bit (or bits) to signal the start of that character, and followed by a stop bit to signal the end of the character.

Data communication lines can be either half-duplex or full-duplex. Half-duplex lines allow transmission only in one direction at one time. Full-duplex lines allow



Main Components of an ADT

simultaneous transmission in both directions.

Half-duplex lines have problems of traffic control since traffic can be allowed in only one direction. Further problems of traffic control arise when terminals, as often happens, are forced to share a line. These problems are resolved by the application of line disciplines, predetermined methods of "rationing" line use to allow the orderly transference of messages. Two brief examples may be given. "Polling" is a line discipline in which the processor polls each terminal in some predetermined pattern to see if there is a message ready for transmission from it. If no terminal has such a message then the processor can transmit to any or all terminals any messages it has. "Contention" is a line discipline in which the terminal and processor contend for the line, each transmitting its messages at regular but different intervals and continuing until acknowledgement has been received from the other end of the line that the message has been received.

The Components of a typical ADT

An alphanumeric display terminal may be viewed as consisting of six basic components. These are the display screen, the memory, the character generating system, the transmit and receive interface and the control logic. Each of these components provides a subheading for this section in which they are further described.

The Transmit and Receive Interface

This interface superintends the transfer of information between the computer and the terminal. The complexity of the interface depends on the line disciplines under which it is designed to operate. Many are very simple - they transmit from the terminal the appropriate character when any key is depressed or else they transmit a block of characters out of the memory when the transmit key is depressed. Characters received from the computer are transferred to the control logic which enters them into the memory if they are displayable or else responds to them if

they are control characters. Such a terminal is intended to have its own line. Where a terminal must share a line, the terminal must be addressable, that is, the interface must be capable of discriminating among messages to determine those which are intended for it. Messages may have to be acknowledged and complicated dialogues may have to take place before a terminal can transmit a message. For example, where a polling line discipline is operating and a terminal has a message to send, it must first wait till it is polled by the computer, then indicate that a message is ready for transmission, and then send the message. Under a contention line discipline, the terminal may have to retransmit the message at regular intervals until an acknowledgement is received. In general the interfaces usually provided for ADTs do not allow particularly sophisticated line disciplines, although there are exceptions (often provided as extras at an additional cost).

The Display Screen

Two types of display screen have been used with ADTs. In most cases the display screen is a cathode ray tube. In a cathode ray tube an electron gun directs a beam of electrons towards a phosphor-coated screen which glows when struck by the beam.

Three different methods are used to draw the characters on the screen. The electron beam may be forced to pass through a mask in the shape of the desired character which is then projected onto the screen. Another method is to construct the characters from small strokes of the electron beam. Both these methods have largely been superseded by the dot matrix method which although it gives a slightly less satisfactory representation is much simpler to implement. The electron beam moves in a regular fashion across the screen moving from side to side and top to bottom. Although for most of its journey it is turned off, as it passes certain points it may be turned on, illuminating at that point a dot on the screen. Each displayed character is built from a matrix of these dots. For example in a 5x7

matrix (which is the most common arrangement) there are 35 dots from which characters are built (as in fig 1.3).

The number of displayable characters depends to some extent on the size of the matrix. A 5x7 matrix is about the smallest sufficient to display uppercase letters, digits and special characters. Lower-case letters can also be displayed with this matrix if the tails of p,q,g and j can be displayed in the two dot line gap between lines of characters. Matrices with more dots give more pleasing representations of the characters.

Some terminals have the ability to display characters with reversed contrast (that is all dots not normally illuminated are illuminated while the others are blanked), to blink between the normal and reversed representations, to display characters in different character fonts, or to display characters at different intensities or even colours.

At some particular point on the screen the current cursor position will be displayed. The cursor indicates the point at which a newly-entered character would be displayed on the screen. The cursor position is moved by the control logic, which keeps its location in special registers, usually storing the x and y displacements. The actual form the cursor takes when displayed depends on the terminal. It may be a blinking line underscoring the character in that position, a character being alternated with its reverse, or any other distinctive feature. Usually some form of blinking is involved so as to attract the user's attention.

Mention must be also made of a less usual form of screen technology just making its appearance. This is the gas plasma screen which consists of a large matrix of gas cells each of which can be selectively illuminated. Characters are built in the dot matrix fashion described previously. Each gas cell is filled with neon gas which can be made to glow. This effect lasts longer than the momentary glow of a phosphor when an electron strikes it. Because the duration of the illumination can be unpredictable, Burroughs "Selfscan" technology in its pioneering use of gas plasma screens refreshes the display constantly in a similar manner

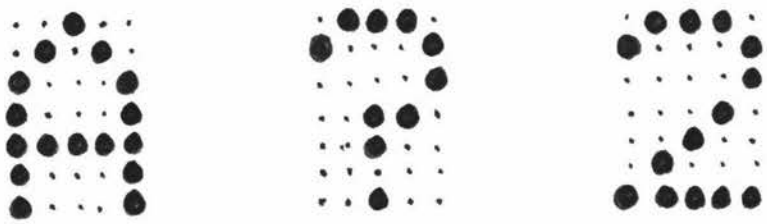


Fig 1.3

Some examples of dot matrix characters

to that used with a cathode ray tube.

Memory

Since the display must be constantly refreshed, each character displayed on the screen must be held in an internal form in permanent storage. This is the main memory of the terminal, sometimes referred to as the buffer. The memory is usually constructed from semi-conductor shift registers since the whole memory is being read serially all the time the device is working. The correspondence between screen and memory is determined in one of two basic ways. Most terminals let each position on the screen correspond to an unique position in the memory. A displayable character (which is often the space code) is always stored there and the corresponding character is displayed on the screen.

The second less common method derives from times when memory was relatively more expensive and sometimes shared between several terminals. A non-spatial memory stores all the character codes, excluding unnecessary space codes but including control character codes, in consecutive locations, the control characters being used to generate the actual position of the displayable characters. Since large portions of the screen are frequently blank, this uses less memory. It can also make some editing features easier.

The same memory may be used to drive several screens or it may be large enough to hold reserve pages for the same screen, each of which can be loaded from the computer and may be used to replace the current display very quickly.

Character Generation

Each character code is fetched at regular intervals of about a fiftieth of second from the main memory and used to refresh the display. The internal code must be changed into the character matrix. The easiest way to do this is to hold the matrix in a read only memory at an address produced from the numerical valve of the code. If the code is ASCII, this is facilitated by the fact that all the displayable characters are in numerical order starting from 20 Hex and differing by 1 - unlike the EBCDIC code for

example where there are gaps and control characters mixed among the displayable characters. The matrix is then used to control the illuminations of the particular dots which make up each character.

Thus, to display a character it is sufficient to enter the character code into the memory and as a result it will be displayed within a fiftieth of a second. Likewise alteration of characters or the cursor position on the screen merely involves the alteration of the memory or the special registers which hold the cursor position.

Keyboard

The keyboard usually resembles that of a teletypewriter, although some manufacturers provide options, usually in the number of control keys available or by the provision of separate numeric pads. The basic action of the keyboard is that when a key is depressed, a character is generated in a special register. This character is usually in the form in which it would be stored in the terminal's memory or transmitted to the processor. Shift and control keys are provided to extend the number of characters generated by the keyboard. In ASCII terminals, keyboards can commonly generate all 128 ASCII characters, the shift and control keys each changing one bit in the final character.

What happens to the character in the register depends on which of the various alternative modes the terminal is operating in. These modes are reviewed in more detail in a later section but their effect will be described briefly. If the terminal is in full-duplex mode, the character is transmitted to the computer. If the terminal is in half-duplex mode, then as well as being transmitted the character is also inserted in memory or if it is a control character, the control logic reacts appropriately. If the terminal is in block mode, the character is not transmitted, but only consumed locally. The reasons for these differing actions are explained later.

As well as keys which generate control and displayable characters, some ADTs also have function keys. Function

keys which are more commonly associated with graphics terminals generate characters which are neither displayable nor cause any action to be taken by the control logic but which, when transmitted to the computer, invoke some programmed response. A common example is the rubout or DEL key specifically provided for in the ASCII character set which can indicate to the program that the previous character is to be ignored. (This is a remnant from teletypes - with an ADT correction is better made by overwriting).

The Control Logic

The control logic performs certain actions to alter memory contents (and hence the display) and cursor position. It does this either automatically on receipt of ordinary, displayable characters from either the keyboard or the processor, or in response to special control characters. This control logic is usually hardwired or at least firm-wired (driven by a micro-program held in read only memory); however, there are some "intelligent" terminals which are programmable.

The basic automatic control function is the insertion of the received character into memory at the point indicated by the cursor and the advance of the cursor one space.

Another commonly provided automatic function is "wraparound" at the screen boundaries. If the cursor is at the end of one line and a new character is to be displayed then the cursor is advanced automatically to the beginning of the next line and the character displayed there. Wraparound may occur at all boundaries or only at the end of lines.

The basic control characters are used to move the cursor, either forward or backward one position, up or down one line, or to the start of the current line (carriage return), or back to the first position on the screen (usually called 'home"). Other necessary control characters cause the erasure of all or just part of the screen.

Less necessary, but still very useful, control functions allow the cursor to move larger distances. Tabbing, where the cursor is moved to the next predetermined position in a line on receipt of the tab control character, is provided on most terminals. The tab positions may be fixed (at multiples of eight character positions for example) or else may be preset by the programmer. Another very useful control feature available on a number of terminals is cursor addressing, where the cursor may be quickly transferred to any other position on the screen. This is generally done by the transmission of three characters, the first a special control character and the other two ordinary characters which are not displayed but whose numerical values give the line and position on that line to which the cursor is to be moved. Less frequently provided but sometimes useful is the ability for the processor to read out the cursor's current address. Some more sophisticated control features are also discussed in a later section in this chapter.

Operating Modes

Since no designers can be sure of the circumstances in which their terminal will operate, they frequently provide switch-selectable alternatives. The unknown factor which they must cater for is the nature of the connection between the terminal and the processor. Three different problems are solved by the provision as options of full-duplex and half-duplex modes, block and conversational modes and receive and transmit modes; and these constitute the subheadings under which each of these alternatives are discussed.

Full-Duplex and Half-Duplex Modes

Full-duplex and half-duplex modes are provided to give a terminal the capability to use both full-duplex and half-duplex lines. The mode is generally selected when the terminal is installed and it is unusual for the mode to be changed after this.

In full-duplex mode the keyboard is detached (logically not physically) from the rest of the terminal and must communicate with the display via the connected processor. This is, when a key is depressed, no character is displayed until it has been "echoed" by the processor. In effect all keys become function keys dependent on a programmed response for any result from the key being depressed. Normally the programmed response will be to "echo" or send back the character received so the user may see what is being typed. Such a system requires the terminal to be connected by a full-duplex line since characters must be sent continually in both directions.

In half-duplex mode the character is immediately entered into the terminal's memory (and so appears on the display) when the key is depressed so a half-duplex line may be used to connect the terminal. A full-duplex line may also be used to connect a terminal in half-duplex mode if it is felt that making the processor echo the characters is an unacceptable overhead or otherwise inconvenient.

Block and Conversational Modes

Unlike half and full duplex modes, block and conversational modes are often not provided as switch selectable alternatives on the same terminal since there is a philosophical disparity between them. Hence the terminal must be selected according to the mode in which it is designed to operate.

Conversational mode terminals (sometimes called character mode terminals) transmit to the processor all characters, even control characters, as they are entered from the keyboard. This requires that they just be connected by an asynchronous line to the processor since the characters will be generated at irregular intervals.

Terminals in block modes, however, do not transmit characters as they are entered but accumulate them in the terminal's memory. When the transmit key is pressed all the characters in the memory (not including any control

characters) between two bounds are transmitted to the processor.

The actual bounds provided on various terminals vary. A good terminal design will let the user set his own with a default if he doesn't. However some terminals provide fixed bounds using the cursor to delimit one end of the message and the start or end of the current line or the screen to delimit the other. In extreme cases both bounds are fixed. Block mode terminals can be used on either synchronous or asynchronous lines (provided the transmit and receive interface allows this) as the message is sent as a whole and so the characters within that message will be sent at regular intervals.

It is worth considering the relation between the block and conversational modes on the one hand and the full and half duplex modes on the other. Clearly a block full duplex mode is an impossibility (as terminals are currently constructed anyway). The fact the terminal is in full-duplex mode means the character will not be entered into the terminal memory when a key is depressed while the fact that it is in block mode means it will not be transmitted to the processor either!

A summary of the possible mode combinations is as follows: conversational full-duplex mode where, when the key is depressed, the character is sent to the processor but not displayed unless echoed; conversational half-duplex mode where the character is both sent to the processor and displayed when the key is depressed; block half-duplex mode where the character is displayed but not sent to the processor when the key is depressed.

Receive and Transmit modes

Half-duplex lines create problems of control of the line traffic because transmission can be in only one direction at one time. So that the processor at one end of the line and the user at the other must coordinate their activity. This means they must have some method of reserving the line for their sole use or letting the other

end know that the line is required. At the most elementary level control is imposed by providing a "break key" for the user which "breaks" the line. This state can be recognised by the processor even if it is currently transmitting data, causing it to stop and receive some input before resuming transmission.

More sophisticated terminals are able to operate in receive or transmit modes. Unlike the modes discussed earlier, these are temporary modes between which the terminal constantly switches. In receive mode the keyboard is locked out while in transmit mode the processor cannot send to the terminal. These modes are either set and reset by control characters transmitted by the processor or by using a switch provided on the terminal or both.

If the processor has sole control over the terminal modes then a break key must still be provided while if the user has sole control some method must be provided for the processor to signal the fact is has a message to send. A "bleeper" serves this purpose admirably.

Additional Control Functions

This section reviews some extra hardware capabilities possessed by a number of terminals and not discussed previously. These have been left to this point because reference is made to operating modes which had not been introduced till this point. They are discussed under the subheadings scrolling and paging, editing capabilities and protected fields.

Scrolling and Paging

A decision which must always be made when using ADTs is what to do if more information must be displayed than can be fitted on the screen. Terminals attempt to deal with this by providing special control functions which are automatically invoked when the cursor is at the bottom of the screen and a new line of data has to be displayed.

Two slightly different approaches exist. The first is scrolling. Scrolling means every line is moved up one position and the top line is lost from the screen and usually from the terminal's memory. The term derives from the illustration commonly used of a scroll being rolled up

at the top and unrolled at the bottom. Scrolling means that new lines are displayed at the bottom of the screen after those previously received. The second approach, paging, is really wraparound in a vertical direction. Paging moves the cursor to the top of the screen where the next line will be displayed. With paging new lines are displayed at the top of the screen above those previously displayed.

Another hardware facility provided to deal with this problem is the provision of memory which can hold more information than may be displayed on the screen. This may take the form of reserve pages or screenfuls of information which almost instantaneously replace the currently displayed page; alternatively the operator may be able to scroll through the memory in both directions. (This fits the scroll analogy much better since information isn't lost when rolled up but may be rolled down again.) The problem however is merely postponed from the boundaries of the screen to the boundaries of the memory.

Both paging and scrolling on their own are inadequate answers to the problem since at the speed information is displayed, either uncontrolled scrolling or paging is too fast to be read by the operator. Such a solution must be a software one, but it will take advantage of the hardware facilities provided by the terminal.

Editing capability

When a terminal operates in block mode, messages may be composed and corrected before being transmitted. The fact that the characters in the terminal's memory can be changed without the processor being aware of it allows editing of messages. The most basic editing capability is the mere replacement of the character at the cursor position. However it is often desirable to allow more than this, the replacement of one character by two or two by one.

The editing logic which is provided on many terminals takes advantage of the fact that all characters are constantly circulating through a temporary register where the insertion or deletion of extra characters or even lines can be accomplished. Thus, character insertion

where the characters to the right of the cursor are repositioned to allow space for the new character and deletion where the unwanted character is removed without leaving a space are quite simple to implement. Not all the characters to the right of and below the cursor need be moved. This usually depends on the terminal or even different switch-selectable modes available on that terminal. The editing action may apply to the end of the screen, to the end of the memory or, the most common, till (in the case of insertion) two adjacent space characters are found one of which is removed, or (in the case of deletion) one space character is found which is replaced by two.

Line insertion and deletion operate in a similar fashion. Line insertion, where space is made for an additional line at any position on the screen, may be regarded as a generalisation of scrolling which makes space for an additional line at the bottom of the screen only. Again, the line at the top of the screen is lost. Line deletion removes a line and creates a blank line at the bottom of the screen.

These hardware editing capabilities must not be confused with software text editing systems which are, of course, must more powerful.

Protected fields

Protected fields are provided so ADTs may be used for data entry. They allow a "form" to be set up on the screen. Protected fields are set up by the transmission of a "start protection" control character which sets a marker in that position followed by the data to be held in the protected field and finally the "stop protection" control character. Thereafter when the cursor in the course of its normal movement comes across the start of a protected field, it skips through it until it reaches the end of the field and then resumes its normal movement. As a result no data can be put into these fields from the

keyboard because the cursor can never be moved into that area. They are unaffected by nearly all other control actions too. They may not be erased, are unaffected by editing commands and when in block mode and a block of characters is being transmitted any characters falling within a protected area in that block are ignored and not transmitted.

The areas can be marked on the screen by the ability of the screen to differentiate characters by blinking, reversed contrast, a different font or other methods.

What in effect is achieved by protected fields is to make part of the screen immutable. Recognition of this fact has led with some terminals to the setting up of blank protection fields with the information in them not being retained in the terminals but being projected on the screen by a projector which can be used in combination with the terminal.