CHAPTER 4
LA120 THEORY OF OPERATION

4.1 BASIC SYSTEM CONFIGURATION

A block diagram of the LA120 is shown in Figure 4-1. The LA120 is a microprocessor-controlled system that utilizes the interaction of firmware programs with hardware circuits to perform control functions and provide functional characteristics.

As shown in Figure 4-1, the microprocessor, memory, and peripheral devices communicate via a common data bus. Peripheral devices in the LA120 include the following major functional items:

1. A single chip LSI printer controller that provides an interface to all functional elements of the printer,

2. A serially accessed non-volatile memory that stores set-up data when machine is turned off,

3. A keyboard consisting of an array of switches and a number of LED displays, and

4. A serial transmitter and receiver that provides data communication interface.

4.1.1 Microprocessor

The LA120 firmware is executed by an 8080A 8-bit microprocessor running from a 2 MHz clock. Although the microprocessor has a 16-bit address space, typically only part of the address is decoded by many devices on the bus. The microprocessor receives interrupt requests and RST vectors from the DC305 printer controller. When powering up, the microprocessor's reset line is asserted, thus starting program execution from location 0 with interrupts disabled.

4.1.2 Device Addressing

Both memory and I/O addressing are used in the LA120. In many cases, address lines are used to output data to devices as well as to select them. Conversely, data bits are used in some cases to select parts of a device.

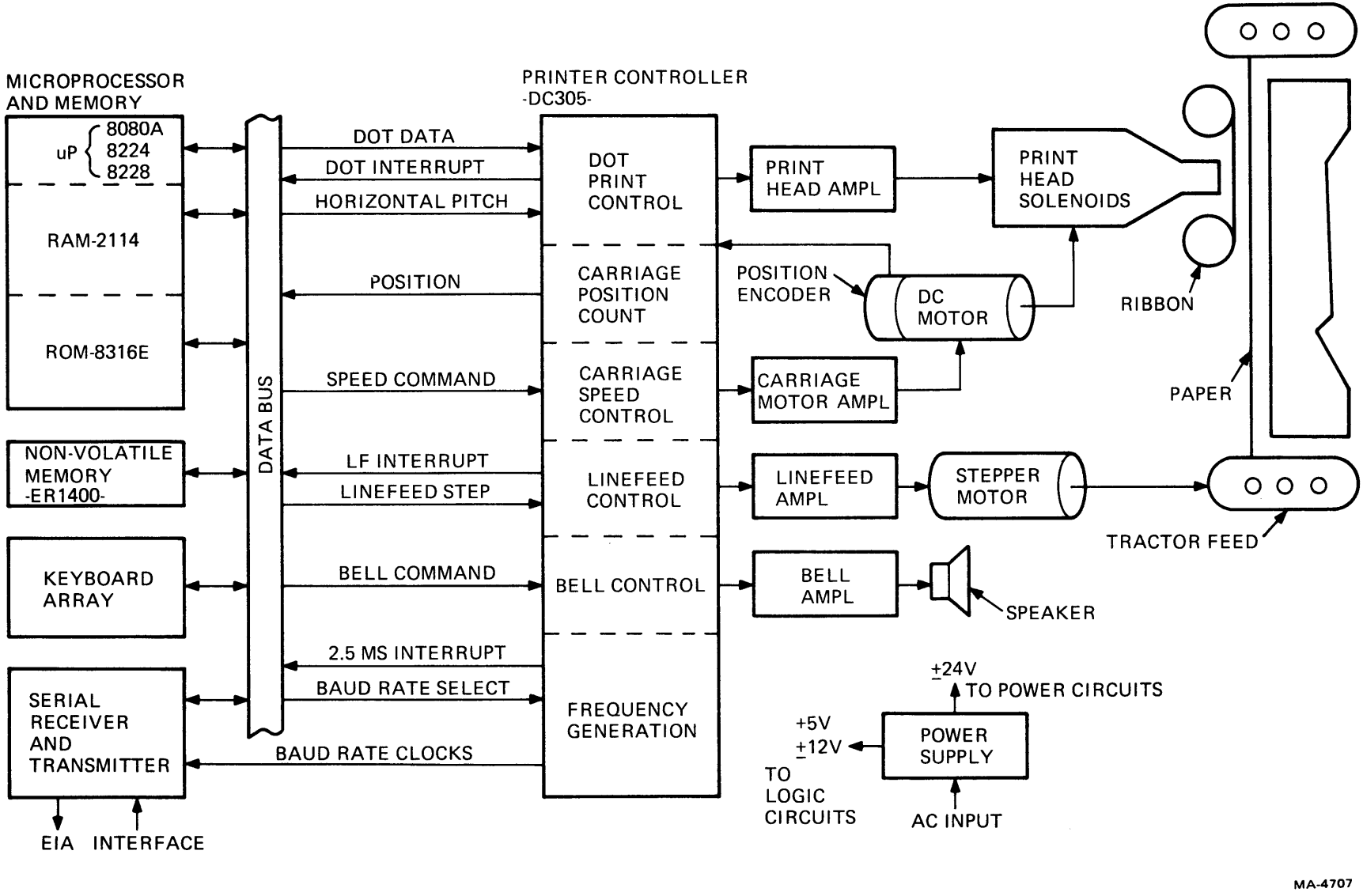


Figure 4-1 LA120 Block Diagram

4.1.2.1 Memory Space – LA120 memory read/write decoding is summarized in Table 4-1.

Table 4-1 LA120 Memory Read/Write Decoding

Address Range*	Device
0000H – 0FFFH	ROM
1000H – 1FFFH	ROM
2000H – 2FFFH	ROM
3000H – 3FFFH	Keyboard (read)/Display (write)
4000H – 4FFFH	RAM
5000H – 5FFFH	RAM
6000H – 6FFFH	Non-volatile memory
7000H – 7FFFH	DC305 printer controller
8000H – FFFH	Not used – reserved for options

*The "H" suffix indicates hexadecimal notation.

4.1.2.2 I/O Space – LA120 I/O space is used as shown in Table 4-2.

Table 4-2 I/O Space Allocation

I/O Address (Binary)*	Device
0xxx xx0x	8251A USART
0xxx xx1x	Utility I/O Ports
1xxx xxxx	Not used – reserved for options

*x = Don't care.

4.1.3 Interrupts

The LA120 receives interrupts from the DC305 printer controller and the 8251A USART. The DC305 will present one of four interrupt vectors to the microprocessor, depending on the source of the interrupt request(s). These are shown in Table 4-3.

Table 4-3 Interrupt Vectors

Interrupt Vector	Source of Interrupt Request Request		
	DC305 Dot/Timer	8251A USART RxRdy	DC305 2.5 ms Clock Tick
RST 3	F	F	T
RST 7	F	T	F
RST 3	F	T	T
RST 5	T	F	F
RST 1	T	F	T
RST 3	T	T	F
RST 1	T	T	T

As Table 4-3 indicates, the USART receiver ready interrupt is masked by either or both of the other interrupt sources.

4.1.4 ROM

The standard LA120 has five 16K bit ROMs in sockets. They occupy addresses 0000H–07FFH, 0800H–0FFFH, ..., 2000H–27FFH, respectively. The European/APL options require additional ROM from 2800H–2FFFH. The ROM sockets actually decode only address bits A12–A15; A11 is a chip select on the 16K ROM chip. The conventional ROM placement is shown in Figure 4-2a.

The European/APL option ROM is a 64K bit device (of which only 32K is used) replacing ROM 4 in the rightmost socket. When 64K bit ROMs are phased in for the standard LA120, the socket now used for ROM 0 will be addressed by 000H–1FFFH and ROM placement will be as shown in Figure 4-2b.

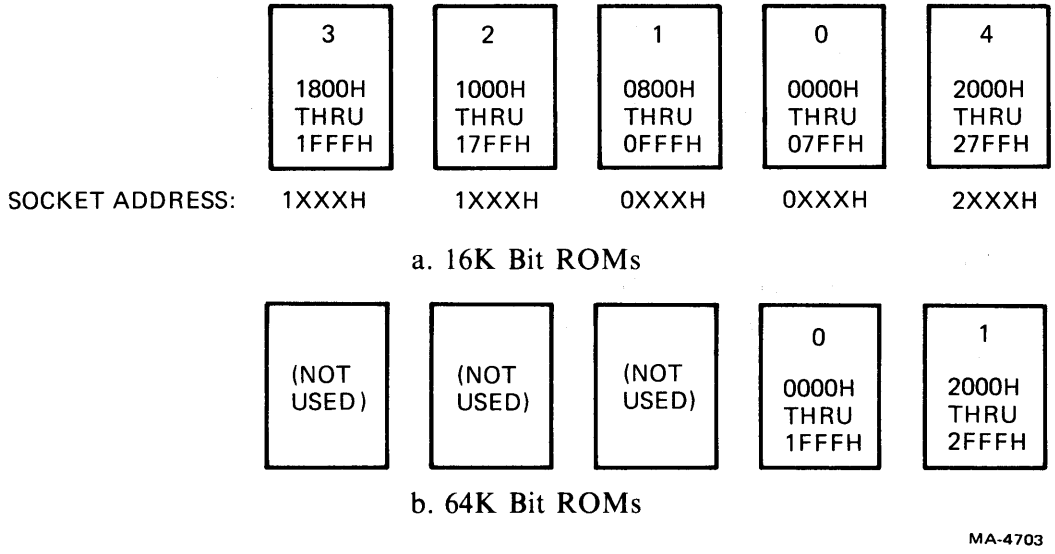


Figure 4-2 ROM Placement

4.1.5 RAM

The standard LA120 has 2K bytes of RAM. This RAM is logically in two parts, one addressable from 4000H–4FFFH and the other from 5000H–5FFFH. RAM addresses are not fully decoded, so that each 1K portion of the RAM has "ghosts" at 1K intervals in its address range. To be exact, address bits A11 and A10 are not significant. Thus, locations 4000H, 4400H, 4800H, and 4C00H are identical. By convention, LA120 firmware defines RAM to be the contiguous 2K space from 4C00H–53FFH. At no time does the firmware rely on the ghosting effect, nor is it tested by RAM diagnostics. The RAM address map is shown in Figure 4-3.

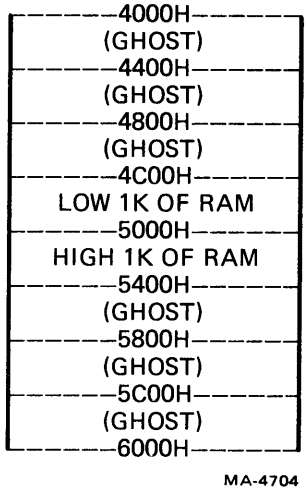


Figure 4-3 RAM Address Map

4.1.6 Keyboard

In the LA120, the microprocessor directly reads the keyboard switch array, filters the data, and generates the proper codes or other actions in firmware, thereby giving the programmer complete freedom in specifying keyboard functionality.

4.1.6.1 Keyboard Addressing (Table 4-4) – The keyboard switch array is read using memory-mapped I/O. All reads in the address range 3000H–3FFFH will select the keyboard switch array.

Keyboard addresses are only partially decoded: in particular, address bits A5–A11 are ignored. The keyboard switch array appears to the microprocessor as 16 bytes of bit-encoded data – one bit per switch. Switches are grouped into two banks: the first bank is selected by A3, the second by A4. If both are selected, the microprocessor will read the logical OR of the two. By convention, LA120 firmware uses addresses 3008H–3017H to access the keyboard which relates, in hardware, to a direction of scan from BREAK toward ESCAPE. The direction was originally arbitrary but is now crucial to proper operation of the receive only control panel in the LA120-RA.

Bits that read as 0's are open, undepressed keys, while 1's are keys that are closed. In both banks, data bits D0–D4 contain the state of regularly decoded switches; bits D5 and D6 are constant 1's. In the first bank only, bit D7 contains data for separately decoded switches; in the second bank, D7 is always 0.

Table 4-4 Keyboard Addressing

Address	Data Bit							
	D7	D6	D5	D4	D3	D2	D1	D0*
3008	space bar	(unused)		BREAK	LOCAL LF	no key	PF4	N.
3009	cover	"	"	BS	DEL	| \	LOCAL FF	PF3
300A	paper	"	"	~ \	opt LF	no key	HERE IS	PF2
300B	SHIFT	"	"	+ =	}]	LINE LOCAL	LF	PF1
300C	CAPS LOCK	"	"	_ -	{ [	" '	N,	N-
300D	CTRL	"	"	) 0	P	: ;	? /	N0
300E	RET	"	"	(9	O	L	> .	N9
300F	SET-UP	"	"	* 8	I	K	< ,	N8
3010	(unused)	"	"	& 7	U	J	M	N7
3011	"	"	"	^ 6	Y	H	N	N6
3012	"	"	"	% 5	T	G	B	N5
3013	"	"	"	$ 4	R	F	V	N4
3014	"	"	"	# 3	E	D	C	N3
3015	"	"	"	@ 2	W	S	X	N2
3016	"	"	"	! 1	Q	A	Z	N1
3017	"	"	"	ESC	TAB	VIEW	RO flag	ENTER

* Nx = key labeled "X" on numeric pad.

4.1.6.2 Switch Matrix – All of the keys in data bits D0–D4 are part of a single two-dimensional switch matrix. As a result of the keyboard's design, simultaneous pressing of three or more keys can make it appear that one or more additional keys are also pressed but in fact are not. These false key-down indications are detected by LA120 firmware, and prevented from generating any codes.

4.1.6.3 Separately Decoded Keys – Certain inputs from the keyboard must always be readable: these have been separately decoded and thus their data is always valid. These inputs are all in the first bank of switches, in data bit D7. The eight separately decoded keys are the space bar, cover interlock, paper out sensor, SHIFT key, CAPS LOCK key, CTRL key, RETURN key, and SET-UP key. The cover interlock will read 0 if open, 1 if closed; the paper out sensor is 0 if paper is out, 1 if paper is present.

4.1.7 Display

The LA120 has a 4-digit seven-segment LED display and eight individual LEDs above the keyboard array. Writes to addresses 3000H–3FFFH will access the display. Address bits A5–A11 are "don't cares." All data bits are "don't cares." In the LA120 firmware, address bits A8–A15 are always 00110000, while bits A5–A7 are not standardized.

4.1.7.1 LEDs (Table 4-5) – The eight LEDs are all randomly accessible and individually latched. The state of the latches on power up is not defined. Writes to binary addresses 0011xxxxxxx0yyy0 will turn off LED yyy. Writes to binary addresses 0011xxxxxxx0yyy1 will turn on LED yyy. In addition, every write to an LED latch resets the 4-bit BCD digit counter.

Table 4-5 LA120 LEDs

Item	Positions and Labels							
Label	ON LINE	LOCAL	ALT CHAR SET	(blank)	CTS	DSR	SET-UP	PAPER OUT
Position	#7	#6	#5	#4	#3	#2	#1	#0

4.1.7.2 Seven Segment Display – The four 7-segment digits are addressed sequentially through a 4-bit counter when a write is done to addresses 0011 xxxx xxx1 xxxx. The counter is reset by writing an LED latch (or it can overflow after 16 increments). Each write to a seven-segment digit increments the counter, as well as sending segment data and turning on the selected digit. The 16 states of the counter are shown in Table 4-6.

Table 4-6 Seven Segment Display Countr

Counter Value	Digit
0000	none – reset to this value by LED write
0001	ones
0010	tens
0011	hundreds
0100	thousands
0101	none
.	.
.	.
.	.
1111	none – next increment wraps around to 0000

Data is sent to the seven-segment digits in address bits A0–A3 in inverted BCD. For example, if A0–A3 are 1111, a zero will be displayed, if 1110, one, etc. The special value 0000 displays a blank. The digit currently selected will remain intensified until the counter is incremented or reset. In order to display a multidigit number, the firmware sends data sequentially to the four digits at a high enough rate to avoid perceived flicker.

4.1.8 Non-Volatile Memory

Memory references to binary addresses 0110xxxxxxxxxxx will access the ER-1400 non-volatile memory. Each such reference will have one wait state inserted. The ER-1400 is a serial memory, with a capacity of 100 14-bit words. The microprocessor talks to it through three general-purpose control lines, assigned to address bits A8–A10, a clock input on A0, and a data/address read/write line in data bit D7.

The ER-1400 only operates when the firmware supplies it with a clock in the frequency range of about 11–18 kHz. While bit-banging the clock, the firmware manipulates control lines and data line to perform the basic operations shown in Table 4-7.

Table 4-7 Non-Volatile Memory Operations

Control Lines	Operation
000	Standby
001	Read
010	Erase
011	Write
100	Not used
101	Shift data out
110	Accept address
111	Accept data

All data and address transfers to and from the ER1400 non-volatile memory are done in bit serial fashion using a single data pin. Timing for the clock signal is generated by the firmware program and depends on instruction execution times. To address a word in the non-volatile memory, the program presents two consecutive 10-bit addresses (total = 20 bits), which have been already decoded into two 1-of-10 codes, corresponding to the row (ten's address) followed by the column (unit's address).

Example: Address 60 - 0001000000 - 0000000001

The electromechanical portions of the LA120 are placed in a quiescent state with interrupts disabled prior to accessing the non-volatile memory. This allows the timing of the firmware-generated clock signal to be precise. Because interrupts are disabled, the non-volatile memory access routines continue to track the print head using a polling technique, thus preventing any possibility of losing track of print head position.

4.1.9 DC305 Printer Controller

Memory references to addresses 0111 xxxx xxxx xxxx will select the DC305 printer controller. Address bits A2–A11 are ignored for both reads and writes; A0 and A1 are also ignored in reads. One wait state is generated for each access to the DC305. Write accesses must be at least 7 μs apart; there is no restriction on reads. Most writes to the DC305 pass additional subaddressing data in the data field.

A full description of the DC305 is given in Paragraph 4.2.

4.1.10 8251A USART

All I/O to ports 0xxxxx0x accesses the 8251A USART. The USART itself decodes address bit A0, so that if A0 = 0 a data port is accessed, while if A0 = 1 a control/status port is accessed. By convention, the LA120 firmware uses the lowest valued I/O port numbers. The USART ports are as shown in Table 4-8.

Table 4-8 8251A USART Ports

Input/Output	Port	Function
Input	0xxx xx00	Received Data Buffer
Output	0xxx xx00	Transmitted Data Buffer
Input	0xxx xx01	Status Flags
Output	0xxx xx01	Mode/Command Control

The 8251A requests an interrupt whenever RxRDY (receiver ready) is asserted. If no other interrupts are pending, this will result in a RST 7 interrupt. Reading the received data buffer resets RxRDY and thus the interrupt request.

LA120 firmware checks the USART transmitter status via TxE (transmitter empty), not TxRDY (transmitter ready). This avoids an extra character time latency in sending XON/XOFF commands.

4.1.11 Utility I/O Ports

I/O addresses 0xxx xx1x select utility I/O ports. These consist of one 6-bit input port and one 6-bit output port. The outputs are reset by the power-up sequence. Bit assignments of these ports are given in Table 4-9.

Table 4-9 Utility I/O Ports

Port	Data Bit	Function
Input	0	Clear To Send
	1	Ring Indicator
	2	Option Present Indicator: 0 = option present
	3	Loopback Gate Indicator
	4	(not used)
	5	(not used)
	6	Secondary Received Line Signal Detect
	7	Received Line Signal Detect
Output	0	Data Terminal Ready
	1	(not used)
	2	(not used)
	3	Loopback Enable
	4	(not used)
	5	(not used)
	6	Baud Rate Clock Division Factor (0 = 16, 1 = 64)
	7	External Reset Signal

The loopback enable disconnects the received and transmitted data serial lines from the external interface and connects them together. It is not currently tested or used by LA120 firmware.

4.2 DC305 PRINTER CONTROLLER

The DC305 printer controller performs functions that are beyond the capabilities of the microprocessor and provides an interface between the microprocessor and the printer electromechanical components.

4.2.1 Dot Print Control

When there is at least one character to print, the microprocessor loads print head dot pattern data into a buffer whose output selects head drivers to be activated. When carriage speed and position are appropriate for printing, the microprocessor enables printing and enables the dot interrupt.

Print head motion signals from the carriage position encoder are combined with horizontal pitch SET-UP data from the microprocessor to trigger a countdown timer that allows selected head drivers to be turned on. At the termination of the count, the drivers are turned off and the buffer advances. When there is room for more dot pattern data in the buffer, a dot interrupt is generated, requesting that the microprocessor load more dot data into the buffer. When there is no more data to be printed, the microprocessor disables the dot interrupt. When the buffer becomes empty, printing ceases.

4.2.2 Carriage Position Count

Print head motion signals consist of a pair of asynchronous logic levels whose states change in quadrature to each other (00, 01, 11, 10, 00) as the carriage moves. After synchronization, the state changes accumulate in an 8-bit binary up/down counter as carriage position. The microprocessor periodically reads the counter to track carriage position over the entire width of the carriage and to determine carriage speed.

4.2.3 Carriage Speed Control

The voltage supplied to the dc motor is controlled by a pair of pulse streams. The first pulse stream has a duty cycle that is proportional to the binary value of a microprocessor-supplied speed command. The second pulse stream has a duty cycle that is proportional to the frequency of print head motion signal state changes (i.e., motor speed). The integrated pulse streams are subtracted to yield the motor voltage. A steering circuit interchanges the roles of the two pulse streams to provide direction control as a function of the directions of the speed command and the motor speed.

4.2.4 Line Feed Control

The line feed stepper motor is controlled by a two bit state word that controls the polarity of the voltage to each of the motor windings and a third bit that controls the amplitude of the current in the windings. Higher current is used to run the motor rather than to hold a steady position.

The timing of state changes during line feeding is completely under control of the microprocessor, with the aid of a programmable timer interrupt that is made available by utilizing the countdown timer of the dot print control circuit in a timer mode rather than in a printing mode.

4.2.5 Bell Control

Latches and gating logic enable the microprocessor to control the propagation of either a 400 Hz signal or a 2400 Hz signal to a small loudspeaker that supplies the bell function and acts as an audible warning device. The microprocessor can also control the application of an ultrasonic chopping signal to the bell drive signal that has the effect of reducing the energy content in the audio range and thus acts as a volume control.

4.2.6 Frequency Generation

Many processes in the LA120 depend on timing signals for proper operation. The master timing source is the 18 MHz crystal that drives the 8224 clock. The 8224 clock provides a 2 MHz signal that is used by the 8080A microprocessor, the 8251A USART, and the DC305 printer controller as their primary timing inputs. The 2 MHz clock input to the printer controller is divided by 13, yielding approximately 153.6 kHz, which is the clock rate that is used to drive most of the sequential logic in the printer controller. A divider chain produces 76.8 kHz, 38.4 kHz, 19.2 kHz, 9.6 kHz, 4.8 kHz, 2.4 kHz, and 1.2 kHz signals. A divide-by-3 circuit produces 400 Hz. The signals from 153.6 kHz to 1.2 kHz drive a bit rate multiplier in the carriage speed control circuit to produce the signal whose duty cycle is proportional to speed command. The 400 Hz signal provides an interrupt to the microprocessor every 2.5 ms, which allows firmware processes to run periodically and thus be capable of performing timing-dependent operations. A baud rate generator circuit further divides various outputs of the divider chain to produce baud rate clock signals for the lower baud rates and the split baud rates. A baud rate selector circuit allows the microprocessor to select various outputs of either the divider chain or the baud rate generator to be supplied as baud rate clock signals to the USART.

4.2.7 Tick Alarm

To prevent a microprocessor failure from causing physical harm to power circuits or electromechanical components, the DC305 has a special monitoring function that requires the microprocessor to prove its integrity by sending a clear tick command every 2.5 ms. Failure to do this is proof of malfunction and results in disabling most chip outputs. The DC305 powers up with tick alarm activated so that outputs are disabled until a programmed initialization sequence has been completed.

4.2.8 Interrupt Vectors

The DC305 can request interrupts at either vector RST 5 or vector RST 3. If both interrupts are requested, an interrupt to vector RST 1 results. The RST 3 interrupt is always due to the 2.5 ms (400 Hz) clock tick, while the RST 5 interrupt may be due to either the real-time clock timeout (typically while line feeding) or due to the dot FIFO being half empty (during printing).

4.3 LA120 FIRMWARE OVERVIEW

4.3.1 Scheduling

Processes run at four priority levels: highest priority is interrupt level (with interrupts disabled), then tick level, interlaced tick level, and (lowest) background level. Note that except for interrupt service routines, interrupts are normally kept enabled. The hardware interrupt vector scheme subdivides interrupts into two priorities, with receiver ready interrupts below all others. Firmware then further divides the higher level so that dot or timer interrupts take priority over tick interrupts.

Processes that run at background are the least time-critical, since background may be locked out by higher priority processes for short periods of time. On the whole, though, background gets the lion's share of processor time, typically more than 50 percent. Tasks that run in background include line image formatting, print scheduling, SET-UP command interpreting, and the ANSI parser. All printing, slewing, and vertical motion is originally scheduled by background level routines, even though executed at higher priority. When the processor is idle, it will be looping at background level.

Interlaced tick level processes execute periodically after noninterlaced tick. Every interlaced tick process runs once every 1, 2, 4 or 8 ticks, where a tick is 2.5 ms. This is accomplished by a round-robin scheduler, where each of eight phases is executed, in turn, on successive ticks. Each phase runs to completion before the next phase is begun. Examples of interlaced tick processes are the keyboard handler, communications protocol handler, and display handler.

Noninterlaced tick processing is begun immediately after taking a tick interrupt; interlaced tick follows its completion. Currently, servo control comprises all of noninterlaced tick processing. This level of processing must be guaranteed to finish before the next tick interrupt comes along. Noninterlaced tick now takes about 700 μs on average.

Interrupt service routines are the most time-critical and are kept as short and fast as possible. They include the dot FIFO interrupt, timer mode interrupt, and USART receiver ready interrupt.

Processes like initialization and non-volatile memory handling require that interrupts be off while running and thus effectively run at highest priority.

4.3.2 ROM Layout

Standard LA120 firmware occupies 10K bytes and thus normally lives in five 16K bit ROMs. The first four ROMs are treated as a group (in anticipation of 64K bit ROMs) and consist of code and tables judged low risk – that is, unlikely to change. The fifth ROM is reserved for high risk data. It is primarily tables but does have some executable code as well.

To get the European/APL options, a sixth ROM must be installed. The presence of this ROM is flagged by its first byte of zero. When optional features are invoked, routines in the standard set of ROMs test this byte to see if the options are enabled.

Each ROM has a one byte checksum. This requires that every byte in each ROM be fully defined. Thus, the fourth and fifth ROMs have large sections of padding (all 0FFH bytes) to fill up empty spaces. Many tables also have smaller sections of padding contiguous with them. Because of this padding and the checksums, any changes to the ROM patterns will usually require recalculating the padding section sizes and checksum values.

4.3.3 RAM Layout

The LA120 has 2K bytes of RAM. The first 1K is used for the input buffer, a circular queue used to store characters received from the USART or that have been echoed locally. The next page (where a page is 256 bytes starting on a multiple of 256) contains several small buffers, the stack, and the tab bit maps. The page after that contains all non-volatile parameters and other variables. Thus, after an initial LXI H,FOOONE is performed, usually only a MVI L,LOW FOOTWO is needed to reference a second variable.

RAM locations are all written to zero at power-up time. If a location needs to be initialized to another value, its initial value is contained in a list processed by a RAM initialization routine.

4.4 PRINT CONTROL FIRMWARE

One of the more complicated operations in the LA120 firmware is the starting of the print operation after a carriage return or tab. The DC305 chip will only begin the printing operation after it has been given a print start command. This command will be executed at the next point at which printing can legitimately be started. This section deals largely with the routines for ensuring that the print start occurs under the proper conditions and at the proper point. In addition, a description of the control of print speed to limit component heating is included at the end of the section.

4.4.1 Starting the Printing Operation

The task of starting the print operation is handled in two steps. The first step is to prepare the DC305 to begin printing. This is done once for every print start. On the next and subsequent ticks, tests are run to see whether the carriage has arrived at the proper location to begin printing and that it is traveling at the proper speed. These tests are repeated until printing begins.

The forward printing process begins by obtaining the first character to be printed from the line buffer and loading its dots into the dot buffer. The first four sets of dots are then loaded into the FIFO of the DC305 chip and the dot interrupt is enabled and the carriage is started on the approach to the starting point of the print string.

The reverse printing process begins the same as that previously described, except that all operations are for backward printing. The right-most character in the line is loaded into the dot buffer.

Print start is enabled by a print start command that is stored by the DC305 chip and printing begins when the print head wires cross the boundary of the intercharacter space in either direction. To initiate a forward print start properly, the command must be given before the impact point crosses from the intercharacter space of the previous character into the cell of the character to be printed.

As a practical matter, the print start command is given only when the impact point is in the Safe Print Start Zone and the carriage is moving in the proper direction. The Safe Print Start Zone begins three transitions to the right of the left-hand cell boundary of the previous character and ends three transitions to the left of the intercharacter space of the same character cell as shown in Figure 4-4.

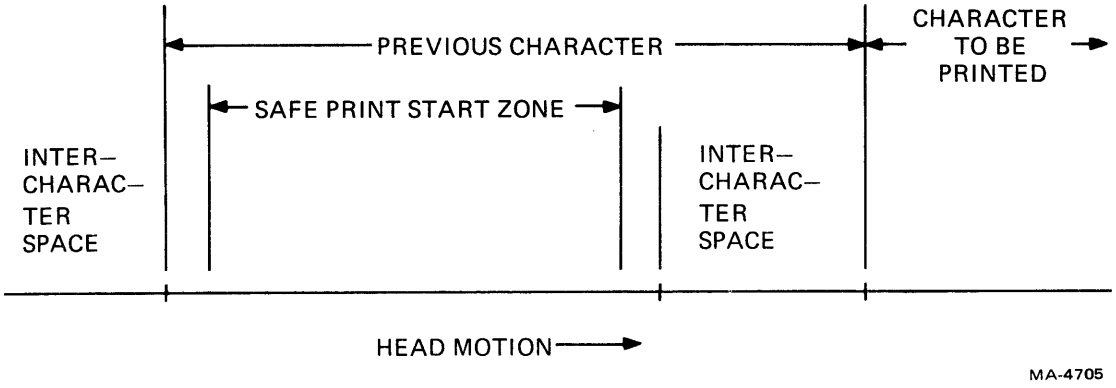


Figure 4-4 Forward Print Start

This scheme ensures that print start is only executed when the impact point crosses the left-hand cell boundary of the character to be printed. The scheme thus avoids premature print starts and print starts in the reverse direction that might otherwise be induced by unintended motions in the reverse direction. Such undesirable print starts would cause the characters printed subsequently to be displaced. Therefore, tests are made by the microprocessor to ensure that the carriage is traveling to the right at a speed that does not exceed the print speed by more than a small amount, that the impact point is in the Safe Print Start Zone, and that other conditions outlined below are met.

A number of tests must be made before forward printing may actually start. The first test is to see whether a vertical motion is in progress or pending, a cover is open, paper is out, etc. If any of these conditions are true, the routine is aborted. Next, the direction in which the carriage is traveling is checked. If it is negative, the routine is aborted; otherwise, the tests are continued by computing the maximum permissible approach speed as a function of the print speed limit currently in effect. Then the position counter is read and a projected impact point is computed. If the impact point is to the right of the Saft Print Start Zone, the carriage has overshot the mark and the routine is aborted. If the impact point is more than 255 transitions to the left of the right-hand border of the Safe Print Start Zone, the routine exits and no further tests are made during this tick interval since there is time to do the tests during the next tick interval. If the impact point is within range, the following tests are made:

1. A check that the carriage speed is less than the maximum allowed,

2. A determination that the impact point is less than 34 transitions to the left of the right-hand border of the Safe Print Start Zone,

3. A test that the print head is not likely to stop or change direction in the near future.

If any of the preceding tests are failed, printing cannot be started at the current tick time and the routine exits; otherwise, a go ahead for printing is issued and a final test is made to ensure that the impact point is to the right of the left-hand boundary of the Safe Print Start Zone. In the event that this test is failed, the routine aborts. If the test is passed, the print start command is given to the DC305 chip and the address of the Dot Service routine is set up in the dot interrupt service routine and the printing operation is finally begun.

The Safe Print Start Zone for a reverse print start is illustrated in Figure 4-5.

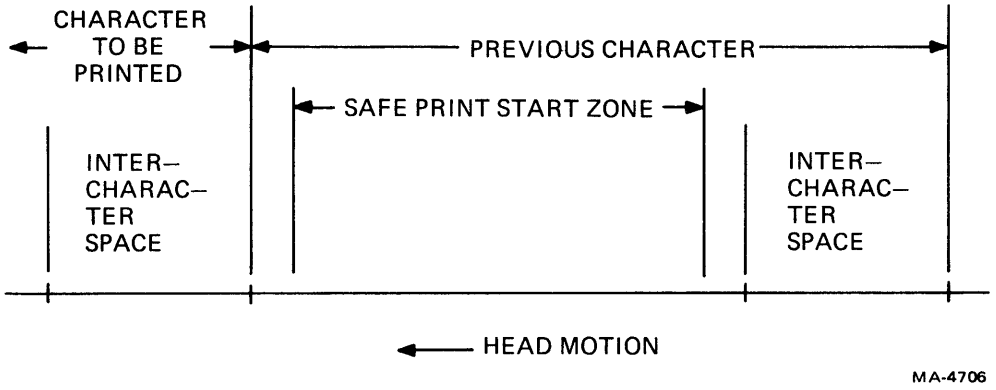


Figure 4-5 Reverse Print Start

The procedure for initiating a reverse print start is the same as that used for the forward print start, except that direction of motion is from right to left instead of from left to right.

4.4.2 Flight Time Compensation

The flight time compensation procedure compensates for the fact that it takes approximately 400 μs from the time the command is given to a solenoid to print a dot until the print wire hits the paper. If this command were to be given when the head is just at the point where the dot is to be printed and the head were traveling at 18 in/s, the dot would be printed 0.0072 inch past the intended point. If the next line were to be printed in the reverse direction, the characters would be displaced by a similar amount in the opposite direction, so that the characters in two successive lines would be very noticeably misaligned. It can be seen that the amount of compensation needed is proportional to the carriage speed. The faster the carriage moves, the earlier the wire has to be fired. Compensation is made by appropriate transition add or subtract commands sent to the DC305 printer controller. These commands cause one transition to be added or subtracted from the transition counter in the chip so that the command to fire is given to the solenoids before the intended print position is reached. The flight time compensation also compensates for some mechanical inaccuracies that are functions of print direction.

4.4.3 Dot Rate Limiting

The LA120 will print reasonable text at a rate of 180 char/s indefinitely. However, heavy printing, such as many repetitions of the character "#" at 180 char/s without intervening spaces, will constitute an overload. The reason for this is that each dot represents a certain amount of energy that must be delivered and is eventually dissipated. If dots were printed too fast for too long a time, there would be heating of the head and certain power supply components. The longterm printing rate of the LA120, therefore, is limited to 2200 dots per second. The necessary speed limiting is performed on the basis of the recent printing history. The limiting function is a constant minus the integral of the difference between the dot rate and the maximum rate. A speed limit is computed from this function which is compared with the maximum permissible print speed for the current font. The lower of the two values is subsequently used to limit the printing speed. The scheme takes into account the thermal mass of the print head and of the power supply components and it allows heavy printing for a period of about 1 second. If the heavy printing continues, the print speed is reduced gradually until the dot rate is equal to 2200 dots per second.

4.5 CARRIAGE SERVO FIRMWARE

The carriage servo firmware ascertains the current position of the carriage and generates the speed commands needed by the carriage servo to move the carriage to the position where it should be. Actual and desired positions are determined every time a tick interrupt is received, which is every 2.5 ms. The speed command given to the servo is then updated on the basis of these position determinations. This command is such that the carriage servo moves the carriage toward the desired position as rapidly as possible. As the carriage approaches the desired position, the speed is reduced to allow for a smooth stop.

4.5.1 Transitions

The position of the carriage is measured by an incremental two-channel encoder mounted on the back of the carriage servo motor. Each channel has a square-wave output with 660 cycles per revolution, with the output of one channel leading the output of the other by 90 degrees. A transition is defined as the change of the output of one channel from the logic "0" level to the logic "1" level or vice versa. Thus, there are two transitions per cycle in each channel and since there are two channels there is a total of four transitions per cycle and 2640 transitions per revolution.

The motor moves the carriage 4 inches per revolution. Hence, there are 660 transitions per inch of carriage motion and

1 Transition = 0.001515 inch

The carriage servo routines are entered every time a tick interrupt is received. The tick interrupt is cleared and the actual position of the carriage is determined. Speed, flight-time correction, carriage position, and error are determined. A real-time clock is also incremented.

4.5.2 Reading the Position Counter

Upon entry, the position counter in the DC305 chip is read. Interrupts are then enabled. The previous reading of the position counter is retrieved and subtracted from the current reading of the counter. The difference is stored for use in subsequent calculations. Since readings are made at fixed time intervals, this difference is a measure of the carriage speed. The value of the current counter reading is stored for use during the next tick interrupt service. This calculation is valid as long as the carriage did not move more than 127 transitions during the preceding tick interval.

4.5.3 Carriage Speed Command

The determination of the command speed for the carriage servo is begun on the basis of how far the carriage is from where it is supposed to be. The actual target speed for the servo is obtained from a table lookup. This target speed is the speed input for subsequent routines where it is processed to generate the actual servo command. The sign of the target speed is chosen so that the carriage moves toward the point at which it is supposed to be and the target speed decreases as the carriage approaches its destination so that it arrives there at a reasonable speed.

In general, the speed command is based on the target speed unless the change from the previous target speed was too large. In that case, the acceleration limit is applied.

4.5.4 Error Conditions

A speed error is computed by comparing measured carriage speed with the target speed. If this error exceeds a limit, it is an indication that the carriage is not moving as fast as it should and that it may, in fact, be stopped completely. If the error exceeds the limit for many ticks in succession, it is presumed that the carriage is stuck. This causes a pause flag to be set. Other pause flags are set when there is no paper or when the cover is open. Before a speed command is sent to the DC305, the pause flags are examined. If none are set, the speed command determined as previously described is sent. If any pause flag is set, a speed command of zero is sent.

4.6 COMMUNICATION FIRMWARE

The LA120 controls transmission and reception of data on its serial interface by the use of the modem control lines to generate several full- and half-duplex protocols. SET-UP commands are used to fine tune the various protocols to suit the various modem and remote end requirements.

4.6.1 Communication Modes

The LA120 has a wide range of communication modes, and while none are highly complex in themselves, their number and subtle differences can be confusing. There are two basic full-duplex modes and three half-duplex modes selected with the modem setup (M):

1. Full duplex without EIA controls
2. Full duplex with EIA controls
3. Half duplex - supervisory control
4. Half duplex - coded control, EOT
5. Half duplex - coded control, ETX

Other SET-UP commands affecting communication control are break enable (U), secondary channel (S), and auto disconnect (D).

4.6.2 Local Mode

Whenever the terminal is in local mode, all modem signals are unasserted. No calls can be generated or received, and transmit and receive are disabled. All protocols except full duplex without EIA controls start at this level.

4.6.3 Disconnects

Each of the following sections on full- and half-duplex modes define disconnects and the line functions that can cause them within each protocol. Additional disconnects can be caused by situations within the printer that cause printing to stop, such as paper out, cover open, or a head jam. Disconnect generation is a function of the SET-UP commands break enable (U), XON/XOFF (X), and auto disconnect (D) as shown in Table 4-10.

In all modes except M=1, a disconnect can be generated from the keyboard with a shift break. A long break disconnect is generated which unasserts all signals and puts a space on transmit data for 3.5 seconds.

4.6.4 Full Duplex Without EIA Controls

This mode asserts DTR and RTS whenever the LA120 is on-line and ignores modem signals to the LA120. This mode, normally used for current loop, has receive and transmit enabled at all times while on-line. The break and speed/restraint function, as described in "Full Duplex with EIA Controls," are operational, but the disconnect functions are not.

4.6.5 Full Duplex with EIA Controls

This mode supports a full modem interface requiring the following conditions to be satisfied in order to establish connection.

1. DSR must be asserted; then the LA120 asserts RTS.

2. RLSD must be asserted for at least 300 ms after DSR is asserted; then the LA120 enables transmission and reception.

This mode also provides automatic disconnect capability by use of the Data Terminal Ready signal.

Table 4-10 Disconnect Generation vs SET-UP Commands

SET-UP Commands		
Auto Disconnect (D)	**Break Enable (U) and XON/XOFF Enable (X)**	**Action**
D = 0	X = 1 U = 0 or 1	XOFF Sent
	X = 0 U = 1	Break
	X = 0 U = 0	No Action
D = 1	X = 1 U = 0 or 1	XOFF Sent, followed by Disconnect
	X = 0 U = 1	Disconnect
	X = 0 U = 0	Disconnect

4.6.5.1 Full-Duplex Break – A break can be generated manually by the break key or automatically by a printer fault condition such as paper out. In full duplex, a break consists of asserting a space on the transmit data line for 220 ms if transmission is enabled. If transmission is disabled, the break will remain pending until transmission is enabled or a disconnect is generated.

4.6.5.2 Full-Duplex Disconnect – In order to cause the modem to disconnect from the telephone line, DTR is deasserted for 80 ms. All signals are deasserted during this time, although it is the DTR signal that the modem recognizes as a disconnect. There are three types of line-generated disconnects:

1. Failure of either DSR or RLSD to assert within 20 seconds after the termination of a ring indication,

2. Failure of RLSD to assert within 5 seconds after DSR when initiating a call,

3. After connection is established the assertion of RI, deassertion of RLSD for 5 seconds or the deassertion of DSR.

In addition, an EOT received or transmitted with auto disconnect enabled will cause a disconnect. When disconnect is initiated by the transmission of an EOT, sufficient time is allowed for the EOT to be completely transmitted before the disconnect in order for the remote end to also disconnect. Also, a long space disconnect can be generated from the keyboard. This produces a space on the transmit data line and deasserts DTR for 3.5 seconds.

4.6.5.3 Restraint Mode – When a full-duplex mode is selected, the secondary channel setup (S=1) selects restraint mode. The secondary request-to-send signal is used to control the reception data in the same manner as XON/XOFF, although it cannot be controlled from the keyboard. This line is used to signal the remote end to stop sending characters in order to keep the input buffer from overflowing due to the receive character rate or a print stop condition (paper out, cover open, etc.). The remote end must maintain the proper signals to have receive enabled before using the restraint line to control data transmission.

4.6.5.4 Speed Control Mode – The speed control function is selected with SET-UP S=0 in a full-duplex mode. While in this mode, the LA120 indicates its selected speed on the speed select line. The signal is asserted if the operator-selected baud rate is 1200 or higher. This allows a high speed call to be initiated from the LA120 without manually setting the high speed button on the modem.

Also, the LA120 will reset its operating baud rate to 1200 baud if the speed indicator signal is asserted. The 1200 baud rate is forced only as long as the line is asserted and does not affect the operator-selected (displayed) baud rate.

4.6.6 Half Duplex

Due to the "one at a time" definition of half duplex, elaborate protocols (compared to full duplex) are needed to define who should transmit at any given time. Each time the transmitter and receiver switch functions, the line is "turned around." This basically consists of switching the end of the line that asserts RTS, which reverses the transmit/receive mode of the modem and switches the carrier generation from one end to the other. Also, when echo suppressors are on the line, it is necessary to turn them around in order to attenuate in the opposite direction. The LA120 incorporates three methods of controlling line turnaround. In supervisory control mode the host controls all line turnarounds by manipulating the secondary control lines. Reverse channel is mandatory for this mode. The two other protocols (coded control with reverse channel and coded control without reverse channel) allow the transmitting device to control line turnaround via specific control characters. If reverse channel is used, these lines provide confidence as to the fate of the transmitted data. Without these signals, the transmission is "blind."

4.6.6.1 Initial Direction Determination – When the LA120 is initially put on-line (or at power up), data can be neither transmitted nor received. When the terminal is called, RI will assert before DSR. In auto answer mode, most modems answer the call (go off hook) before asserting DSR. However, some modems allow DSR to assert after a couple of rings but before the call is answered. With this sequence, the terminal attempts to establish receive mode. If the terminal operator is initiating the call, DSR asserts when the modem is placed in data mode. Since DSR is asserted without RI, the terminal attempts to enter transmit or receive mode, depending on the HDX initial state SET-UP command. If the terminal attempts to enter receive mode and RLSD is not asserted within 5 seconds, the normal timeout disconnect occurs.

4.6.6.2 Reverse Channel – The reverse channel transmits supervisory or error control signals. These signals flow in the opposite direction to which data is being transferred. Due to the relative lower bandwidth of the reverse channel (to the forward channel), it is not used for data exchange.

4.6.6.3 Request to Send Delay – As noted in the RLSD definition, the analog loopback option turns around certain lines to the LA120:

1. RTS asserted causes RLSD to be true.
2. SRTS asserted causes SRLSD to be true.
3. Receive data mimics transmit data (local copy).

For this reason, whenever RLSD or SRLSD is to be used, 300 ms must have lapsed since the local driving force (RTS, SRTS) has been removed. Up until that time the signals do not represent the remote end.

Also, RTS cannot be lowered until the last character is completely serialized (transmit complete).

4.6.6.4 Turnaround Characters – The turnaround characters (EOT or ETX) initiate the line turnaround when received or transmitted. To eliminate the need for the operator to generate the turnaround control code, the LA120 automatically sends the control code after RETURN or ENTER key is typed, after sending the normal code for that key.

4.6.6.5 Half-Duplex Break – A break can be generated manually by the BREAK key or automatically by a printer fault condition such as paper out. There are three cases of break processing in half duplex:

1. Transmit mode (RTS true) – The break is transmitted as a space on the transmit data line for 220 ms.

2. Receive mode (RTS false) – The break is transmitted as a space on the SRTS line for 220 ms. When operating with "coded-no reverse channel," the break is ignored when in receive mode.

3. While switching modes – If neither receive nor transmit is enabled, the break will not be processed until a definite line direction is established.

4.6.6.6 Half-Duplex Disconnect – Hanging up the telephone to disconnect from the line is accomplished by dropping DTR for 80 ms and resetting all control lines to their initial state.

There are five line conditions that will cause a DTR disconnect:

1. Failure of either DSR or RLSD to assert within 20 seconds of the termination of a ring indication.

2. Failure of SRLSD to assert within 5 seconds after DSR, when initiating a call with reverse channel.

3. Failure of a line turnaround to complete within 5 seconds.

4. In coded control, deassertion of RLSD or SRLSD for 5 seconds without the turnaround character. (If no reverse channel, only RLSD is monitored.)

5. After a valid line direction is established, the assertion of RI or the deassertion of DSR.

In addition, a command can be used to initiate a disconnect. An EOT or DLE-EOT from the keyboard or line will disconnect the line. If EOT is used as the turnaround character, DLE-EOT is used as the disconnect command. When a disconnect is initiated from the keyboard, the EOT or DLE-EOT is sent to the remote end before the disconnect in order for the remote end to also disconnect. Also, a long break disconnect can be generated from the keyboard. This produces a space on the transmit data line and drops DTR for 3.5 seconds.

4.6.7 Communication State Control

The communications firmware consists of three modules and various pieces of code in other modules. The controlling module is the communication handler that decodes the communication state table to generate the protocols. The third dedicated module is the input scanner, which scans the input data stream for characters that affect the communications protocol. These modules operate at tick level and are always monitoring and updating the communication operation.

4.6.7.1 Communication State Table – This table consists of encoded input and output conditions that reflect the protocol diagrams presented in a following section. Inputs consist of signal values from the communication line (DSR, CD, etc.) and terminal functions (break request, etc.). Outputs are signals that the LA120 can assert (DTR, RTS, etc.) and terminal control (transmit disable, etc.). There is also a real-time clock output (set timer) and input (time expired) that allows timing of functions in the range of 20 ms to 90 seconds. Timed functions include breaks, turnarounds, etc.

Each set of input and output conditions is considered a state. When a state is entered, the outputs for that state are invoked, providing a strict definition of the communication line at that time. These outputs are held until one of the inputs associated with that state is of the desired value, at which time the pointer associated with the input is used to access a new state and new outputs are generated. These states are linked in such a manner as to generate the various protocols.

The conditional branch entry is an addition to the state table entries. This type of entry does not generate new outputs, but allows the linkage of states (and thus the protocol) to be modified according to various SET-UP conditions. This ability to branch on SET-UP conditions allows communication states to be shared between protocols. An example is that most protocols use the same series of answer states and then branch after a connection is established to implement the actual protocol.

4.6.7.2 Communication Handler – The main function of the communication handler is to implement state transitions as defined in the state table. There are also some functions that are always present (or almost) and, therefore, are not carried in each entry of the table, but are handled as a parallel function in the communication handler code. The communication handler is broken down into two execution blocks that run at tick level. Each block is called alternately by the LA120 scheduler every four ticks, generating a total pass of the handler in 20 ms. In addition to these two blocks, there is the initialize section of code that is called at power-up and when communication affecting setups is changed. The functions included within the handler are:

1. Control LINE/LOCAL, DSR, and CTS keyboard lights
2. Gather inputs for table exit conditions
3. Control timer
4. Check for disconnects
5. Speed/restraint for full duplex
6. Compare exit conditions to valid exits for the state
7. Invoke new state outputs
8. Detect and process branch table entries.

Break Processing – Breaks are received via a break request flag, set outside the communication handler due to keyboard action or a printer fault condition such as paper out. The break request is passed to the exit condition inputs if breaks are enabled and is implemented by a series of normal state transitions if the selected protocol uses breaks.

Disconnect Processing – A disconnect request flag is set outside the communication handler if the appropriate code is received or transmitted, or a printer fault condition such as paper out occurs and auto disconnect is selected. Disconnects are implemented by starting the protocol at its initial state after any pending XON/XOFFs are sent. By forcing the initial state, modem SET-UP 1 (full duplex without EIA controls) will not perform a disconnect because its initial state is with modem signals asserted, although internal initializations, such as resetting the USART, will occur. For this reason, it is always recommended to be sure that auto disconnect is not selected when using modem SET-UP 1.

A long break disconnect request causes a special state to be entered for the 3.5 seconds, then exited to the initial state for the protocol in effect.

Turnaround Processing – A turnaround request flag is used by the coded control protocols to initiate a line turnaround. The flag is set outside the communication handler by reception or transmission of the appropriate control code for turnaround. This flag is placed into the input byte and is available as an exit condition to be used if the protocol in effect uses turnaround characters. The flag is reset by the communication handler.

Timer – The communication handler contains a software timer that can be set by an output value from the state table. Each state that sets the timer carries an extra byte to define the duration. When the countdown timer has expired, a bit in the input byte is set that can be tested as an exit condition and the necessary states can be started to process the timeout. All time values mentioned in the state diagram are implemented using this timer.

SRTS Flag – The SRTS flag is a byte containing the value of secondary request to send when in full duplex. This byte is always ORed with the outputs from the table to set SRTS when a function outside of the table (speed/restraint) needs to control that signal. The state outputs never assert SRTS while the SRTS flag is controlling SRTS; when the table is controlling SRTS, the SRTS flag is always zero.

Restraint Processing – The restraint function is implemented if in full duplex and secondary channel (S=1) is selected. If any of the appropriate printer pause conditions are true, the SRTS flag is set and a new value sent immediately.

Speed Indicator/Select Processing – Speed information is sent to the modem if in full duplex and SET-UP S=0. The operator selected baud rate is used to define the value for the signal sent using the SRTS flag and an immediate output.

Speed select to the LA120 is implemented whenever the value of secondary carrier is changed during full duplex with SET-UP S=0. When this occurs, the proper baud rate is generated according to the new value of the secondary carrier.

Branch States – These entries in the table do not generate any new outputs, but can change the state pointer according to the value of the following setups:

1. Full duplex
2. Coded control
3. Supervisory channel
4. HDX initial state.

These conditional branch entries contain an index value that is used to generate the address of a routine that will test the appropriate SET-UP status. If the routine returns with the zero bit set, the address contained in this entry is used as the next state to go to, and generates those outputs.

Auto Answerback – This function allows the answerback message to be sent the first time the terminal is transmit-ready after establishing a valid connection. When the transmitter is enabled, a flag is checked to see if it is the first time; if so, the answerback message is sent.

4.6.8 Control Code Generation and Detection

4.6.8.1 Input Scanning – The input scanner runs at tick level and scans the input buffer for characters that are relevant to the communication functions. The scanner looks for EOT, DLE-EOT, and ETX and sets the appropriate flag for the current operating mode of the LA120. The scanner keeps its own pointer and is asynchronous to the normal input buffer processor. The scanning was deemed necessary due to possible latencies in normal character processing when the buffer is full.

4.6.8.2 Transmit Scanning – Characters that have communication meanings are also scanned in the transmit routine. The same characters are checked and processed as in the input scanner, but it is done as each character is sent. Control codes that effect communications are never echoed within the terminal. The transmit routine also refreshes the USART command, sometimes generated by the communication handler.

4.6.8.3 Control Code Generation – Control codes generated from the keyboard are sent and scanned normally. When a half-duplex coded control protocol is selected, the appropriate turnaround character is built into the carriage return transmit string to relieve the operator of this code generation. These "automatic" codes are detected in the normal manner.

4.6.9 Functional State Diagrams

This section contains diagrams (Figures 4-7, 4-8, 4-9, and 4-10) that define the status of the communication signals and functions at any one time and what causes a change from one state to another. The communication state table is encoded from these diagrams.

In these figures, a state is an area of an ON and/or OFF function. These areas relate to which signal is changed and to what value. Only the signals specifically set/reset within a state are affected. The N/C state is one that has no change to the outside world but the exit conditions are now different.

Each exit path from a state has an associated condition typed next to it. If this condition is true, the path is taken and the next state entered. Those exit paths with more than one condition require all conditions to be true unless specifically separated by "OR."

The half duplex diagrams contain Note A and Note A1.

Note A is not a state but a branch of the form shown in Figure 4-6.

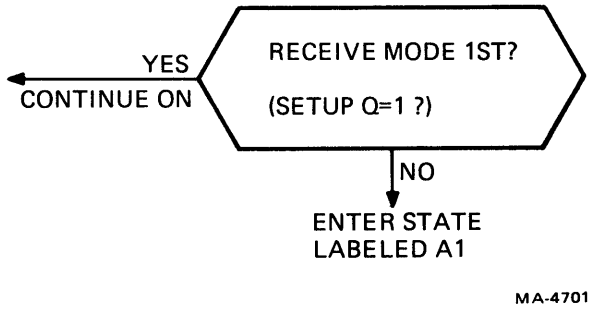


Figure 4-6 Note A Branch Form

Note A1 identifies the state entered after originating a call and the operator has selected to enter receive mode first.

As noted earlier, there are functions in parallel to the state diagrams such as disconnect request, line/local, etc.

The fundamental flow is clockwise when turning the line around in half duplex. When put on-line, the OFF 1, 2, 3 state is entered, requiring either RING or DSR in order to exit the state.

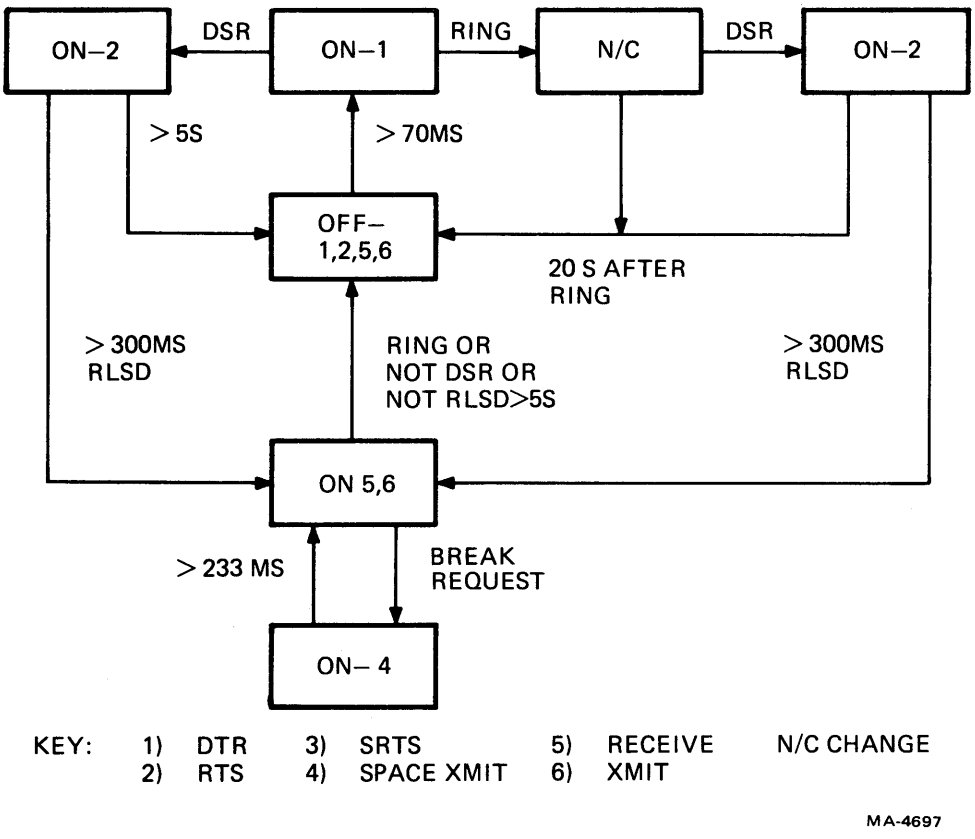
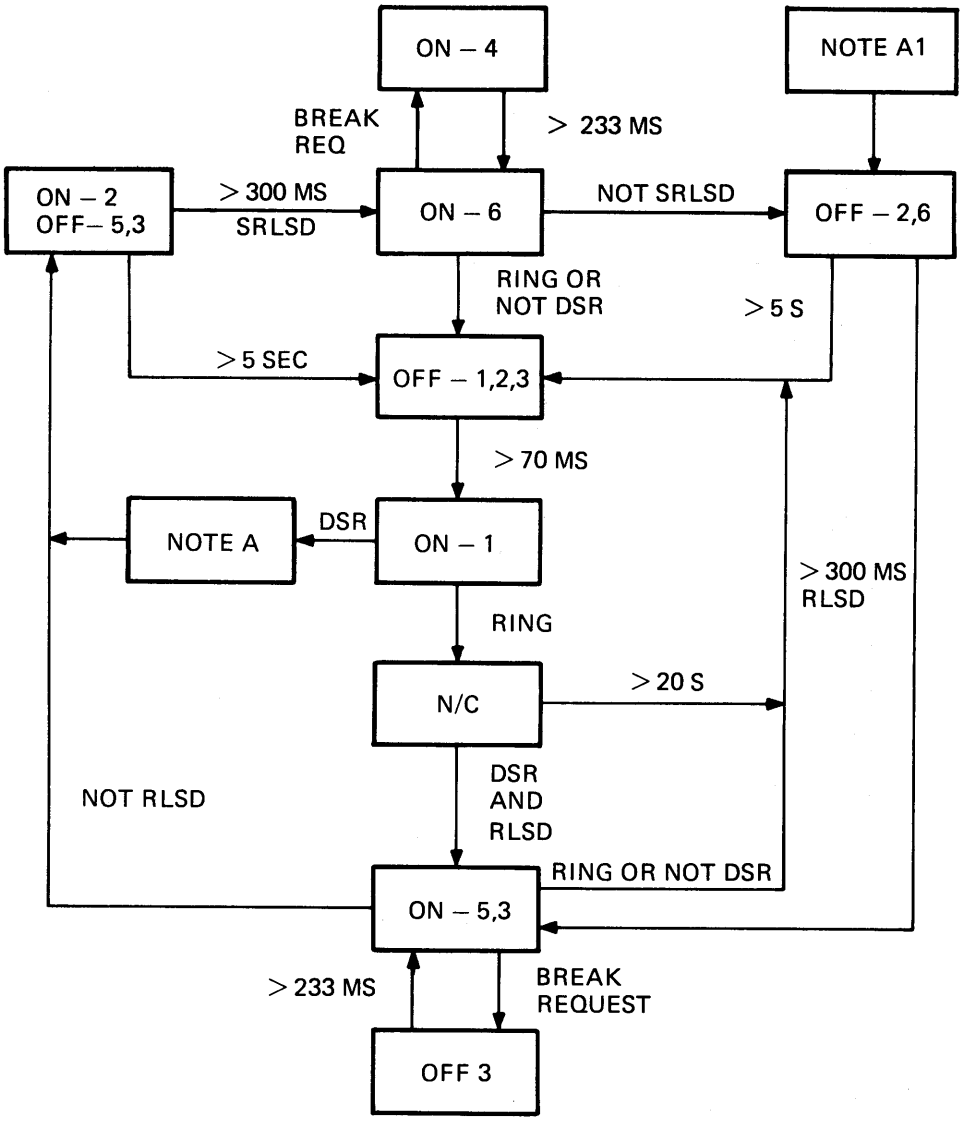


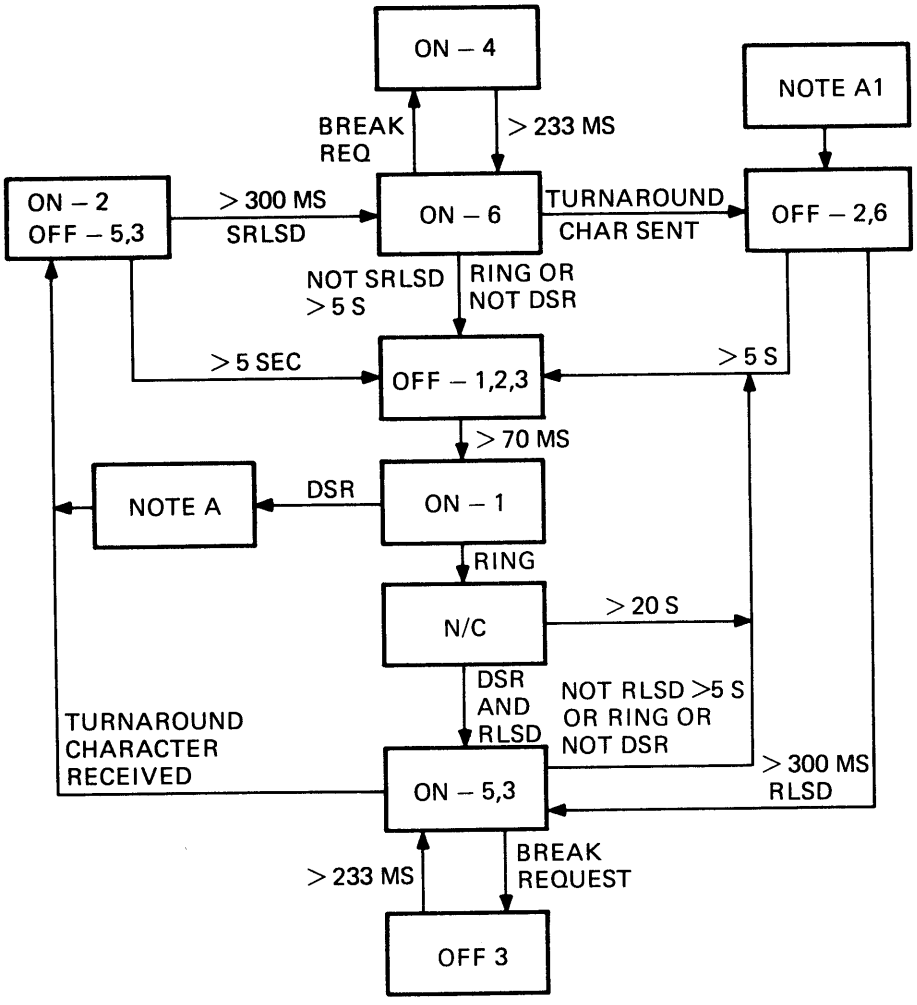
Figure 4-7 Full Duplex



KEY: 1) DTR 3) SRTS 5) RECEIVE N/C) NO CHANGE
2) RTS 4) SPACE XMIT 6) XMIT

MA-4696

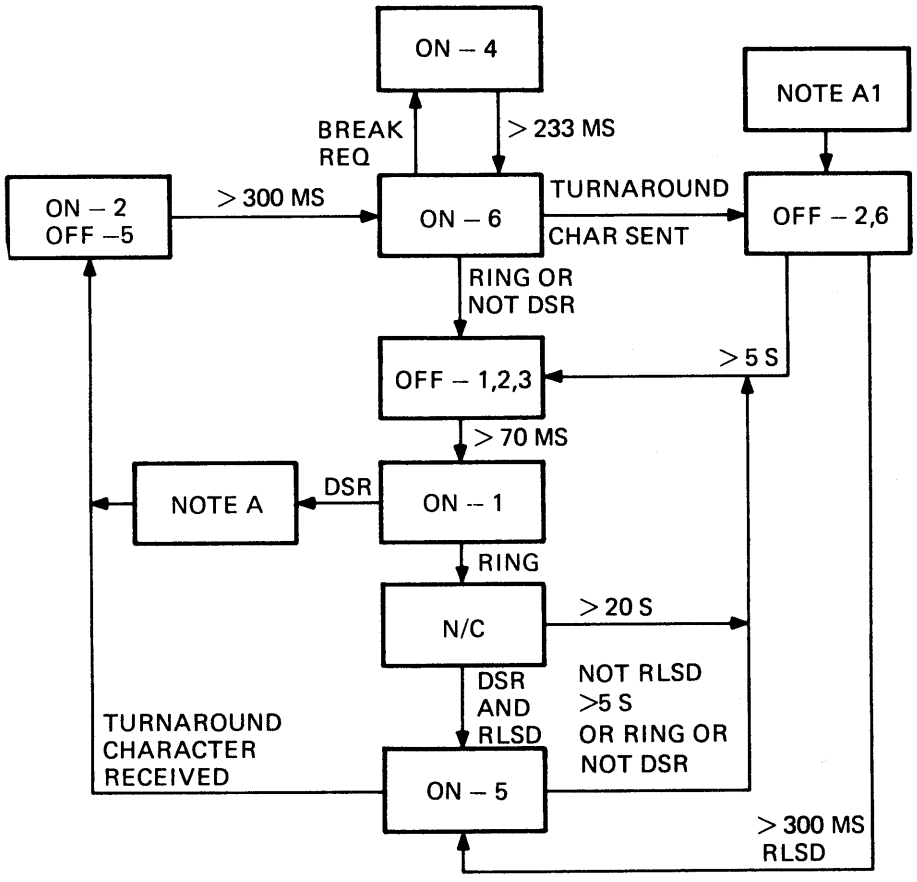
Figure 4-8 Supervisory Control



KEY: 1) DTR 3) SRTS 5) RECEIVE N/C) NO CHANGE
2) RTS 4) SPACE XMIT 6) XMIT

MA-4700

Figure 4-9 Coded Control with Reverse Channel



KEY: 1) DTR 3) STRS 5) RECEIVE N/C) NO CHARGE
2) RTS 4) SPACE XMIT 6) XMIT

MA-4698

Figure 4-10 Coded Control Without Reverse Channel

4.7 ESCAPE SEQUENCE PROCESSING

Escape sequences are processed by a table-driven parser. There are two major tables that are used in parsing escape sequences, a state transition table and a jump table.

4.7.1 Escape Sequence State Transition Table

The escape sequence state transition table consists of seven individual tables or elements numbered 1 through 7. At any given time, there is a pointer in RAM that points to one of the elements in this table.

Each element of the table is used (at different times) to scan the next character in an escape sequence. Basically, each element consists of one or more entries containing the following items:

1. The character against which the input character is to be compared (either for an exact match or lower than or equal for a "range" test).

2. A pointer to the next state transition table element to be used to test the next character of the escape sequence if the test in item 1 succeeds.

3. An index into a jump table that provides the address of a routine to be executed if the test in item 1 succeeds.

These entries can be broken down into two different types within each element of the table, depending on how the first byte of the entry is used.

1. The first byte of a table entry can be used to test for an exact match.

2. The first byte can be used as a "range" test. The range is satisfied if the input character from the escape sequence is less than or equal to the first byte of the table entry.

All of the exact match table entries come first in the table element, and can be in any order (but are arranged in ascending order for ease of maintenance.) The range tests come last in the element, and must be in ascending order. The last entry in each element is a range test entry for octal code 377 to force all of the remaining illegal characters through the parser. The two types of entries in each element are separated by a zero byte. Figure 4-11 illustrates how the entries can be organized within each of the table elements.

NOTE

The second type of element may have no exact matches, so it starts out with a mid-table separator (zero byte).

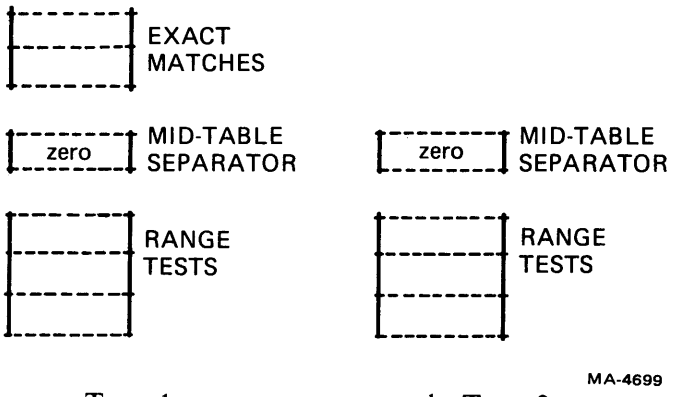


Figure 4-11 Entry Organization Within a Table Element

4.7.1.1 Element No. 1 – This is always the first table used in parsing an escape sequence. The parser is initialized to this element after a valid escape sequence has been parsed, whether the LA120 actually performs the designated function, or just skips over it. Prior to using this first table element, the background executive routine has detected the first escape character denoting the start of an escape sequence, and sets a flag so that any printable characters that follow will be routed to the parser. This first element does the following functions:

1. Strips out the escape sequences that are used to designate a G0 graphic set by finding the left parenthesis (octal code 050) after the initial escape character, and sets the pointer to element no. 7 to parse the next character anticipated in these sequences.

2. Jumps to the appropriate action routines required to process each of the 2-character escape sequences after finding the final character.

3. Strips out the 2-character escape sequences single shift 2 (SS2) and single shift 3 (SS3) and the pointer to element no. 2 to parse the next character anticipated in these sequences.

NOTE

SS2 and SS3 are nonlocking shift-outs to predefined graphic sets (like the S0 control code), but only apply to the next character in the sequence, then automatically shift back in again. Since the LA120 does not support SS2 and SS3, it simply skips over the next character.

4. Detects the start of four different types of control strings: device control string (DCS), operating system command (OSC), privacy message (PM), and application program command (APC). It sets the pointer to element no. 3 to parse the next character anticipated in these sequences.

NOTE

All control strings must be terminated by a string terminator (ST), which consists of another escape character followed by a backslash. Since the LA120 terminal does not support any control strings, it skips over all of this.

5. Detects the start of a control sequence introducer (CSI), and sets the pointer to element no. 5 to parse the next character anticipated in this type of sequence. It also prepares to accept multiple decimal input parameters.

NOTE

CSIs (in 7-bit mode) consist of the escape character followed by octal code 133 (left square bracket in US/UK), a number of multidigit decimal parameters separated by octal code 073 (semicolon), and terminated by a final character (octal codes 100–176).

6. Strips out all intermediate characters (octal codes 040 to 057), with the exception of 050 (left parenthesis), which is used when designating graphic sets; and sets the pointer to element no. 4 to process the next character anticipated in these sequences.

NOTE

The LA120 terminal does not support any three-character (or longer) escape sequences that use intermediate characters except as previously noted.

7. Any other character received while in element no. 1 must be a final character. It is considered to be a valid 2-character escape sequence not implemented on the LA120. [By default, this also includes a string terminator (ST) not immediately preceded by a control string.] Therefore, the current sequence is considered complete, and the parser is reinitialized.

4.7.1.2 Element No. 2 – This table element unconditionally skips the next character in the current escape sequence, after which it reinitializes the parser. It is entered after encountering the start of a single shift 2 (SS2) or single shift 3 (SS3) in element no. 1 or the start of a string terminator (ST) in element no. 3.

4.7.1.3 Element No. 3 – This table element is entered after finding the start of one of the four types of control strings (DCS, OSC, PM, or APC) in element no. 1. It unconditionally skips all input characters, both intermediates and finals, until an escape character is encountered signifying the start of a string terminator (ST). When the escape character is eventually found, it sets the pointer to element no. 2, which will skip over the final character of the string terminator. It also prevents the background executive routine from sending another escape character.

4.7.1.4 Element No. 4 – This table element is used to finish parsing an escape sequence in which an unexpected intermediate character was encountered before the final character. It can be entered from element no. 1 or element no. 7. It skips over any more intermediate characters until a final character is found, after which it reinitializes the parser.

4.7.1.5 Element No. 5 – This table element is used to process a control sequence introducer (CSI). It is originally entered from element no. 1 after an escape character followed by a left square bracket (octal code 133) has been encountered. The remainder of the CSI escape sequence will "loop" in this table element until either the final character is found, or an error condition is detected. Errors are considered to be characters in the octal code range of 072 to 077 (for example, a question mark or colon); or an intermediate character (octal codes 040 to 057). Errors will cause the pointer to be set to element no. 6 to properly complete the parsing of this type of escape sequence.

ASCII digits that are encountered are considered to be part of a decimal parameter, and are appropriately processed. Semicolons that are found here are separator characters used between the parameters, and are so processed.

If a final character is received by this table element, it jumps to the associated "action" routine. After the final character "action" routine has completed, the parser is reinitialized.

4.7.1.6 Element No. 6 – This table element is entered after finding an illegal character while processing a CSI in element no. 5. The purpose of this table is to properly complete the parsing of this type of escape sequence by skipping all additional characters until a final character (octal codes 100 to 176) is found, after which the parser is reinitialized.

4.7.1.7 Element No. 7 – This table element is entered from element no. 1 after finding the start of a Designate G0 Graphic Set escape sequence, which is an escape character followed by a left parenthesis. If an additional intermediate character is found, the pointer will be set to element no. 4 to finish parsing the remainder of the sequence correctly. If a final character is found corresponding to one of the national character sets supported by the LA120, this table element will cause a jump to the "action" routine that changes the printer character set. If any other final characters (octal codes 060 to 176) are encountered, they will cause the parser to be reinitialized.

4.7.2 Escape Sequence Jump Table

The second major table used in parsing escape sequences is the escape sequence jump table. It consists of a number of entries containing the addresses of routines to be executed when an exact match or range test is satisfied by one of the table entries in one of the seven elements of the state transition table. It is these routines that do the actual "work" required by the escape sequence, such as changing variables, setting flags, etc.

4.7.3 Escape Sequence Parser

The parser is the set of routines that does the parsing or breakdown of an escape sequence. Three routines control the logical flow of parsing an escape sequence:

1. The routine that begins the parsing of escape sequences.
2. The routine that initializes the parser.
3. The routine that controls the flow through the state transition tables.

4.7.3.1 Begin Parsing – The begin parsing routine is called by the background executive routine that is normally processing characters when it finds the first escape character that denotes the start of an escape sequence. This routine starts the parsing routine by setting a flag that causes subsequent printable characters to be routed to the parser until the escape sequence has been terminated properly.

NOTE

Even though the parser is enabled, control characters (octal codes 000 to 037) are skimmed off and acted upon, and never get to the parser. With the exception of device control strings (DCSs), if another escape character is received before the current escape sequence has completed, the current escape sequence is aborted and a new escape sequence will be started with the second escape character. This is handled outside the parser by the executive routine.

4.7.3.2 Initialize Parser – The initialize parser routine reinitializes the escape sequence parsing mechanism to start a new escape sequence as soon as the background executive routine encounters the next escape character. This routine only does two things: it resets the parser flag (see Begin Parsing, Paragraph 4.7.3.1) and it resets the state table pointer to element no. 1.

This routine is called under the following circumstances:

1. During power-up initialization.

2. After an escape sequence has been properly terminated or completed.

3. When a cancel or substitute character (or delete character in local mode) is encountered in the input stream [(or when another escape character is encountered if not processing a device control string (DCS)].

NOTE

The substitute character may be generated internally if a communications error is detected. Terminating an escape sequence when an error is encountered prevents the terminal from hanging in a nonprinting state.

4.7.3.3 Flow Controller – The flow controller is a very short routine. The reason for this is that the majority of the escape sequence parsing logic is contained in the state transition tables.

The flow controller consists primarily of two table look-up routines. When it is entered, the state pointer is pointing to one of the seven elements of the state transition table that has been assigned to examine the next (or first) character anticipated in the current (or next) escape sequence. The first of the two table look-ups is looking for an exact match against the input character. This look-up is terminated by either a "hit" (i.e., an exact match), or by finding the "mid-table" separator, which is a zero byte.

If the mid-table separator was encountered first because there was not an exact match, the flow controller then proceeds to the second table look-up. The second table look-up is similar to the first table look-up, except that a "hit" is considered to be any input character that is lower than or equal to the first byte of a table entry. These "range test" table entries are in ascending order and are constructed so that all input characters will get a "hit," including all erroneous ones.

Having gotten a "hit" with the first byte of a table entry from one of the seven escape tables, either as the result of an exact match or a range test, the flow controller picks up the new pointer value from the "matching" table entry. This designates the proper table element to use for the next character in the escape sequence.

Next the flow controller picks up the jump index from the "matching" table entry, uses it to index into the jump table, and jumps to the appropriate "action" routine.

4.7.4 CSI Parameters

The rules for CSI escape sequences allow any number of parameters: but in the LA120, only the last 16 are saved. Of the 16 (or less) that remain, they are processed in first-in, first-out (FIFO) order.

4.7.5 Control Strings

The following escape sequences are considered control strings: device control string (DCS), operating system command (OSC), privacy message (PM), and application program command (APC). The LA120 does not support any control strings, but it will parse them correctly and skip over them. When parsing control strings, the background executive routine will pass an escape character through to the parser which it would not normally do. The escape character is needed to terminate the control string.

4.7.6 Final Character Perform-Function Routines

The actual performance of an escape sequence must wait until the "final" character has been received, parsed, and interpreted before the actual desired function can be performed. The final character action routines contained in the LA120 follow:

Horizontal Tab Set
Vertical Tab Set
Clear Horizontal Tab at Active Column
Clear Vertical Tab at Active Line
Clear All Horizontal Tabs
Clear All Vertical Tabs
Enter Alternate Keypad Mode
Enter Normal Keypad Mode
Index Down One Line
Next Line
Horizontal Position Absolute
Horizontal Position Relative
Device Attributes (Product ID)
Vertical Position Absolute
Vertical Position Relative
ANSI Tab Clears (0–4)
Set Mode (LNM)
Reset Mode (LNM)
Set Top and Bottom Margins
Set Left and Right Margins
Set Number of Lines Per Page
Set Multiple Horizontal Tabs
Set Multiple Vertical Tabs
Set Horizontal Pitch (char/in)
Set Vertical Pitch (lines/in)
Designate G0 Graphic Set

4.8 SET-UP COMMAND PROCESSING

SET-UP commands allow many features of the LA120 terminal to be controlled by the operator via the keyboard. Rather than requiring the operator to enter a myriad of "free-form" commands with all of the attendant checking that would be required by the firmware, it was decided to present the operator with a list of only the valid choices. Thus, the requirement of having to validate free-form commands was totally eliminated. The method chosen to present these selections was to display them sequentially in the 4-digit, 7-segment display. The last choice displayed is the current selection. Entering an invalid SET-UP command will cause the audible alarm to sound.

4.8.1 SET-UP Command Implementation

To perform a SET-UP command, the operator must first place the LA120 in SET-UP mode using the SET-UP key with or without the control key. While in SET-UP mode, the SET-UP LED indicator will be flashing to confirm that the terminal is in SET-UP mode. In SET-UP mode, most of the keys on the keyboard can be used to perform SET-UP commands.

While in SET-UP mode, printable characters go to the SET-UP routines, and control characters are routed to the printer. SET-UP commands fall into two general categories:

1. Multiple choice SET-UP commands, and
2. Immediate action SET-UP commands.

4.8.1.1 Multiple Choice SET-UP Commands – If a SET-UP command has multiple choices, executing the SET-UP command the first time displays the current setting of that function without changing it. Executing the same SET-UP command a second or subsequent number of times without leaving SET-UP mode (which is defined as releasing or unlocking the SET-UP key) will cause the display to cycle through, one at a time, all of the valid choices for that command. After reaching the end of the selection list, the display will recycle and start over again from the beginning.

The alphabetic keys ("A" to "Z" in US/UK) are used to perform multiple choice SET-UP commands, with the exception of "I" (Initialize) and "T" (printing self-Test). Alphabetic SET-UP commands work the same, whether shifted or unshifted. Multiple choice commands are broken down into two logical groups. All of the members of one group only have two choices. The number displayed in the display is either a zero or one, which in general means that the selected function is either disabled (0) or enabled (1). The flags for these commands are stored as consecutive bits in RAM.

Multiple choice commands with more than two possible selections use an entire byte to store their current setting. The value displayed is usually the value of this "index" byte plus one. These indices are stored in consecutive bytes in RAM. There are four exceptions to displaying the index value plus one. They are:

1. Receive Baud Rate (SET-UP 0)
2. Transmit Baud Rate (SET-UP Shifted 0)
3. Horizontal Pitch or char/in (SET-UP H)
4. Vertical Pitch or lines/in (SET-UP V)

Since the values being selected by these commands are displayable as decimal integers (suitably rounded in the case of some horizontal pitches), these values are displayed instead of the index numbers. For example, if the "H" SET-UP command results in a display of "10," that means 10 char/in, not the tenth choice in the list. Similarly a SET-UP "0" display of 1200 means a receive speed of 1200 baud.

4.8.1.2 Immediate Action SET-UP Commands – In an immediate action SET-UP command, the operator is not concerned with the current setting (if any). The operator just wants the terminal to do something immediately. For example, to start the printing self-test, the operator uses SET-UP "T." Sometimes the active position is an implied input to the action routine, such as in setting a horizontal tab stop (SET-UP "1"). The top row of the keyboard is assigned to the immediate action commands with the exception of SET-UP "0" (receive baud rate) and SET-UP shifted "0" (transmit baud rate), which are multiple choice.

Normally, the 4-digit, 7-segment display will contain the column number of the next character to be printed. However, if the SET-UP key is pressed (or locked) and the SHIFT key is also pressed, it will contain the current line number. For this reason, all vertical tab and margin commands have been implemented as shifted. This makes them easier for the operator to set up. If the TOF (top-of-form) command (SET-UP 4) is done shifted (and the top margin is clear), the operator will see the current line number change to "1."

The nonprinting self-test (SET-UP shifted >) was assigned to the greater than character because on both the US/UK and all other foreign keyboards this character has to be shifted to be entered, thus reducing the likelihood of accidentally triggering this seldom-used command. The request for current status message (SET-UP 8) cannot be entered shifted. This was done to eliminate some possible firmware interactions concerning the display that would have required additional ROM to solve. The status message is "semi-recursive," and is described separately below.

The SET-UP routines also respond to numeric codes from the optional keypad, since the SET-UP commands are defined by code rather than key position. The following three keys are independent of SET-UP mode: LINE/LOCAL, LOCAL FORM FEED, and LOCAL LINE FEED. The HERE IS key is also controlled by the keyboard handler.

4.8.2 SET-UP Handler

Whenever the operator enters SET-UP mode, the background executive program invokes the SET-UP handler for each character of the command whose octal code is greater than 040 (the space character) and less than 177 (delete/rubout). If the character received is not from the US/UK keyboard, the SET-UP handler translates the character into a code it would have received if the key had been struck on the US/UK keyboard such that top row SET-UP commands, whose legends appear on the trim strip, do not change position from character set to character set. Alphabetic SET-UP commands are not translated, so they are always identified by letter, not by key location.

The SET-UP handler's tables handle only octal codes 041 through 137 (which includes the uppercase letters). Octal codes 140 through 176 (which include the lower case letters) are translated to the 041 through 137 codes. This is done to make the SET-UP commands independent of the SHIFT key or the CAPS LOCK key. For example, the small letter "a" is translated to the uppercase "A."

The SET-UP characters are now used to access the first level command table. This table contains information that identifies the class of SET-UP commands to which this command belongs. It is possible that no SET-UP command at all is associated with this character (identified by a zero byte in the table), in which case a bell is rung. Otherwise, the command will select one of several general-purpose subroutines. Similar SET-UP commands that are in the same category are handled by the same subroutines. Basically there are:

1. groups of commands that are on/off or 1/0 operations,
2. groups of commands that are multiple choices (like baud rates of 300, 600, 1200, etc.), and
3. groups of immediate action commands that do not alter parameters at all (like self-test, save/recall, set tabs, etc.). These do not show results on the LED display.

Once the SET-UP command has identified the type of action, subroutines are called to implement the action. If the command is one of the on/off or multiple-choice type, then the subroutine determines whether the character has been typed once or several times in a row during this SET-UP sequence. The first time a SET-UP command is typed, the value of the parameter's current setting is displayed. Upon repeats of the command, the value of the parameter is altered and displayed on the LED display. For each type of SET-UP command, there is a subroutine that handles the first input of the command and a subroutine that handles subsequent SET-UP change commands. Finally, certain SET-UP commands need to call an initialization subroutine every time the SET-UP command parametric value has been altered (like moving the head slightly after changing horizontal pitch).

In summary, the basic input to the SET-UP command handler is a character coming from the keyboard. The other input to the SET-UP handler is the current parameter from the RAM. The SET-UP handler output is the new value of the operating parameter. It also outputs new data to the numeric display.

4.8.3 SET-UP/Keyboard Handler Relationship

Part of the work associated with SET-UP commands is performed by the keyboard handler. The process of entering or leaving SET-UP mode is detected by the keyboard handler, which also controls the corresponding LED indicator. While in SET-UP mode, printable characters go to the SET-UP parsing routines, and control characters are routed to the printer. All other action associated with the SET-UP functions is performed by the SET-UP handler with the exception of entering the answerback message, which is performed by the keyboard handler. The SET-UP commands are code sensitive rather than key-position dependent. Consequently, the SET-UP parsing routines also respond to numeric codes from the optional keypad. The following three keys are independent of SET-UP mode: LINE/LOCAL, LOCAL FORM FEED, and LOCAL LINE FEED. The HERE IS key is also controlled by the keyboard handler.

4.9 CHARACTER PROCESSING

4.9.1 Character Reception

When a character is received and fully assembled by the USART, an interrupt is sent to the microprocessor. This interrupt is serviced by the character reception routine. This routine reads the character from the USART and checks for parity error. If a parity error has been detected by the USART, the character is converted to a SUB character. If the character is neither NUL nor DEL (the all zeros and all ones codes), it is stored in the input buffer in RAM to be processed by either the input buffer scanner or the background executive routine.

4.9.2 Background Executive

Characters are processed by the background executive routine. The source of characters may be either the input buffer or the local character slot, which may contain characters generated by the keyboard either in local or SET-UP mode, or if local echo is enabled. The LOCAL FORM FEED and LOCAL LINE FEED keys also use the local character slot, even when on-line.

When the background executive has a character to process, depending on the mode and state of the terminal, it forwards it to one of the following processes:

1. Print Line Builder
2. Escape Sequence Parser
3. SET-UP Command Handler
4. Answerback Entry Handler

Processing of escape sequences and SET-UP commands has been covered in detail earlier in this chapter. Building print lines and processing answerback entry is covered in the following paragraphs.

4.9.3 Print Line Builder

Printable characters are routed to the print line builder. Upon receiving a character, the first task performed by the print line builder is to determine if it is a control character or a graphic character. If the character is a control character, it is used to index a table of control routine addresses and an appropriate control routine is executed. For example, routines are required for handling horizontal and vertical tabs. They call other subroutines, which search a tab bit map table to find the first tab set past the current printing position. Each routine returns with a value that specifies what the active line or column should be when the routine has finished. If the character is a graphic character, it is moved into one of two print line buffers, after a certain amount of processing.

The processing of a graphic character requires look-up tables to convert certain codes in the various national languages to an internal code that will point to dot matrix representation of the character. Also, if the printer is in the alternate character set, the character is converted so that it points into the alternate character set dot table.

The print line buffer is then checked to ensure that the active column (the column in which the character is supposed to be deposited) is within the left and right margins. If it is outside these margins and auto newline is enabled, the character is not deposited in the current print line. The current print line is forced to be printed followed by a line feed. The active column is returned to the left margin. The result is as though a carriage return and line feed sequence was received just before the character. The character is then put into the next print line buffer at the left margin. If the character is outside the margin and the auto newline is disabled, the presence of the character means that printing is being attempted beyond the right margin. The character is discarded and a low frequency bell is sounded.

Before attempting to deposit a character in the print line buffer, a test is made to be sure that a character is not already in the buffer at the active column. If there is, an overprint condition exists. Detection of an overprint prevents adding any more characters to the print line buffer until it has been emptied (printed). A new 217-byte print line buffer is made available and the character is written into the new buffer at the active column.

Depositing a character in a print line buffer causes the active column to be incremented by one. Tests are performed to determine if the right margin warning bell should be sounded and to see if the right margin has been reached. Other types of information stored in the print line buffers have to do with producing vertical motion between lines and ringing the bell. Thus, a complete print line contains three types of information: bell count, line feed step count, and character codes. The character codes are positioned in the print line buffer at the same spot that they would appear on the paper. When a print line buffer is full or its printing is forced (due to overprinting, etc.), then the print line buffer is given up by the print line builder and traded for an empty print line buffer. When the full print line buffer is acted on by the printing routines, it has to be prepared before it can be used to start printing characters.

When a line is to be printed, a decision has to be made as to which direction the line is to be printed. A line is always printed forward unless it has vertical motion both before and after it. Also, a line is never printed in reverse which contains overprinting (i.e., a / and an 0 to make Ø). Recall that a print line buffer is forced to print when overprinting occurs. When forced to print, the buffer is swapped with the empty buffer and the printing routines take over. Since the forced-to-print buffer is not full, it does not have a vertical motion code at the end of the buffer. It has been sent to the printer because of overprinting, not because there is a terminating line feed character at the end of the line. Consequently, the direction decision algorithm would find that this print line buffer did not have vertical motion both before and after it.

If the print line buffer is a complete line bracketed by vertical motion on both sides, then the print direction is decided by the direction algorithm. The algorithm remembers the last column printed on the previous line. If the previous line had characters from columns 1 to 100 and was printed from left to right, then the last column was 100. If the next line has columns from 25 to 125, the direction algorithm finds which of the limits is closest to the last column of the previous line. In this example, 100 is closer to 125 than 25; therefore, the new print line is printed from right to left.

After the print direction has been established, it is necessary to look at the print line buffer in the interval between the left-most and right-most characters to determine if a long sequence of empty positions is in the print line. Otherwise, the interrupt routine would have to search through as many as 215 empty positions to locate the next printable character. To prevent this, the firmware performs a housekeeping routine that scans the print line and converts the empty positions to real spaces (octal code 40) or inserts a special tabbing command. The tabbing commands are interpreted by the printing routine to indicate that there is a long sequence of white space; consequently, stop printing and start slewing the print head. The routine also specifies where to stop slewing and again start printing characters. As an example, suppose a print line has characters from columns 1 to 10, then blanks from columns 10 through 100, and characters again from columns 100 to 110. The housekeeping routine would search the columns until it got to the blank in column 11. It would then start to count blanks. If it finds five empty positions in a row, the routine leaves the printing mode and executes a high speed slew to the next column in which characters will be found. When it finds the next character (in column 100), it will go back to column 11 and write a -1 value in that slot. This flags the column as the beginning of a high speed slew. During printing, when the print routine encounters the -1, it causes the print routine to stop and pass control to a high speed slew routine. The next column (12) contains the address of the column (100) in which is located the next printable character. Thus, the "look-ahead" housekeeping routine functions to speed up the rate of printing.

When the print line buffer is completely filled and the printing interrupt routine is free, the filled print line buffer is passed over to the background printing routine. If there are any bells specified in the print line, the code is passed over to the bell handler routine, which will sound the bell at a 400 or 2400 Hz rate at the proper time. If there are any line feed steps to be performed, the line feed routine will be called, at the appropriate time, to index the stepper motor. Any characters contained within the print line will call the printing routines, which will be scheduled at the appropriate column, and printing will occur. When the print line buffer has been completely processed by the printing routines, an executive routine pointer will exchange the empty print line buffer for the buffer just filled. The empty buffer will be zeroed and returned to the background where it will start accepting new characters.

4.9.4 Answerback Entry Handler

When the LA120 is in SET-UP mode and CTRL-HERE IS has been typed, a flag is set that causes the background executive to pass all characters to the answerback entry handler. This routine builds a string of characters in RAM to be used as the answerback message. The characters are echoed to the printer by calling the print line builder. Control characters are echoed as "^" followed by the corresponding character from one of the uppercase alphabetic columns of ASCII, which has a code value 64 greater than the control character being entered in the string.

If more than 30 characters is entered, or SET-UP mode is exited during answerback entry, the string of characters is not stored in the non-volatile memory, although it does function as the answerback string until power is cycled or answerback entry is performed again.

If CTRL-HERE IS is typed while performing answerback entry, the answerback message is stored in the non-volatile memory and subsequent characters are treated as SET-UP commands until SET-UP mode is exited.