GT40/GT42 user's guide

decgraphic11
decgraphic11
decgraphic11
decgraphic11
decgraphic11
decgraphic11
decgraphic11

digital

EK-GT40-OP-002

GT40/GT42 user's guide

digital equipment corporation • maynard. massachusetts

1st Edition, June 1973
2nd Printing, September 1973
3rd Printing (Rev), November 1974
4th Printing February 1975

CONTENTS

CONTENTS (Cont)

ILLUSTRATIONS

ILLUSTRATIONS (Cont)

TABLES

GT40/GT42 GRAPHIC DISPLAY TERMINAL

1.1 PURPOSE AND SCOPE

This guide describes the operation of the GT40 and GT42 Graphic Display Terminals. The following information is included: start-up procedures, equipment specifications, programming techniques, interfacing, and a description of the ROM Bootstrap.

1.2 GENERAL DESCRIPTION

The GT40/42 Graphic Display Terminal (Figures 1 through 5) is a high performance graphic display system that operates through a PDP-11/10 computer. The GT40/42 is designed for applications that require both a visual display and a computation capability. The system can display either alphanumeric information, graphic data such as drawings, diagrams, and patterns, or any combination of these. It is particularly valuable for displaying dynamic, fast-changing data such as waveforms. The GT40/42 can function as a general purpose computer when not performing as a display terminal. In this nondisplay mode of operation, it can operate as a stand-alone system or initiate communications with a host computer as part of a computer network.

1.3 SYSTEM ORGANIZATION

The GT40/42 consists of eight basic components organized to form the system described above. These components are:

- Central Processor Unit (CPU)
- Display Processor Unit (DPU) in which is included the Bootstrap Read Only Memory (ROM)
- Communications Interface Module
- Memory
- Keyboard
- Cathode Ray Tube (CRT) Monitor
- Light Pen
- Power Supply

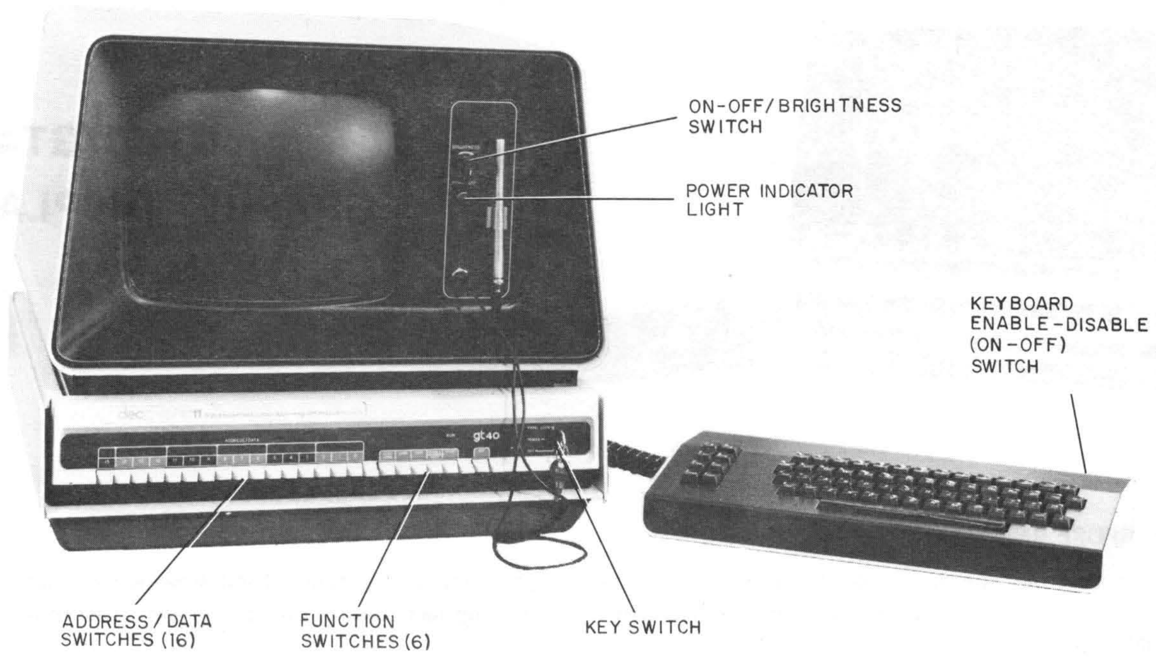


Figure 1 GT40 Graphic Display Terminal

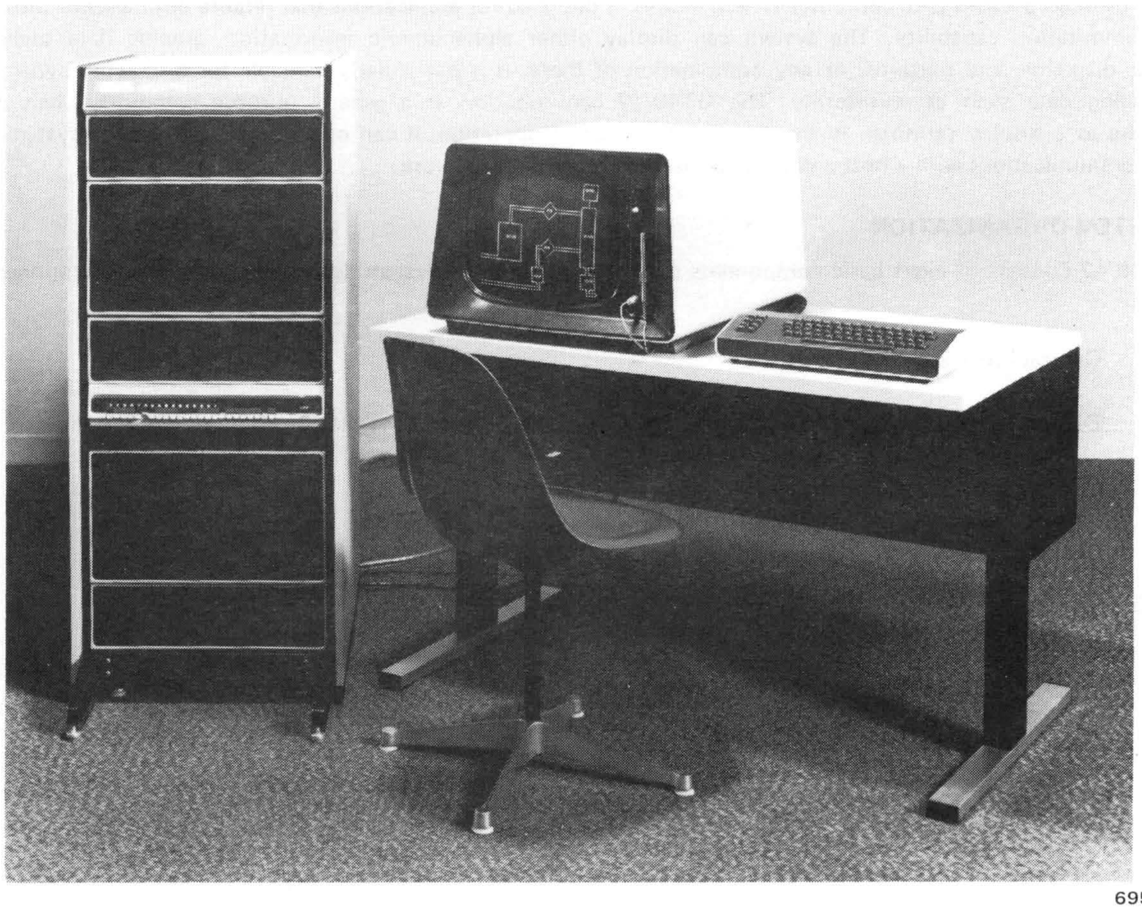


Figure 2 GT42 Graphic Display Terminal

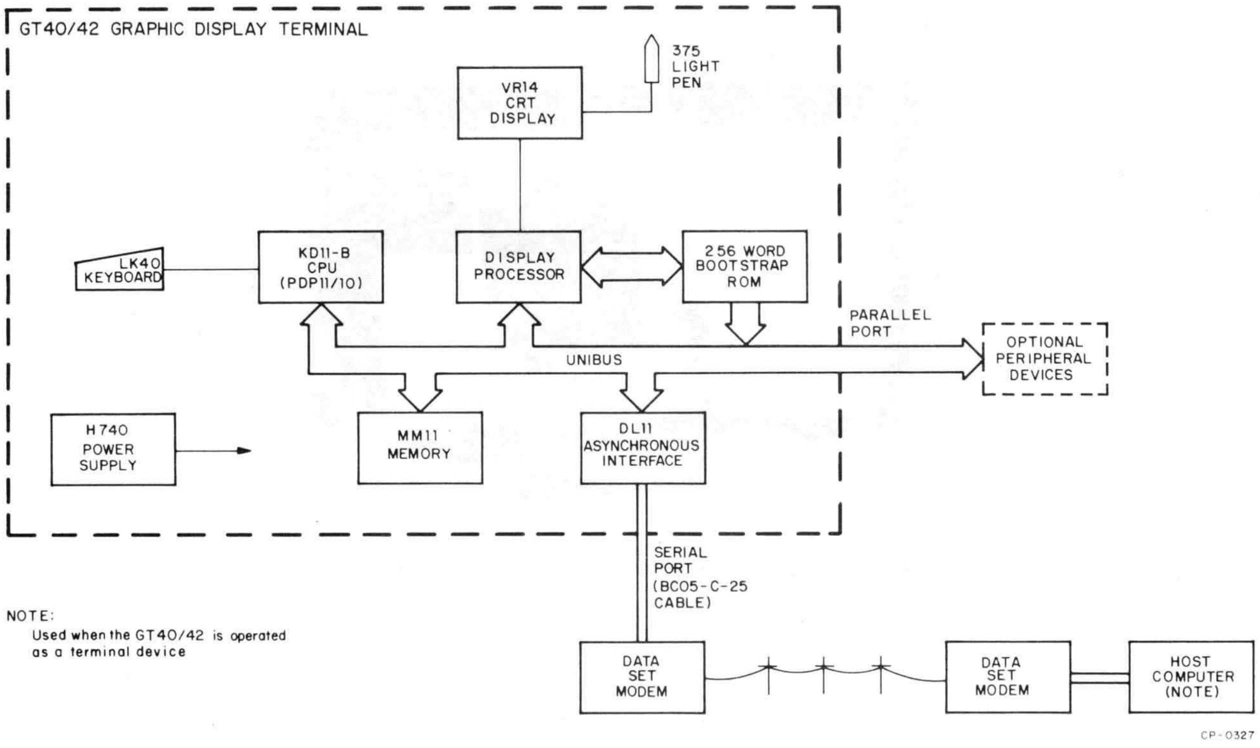


Figure 3 GT40/42 Graphic Display Terminal, Block Diagram

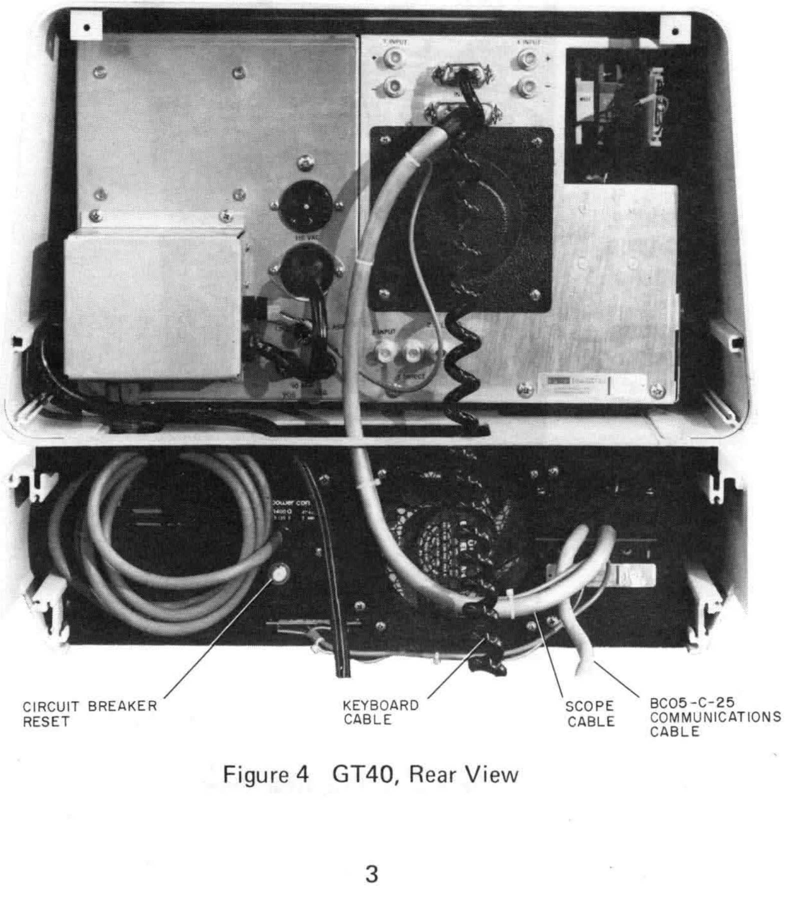
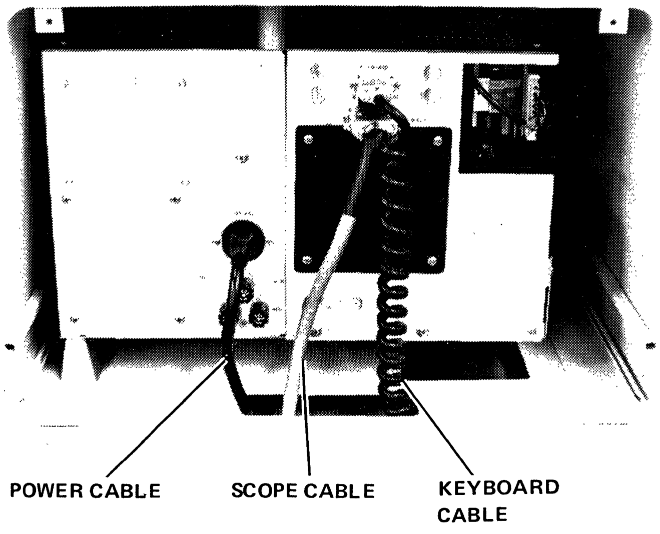


Figure 4 GT40, Rear View



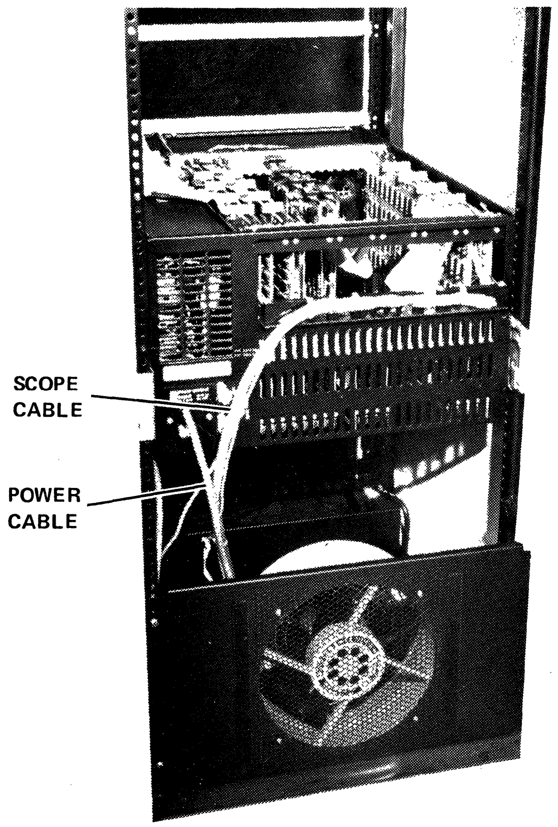


Figure 5 GT42, Rear View

1.4 SYSTEM OPERATION

The GT40/42 is a stable system that requires only minimum adjustments because it employs a combination of digital and analog techniques as opposed to analog circuits alone. The vector function operates efficiently, providing a good compromise of speed and accuracy and assuring a precise digital vector calculation. The presentation and accumulation of vectors means that every point of a vector is available in digital form.

During plotting, the end-point position is automatically retained, preventing accumulated errors or drift. Four different vector types – solid, long dash, short dash, and dot dash – are possible through standard hardware.

The GT40/42 character generator has both upper and lower case capability with a large repertoire of displayable characters. The display is the automatically refreshing type rather than the storage type so that a bright, continuous image, with excellent contrast ratio, is provided during motion or while changes are being made in the elements of the picture. A hardware blink feature is applicable to any characters or graphics drawn on the screen. A separate line clock in the display permits the GT40/42 to be synchronized to the line frequency. Scope resolution is precise enough to allow overprinting.

The terminal includes logic for descender characters such as "p" and "g," positioning them correctly with respect to the text line. In addition to the 96 ASCII printing characters, 31 special characters are included which are addressed through the shift-in/shift-out control codes. These special characters include some Greek letters, architectural symbols, and math symbols. Characters can be drawn in italics simply by selecting the feature through the status instruction bit. Brightness and contrast are such that the scope can be viewed in a normally lighted room.

The instruction set consists of four control-state instructions and five data-state formats. The control instructions set the mode of data interpretation, set the parameters of the displayed image, and allow branching of the instruction flow. Data can be interpreted in any of five different formats, allowing tasks to be accomplished efficiently from both a core usage and time standpoint. The graph/plot feature of the GT40/42 automatically plots the x or y values according to preset distances as values for the opposite axis are recorded.

1.5 EQUIPMENT SPECIFICATIONS

The GT40/42 Graphic Display Terminal operating requirements and physical characteristics are listed by component in the following paragraphs. Refer to Volume 2 of the *GT40 Graphic Display Terminal Maintenance Manual* for the specifications pertaining to the KD11-B Processor (PDP-11/10).

Display Processor

Instruction Word Length	16 bits
Raster Definition	10 bits

Viewable Area	x = 1024 raster unit (1777_8)
	y = 768 raster units (1377_8)

Paper Size	12 bits
Hardware Blink	Programmable
Hardware Intensity Levels	8
Line Frequency Synchronization	Hardware programmable
Character Font	6 X 8 dot matrix
Characters/Line	73

Number of Lines	31
Character Set	96 ASCII – upper and lower case plus 31 specials (Greek letters, math symbols, etc.) (Refer to the appendix)
Control Characters	Carriage return Line feed Backspace
Bell Tone	Programmable
Italics	Hardware programmable
Line Type	Solid Long dash Short dash Dot dash
Data formats	Character (2 char/word) Short Vector (1 word) Long Vector (2 words) Point (2 words) Relative Point (1 word) Graphplot x/y (1 word/pt)
DPU Instructions	Set Graphic Modes Jump No operation (NOP) Load Status Register A Load Status Register B

DL11 Communications Interface Operating Specifications

Data Input and Output	Serial data, EIA and CCITT specifications compatible with Bell 103 and 202 Data Sets
Data Format	1 start bit 5, 6, 7, 8 data bits 1, 1.5, or 2 stop bits, odd, even or no parity.
Power Required	1.8A @ +5V 0.150A @ –15V 0.050A @ +9 to +15V
Cable Length EIA	All baud rates: 50 ft (15.24m)
Noise Margin EIA	5V

MM11 Core Memory (refer also to Volume 2 of the *GT40 Graphic Display Terminal Maintenance Manual*).

Type	Magnetic core, read/write, coincident current, random access
Organization Capacity	Planar, 3D, 3-wire

Access Time	
DATI	400 ns
DATIP	400 ns
DATO, DATOB	200 ns
Cycle Time	
DATI	900 ns
DATIP	450 ns
DATO, DATOB (PAUSE L)	900 ns
DATO, DATOB (PAUSE H)	450 ns

LK40 Keyboard

Number of Keystations	58 (Major board) 8 (Minor board)
Encoding Format	1968 USASCII
Number of Codes	Either 96 or 128 codes (internal switch controllable,
Output Data Format	8-bit ASCII 1 start bit 7 data bits 2 stop bits
Baud Rate	Approximately 150 baud
Output Signal	20-mA current loop
Bell	Tone generator
Controls	Enable/Disable transmit

CRT Monitor

Viewable Area	
GT40	6.75 X 9 in. (17.145 X 22.86 cm)
GT42	8.5 X 11 in. (21.590 X 27.940 cm)
Brightness	> 30 fL (measured using a shrinking raster technique)
Contrast Ratio	> 10:1
Phosphor Type	P39 doped with IR
Pincushion	±1% of full scale to best fit line
Spot Size	< 20 mils inside the usable screen area at a brightness of 30 fL [Full Width at Half Maximum (FWHM)]

Jitter	$< \pm 1/2$ spot diameter
Repeatability	$< \pm 1$ spot diameter (repeatability is the deviation from the nominal location of any given spot)
Gain Change	From a fixed point on the screen, less than ±0.3% gain change for each ±1% line voltage variation
Temperature Range	0° to 50°C (operating)
Relative Humidity	10 to 90% (noncondensing)
Linearity	Maximum deviation of any straight line will be < 1% of the line length measured perpendicular to a best-fit straight line
Deflection Method	Magnetic (70° diagonal deflection angle)
Focus Method	Electrostatic
High Voltage	10.5 kV dc nominal (voltage proportional to input line voltage). Supply is self-contained and equipped with a bleeder resister.
Shielding	CRT is fully enclosed in a magnetic shield.
Overload Protection	Unit is protected against fan failure or air blockage by thermal cutouts. Power supply and amplifiers are current limited. Phosphor protection is provided against fault conditions.

Light Pen

Length	5.0 in. (12.7 cm)
Diameter	0.45 in. (tapered to 0.35 in.) (1.143 cm) (0.889 cm)
Light Sensing	Phototransistor
Connector	Phono Plug
Signal Amplification	G840 Light Pen Amplifier module in VR14 CRT Display

Power Supply
Refer to Volume 2 of the *GT40 Graphic Display Terminal Maintenance Manual* for a detailed list of power supply specifications.

Environmental

Shock, Nonoperating	DEC STD 102, 205 at 30 ± 10 ms half-sine
Vibration, Nonoperating	DEC STD 102, Vertical 1.89 G rms 10 – 300 Hz
Operating Ambient Temperature	DEC STD 102, Class A, 60° – 95°F (16° – 35°C)
Relative Humidity (noncondensing)	DEC STD 102, Class 2, 20 – 80%

Physical

Weight

	GT40	GT42
CRT Monitor	80 lb (36.24 kg)	85 lb (38.55 kg)
Processor Cabinet	60 lb (27.18 kg)	275 lb (124.74 kg)
Keyboard	6.25 lb (2.83 kg)	6.25 lb (2.83 kg)

GT40 Size

	Height	Width	Depth
CRT Monitor	12.5 in. (31.75 cm)	19.75 in. (50.165 cm)	22.25 in. (56.515 cm)
Processor Cabinet	5.25 in. (13.335 cm)	19.75 in. (50.165 cm)	23.25 in. (59.055 cm)
Keyboard	3.0 in. (7.62 cm)	15.625 in. (42.227 cm)	6.625 in. (16.827 cm)

GT42 Size

	Height	Width	Depth
CRT Monitor	15 in. (38.10 cm)	21.5 in. (54.61 cm)	27 in. (68.58 cm)
Processor Cabinet	50 in. (127.00 cm)	21 in. (53.34 cm)	38 in. (96.52 cm)
Keyboard	3 in. (7.62 cm)	16.625 in. (42.227 cm)	6.625 in. (16.827 cm)

2.1 GT40/GT42 START-UP PROCEDURES

The procedure used to start the GT40/GT42 Graphic Display Terminal is determined by the system configuration. A GT40/GT42 that operates as a terminal in a larger system is started differently than a GT40/GT42 that functions as a stand-alone device. Four procedures are presented in the following paragraphs: GT40/GT42 Terminal Systems, GT42 Paper Tape Systems, GT40 Paper Tape Systems, and GT42 Bootstraps for Other Devices.

2.1.1 GT40/GT42 Terminal Systems

The following procedure is used to initiate the ROM Bootstrap from the PDP-11/10 console on the GT40/42.

1. Determine that the GT40/42 power cord is connected to an appropriate electrical outlet.
2. Turn the console key switch (Figure 1) to the POWER position.
3. Turn the front panel ON-OFF/BRIGHTNESS switch fully counterclockwise and then 3/4 of the way in the clockwise direction. The red power indicator light just below the switch should be on at this time.
4. Press the console ENABLE/HALT switch down to halt the computer.
5. Press the spring-loaded START switch twice; this resets the computer.
6. Place 166000_8 in the Switch register (SR). This is the starting address for the Bootstrap program in the Read-Only Memory (ROM) (Figure 20).
7. Press LOAD ADDRESS to load the address into the computer.
8. Return the ENABLE/HALT switch to the up-most position.
9. Press the START switch. The RUN indicator light should be on at this time.

10. Ensure that the LK40 keyboard ENABLE/DISABLE (On-Off) switch is in the ON position (Figure 6).

11. The GT40/42 is now ready to transmit data to and receive data from the host computer via the DL11 Asynchronous Interface module.

NOTE

A detailed description of the ROM Bootstrap and the loading procedure from a host computer are contained in Paragraph 5.1.

2.1.2 GT42 Paper Tape Systems

The following procedure is used to initiate the ROM Bootstrap from the PDP-11/10 console on the GT42.

1. Determine that the GT42 power cord is connected to an appropriate electrical outlet.

2. Turn the console key switch (Figure 2) to the POWER position.

3. Turn the front panel ON-OFF/BRIGHTNESS switch fully counterclockwise and then 3/4 of the way in the clockwise direction. The red power indicator light just below the switch should be on at this time.

4. Press the console ENABLE/HALT switch down to halt the computer.

5. Press the spring-loaded START switch twice; this resets the computer.

6. Place 167400_8 in the Switch register (SR). This is the starting address for the paper tape Bootstrap program in the Read Only Memory (ROM).

7. Press LOAD ADDRESS to load the address into the computer.

8. Return the ENABLE/HALT switch to the up-most position.

9. Place the Absolute Loader in the specified reader with the special bootstrap leader code over the reader sensors (under the reader station).

10. Press START. The Absolute Loader tape will pass through the reader as data is being loaded into core.

11. The tape stops after the last frame of data has been read into core. The Absolute Loader is now in core. If the Absolute Loader tape does not read in immediately after depressing the START switch, perform steps 26 and 27 of Paragraph 2.1.3.

2.1.3 GT40 Paper Tape Systems

1. Determine that the GT40 power cord is connected to an appropriate electrical outlet.

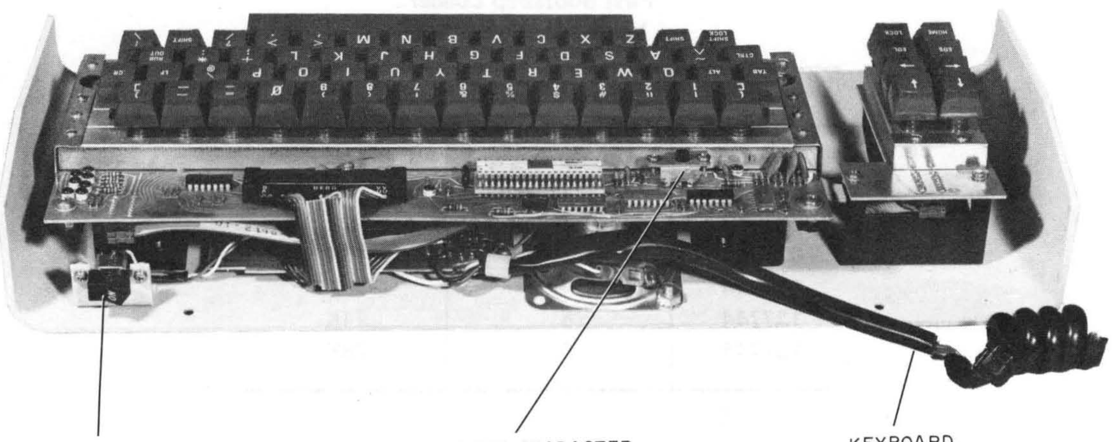
2. Turn the console key switch (Figure 1) to the POWER position.

3. Turn the front panel ON-OFF/BRIGHTNESS switch fully counterclockwise and then 3/4 of the way in the clockwise direction. The red power indicator light just below the switch should be on at this time.

4. Press the console ENABLE/HALT switch down to halt the computer.

5. Press the spring-loaded START switch twice; this resets the computer.

6. The Bootstrap Loader will now be loaded (toggled) into the highest core memory bank. The locations and corresponding instructions of the Bootstrap Loader are listed in Table 1.


ENABLE/DISABLE (ON-OFF) SWITCH
96/128 CHARACTER SET SELECT SWITCH
KEYBOARD CABLE
Figure 6 LK40 Keyboard (cover removed)

The Bootstrap Loader program instructs the computer to accept and store in core memory data that is punched on paper tape in bootstrap format. The Bootstrap Loader is used to load very short paper tape programs of 162_8 16-bit words or less (primarily the Absolute Loader and Memory Dump programs). Programs longer than this must be assembled into absolute binary format using the PAL-11A Assembler and loaded into memory using the Absolute Loader (step 19).

Table 1
Bootstrap Loader Instructions

Location	Instruction
xx7744	016701
xx7746	000026
xx7750	012702
xx7752	000352
xx7754	005211
xx7756	105711
xx7760	100376
xx7762	116162
xx7764	000002
xx7766	xx7400
xx7770	005267
xx7772	177756
xx7774	000765
xx7776	YYYYYY

In Table 1, xx represents the highest available memory bank. For example, the first location of the loader would be 037744_8 if the system contained an 8K memory. Table 2 lists the locations for the first Bootstrap Loader instruction as determined by the memory size. All other locations, for a given memory, are prefixed with the same two digits.

Table 2
First Bootstrap Loader
Instruction Locations

Location	Memory Bank	Memory Size
017744	0	4K
037744	1	8K
057744	2	12K
077744	3	16K
117744	4	20K
137744	5	24K
157744	6	28K

The contents of location xx7776 (YYYYYY in the Instruction column of Table 1) should contain the device status register address of the paper tape reader to be used when loading the bootstrap formatted tapes. Either paper tape reader may be used; their respective addresses are:

Teletype Paper Tape Reader – 177560
High Speed Paper Tape Reader – 177550

7. Set xx7744 in the Switch register (SR) and press the LOAD ADDRess switch (xx7744 will be displayed in the address register).

8. Set the first instruction, 016701, in the SR and lift the DEPosit switch (016701 will be displayed in the data register).

NOTE
When DEPositing data into consecutive words, the DEPosit automatically increments the address register to the next word.

9. Set the next instruction, 000026, in the SR and lift DEPosit (000026 will be displayed in the data register).

10. Set the next instruction in the SR and press the DEPosit switch. Continue depositing subsequent instructions until 000765 is stored in location xx7774.

11. Deposit the desired device status register address in location xx7776, the last location of the Bootstrap Loader.

12. Good programming procedure requires the verification of data that has been stored.

13. Set xx7744 in the SR and press the LOAD ADDRess switch.

14. Press the EXAMine switch. The octal instruction in location xx7744 will be displayed so that it can be compared with the correct instruction: 016701. If the instruction is correct, proceed to step 15, otherwise go to step 17.

15. Press the EXAMine switch. When the switch is held depressed, the ADDRESS/DATA indicators display the memory address. On releasing the switch, the instruction at that address is displayed. Compare the indicator display with the required instruction (Table 1). (The EXAMine switch automatically increments the address register.)

16. Repeat step 15 until all instructions have been verified or go to step 17 whenever the correct instruction is not displayed.

NOTE

Whenever an incorrect instruction is displayed, it can be corrected by performing steps 17 and 18.

17. When an incorrect instruction is displayed in the ADDRESS/DATA indicators, set the correct instruction in the SR and lift the DEPosit switch.

18. Press and release the EXAMine switch to verify that the correct instruction has been deposited. Continue the checking (step 15) until all the instructions have been verified.

19. The Absolute Loader program will be loaded into core memory at this time. The Absolute Loader is a system program which, after being loaded into memory, allows the operator to load, into any core memory bank, data punched on paper tape in absolute binary format. It is used primarily to load the paper tape system software (excluding certain subprograms) and the user's object programs assembled with PAL-11A. The major features of the Absolute Loader include:

- Testing of the checksum on the input tape to ensure complete, accurate loads.
- Starting the loaded program upon completion of loading without additional user action, as specified by the .END in the program just loaded.
- Specifying the load address of position independent programs at load time rather than at assembly time, by using the desired loader switch register option.

With the Bootstrap Loader in core memory, the Absolute Loader is loaded into memory starting anywhere between locations xx7500 and xx7742, i.e., 162_{10} words. The paper tape input device used is specified in location xx7776 (step 11). The Absolute Loader tape begins with about two feet of special bootstrap leader code (ASCII code 351), not blank leader tape.

20. Set the ENABLE/HALT switch to HALT.

21. Place the Absolute Loader in the specified reader with the special bootstrap leader code over the reader sensors (under the reader station).

22. Set the SR to xx7744 (the starting address of the Bootstrap Loader) and press LOAD ADDRess.

23. Set the ENABLE/HALT switch to ENABLE.

24. Press START. The Absolute Loader tape will pass through the reader as data is being loaded into core.

25. The tape stops after the last frame of data has been read into core. The Absolute Loader is now in core.

26. If the Absolute Loader tape does not read in immediately after depressing the START switch (step 24), it is due to one of the following causes:

- Bootstrap Loader not correctly loaded.
- The wrong input device was used.
- Code 351_8 was not directly over the reader sensors.
- The Absolute Loader tape was not properly positioned in the reader.

27. Any paper tape punched in absolute binary format is referred to as an absolute tape, and is loaded into memory using the Absolute Loader. When using the Absolute Loader, there are two methods of loading available: normal and relocated.

    A normal load occurs when the data is loaded and placed in core according to the load addresses on the object tape. It is specified by setting bit 0 of the Switch register to zero immediately before starting the load.

There are two types of relocated loads.

a. Loading to continue from where the loader left off after the previous load. This is used, for example, when the object program being loaded is contained on more than one tape. It is specified by setting the Switch register to 000001 immediately before starting the load.

b. Loading into a specific area of core. This is normally used when loading position independent programs. A position independent program is one that can be loaded and run anywhere in available core. The program is written using the position independent instruction format. This type of load is specified by setting the Switch register to the load address and adding 1 to it, i.e., setting bit 0 to 1.

Optional Switch register settings for the three types of loads are listed in Table 3.

Table 3
Switch Register Configuration for Loading

	Switch Register	
Type of Load	**Bits 1–14**	**Bit 0**
Normal	(ignored)	0
Relocated – continue loading where left off	0	1
Relocated – load in specified area of core	nnnnn (specified address)	1

The absolute tape is now loaded using either of the paper tape readers. The desired reader is specified in the last word of available core memory (xx7776), the input device status word, as explained in step 6. The input device status word can be changed at any time prior to loading the absolute tape.

28. Set the ENABLE/HALT switch to HALT.

    To use an input device different from that used when loading the Absolute Loader, change the address of the device status word (in location xx7776) to reflect the desired device, i.e., 177560 for the Teletype® reader or 177550 for the high speed reader.

29. Set the SR to xx7500 and press LOAD ADDR.

30. Set the SR to reflect the desired type of load.

31. Place the absolute tape in the proper reader with blank leader tape directly over the reader sensors.

32. Set ENABLE/HALT to ENABLE.

33. Press START. The absolute tape will begin passing through the reader station as data is being loaded into core.

34. The Absolute Loader was not correctly stored in memory if the absolute tape does not begin passing through the reader station. If this occurs, reload the loader (steps 20–25) and then the absolute tape (starting at step 28).

    If the absolute tape halts in the middle of the tape, a checksum error occurred in the last block of data read. Normally, the absolute tape will stop passing through the reader station when it encounters the transfer address as generated by the .END statement, denoting the end of a program. If the system halts after loading, check that the low byte of the data register is zero. If so, the tape is correctly loaded. If not zero, a checksum error has occurred in the block of data just loaded, indicating that some data was not correctly loaded. Thus, the tape should be reloaded starting at step 1.

    When loading a continuous relocated load, subsequent blocks of data are loaded by placing the next tape in the appropriate reader and pressing the CONTinue switch.

35. The Absolute Loader may be restarted at any time by starting at step 1.

2.1.4 GT42 Bootstraps For Other Devices

The GT42 contains bootstrap programs for the following devices:

Device	Starting Address (Octal)
TA11 Cassette	167500
RF11 Fixed Head Disk	167600
RC11 Fixed Head Disk	167720
RK11 Disk Cartridge	167610
RP11 Disk Pack	167654
TC11 DECtape	167620
TM11 Magnetic Tape	167636

The following procedure is used to initiate one of the above devices from the PDP-11/10 console of the GT42.

1. Determine that the GT42 power cord is connected to an appropriate electrical outlet.

2. Turn the console key switch to the POWER position.

3. Press the console ENABLE/HALT switch down to halt the computer.

®Teletype is a registered trademark of Teletype Corporation.

4. Press the spring-loaded START switch twice; this resets the computer.

5. Place the address of the device to be started into the Switch register. The device starting addresses are listed above.

6. Press the LOAD ADDRESS switch to load the address into the computer.

7. Return the ENABLE/HALT switch to the up-most position.

8. Press the START switch.

2.1.5 GT42 Graphics Test

The GT42 contains a short program which tests the fundamental graphic capabilities of the display processor. The program, which starts at octal address 167204, displays several lines and points on the CRT.

2.2 GT40/42 FAILURE PROCEDURES

The following procedures should be followed in the event the GT40/42 fails to operate properly. If, after performing these checks, equipment operation is still unsatisfactory, the user should notify the DEC Field Service Office of the problem.

If the GT40/42 is completely inoperative:

1. Check the circuit breaker on the rear panel of the GT40 (Figure 4) or in the cabinet of the GT42. Press the button to reset the circuit breaker.

2. Check the power cord to the wall receptacle. It should be properly seated.

3. Determine that the required power (115 or 230 Vac) is present at the wall receptacle.

If the display scope fails to turn on:

1. Check the keyboard cable connector on the GT40/42 rear panel for proper seating.

2. Check the power plugs on the rear panel and the power control box for proper seating.

3. Determine that the front panel ON-OFF/BRIGHTNESS switch is in the ON position (clockwise).

4. Check the following fuses on the rear panel and the power control box:

   - 5A SB (115 V system)
     (or 3A SB for 230 V systems)

   - 10A (115 V systems)
     (or 5A for 230 V systems)

If the keyboard is incapable of transmitting data:

1. Check the ON/OFF switch on the rear of the keyboard (Figure 6). Place it in the ON position.

2. Check the cable connectors on the GT40/42 rear panel_(particularly the keyboard cable) for proper seating.

3.1 GT40/42 INTERFACES

Transferral of information between GT40/42 components and devices external to the basic system requires a means for connecting or interfacing an extended system. The interface can be considered to be the physical boundary between the GT40/42 and attached units; it provides the communication link between the display terminal and associated devices such as a host computer or additional memory units.

3.2 PARALLEL PORT

The GT40/42 possesses two interfaces. One, called the *parallel port,* uses conventional Unibus signals and connections to transfer data in parallel format. The other interface is employed in the transfer of asynchronous data, in a serial format, over a longer communications line. The two interfaces and their relation to the GT40/42 are shown in Figure 7.

The parallel port is used typically to interface local high speed peripheral devices such as additional core memory, disk storage units, etc. The parallel port is basically an extension of the PDP-11 family Unibus.

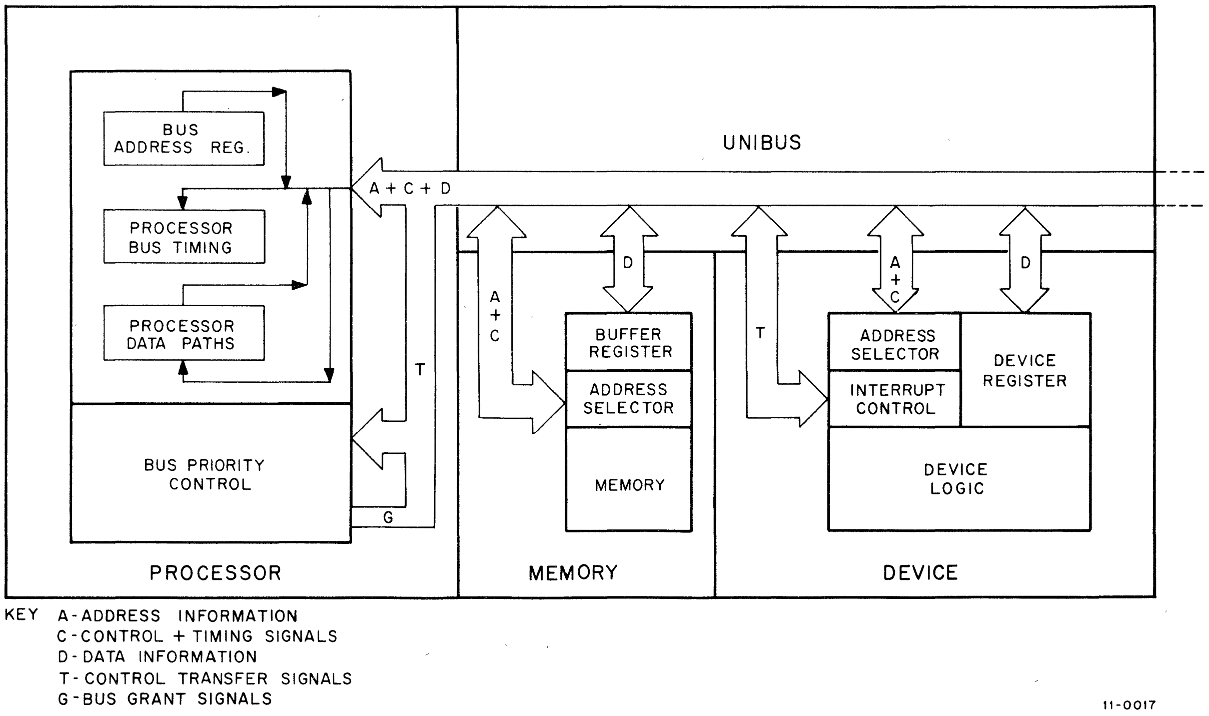


Figure 7 Unibus Interface Block Diagram

3.2.1 Unibus Structure

The Unibus is a single common path that connects the processor, memory, and all peripherals. Addresses, data, and control information are transmitted along the 56 lines of the bus. All 56 signals and their functions are listed in Table 4.

Every device on the Unibus employs the same form of communication; thus, the processor uses the same set of signals to communicate with memory and with peripheral devices. Peripheral devices also communicate with the processor, memory, or other peripheral devices via the same set of signals.

All instructions applied to data in memory can be applied equally well to data in peripheral device registers, enabling peripheral device registers to be manipulated by the processor with the same flexibility as memory. This feature is especially powerful, considering the capability of PDP-11 instructions to process data in any memory location as though it were an accumulator.

Table 4
Unibus Signals

Name	Mnemonic	Source	Destination	Timing	Function
Data Transfer Signals (For transfer of data to or from master)					
Address	A(17:00)	Master	All	MSYN	Selects slave device
Data	D(15:00)	Master	Slave	MSYN (DATO, DATOB)	
		Slave	Master	SSYN (DATI, DATIP)	
Control	C(1:0)	Master	Slave	MSYN	Selects transfer operation
Master Sync	MSYN	Master	Slave	Beginning of transfer	Initiates operation and gates A, C, and D signals
Slave Sync	SSYN	Slave	Master	Data accepted (DATO, DATOB) Data Available (DATI, DATIP)	Response to MSYN
Parity Bit Low	PA	Master	Slave	Same as Data	Transmits parity bit, low byte
Parity Bit High	PB	Master	Slave	Same as Data	Transmits parity bit, high byte
Priority Transfer Signals (For transfer of bus control to a priority-selected master)					
Non-Processor Request	NPR	Any	Processor	Asynchronous	Highest priority bus request
Bus Request	BR(7:4)	Any	Processor	Asynchronous	Requests bus mastership
Non-Processor Grant	NPG	Processor	Next master	In parallel with data transfer	Transfers bus control
Bus Grant	BG(7:4)	Processor	Next master	After instruction	Transfers bus control
Selection Acknowledge	SACK	Next Master	Processor	Response to NPG or BG	Acknowledges grant & inhibits further grants
Bus Busy	BBSY	Master	All	except during transfer of control	Asserts bus mastership

Table 4 (Cont)
Unibus Signals

Name	Mnemonic	Source	Destination	Timing	Function
Interrupt	INTR	Master	Processor	After asserting BBSY (not after NPR), device may perform several transfers before asserting INTR.	Transfers bus control to handling routine in processor
Miscellaneous Signals					
Initialize	INIT	Processor	All	Asynchronous	Clear and reset signal
AC Low	AC LO	Power	All	Asynchronous	Indicates impending power failure
DC Low	DC LO				Indicates dc voltages out of tolerance, and system operation must be suspended.

NOTE
Signals on the Unibus are asserted when low (except for the unidirectional bus grant lines).

3.2.1.1 Bidirectional Lines – Most Unibus lines are bidirectional, allowing input lines to also be driven as output lines. This is significant in that a peripheral device register can be either read or used for transfer operations. Thus, the same register can be used for both input and output functions.

3.2.1.2 Master/Slave Relationship – Communication between two devices on the bus is based on a master/slave relationship. During any bus operation, one device, referred to as the bus master, has control of the bus when communicating with another device, the slave. A typical example of this relationship is the processor (master) transferring data to memory (slave). Master/slave relationships are dynamic. The processor, for example, passes bus control to a disk; the disk, as master, then communicates with a slave memory.

The Unibus is used by the processor and all I/O devices; thus, a priority structure determines which device gains control of the bus. Consequently, every device on the Unibus capable of becoming bus master has an assigned priority. When two devices capable of becoming bus master have identical priority values and simultaneously request use of the bus, the device that is electrically closest to the bus receives control.

3.2.1.3 Interlocked Communication – Communication on the Unibus is interlocked between devices. Each control signal issued by the master device must be acknowledged by a response from the slave to complete the transfer. Consequently, communication is independent of the physical bus length and the response time of the master and slave devices. The maximum transfer rate on the Unibus, with optimum device design, is one 16-bit word every 400 ns or 2.5 million 16-bit words per second.

3.2.2 Peripheral Device Organization and Control

Peripheral device registers are assigned addresses similar to memory; thus, all PDP-11 instructions that address memory locations can become I/O instructions, enabling data registers in peripheral devices to take advantage of all the arithmetic power of the processor.

The PDP-11 controls devices differently than most computer systems. Control functions are assigned to a register address, and then the individual bits within that register can cause control operations to occur. For example, the command to make the paper tape reader read a frame of tape is provided by setting a bit (the reader enable bit) in the control register of the device. Instructions such as MOV and BIS may be used for this purpose. Status conditions are also handled by the assignment of bits within this register, and the status is checked with TST, BIT, and CMP instructions.

3.2.3 Unibus Control Arbitration

The Unibus is capable of performing two basic and parallel tasks in order to allow transfers by multiple peripherals at maximum speed. The first is the actual transfer of data between the current bus master and its addressed slave. The second is the selection of the next bus master, the peripheral which will be allowed to assert control as soon as the bus becomes free. It is important to note that the granting of future mastership is in no way influenced by either the current master or its method of obtaining the bus. It is this fact which allows these functions to be performed in parallel and allows transfers on the bus at a maximum rate.

3.2.3.1 Priority Transfer Requests – To gain mastership of the Unibus, a peripheral must first make a request to the processor for the bus and then wait for its selection. The processor contains the logic necessary to arbitrate these requests because normally there are several requests pending at any given time.

There are two classes of requests: bus requests and non-processor requests. A bus request (BR) is simply a request by a peripheral to obtain control of the Unibus with the understanding by the processor that the peripheral may end its use of the bus with a processor interrupt. An interrupt is a command to the processor to begin executing a new routine pointed to by a location selected by a device. A non-processor request (NPR) is similarly a request for the bus, but with the exception that it may not interrupt the processor. Since the granting of an NPR cannot affect the execution of the processor, it can occur during or between instructions. BRs, however, by possibly causing execution to be diverted to a totally new routine, can only be granted between instructions. In this way, NPRs are assigned priority over any BR.

Between bus requests, there are four levels of priority created by four separate request lines. They are assigned priority levels 4 through 7; BR4 is the lowest and BR7 is the highest. These levels are associated with the program controlled priority level of the processor, controlled by bits 7, 6, and 5 of the processor status register. Only BRs on a priority level higher than the level of the processor are eligible for receiving a bus grant. Thus, during high priority program tasks, all or selected Unibus requests (hence interrupts) can be inhibited by raising the level of the processor priority.

Another form of priority arbitration occurs through the system configuration. When the processor grants a request, the grant travels along the bus until it reaches the first requesting device which terminates the grant. Therefore, along the same grant line, the device electrically nearest the processor has the highest priority. Also note that in the KD11-B, the internal line clock is logically the last device on BR6, and the keyboard or Teletype interface is logically the last device on BR4.

The GT40/42 relationship to this priority scheme is indicated in Table 5.

After a requesting device receives a bus grant it asserts its selection as next bus master until the bus is free, thus inhibiting other requests from being granted. When the bus becomes free, the selected device asserts control of the bus and relinquishes its selection as next bus master so that the priority arbitration among pending requests may continue.

Table 5
GT40/42 Priority

GT40/42 Component	Priority Level	Relative Physical Position from the CPU
DL11 Asynchronous Interface	BR5	2
Display Processor	BR4	1
Unibus Output Slot (Parallel Port)	—	3

NOTE: The MM11 memory is not shown as an active device because it always functions as a slave, never asserting a bus request itself.

3.2.3.2 Processor Interrupts — After gaining control of the bus through a BR, a device can perform one or more transfers on the bus and/or request a processor interrupt. This is typically requested after a device has completed a given task, e.g., typing a character or completing a block data transfer through NPRs. If a peripheral wishes to interrupt the processor, it must assert the interrupt after gaining control of the bus but before relinquishing its selection as next bus master. Thus the processor knows that it may not fetch the next instruction, but must wait for the interrupt to be completed. Along with asserting the interrupt, the device asserts the unique memory address, known as the interrupt vector address, containing the starting address of the device service routine. Address vector +2 contains the new processor status word (PSW) to be used by the processor when beginning the service routine. After recognizing the interrupt, the processor reads the vector address and saves it in an internal register. It then pushes the current PSW and program counter onto the stack and loads the new program counter (PC) and PSW from the vector address specified. The service routine is then executed.

NOTE
These operations are performed automatically and no device polling is required to determine which routine to execute.

The device service routine can cause the processor to resume the interrupted process by executing the return from interrupt (RTI) instruction which pops the top two words from the processor stack and transfers them back to the PC and PS registers.

3.2.3.3 Data Transfers — After asserting control of the Unibus, the device does not release control until it has completed either one or more data transfers or an interrupt. Typically, only one transfer is completed each time the device gains control of the bus because few single devices can give or receive information at the maximum Unibus rate. Holding the bus for multiple transfers inhibits other devices from using the bus.

A transfer is initiated by the master device asserting a slave address and control signals on the bus and a master or address validity signal. The appropriate slave recognizes the valid address, reads or writes the data, and responds with a transfer complete signal. The master recognizes the transfer complete, sends or accepts data, and drops the address validating signal. It can then assert a new address and repeat the process or release control of the bus completely.

The importance of this type of structure is that it enables direct device-to-device transfers without any interaction from the central processor. An NPR device, such as the high speed CRT display, can gain fast access to the bus and transfer data at high rates while refreshing itself from memory without slowing down the processor.

For a more detailed description of the Unibus and its function, refer to the *GT40 Graphic Display Terminal Maintenance Manual,* Volume 2 or to the *PDP-11 Peripherals Handbook.*

3.3 SERIAL PORT

The serial port is the primary means of interfacing the GT40/42 with a host or remote computer. Access to this port is through the DL11 Asynchronous Interface module and the 25-ft BC05-C-25 cable which terminates in a 25-pin, RS232-defined connector at a data set modem (Figure 3 and Table 6).

Table 6
BC05-C-25 Cable Output Connections

CINCH Connector Pin No. (to modem)	Signal
1	Ground
2	Transmitted Data
3	Received Data
4	Request to Send
5	Clear to Send
6	Data Set Ready
7	Ground
8	Carrier
9	+ Power
10	– Power
11	202 Secondary Transmit
12	202 Secondary Receive
13	Secondary Clear to Send
14	EIA Secondary Transmit
15	Serial Clock Transmit
16	EIA Secondary Receive
17	Serial Clock Receive
18	Unassigned
19	Secondary Request to Send
20	Data Terminal Ready
21	Signal Quality
22	Ring
23	Signal Rate
24	External Clock
25	Force Busy

3.4 DL11 PROGRAMMING

All software control of the DL11 Asynchronous Line Interface is performed by means of four device registers. These registers have been assigned bus addresses and can be read or loaded (with the exceptions noted) using any PDP-11 instruction referring to their addresses. Address assignments can be changed by altering jumpers on the address selection logic to correspond to any address within the range of 174000 to 177777. However, register addresses for the DL11 in the GT40/42 fall within the range of 175610 to 175616.

The four device registers and associated DL11 addresses are listed in Table 7.

Table 7
Standard DL11 Register Assignments for the GT40/42

Register	Mnemonic	Address
Receiver Status Register	RCSR	175610
Receiver Buffer Register	RBUF	175612
Transmitter Status Register	XCSR	175614
Transmitter Buffer Register	XBUF	175616

Figures 8 through 11 show the bit assignments for the four device registers. The unused and load-only bits are always read as 0s. Loading unused or read-only bits has no effect on the bit position. The mnemonic INIT refers to the initialization signal issued by the processor. Initialization is caused by one of the following: issuing a programmed RESET instruction; depressing the START switch on the processor console; or the occurrence of a power-up or power-down condition of the processor power supply.

In the following descriptions, *transmitter* refers to those registers and bits involved in accepting a parallel character from the Unibus for serial transmission to the external device; *receiver* refers to those registers and bits involved with receiving serial information from the external device for parallel transfer to the Unibus.

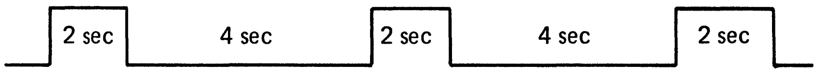
15	14	13	12	11	10		7	6	5		3	2	1	0
DSET INT	RING	CLR TO SEND	CARR DET	RCVR ACT	SEC REC	▨	RCVR DONE	RCVR INT ENB	DSET INT ENB	▨	SEC XMIT	REQ TO SEND	DATA TERM RDY	RDR ENB *

RCSR = 175610 * Not used for data operations.

Figure 8 Receiver Status Register (RCSR) – Bit Assignments

3.4.1 Receiver Status Register

Bit	Name	Meaning and Operation
15	DATASET INT (Dataset Interrupt)	This bit initiates an interrupt sequence provided the DATASET INT ENB bit (05) is also set. This bit is set whenever CAR DET, RCVR ACT, or SEC REC changes state, i.e., on a 0 to 1 or 1 to 0 transition of any one of these bits. It is also set when RING changes from 0 to 1.

		Cleared by INIT or by reading the RCSR. Because reading the register clears the bit, it is, in effect, a "read-once" bit.
14	RING	When set, indicates that a RINGING signal is being received from the dataset. Note that the RINGING signal is not a level but an EIA control signal with the cycle time as shown below:  Read-only bit
13	CLR TO SEND (Clear to Send)	The state of this bit is dependent on the state of the CLEAR TO SEND signal from the dataset. When set, this bit indicates an ON condition; when clear, it indicates an OFF condition. Read-only bit.
12	CAR DET (Carrier Detect)	This bit is set when the data carrier is received. When clear, it indicates either the end of the current transmission activity or an error condition. Read-only bit.
11	RCVR ACT (Receiver Active)	When set, this bit indicates that the DL11 interface receiver is active. The bit is set at the center of the START bit which is the beginning of the input serial data from the device and is cleared by the leading edge of RCVR DONE. Read-only bit; cleared by INIT or by RCVR DONE (bit 07).
10	SEC REC (Secondary Receive or Supervisory Received Data)	This bit provides a receive capability for the reverse channel of a remote station. A space (+6V) is read as a 1. (A transmit capability is provided by bit 03.) Read-only bit; cleared by INIT.
9–8	Unused	Not applicable.
07	RCVR DONE (Receiver Done)	This bit is set when an entire character has been received and is ready for transfer to the Unibus. When set, initiates an interrupt sequence provided RCVR INT ENB (bit 06) is also set. Cleared whenever the receiver buffer (RBUF) is addressed or whenever RDR ENB (bit 00) is set. Also cleared by INIT. Read-only bit.

Bit	Name	Meaning and Operation
06	RCVR INT ENB (Receiver Interrupt Enable)	When set, allows an interrupt sequence to start when RCVR DONE (bit 07) sets. Read/write bit; cleared by INIT.
05	DATASET INT ENB (Dataset Interrupt Enable)	When set, allows an interrupt sequence to start when DATASET INT (bit 15) sets. Read/write bit; cleared by INIT.
04	Unused	Not applicable.
03	SEC XMIT (Secondary Transmit or Supervisory Transmitted Data)	This bit provides a transmit capability for a reverse channel of a remote station. When set, transmits a space (+6V). (A receive capability is provided by bit 10.) Read/write bit; cleared by INIT.
02	REQ TO SEND (Request to Send)	A control lead to the dataset which is required to transmission. A jumper ties this bit to REQ TO SEND or FORCE BUSY in the dataset. Read/write bit; cleared by INIT.
01	DTR (Data Terminal Ready)	A control lead for the dataset communication channel. When set, permits connection to the channel. When clear, disconnects the interface from the channel. Read/write bit; must be cleared by the program, is not cleared by INIT.

NOTE
The state of this bit is not defined after power-up.

Bit	Name	Meaning and Operation
00	RDR ENB (Reader Enable)	When set, this bit advances the paper-tape reader in ASR Teletype units and clears the RCVR DONE bit (bit 07). This bit is cleared at the middle of a START bit which is the beginning of the serial input from an external device. Also cleared by INIT. Write-only bit. Not used in dataset configurations.

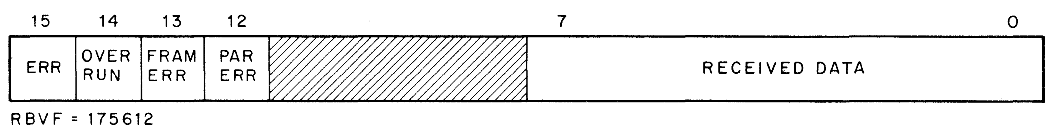


Figure 9 Receiver Buffer Register (RBUF) – Bit Assignments

3.4.2 Receiver Buffer Register

Bit	Name	Meaning and Operation
15	ERROR (Error)	Used to indicate that an error condition is present. This bit is the logical OR of OR ERR, FR ERR, and P ERR (bits 14, 13, and 12, respectively). Whenever one of these bits is set, it causes ERROR to set. This bit is not connected to the interrupt logic. Read-only bit; cleared by removing the error-producing condition.

NOTE
Error indications remain present until the next character is received, at which time the error bits are updated. INIT does not necessarily clear the error bits.

Bit	Name	Meaning and Operation
14	OR ERR (Overrun Error)	When set, indicates that reading of the previously received character was not completed (RCVR DONE not cleared) prior to receiving a new character. Read-only bit; cleared in the same manner as ERROR (bit 15).
13	FR ERR (Framing Error)	When set, indicates that the character that was read had no valid STOP bit. Read-only bit; cleared in the same manner as ERROR (bit 15).
12	P ERR (Parity Error)	When set, indicates that the parity received does not agree with the expected parity. This bit is always 0 if no parity is selected. Read-only bit; cleared in the same manner as ERROR (bit 15).
11—08	Unused	Not applicable.
07—00	RECEIVED DATA BITS	Holds the character just read. If less than eight bits are selected, then the buffer is right-justified into the least significant bit positions. In this case, the higher unused bit or bits read as 0s. Read-only bits; not cleared by INIT.

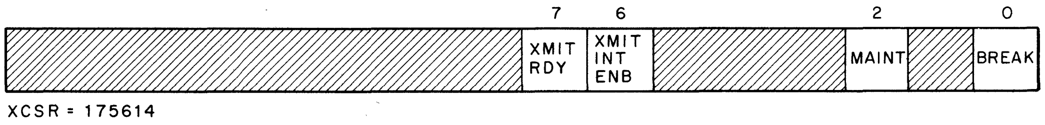


Figure 10 Transmitter Status Register (XCSR) — Bit Assignments

3.4.3 Transmitter Status Register

Bit	Name	Meaning and Operation
15–08	Unused	Not applicable.
07	XMIT RDY (Transmitter Ready)	This bit is set when the transmitter buffer (XBUF) can accept another character. When set, it initiates an interrupt sequence provided XMIT INT ENB (bit 06) is also set. Read-only bit. Set by INIT. Cleared by loading the transmitter buffer.
06	XMIT INT ENB (Transmitter Interrupt Enable)	When set, allows an interrupt sequence to start when XMIT RDY (bit 07) sets.
05–03	Unused	Not applicable.
02	MAINT (Maintenance)	Used for maintenance function. When set, disables the serial line input to the receiver and connects the transmitter output to the receiver input which disconnects the external device input. It also forces the receiver to run at transmitter speed. Read/write bit; cleared by INIT.
01	Unused	Not applicable.
00	BREAK	When set, transmits a continuous space to the external device. Read/write bit; cleared by INIT.

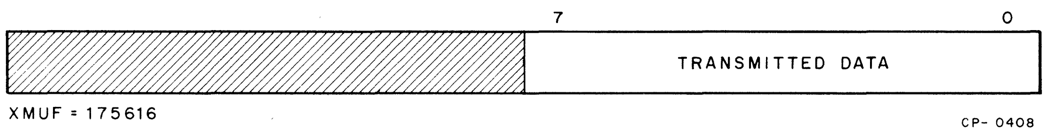


Figure 11 Transmitter Buffer Register (XBUF) – Bit Assignments

3.4.4 Transmitter Buffer Register

Bit	Name	Meaning and Operation
15–08	Unused	Not applicable.
07–00	TRANSMITTER DATA BUFFER	Holds the character to be transferred to the external device. If less than eight bits are used, the character must be loaded so that it is right-justified into the least significant bits. Write-only bits.

3.4.5 Interrupts

The DL11 interface uses BR interrupts to gain control of the bus to perform a vectored interrupt, thereby causing a branch to a handling routine. The DL11 has two interrupt channels: one for the receiver section and one for the transmitter section. These two channels operate independently; however, if simultaneous interrupt requests occur, the receiver has priority. The receiver section is capable of handling multiple source interrupts.

A transmitter interrupt can occur only if the interrupt enable bit (XMIT INT ENB) in the transmitter status register is set. With XMIT INT ENB set, setting the transmitter ready (XMIT RDY) bit initiates an interrupt request. When XMIT RDY is set, it indicates that the transmitter buffer is empty and ready to accept another character from the bus for transfer to the external device.

A receiver data interrupt can occur only if the interrupt enable (RCVR INT ENB) bit in the receiver status register is set. With RCVR INT ENB set, setting the receiver done (RCVR DONE) bit initiates an interrupt request. When RCVR DONE is set, it indicates that an entire character has been received and is ready for transfer to the bus. The additional interrupt request sources for the DL11 option are discussed in the following paragraphs.

The receiver portion of the DL11 in the GT40/42 dataset configuration can service multiple source interrupts. One of the receiver interrupt circuits is activated by RCVR INT ENB and RCVR DONE. The additional interrupt circuit can cause an interrupt only if the dataset interrupt enable bit (bit 05, DATASET INT ENB) in the receiver status register is set. With DATASET INT ENB set, setting the DATASET INT bit initiates an interrupt request. The DATASET INT bit can be set by one of four other bits: CAR DET, CLR TO SEND, SEC REC, or RING.

When servicing an interrupt for one condition, if a second interrupt condition develops, a unique second interrupt, as well as all subsequent interrupts, may not occur. To prevent this, either all possible interrupt conditions should be checked after servicing one condition or both interrupt enable bits (bits 05 and 06) should be cleared upon entry to the service routine for vector XX0 and then set again at the end of service.

The interrupt priority level is 5 with the receiver having a slightly higher priority than the transmitter in all cases. Note that the priority level can be changed with a priority plug.

Any DEC programs or other software referring to the standard BR level or vector addresses must also be changed if the priority plug or vector address is changed.

3.4.6 Timing Considerations

When programming the DL11 Asynchronous Line Interface, it is important to consider timing of certain functions in order to use the system in the most efficient manner. Timing considerations for the receiver transmitter, and break generation logic are discussed in the following paragraphs.

3.4.6.1 Receiver – The RCVR DONE flag (bit 07 in the RCSR) sets when the Universal Asynchronous Receiver/Transmitter (UART) has assembled a full character. This occurs at the middle of the first STOP bit. Because the UART is double buffered, data remains valid until the next character is received and assembled. This permits one full character time for servicing the RCVR DONE flag.

3.4.6.2 Transmitter – The transmitter section of the UART is also double buffered. The XMIT RDY flag (bit 07 in the XCSR) is set after initialization. When the buffer (XBUF) is loaded with the first character from the bus, the flag clears but then sets again within a fraction of a bit time. A second character can then be loaded, which clears the flag again. The flag then remains cleared for nearly one full character time.

3.4.6.3 Break Generation Logic – When the BREAK bit (bit 00 in the XCSR) is set, it causes transmission of a continuous space. Because the XMIT RDY flag continues to function normally, the duration of a break can be timed by the pseudo-transmission of a number of characters. However, because the transmitter section of the UART is double buffered, a null character (all 0s) should precede transmission of the break to ensure that the previous character clears the line. In a similar manner, the final pseudo-transmitted character in the break should be null.

3.4.7 Program Notes

The following notes pertain to programming the DL11 interface and contain information that may be useful to the programmer. More detailed programming information is given in the *Paper Tape Software Programming Handbook*, DEC-11-GGPC-D and in the individual program listings.

a. Character Format – The character format for the DL11 consists of a START bit, five to eight DATA bits, 1, 1.5, or 2 STOP bits and the option of PARITY (odd or even) or no parity. This is illustrated in Figure 12. Note that when less than eight DATA bits are used, the character must be right-justified to the least significant bit. The character format pertains to both the receiver and the transmitter.

b. Maintenance Mode – The maintenance mode is selected by setting the MAINT bit (bit 02) in the XCSR. In this mode, the interface disables the normal input to the receiver and replaces it with the output of the transmitter. The programmer can then load various bits into the transmitter and read them back from the receiver to verify proper operation of the DL11 logic circuits.

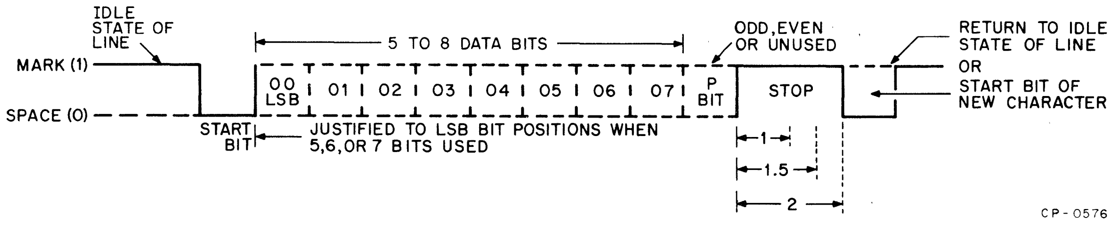


Figure 12 Serial Character Format

3.4.8 Program Example

Figure 13 is an example of a typical program that can be used as an echo program for a Type 103 dataset. When a remote terminal dials in, this program answers the call and provides a character-by-character echo. Characters are also copied onto the console device.

4.1 PROGRAMMING THE GT40/42

4.2 PROGRAMMING CONCEPT

The user should view the GT40/42 Graphic Display Terminal as two separate, programmed processors: a PDP-11/10 computer (CPU) and a special display processor (DPU). The PDP-11/10 is programmed to initiate the display, and is then free to execute its own program. All instructions available on the PDP-11/10 are executable in the GT40/42. Figure 14 shows the relationship of the GT40/42 components to the Unibus (the inset illustrates specific GT40/42 data flow via the Unibus).

The DPU communicates directly with the MM11 memory by way of non-processor requests (NPR), i.e., DMA requests. The PDP-11/10, connected in parallel, also uses the MM11 memory for executing its own PDP-11 code. The DPU executes display instructions stored in semi-contiguous memory locations called display lists. A memory layout example is shown in Figure 15. The Display Program Counter (DPC) in the DPU is addressed by the CPU, via the Unibus, and the data MOVed to the DPC becomes the starting address of the display list. All addresses placed on the Unibus are even numbers, i.e., word addresses.

```
        000200                              .=200
000200  000167  001616             START:   JMP     BEGIN           ;JUMP TO BEGINNING OF PROGRAM

                                   ;SYMBOL DEFINITIONS

        040000                     RING=    040000                  ;BIT 14 OF RCSR, RING
        020000                     CTS=     020000                  ;BIT 13 OF RCSR, CLEAR TO SEND
        000200                     RDONE=   000200                  ;BIT 07 OF RCSR, RECEIVER DONE
        000002                     DTR=     000002                  ;BIT 01 OF RCSR, DATA TERMINAL READY
        000200                     XRDY=    000200                  ;BIT 07 OF XCSR, TRANSMITTER READY

        002000                              .=2000
002000  175610                     RCSR:    175610                  ;CSR OF RECEIVER
002002  175612                     RBUF:    175612                  ;BUF OF RECEIVER
002004  175614                     XCSR:    175614                  ;CSR OF TRANSMITTER
002006  175616                     XBUF:    175616                  ;BUF OF TRANSMITTER
002010  177564                     CXCSR:   177564                  ;CSR OF CONSOLE TRANSMITTER
002012  177566                     CXBUF:   177566                  ;BUF OF CONSOLE TRANSMITTER
002014  000000                     BUFFER:  0                       ;HOLDS CHARACTER RECEIVED
002016  000000                     DELAY:   0                       ;HOLDS DELAY COUNT, HIGH ORDER
002020  000000                              0                       ;HOLDS DELAY COUNT, LOW ORDER


                                   ;BEGINNING OF ECHO PROGRAM

002022  005077  177752             BEGIN:   CLR     @RCSR           ;START BY INITIALIZING ALL BITS TO ZERO

002026  032777  040000  177744     LOOP1:   BIT     #RING,@RCSR     ;CHECK FOR INCOMING CALL
002034  001774                              BEQ     LOOP1           ;BRANCH IF PHONE IS NOT RINGING
002036  052777  000002  177734              BIS     #DTR,@RCSR      ;PHONE IS RINGING, SO ANSWER WITH DTR
002044  012767  000005  177744              MOV     #5,DELAY        ;SET UP COUNT FOR DELAY

002052  032777  020000  177720     LOOP2:   BIT     #CTS,@RCSR      ;CHECK FOR CLEAR TO SEND
002060  001007                              BNE     LOOP3           ;BRANCH IF ON
002062  162767  000001  177730              SUB     #1,DELAY+2      ;CHECK DELAY
002070  005667  177722                      SBC     DELAY           ;DECREMENT A TWO-WORD INTEGER
002074  001752                              BEQ     BEGIN           ;BRANCH IF WE HAVE WAITED TOO LONG
002076  000765                              BR      LOOP2           ;BRANCH AND CONTINUE TO WAIT FOR CTS

002100  032777  020000  177672     LOOP3:   BIT     #CTS,@RCSR      ;IS CHANNEL STILL ESTABLISHED?
002106  001745                              BEQ     BEGIN           ;BRANCH IF CTS NOT PRESENT
002110  032777  000200  177662              BIT     #RDONE,@RCSR    ;CHECK FOR RECEIVED CHARACTER
002116  001770                              BEQ     LOOP3           ;BRANCH IF NO CHARACTER RECEIVED
002120  017767  177656  177666              MOV     @RBUF,BUFFER    ;READ RECEIVED CHARACTER INTO BUFFER

002126  032777  000200  177650     LOOP4:   BIT     #XRDY,@XCSR     ;CHECK FOR TRANSMITTER READY
002134  001774                              BEQ     LOOP4           ;BRANCH IF NOT READY
002136  016777  177652  177642              MOV     BUFFER,@XBUF    ;TRANSMIT CHARACTER TO REMOTE TERMINAL

002144  032777  000200  177636     LOOP5:   BIT     #XRDY,@CXCSR    ;CHECK FOR CONSOLE TRANSMITTER READY
002152  001774                              BEQ     LOOP5           ;BRANCH IF NOT READY
002154  016777  177634  177630              MOV     BUFFER,@CXBUF   ;TRANSMIT CHARACTER TO CONSOLE
002162  000746                              BR      LOOP3           ;BRANCH AND WAIT FOR NEXT CHARACTER
```

Figure 13 Program Example

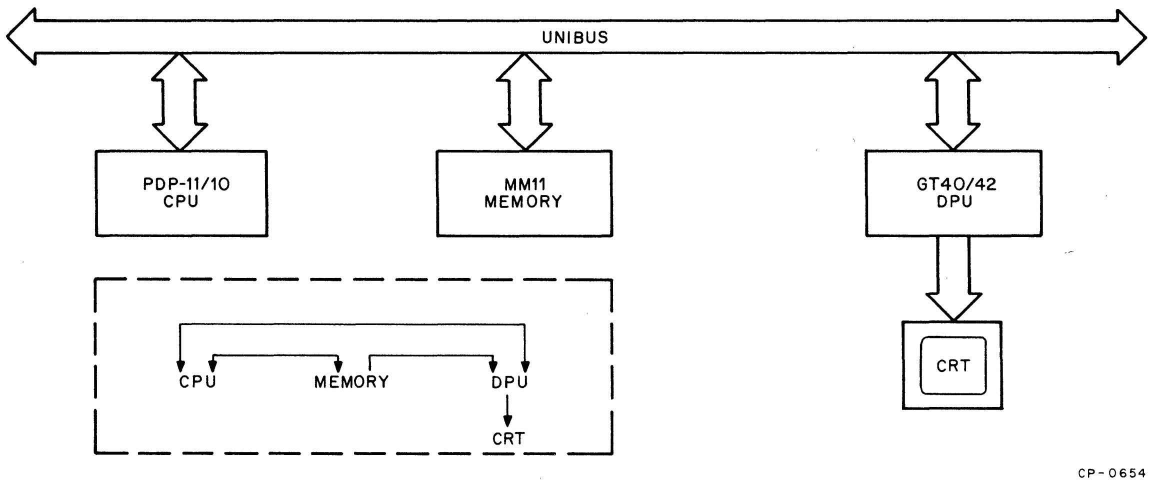


Figure 14 GT40/42 Data Paths

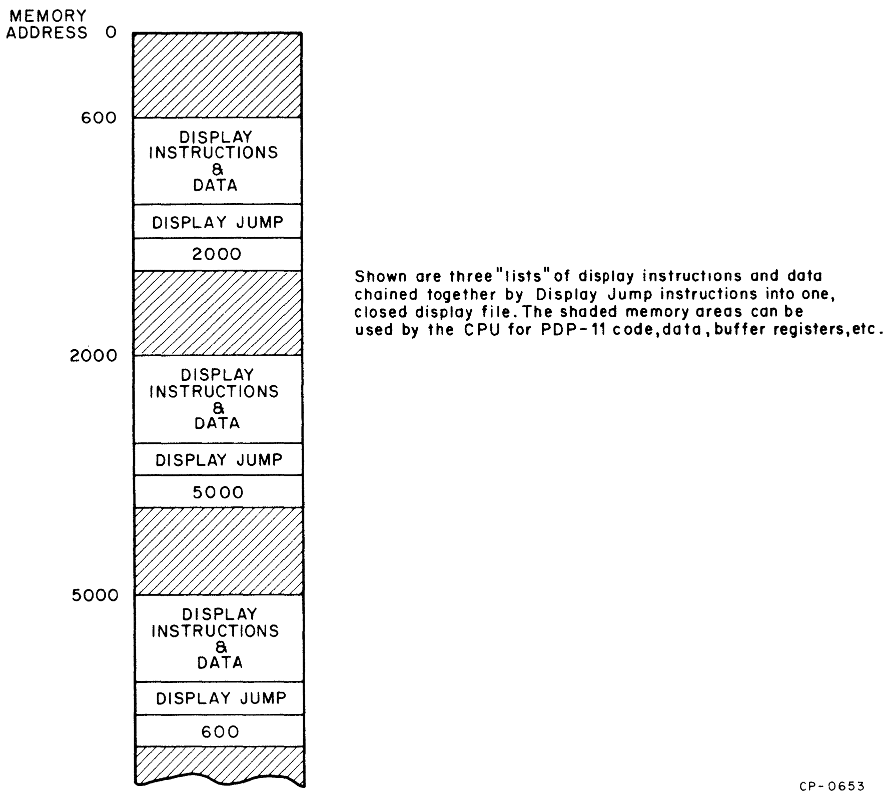

MEMORY ADDRESS 0
600
DISPLAY INSTRUCTIONS & DATA
DISPLAY JUMP
2000
2000
DISPLAY INSTRUCTIONS & DATA
DISPLAY JUMP
5000
5000
DISPLAY INSTRUCTIONS & DATA
DISPLAY JUMP
600
Shown are three "lists" of display instructions and data chained together by Display Jump instructions into one, closed display file. The shaded memory areas can be used by the CPU for PDP-11 code, data, buffer registers, etc.
CP-0653

Figure 15 Memory Layout Example

4.3 IMPORTANT REGISTERS (all addresses are in octal)

Display Addresses:

Display Program Counter (DPC) = 172000 (Read/Write)
Resume Address (RA) = 172000 (Write)

(To resume a display, for example after a light pen hit, bit 0 (LSB) = 1 should be MOVed to the RA, i.e., MOV #1, RA.)

Display Status Register = 172002 (Read/Write)

Contents (Read):		
Stop Flag	Bit	(15) (MSB)
Mode		(14:11)
Intensity		(10:8)
Light Pen Flag		(7)
Shift Out		(6)
Edge Indicator		(5)
Italics		(4)
Blink		(3)
Spare (Not Used)		(2)
Line		(1:0)

(If an attempt is made to write to address 172002, the effect is to ring the BELL in the GT40/42, e.g., MOV #2, 172002.)

X Status Register = 172004 (Read only)

Contents:

X Position	Bits	(9:0)
Graphplot Increment		(15:10)

Y Status Register = 172006 (Read only)

Contents:

Y Position	Bits	(9:0)
Character Register		(15:10)

(Note: When in the SHIFTED OUT character mode, and an illegal code ($040 \rightarrow 137_8$) is fetched, the program is interrupted. The Character Register can then be read to find the dispatch to a user routine that is used to draw some special character.)

Display Interrupt Vector Addresses:

Stop Interrupt = 320/322
Light Pen Interrupt = 324/326
Time Out and Shift Out Interrupt = 330/332
(All display interrupts are requested at level BR4.)

DL11 Communications Interface Addresses:

Receive Status Register (RCSR) = 175610
Receive Buffer (RBUF) = 175612
Transmitter Status Register (XCSR) = 175614
Transmitter Buffer (XBUF) = 175616
(Additional DL11 programming information is included in Paragraph 3.1.)

DL11 Interrupt Vector Addresses:

Receiver Interrupt = 300/302
Transmitter Interrupt = 304/306
(DL11 interrupts are requested on level BR5.)

Miscellaneous Addresses:

CPU General Register (only console addressable) R0 = 177700
⋮
R7 = 177707

CPU Console Switches SWR = 177570
(console and CPU addressable)

CPU Status PS = 177776
(console and CPU addressable)

Keyboard Command and Status (KCSR) = 177560

Keyboard Data Buffer (KDBR) = 177562

Keyboard Interrupt Vector = 60/62

Line Frequency Clock (KW11-L) = 177546

ROM Bootstrap Memory = 166000
(Starting Address)

4.4 PDP-11 INSTRUCTION SET

A detailed description of the PDP-11 instruction set can be found in *GT40 Graphic Display Terminal, Volume 2* (DEC-11-HGTMA-A-D). This manual assumes the reader is familiar with the instruction set and general operation of the PDP-11/10.

4.5 GT40/42 DISPLAY PROCESSOR INSTRUCTION SET

The display processor instruction set consists of five basic instructions: Set Graphic Mode, Jump, No-op, Load Status Register A, and Load Status Register B. Figure 16 shows the breakdown, by bit position, of each instruction. Figure 17 provides similar information for the data words that accompany the instructions.

NOTE
The user should not insert 1-bits into those positions indicated as spare or unused.

4.6 PROGRAMMING EXAMPLES

The following programming examples are meant to provide the user with a basic introduction to GT40/42 programming technique. They have been kept brief in order that the points being illustrated not be lost as would be the case if larger, operational program examples were used.

Table 8 is a list of suggested mnemonics for GT40/42 operation.

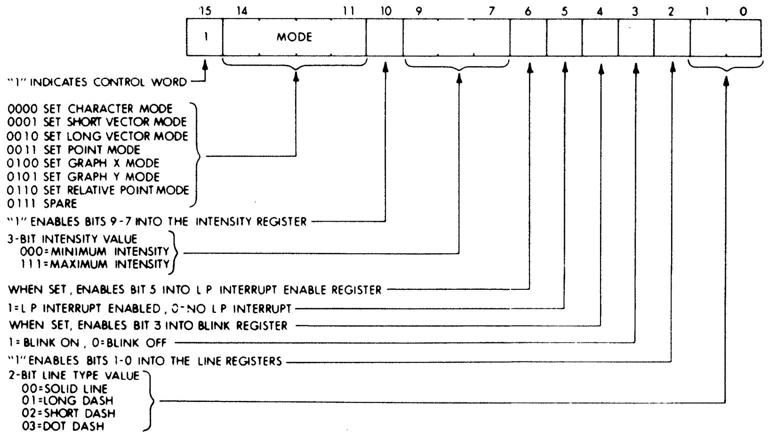
4.6.1 Initializing the Display Processor

To start the DPU, the CPU executes a short program that loads the Display (processor) Program Counter (DPC) with the starting address (SA) of the display file. The Stack Pointer must also be initialized to an address above 400_8 to prevent a stack overflow if an interrupt occurs.

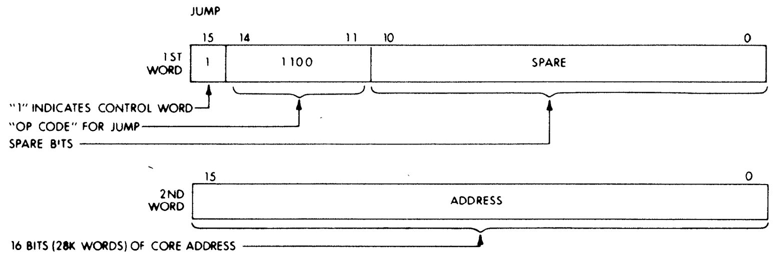
The following program performs these two operations.

Address	Instruction/Data	Mnemonic	Comment
1000	012706	MOV #500, R6	Initialize the
1002	500		stack pointer
1004	012737	MOV #SA, @ #DPC	Load the DPC
1006	2000		with SA = 2000
1010	172000		
1012	00001	WAIT	Wait (or other PDP-11 code)

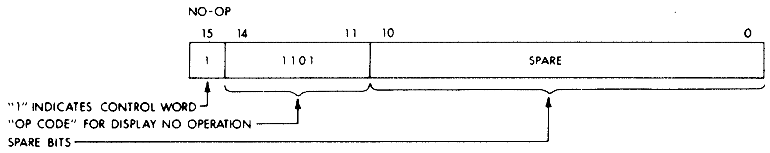
SET GRAPHIC MODE


15 14 11 10 9 7 6 5 4 3 2 1 0
1
MODE
"1" INDICATES CONTROL WORD
0000 SET CHARACTER MODE
0001 SET SHORT VECTOR MODE
0010 SET LONG VECTOR MODE
0011 SET POINT MODE
0100 SET GRAPH X MODE
0101 SET GRAPH Y MODE
0110 SET RELATIVE POINT MODE
0111 SPARE
"1" ENABLES BITS 9-7 INTO THE INTENSITY REGISTER
3-BIT INTENSITY VALUE
000=MINIMUM INTENSITY
111=MAXIMUM INTENSITY
WHEN SET, ENABLES BIT 5 INTO L P INTERRUPT ENABLE REGISTER
1=L P INTERRUPT ENABLED, 0=NO L P INTERRUPT
WHEN SET, ENABLES BIT 3 INTO BLINK REGISTER
1=BLINK ON, 0=BLINK OFF
"1" ENABLES BITS 1-0 INTO THE LINE REGISTERS
2-BIT LINE TYPE VALUE
00=SOLID LINE
01=LONG DASH
02=SHORT DASH
03=DOT DASH

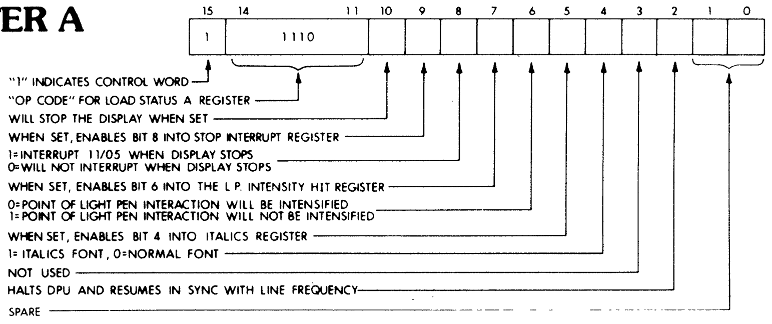
JUMP



NO-OP


NO-OP
15 14 11 10 0
1
1101
SPARE
"1" INDICATES CONTROL WORD
"OP CODE" FOR DISPLAY NO OPERATION
SPARE BITS

LOAD STATUS REGISTER A


15 14 11 10 9 8 7 6 5 4 3 2 1 0
1
1110
"1" INDICATES CONTROL WORD
"OP CODE" FOR LOAD STATUS A REGISTER
WILL STOP THE DISPLAY WHEN SET
WHEN SET, ENABLES BIT 8 INTO STOP INTERRUPT REGISTER
1=INTERRUPT 11/05 WHEN DISPLAY STOPS
0=WILL NOT INTERRUPT WHEN DISPLAY STOPS
WHEN SET, ENABLES BIT 6 INTO THE L.P. INTENSITY HIT REGISTER
0=POINT OF LIGHT PEN INTERACTION WILL BE INTENSIFIED
1=POINT OF LIGHT PEN INTERACTION WILL NOT BE INTENSIFIED
WHEN SET, ENABLES BIT 4 INTO ITALICS REGISTER
1=ITALICS FONT, 0=NORMAL FONT
NOT USED
HALTS DPU AND RESUMES IN SYNC WITH LINE FREQUENCY
SPARE

LOAD STATUS REGISTER B

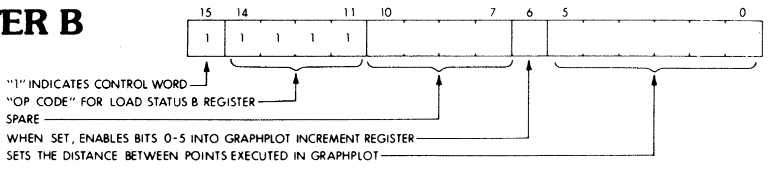
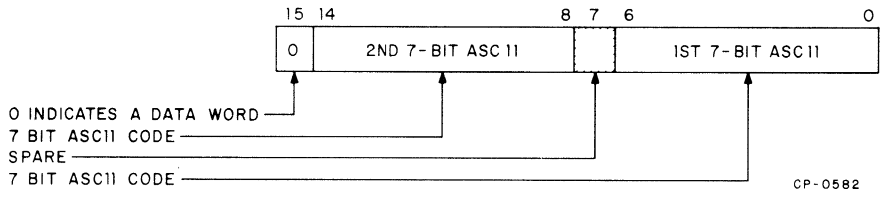
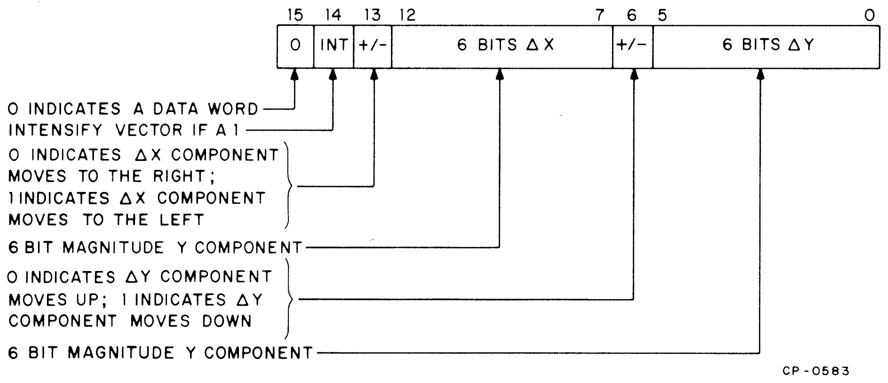

15 14 11 10 7 6 5 0
1 1 1 1 1
"1" INDICATES CONTROL WORD
"OP CODE" FOR LOAD STATUS B REGISTER
SPARE
WHEN SET, ENABLES BITS 0-5 INTO GRAPHPLOT INCREMENT REGISTER
SETS THE DISTANCE BETWEEN POINTS EXECUTED IN GRAPHPLOT

Figure 16 Instruction Word Functions

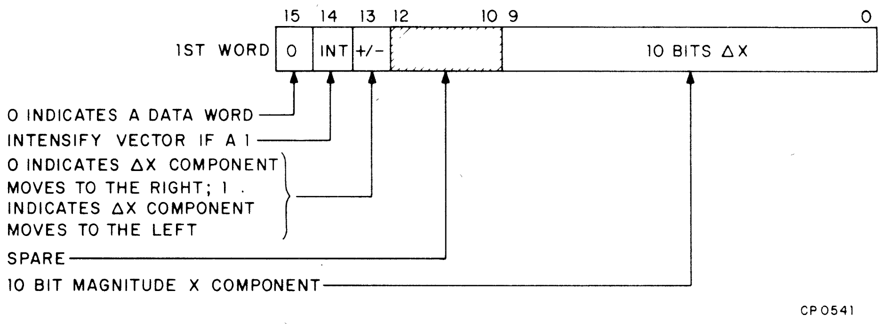
CHARACTER DATA FORMAT- Mode 0000



SHORT VECTOR MODE- Mode 0001


15
14
13
12
7
6
5
0
0
INT
+/-
6 BITS ΔX
+/-
6 BITS ΔY
0 INDICATES A DATA WORD
INTENSIFY VECTOR IF A 1
0 INDICATES ΔX COMPONENT MOVES TO THE RIGHT; 1 INDICATES ΔX COMPONENT MOVES TO THE LEFT
6 BIT MAGNITUDE Y COMPONENT
0 INDICATES ΔY COMPONENT MOVES UP; 1 INDICATES ΔY COMPONENT MOVES DOWN
6 BIT MAGNITUDE Y COMPONENT
CP-0583

LONG VECTOR DATA FORMAT- 0010


15
14
13
12
10
9
0
1ST WORD
0
INT
+/-
10 BITS ΔX
0 INDICATES A DATA WORD
INTENSIFY VECTOR IF A 1
0 INDICATES ΔX COMPONENT MOVES TO THE RIGHT; 1 INDICATES ΔX COMPONENT MOVES TO THE LEFT
SPARE
10 BIT MAGNITUDE X COMPONENT
CP0541

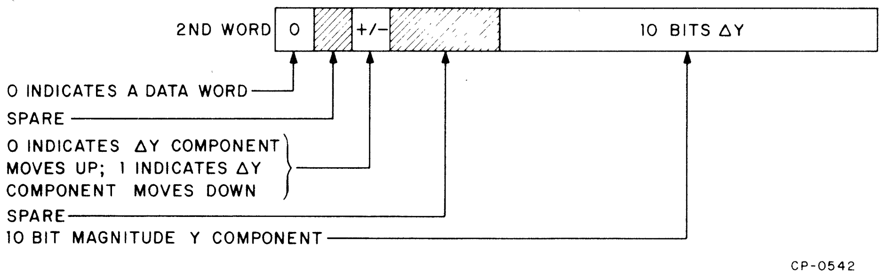
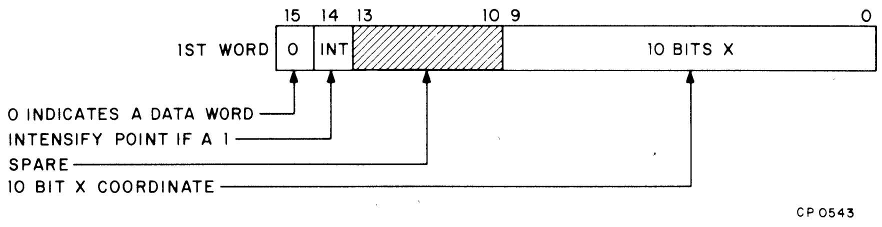
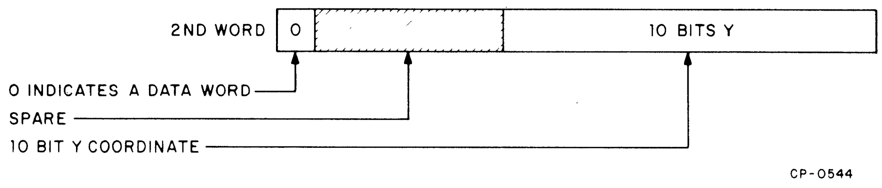

2ND WORD
0
+/-
10 BITS ΔY
0 INDICATES A DATA WORD
SPARE
0 INDICATES ΔY COMPONENT MOVES UP; 1 INDICATES ΔY COMPONENT MOVES DOWN
SPARE
10 BIT MAGNITUDE Y COMPONENT
CP-0542

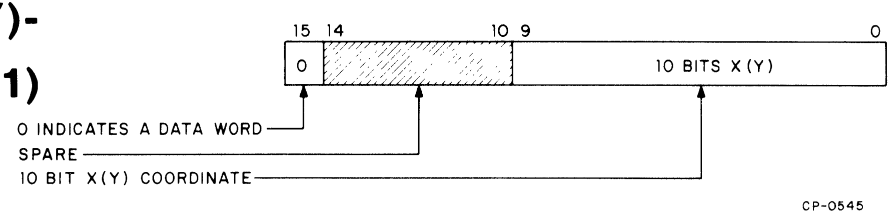
Figure 17 Data Word Formats (Sheet 1 of 2)

POINT DATA MODE- Mode 0011





GRAPHPLOT X(Y)- Mode 0100 (0101)



RELATIVE POINT MODE- Mode 0110

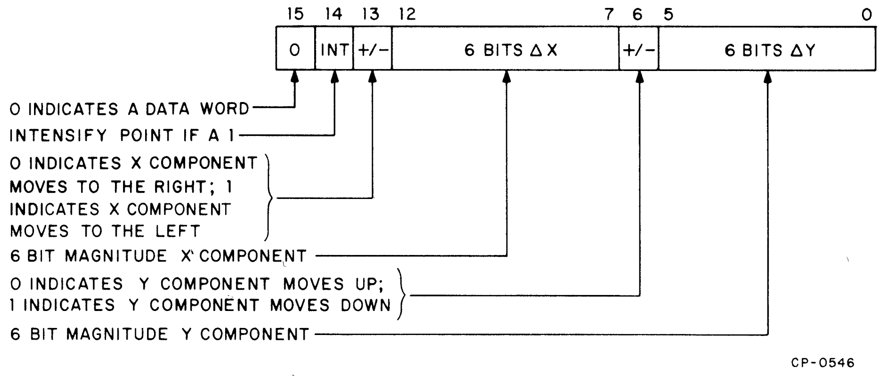

15
14
13
12
7
6
5
0
0
INT
+/-
6 BITS ΔX
+/-
6 BITS ΔY
0 INDICATES A DATA WORD
INTENSIFY POINT IF A 1
0 INDICATES X COMPONENT MOVES TO THE RIGHT; 1 INDICATES X COMPONENT MOVES TO THE LEFT
6 BIT MAGNITUDE X COMPONENT
0 INDICATES Y COMPONENT MOVES UP; 1 INDICATES Y COMPONENT MOVES DOWN
6 BIT MAGNITUDE Y COMPONENT
CP-0546

Figure 17 Data Word Formats (Sheet 2 of 2)

Table 8
Recommended GT40/42 Mnemonics

Group	Mnemonic	=	Value	Function
Group 1				
	CHAR	=	100000	Character Mode
	SHORTV	=	104000	Short Vector Mode
	LONGV	=	110000	Long Vector Mode
	POINT	=	114000	Point Mode
	GRAPHX	=	120000	Graphplot X Mode
	GRAPHY	=	124000	Graphplot Y Mode
	RELATV	=	130000	Relative Point Mode
	INT0	=	2000	Intensity 0 (Dimmest)
	INT1	=	2200	Intensity 1
	INT2	=	2400	Intensity 2
	INT3	=	2600	Intensity 3
	INT4	=	3000	Intensity 4
	INT5	=	3200	Intensity 5
	INT6	=	3400	Intensity 6
	INT7	=	3600	Intensity 7 (Brightest)
	LPOFF	=	100	Light Pen Off
	LPON	=	140	Light Pen On
	BLKOFF	=	20	Blink Off
	BLKON	=	30	Blink On
	LINE0	=	4	Solid Line
	LINE1	=	5	Long Dash
	LINE2	=	6	Short Dash
	LINE3	=	7	Dot Dash
Group 2				
	DJMP	=	160000	Display Jump
Group 3				
	DNOP	=	164000	Display No Operation
Group 4				
	STATSA	=	170000	Load Status A Instruction
	DSTOP	=	173400	Display Stop and Interrupt
	SINON	=	1400	Stop Interrupt On
	SINOF	=	1000	Stop Interrupt Off

Table 8 (Cont)
Recommended GT40/42 Mnemonics

Mnemonic = Value				Function
	LPLITE	=	200	Light Pen Hit On
	LPDARK	=	300	Light Pen Hit Off
	ITAL0	=	40	Italics Off
	ITAL1	=	60	Italics On
	SYNC	=	4	Halt and Resume in Sync
Group 5				
	STATSB	=	174000	Load Status B Instruction
	INCR	=	100	Graphplot Increment
Group 6 (Vector/Point Mode)				
	INTX	=	40000	Intensify Vector or Point
	MAXX	=	1777	Maximum Δ X Component
	MAXY	=	1377	Maximum Δ Y Component
	MINUSX	=	20000	Negative Δ X Component
	MINUSY	=	20000	Negative Δ Y Component
Group 7 (Short Vector Mode)				
	MAXSX	=	17600	Maximum Δ X Component
	MAXSY	=	77	Maximum Δ Y Component
	MISVX	=	20000	Negative Δ X Component
	MISVY	=	100	Negative Δ Y Component

4.6.2 Display File

The following program causes a 200_8 unit box to be drawn with the lower left corner at screen location $500,500_8$. Initially, the DPC is loaded with the starting address. Then the display parameters, e.g., intensity, are established and the mode set to Point. The four vectors are drawn after the Point is executed and, to conclude the file, the last commands reload the DPC with the display file starting address. This results in the display file being re-executed; the CRT display is refreshed.

Address	Instruction/Data	Mnemonic	Comment
		. = 100	
100	012706	MOV #500, R6	Initialize the
102	500		stack pointer
104	012737	MOV #2000, @ #DPC	Load the DPC
106	2000		with SA = 2000
110	172000		
112	000001	WAIT	Wait
2000	117124	.=2000	Point mode, intensity
		POINT+INT4+LPOFF	4, no light pen, no
		+BLKOFF+LINED	blink, solid lines.
2002	500	500	Unintensified point
2004	500	500	at X = 500, Y = 500
2006	110000	LONGV	Long vector mode
2010	40200	200+INTX	Δ X = 200, Δ Y = 0,
2012	0	0	intensified
2014	40000	0+INTX	Δ X = 0, Δ Y = 200,
2016	200	200	intensified
2020	60200	200+INTX+MINUS	Δ X = −200, Δ Y = 0,
2022	0	0	intensified
2024	40000	0+INTX	Δ X = 0, Δ Y = −200,
2026	20200	200+MINUS	intensified
2030	160000	DJMP	Jump to start of
2032	2000	2000	display file.

Note that since the parameters (intensity level, no blink, and line type) are specified in the point instruction, they need not be re-specified in the long vector instruction (2006) because they will not change unless the appropriate enable bits are set. The enable bits also allow the user to change, for example, the line type but not the intensity. In this case, only the line type enable bit is changed, not the intensity enable bit. This retention of current, not-to-be-changed, values saves both execution time and memory storage space.

4.6.3 Application of the Stop Interrupt

The Stop Interrupt provides close interaction between the CPU and the DPU. The following program restarts the display after the halt and interrupt sequence. This occurs at the end of each pass.

Address	Instruction/Data	Mnemonic	Comment
		. = 100	
100	012706	MOV #500, R6	Initialize the
102	500		stack pointer
104	012737	MOV #2000, @ #DPC	Load the DPC with
106	2000		SA = 2000
110	172000		
112	000001	WAIT	Wait for interrupt
114	776	BR.− 2	Jump back one
		. = 320	instruction
320	400	400	Address of next instruction to be executed after a Stop interrupt
322	200	200	Processor status
		. = 400	(BR level 4)
400	012737	MOV #1, @ #DPC	Resume the display

Address	Instruction/Data	Mnemonic	Comment
402	01		
404	172000		
406	02	RTI	Return from interrupt
2000	117124	.=2000 POINT+INT4+LPOFF +BLKOFF+LINED	Point mode, intensity 4, no light pen, no blink, solid lines.
2002	500	500	Unintensified point
2004	500	500	at X = 500, Y = 500
2006	110000	LONGV	Long vector mode
2010	40200	200+INTX	Δ X = 200, Δ Y = 0,
2012	0	0	intensified
2014	40000	0+INTX	Δ X = 0, Δ Y = 200,
2016	200	200	intensified
2020	60200	200+INTX+MINUS	Δ X = -200, Δ Y = 0,
2022	0	0	intensified
2024	40000	0+INTX	Δ X = 0, Δ Y = -200
2026	20200	200+MINUS	intensified
2030	173400	DSTOP	Enable Stop interrupt, Stop
2032	160000	DJMP	Jump to start of
2034	2000	2000	display file after a Resume

After initializing the DPU, the CPU WAITs for an interrupt. The DPU executes the display file, eventually performing the STOP with interrupt enabled. This causes a vectored interrupt to address 320_8.

Since the Stack Pointer was initialized to 500_8, the CPU stores its processor status and program counter in location 500_8 and 476_8 respectively; it pushes them on the "stack." Once stored, the CPU goes to location 320_8 and uses its contents as the address of the interrupt routine. The CPU takes the contents of location 322_8 as its new processor status. In this example, location 400_8 is the address of the interrupt handler and the CPU proceeds to that location.

The interrupt handler simply MOVes the number 1 to the DPC which is interpreted as a RESUME by the DPU. As the DPU resumes operation, it will fetch and interpret the next instruction after stopping, in this case a DJMP, back to the start of the display file. The final instruction of the interrupt handler is a Return from Interrupt (RTI), restoring the CPU to the status and location present before the interrupt, i.e., it pops two words off the stack. A computer branch back one instruction is executed, thus placing the CPU in a WAIT condition again.

4.7 PROGRAMMING RESTRICTIONS

As with any complex system, certain restrictions must be observed by the user if trouble-free operation is to be expected. In the case of the GT40/42, the programmer should be aware of certain programming limitations so that the hardware may be exercised more proficiently without violating hardware rules.

4.7.1 Stop and Sync, Microcoding

Stop and Sync appear in the Load Status A instruction. However, selection of both conditions in any given Load Status A instruction should be avoided. Priorities have been built into the GT40/42 hardware concerning the action on the microcoding of these bits. The rules are as follows:

1. Sync and Stop
   Sync will override Stop. The display will stop but will resume in sync with the line frequency.

2. Stop and Sync with Stop Interrupt Enabled
   Setting Stop with the Stop Interrupt enabled and Sync must be avoided. Under these conditions, the DPU will stop, post an interrupt, and restart automatically in sync with the line frequency. Since the Sync resume happens rather randomly with respect to the interrupt, the effect of this microcoding is undetermined.

4.7.2 Display File Changes

Restarting a Running Display – Restarting the DPU while the DPU is running should be avoided. It is possible to "catch" the DPU in the middle of a bus operation causing inconsistent or undetermined operation.

It is recommended that the DPU be halted with a Stop instruction before restarting it again.

Modification of the File – Dynamic modification of the display file should be avoided when possible. Normally the file can be modified dynamically without consequence. However, it is possible to cause problems when modifying two word instructions such as a Display Jump. For example, if the DPU fetched the first part of a DJMP while the CPU modified the second word, the DPU will process the DJMP order code and will take the modified second word as a correct address, causing the DPU to branch to a non-intended address. It is recommended that the DPU be halted before modifying the display file and that care be exercised in selecting the sequence of commands used to modify the file.

4.7.3 Non-Flicker Display

The quality of the image displayed on the screen is determined by many factors. Primarily, the display is controlled by internal adjustments (contrast, focus, etc.) and the external BRIGHTNESS control on the front panel. However, programming is also instrumental in producing better image quality. The selectable brightness feature, one of the display parameters controlled by the Set Graphic Mode instruction, is one example of the role that programming plays. Another is the control of image flicker, the repetitive dimming and brightening of all vectors and characters on the screen. Flicker, in this case, is caused by a relatively long program execution time, i.e., the time from the beginning of the display frame until the program recycles and the display is repeated. If this time is longer than about 1/30 of a second the screen fluorescence will decay (the image will become dimmer), and then brighten when the next frame begins, to the point where flicker is apparent. When the program time is less than 1/30 second, the display is reintensified before the image dims noticeably and there is no apparent flicker. Consequently, the objective, from a programming standpoint, is not to exceed this (1/30 second) execution period when designing a display program.

Program time, as defined above, and where vectors make up most of the display, is primarily determined by two factors: vector magnitude or length, and the number of vectors in the display frame. The longer the vectors and the greater the number of vectors the longer the display frame will be. Figure 18 shows the allowable limits, considering these two factors, for a flickerless display, defined here as display frames $\leqslant$ 32 ms (about 1/30 second). Note that a third factor is also present: the vector to mode word ratio. If this is a 1:1 ratio, then fewer vectors are allowed because the mode word itself requires time to be decoded – time that must be subtracted from the 32 ms period. However, this time is more efficiently used when the ratio increases, i.e., when a mode word is accompanied by a number of vectors; the total number of allowable vectors is increased. This is shown in Figure 18 as the shaded area for each vector length with the top line being the practical limit. If vector lengths vary, as is usually the case, the total number of each length must be taken into account; the aggregate must not cause the frame time to exceed 32 ms.

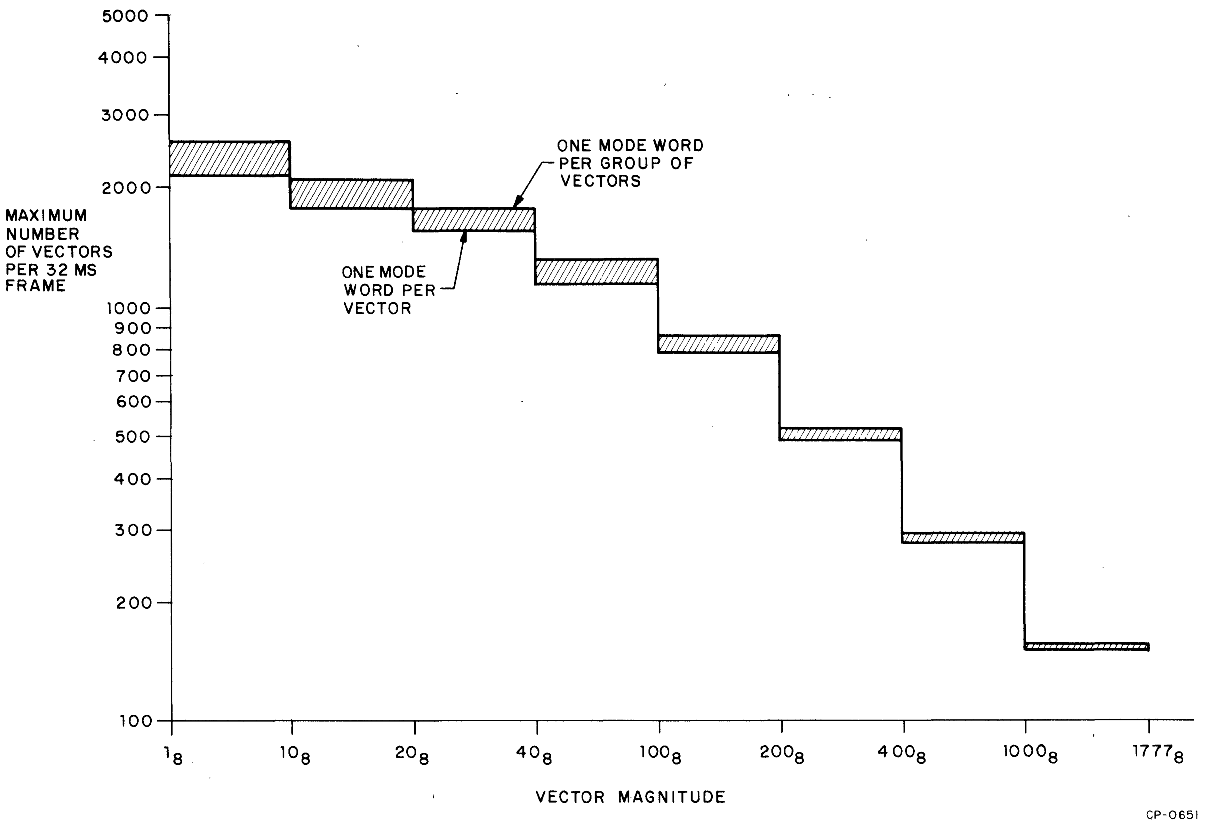


Figure 18 Non-Flicker Display as Determined by Vector Quantity and Magnitude

4.8 ADVANCED PROGRAMMING TECHNIQUES

4.8.1 Subroutines

This programming method is used when a section of display code is repeated a number of times during the execution of a display file. It precludes the need to store multiple copies of the routine in memory and therefore makes more efficient use of available storage space. Writing effective display subroutines is accomplished through use of the stop interrupt instruction (DSTOP) followed by an identifier that informs the interrupt service routine what to do or where to go. Figure 19 shows an example of how a display subroutine can be repeatedly called by the main display file. An example of an interrupt service routine is shown below. It is assumed that register R5 is used for the subroutine stack. STKST is the starting location for the subroutine stack.

	Mnemonic	Comment
STPINT:	TST @ DPC	Test the DPC
	BEQ STOPO	If it contains a valid, non-zero address go to the next instruction; if not go to STOPO
	MOV DPC,-(R5)	Push current DPC on stack
	ADD #2,@ R5	The stack now contains the return address from the subroutine.

```
            Mnemonic                Comment

            MOV @ DPC, DPC          Move address pointed to by DPC into the
                                    DPC, i.e., go to the subroutine.

            RTI                     Exit

STOPO:      CMP R5, STKST           Is the subroutine stack empty?

            BEQ TOP                 Yes, go to top of file

            MOV (R5)+,DPC           No, pop off a word and go there

            RTI                     Exit

TOP:        MOV#START,DPC           Restart at TOP
            RTI                     and exit
```

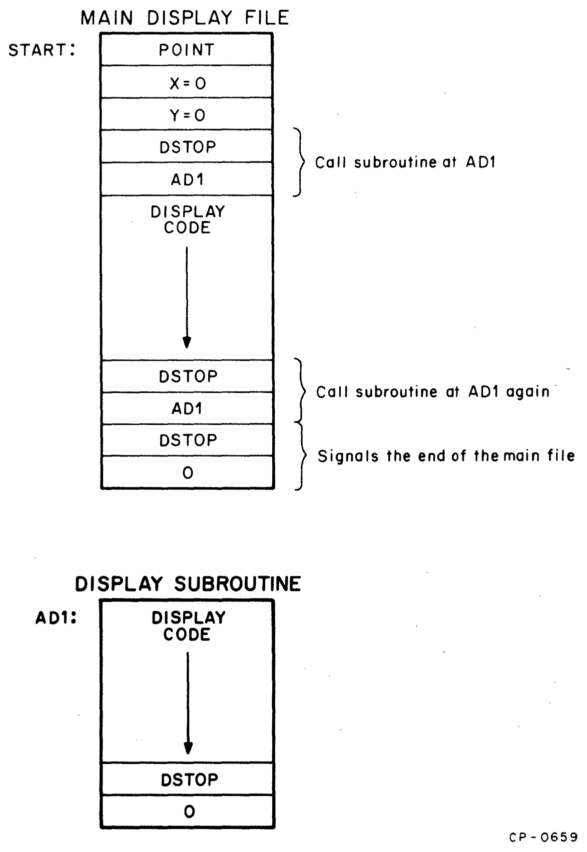


Figure 19 Subroutining Example

4.8.2 Light Pen Interaction

The DPU is stopped when a light pen "hit" occurs during the display of a vector, character, or point, provided light pen interrupts are permitted (bits 5 and 6 of the Set Graphic Mode word must both be true to enable the LP interrupt function).

Priorities permitting, the LP hit interrupts the PDP-11. The interrupt service routine that is called in as a result of the LP interrupt has access to three data in the DPU (the data can be read by specifying the addresses indicated):

- Display Program Counter (DPC) Addr = 172000. Points to the instruction/data word following the data word on which the LP hit occurred.
- The X position of the display at the time the DPU stopped, Addr = 172004. A 10-bit absolute number.
- The Y position of the display at the time the DPU stopped, Addr = 172006. A 10-bit absolute number.

The service routine can respond to the LP interrupt by restarting the display in one of two ways:

- Resume the display – the operation in progress at the time of the interrupt is allowed to continue. Program example: MOV #1, DPC
- Restart the display – the operation in progress at the time of the interrupt is abandoned and a new display program routine is initiated. Program example: MOV #SA, DPC

4.8.3 Special Characters

The 31 special characters in the GT40/42 display character set are addressed through use of ASCII codes Shift Out (016_8) and Shift In (017_8).

When the DPU detects the character code 016_8, the hardware enters the shift mode. In this mode codes 000 through 037_8 are decoded as special characters. (Appendix C contains a list of GT40/42 character codes.) Note that when the DPU is in the shift mode, the Shift Out code (016_8) itself is a legitimate printing character. The DPU is returned to the non-special character ASCII set (non-shift mode) when Shift In is decoded. Unlike the Shift Out code, the Shift In code (017_8) does not cause a special character to be displayed. If, when in the shift mode, the DPU detects a code $\geqslant 040_8$, the PDP-11 is interrupted by a Shift In/Time Out interrupt vector. This is because only the special characters (codes 000 through 037_8) are legal when in the shift mode. The PDP-11 now has access to the 6 low order bits of the 7-bit illegal code. These 6 bits could be used, for example, as an index to a table of software generated characters.

4.8.4 Edge Violations

An edge violation occurs if either the X or Y coordinate indicated for a relative display causes the display to go outside the physical limits of the CRT face. (Vectors, relative points, characters, and Graphplots are classified as relative type displays.) In the event of an edge violation, the edge flag in the status word is set and the display is clipped (terminated) at the edge of the screen; wrap-around does not take place. However, there is one exception in which wrap-around can occur. The GT40/42 hardware is capable of counting only up to 4095_{10}, i.e., 12 bits. Therefore, if the vector position exceeds this 12-bit limit, the count overflows to 0 and wrap-around occurs. For example, if four consecutive vectors with the same coordinates ($\Delta X = 1023, \Delta Y = 1$) are read, only the first vector is displayed; it is the only one that can be displayed within the physical address space. The other three vectors cause the count to legally exceed the 12-bit field. If a fifth vector, with the coordinates of $\Delta X = 10$ and $\Delta Y = 0$, is decoded, the vector will appear on the left of the display; the hardware has caused the display to wrap around. This relative X and Y counting is performed in a 12-bit circular fashion. Absolute points are limited to 10-bit addressing.

5.1 COMMUNICATIONS BOOTSTRAP READ-ONLY MEMORY (ROM)

The communications bootstrap ROM in the GT40 and the GT42 connects the Graphic Display Terminal to a host computer by way of the DL11 Asynchronous Line Interface. Two functions are performed:

1. The program allows ASCII dialogue with the host computer in order to perform such functions as logging in, etc., which presumably leads to

2. The ability to load the Graphic Display Terminal's core memory with an absolute PDP-11 program. This function is typically called a down-line load.

The ROM Bootstrap program is stored in a bipolar ROM contained in the display processor (M7014 module). The memory is assigned addresses starting at 166000_8 and is accessed via the Unibus and the display processor addressing hardware. Although physically located in the display processor, the communications ROM should be considered a separate, Unibus connected, memory device. In the GT40, the ROM contains 256 words; in the GT42, the ROM contains 512 words.

Appendix D contains a program listing of the ROM Bootstrap for the GT40 and Figure D-1 is a flow diagram for the program. Appendix E contains a program listing of the ROM Bootstrap for the GT42 and Figure E-1 is a flow diagram for the program.

5.1.1 Bootstrap Loader

The communications down-line loader portion of the Bootstrap allows loading programs in all memory locations except for the absolute addresses 15700 through 15776_8, which are used by the loader itself. If the user finds this restriction unacceptable, it is possible to reassemble a copy of the Bootstrap program with the tag COREND equal to the highest address in the user's memory, e.g., COREND = 57776_8 for a 12K memory. The procedure then is to load this modified Bootstrap first and then the user's program.

The loader will accept properly encoded ASCII strings and effect the loading of a PDP-11 absolute program. The encoding and decoding scheme is shown pictorially in Figure 20.

The loading procedure, from the host computer, is presented below in brief terms:

1. Initiate the Bootstrap by placing 166000 in the SR switches; press LOAD ADDRESS and START.

2. Transmit } (175_8) and then R (122_8) to reset the Bootstrap.

3. Transmit } (175_8) and then L (114_8) to start the Loader.

4. Transmit encoded characters representing the binary program to be loaded.

5. If a checksum error occurs during a load, B (102_8) and } (175_8) will be returned.

6. If the program loads but does not self-start, G (107_8) and } (175_8) are returned.

7. There is no return if the program is properly loaded and started.

To enable synchronization of the loader at high transfer rates, the host computer should transmit filler characters after step 3 above. These fillers should be nulls in multiples of three, as indicated in Figure 21. The @ symbol (100_8) is transmitted because 100_8 is added to all characters less than 040_8; therefore, null (000) + 100_8 = 100_8. The filler requirement is satisfied by six nulls, i.e., eight @ symbols.

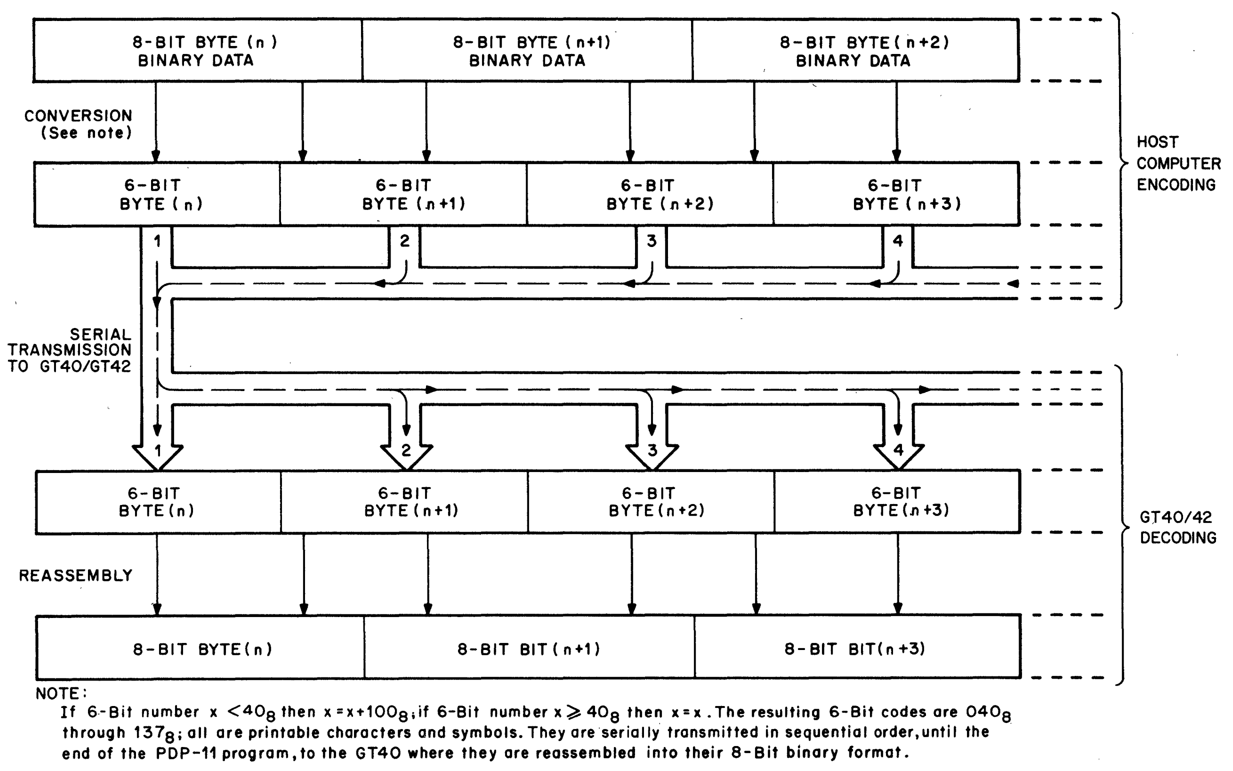

8-BIT BYTE (n) BINARY DATA
8-BIT BYTE (n+1) BINARY DATA
8-BIT BYTE (n+2) BINARY DATA
CONVERSION (See note)
6-BIT BYTE (n)
6-BIT BYTE (n+1)
6-BIT BYTE (n+2)
6-BIT BYTE (n+3)
HOST COMPUTER ENCODING
1
2
3
4
SERIAL TRANSMISSION TO GT40/GT42
GT40/42 DECODING
REASSEMBLY
8-BIT BYTE(n)
8-BIT BIT(n+1)
8-BIT BIT(n+3)
NOTE:
If 6-Bit number x <40₈ then x=x+100₈; if 6-Bit number x ≥ 40₈ then x=x. The resulting 6-Bit codes are 040₈ through 137₈; all are printable characters and symbols. They are serially transmitted in sequential order, until the end of the PDP-11 program, to the GT40 where they are reassembled into their 8-Bit binary format.
CP-0650

Figure 20 Encoding and Decoding of Serial Data

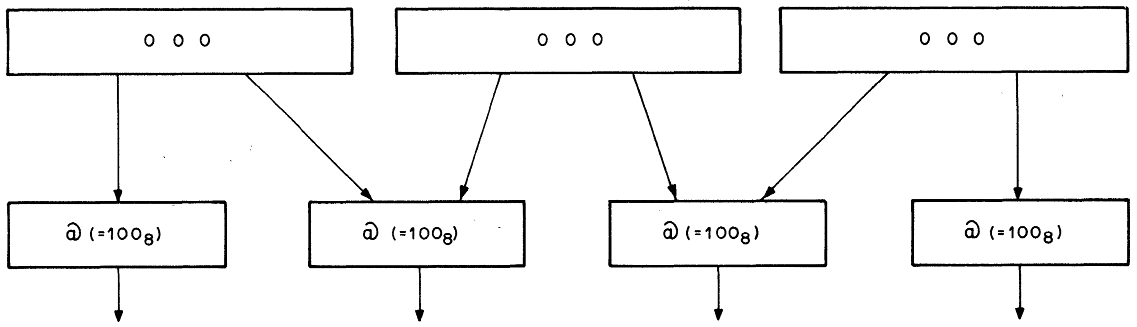


Figure 21 Filler Character Transmission to the GT40/42

It is necessary to preface the first "one" byte in the absolute program with a "zero" byte in order to save Bootstrap code. A normal absolute program, in octal, before encoding into the 6-bit tape format, is transmitted in the order shown in Figure 22. An example of a short program (in octal) and the resultant encoded characters transmitted are shown in Figure 23.

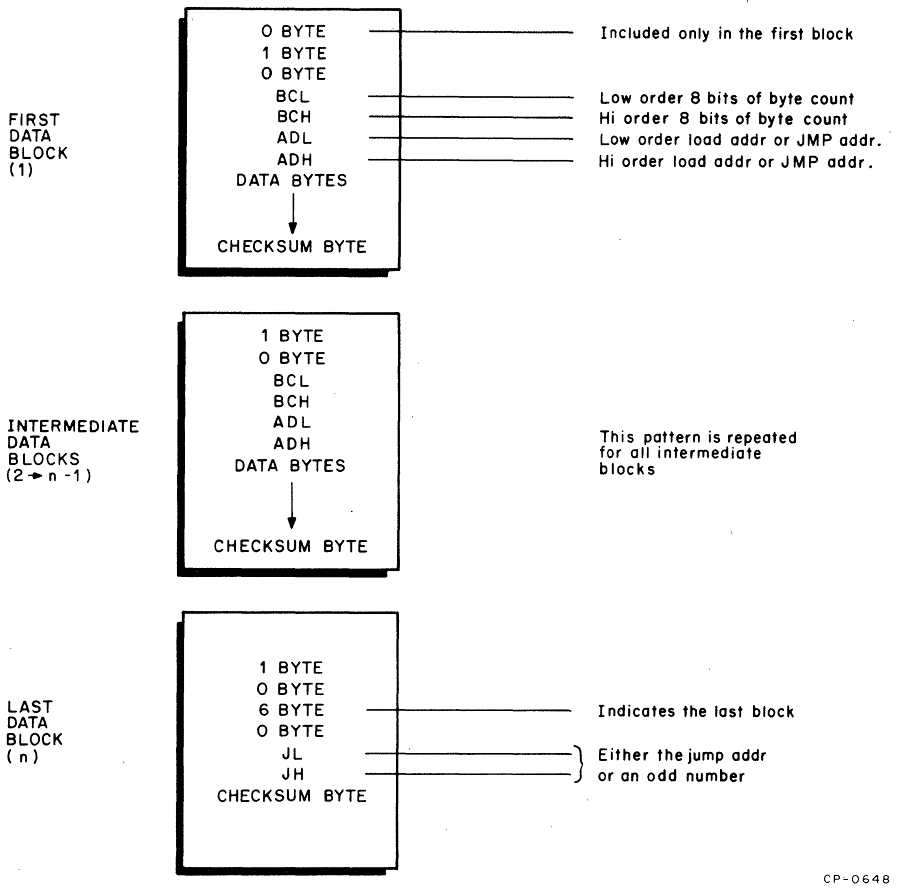

FIRST DATA BLOCK (1)
0 BYTE
1 BYTE
0 BYTE
BCL
BCH
ADL
ADH
DATA BYTES
CHECKSUM BYTE
Included only in the first block
Low order 8 bits of byte count
Hi order 8 bits of byte count
Low order load addr or JMP addr.
Hi order load addr or JMP addr.
INTERMEDIATE DATA BLOCKS (2→n-1)
1 BYTE
0 BYTE
BCL
BCH
ADL
ADH
DATA BYTES
CHECKSUM BYTE
This pattern is repeated for all intermediate blocks
LAST DATA BLOCK (n)
1 BYTE
0 BYTE
6 BYTE
0 BYTE
JL
JH
CHECKSUM BYTE
Indicates the last block
Either the jump addr or an odd number
CP-0648

Figure 22 Absolute Program, Octal Format

5.1.2 Character Echoing

When not running in the LOADER mode, the Bootstrap allows the GT40/42 to communicate with the host computer in ASCII. Depressing a key on the LK40 keyboard at this time causes the ASCII character for that key to be sent to the host computer. If the host computer echoes the character, it will appear on the GT40/42 display (providing it is printable).

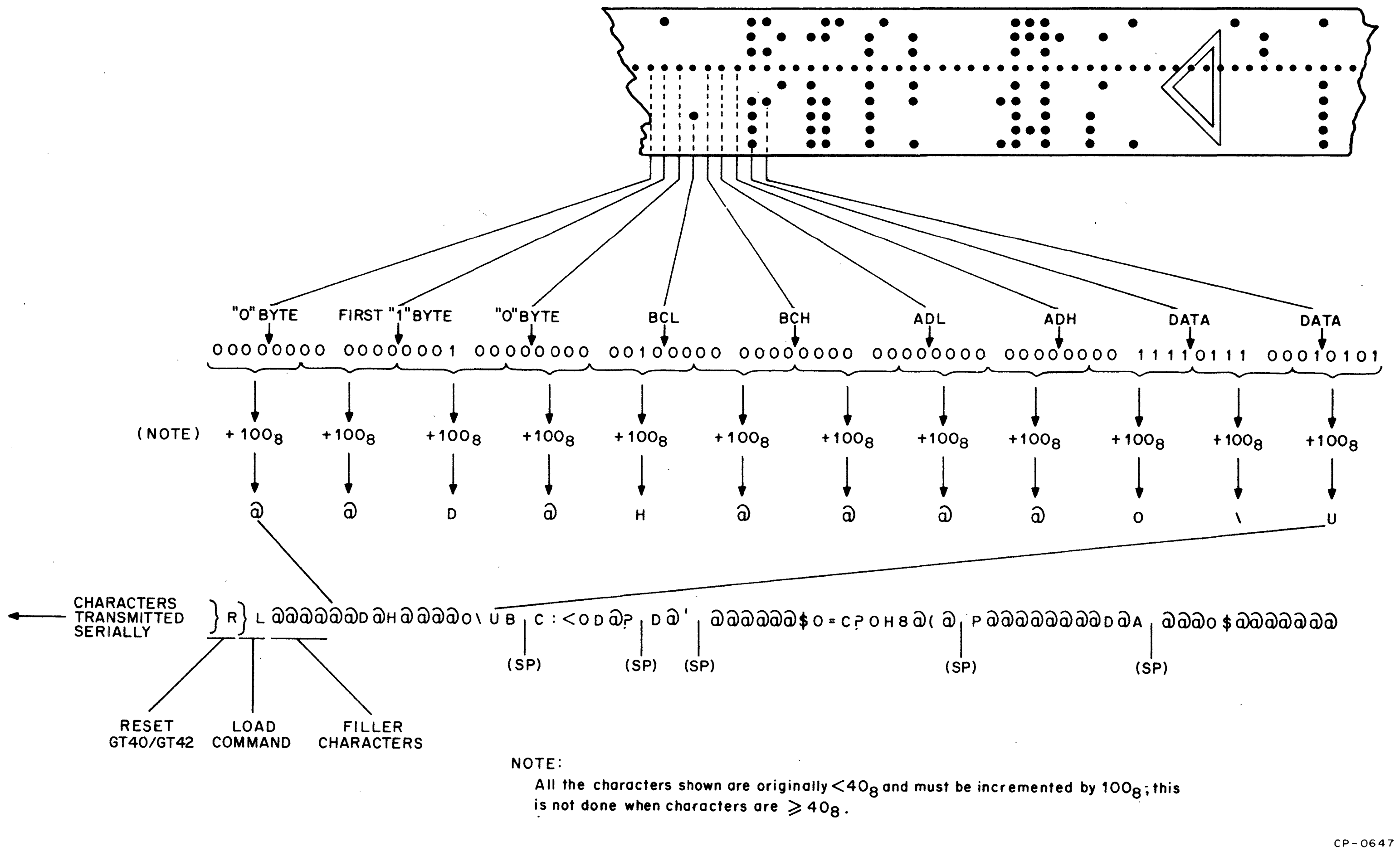
In reference to this type of display, several characteristics should be noted:

- The GT40 Bootstrap does not scroll. If the initial dialogue runs off the bottom of the screen, the operator must again depress START; the dialogue will then return to the top of the screen. In the GT42, the dialogue appears at the bottom of the screen and scrolls off the top when the screen is full.

- With the exception of 175_8 characters with codes of from 040_8 through 176_8 will be displayed on the screen. Code 175_8 is used to initiate restarting and loading of the GT40/42.

- In the GT40 the only control characters which affect the display are CARRIAGE RETURN, LINE FEED, and BACKSPACE. TAB, FORM FEED, etc. are not understood. In the GT42, TAB and FORM FEED characters are understood.

- The host computer should not send SHIFT OUT (016_8) because this character causes the GT40/42 hardware to generate a special character set. (This restriction applies only to the Bootstrap because of space limitations in this program. Normally the software would monitor all characters before inserting them into the display file.)



NOTE:
All the characters shown are originally $<40_8$ and must be incremented by 100_8; this is not done when characters are $\geqslant 40_8$.

Figure 23 Absolute Program Conversion and Transmission

APPENDIX A
KEY BOARD LAYOUT

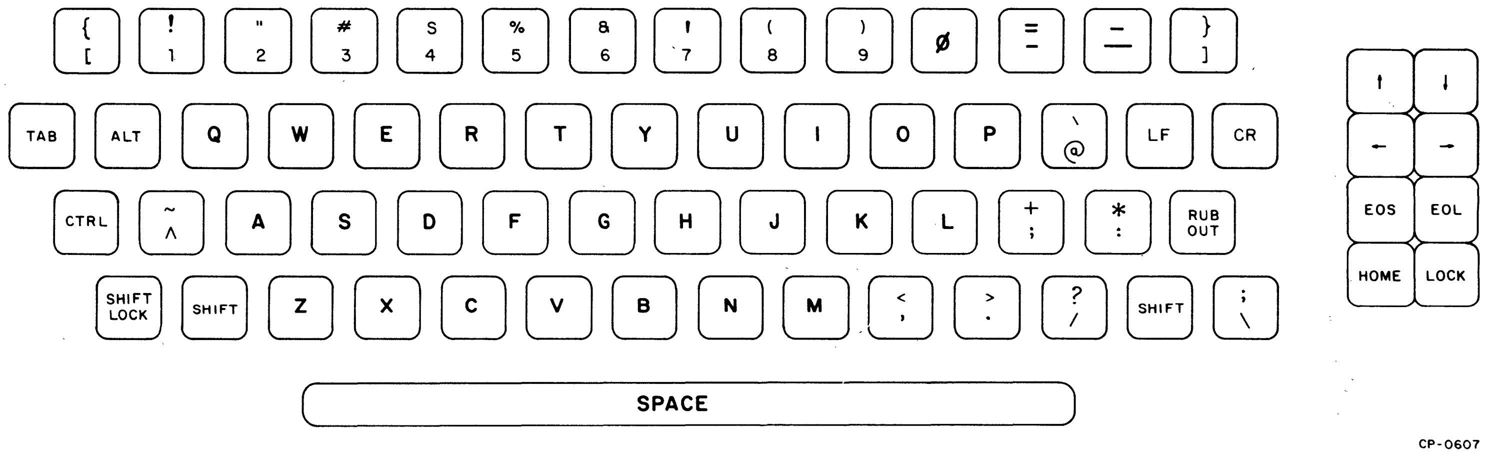


Figure A-1 Keyboard Key Configuration

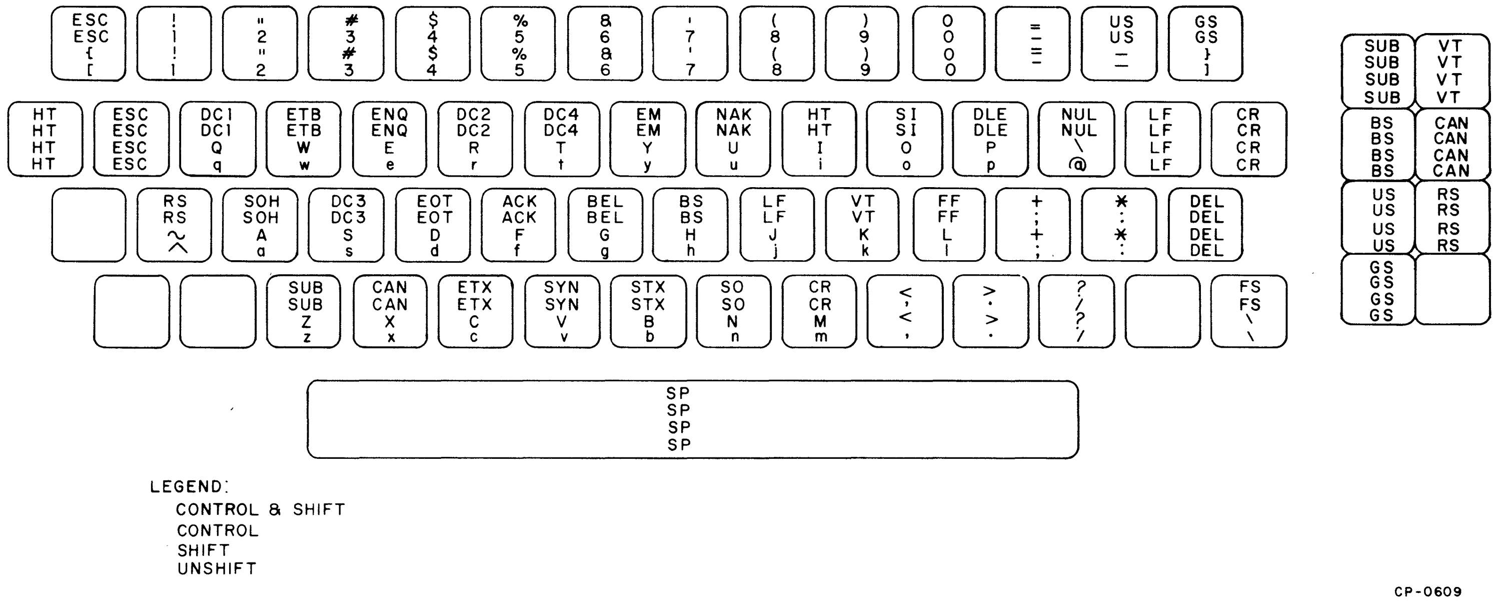


Figure A-2 128-Character Keyboard (Position 1)

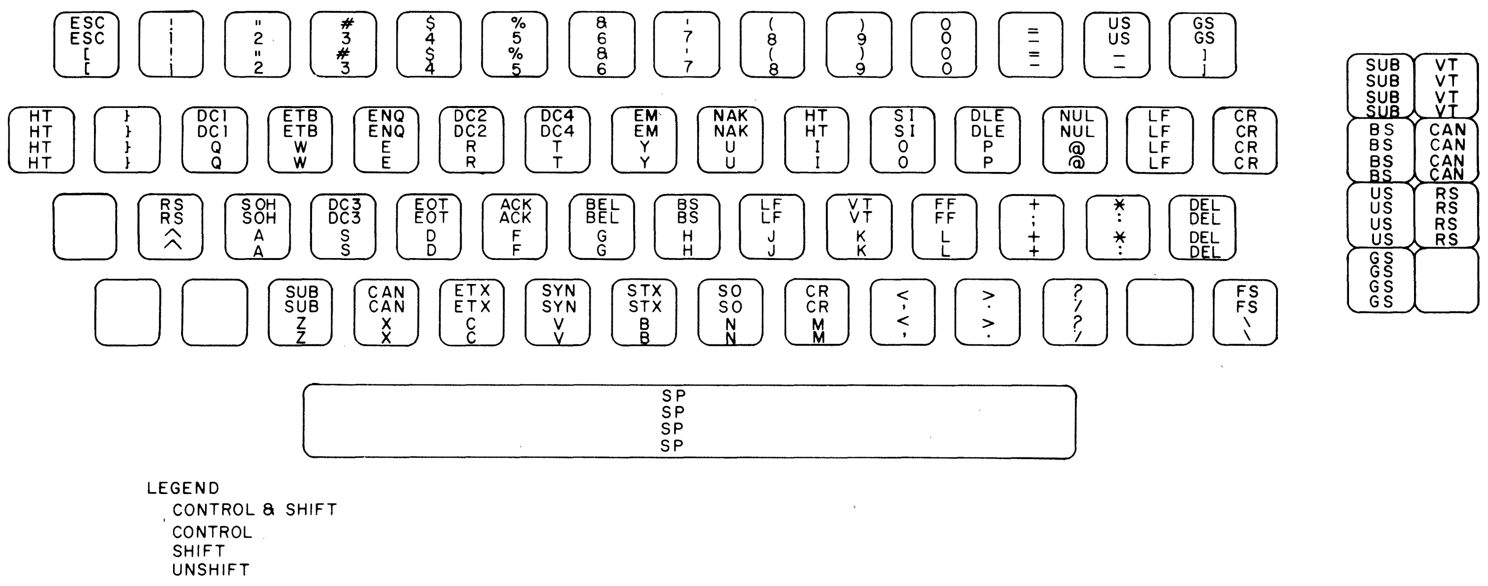


Figure A-3 64-Character Keyboard (Position 2)

APPENDIX B
ADDRESS MAPPING

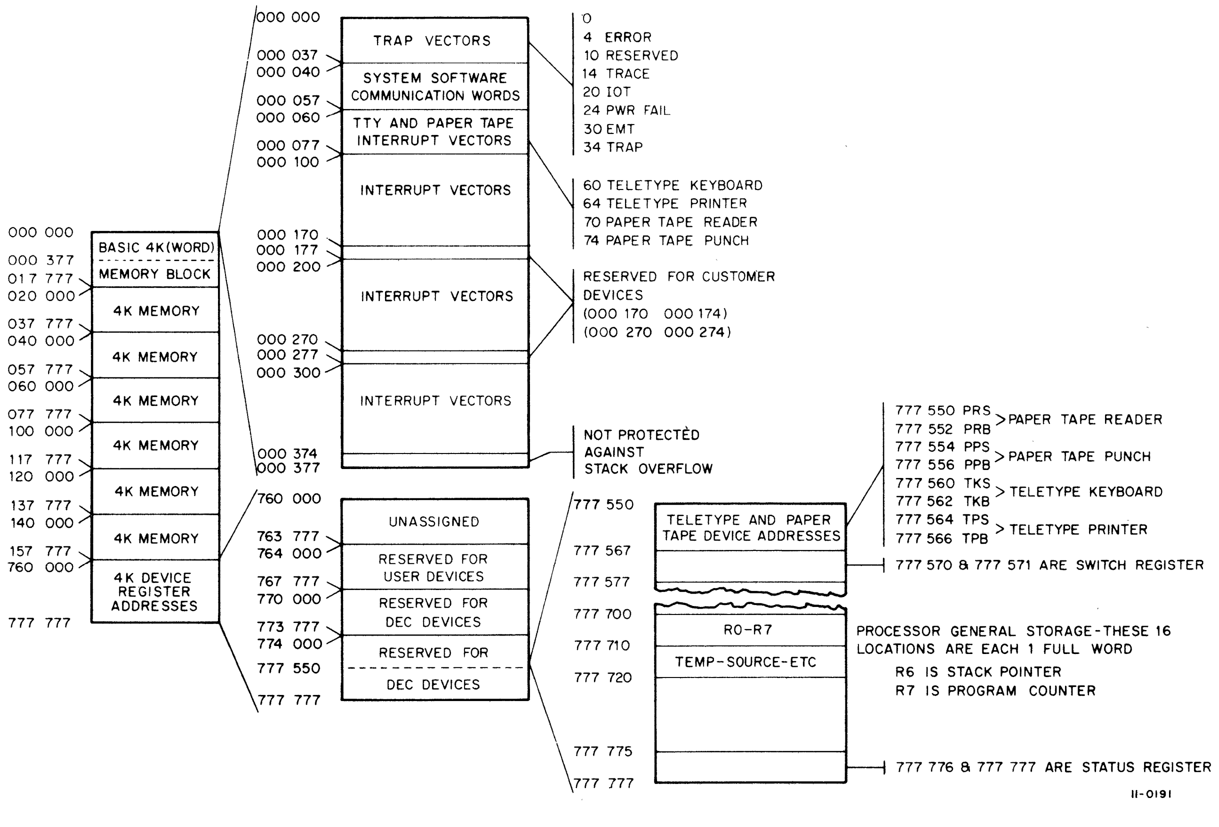


Figure B-1 Address Mapping

APPENDIX C
CHARACTER CODES

7 Bit (octal)	ASCII Representation	Keyboard	GT40/42 Printing	GT40/42 Printing When Preceded By Shift-Out = 016
000	NUL	CTRL @		λ
001	SOH	CTRL A		α
002	STX	CTRL B		ϕ
003	ETX	CTRL C		Σ
004	EOT	CTRL D		δ
005	ENQ	CTRL E		Δ
006	ACK	CTRL F		∿
007	BEL	CTRL G		γ
010	BS	CTRL H	Backspace	∩
011	HT	CTRL I (TAB)		ψ
012	LF	CTRL J (LF)	Line Feed	÷
013	VT	CTRL K		o
014	FF	CTRL L		∴
015	CR	CTRL M (CR)	Carriage Return	μ
016	SO	CTRL N		£
017	SI	CTRL O		Shift In
020	DLE	CTRL P		π
021	DC1	CTRL Q		‖
022	DC2	CTRL R		Ω
023	DC3	CTRL S		σ
024	DC4	CTRL T		ϒ
025	NAK	CTRL U		ϵ
026	SYN	CTRL V		←
027	ETB	CTRL W		→
030	CAN	CTRL X		↑
031	EM	CTRL Y		↓
032	SUM	CTRL Z		┌
033	ESC	CTRL [(ALT)		⊥
034	FS	CTRL \		≠
035	GS	CTRL]		≈
036	RS	CTRL~		∨
037	US	CTRL –		□
40	SP	SPACE BAR	Space 1 character	
41	!	SHIFT 1	!	
42	“	SHIFT 2	“	
43	#	SHIFT 3	#	

7 Bit (octal)	ASCII Representation	Keyboard	GT40/42 Printing	GT40/42 Printing When Preceded By Shift-Out = 016
44	$	SHIFT 4	$	
45	%	SHIFT 5	%	
46	&	SHIFT 6	&	
47	'	SHIFT 7	'	
50	(	SHIFT 8	(	
51	)	SHIFT 9	)	
52	*	SHIFT :	*	
53	+	SHIFT ;	+	
54	,	,	,	
55	– (minus)	–	–	
56	.	.	.	
57	/	/	/	
60	0	0	0	
61	1	1	1	
62	2	2	2	
63	3	3	3	
64	4	4	4	
65	5	5	5	
66	6	6	6	
67	7	7	7	
70	8	8	8	
71	9	9	9	
72	:	:	:	
73	;	;	;	
74	<	SHIFT ,	<	
75	=	SHIFT -	=	
76	>	SHIFT .	>	
77	?	SHIFT /	?	
100	@	@	@	
101	A	SHIFT A	A	
102	B	SHIFT B	B	
103	C	SHIFT C	C	
104	D	SHIFT D	D	
105	E	SHIFT E	E	
106	F	SHIFT F	F	
107	G	SHIFT G	G	
110	H	SHIFT H	H	
111	I	SHIFT I	I	
112	J	SHIFT J	J	
113	K	SHIFT K	K	
114	L	SHIFT L	L	
115	M	SHIFT M	M	
116	N	SHIFT N	N	
117	O	SHIFT O	O	
120	P	SHIFT P	P	
121	Q	SHIFT Q	Q	
122	R	SHIFT R	R	
123	S	SHIFT S	S	
124	T	SHIFT T	T	

7 Bit (octal)	ASCII Representation	Keyboard	GT40/42 Printing	GT40/42 Printing When Preceded By Shift-Out = 016
125	U	SHIFT U	U	
126	V	SHIFT V	V	
127	W	SHIFT W	W	
130	X	SHIFT X	X	
131	Y	SHIFT Y	Y	
132	Z	SHIFT Z	Z	
133	[	[	[	
134	\	\	\	
135	]	]	]	
136	∧	∧	∧	
137	-	-	-	
140	`	SHIFT @	`	
141	a	A	a	
142	b	B	b	
143	c	C	c	
144	d	D	d	
145	e	E	e	
146	f	F	f	
147	g	G	g	
150	h	H	h	
151	i	I	i	
152	j	J	j	
153	k	K	k	
154	l	L	l	
155	m	M	m	
156	n	N	n	
157	o	O	o	
160	p	P	p	
161	q	Q	q	
162	r	R	r	
163	s	S	s	
164	t	T	t	
165	u	U	u	
166	v	V	v	
167	w	W	w	
170	x	X	x	
171	y	Y	y	
172	z	Z	z	
173		SHIFT [	{	
174		SHIFT \	|	
175		SHIFT]	}	
176	~	SHIFT ∧	~	
177	RUB OUT	R.O.	■	

Function Key Codes

← 10	↑ 32	Home 35	EOS 37
→ 30	↓ 33	EOL 36	

APPENDIX D
ROM BOOTSTRAP LOADER PROGRAM - GT40

```
                     ;BOOTVT.S09  5/2/72

                     ;       VT-40 BOOTSTRAP LOADER, VERSION S09, RELEASE R01, 5/2/72

                     ;       COPYRIGHT 1972, DIGITAL EQUIPMENT CORPORATION.
                     ;       146 MAIN STREET
                     ;       MAYNARD, MASSACHUSSETTS
                     ;                               01754

                     ;       WRITTEN BY JACK BURNESS, SENIOR SYSTEMS ARCHITECT!



                     ;       THIS ROUTINE IS INTENDED TO BE LOADED IN THE ROM PORTION OF THE VT-40.


                     ;       REGISTER DEFINITIONS:

        000000               R0=%0
        000001               R1=%1
        000002               R2=%2
        000003               R3=%3
        000004               R4=%4
        000005               R5=%5
        000006               R6=%6
        000007               R7=%7

        000006               SP=R6
        000007               PC=R7

        000000               RET1=R0                         ;RETURN OF VALUE REGISTER.
        000001               INP1=R1                         ;ARGUMENT FOR CALLED FUNCTION
        000002               INP2=R2                         ;SECOND ARGUMENT.
        000003               WORK1=R3                        ;FIRST WORK REGISTER.
        000004               WORK2=R4                        ;SECOND WORKING REGISTER.
        000005               SCR1=R5                         ;SCRATCH REGISTER.

        000003               L.CKSM=WORK1                    ;OVERLAPPING DEFINITIONS FOR LOADER PORTION.
        000000               L.BYT=RET1
        000005               L.BC=SCR1
        000001               L.ADR=INP1



        016000               COREND=16000                    ;FIRST LOCATION OF NON-CORE.
        166000               ROMORG=166000                   ;WHERE THE ROM PROGRAM SHOULD GO.

        000000               STARTX=0                        ;WHERE TO START DISPLAYING THE X POSITIONS.
        001360               STARTY=1360                     ;WHERE TO START DISPLAYING THE Y.
```

D-2

```
        172000                      VT40PC=172000              ;VT40 PROGRAM COUNTER.
        177560                      KBDIS=177560               ;TTY INPUT STATUS.
        175614                      P100S=175614               ;PDP-10 OUTPUT STATUS.
        175610                      P10IS=175610               ;PDP-10 INPUT STATUS.

        177562                      KBDIB=KBDIS+2              ;TTY INPUT BUFFER.
        175612                      P10IB=P10IS+2              ;PDP-10 INPUT CHARACTER.
        175616                      P100B=P100S+2              ;PDP-10 OUTPUT BUFFER.

        015776                      P100C=COREND-2             ;CHARACTER TO BE SENT TO THE PDP-10
        015772                      P10IC=P100C-4              ;INPUT CHARACTER FROM 10 PLUS ONE SAVE CHARACTER
        015770                      STKSRT=P10IC-2             ;FIRST LOCATION OF STACK.

        160000                      JMPDIS=160000              ;THE VT-40 DISPLAY JUMP INSTRUCTION.

        000024                      PWRFAL=24                  ;POWER FAIL RESTART LOCATION.

        166000                      .=ROMORG                   ;SET THE ORIGIN NOW!!!!

166000  012705  000026      START:  MOV     #PWRFAL+2,SCR1     ;PICK UP POINTER TO P.F. STATUS.
166004  005015                      CLR     @SCR1              ;CLEAR IT OUT TO BE SURE.
166006  010745                      MOV     PC,-(SCR1)         ;SET UP THE RESTART LOCATION.

166010  000005                      RESET                      ;RESET THE BUS.

166012  012767  000007  007570      MOV     #7,P10IS           ;INITIALIZE PDP-10 INPUT
166020  012767  000001  011532      MOV     #1,KBDIS           ;INITIALIZE TTY INPUT.
166026  012767  000201  007560      MOV     #201,P100S         ;INITIALIZE PDP-10 OUTPUT.

166034  012706  015770      RESTRT: MOV     #STKSRT,SP         ;SET UP THE STACK NOW!
166040  005001                      CLR     L.ADR              ;CLEAR ADDRESS POINTER.
166042  012702  160000              MOV     #JMPDIS,INP2       ;PLACE A DISPLAY JUMP INSTRUCTION IN A REGISTER.

166046  010221                      MOV     INP2,(L.ADR)+      ;MOVE IT TO LOCATION 0.
166050  012711  166756              MOV     #DISPRG,(L.ADR)    ;MOVE ADDRESS POINTER INTO 2.
166054  012701  000030              MOV     #PWRFAL+4,L.ADR    ;SET UP WHERE WE WILL STORE CHARACTERS.
166060  005000                      CLR     RET1               ;PREPARE TO INSERT A ZERO CHARACTER.
166062  004767  000022              JSR     PC,DOCHAR          ;INSERT IT NOW.
166066  005067  003706              CLR     VT40PC             ;CLEAR THE DISPLAY PROGRAM COUNTER AND START.

166072  004767  000210      MAJOR:  JSR     PC,GETCHR          ;GET A CHARACTER NOW.
166076  000240                      NOP
166100  000240                      NOP
166102  000240                      NOP
166104  012746  166072              MOV     #MAJOR,-(SP)       ;INSERT IN DISPLAY BUFFER NOW.
```

```
166110  010105                DOCHAR: MOV     L.ADR,SCR1              ;GET CURRENT BUUFER POSITION NOW.
166112  022525                        CMP     (SCR1)+,(SCR1)+         ;BYPASS CURRENT DISPLAY JUMP.
166114  005025                        CLR     (SCR1)+                 ;CLEAR FUTURE ADDRESS FOR JUMP.
166116  010225                        MOV     INP2,(SCR1)+            ;STICK IN TEMPORARY JUMP WHILE WE REPLACE CURREN
166120  005015                        CLR     (SCR1)                  ;A DISPLAY JUMP TO ZERO.
166122  005011                        CLR     (L.ADR)                 ;NOW REPLACE CURRENT DISPLAY JUMP BY THE CHARACT
166124  050021                        BIS     RET1,(L.ADR)+           ;IT'S DONE THIS WAY TO WASTE 2 CYCLES.
166126  010211                        MOV     INP2,(L.ADR)            ;TO AVOID TIMING PROBLEMS WITH THE VT40.
166130  000207                        RTS     PC                      ;AND FINALLY RETURN.


166132  004767  000124        GET8:   JSR     PC,GETSIX               ;GET SIX BITS NOW.
166136  010046                        MOV     RET1,-(SP)              ;SAVE THE CHARACTER NOW.
166140  000401                        BR      GETP84                  ;BYPASS THE 8'ER
166142  005002                GET84:  CLR     INP2                    ;RESET THE MAGIC REGISTER NOW.
166144  005722                GETP84: TST     (INP2)+                 ;INCREMENT WHERE TO GO.
166146  066207  166250                ADD     GET8TB(INP2),PC         ;UPDATE PC NOW.

        166152                GET8P=.

166152  004767  000104        GET81:  JSR     PC,GETSIX               ;GET A CHARACTER NOW.
166156  010004                        MOV     RET1,WORK2              ;SAVE FOR A SECOND.
166160  006300                        ASL     RET1
166162  006300                        ASL     RET1                    ;SHIFT TO LEFT OF BYTE
166164  106300                        ASLB    RET1
166166  106116                        ROLB    @SP                     ;PACK THEM IN.
166170  106300                        ASLB    RET1


166172  106116                        ROLB    @SP                     ;A GOOD 8 BIT THING.
166174  012600                        MOV     (SP)+,RET1              ;POP AND RETURN NOW.
166176  000207                        RTS     PC

166200  006300                GET82:  ASL     RET1                    ;WORST CASE. SHIFT 4
166202  006300                        ASL     RET1
166204  106300                        ASLB    RET1
166206  106104                        ROLB    WORK2
166210  106300                        ASLB    RET1
166212  106104                        ROLB    WORK2
166214  106300                        ASLB    RET1
166216  106104                        ROLB    WORK2
166220  106300                        ASLB    RET1
166222  106104                        ROLB    WORK2
166224  010400                        MOV     WORK2,RET1
166226  012604                        MOV     (SP)+,WORK2
166230  000207                        RTS     PC

166232  006100                GET83:  ROL     RET1
166234  006100                        ROL     RET1
166236  006004                        ROR     WORK2
166240  106000                        RORB    RET1
166242  006004                        ROR     WORK2
166244  106000                        RORB    RET1                    ;FINAL CHARACTER ASSEMBLED.
166246  005726                        TST     (SP)+                   ;FUDGE STACK.
166250  000207                        RTS     PC                      ;AND RETURN NOW.
```

```
        166250                 GET8TB  =       .-2                     ;PUSH ZERO CONDITION BACK INTO NEVER-NEVER LAND.

166252  000000                         .WORD   GET81-GET8P
166254  000026                         .WORD   GET82-GET8P
166256  000060                         .WORD   GET83-GET8P
166260  177770                         .WORD   GET84-GET8P


166262  004767  000020         GETSIX: JSR     PC,GETCHR
166266  020027  000040                 CMP     RET1,#40
166272  002546                         BLT     L,BAD
166274  020027  000137                 CMP     RET1,#137
166300  003143                         BGT     L,BAD
166302  000207                         RTS     PC


166304  005726                 GETCHP: TST     (SP)+                   ;UPDATE THE STACK.

166306  012700  015772         GETCHR: MOV     #P10IC,RET1             ;SET UP POINTER TO THE INPUT CHARACTER.
166312  004767  000064         GETCHL: JSR     PC,CHECK
166316  005710                         TST     @RET1                   ;ANY CHARACTERS THERE?
166320  001774                         BEQ     GETCHL
166322  011046                         MOV     @RET1,-(SP)             ;PUSH THE CHAR ON THE STACK.
166324  005020                         CLR     (RET1)+                 ;CLEAR THE CHAR GOT FLAG NOW.
166326  042716  177600                 BIC     #-200,(SP)              ;CLEAR AWAY PARITY NOW.
166332  001764                         BEQ     GETCHP                  ;IF ZERO, GET ANOTHER


166334  022716  000177                 CMP     #177,(SP)
166340  001761                         BEQ     GETCHP                  ;ALSO IGNORE RUBOUTS.
166342  022710  000175                 CMP     #175,@RET1              ;WAS IT A "175"
166346  001007                         BNE     GETNP                   ;NOPE.
166350  011610                         MOV     (SP),@RET1              ;YEP. RESET IN CASE OF ABORT.
166352  021027  000122                 CMP     @RET1,#122              ;IS IT AN R
166356  001626                         BEQ     RESTRT                  ;YEP. RESTART
166360  021027  000114                 CMP     @RET1,#114              ;IS IT AN L
166364  001455                         BEQ     LOAD                    ;YEP. LOAD.

166366  011610                 GETNP:  MOV     (SP),@RET1              ;NOW DO THE FDUGING.
166370  012600                         MOV     (SP)+,RET1
166372  020027  000175                 CMP     RET1,#175
166376  001743                         BEQ     GETCHR                  ;IF ALTMODE, LOOP
166400  000207                         RTS     PC


166402  005767  027370         CHECK:  TST     P10OC                   ;DO WE WANT TO OUTPUT?
166406  001410                         BEQ     CHECK1                  ;NO.
166410  105767  007200                 TSTB    P10OS                   ;WE DO. IS THE 10 READY?
166414  100005                         BPL     CHECK1                  ;NOT QUITE.
166416  016767  027354  007172         MOV     P10OC,P10OB             ;IT'S READY. SEND THE CHARACTER.
166424  005067  027346                 CLR     P10OC                   ;AND THE SAVED CHARACTER.

166430  105767  011124         CHECK1: TSTB    KBDIS                   ;HEY, IS THE KEYBOARD READY?
166434  100014                         BPL     CHECK3                  ;NOPE. NO LUCK.
```

```
166436  116746  011120                  MOVB    KBDIB,-(SP)             ;YEP. SAVE THE CHARACTER NOW.
166442  012767  000001  011110          MOV     #1,KBDIS                ;AND REENABLE THE COMMUNICATIONS DEVICE.

166450  004767  177726          CHECK2: JSR     PC,CHECK                ;IS THE OUTPUT READY?
166454  005767  027316                  TST     P100C
166460  001373                          BNE     CHECK2                  ;IF NOT, WAIT TILL DONE,
166462  012667  007130                  MOV     (SP)+,P100B             ;AND THEN SEND OUT THE CHARACTER.


166466  105767  007116          CHECK3: TSTB    P10IS                   ;IS THE 10 TALKING TO ME.-
166472  100011                          BPL     CHECK4                  ;NOPE. EXIT.
166474  116767  007112  027270          MOVB    P10IB,P10IC             ;GET THE CHARACTER NOW.
166502  052767  177400  027262          BIS     #-400,P10IC             ;MAKE SURE IT'S NONE ZERO.
166510  012767  000007  007072          MOV     #7,P10IS                ;REINITIALIZE COMMUNICATION LINE.

166516  000207                  CHECK4: RTS     PC                      ;AND RETURN.


                                ;       THE     L O A D E R

166520  005002                  LOAD:   CLR     INP2                    ;RESET TO FIRST 8 BIT CHARACTER.
166522  012712  172000                  MOV     #172000,(INP2)          ;AND ALSO CLEVERLY STOP THE VT40.
166526  012706  015770                  MOV     #STKSRT,SP              ;RESET STACK POINTER NOW.

166532  005003                  L.LD2:  CLR     L.CKSM                  ;CLEAR THE CHECKSUM
166534  004767  000070                  JSR     PC,L.PTR                ;GET A BYTE NOW.
166540  105300                          DECB    L.BYT                   ;IS IT ONE?
166542  001373                          BNE     L.LD2                   ;NOPE. WAIT AWHILE
166544  004767  000060                  JSR     PC,L.PTR                ;YEP. GET NEXT CHARACTER.

166550  004767  000072                  JSR     PC,L.GWRD               ;GET A WORD.
166554  010005                          MOV     L.BYT,L.BC              ;GET THE COUNTER NOW.
166556  162705  000004                  SUB     #4,L.BC                 ;CHOP OFF EXTRA STUFF.
166562  022705  000002                  CMP     #2,L.BC                 ;NULL?
166566  001437                          BEQ     L.JMP                   ;YEP. MUST BE END.
166570  004767  000052                  JSR     PC,L.GWRD               ;NOPE. GET THE ADDRESS.
166574  010001                          MOV     L.BYT,L.ADR             ;AND REMEMBER FOR OLD TIMES SAKE.

166576  004767  000026          L.LD3:  JSR     PC,L.PTR                ;GET A BYTE (DATA)
166602  002010                          BGE     L.LD4                   ;ALL DONE WITH THE COUNTER?
166604  105703                          TSTB    L.CKSM                  ;YEP. GOOD CHECK SUM?
166606  001751                          BEQ     L.LD2                   ;NOPE. LOAD ERROR.

166610  012700                  L.BAD:  MOV     (PC)+,RET1              ;SEND OUT SOME CHARACTERS NOW.
166612     175     102                  .BYTE   175,102                 ;"CTRL BAD"
166614  004767  000110                  JSR     PC,SENDIT
166620  000167  177210                  JMP     RESTRT

166624  110021                  L.LD4:  MOVB    L.BYT,(L.ADR)+          ;PLACE THE BYTE IN CORE.
166626  000763                          BR      L.LD3                   ;GET ANOTHER ONE.

166630  004767  177276          L.PTR:  JSR     PC,GET8                 ;GET 8 BITS NOW.
166634  060003                          ADD     L.BYT,L.CKSM            ;UPDATE CHECKSUM
166636  042700  177400                  BIC     #177400,L.BYT           ;CLEAN UP THE BYTE NOW.
166642  005305                          DEC     L.BC                    ;UPDATE THE COUNTER.
166644  000207                          RTS     PC                      ;RETURN NOW.
```

```
166646  004767  177756          L.GWRD: JSR     PC,L.PTR                ;GET A CHARACTER.
166652  010046                          MOV     L.BYT,-(SP)             ;SAVE FOR A SECOND.
166654  004767  177750                  JSR     PC,L.PTR                ;GET ANOTHER CHARACTER.
166660  000300                          SWAB    L.BYT                   ;NOW ASSEMBLE THE WORD.
166662  052600                          BIS     (SP)+,L.BYT             ;AND RETURN WITH A 16 BITER.


166664  000207                          RTS     PC

166666  004767  177754          L.JMP:  JSR     PC,L.GWRD               ;GET A WORD
166672  010046                          MOV     L.BYT,-(SP)             ;SAVE ON THE STACK.
166674  004767  177730                  JSR     PC,L.PTR                ;GET A CHARCTER.
166700  105703                          TSTB    L.CKSM                  ;IS IT ZERO?
166702  001342                          BNE     L.BAD                   ;YEP. WHAT CRAP.
166704  032716  000001                  BIT     #1,(SP)                 ;IS IT ODD?
166710  001406                          BEQ     L.JMP1                  ;YEP. START PROGRAM GOING NOW.
166712  012700                          MOV     (PC)+,RET1              ;TELL PDP-10 WE'VE LOADED OK.
166714     175     107                  .BYTE   175,107
166716  004767  000006                  JSR     PC,SENDIT
166722  000000                          HALT
166724  000776                          BR      .-2

166726  000136                  L.JMP1: JMP     @(SP)+                  ;AND AWAY WE GO.


166730  004767  177446          SENDIT: JSR     PC,CHECK                ;POLL THE OUTPUT DEVICE NOW.
166734  005767  027036                  TST     P10OC                   ;OUTPUT CLEAR?
166740  001373                          BNE     SENDIT                  ;NOPE. LOOP AWHILE LONGER.
166742  010067  006650                  MOV     RET1,P10OB              ;SEND OUT THE CHARACTER.
166746  105000                          CLRB    RET1                    ;CLEAR THE BYTE.
166750  000300                          SWAB    RET1                    ;AND SWAP THEM NOW.
166752  001366                          BNE     SENDIT                  ;IF NOT EQUAL, REPEAT.
166754  000207                          RTS     PC

                                ;               THIS IS THE INITIALIZING VT40 PROGRAM WHICH WILL
                                ;               JUMP TO THE PROGRAM AFTER THE POWER FAIL LOCATIONS
                                ;               WHICH WILL JUMP TO ZERO WHICH WILL JUMP BACK TO HERE.



166756  170256                  DISPRG: .WORD   170256                  ;LOAD STATUS REGISTER FOR NORMAL OPERATION.
166760  115124                          .WORD   115124                  ;SET POINT MODE, "NORMAL".
166762  000000                          .WORD   STARTX                  ;X COORDINATE
166764  001360                          .WORD   STARTY                  ;Y COORDINATE
166766  100000                          .WORD   100000                  ;SET CHARACTER MODE.
166770  160000                          .WORD   JMPDIS                  ;THEN JUMP TO THE POWERFAIL LOCATION.
166772  000030                          .WORD   PWRFAL+4                ;TO DISPLAY USERS CHARACTERS.

        000001                          .END
```

```
                    SYMBOL TABLE

CHECK     166402        CHECK1    166430        CHECK2    166450        CHECK3    166466
CHECK4    166516        COREND =  016000        DISPRG    166756        DOCHAR    166110
GETCHL    166312        GETCHP    166304        GETCHR    166306        GETNP     166366
GETP84    166144        GETSIX    166262        GET8      166132        GET8P  =  166152
GET8TB =  166250        GET81     166152        GET82     166200        GET83     166232
GET84     166142        INP1     =%000001       INP2     =%000002       JMPDIS =  160000
KBDIB  =  177562        KBDIS  =  177560        LOAD      166520        L.ADR    =%000001
L.BAD     166610        L.BC     =%000005       L.BYT    =%000000       L.CKSM   =%000003
L.GWRD    166646        L.JMP     166666        L.JMP1    166726        L.LD2     166532
L.LD3     166576        L.LD4     166624        L.PTR     166630        MAJOR     166072
PC       =%000007       PWRFAL =  000024        P10IB  =  175612        P10IC  =  015772
P10IS  =  175610        P100B  =  175616        P100C  =  015776        P100S  =  175614
RESTRT    166034        RET1     =%000000       ROMORG =  166000        R0       =%000000
R1       =%000001       R2       =%000002       R3       =%000003       R4       =%000004
R5       =%000005       R6       =%000006       R7       =%000007       SCR1     =%000005
SENDIT    166730        SP       =%000006       START     166000        STARTX =  000000
STARTY =  001360        STKSRT =  015770        VT40PC =  172000        WORK1    =%000003
WORK2    =%000004       .      =  166774
```

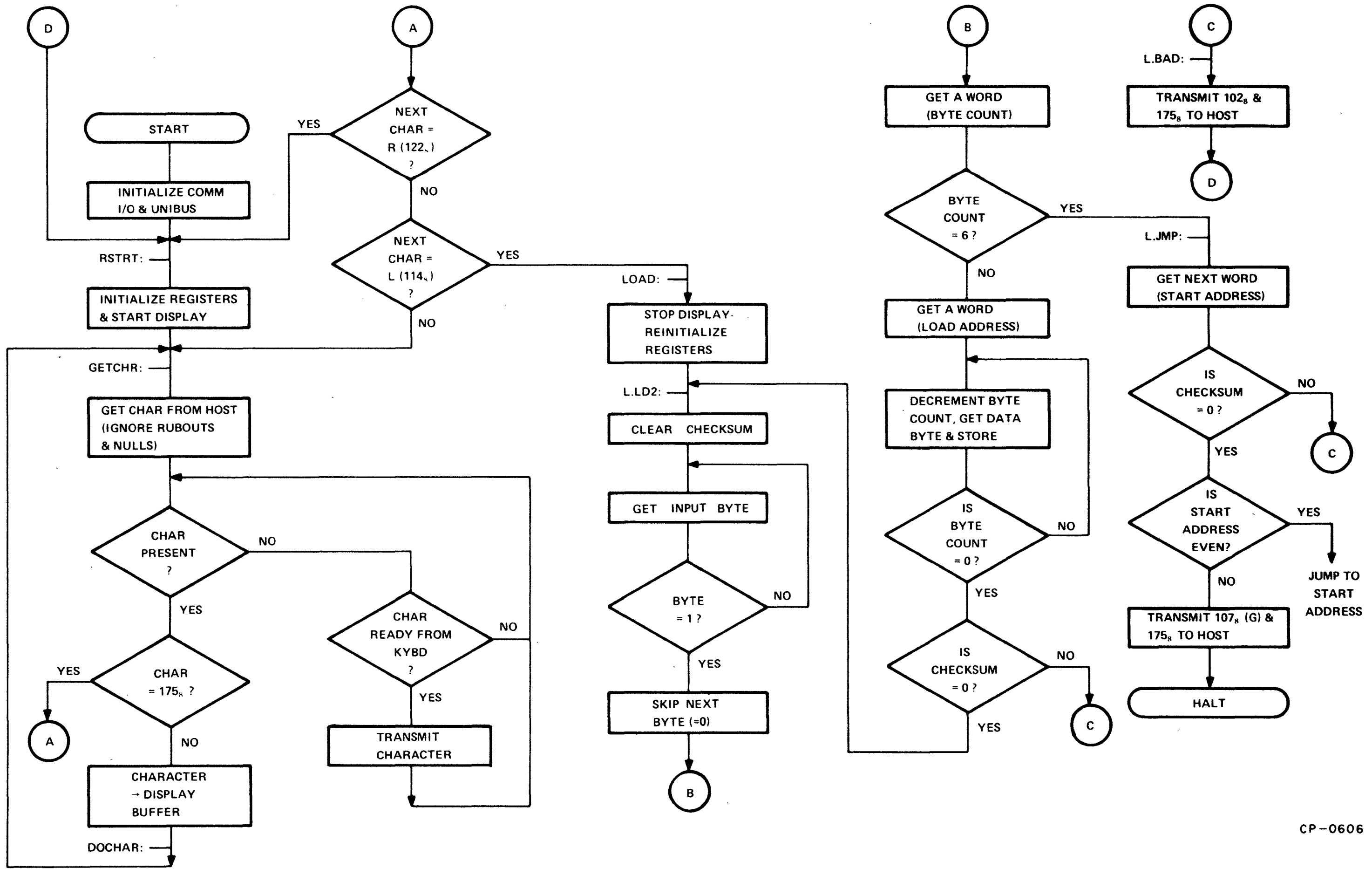


Figure D-1 Communications Bootstrap Loader Flow Diagram

APPENDIX E
SCROLLING ROM BOOTSTRAP LOADER PROGRAM - GT42

```
                            .TITLE  SCROLLING ROM BOOTSTRAP FOR THE GT40







                    ;                          BOOTGT.T15      JUNE 24,1973




                    ;               COPYRIGHT 1973, DIGITAL EQUIPMENT CORPORATION
                    ;               146 MAIN STREET
                    ;               MAYNARD, MASSACHUSETTS
                    ;                                          01754




                    ;               WRITTEN BY JACK BURNESS.





                    ;               THIS PROGRAM IS THE SECOND VERSION THE THE ROM BOOTSTRAP FOR
                    ;       THE GT40 DISPLAY TERMINAL. IT INCLUDES SCROLLING AND AN END OF
                    ;       MEMORY SEARCH FOR THE LOADER.




                            .ENABL  ABS,AMA                 ;ASSEMBLER DIRECTIVES FOR ABSOLUTE BINARY OUTPUT
                                                            ;  NOTE: USE "MACDLX" TO ASSEMBLE THIS PROGRAM.

















                            .SBTTL  DEFINITION SECTION
```

```



                                 ;                        REGISTER DEFINITIONS
                                 ;                        -------- -----------


                                 ;               BASIC DEFINTIONS
                                 ;               ----- ----------


              000000                    R0=%0                          ;DEFINE STANDARD VALUES.
              000001                    R1=%1
              000002                    R2=%2
              000003                    R3=%3
              000004                    R4=%4
              000005                    R5=%5
              000006                    SP=%6
              000007                    PC=%7



                                 ;               GT40 DEFINTIONS
                                 ;               ---- ----------


              000000                    CHAR=R0                        ;CONTAINS THE INPUT CHARACTER.
              000001                    POINTR=R1                      ;POINTS TO NEXT INSERTION BYTE IN DISPLAY BUFFER
              000002                    TABCNT=R2                      ;CHARACTER COUNTER FOR THE "TAB" FEATURE.
              000003                    SCAN=R3                        ;GENERALLY CONTAINS A POINTER WHICH
                                                                       ;IS USED WHEN SCANNING MEMORY FOR SOMETHING.
              000004                    HOLD=R4                        ;TYPICALLY A TEMPORARY WHICH IS USED TO RETAIN
                                                                       ;A VALUE FOR A SHORT TIME.
              000005                    COUNTR=R5                      ;TYPICALLY USED AS A COUNTER.





                                 ;               LOADER DEFINITIONS
                                 ;               ------ -----------





              000000                    L.BYT=CHAR                     ;CHARACTER INPUT FOR THE LOADER.
              000001                    L.ADR=POINTR                   ;CURRENT MEMORY ADDRESS TO BE LOADED.
              000002                    L.BC=TABCNT                    ;NUMBER OF DATA ITEMS TO LOAD.
              000005                    L.CKSM=COUNTR                  ;CHECKSUM ON THE INPUT DATA.
              000003                    INDEX=SCAN                     ;INDICATES HOW TO ASSEMBLE THE 8 BIT CHARACTER.


```

```
                                        ;                         MAJOR SYSTEM DEFINITIONS
                                        ;                         ----- ------ -----------


        166000                   ORIGIN=166000                    ;ORIGIN OF THE BOOTSTRAP.

        175610                   DL11IS=175610                    ;INPUT STATUS REGISTER OF DL11
        175612                   DL11IB=DL11IS+2                  ;INPUT CHARACTER FROM DL11
        175614                   DL11OS=DL11IB+2                  ;OUTPUT STATUS OF THE DL11
        175616                   DL11OB=DL11OS+2                  ;OUTPUT CHARACTER TO THE DL11

        177560                   KBDIS=177560                     ;KEYBOARD INPUT STATUS
        177562                   KBDIB=KBDIS+2                    ;CURRENT CHARACTER FROM KEYBOARD.

        172000                   GT40PC=172000                    ;GT40 PROGRAM COUNTER.
        172002                   GT40SR=GT40PC+2                  ;GT40 STATUS REGISTER ADDRESS.


        001000                   BSTART=1000                      ;START OF THE DISPLAY BUFFER
        007000                   BLIMIT=7000                      ;APPROXIMATE END OF THE DISPLAY BUFFER.
        007776                   TMPEND=7776                      ;LOCATION OF INITIALIZATION STACK.
        000004                   CORSTR=4                         ;LOCATION OF PDP-11 TRAP VECTOR.
        007012                   JMPADD=BLIMIT+10.                ;WHERE THE POINTER IS TO FIRST CHAR ON SCREEN
        000040                   NUMLIN=32.                       ;NUMBER OF LINES ON TEXT TO SHOW ON THE SCREEN

        005015                   CRLF=5015                        ;CARRIAGE RETURN - LINE FEED
        000175                   ALTMOD=175                       ;THE "KEY" CHARACTER [I.E. ALTMODE].

        160000                   DISJMP=160000                    ;THE GT40 JMP INSTRUCTION
        173000                   DISTOP=173000                    ;THE GT40 STOP DISPLAY INSTRUCTION.



















                                 .SBTTL  INITIALIZATION AND RESTART CODE
```

```



                                  ;               GT4Ø BOOTSTRAP CODE
                                  ;               ---- --------- ----





          1660ØØ                            .=ORIGIN                           ;DEFINE ORIGIN OF THE BOOTSTRAP.







                                  ;               COLD INITIALIZATION CODE
                                  ;               ---- -------------- ----



  1660ØØ  ØØØØØ5                  START:  RESET                              ;RESET ALL HARDWARE NOW.
  1660Ø2  Ø12737  ØØØØØ7  17561Ø          MOV     #7,DL11IS                  ;INITIALIZE DL-11 INPUT NOW.
  16601Ø  Ø127Ø6  ØØ7776                  MOV     #TMPEND,SP                 ;A GOOD TEMPORARY STACK
  166Ø14  ØØ5237  175614                  INC     DL11OS          ;SET BREAK BIT
  166Ø2Ø  ØØ4337  166652                  JSR     SCAN,OUTLIT                ;FOR 2 CHARACTER TIMES
  166Ø24  ØØØØØØ                          .WORD Ø                            ;SNED TWO ZERO'S

  166Ø26  Ø127Ø3  ØØØØØ4                  MOV     #CORSTR,SCAN               ;GET ADDRESS OF BAD CORE TRAP VECTOR.
  166Ø32  Ø12723  166Ø42                  MOV     #NOTHERE,(SCAN)+           ;AND INSERT A POINTER TO US THERE.

  166Ø36  ØØ5Ø23                  ENDCOR: CLR     (SCAN)+                    ;NOW CLEAR ALL OF MEMORY BEYOND THE POINTER,
  166Ø4Ø  ØØØ776                          BR      ENDCOR                     ;UNTIL WE RUN OUT OF MEMORY AND TRAP.


  166Ø42  ØØ5743                  NOTHER: TST     -(SCAN)                    ;WHEN WE TRAP OUT, WE COME HERE.
                                                                             ;WE BACK UP POINTER TO GOOD CORE.
                                                                             ;NOTE THAT IF WE TRAP OUT AGAIN, IT
                                                                             ;IS STILL OK, BECAUSE WE WILL LOOP
                                                                             ;UNTIL WE GET A GOOD CORE ADDRESS.
  166Ø44  Ø1Ø3Ø6                          MOV     SCAN,SP                    ;WHEN WE GET ONE, THAT IS LAST LOCATION
                                                                             ;IN THE MACHINE, AND HENCE OUR SP.
  166Ø46  1Ø5737  175614          1$:     TSTB    DL11OS                     ;SEE IF BREAK IS DONE
  166Ø52  1ØØ375                          BPL     1$                         ;NO GO BACK
  166Ø54  ØØ5Ø37  175614                  CLR     DL11OS                     ;CLEAR BREAK BIT






                                  ;               RESTART INITIALIZATION CODE WHEN COMMUNICATIONS IS WORKING.
                                  ;               ------- -------------- ---- ---- -------------- -- -------
```

```
                                                           ------- ------------- ---- ---- ------------- -- -------



166Ø6Ø  Ø527Ø6  ØØ7776          RESTRT: BIS     #TMPEND,SP              ;FORCE THE SP TO LIMIT OF EXISTING CORE.


166Ø64  Ø127Ø3  ØØ67ØØ                  MOV     #BLIMIT-NUMLIN-NUMLIN,SCAN      ;NOW WE WILL FILL THE KEY AREAS OF THE
166Ø7Ø  Ø127Ø2  ØØØØ4Ø                  MOV     #NUMLIN,TABCNT          ;DISPLAY BUFFER WITH INITIAL CR-LF'S.

166Ø74  Ø12723  ØØ5Ø15          SETLP1: MOV     #CRLF,(SCAN)+           ;INSERT A CRLF NOW.
1661ØØ  ØØ53Ø2                          DEC     TABCNT                  ;AND LOOP UNTIL DONE.
1661Ø2  ØØ3374                          BGT     SETLP1                  ;THUS DISPLAY CORE IS ALMOST CORRECT.


1661Ø4  Ø127Ø3  166432                  MOV     #SETUP,SCAN             ;NOW WE WILL INITALIZE CORE FOR THE
                                                                        ;DISPLAY. PICK UP POINTER TO LIST.

16611Ø  Ø123Ø2                  SETLP2: MOV     (SCAN)+,TABCNT          ;GET NUMBER OF ITEMS TO INSERT.
166112  ØØ14Ø5                          BEQ     SETDUN                  ;IF ZERO, WE ARE DONE.
166114  Ø123Ø1                          MOV     (SCAN)+,POINTR          ;PICK UP FIRST CORE ADDRESS POINTER.

166116  Ø12321                  SETLP3: MOV     (SCAN)+,(POINTR)+       ;MOVE OVER A DATA ITEM NOW.
16612Ø  ØØ53Ø2                          DEC     TABCNT                  ;ALL DONE?
166122  ØØ3375                          BGT     SETLP3                  ;NOPE, MOVE OVER THE NEXT.
166124  ØØØ771                          BR      SETLP2                  ;YES, GET NEXT MAJOR LIST TO INSERT.


166126  Ø127Ø1  ØØ6776          SETDUN: MOV     #BLIMIT-2,POINTR        ;ESTABLISH THE BUFFER POINTER NOW.












                                        .SBTTL  VTØ5 SIMULATOR
```

```



                                   ;           VTØ5 (SCROLLING) PORTION OF THE BOOTSTRAP
                                   ;           ----  -----------  ------- -- --- ---------



 166132  ØØ4737  166564    NXTCHR: JSR     PC,GETCHR               ;GET A CHARACTER NOW.
 166136  Ø2ØØ27  ØØØ177            CMP     CHAR,#177               ;IS IT OUT OF RANGE?
 166142  ØØ2373                    BGE     NXTCHR                  ;YEP, GET ANOTHER ONE.
 166144  Ø2ØØ27  ØØØØ4Ø            CMP     CHAR,#4Ø                ;IS IT A PRINTING CHARACTER?
 16615Ø  ØØ2Ø2Ø                    BGE     NORMAL                  ;YES, IT'S A NORMAL PRINTING CHARACTER.
 166152  Ø1ØØØ3                    MOV     CHAR,SCAN               ;MOVE IT OVER SO WE CAN PLAY WITH IT.
 166154  1627Ø3  ØØØØØ7            SUB     #7,SCAN                 ;BIAS SO THAT BELL [7] IS ZERO.
 16616Ø  Ø2Ø327  ØØØØØ7            CMP     SCAN,#7                 ;IF CHARACTER IS LESS THEN BELL OR
 166164  1Ø3362                    BHIS    NXTCHR                  ;GREATER THEN CR, THEN IGNORE.
 166166  ØØ63Ø3                    ASL     SCAN                    ;IF GOOD, MAKE IT WORD INDEX.
 16617Ø  Ø6Ø3Ø7                    ADD     SCAN,PC                 ;AND GO TO THE CORRECT ROUTINE.

 166172  ØØØ426                    BR      BELL                    ;7=BELL
 166174  ØØØ4Ø6                    BR      NORMAL                  ;1Ø=BACKSPACE
 166176  ØØØ411                    BR      TAB                     ;11=TAB
 1662ØØ  ØØØ437                    BR      LF                      ;12=LINE FEED [LF]
 1662Ø2  ØØØ42Ø                    BR      VT                      ;13=VERTICAL TAB [VT]
 1662Ø4  ØØØ424                    BR      FF                      ;14=FORM FEED [FF]
                                                                   ;15=CARRIAGE RETURN [CR]


 1662Ø6  Ø127Ø2  177777    CR:     MOV     #-1,TABCNT              ;RESET TAB POSITION ON A CR, AND
                                                                   ;FALL THROUGH TO INSERT THE CHARACTER.


 166212  ØØ4737  16635Ø    NORMAL: JSR     PC,INSERT               ;INSERT THE CHARACTER IN THE BUFFER.
 166216  ØØ52Ø2                    INC     TABCNT                  ;UPDATE TAB POSITION NOW.
 16622Ø  ØØØ744                    BR      NXTCHR                  ;AND GET NEXT CHARACTER.




 166222  Ø127ØØ  ØØØØ4Ø    TAB:    MOV     #4Ø,CHAR                ;ON A TAB, INSERT BLANKS UNTIL THE
 166226  ØØ4737  16635Ø            JSR     PC,INSERT               ;NEXT CHARACTER POSITION IS A MULTIPLE
 166232  ØØ52Ø2                    INC     TABCNT                  ;OF 8.
 166234  Ø327Ø2  ØØØØØ7            BIT     #7,TABCNT               ;ARE WE DONE YET?
 16624Ø  ØØ137Ø                    BNE     TAB                     ;NOPE.
 166242  ØØØ733                    BR      NXTCHR                  ;YES.


 166244  1117Ø5            VT:     MOVB    (PC),COUNTR             ;THIS PUTS THE LOW BYTE OF THE
                                                                   ;BRANCH CODE IN COUNTR-SAVE A WORD
 166246  ØØØ4Ø5                    BR      FFLOOP
                                                                   ;
 16625Ø  ØØ5Ø37  172ØØ2    BELL:   CLR     GT4ØSR                  ;RING BELL -WRITE IN GT4ØSR
 166254  ØØØ726                    BR      NXTCHR                  ;AND LOOP BACK

 166256  012705  000040          FF:     MOV     #NUMLIN,COUNTR          ;FORM FEED IS DONE BY INSERTING LF'S.

 166262  012700  000012          FFLOOP: MOV     #12,CHAR                ;MAKE THE CHARACTER A LINEFEED.
 166266  004737  166304                  JSR     PC,LFSUB                ;DO A LINEFEED.
 166272  005305                          DEC     COUNTR                  ;DONE?
 166274  003372                          BGT     FFLOOP                  ;NOPE, KEEP SENDING THEM.
 166276  000715                          BR      NXTCHR                  ;YES, NOW RETURN, DO NOT FALL THROUGH.


 166300  012746  166132          LF:     MOV     #NXTCHR,-(SP)           ;RETURN TO NXTCHR AFTER PROCESSING
                                                                         ;THE LF BY FAKING A JSR.

 166304  013703  007012          LFSUB:  MOV     JMPADD,SCAN             ;GET POINTER TO FIRST CHAR ON SCREEN

 166310  122300                  LFLOOP: CMPB    (SCAN)+,CHAR            ;AND LOOK FOR A LINEFEED.
 166312  001406                          BEQ     LFOUND                  ;GOT IT, SEARCH HAS ENDED.
 166314  020327  007000                  CMP     SCAN,#BLIMIT            ;ARE WE AT END OF BUFFER?
 166320  103773                          BLO     LFLOOP                  ;NOPE, KEEP ON LOOKING.
 166322  012703  001000                  MOV     #BSTART,SCAN            ;IF AT TOP, RESET TO BOTTOM OF BUFFER
 166326  000770                          BR      LFLOOP                  ;AND KEEP ON LOOKING.

 166330  005203                  LFOUND: INC     SCAN                    ;WE'VE GOT THE LINE FEED, STOP SHOWING
 166332  042703  000001                  BIC     #1,SCAN                 ;FIRST LINE BY CHANGING THE "DISJMP"
 166336  010337  007012                  MOV     SCAN,JMPADD             ;INSTRUCTION TO FIRST CHAR BEYOND LF.
 166342  004737  166350                  JSR     PC,INSERT               ;INSERT THE LF IN THE BUFFER.
 166346  005000                          CLR     CHAR                    ;AND THEN INSERT ONE NULL CHARACTER BECAUSE
                                                                         ;THE "DISJMP" ADDRESS MUST BE EVEN, AND
                                                                         ;THIS GUARANTEES WE WILL NOT LOSE A
                                                                         ;A GOOD DATA CHARACTER. WE FALL THROUGH
                                                                         ;TO INSERT THE NULL IN THE BUFFER.


 166350  110021                  INSERT: MOVB    CHAR,(POINTR)+          ;STICK IN THE CHARACTER NOW.
 166352  032701  000001                  BIT     #1,POINTR               ;IS NEXT POSITION EVEN OR ODD?
 166356  001021                          BNE     INSRTX                  ;ODD, NO PROBLEMS, SPACE IS ALLOCATED.
 166360  020127  007000                  CMP     POINTR,#BLIMIT          ;EVEN, ARE WE AT THE END OF THE BUFFER?
 166364  103410                          BLO     INSRTL                  ;NO, JUST MAKE ROOM FOR ANOTHER WORD.
 166366  010103                          MOV     POINTR,SCAN             ;AT THE END, MOVE THE STUFF TO THE
 166370  012701  001000                  MOV     #BSTART,POINTR          ;BEGINNING OF THE BUFFER.
 166374  004737  166406                  JSR     PC,INSRTL               ;CALL THE ROUTINE TO SAVE SPACE.
 166400  005023                          CLR     (SCAN)+                 ;AND CLEAR UP THE INSTRUCTIONS AT THE
 166402  005013                          CLR     (SCAN)                  ;END OF THE BUFFER.
 166404  000207                          RTS     PC                      ;AND THEN RETURN.

 166406  022121                  INSRTL: CMP     (POINTR)+,(POINTR)+     ;BYPASS THE "DISJMP" BY ADDING 4 TO POINTR.
 166410  012711  166474                  MOV     #HEADER,(POINTR)        ;NOW INSERT THE DISJMP INSTRUCTION TO OUR HEADER
 166414  012741  160000                  MOV     #DISJMP,-(POINTR)       ;AND IT'S ADDRESS (PUT THEM IN BACKWARDS).
 166420  005041                          CLR     -(POINTR)               ;MAKE AVAILABLE A NEW CHARACTER SPOT.

 166422  000207                  INSRTX: RTS     PC                      ;FINALLY RETURN TO THE CALLER.



```

```
166424  012737  001000  172000  GTBUSE: MOV     #BSTART,GT40PC          ;ON A BUS ERROR, WE MERELY RESTART THE GT40 AT

                                                                        ;THE RTI FOR THIS ROUTINE
                                                                        ;IS THE FIRST WORD OF THE TABLE
                                                                        ;BELOW-IT SAVES A WORD!











                                ;                       INITIALIZATION TABLE FOR THE SCROLLER
                                ;                       -------------- ----- --- --- --------



166432  000002                  SETUP:  .WORD   2                       ;INITIALIZE 2 WORDS,--ALSO RTI FROM ABOVE
166434  000330                          .WORD   330                     ;STARTING AT LOCATION 330
166436  166424                          .WORD   GTBUSE                  ;FIRST WORD IS POINTER TO BUS ERROR ROUTINE,
166440  000200                          .WORD   200                     ;SECOND WORD IS NEW STATUS WORD ON INTERUPT,

166442  000007                          .WORD   7                       ;INITIALIZE THE END OF THE BUFFER TO
166444  006776                          .WORD   BLIMIT-2                ;A CLEAR SPACE TO INSERT THE CHARACTER,
166446  000000                          .WORD   0                       ;THIS IS THE "RUNNING" START, THIS IS
166450  160002  166474                  .WORD   DISJMP,HEADER           ;FOLLOWED BY A DISJMP TO OUR HEADER BLOCK
166454  160000  001000                  .WORD   DISJMP,BSTART           ;AND THEN A DISJMP TO THE START OF THE BUFFER
166460  160000  006700                  .WORD   DISJMP,BLIMIT-NUMLIN-NUMLIN     ;AND A DISJMP TO THE FIRST CHAR ON SCREE

166464  000001                          .WORD   1                       ;FINALLY START THE GT40 GOING AT
166466  172000                          .WORD   GT40PC                  ;THE POSITION INSTRUCTION IN THE
166470  166474                          .WORD   HEADER                  ;HEADER BLOCK,

166472  000000                          .WORD   0                       ;END OF INIT CODE
                                ;
                                ;               HEADER BLOCK FOR THE SCROLLER

166474  103334                  HEADER: .WORD   103334                  ;ENABL CHAR MODE,BLINKING
166476  000177                          .WORD   177                     ;A BLINKING BOX-RUB OUT!
166500  116124                          .WORD   116124                  ;GO TO POINT MODE
166502  171340                          .WORD   171340                  ;LOAD STATUS REGISTER
166504  000000  001352                  .WORD   0,1352                  ;POINT TO UPPER LEFT
166510  103324                          .WORD   103324                  ;BACK TO CHAR MODE
166512  160000  007010                  .WORD   DISJMP,JMPADD-2         ;AND TO THE CHANGING JMP INST,




                                                  .SBTTL  COMMUNICATIONS AND MISC. SUPPORT ROUTINES
```

```



                                      ;                       COMMUNICATIONS HANDLING ROUTINES
                                      ;                       -------------- -------- --------







                                      ;              THE DL-11 HANDLER
                                      ;              --- ----- -------



166516  105737  175610         GETDL:  TSTB     DL11IS                  ;CHECK THE HOST INPUT STATUS.
166522  100011                         BPL      GETDL1                  ;HOST DID NOT SEND ANYTHING, YET.
166524  113700  175612                 MOVB     DL11IB,CHAR             ;HOST SENT US A CHARACTER, PROCESS IT.
166530  012737  000007  175610         MOV      #7,DL11IS               ;REENABLE THE HOST TELECOMMUNICATIONS.
166536  042700  177600                 BIC      #-200,CHAR              ;MAKE CHARACTER JUST SEVEN BITS.
166542  001765                         BEQ      GETDL                   ;IF NULL, IGNORE IT.
166544  000207                         RTS      PC                      ;ELSE RETURN NOW.

166546  105737  177560         GETDL1: TSTB     KBDIS                   ;DID USER TYPE A CHARACTER?
166552  100361                         BPL      GETDL                   ;NO, GO BACK AND CHECK HOST MACHINE.
166554  113737  177562  175616         MOVB     KBDIB,DL11OB            ;MOVE THE CHARACTER TO THE HOST.
166562  000755                         BR       GETDL                   ;AND CHECK AGAIN FOR INPUT.






                                      ;              THE "GET CHARACTER" ROUTINE
                                      ;              --- --- --------- -------



166564  004737  166516         GETCHR: JSR      PC,GETDL                ;GET A CHARACTER FROM THE HOST NOW.
166570  020027  000175                 CMP      CHAR,#ALTMOD            ;IS IT AN "ALTMODE"
166574  001025                         BNE      GETEXT                  ;NO. EXIT NOW.

166576  004737  166516                 JSR      PC,GETDL                ;YES, GET ANOTHER ONE NOW.
166602  020027  000114                 CMP      CHAR,#'L                ;IS IT AN "L"
166606  001501                         BEQ      LOADER                  ;YES, START LOADING NOW.
166610  020027  000122                 CMP      CHAR,#'R                ;IS IT AN "R"
166614  001015                         BNE      GETEXT                  ;NO, IGNORE THE ALTMODE AND JUST RETURN THE CHAR

166616  012737  173000  007010         MOV      #DISTOP,JMPADD-2        ;YES, RESET, STOP DISPLAY BY INSERTING A "DISTOP
166624  000137  166060         PRESTR: JMP      RESTRT                  ;INSTRUCTION IN THE BUFFER, AND RESTART.




```

E-11

```

                                        ;               THE "GET A SIX BIT CHARACTER" ROUTINE
                                        ;               --- --- - --- --- ---------  -------



166630  004737  166564          GETSIX: JSR     PC,GETCHR               ;GET A CHARACTER NOW.
166634  020027  000040                  CMP     CHAR,#40                ;IS IT A LEGAL PRINTING CHARACTER?
166640  002517                          BLT     L.BAD                   ;NOPE, ABORT
166642  020027  000137                  CMP     CHAR,#137               ;IT'S BIG ENOUGH, IS IT TOO BIG?
166646  003114                          BGT     L.BAD                   ;YEP, ABORT.

166650  000207                  GETEXT: RTS     PC                      ;RETURN TO THE CALLER.


                                        ;               THIS OUTPUTS TWO CHARACTERS VIA A
                                        ;               JSR SCAN,OUTLIT
                                        ;               'TWO CHARACTERS'

166652  112337  175616          OUTLIT: MOVB    (SCAN)+,DL11OB
166656  112337  175616                  MOVB    (SCAN)+,DL11OB          ;DOUBLE BUFFERED
166662  000203                          RTS     SCAN                    ;RETURN








                                        ;               THE "GET AN EIGHT BIT CHARACTER" ROUTINE
                                        ;               --- --- -- ----- --- ---------  -------




                                        ;                       THIS ROUTINE DIFFERS FROM THE PREVIOUS ROUTINES
                                        ;               IN THAT IT WILL TAKE SIX BIT CHARACTERS AND ASSEMBLE
                                        ;               THEM FOR THE LOADER TO USE. NOTE THAT FROM THIS POINT
                                        ;               ON WE WILL SWITCH TO THE LOADER DEFINITIONS OF THE
                                        ;               REGISTERS. THUS THE CHARACTER IS RETURNED IN
                                        ;               REGISTER "L.BYT" RATHER THAN CHAR (THOUGH THEY ARE
                                        ;               PHYSICALLY THE SAME).



166664  004737  166630          GET8:   JSR     PC,GETSIX               ;GET A SIXBIT CHARACTER.
166670  010046                          MOV     L.BYT,-(SP)             ;SAVE IT ON THE STACK.
166672  005723                          TST     (INDEX)+                ;UPDATE INDEX TO NEXT ITEM (ALL ARE *2)
166674  000163  166676                  JMP     GET8TB-2(INDEX)         ;AND DISPATCH ACCORDING TO THE INDEX.

166700  000404                  GET8TB: BR      GET81                   ;INDEX=2: ASSEMBLE FIRST CHAR
166702  000416                          BR      GET82                   ;INDEX=4: ASSEMBLE SECOND CHAR
166704  000432                          BR      GET83                   ;INDEX=6: ASSEMBLE THIRD AND LAST CHAR
                                                                        ;INDEX=8: RESET INDEX TO 0 [2] AND RETRY.

```

```
166706  012703  000002          GET84:  MOV     #2,INDEX                ;THE FOURTH INDEX IS THE SAME AS THE FIRST
                                                                        ;INDEX, JUST RESET IT AND FALL THROUGH.


166712  004737  166630          GET81:  JSR     PC,GETSIX               ;GET ANOTHER CHARACTER NOW.
166716  010004                          MOV     L.BYT,HOLD              ;AND PRESERVE IT FOR NEXT TIME THROUGH.
166720  006300                          ASL     L.BYT                   ;NOW THROW AWAY LEFT MOST BITS OF
166722  006300                          ASL     L.BYT                   ;THE 8 BIT CHARACTER, NOW MERGE IN
166724  106300                          ASLB    L.BYT                   ;THE LEFT TWO BITS OF THE
166726  106116                          ROLB    (SP)                    ;NEW SIX BIT CHARACTER WITH THE SIX
166730  106300                          ASLB    L.BYT                   ;BITS FROM THE CHARACTER ON THE
166732  106116                          ROLB    (SP)                    ;STACK. 1ST CHARACTER IS NOW ASSEMBLED.
166734  012600                          MOV     (SP)+,L.BYT             ;SO WE'LL RETURN IT TO THE USER.
166736  000207                          RTS     PC                      ;AND THEN WE SHALL RETURN TO HIM.


166740  006300                  GET82:  ASL     L.BYT                   ;THE SECOND CHARACTER IS CREATED FROM
166742  006300                          ASL     L.BYT                   ;THE 4 RIGHT BITS OF THE PREVIOUS CHARACTER
166744  106300                          ASLB    L.BYT                   ;AND THE FOUR MIDDLE BITS OF THE PRESENT
166746  106104                          ROLB    HOLD                    ;8 BIT CHARACTER.
166750  106300                          ASLB    L.BYT                   ;WE WILL CREATE THE NEW 8 BIT
166752  106104                          ROLB    HOLD                    ;IN THIS REGISTER, SINCE IT
166754  106300                          ASLB    L.BYT                   ;MORE CONVIENT. WE WILL MOVE OVER THE
166756  106104                          ROLB    HOLD                    ;ANSWER AT THE END.
166760  106300                          ASLB    L.BYT                   ;ONE MORE TO GO
166762  106104                          ROLB    HOLD                    ;DONE.
166764  010400                          MOV     HOLD,L.BYT              ;BRING OVER THE VALUE.
166766  012604                          MOV     (SP)+,HOLD              ;AND REMEMBER THE LAST CHARACTER WE RECEIVED.
166770  000207                          RTS     PC                      ;AND RETURN TO THE CALLER.


166772  006100                  GET83:  ROL     L.BYT                   ;FINAL CHARACTER IS EASY. JUST A
166774  106100                          ROLB    L.BYT                   ;SIMPLE MERGER OF LEFT TWO BITS OF
166776  006004                          ROR     HOLD                    ;PREVIOUS VALUE WITH RIGHT SIX BITS
167000  106000                          RORB    L.BYT                   ;OF LAST (4TH) CHARACTER RECEIVED.
167002  006004                          ROR     HOLD
167004  106000                          RORB    L.BYT                   ;AND WE ARE DONE.
167006  005726                          TST     (SP)+                   ;FINALLY THROW AWAY STACK.
167010  000207                          RTS     PC                      ;AND RETURN TO THE CALLER.
```

```
                                                   .SBTTL  THE LOADER
```

```



                                         ;              THE LOADER
                                         ;              --- ------




 167012  012737  173000  007010  LOADER: MOV    #DISTOP,JMPADD-2        ;STOP THE GT40 BY INSERTING A "DISTOP" IN THE LI

 167020  005003                          CLR    INDEX                   ;RESET THE 8 BIT ASSEMBLER TO THE FIRST CHAR


 167022  005005                  L.LD2:  CLR    L.CKSM                  ;CLEAR THE CHECKSUM
 167024  004737  167114                  JSR    PC,L.PTR                ;GET A BYTE NOW.
 167030  105300                          DECB   L.BYT                   ;IS IT A ONE (HEADER)?
 167032  001373                          BNE    L.LD2                   ;NO, WAIT FOR THE ONE.

 167034  004737  167114                  JSR    PC,L.PTR                ;YES, SKIP OVER THE NEXT CHARACTER NOW.

 167040  004737  167126                  JSR    PC,L.GWRD               ;ASSEMBLE A WORD NOW.
 167044  010002                          MOV    L.BYT,L.BC              ;MOVE OVER TO THE COUNTER.
 167046  162702  000004                  SUB    #4,L.BC                 ;REDUCE TO ACTUAL DATA COUNT.
 167052  022702  000002                  CMP    #2,L.BC                 ;ANY DATA AT ALL?
 167056  001433                          BEQ    L.JMP                   ;NO, MUST BE END
 167060  004737  167126                  JSR    PC,L.GWRD               ;YES, ASSEMBLE A DATA WORD NOW.
 167064  010001                          MOV    L.BYT,L.ADR             ;AND THIS MUST BE THE FIRST ADDRESS.


 167066  004737  167114          L.LD3:  JSR    PC,L.PTR                ;GET A BYTE OF DATA NOW.
 167072  002006                          BGE    L.LD4                   ;ALL DONE?
 167074  105705                          TSTB   L.CKSM                  ;YEP, COUNTER IS MINUS, CHECK CHECKSUM.
 167076  001751                          BEQ    L.LD2                   ;CHECKSUM GOOD, GET NEXT COMMAND.


 167100  004337  166652          L.BAD:  JSR    SCAN,OUTLIT      ;BAD LOAD INFORM HOST
 167104     175     102                  .BYTE  ALTMOD,'B               ;SEND ALTMODE B
 167106  000646                          BR     PRESTRT                 ;AND RESTART THE DISPLAY.


 167110  110021                  L.LD4:  MOVB   L.BYT,(L.ADR)+          ;INSERT BYTE INTO MEMORY.
 167112  000765                          BR     L.LD3                   ;AND GET THE NEXT BYTE.



 167114  004737  166664          L.PTR:  JSR    PC,GET8                 ;ASSEMBLE AN 8 BIT CHARACTER NOW.
 167120  060005                          ADD    L.BYT,L.CKSM            ;UPDATE THE CHECKSUM NOW.
 167122  005302                          DEC    L.BC                    ;DECREMENT THE CHARACTER COUNTER.
 167124  000207                          RTS    PC                      ;AND RETURN TO THE CALLER NOW.



 167126  004737  167114          L.GWRD: JSR    PC,L.PTR                ;ASSEMBLE A WORD, FIRST GET A CHARACTER
```

```
167132  010046                         MOV     L.BYT,-(SP)             ;AND SAVE IT.
167134  004737  167114                 JSR     PC,L.PTR                ;AND THEN GET ANOTHER ONE.
167140  000300                         SWAB    L.BYT                   ;AND THEN REASSEMBLE THE MESS.
167142  152600                         BIS     (SP)+,L.BYT             ;WITH THE FEARSOME POWER OF THE 11.
167144  000207                         RTS     PC                      ;AND RETURN TO THE CALLER.




167146  004737  167126         L.JMP:  JSR     PC,L.GWRD               ;ALL DONE WITH THE LOAD. ASSEMBLE
167152  010046                         MOV     L.BYT,-(SP)             ;THE STARTING ADDRESS NOW.
167154  004737  167114                 JSR     PC,L.PTR                ;AND DON'T FORGET TO CHECKSUM IT.
167160  105765                         TSTB    L.CKSM
167162  001346                         BNE     L.BAD                   ;A BAD CHECKSUM. ALL IS EVIL.

167164  004337  166652                 JSR     SCAN,OUTLIT             ;GOOD CHKSUM,INFORM HOST
167170     175     107                 .BYTE   ALTMOD,'G               ;WITH ALTMOD G

167172  032716  000001                 BIT     #1,(SP)                 ;DO WE WANT TO START EXECUTION?
167176  001401                         BEQ     L.JMP1                  ;YES. AWAY WE GO.

167200  000000                 L.HALT: HALT                            ;IF NOT, HALT.

167202  000136                 L.JMP1: JMP     @(SP)+                  ;IF GO, THEN GO ALREADY. WHEEEE!
```

```
                                                           ;THIS IS GT40 QUICK TEST
                                                                   ;GIVES QUICK VISUAL TEST
                                                                   ;OF CONDITION OF MACHINE
                                                                   ;WITHOUT READING IN DIAG.
          100000                  CHAR=100000
          104000                  SHORTV=104000
          110000                  LONGV=110000
          114000                  POINT=114000
          120000                  GRAPHX=120000
          124000                  GRAPHY=124000
          130000                  RELATV=130000

          002000                  INT0=2000
          002200                  INT1=2200
          002400                  INT2=2400
          002600                  INT3=2600
          003000                  INT4=3000
          003200                  INT5=3200
          003400                  INT6=3400
          003600                  INT7=3600                        ;BRIGHTEST

          000100                  LPOFF=100
          000140                  LPON=140
          000020                  BLKOFF=20
          000030                  BLKON=30

          000004                  LINE0=4
          000005                  LINE1=5
          000006                  LINE2=6
          000007                  LINE3=7

          160000                  DJMP=160000
          164000                  DNOP=164000
          170000                  STATSA=170000
          173400                  DSTOP=173400                     ;STOP INTERRUPT

          000300                  LPLITE=300
          000200                  LPDARK=200
          000040                  ITAL0=40                         ;ITALICS OFF
          000060                  ITAL1=60                         ;"       ON
          000004                  SYNON=4                          ;SYNC ON


          174000                  STATSB=174000

          000100                  INCR=100                         ;LOAD GRAPH INCR
          040000                  INTX=40000                       ;INTENSIFY BIT
          001777                  MAXX=1777                        ;BIGGEST X VECTOR
          001377                  MAXY=1377                        ;BIGGEST Y VECTOR
          020000                  MINUSX=20000                     ;THE MINUS BIT
          020000                  MINUSY=MINUSX
          017600                  MAXSX=17600                      ;BIGGEST X IN SHORTVEC
          000077                  MAXSY=77                         ;   "    Y IN  "
          000100                  MINSUY=100                       ;MINUS BIT FOR Y IN SHORTVEC
```

```


167204  012737  167214  172000            MOV     #FILE0,GT40PC          ;START THE GT40
167212  000001                            WAIT                           ;AND WAIT

167214  114020                   FILE0:   POINT!BLKOFF                   ;POINT--INVISIBLE
167216  000000                            0
167220  001377                            MAXY

167222  112004                            LONGV!INT0!LINE0               ;DRAW TOP LINE
167224  041777                            INTX!MAXX
167226  000000                            0

167230  112405                            LONGV!INT2!LINE1
167232  040000                            INTX                           ;DRAW LINE TO RIGHT
167234  021377                            MINUSX!MAXY

167236  113006                            LONGV!INT4!LINE2
167240  061777                            INTX!MINUSX!MAXX               ;DRAW BOTTOM LINE
167242  000000                            0

167244  113407                            LONGV!INT6!LINE3
167246  040000                            INTX
167250  001377                            MAXY                           ;DRAW LINE TO LEFT

167252  114000                            POINT
167254  000400                            400
167256  000500                            500
167260  106200                            SHORTV!INT1
167262  057677                            57677                          ;+X+Y
167264  106600                            SHORTV!INT3
167266  077677                            77677                          ;+X-Y
167270  107200                            SHORTV!INT5
167272  077777                            77777                          ;-X-Y
167274  107600                            SHORTV!INT7
167276  057777                            57777                          ;-X+Y

167300  114000                            POINT
167302  001400                            1400
167304  000500                            500
167306  133030                            RELATV!INT4!BLKON
167310  057677                            57677                          ;+X+Y
167312  077677                            77677                          ;+X-Y
167314  077777                            77777                          ;-X-Y
167316  057777                            57777                          ;-X+Y

167320  114000                            POINT
167322  000400                            400
167324  000100                            100
167326  174120                            STATSB!INCR+20                 ;TRY GRAPH MODES
167330  114000                            POINT
167332  001000                            1000
167334  000200                            200

```

```
167336  120000                              GRAPHX
167340  001010                              1010
167342  001020                              1020
167344  001030                              1030
167346  001040                              1040
167350  001050                              1050

167352  114000                              POINT
167354  001000                              1000
167356  001200                              1200

167360  124000                              GRAPHY
167362  001020                              1020
167364  001030                              1030
167366  001040                              1040
167370  001050                              1050
167372  001060                              1060

167374  160000                              DJMP
167376  167214                              FILE0
```

```
                                       ;
                                       ; PAPER TAPE BOOT
                                       ;
           177550                      HSR=177550                          ;HIGH SPEED READER ADDRESS
           177560                      LSR=177560                          ;LOW SPEED READER ADDRESS
           167400                      .=ORIGIN+1400
                                       ;
   167400  012701  160000              PTBOOT: MOV     #160000,R1          ;SET MEMORY CHECK LIMITS
   167404  012702  000004                      MOV     #4,R2               ;TRAP ADDRESS IS LOC. 4
   167410  012703  167500                      MOV     #DEV+4,R3           ;POINTER TO DEVICE ADDRESSES
   167414  010712                              MOV     PC,@R2              ;PRESET TRAP ADDRESS IN LOC. 4
   167416  012706  000024                      MOV     #24,SP              ;STACK SET UP AT SPECIAL ADDRESS
   167422  014304                      DEV1:   MOV     -(R3),R4            ;GET DEVICE ADDRESS
   167424  005714                              TST     @R4                 ;CHECK AVAILABILITY OF DEVICE
   167426  100775                              BMI     DEV1                ;CHECK DEVICE FOR ERRORS
   167430  010712                              MOV     PC,@R2              ;RESET TRAP ADDRESS AT LOC. 4
   167432  012706  000024                      MOV     #24,SP              ;SPECIAL ADDRESS USED AS MASK LATER
   167436  010441                              MOV     R4,-(R1)            ;DO MEM CHK:READER STATUS ADDRESS
                                                                           ;IS MOVED
   167440  040601                              BIC     SP,R1               ;SET R1=X7752,MASK IN SP=24
   167442  010111                              MOV     R1,@R1              ;STORE OWN ADDRESS IN POINTER
   167444  011102                      LOOP:   MOV     @R1,R2              ;GET BYTE POINTER
   167446  005214                              INC     @R4                 ;ENABLE READER
   167450  105714                              TSTB    @R4                 ;TEST DONE BIT
   167452  100376                              BPL     .-2                 ;WAIT UNTIL READY
   167454  116412  000002                      MOVB    2(R4),@R2           ;THEN PICK IT UP AND STORE IT
   167460  005211                              INC     @R1                 ;BUMP POINTER
   167462  120227  000375                      CMPB    R2,#375             ;STORED JUMP OFFSET?
   167466  001366                              BNE     LOOP                ;NOT YET
   167470  105222                              INCB    (R2)+               ;YES,ALL DONE
   167472  000142                              JMP     -(R2)               ;GO EXECUTE AS BRANCH
                                       ;
                                       ; DEVICE ADDRESSES FOLLOW - DO NOT CHANGE THE ORDER
                                       ;
   167474  177560                      DEV:    LSR                         ;LOW SPEED READER
   167476  177550                              HSR                         ;HIGH SPEED READER
```

```
                                   ;
                                   ; CASSETTE BOOT
                                   ;
        177500                     TACS=177500                         ;TA-11 CONTROL AND STATUS REGISTER
        167500                     .=ORIGIN+1500
167500  012700  177500             TABOOT: MOV     #TACS,RØ
167504  005010                             CLR     (RØ)                ;SELECT UNIT #Ø
167506  010701                     RES:    MOV     PC,R1               ;USE FOR PIC
167510  062701  000052                     ADD     #TABLE-.,R1         ;R1 HOLDS ADDR. OF COMMAND TABLE
167514  012702  000375                     MOV     #375,R2             ;MEMORY PTR. AND DATA FLAG
167520  112103                             MOVB    (R1)+,R3            ;TEST BITS
                                   ;
167522  112110                     LOOP1:  MOVB    (R1)+,(RØ)          ;COMMAND FROM TABLE TO TACS
167524  100413                             BMI     DONE                ;WHEN COMMAND CODE NEG., QUIT
167526  130310                     LOOP2:  BITB    R3,(RØ)             ;TEST READY AND T-REQ BITS IN TACS
167530  001776                             BEQ     LOOP2               ;LOOP 'TIL SOMETHING COMES UP
167532  105202                             INCB    R2                  ;ADVANCE MEMORY POINTER
167534  100772                             BMI     LOOP1               ;IF MINUS, TRY NEXT COMMAND
167536  116012  000002                     MOVB    2(RØ),(R2)          ;READ DATA INTO MEMORY
167542  120337  000000                     CMPB    R3,@#Ø              ;FIRST BYTE READ SHOULD BE '24Ø'
167546  001767                             BEQ     LOOP2               ;IF O.K., GO READ ANOTHER BYTE
167550  000000                     STOP:   HALT                        ;HALT ON ERROR
167552  000755                             BR      RES                 ;RESTART ON CONTINUE
                                   ;
167554  005710                     DONE:   TST     (RØ)                ;CHECK FOR ERROR
167556  100774                             BMI     STOP                ;HALT ON ERROR
167560  005007                             CLR     PC                  ;= 'JMP @#Ø'
                                   ;
167562  017640                     TABLE:  .WORD   1764Ø               ;.BYTE 24Ø: READY+T-REQ.
                                                                       ;.BYTE 37:  ILBS+READY+GO
167564  002415                             .WORD   2415                ;.BYTE  15: SFB+GO
                                                                       ;.BYTE   5: READ+GO
167566  112024                             .WORD   112024              ;.BYTE  24: READ+ILBS
                                                                       ;.BYTE 224: READ+ILBS+E.O.TABLE
167570  000000  000000                     .WORD   Ø,Ø                 ;THESE ARE FILLER WORDS
167574  167500                             .WORD   TABOOT              ;POWER UP VECTOR AND PRIORITY
167576  000340                             .WORD   34Ø                 ;
```

```
                                ;MR11-DB BULK STORAGE PROGRAM LOADER LISTING

        167600                  .=ORIGIN+1600                   ;KEEP TRACK OF ORIGIN

167600  010702                  RF11:   MOV PC,R2               ;FIXED HEAD DISK (256 KW)
167602  000451                          BR OTHER
167604  177462                          177462
167606  000005                          5

167610  010702                  RK11:   MOV PC,R2               ;MOVING HEAD DISK (CARTRIDGE)
167612  000445                          BR OTHER
167614  177406                          177406
167616  000005                          5


167620  010702                  TC11:   MOV PC,R2
167622  000417                          BR TAPES
167624  177344                          177344                  ;ADDRESS OF WORD COUNT
167626  000005                          5                       ;LAST COMMAND
167630  004003                          4003                    ;FIRST COMMAND
167632  100000                          100000                  ;DONE MASK
167634  024000                          24000                   ;ERROR MASK


167636  010702                  TM11:   MOV PC,R2
167640  000410                          BR TAPES
167642  172524                          172524                  ;ADDRESS OF BYTE COUNT
167644  060003                          60003                   ;LAST COMMAND
167646  060011                          60011                   ;FIRST COMMAND
167650  000200                          200                     ;DONE MASK
167652  100000                          100000                  ;ERROR MASK


167654  010702                  RP11:   MOV PC,R2               ;MOVING HEAD DISK (PACK)
167656  000423                          BR OTHER
167660  176716                          176716


167662  000005                  TAPES:  RESET
167664  010200                          MOV R2,R0               ;GET THE ADDRESS OF THE BRANCH
167666  005720                          TST (0)+                ;R0 TO POINT AT LAST COMMAND
167670  012001                          MOV (0)+,R1             ;GET THE WORD COUNT ADDRESS
167672  005311                          DEC (1)                 ;SET UP FOR ADVANCE 1 RECORD
167674  005720                          TST (0)+                ;MOVE R0 TO FIRST COMMAND
167676  012041                          MOV (0)+,-(1)           ;COMMAND WORD TO COMMAND REG.
167700  031011                          BIT (0),(1)             ;LOOK FOR DONE INDICATORS
167702  001776                          BEQ .-2                 ;NONE SET, TRY AGAIN
167704  005720                          TST (0)+                ;DONE FIRST COMMAND, CHECK FOR ERROR
167706  031041                          BIT (0),-(1)            ;LOOK FOR SET ERROR BITS
167710  001406                          BEQ OTHER               ;NO ERRORS - TRY THE READ
167712  000112                AGAIN:    JMP (2)                 ;RERUN FOR ERRORS


167714  167600                  RFVEC:  RF11                    ;RF11 POWER UP VECTOR
```

```
167716  ØØØ34Ø                               34Ø

167720  Ø107Ø2                       RC11:   MOV PC,R2              ;FIXED HEAD DISK (64KW)
167722  ØØØ4Ø1                               BR OTHER
167724  17745Ø                               17745Ø                 ;ADRS OF WORD COUNT (COMMAND+2)
                                                                    ;COMMAND WORD (5) IS THE RESET

167726  ØØØØØ5                       OTHER:  RESET
16773Ø  Ø102ØØ                               MOV R2,RØ              ;RØ TO POINT AT WORD COUNT ADRS
167732  ØØ572Ø                               TST (Ø)+               ;POINT TO ADDRESS
167734  Ø12ØØ1                               MOV (Ø)+,R1            ;WORD COUNT ADDRESS TO R1
167736  Ø12711  177ØØØ                       MOV #-1ØØØ,(1)         ;LOAD WORD COUNT
167742  Ø11Ø41                               MOV (Ø),-(1)           ;COMMAND TO COMMAND REGISTER
167744  Ø32711  1ØØ2ØØ                       BIT #1ØØ2ØØ,(1)        ;CHECK FOR ERROR OR DONE
16775Ø  ØØ1775                               BEQ .-4                ;IF NEITHER, KEEP LOOKING
167752  1ØØ757                               BMI AGAIN              ;ERROR, TRY AGAIN
167754  ØØ5ØØ7                               CLR PC

167756  ØØØØØØ                               Ø                      ;FILLER
16776Ø  16761Ø                       RKVEC:  RK11                   ;RK POWER UP VECTOR
167762  ØØØ34Ø                               34Ø
167764  16772Ø                       RCVEC:  RC11                   ;RC POWER UP VECTOR
167766  ØØØ34Ø                               34Ø
16777Ø  167654                       RPVEC:  RP11                   ;RP POWER UP VECTOR
167772  ØØØ34Ø                               34Ø
167774  16762Ø                       TCVEC:  TC11                   ;TC11 POWER UP VECTOR
167776  ØØØ34Ø                               34Ø

```

```
  1008          000001                          .END
```

```
SCROLLING ROM BOOTSTRAP FOR THE GT4Ø    MACY11,624  16-JUL-73  1Ø:Ø4  PAGE 6-1
BOOT,T16          CROSS REFERENCE TABLE

AGAIN    167712           974#    993
ALTMOD = ØØØ175           139#    465     633     666
BELL     16625Ø           278     31Ø#
BLIMIT = ØØ7ØØØ           132#    135     223     244     329     348     395     399
BLKOFF = ØØØØ2Ø           713#    749
BLKON  = ØØØØ3Ø           714#    784
BSTART = ØØ1ØØØ           131#    331     351     368     398
CHAR   = 1ØØ7ØØ            82#    1Ø2     267     269     271     298*    315*    327     338*    345     444*    446*    465
                          469     471     487     489     694#
CORSTR = ØØØ2Ø4           134#    192
COUNTR =%ØØØØØ5            89#    1Ø5     3Ø6*    313*    317*
CR       166266           267#
CRLF   = ØØ5215           138#    226
DEV      167474           857     882#
DEV1     167422           86Ø#    862
DISJMP = 16ØØØØ           141#    359     397     398     399     415
DISTOP = 173ØØØ           142#    474     6Ø5
DJMP   = 16ØØØØ           721#    816
DL11IB = 175612           12Ø#    121     444
DL11IS = 17561Ø           119#    12Ø     186*    442     445*
DL11OB = 175616           122#    452*    499*    52Ø*
DL11OS = 175614           121#    122     188*    226     228*
DNOP   = 164ØØØ           722#
DONE     167554           899     91Ø#
DSTOP  = 1734ØØ           724#
ENDCOR   166136           195#    196
FF       166256           283     313#
FFLOOP   166262           3Ø8     315#    318
FILEØ    167214           746     749#    817
GETCHR   166564           266     464#    486
GETDL    166516           442#    447     451     453     464     468
GETDL1   166546           443     45Ø#
GETEXT   16665Ø           466     472     492#
GETSIX   16663Ø           486#    524     539
GET8     166664           524#    642
GET8TB   1667ØØ           527     529#
GET81    166712           529     539#
GET82    16674Ø           53Ø     551#
GET83    166772           531     566#
GET84    1667Ø6           535#
GRAPHX = 12ØØØØ           698#    798
GRAPHY = 124ØØØ           699#    8Ø9
GTBUSE   166424           368#    391
GT4ØPC = 172ØØØ           127#    128     368*    4Ø2     746*
GT4ØSR = 172ØØ2           128#    31Ø*
HEADER   166474           358     397     4Ø3     4Ø9#
HOLD   =%ØØØØØ4            87#    54Ø*    554*    556*    558*    56Ø*    561     562*    568*    57Ø*
HSR    = 17755Ø           851#    883
INCR   = ØØØ1ØØ           735#    793
INDEX  =%ØØØØØ3           1Ø6#    526     527     535*    6Ø7*
INSERT   16635Ø           291     299     337     345#
INSRTL   1664Ø6           349     352     357#
INSRTX   166422           347     362#
INTX   = Ø4ØØØØ           736#    754     758     762     766
```

E-26

```
SCROLLING RCM BOOTSTRAP FOR THE GT40    MACY11,624  16-JUL-73  10:04  PAGE 6-2
BOOT,T16          CROSS REFERENCE TABLE

INTØ    = ØØ2ØØØ            7Ø2#     753
INT1    = ØØ22ØØ            7Ø3#     772
INT2    = ØØ24ØØ            7Ø4#     757
INT3    = ØØ26ØØ            7Ø5#     774
INT4    = ØØ3ØØØ            7Ø6#     761      784
INT5    = ØØ32ØØ            7Ø7#     776
INT6    = ØØ34ØØ            7Ø8#     765
INT7    = ØØ36ØØ            7Ø9#     778
ITALØ   = ØØØØ4Ø            728#
ITAL1   = ØØØØ6Ø            729#
JMPADD  = ØØ7Ø12            135#     325      336*     415      474*     6Ø5*
KBDIB   = 177562            125#     452
KBDIS   = 17756Ø            124#     125      45Ø
LF        1663ØØ            281      322#
LFLOOP    16631Ø            327#     33Ø      332
LFOUND    16633Ø            328      334#
LFSUB     1663Ø4            316      325#
LINEØ   = ØØØØØ4            716#     753
LINE1   = ØØØØØ5            717#     757
LINE2   = ØØØØØ6            718#     761
LINE3   = ØØØØØ7            719#     765
LOADER    167Ø12            47Ø      6Ø5#
LONGV   = 11ØØØØ            696#     753      757      761      765
LOOP      167444            869#     876
LOOP1     167522            898#     9Ø3
LOOP2     167526            9ØØ#     9Ø1      9Ø6
LPDARK  = ØØØ2ØØ            727#
LPLITE  = ØØØ3ØØ            726#
LPOFF   = ØØØ1ØØ            711#
LPON    = ØØØ14Ø            712#
LSR     = 17756Ø            852#     882
L,ADR   =%ØØØØØ1            1Ø3#     623*     637*
L,BAD     1671ØØ            488      49Ø      632#     663
L,BC    =%ØØØØØ2            1Ø4#     618*     619*     62Ø      644*
L,BYT   =%ØØØØØØ            1Ø2#     525      54Ø      541*     542*     543*     545*     547*     551*     552*     553*     555*     557*
                            559*     561*     566*     567*     569*     571*     612*     618      623      637      643      65Ø      652*
                            653*     66Ø
L,CKSM  =%ØØØØØ5            1Ø5#     61Ø*     628      643*     462
L,GWRD    167126            617      622      649#     659
L,HALT    1672ØØ            671#
L,JMP     167146            621      659#
L,JMP1    1672Ø2            669      673#
L,LD2     167Ø22            61Ø#     613      629
L,LD3     167Ø66            626#     638
L,LD4     16711Ø            627      637#
L,PTR     167114            611      615      626      642#     649      651      661
MAXSX   = Ø176ØØ            741#
MAXSY   = ØØØ377            742#
MAXX    = ØØ1777            737#     754      762
MAXY    = ØØ1377            738#     751      759      767
MINSUY  = ØØØ1ØØ            743#
MINUSX  = Ø2ØØØØ            739#     74Ø      759      762
MINUSY  = Ø2ØØØØ            74Ø#
NORMAL    166212            272      279      291#
```

```
SCROLLING ROM BOOTSTRAP FOR THE GT4Ø    MACY11,624  16-JUL-73  1Ø:Ø4  PAGE 6-3
BOOT.T16         CROSS REFERENCE TABLE

NOTHER   166Ø42        193     199#
NUMLIN = ØØØØ4Ø        136#    223     224     313     399
NXTCHR   166132        266#    268     274     293     3Ø3     311     319     322
ORIGIN = 166ØØØ        117#    173     853     89Ø     926
OTHER    167726        929     934     958     973     981     985#
OUTLIT   166652        189     499#    632     665
PC      =%ØØØØØ7        74#    266*    276*    291*    299*    3Ø6     316*    337*    352*    355*    362*    448*    464*
                       468*    486*    492*    524*    539*    548*    563*    573*    611*    615*    617*    622*    626*
                       642*    645*    649*    651*    654*    659*    661*    858     863     893     912*    928     933
                       939     948     957     98Ø     994*
POINT  = 114ØØØ        697#    749     769     781     79Ø     794     8Ø5
POINTR =%ØØØØØ1         63#    1Ø3     236*    238*    244*    345*    346     348     35Ø     351*    357     358*    359*
                       362*
PRESTR   166624        475#    634
PTBOOT   1674ØØ        855#
RCVEC    167764        999#
RC11     16772Ø        98Ø#    999
RELATV = 13ØØØØ        7ØØ#    784
RES      1675Ø6        893#    9Ø8
RESTRT   166Ø6Ø        22Ø#    475
RFVEC    167714        977#
RF11     1676ØØ        928#    977
RKVEC    16776Ø        997#
RK11     16761Ø        933#    997
RPVEC    16777Ø       1ØØ1#
RP11     167654        957#   1ØØ1
RØ      =%ØØØØØØ        67#     82     891*    892*    818*    9ØØ     9Ø4     91Ø     963*    986*
R1      =%ØØØØØ1        68#     83     855*    865*    817*    868*    869     874*    893*    894*    896     898     965*
                       988*
R2      =%ØØØØØ2        69#     84     856*    858*    813*    869*    873*    875     877*    878     895*    9Ø2*    9Ø4*
                       928*    933*    939*    948*    917*    963     98Ø*    986
R3      =%ØØØØØ3        7Ø#     85     857*    86Ø     8Ø6*    9ØØ     9Ø5
R4      =%ØØØØØ4        71#     87     86Ø*    861     865     87Ø*    871     873
R5      =%ØØØØØ5        72#     89
SCAN    =%ØØØØØ3        85#    1Ø6     189*    192*    193*    195*    199     2Ø4     223*    226*    231*    234     236
                       238     271*    272*    273     275*    276     325*    327     329     331*    334*    335*    336
                       35Ø*    353*    354*    499     51Ø     5Ø1*    632*    665*
SETDUN   166126        235     244#
SETLP1   166Ø74        226#    228
SETLP2   16611Ø        234#    241
SETLP3   166116        238#    24Ø
SETUP    166432        231     389#
SHORTV = 124ØØØ        695#    772     774     776     778
SP      =%ØØØØØ6        73#    187*    2Ø4*    22Ø*    122*    525*    544*    546*    547     562     572     65Ø*    653
                       66Ø*    668     673     859*    164*    867
START    166ØØØ        185#
STATSA = 17ØØØØ        723#
STATSB = 174ØØØ        733#    793
STOP     16755Ø        9Ø7#    911
SYNON  = ØØØØØ4        73Ø#
TAB      166222        28Ø     298#    3Ø2
TABCNT =%ØØØØØ2         84#    1Ø4     224*    227*    234*    239*    287*    292*    3ØØ*    3Ø1
TABLE    167562        894     914#
TABOOT   1675ØØ        891#    921
```

```
SCROLLING ROM BOOTSTRAP FOR THE GT4Ø    MACY11,624  16-JUL-73  1Ø:Ø4  PAGE 6-4
BOOT.T16        CROSS REFERENCE TABLE

TACS    = 1775ØØ            889#     891
TAPES     167662            94Ø      949      962#
TCVEC     167774           1ØØ3#
TC11      16762Ø            939#    1ØØ3
TMPEND  = ØØ7776            133#     187      22Ø
TM11      167636            948#
VT        166244            232      3Ø6#
.       = 172ØØØ            173#     853#     872      89Ø#     894      926#     97Ø      992
```

```
SCROLLING ROM BOOTSTRAP FOR THE GT4Ø    MACY11,624  16-JUL-73  1Ø:Ø4  PAGE 6-5
BOOT,T16          CROSS REFERENCE TABLE

ADD        276      643      894
ASL        275      541      542      551      552
ASLB       543      545      553      555      557      559
BEQ        235      328      447      47Ø      621      629      !69      9Ø1      9Ø6      97Ø      973      992
BGE        268      27Ø      627
BGT        226      24Ø      318      49Ø
BHIS       274
BIC        335      446      867
BIS        22Ø      653
BIT        3Ø1      346      668      969      972      991
BITB       9ØØ
BLO        33Ø      349
BLT        468
BMI        862      899      9Ø3      911      993
BNE        3Ø2      347      466      472      613      663      !76
BPL        2Ø7      443      451      872
BR         196      241      278      279      28Ø      281      !82      283      293      3Ø3      3Ø8      311      319      332      453
           529      53Ø      531      634      638      9Ø8      929      934      94Ø      949      958      981
CLR        195      2Ø6      319      338      353      354      36Ø      6Ø7      61Ø      892      912      994
CMP        267      269      273      329      348      357      465      469      471      487      489      62Ø
CMPB       327      875      9Ø5
DEC        227      239      317      644      966
DECB       612
HALT       671      927
INC        188      292      3ØØ      334      87Ø      874
INCB       877      9Ø2
JMP        475      527      673      878      974
JSR        189      266      291      299      316      337      !52      464      468      486      524      539      611      615      617
           622      626      632      642      649      651      !59      661      665
MOV        186      187      192      193      2Ø4      223      !24      226      231      234      236      238      244      271      287
           298      313      315      322      325      331      !36      35Ø      351      358      359      368      445      474      525
           535      54Ø      547      561      562      6Ø5      !18      623      65Ø      66Ø      746      855      856      857      858
           859      86Ø      863      864      865      868      !69      891      893      895      928      933      939      948      957
           963      965      968      98Ø      986      988      !89      99Ø
MOVB       326      345      444      452      499      5ØØ      !37      873      896      898      9Ø4
RESET      185      962      985
ROL        566
ROLB       544      546      554      556      558      56Ø      !67
ROR        568      57Ø
RORB       569      571
RTS        355      362      448      492      5Ø1      548      !63      573      645      654
SUB        272      619
SWAB       652
TST        199      526      572      861      91Ø      964      967      971      987
TSTB       2Ø6      442      45Ø      628      662      871
WAIT       747
.BYTE      633      665
.ENABL      36
.END      1ØØ7
.PAGE       55      1Ø9      162      257      425      595      !89      82Ø
.SBTTL      54      161      256      424      594
.TITLE       2
.WORD      19Ø      389      39Ø      391      392      394      395      396      397      398      399      4Ø1      4Ø2      4Ø3      4Ø5
           4Ø9      41Ø      411      412      413      414      415      914      916      918      92Ø      921      922
```

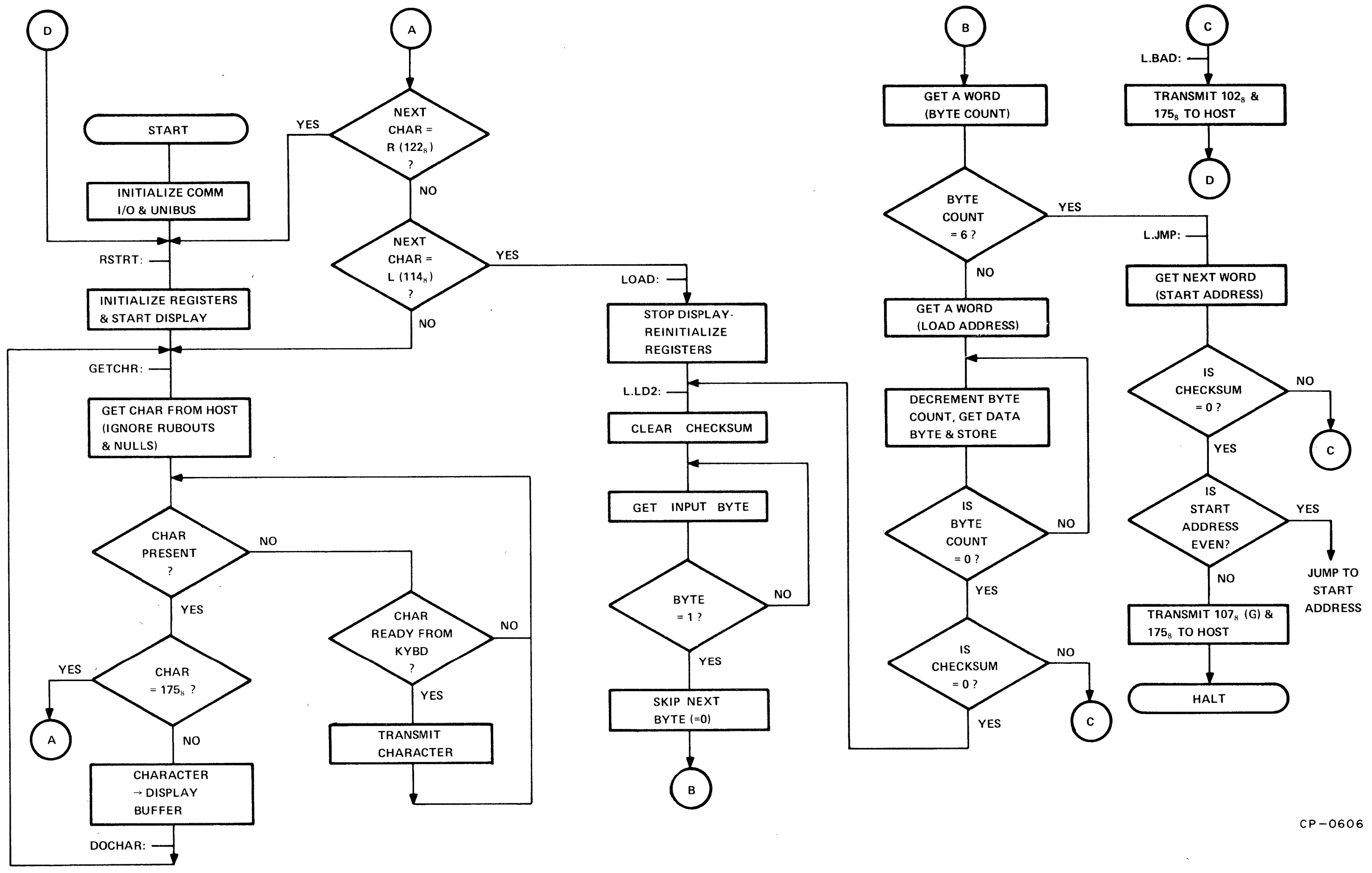

D
A
B
C
START
INITIALIZE COMM I/O & UNIBUS
RSTRT:
INITIALIZE REGISTERS & START DISPLAY
GETCHR:
GET CHAR FROM HOST (IGNORE RUBOUTS & NULLS)
CHAR PRESENT ?
YES
NO
CHAR = 175_8 ?
YES
NO
A
CHARACTER → DISPLAY BUFFER
DOCHAR:
CHAR READY FROM KYBD ?
NO
YES
TRANSMIT CHARACTER
NEXT CHAR = R (122_8) ?
YES
NO
NEXT CHAR = L (114_8) ?
YES
NO
LOAD:
STOP DISPLAY-REINITIALIZE REGISTERS
L.LD2:
CLEAR CHECKSUM
GET INPUT BYTE
BYTE = 1 ?
NO
YES
SKIP NEXT BYTE (=0)
B
GET A WORD (BYTE COUNT)
BYTE COUNT = 6 ?
YES
NO
GET A WORD (LOAD ADDRESS)
DECREMENT BYTE COUNT, GET DATA BYTE & STORE
IS BYTE COUNT = 0 ?
NO
YES
IS CHECKSUM = 0 ?
NO
YES
C
L.BAD:
TRANSMIT 102_8 & 175_8 TO HOST
D
L.JMP:
GET NEXT WORD (START ADDRESS)
IS CHECKSUM = 0 ?
NO
YES
C
IS START ADDRESS EVEN?
YES
NO
JUMP TO START ADDRESS
TRANSMIT 107_8 (G) & 175_8 TO HOST
HALT
CP-0606

Figure E-1 Communications Bootstrap Loader Flow Diagram

Reader's Comments

GT40/42 USER'S GUIDE
EK-GT40-OP-002

Your comments and suggestions will help us in our continuous effort to improve the quality and usefulness of our publications.

What is your general reaction to this manual? In your judgment is it complete, accurate, well organized, well written, etc.? Is it easy to use? ____________________

What features are most useful? ____________________

What faults do you find with the manual? ____________________

Does this manual satisfy the need you think it was intended to satisfy? ____________________

Does it satisfy *your* needs? __________ Why? ____________________

Would you please indicate any factual errors you have found. ____________________

Please describe your position. ____________________

Name ____________________ Organization ____________________

Street ____________________ Department ____________________

City __________ State __________ Zip or Country __________

CUT OUT ON DOTTED LINE

Fold Here

Do Not Tear - Fold Here and Staple

FIRST CLASS
PERMIT NO. 33
MAYNARD, MASS.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

Postage will be paid by:

Digital Equipment Corporation
Technical Documentation Department
146 Main Street
Maynard, Massachusetts 01754