EK-OTU58-UG-003

# TU58 DECtape II

## User Guide

digital

EK-0TU58-UG-003

# TU58 DECtape II

## User Guide

Prepared by Educational Services
of
Digital Equipment Corporation

1st Edition, October 1978
2nd Edition, June 1981
3rd Edition, October 1982



Printed in U.S.A.



10/82-14

# CONTENTS

## CHAPTER 3 PROGRAMMING



## CHAPTER 4 INSTALLATION

## CHAPTER 5 OPTIONS



## APPENDIX A TU58/PDP-11 TOGGLE-IN BOOT

## APPENDIX B RSP SEQUENCE

## APPENDIX C SAMPLE DEVICE HANDLERS

## APPENDIX D CARTRIDGE REPAIR



## APPENDIX E FIELD REPLACEABLE UNIT SPARES LIST

## FIGURES

**TABLES**

# CHAPTER 1
# INTRODUCTION

## 1.1 SCOPE

The TU58 DECtape II is a low-cost, mass-storage device that may be used in a wide variety of applications. This manual provides information that a user needs to install, interface, and operate the tape system. (For specific information about using the TU58 under DIGITAL operating systems, refer to the individual system manuals.)

Chapter 1 provides a general description of the TU58 and a list of its specifications, including electrical and mechanical requirements. The configurations section describes the available variations of the TU58.

Chapter 2 contains important information for daily operation and routine maintenance. It is the system operator's reference section.

Chapter 3 is a programming guide. It contains functional descriptions of the TU58 command set, illustrates command sequences, explains the details of the radial serial protocol (RSP) and the modified radial serial protocol (MRSP), lists system instruction codes and byte sequences, and includes a general purpose programming example for a TU58 device handler.

Chapter 4 describes instructions for jumper selection; mechanical, electrical, and interface installation; and operational checkout of the tape system.

Chapter 5 describes the optional features available in the TU58.

Appendix A lists a PDP-11 toggle-in bootstrap for the TU58.

Appendix B contains an RSP sequence to exercise a new cartridge.

Appendix C lists sample device handlers written in PDP-11 FORTRAN IV and PDP-11 MACRO-11 assembly language.

Appendix D covers cartridge repair procedures.

Appendix E lists the field replaceable units (FRUs) in the TU58.

## 1.2 GENERAL DESCRIPTION

The TU58 is a random-access, fixed-length-block, mass-storage tape system. It uses preformatted tape cartridges which store 262 kilobytes of data in 512-byte blocks. There are 256 blocks on each of two tracks. They may be accessed by a program in a fashion similar to that employed for data stored on disks or DECtape, using a new, high-level instruction set. A file-oriented structure is easily implemented in an operating system by setting aside several blocks on the tape to store a directory.

The TU58 is compact and mechanically simple. The tape cartridges are DIGITAL-preformatted, miniature, reel-to-reel packages containing 42.7 m (140 ft) of 3.81 mm (0.150 in) wide tape. A single puck drives the tape by engaging a roller which moves an elastomer drive belt in the cartridge. This belt loops around both tape spools and provides uniform tension and spill-free winding without mechanical linkages (Figure 1-1). The simple, single-point drive mechanism provides high reliability for the entire system.

The control and drive circuitry of the TU58 is located on a single circuit board. The controller uses a microprocessor ($\mu$P) to reduce the tape handling and communications management load on the host system.

The motor and tape head control, driver, and switching circuits that manage the two tape drives are on the printed circuit board with the $\mu$P. The controller supports one or two drives, but only one drive can operate at a time. The $\mu$P controls all activities of the TU58. Head and motor selection, speed and direction changes, etc. are managed by outputs from I/O ports on a peripheral integrated circuit (IC). The mechanical actions of the drives are supervised by the $\mu$P in order to improve system performance.

Operational amplifiers, comparators, and logic circuits perform amplification, signal switching and conditioning, proportional control, and logic steering functions in the controller. The tape is protected by motor current limiting and an anti-runaway timer.

The $\mu$P intelligence requires that requests from the host for data retrieval or storage contain only simple specifications about the transfer. The controller positions the tape and performs the transfer without supervision from the host.

The host and controller communicate in a format called either radial serial protocol (RSP), or modified radial serial protocol (MRSP). RSP uses two kinds of byte sequences called message packets. Both command and data packets have protocol information placed in specific locations in the byte sequence. This format is easily generated by the TU58, making host-peripheral interaction possible at a high level with low cost. Figure 1-2 illustrates a typical RSP exchange between a host computer and the TU58. See Chapter 3 for a full discussion of RSP implementation.

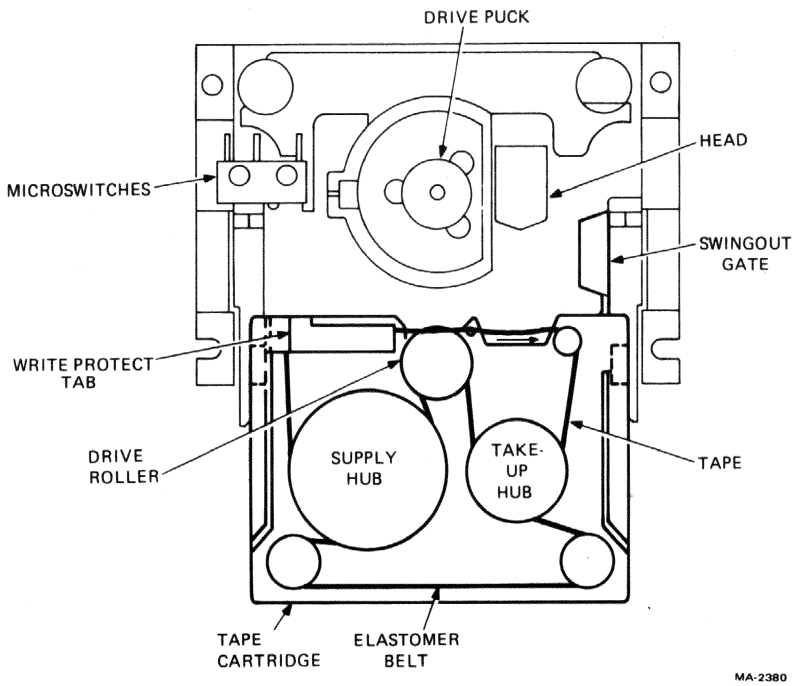


Figure 1-1 Tape Cartridge Partially Inserted into Drive (Top View)

When, owing to the data transfer rate selected, the buffer is unable to accept an entire transaction, modified serial protocol (MRSP) is utilized. MRSP is implemented by using the command packet switch byte. See Paragraph 3.2 for a more detailed description of MRSP implementation.

The serial host interface operates on full-duplex, asynchronous, 4-wire lines at jumper-selectable rates of 150 to 38.4K baud. Send and receive rates may be independently set with jumpers to operate in accordance with Electronic Industries Association (EIA) standards RS-422 or RS-423. When set to RS-423, the TU58 is also compatible with devices complying with RS-232-C.

## 1.3 BLOCK DIAGRAM

Figure 1-3 illustrates the structure of the TU58 system. The data path is along the top of the diagram, passing to the host through the processor at the right. The drive control is at the lower left, also closely associated with the processor through the I/O ports. The ports, memory, and universal asynchronous receiver-transmitter (UART) are connected to the processor by an 8-bit-wide data/address bus.

### 1.3.1 Drive Control

The cartridge drive motors are powered by servo-regulated speed and direction circuits. These are controlled by the processor, which monitors with tachometers and with signals from the tape. The heads are selected by processor-controlled switches and either feed the automatic-gain-controlled (AGC) read amplifier and decoder circuits or are driven by write currents encoded by the processor.

### 1.3.2 Processor

The processor consists of an 8085 processor supported by firmware in a 2-kilobyte, read only memory (ROM) and by scratchpad and data buffer memory in a 256-byte random access memory (RAM). The processor communicates with the drive control circuitry through a bidirectional I/O port. The UART exchanges data between the TU58 processor bus and the host computer via the serial line drivers and receivers.

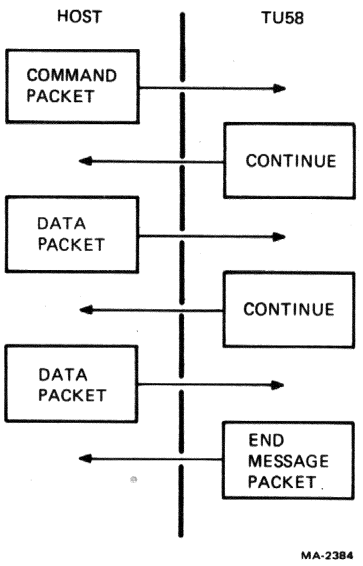


Figure 1-2 An Exchange in Radial Serial Protocol

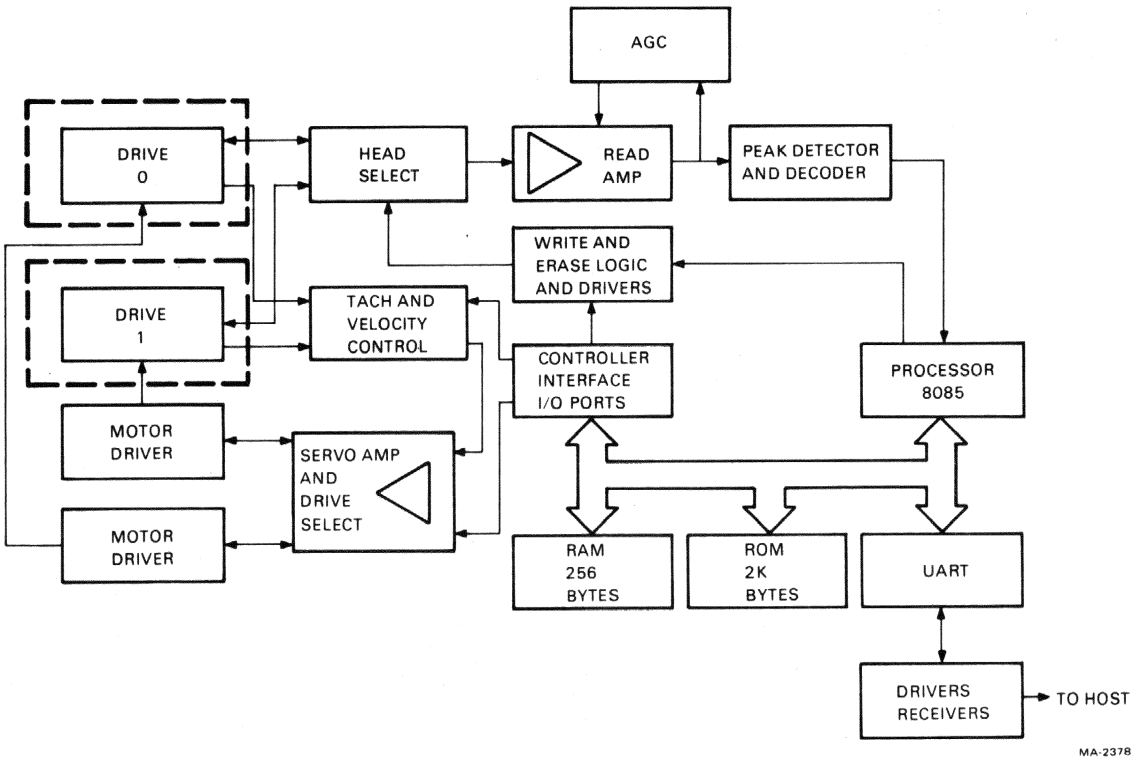


Figure 1-3 TU58 Block Diagram

## 1.4 SPECIFICATIONS

### 1.4.1 Performance

| | |
|---|---|
| Capacity per cartridge | 262,144 bytes, formatted in 512 blocks of 512 bytes each |
| Data transfer rate | |
| Read/write on tape | 41.7 $\mu$s/data bit, 24 Kbits/s |
| Data buffer to interface | 150 to 38.4 kbaud, jumper selected |
| Cartridge life | 5000 minimum end-to-end-and-back tape passes |
| Data reliability | |
| Soft data error rate | 1 in $10^7$ bits read (before self-correction) |
| Hard error rate | 1 in $10^9$ bits read (unrecoverable within 8 automatic retries) |
| Hard error rate with write-verify and system correction | 2 in $10^{11}$ bits read/written |
| Error checking | Checksum with rotation |

| | |
|---|---|
| Average access time | 9.3 seconds |
| Maximum access time | 28 seconds |
| Read/write tape speed | 76 cm/s (30 in/s) |
| Search tape speed | 152 cm/s (60 in/s) |
| Bit density | 315 bits/cm (800 bits/in) |
| Flux reversal density | 945 fr/cm (2400 fr/in) |
| Recording method | Ratio encoding |
| Medium | DECtape II cartridge with 42.7 m (140 ft) of 3.81 mm (0.150 in) tape. Size: 6.1 × 8.1 × 1.3 cm (2.4 × 3.2 × 0.5 in). Order TU58-K. |
| Track format (Figure 1-4) | Two tracks, each containing 1024 individually numbered, firmware-interleaved "records." Firmware manipulates four records at each operation to form 512-byte blocks. |
| Drive | Single motor, head integrally cast into molded chassis. |
| Drives per controller | 1 or 2 (only one may operate at a time) |

### 1.4.2 Electrical

| | |
|---|---|
| Power consumption | |
| Board and 1 or 2 drives | 11 W typical, drive running<br>+5 V ±5% at 0.75 A maximum<br><br>+12 V + 10% −5% at 1.2 A, peak<br>0.6 A average running<br>0.1 A idle<br><br>These voltages need not stabilize simultaneously upon power-on. |

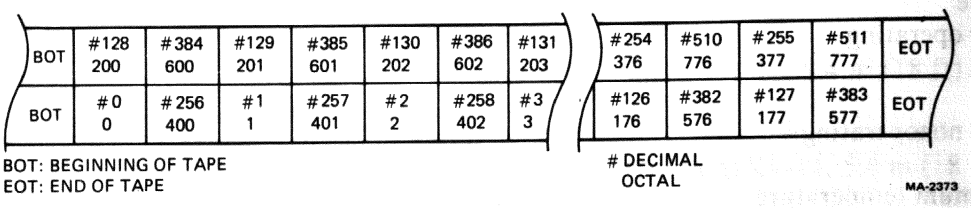


Figure 1-4 Block Locations on Tape

| | |
|---|---|
| Rackmount | 90 – 128 Vac, 180 – 256 Vac, 47 – 63 Hz, 35 W maximum |
| Serial interface standards | In accordance with RS-422 or RS-423; compatible with RS-232-C. |

### 1.4.3 Mechanical

| | |
|---|---|
| Drive | 8.1 H × 8.3 D × 10.6 W cm (3.2 × 3.3 × 4.1 in) with 19 cm (7.5 in) cable; 0.23 kg (0.5 lb) |
| Board | 13.2 H × 26.5 D × 3.5 W cm (5.19 × 10.44 × 1.4 in); 0.24 kg (0.53 lb) |
| TU58-DA | Same rackspace as -CA. See -EA for chassis. |
| TU58-CA rackmount cabinet | 13.2 H × 38.1 D × 48.3 W cm (5.19 × 15.0 × 19.0 in); 9 kg (20 lbs) |
| TU58-EA, -EB, -VA | 9.2 H × 29.5 D × 33.7 W cm (3.6 × 11.6 × 13.3 in); with rubber feet, add 1.5 H cm (0.6 in) |
| Power connector to board | AMP 87159-6 with 87027-3 contacts DIGITAL PN 12-12202-09, 12-12203-00) |
| Power connector to rackmount | European IEC standard |
| Interface connector to board | AMP 87133-5 with 87124-1 locking clip contacts and 87179-1 index pin (PN 12-14268-02, 12-14267-00, 12-15418-00) |

### 1.4.4 Environmental

When the TU58-AB or -BB is integrated in a host device such as a terminal, convection provides adequate cooling if the interior temperature is below 50° C (122° F) dry bulb, 26° C (79° F) wet bulb.

| | |
|---|---|
| Maximum dissipation | |
| TU58-CA, -DA, -EA, -EB | 120 Btu/hour |
| TU58-AB, -BB, -VA | 34 Btu/hour |
| Temperature | |
| TU58 operating | 10° C (50° F) to 50° C (122° F) ambient |
| TU58 nonoperating | −34° C (−30° F) to 60° C (140° F) |
| Maximum temperature difference between ambient and TU58 board | 18° C (32.4° F) |

| | |
|---|---|
| Relative humidity, noncondensing | |
| TU58 operating | |
| Maximum dew point | 23° C (73.4° F) |
| Minimum dew point | 2° C (36° F) |
| Relative humidity | 10% to 90% |
| TU58 nonoperating | 5% to 98% |

**CAUTION**

**If a cartridge has been exposed to either the maximum or minimum temperature extreme, the tape should be rewound one complete cycle before using. This is done to bring the tape to the proper tension.**

## 1.5 CONFIGURATIONS

The TU58 is available in the following configurations with accompanying designations.

| | |
|---|---|
| TU58-CA | Rackmount, large chassis, two drives, serial interface controller board, power supply 115/230 V switch-selectable, detachable line cords and fuses for 115 V and 230 V, two cartridges, boot ROM for MR11-EA, User Guide, Field Maintenance Print Set (MP00747), two I/O cables (BC17A-18 and BC17B-18), diagnostic kit (ZJ287-RG). |
| TU58-DA | Rackmount, tabletop chassis, two drives, serial interface controller board, power supply 115/230 V switch-selectable, detachable line cords and fuses for 115 V and 230 V, two cartridges, two I/O cables (BC17A-18 and BC17B-18), boot ROM for MR11-EA, accessory assembly hardware kit (70-16753-00), User Guide, Field Maintenance Print Sets (MP01014 and MP01063). |
| TU58-EA | Tabletop, two drives, serial interface controller board, power supply 115/230 V switch-selectable, detachable line cord and fuse for 115 V, accessory assembly hardware kit (70-16753-00), User Guide, Field Maintenance Print Set (MP01014). |
| TU58-EB | Tabletop, two drives, serial interface controller board, power supply 115/230 V switch-selectable, detachable line cords and fuses for 115 V and 230 V, two cartridges, two I/O cables (BC17A-18 and BC17B-18), boot ROM for MR11-EA, accessory assembly hardware kit (70-16753-00), User Guide, Field Maintenance Print Set (MP01014). |
| TU58-VA | Tabletop, two drives, serial interface controller board, dc power cable (70-17569-1C), I/O cable (70-17568-1F), two cartridges, MXV11-A-2 boot ROM, User Guide, Configuration Guide, Field Maintenance Print Set (MP01013), accessory assembly hardware kit (70-16753-01). |

**Additional Supplies**

| | |
|---|---|
| BC17A-18 | Interface cable from TU58 to DL-11 and DLV-11, 5.4 m (18 ft) (10-pin-to-40-pin connector). |
| BC17B-18 | Interface cable from TU58 to DLV-11J and MXV-11, 5.4 m (18 ft) (10-pin-to-10-pin connector). |

**NOTE**

**BC17A-18 and BC17B-18 cables replace BC20Y-25 and BC20Z-25, respectively. The new cables have an improved shield connection to improve system compliance with FCC regulations, Section 15, Subpart J.**

| | |
|---|---|
| BC20M-50 | Interface cable from TU58 to DLV-11J and MXV-11, 15 m (50 ft) (10-pin-to-10-pin connector). |
| BC20N-05 | Null modem cable from TU58 to EIA connector, 1.5 m (5 ft) (10-pin-to-DB25-S female). |
| BC21B-05 | Modem cable from TU58 to EIA connector, 1.5 m (5 ft) (10-pin-to-DB25-P male). |
| TU58-K | Preformatted tape cartridges, available singly or in packs of five. |
| TUC-01 | Tape Drive Cleaning Kit. |
| TU58-DB | Rackmount installation kit for tabletop versions -EA, -EB, -VA. |
| TU58-EC | Accessory kit containing detachable line cord for 115 V, accessory assembly hardware kit (70-16753-00), User Guide, Field Maintenance Print Set (MP01014). |
| TU58-ED | Accessory kit containing detachable line cords for 115 V and 230 V and fuse for 230 V, two cartridges, two I/O cables (BC17A-18 and BC17B-18), boot ROM for MR11-EA, accessory assembly hardware kit (70-16753-00), User Guide, Field Maintenance Print Set (MP01014). |
| TU58-VB | Accessory kit containing dc power cable (70-17569-1C), I/O cable (70-177568-1F), two cartridges, MXV11-A2 boot ROM, User Guide, Configuration Guide, Field Maintenance Print Set (MP01013), accessory assembly hardware kit (70-16753-01). |
| 17-00090-00 | Line cord 250 V. |
| 70-16753-00 | Accessory assembly hardware kit with brackets for mounting TU58 tabletop versions to flat surface. |
| 70-16753-01 | Accessory assembly hardware kit with brackets for mounting TU58 tabletop versions below a flat surface. |
| 23-126F3-0-0 | Boot ROM for BDV11. |
| MXV11-A-2 | Boot ROM for MXV11. |
| 23-765A9-00 | Boot ROM for MR11-EA. |

## 1.6 HARDWARE DOCUMENTATION ORDERING INFORMATION

The following TU58 DECtape II Tape Subsystem hardware manuals can be purchased from DIGITAL's Accessory and Supplies Group.

| Part No | Title |
|---|---|
| EK-0TU58-UG | TU58 DECtape II User Guide |
| EK-0TU58-PS | TU58 DECtape II Pocket Service Guide |
| EK-0TU58-TM | TU58 DECtape II Technical Manual (microfiche or paper) |
| EK-0TU58-IP | TU58 DECtape II Illustrated Parts Breakdown |
| MP00747 | TU58-CA Field Maintenance Print Set |
| MP01014-00 | TU58-EA Field Maintenance Print Set |
| MP01013-00 | TU58-VA Field Maintenance Print Set |
| MP01063 | TU58-DB Field Maintenance Print Set |

**ORDERING**

You can order supplies and accessories from one of the following addresses, according to your location.

Continental USA

Call 800-258-1710, or mail order to:

Digital Equipment Corporation
P.O. Box CS2008
Nashua, NH 03061

New Hampshire

Call 603-884-6660, or mail order to:

Digital Equipment Corporation
P.O. Box CS2008
Nashua, NH 03061

Alaska or Hawaii

Call 408-734-4915, or mail order to:

Digital Equipment Corporation
632 Caribbean Drive
Sunnyvale, CA 94086

Canada

Call 800-267-6146, or mail order to:

Digital Equipment of Canada LTD.
P.O. Box 13000
Kanata, Ontario, Canada K2K 2A6
Att: A&SG Business Manager
Telex: 610-562-8732

# CHAPTER 2
# OPERATION

## 2.1 TU58-DA, -CA RACKMOUNT CONTROLS AND INDICATORS

### 2.1.1 Front Panel
The front panel (Figure 2-1) has two slots for the tape cartridges and two tape motion indicators for the drives. In addition, the decorative bezel has a small compartment that can store up to four cartridges (six on the -CA) in their boxes.

### 2.1.2 Run Indicator
Each tape drive has an indicator that lights to show tape motion. Since data loss can occur if a cartridge is removed while the tape is being written, the cartridge should not be removed if the indicator is on.

### 2.1.3 Application and Removal of Power
The TU58-DA has a power switch on its backpanel, while the TU58-CA does not. If an outlet is available on a system power controller, the TU58 may be plugged into the controller. Otherwise, it does not need to be turned off. Its idling power consumption is less than 20 W.

When power is applied, the TU58 initializes itself, performs internal diagnostic tests, and then asks the host for an acknowledgement before it settles down to wait for instructions. See Paragraph 3.2.2 for a description of the required exchange.

If power is removed while a tape is being written, data may be lost. There are no other restrictions on power removal.

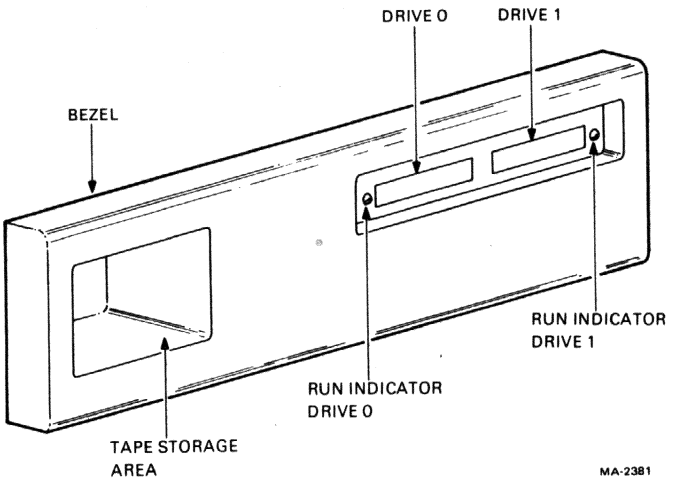


Figure 2-1 TU58-DA and -CA Rackmount Front Panel

## 2.2 TU58-EA, -EB CONTROLS AND INDICATORS

### 2.2.1 Front Panel
The front panel (Figure 2-2) has two slots for the tape cartridges and two tape motion indicators for the drives.

### 2.2.2 Run Indicator
Each tape drive has an indicator that lights to show tape motion in that drive.

### 2.2.3 Application and Removal of Power
The TU58-EA and -EB versions have power switches on their back panels. If an outlet is available on a system power controller, these versions may be plugged into the controller. Otherwise, they do not need to be turned off. Their idling power consumption is less than 20 W.

When power is applied, the TU58 initializes itself, performs internal diagnostic tests, and then asks the host for an acknowledgement before it settles down to wait for instructions. See Paragraph 3.2.2 for a description of the required exchange.

If power is removed while a tape is being written, data may be lost. There are no other restrictions on power removal.

## 2.3 TU58-VA CONTROLS AND INDICATORS

### 2.3.1 Front Panel
The front panel (Figure 2-2) has two slots for the tape cartridges and two tape motion indicators for the drives.

### 2.3.2 Run Indicator
Each tape drive has an indicator that lights to show tape motion in that drive.

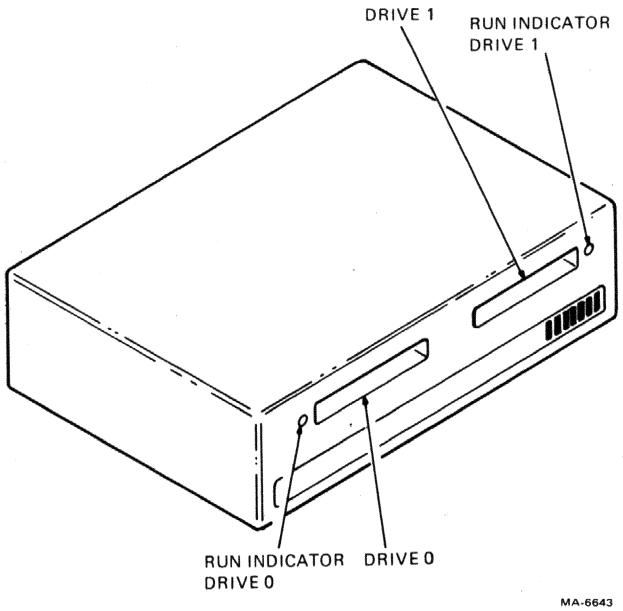


Figure 2-2 TU58-EA, -EB, and -VA Front Panel

### 2.3.3 Application and Removal of Power

The TU58-VA requires +5 V and +12 V from the device to which it connects. See the electrical specifications in Paragraph 1.4.2 for power requirements of a controller board and two drives. The part number of dc power cable supplied with the TU58-VA is 70-17569-1C. See Chapter 4 for installation information.

When power is applied, the TU58 initializes itself, performs internal diagnostic tests, and then asks the host for an acknowledgement before it settles down to wait for instructions. See Paragraph 3.2.2 for a description of the required exchange.

If power is removed while a tape is being written, data may be lost. There are no other restrictions on power removal.

## 2.4 TU58 COMPONENTS CONTROLS AND INDICATORS

See Chapter 5 for installation and operation of optional features.

### 2.4.1 Application and Removal of Power

The TU58 may be supplied with power from a host system. It is ready for operation within one second of voltage stabilization. It does not need to be turned off when not in use; its idling power consumption is less than 5 W.

When power is applied, the TU58 initializes itself, performs internal diagnostic tests, and then asks the host for an acknowledgment before it settles down to wait for instructions. See Paragraph 3.2.2 for a description of the required exchange.

If power is removed while a tape is being written, data may be lost. There are no other restrictions on power removal.

## 2.5 CARTRIDGE

### 2.5.1 Cartridge Loading

The TU58 drive is designed to make correct loading easy. To load the cartridge, hold it label-up, line it up with the grooves in the chassis, and slide it in with a firm push. Figure 2-3 illustrates the fit of the cartridge into the drive chassis grooves.

### 2.5.2 Cartridge Unloading

Unloading the cartridge is as simple as loading. Just pull it straight out. It is best to wait for the tape to stop (run indicator turns off) before removing the cartridge. The mechanism cannot be damaged by removing the cartridge while the tape is moving, but if a write is in progress, data may be lost. An error message is sent to the host if a command is interrupted by removal of a cartridge. The cartridge may be left in the drive as long as needed.

### 2.5.3 Keeping Track of Cartridges

If the TU58 is used in a non-file-structured system, the cartridge does not have an identifying number or label recorded on the tape. If a cartridge is changed, the TU58 does not know that a different cartridge was loaded; the operator must keep track of the contents of various cartridges.

### 2.5.4 Write Protect Tab

Each tape cartridge has a movable tab which, when properly positioned, protects data on the tape from unintended write operations. When this write protect tab (Figure 2-4) is in the inner position (toward the drive roller), it locks out the write circuitry.

When the write protect tab is in the outer position, it closes a switch in the chassis and allows the controller to write when it is commanded. The operator should be sure that system or program tapes are backed up with copies before loading them into the TU58 with their write protect tabs set to record.

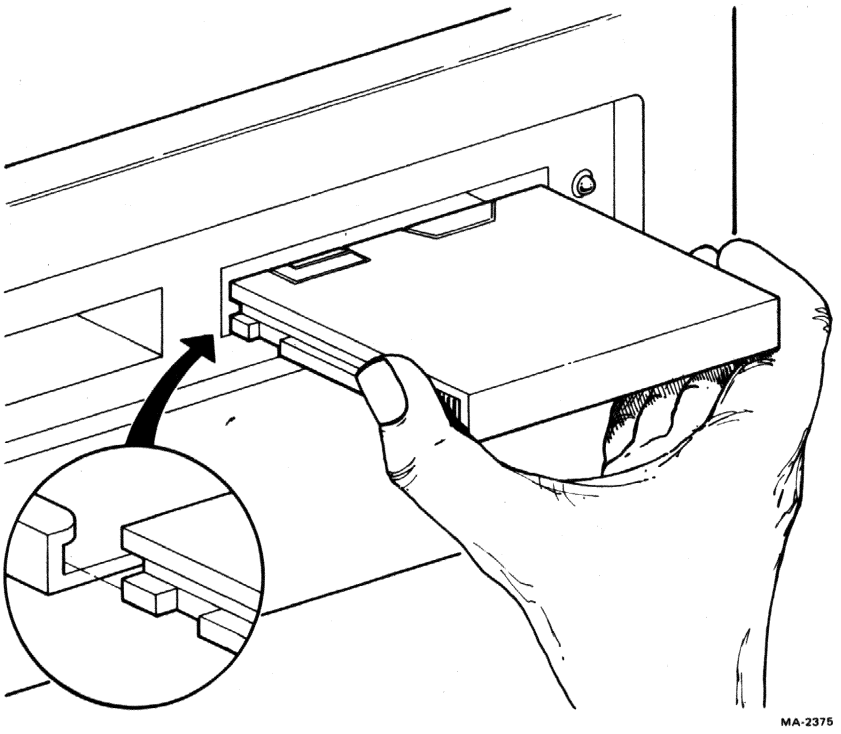


Figure 2-3 Cartridge Loading

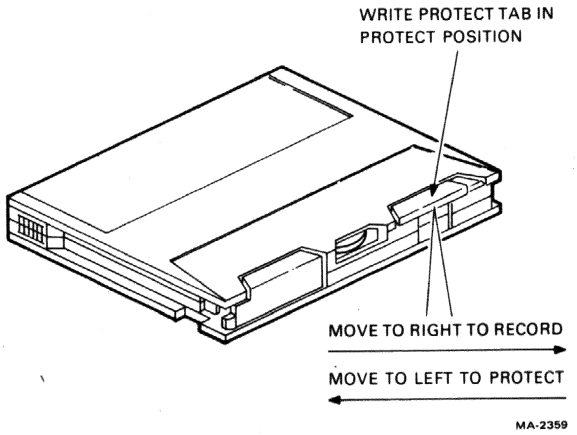


Figure 2-4 Write Protect Tab

The write protect tab can be completely removed for long-term write protection. On the metal-base cartridge, use a fingernail under the protruding end to lift the protect tab. Replace it by dropping it into its slot and pressing on it until it snaps. On the plastic-base cartridge, pry up the tab from its back edge part way and then lift from the front. To replace it, drop it into its slot and press forward and down.

### 2.5.5 Cartridge Storage and Care

Store cartridges in their cases, away from dust, heat, and direct sunlight. Do not touch the tape; there is no safe way to clean the tape and permanent errors may result. Keep tools and other ferrous or magnetic objects away. If it is possible that a tape has been exposed to environmental extremes (as listed in the specifications), and if the software operating system permits, wind it all the way through with a Newtape (Paragraph 3.1.2) or equivalent command, or by requesting positionings to blocks at each end of the tape before attempting to store data on the cartridge.

## 2.6 MAINTENANCE

### 2.6.1 Head and Puck Cleaning

After the first 20 hours of break-in runtime on each drive, clean the head and motor puck with a long-handled cotton applicator moistened with DIGITAL cleaning fluid (from cleaning kit TUC-01), 95 percent isopropyl alcohol, fluorocarbon TF, 113 or equivalent (Figure 2-5). Push the puck around with the applicator to clean its entire surface. After the first cleaning, repeat the procedure after every 100 hours of runtime. Regular cleaning minimizes tape and head wear and prevents tape damage and data errors caused by contamination. This is the only regular maintenance required by the TU58.

### 2.6.2 Operator Trouble Isolation

Table 2-1 lists potential problems and possible corrective actions and comments. (Some items are not applicable to components.)

### 2.6.3 Cartridge Wear

Cartridge tape is expected to last for 5000 end-to-end-and-back passes. If a cartridge is at the end of its life, a read operation may require several retries to get the data in the presence of soft errors. A soft error is a temporary data loss which is usually caused by a speck of dirt or oxide on the tape or head surface. This speck lifts the tape away from the head and causes signal loss and consequent read errors. A few extra passes of the tape past the head may knock the speck away and allow error-free reading. If it happens often, the tape is probably old and shedding oxide and should be copied and discarded as soon as possible.

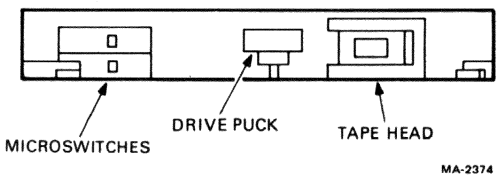


Figure 2-5 View Into Tape Drive Cartridge Slot

**Table 2-1 Operator Trouble Isolation**

| Symptom | Action/Comments |
|---|---|
| TU58 does not respond to host | 1. Ensure that the TU58-CA, -DA, -EA, or -EB is plugged into a live ac socket (or proper dc source for -VA or components).<br>2. Check that the voltage selection switch is properly set.<br>3. Ensure that the fuse and power cord are intact and properly inserted.<br>4. Check that the baud rates and interface standards are the same for both the TU58 and the host interface board (Paragraph 4.7).<br>5. If possible, observe the self-test indicator on the controller board. Remove the bezel on the rackmount version. When power is applied, the indicator should light for a half second, go out for another half second, and then relight. This means the controller has passed its automatic self-test and is ready for operation. If the indicator remains off, there is some problem within the board or in the interface.<br>Check that the interface cable is intact and properly inserted. If the serial interface is suspected and the standards are correct, try a new interface cable. An open wire in the line from the host prevents the indicator from coming on. Other causes require servicing. |
| TU58 does not write (reads okay) | 1. Check that the write protect tab is set correctly on the cartridge (Figure 2-4).<br>2. The trouble may be in a drive. Try writing on the other drive. Any problem except the write protect tab setting requires service. |
| Read errors (some host operating systems may provide this or a similar message) | 1. Clean the head. Dirt and tape oxide buildup can cause errors.<br>2. The tape may contain errors that were written onto it. If a tape is in poor condition or if data is not verified at write-time, errors may become a permanent part of the recording. A new cartridge with format problems will produce the same error message. Try another cartridge.<br>3. Motor or head is reaching end of life. Replace drive. |
| TU58 sends motor-stopped error messages | This indicates that a malfunction has occurred in the data recovery section and the runaway timer has stopped the motor. The TU58 should not be commanded to move tape more than twice under these conditions without checking the cartridge. Make sure that the tape is not getting near the end where it might come free of the hub. |

# CHAPTER 3
# PROGRAMMING

## 3.1 GENERAL PRINCIPLES

The TU58 is controlled by a microprocessor that frees the host computer from device-related operations, such as tape positioning and error retry. Only one high-level command to the microprocessor is necessary to initiate a complex operation. The host and TU58 communicate via strings of one or more bytes called packets. One brief packet can contain a message which completely describes a high-level command. The handshaking sequences between host and TU58 as well as packet format are defined by the radial serial protocol (RSP), or the modified radial serial protocol (MRSP), and were designed to be suitable for transmission by asynchronous interfaces.

### 3.1.1 Block Number, Byte Count, and Drive Number

The TU58 uses a drive number, block number, and byte count to write or read data. Figure 1-4 (Chapter 1) shows the locations of blocks on the tape. If all of the desired data is contained within a single 512-byte block, the byte count will be 512 or less. When the host asks for a particular block and a 512-or-less byte count, the TU58 positions the specified drive (unit) at that block and transfers the number of bytes specified. If the host asks for a block and also a byte count greater than that of the 512-byte boundary, the TU58 reads as many sequential blocks as are needed to fulfill the byte count. The same process applies to the write function. This means that the host software or an on-tape file directory need only store the number of the first block in a file and the file's byte count to read or write all the data without having to know the additional block numbers.

### 3.1.2 Special Handler Functions

Some device-related functions are not dealt with directly in the RSP, the MRSP, or in the TU58 firmware.

1. A short routine called Newtape (Appendix B) should be included in a TU58 handler to provide a complete wind-rewind for new or environmentally stressed tape cartridges. This procedure brings the tape to proper operating tension levels.

2. A TU58 handler should check the success code (byte 3 of the RSP or MRSP end message) for the presence of soft errors. This enables action to be taken before hard errors (permanent data losses) occur.

## 3.2 RADIAL SERIAL PROTOCOL (RSP) AND MODIFIED RSP (MRSP)

### 3.2.1 Packets

All communication between the host and the TU58 is accomplished via sequences of bytes called packets. There are two types of multi-byte packets: Control (Command) and Data. Either RSP or MRSP may be selected using the command packet switch byte. In addition, there are three single-byte packets used to manage protocol and control the state of the system: INIT, Continue, and XOFF.

**Control (Command)** – A Control packet is sent to the TU58 to initiate all operations. The packet contains a message completely describing the operation to be performed. In the case of a read or write operation, for example, the message includes the function to be performed, unit (drive) number, byte count and block number.

A special case of the Control packet, called an End packet, is sent from the TU58 to the host after completion of an operation or on an error. The End packet includes the status of the completed or aborted operation.

**Data** – The Data packet holds messages of between 1 and 128 bytes. This message is actually the data transferred from or to the TU58 during a read or write operation. For transmissions of larger than 128 bytes, the transfer is broken up and sent 128 bytes at a time.

**INIT** – This single-byte packet is sent to the TU58 to cause the power-up sequence. The TU58 returns Continue after completion, to indicate that the power-up sequence has occurred. When the TU58 makes a protocol error or receives an invalid command, it reinitializes and sends INIT continuously to the host. When the host recognizes INIT, it sends Break to the TU58 to restore the protocol.

**Bootstrap** – A flag byte saying Bootstrap (octal 10), followed by a byte containing a drive number, causes the TU58 to read block 0 of the selected drive. It returns the 512 bytes without radial serial packaging. This simplifies bootstrap operations. Bootstrap may be sent by the host instead of a second INIT as part of the initialization process described below.

**Continue** – Before the host sends a Data packet to the TU58, it must wait until the TU58 sends Continue. This permits the TU58 to control the rate that data packets are sent to it.

**XON** – An alternate term for Continue.

**XOFF** – Ordinarily, the TU58 does not have to wait between messages to the host. However, if the host is unable to receive all of a message from the peripheral at once, it may send XOFF. The TU58 stops transmitting immediately and waits until the host sends Continue to complete the transfer when it is ready. (Two characters may be sent by the UART to the host after the TU58 receives XOFF.)

**3.2.1.1 Packet Usage** – Position within the packet determines the meaning of each byte. All packets begin with a flag byte, which announces the type of packet to follow. Flag byte numeric assignments are as follows.

| | **Flag Byte Value** | |
|---|---|---|
| **Packet Type** | **Octal** | **Binary** |
| Data | 01 | 00001 |
| Control (Command) | 02 | 00010 |
| INIT | 04 | 00100 |
| Bootstrap | 10 | 01000 |
| Continue | 20 | 10000 |
| XON | 21 | 10001 |
| XOFF | 23 | 10011 |

(Bits 5 – 7 of the flag byte are reserved.)

Multiple-byte (Control and Data) packets also contain a byte count byte, message bytes, and two checksum bytes. The byte count byte is the number of message bytes in the packet. The two checksum bytes are a 16-bit checksum. The checksum is formed by summing successive byte-pairs taken as 16-bit words while adding any carry back into the sum (end-around carry). The flag and byte count bytes are included in the checksum. (See example in Appendix B.)

### 3.2.2 Break and Initialization

Break is a unique logic entity that can be interpreted by the TU58 and the host regardless of the state of the protocol. This is the logical equivalent of a bus init or a master reset. Break is transmitted when the serial line, which normally switches between two logic states called mark and space, is kept in the space condition for at least one character time. This causes the TU58's UART to set its framing error bit. The TU58 interprets the framing error as Break.

If communications break down, due to any transient problem, the host may restore order by sending Break and INIT as outlined above. The faulty operations are cancelled, and the TU58 reinitializes itself, returns Continue, and waits for instructions.

With DIGITAL serial interfaces, the initialize sequence may be sent by the following sequence of operations. Set the Break bit in the transmit control status register, then send two null characters. When the transmit ready flag is set again, remove the Break bit. This times Break to be one character time long. The second character is discarded by the TU58 controller. Next, send two INIT characters. The first is discarded by the TU58. The TU58 responds to the second INIT by sending Continue. When Continue has been received, the initialize sequence is complete and any command packet may follow.

### 3.2.3 Command Packets

The command packet format is shown in Table 3-1. Bytes 0, 1, 12, and 13 are the message delivery bytes. Their definitions follow.

**Table 3-1 Command Packet Structure**

| Byte | Byte Contents |
|---|---|
| 0 | Flag = 0000 0010($02_8$) |
| 1 | Message Byte Count = 0000 1010($12_8$) |
| 2 | Op Code |
| 3 | Modifier |
| 4 | Unit Number |
| 5 | Switches |
| 6 | Sequence Number – Low |
| 7 | Sequence Number – High |
| 8 | Byte Count – Low |
| 9 | Byte Count – High |
| 10 | Block Number – Low |
| 11 | Block Number – High |
| 12 | Checksum – Low |
| 13 | Checksum – High |

| | | |
|---|---|---|
| 0 | Flag | This byte is set to 00000010 to indicate that the packet is a Command packet. |
| 1 | Message Byte Count | Number of bytes in the packet, excluding the four message delivery bytes. This is decimal 10 for all command packets. |
| 12,13 | Checksum | The 16-bit checksum of bytes 0 through 11. The checksum is formed by treating each pair of bytes as a word and summing words with end-around carry. |

The remaining bytes are defined below.

| | | |
|---|---|---|
| 2 | Op Code | Operation being commanded. (See Table 3-4 and Paragraph 3.3 for definitions.) |
| 3 | Modifier | Permits variations of commands. |
| 4 | Unit Number | Selects drive 0 or 1. |
| 5 | Switches | Selects maintenance mode and specifies RSP or MRSP. |
| 6,7 | Sequence Number | Always zero for TU58. |
| 8,9 | Byte Count | Number of bytes to be transferred by a read or write command. Ignored by other commands. |
| 10,11 | Block Number | The block number to be used by commands requiring tape positioning. |

**3.2.3.1 Maintenance Mode** – Setting bit 4 of the switches byte (byte 5) to 1 in a read command inhibits retries on data errors. Instead, the incorrect data is delivered to the host followed by an end packet. The success code in the end packet indicates a hard data error. Since data is transmitted in 128-byte packets, a multiple packet read progresses normally until a checksum mismatch occurs. Then the bad data packet is transmitted, followed by the end packet, and the operation terminates.

**3.2.3.2 Special Address Mode** – Setting the most significant bit of the modifier byte (byte 3) to 1 selects special address mode. In this mode all tape positioning operations are addressed by 128-byte records (0-2047) instead of 512-byte blocks (0-511). Zero-fill in a write operation only fills out to a 128-byte boundary in this mode. To translate between normal addressing and special addressing, multiply the normal address by 4. The result is the address of the first 128-byte record of the block. Add 1, 2, or 3 to get to the next three 128-byte records.

### 3.2.4 Data Packets

**3.2.4.1 Radial Serial Protocol** – A data transfer operation uses three or more message packets. The first packet is the command packet from host to the TU58. Next, the data is transferred in 128-byte packets in either direction (as required by read or write). After all data is transferred, the TU58 sends an end packet. If the TU58 encounters a failure before all data has been transferred, it sends the end packet as soon as the failure occurs.

The data packet is shown in Table 3-2. The flag byte is set to $001_8$. The number of data bytes may be between 1 and 128 bytes. For data transfers larger than 128 bytes, the transaction is broken up and sent 128 bytes at a time. The host is assumed to have enough buffer capacity to accept the entire transaction, whereas the TU58 only has 128 bytes of buffer space. For write commands, the host must wait between message packets for the TU58 to send the Continue flag $020_8$ before sending the next packet. Because the host has enough buffer space, the TU58 does not wait for a Continue flag between message packets when it sends back read data.

**3.2.4.2 Modified Radial Serial Protocol** – When the host does not have sufficient buffer space to accept entire transactions at the hardware selected data transfer rate, modified radial serial protocol (MRSP) may be specified using the command packet switch byte. Bit 3 of the switch byte is set to specify the MRSP. Bit 3 remains set until intentionally cleared or cleared during power up. A good practice is to set bit 3 in every MRSP command packet.

MRSP is identical to RSP except during transmission to the host. When a command packet specifies MRSP for the first time (that is, bit 3 of the switch byte was previously cleared or cleared during power up), the TU58 will send one data or end packet byte (whichever occurs first). The subsequent bytes, up to and including the last byte of the end packet, will not be transmitted until a Continue or an XON is received from the host. To prevent a protocol error from occurring, it is necessary to transmit Continue or XON before transmitting any command packets. If a protocol error is detected, continuous INITs are sent with the Continue handshake. If a bootstrap is being transmitted, however, no handshake is employed.

### 3.2.5 End Packets

The end packet is sent to the host by the TU58 after completion or termination of an operation or an error. End packets are sent using RSP or MRSP as specified by the last command packet. The end packet is shown in Table 3-3.

**Table 3-2 Data Packets**

| Byte | Byte Contents |
|---|---|
| 0 | Flag = 0000 0001 |
| 1 | Byte Count = M |
| 2 | First Data Byte |
| 3 | Data |
| . | . |
| . | . |
| . | . |
| M | Data |
| M + 1 | Last Data Byte |
| M + 2 | Checksum L |
| M + 3 | Checksum H |

**Table 3-3 End Packet**

| Byte | Byte Contents |
|---|---|
| 0 | Flag = 0000 0010 |
| 1 | Byte Count = 0000 1010 |
| 2 | Op Code = 0100 0000 |
| 3 | Success Code |
| 4 | Unit |
| 5 | Not Used |
| 6 | Sequence No. L |
| 7 | Sequence No. H |
| 8 | Actual Byte Count L |
| 9 | Actual Byte Count H |
| 10 | Summary Status L |
| 11 | Summary Status H |
| 12 | Checksum L |
| 13 | Checksum H |

The definition of bytes 0, 1, 12, and 13 are the same as for the command packet. The remaining bytes are defined as follows,

Byte 2 — Op Code – 0100 0000 for end packet

Byte 3 — Success Code

| Octal | Decimal | | |
|---|---|---|---|
| 0 | 0 | = | Normal success |
| 1 | 1 | = | Success but with retries |
| 377 | −1 | = | Failed self test |
| 376 | −2 | = | Partial operation (end of medium) |
| 370 | −8 | = | Bad unit number |
| 367 | −9 | = | No cartridge |
| 365 | −11 | = | Write protected |
| 357 | −17 | = | Data check error |
| 340 | −32 | = | Seek error (block not found) |
| 337 | −33 | = | Motor stopped |
| 320 | −48 | = | Bad op code |
| 311 | −55 | = | Bad block number (>511) |

Byte 4 — Unit Number 0 or 1 for drive number

Byte 5 — Always 0.

Bytes 6,7 — Sequence number – always 0 as in command packet.

Bytes 8,9 — Actual byte count – number of bytes handled in transaction. In a good operation, this is the same as the data byte count in the command packet.

Bytes 10,11 — Summary Status

| | | |
|---|---|---|
| Byte 10 Bit 0 ↓ Bit 7 | | |
| Byte 11 Bit 0 | Reserved | |
| 1 | Reserved | |
| 2 | Reserved | |
| 3 | Reserved | |
| 4 | Logic error | |
| 5 | Motion error | |
| 6 | Transfer error | |
| 7 | Special condition (errors) | |

## 3.3 INSTRUCTION SET

The operation performed by the TU58 when it receives a Control (command) packet is determined by the op code byte in the control packet message. Note that while any command can specify modified radial serial protocol with the switch byte, the response will not be MRSP if a boot operation is being performed. Instruction set op code byte assignments are listed in Table 3-4.

To allow for future development, certain op codes in the command set have been reserved. These commands have unpredictable results and should not be used. Op codes not listed in the command set are illegal and result in the return of an end packet with the "bad op code" success code.

**Table 3-4 Instruction Set**

| Op Code Decimal | Op Code Octal | Instruction Set |
|---|---|---|
| 0 | 0 | NOP |
| 1 | 1 | INIT |
| 2 | 2 | Read |
| 3 | 3 | Write |
| 4 | 4 | (Reserved) |
| 5 | 5 | Position |
| 6 | 6 | (Reserved) |
| 7 | 7 | Diagnose |
| 8 | 10 | Get status |
| 9 | 11 | Set status |
| 10 | 12 | (Reserved) |
| 11 | 13 | (Reserved) |

The following is a brief description and usage example of each.

**OP CODE 0 NOP**

This instruction causes the TU58 to return an end packet. There are no modifiers to NOP. The NOP packet is shown below.

| BYTE | | | | |
|---|---|---|---|---|
| 0 | 0000 | 0010 | FLAG | |
| 1 | 0000 | 1010 | MESSAGE BYTE CNT | |
| 2 | 0000 | 0000 | OPCODE | |
| 3 | 0000 | 0000 | MODIFIER | |
| 4 | 0000 | 000X | UNIT NUMBER (IGNORED) | |
| 5 | 0000 | 0000 | SWITCHES (NOT USED) | |
| 6 | 0000 | 0000 | SEQ NO. (NOT USED) | |
| 7 | 0000 | 0000 | SEQ NO. (NOT USED) | |
| 8 | 0000 | 0000 | BYTE COUNT L | NO DATA |
| 9 | 0000 | 0000 | BYTE COUNT H | INVOLVED |
| 10 | 0000 | 0000 | BLOCK NO. L | NO TAPE |
| 11 | 0000 | 0000 | BLOCK NO. H | POSITION |
| 12 | 0000 | 001X | CHECKSUM L | |
| 13 | 0000 | 1010 | CHECKSUM H | |

The TU58 returns the following end packet.

| | | | | |
|---|---|---|---|---|
| 0 | 0000 | 0010 | FLAG | |
| 1 | 0000 | 1010 | MESSAGE BYTE CNT | |
| 2 | 0100 | 0000 | OPCODE | |
| 3 | 0000 | 0000 | SUCCESS CODE | |
| 4 | 0000 | 000X | UNIT (IGNORED) | |
| 5 | 0000 | 0000 | NOT USED | |
| 6 | 0000 | 0000 | SEQ. L | |
| 7 | 0000 | 0000 | SEQ. H | |
| 8 | 0000 | 0000 | ACTUAL BYTE CNT L | NO DATA |
| 9 | 0000 | 0000 | ACTUAL BYTE CNT H | INVOLVED |
| 10 | 0000 | 0000 | SUMMARY STATUS L | |
| 11 | XXXX | XXXX | SUMMARY STATUS H | |
| 12 | 000X | XXXX | CHECKSUM L | |
| 13 | XXXX | XXXX | CHECKSUM H | |

**OP CODE 1 INIT**
This instruction causes the TU58 controller to reset itself to a ready state. No tape positioning results from this operation. The command packet is the same as for NOP except for the op code and the resultant change to the low order checksum byte. The TU58 sends the same end packet as for NOP after reinitializing itself. There are no modifiers to INIT.

**OP CODE 2 Read, and Read with Decreased Sensitivity**
This instruction causes the TU58 to position the tape in the drive selected by Unit Number to the block designated by the block number bytes. It reads data starting at the designated block and continues reading until the byte count (command bytes 8 and 9) is satisfied. After data has been sent, the TU58 sends an end packet. Byte 3 indicates success, success with retries, or failure of the operation. In the event of failure, the end packet is sent at the time of failure without filling up the data count. The end packet is recognized by the host by the flag byte. The host sees a command flag (0000 0010) instead of a data flag (0000 0001).

There are two modifiers to the read command. Setting the least significant bit of byte 3 to 1 causes the TU58 to read the tape with decreased sensitivity in the read amplifier. This makes the read amplifier miss data if any weak spots are present. Thus, if the TU58 can read error-free in this mode, the data is healthy. The read transaction between TU58 and host is shown for 510 bytes (just under a full block) in Figure 3-1. Setting the most significant bit of byte 3 to 1 selects special address mode. See Paragraphs 3.2.3.1 and 3.2.3.2.

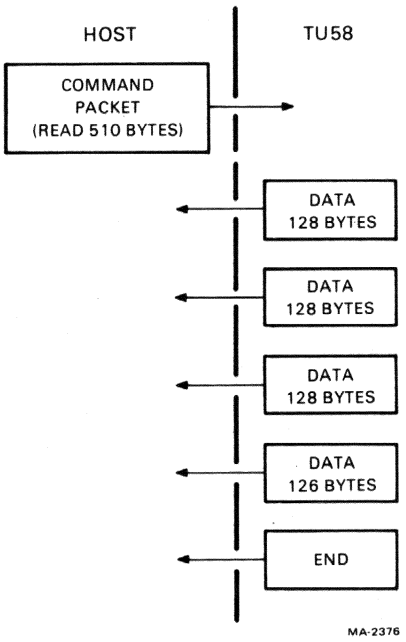


Figure 3-1 Read Command Packet Exchange

**OP CODE 3 Write, and Write and Read Verify**
This op code causes the TU58 to position the tape in the selected drive to the block specified by the number in bytes 10,11 of the command packet and write data from the first data packet into that block. It writes data from subsequent data packets into one or more blocks until the byte count called out in bytes 8, 9 of the command packet has been satisfied.

The controller automatically zero-fills any remaining bytes in a 512-byte tape block.

There are two modifiers permitted with the write command. Setting the least significant bit of byte 3 to 1 causes the TU58 to write all of the data and then back up and read the data just written with decreased sensitivity and test the checksum of each record. If all of the checksums are correct, the TU58 sends an end packet with the success code set to 0 (or 1 if retries were necessary to read the data). Failure to read correct data results in a success code of – 17 ($357_8$) to indicate a hard read error. Setting the most significant bit of byte 3 to 1 selects special address mode. See Paragraph 3.2.3.2.

The write operation has to cope with the fact that the TU58 only has 128 bytes of buffer space. It is necessary for the host to send a data packet and wait for the TU58 to write it before sending the next data packet. This is accomplished using the continue flag. The continue flag is a single byte response of 0001 0000 from TU58 to host. The RSP write transaction for both write and write/verify operations is shown in Figure 3-2. The MRSP write transaction for both write and write/verify operations is shown in Figure 3-3.

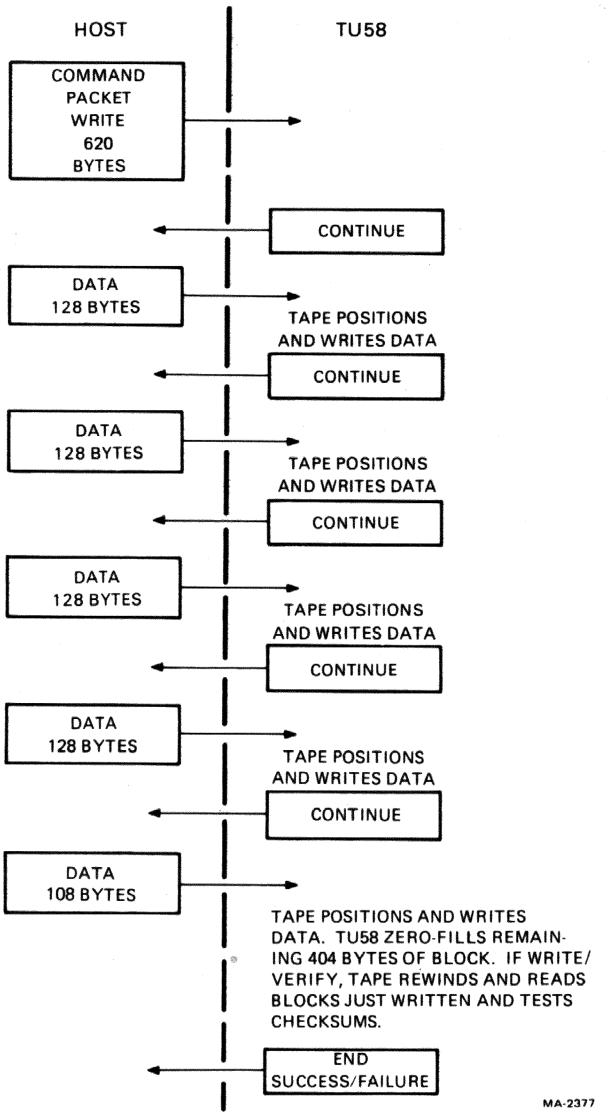


Figure 3-2 RSP Write Transaction

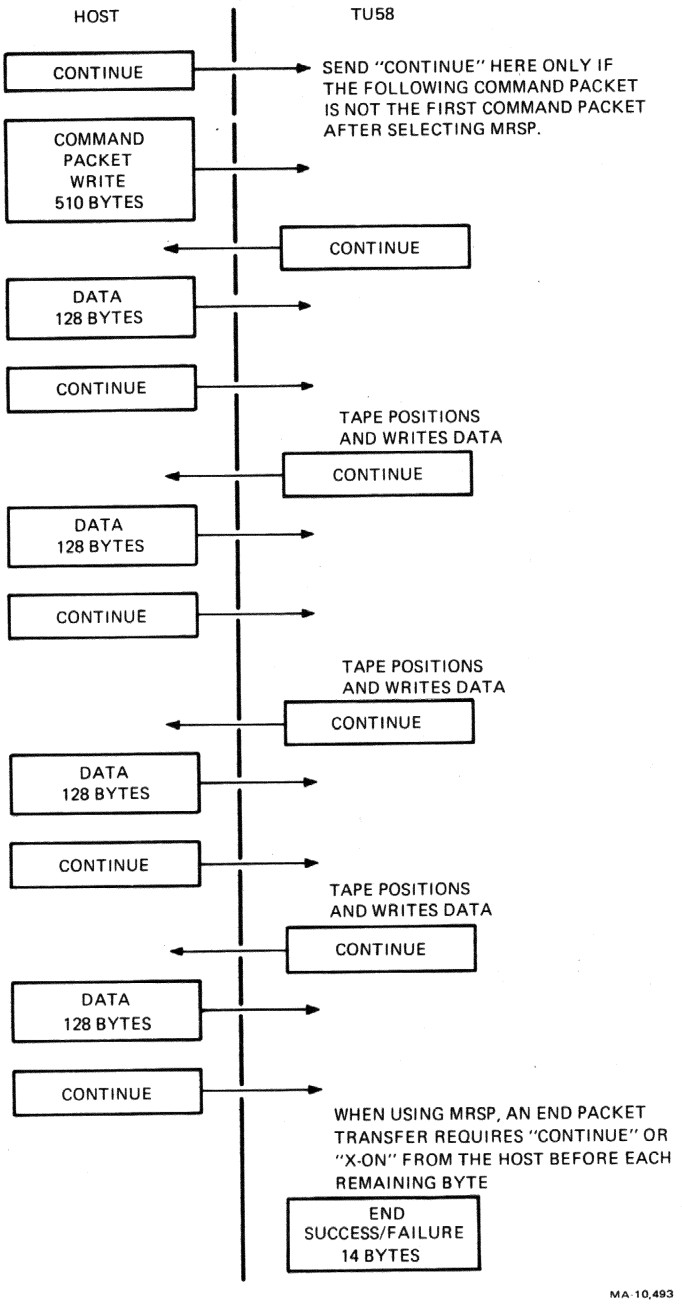


Figure 3-3 MRSP Write Transaction

**OP CODE 4 (Reserved)**

**OP CODE 5 Position**
This command causes the TU58 to position tape on the selected drive to the block designated by bytes 10, 11. After reaching the selected block, it sends an end packet. See Paragraph 3.2.3.2.

**OP CODE 6 (Reserved)**

**OP CODE 7 Diagnose**
This command causes the TU58 to run its internal diagnostic program which tests the processor, ROM, and RAM. Upon completion, TU58 sends an end packet with appropriate success code (0 = Pass, −1 = Fail). Note that if the bootstrap hardware option is selected, boot information will be transmitted without handshaking even if the switch byte specifies MRSP.

**OP CODE 8 Get Status**
This command is treated as a NOP. The TU58 returns an end packet.

**OP CODE 9 Set Status**
This command is treated as a NOP because TU58 status cannot be set from the host. The TU58 returns an end packet.

**OP CODE 10 (Reserved)**

**OP CODE 11 (Reserved)**

## 3.4 PASCAL TU58 HANDLER ALGORITHM DEFINITIONS

The following collection of algorithms describes the basic functions required for using the TU58 in a system. These algorithms are written in a pseudo-Pascal language and are therefore only designed to illustrate the logic of operations involved in causing the TU58 to perform the intended function. Actual software for a particular host computer may be written using these algorithms along with the program examples found in Appendix C. The following is a list of the functions described.

1. **tudiagnose** – Constructs and sends the command packet causing the TU58 to execute its built-in, self-test diagnostic. Returns the TU58 end packet success code as the result.

2. **tuseek** – Constructs and sends the command packet which causes the TU58 to position the tape inserted in the specified drive to the specified block. Returns the success code obtained from the TU58 end packet as the result.

3. **turead** – Constructs and sends the command packet which causes the TU58 to read data from the tape in the specified drive into a buffer area. Returns the success code obtained from the TU58 end packet as the result.

4. **tuwrite** – Constructs and sends the command packet which causes the TU58 to write data from the buffer area specified to the specified TU58 tape unit. Returns the success code obtained from the TU58 end packet as the result.

In addition to the above specific functions, algorithms for supporting routines are also provided. These routines are shared by the TU58 function routines and are included for the sake of completeness.

Digital Equipment Corporation assumes no responsibility for the correctness of these algorithms, nor offers corrections should errors be present. These algorithms may be copied for use on any computer system by any suitable language implementation.

```
CONSTANTS  { Defines some interesting and useful constants }

   { Define single byte packet flags }

   data              = 1;
   control           = 2;
   init              = 4;
   continue          = 16;
   xoff              = 19;

   { Define multi-byte packet opcodes }

   read_opcode       = 2;
   write_opcode      = 3;
   position_opcode   = 5;
   diagnose_opcode   = 7;
   end_pack_opcode   = 64;

   { Define initialization success codes }

   success           = 0;
   failure           = -127;

   { Define some useful subscript values }

   command_flag      = 0;
   command_count     = 1;
   command_opcode    = 2;
   command_unit      = 4;
   data_count_low    = 8;
   data_count_high   = 9;
   command_block_low = 10;
   command_block_high = 11;

   { Define length of command/data messages }

   command_length    = 10;
   data_block        = 128;


GLOBAL_VARIABLES  { Indicate quantities used by all functions }

     single_byte_packet : byte;

     { Note the variable length of the array defined below.
     Its length is a function of the type of message to be
     sent, i.e., N is the number of bytes contained in the
     message }

     multi_byte_packet : ARRAY[0..N+3] OF byte;


CALLING_PARAMETERS { Parameters defined by the calling routines}

     unit_number, block_number :INTEGER; { Specified unit/block}
     no_bytes : INTEGER; { Number of bytes in message }
     buffer : ARRAY[1..no_bytes] OF byte; { Data/message space }
```

```
tudiagnose;

{ This routine runs the TU58 self-test diagnostic routine }

BEGIN
     { Construct and send the command packet
       necessary to cause the TU58 controller
       to execute its self-test diagnostic.
       Return the value of the success code
       contained in the TU58 end packet. }

     IF initialize(diagnose_opcode)=success THEN BEGIN
          send_packet(packet, command_length+2);
          tudiagnose := get_end_packet
          END
     ELSE
          tudiagnose := failure
END; { tudiagnose }



tuseek (unit_number, block_number);

{ Construct and send a command packet which indicates the
  TU58 should position the specified unit to the specified
  block.  This routine returns the success code sent in
  the TU58 end packet as its result. }

BEGIN
     IF initialize(position_opcode)=success THEN BEGIN

          { Construct/send a command packet }

          packet[command_unit] := unit_number;
          packet[command_block_low] := low(block_number);
          packet[command_block_high] := high(block_number);
          send_packet(packet, command_length);

          { Conclude with an end packet }

          tuseek := get_end_packet;
          END;
     ELSE
          tuseek := failure;
END; { tuseek }


turead (unit_number, block_number, buffer, no_bytes);

{ Constuct and send the command packet required to read
  'no_bytes' from the unit and block specified.  Reads data
  from the tape into a buffer space, returning the value of
  the success code contained in the TU58 end packet. }

BEGIN
     IF initialize(read_opcode)=success THEN BEGIN

          { Construct command packet; operators 'low','high'
            return low byte, high byte respectively }

          packet[command_unit] := unit_number;
          packet[command_block_low] := low(block_number);
          packet[command_block_high] := high(block_number);
          send_packet(packet, command_length);
```

```
          { Get data output and stuff in buffer }

          IF get_data_packet(buffer)=success THEN
               turead := get_end_packet
          ELSE turead := failure
     END
     ELSE
          turead := failure
END; { turead }

tuwrite (unit_number, block_number, buffer, no_bytes);

{ Construct and send a command packet specifying the
  unit and block for writing data.  Write the data from
  the specified buffer area to the tape.  Return the
  success code obtained from the TU58 end packet as the
  result.}

LOCAL_VARIABLES
     count, data_count : INTEGER;
BEGIN
     IF initialize(write_opcode)=success THEN BEGIN

          { Stuff parameters into command packet }
          packet[command_unit] := unit_number;
          packet[command_block_low] := low(block_number);
          packet[command_block_high] := high(block_number);
          send_packet(packet, command_length);

          { If continue is received, send data; 'get_byte'
            'put_byte'are implementation-dependent function
            calls }

          data_count := no_bytes

          WHILE get_byte=continue THEN DO BEGIN
               { Make blocks maximum of 128 bytes each }
               count := minimum(data_count, data_block);
               put_byte(count);
               send_packet(buffer[no_bytes-data_count+1],
                         count);
               data_count := data_count - count
          END;

          { In any event, try for an end packet at end }

          tuwrite := get_end_packet
          END
     ELSE
          tuwrite := failure
END; { tuwrite }


initialize (op_code);

{ Initializes the TU58 by sending break characters on the
  communication line, followed by the single byte packet
  for INIT sent twice.  The specified operation is then
  transmitted to the TU58 as either a single byte packet
  or the first (command) byte of a multi-byte packet. }

BEGIN
     { Set communication line control to BREAK }
```

```
      set_break_bit;

      { Delay multiple character time to insure framing
       error }

      wait(n_character_times);

      { Remove BREAK condition from line }

      reset_break_bit;

      { Initialize command packet area }

      packet[command_flag] := control;
      packet[command_opcode] := op_code;

      { Send INIT flag to TU58 }

      single_byte_packet := init;
      put_byte(single_byte_packet);

      IF get_byte=continue THEN initialize := success
      ELSE initialize := failure;

END; { initialize }

send_packet (buffer, no_bytes);

{ Send information contained in 'buffer' to TU58 }

LOCAL_VARIABLES
      index, check_sum, check_word : INTEGER;

BEGIN
      { Must begin with checksum initialization }

      check_sum := 0;
      check_word := 0;

      { Check for even/odd bytes, performing checksum only
      if even;  the operator 'odd' returns a Boolean value
      of TRUE if the argument is odd }

      FOR index := 1 TO no_bytes DO BEGIN
            IF odd(index) THEN
                  check_word := buffer[index]
            ELSE BEGIN
                  check_word := buffer[index]*256 + check_word;
                  check_sum := check(check_sum, check_word)
            END;
            put_byte(buffer[index])
            END;

      IF odd(no_bytes) THEN
            check_sum := check(check_sum, check_word);

      { Now output checksum information; operators 'low',
      'high' return low byte, high byte respectively }

      put_byte(low(check_sum));
      put_byte(high(check_sum))

END; { send_packet }
```

```
get_data_packet (buffer);

{ Gets the data sent from the TU58 and stuffs it into
  'buffer' }

LOCAL_VARIABLES
     index, check_sum, data_count : INTEGER;

BEGIN
  { Initialize checksum variables }

  check_word := 0;
  check_sum := get_byte;

  { Look for valid data packet structure }

  IF (check_sum<>data) THEN get_data_packet := failure

   ELSE BEGIN
        { Get data packet from TU58, calculating
          checksum as the buffer is being filled }

        data_count := get_byte;
        check_sum := check_sum + data_count*256;

        FOR index := 1 TO data_count DO BEGIN
            buffer[index] := get_byte;
            IF odd(index) THEN
                check_word := buffer[index]
            ELSE BEGIN
                check_word := buffer[index]*256 + checkword;
                check_sum := check(check_sum, check_word)
            END
        END

        IF odd(data_count) THEN
            check_sum := check(check_sum, check_word);

        { Make sure packet was not in error }

        check_word := get_byte;
        check_word := check_word + get_byte*256;

        IF check_word<>check_sum THEN
            get_data_packet := failure
        ELSE get_data_packet := success
   END

END;    { get_data_packet }

get_end_packet;
{ Gets an end packet from the TU58, returning success code
as result }

LOCAL_VARIABLES
    index, check_sum, check_word : INTEGER;

BEGIN
    check_sum := get_byte;
    { Look for valid command packet structure }

    IF (check_sum<>command) OR (get_byte<>command_length)
        OR (get_byte<>end) THEN get_end_packet := failure
```

```
    ELSE BEGIN
        { Get success code from command packet }

        get_end_packet := get_byte;

        { Now do the checksum calculation }

        check_sum := check(check_word, get_end_packet*256);
        FOR index := 1 TO 4 DO BEGIN
            check_word := get_byte;
            check_word := check_word + get_byte*256;
            check_sum := check(check_sum, check_word)
        END;

        { Make sure packet was not in error }

        check_word := get_byte;
        check_word := check_word + get_byte*256;
        IF check_word<>check_sum THEN
            get_end_packet := failure
    END
END;    { get_end_packet }


check (arg1, arg2);

{ Computes the 16 bit checksum of arg1 and arg2, using end-
  around carry technique of TU58 }

BEGIN
    { The function 'carry' returns a value of
      1 if the sum of the arguments results in
      a carry; a value of 0 is returned otherwise }

    check := arg1 + arg2 + carry(arg1,arg2)

END;    { check}



        { End of algorithm definitions }
```

# CHAPTER 4
# INSTALLATION

## 4.1 INTRODUCTION

This chapter contains installation, configuration, and checkout procedures for all the versions of the TU58 DECtape II (-DA, -CA, -EA, -EB, -VA components).

## 4.2 RACK INSTALLATION (-DA Version)

### 4.2.1 Rackmount

The TU58-DA mounts in 13.2 cm (5.2 in) of standard 48.3 cm (19 in) width rack. It should be located so that the 2 m (6 ft) power cord can reach a power controller outlet box such as the DIGITAL 861 or any power outlet.

### 4.2.2 Unpacking

The TU58-DA shipping carton contains the following items for rackmounting.

| | |
|---|---|
| 1 | TU58-EB |
| 1 | Bezel |
| 2 | Mounting brackets |
| 2 | Support brackets |
| 2 | Support bracket extenders |
| 24 | Phillips trusshead screws 10-32 × 1/2 in |
| 24 | Internal lock washers |
| 12 | U-Nut retainers |
| 6 | Kep lock nuts 10-32 × 3/8 |
| 1 | Line cord (120 V) |
| 2 | Fuses (3/8 A and 3/4 A slow-blow) |

### 4.2.3 Power Selection

Detachable line cords for 115 V and 230 V, and two fuses are supplied with the TU58-DA. The line cord receptacle meets European IEC standards. A switch on the back of the tape drive rear panel selects 115 V or 230 V (Figure 4-1).

1. Set the voltage switch to the correct value using a small screwdriver.

| Switch Position | Voltage | Range |
|---|---|---|
| Left | 115 V | 90 – 128 Vrms |
| Right | 230 V | 180 – 256 Vrms |

**CAUTION**

**If the TU58 is plugged into a 230 V circuit while set for 115 V, it may be severely damaged.**

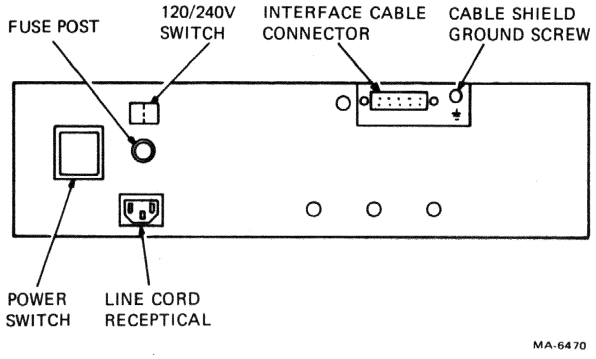


Figure 4-1 TU58-DA Rear Panel

2. From the two fuses provided, select and install the proper fuse in the fuse post. To open the fuse post, use a 3/16-inch blade type screwdriver. Press in the fuse post cap and turn it counterclockwise. To close the fuse post, use the screwdriver to press in the cap and turn it clockwise.

| Voltage | Fuse |
|---|---|
| 115 V | 3/4 A slow-blow |
| 230 V | 3/8 A slow-blow |

### 4.2.4 Removing Bottom Plates for Controller Board Configuration

The TU58 is shipped prewired for operation at 38.4K baud transmit and receive on RS-423. If a configuration change is necessary, the bottom plates must be removed in order to gain access to the controller board. Use the following procedure to remove the bottom plates. (See Paragraph 4.7 for configuration information.)

1. Disconnect the power cord and interface cable from the rear panel of the TU58 (Figure 4-1).

2. Place the TU58 upside-down on a flat working surface so the rear panel faces you.

3. Remove the two Phillips head screws and lock washers from the front plate. Remove the front plate, exposing the two tape drives.

4. Remove two Phillips head screws and lock washers from the bottom of the rear plate and one flat Phillips head screw from the rear panel at the left side of the interface cable connector. Remove the power supply assembly by lifting it out of the housing (with internal cables still attached) and placing it rightside-up next to the TU58.

### 4.2.5 Rackmounting Procedure

The following procedure enables one person to install the TU58-DA in the rack using a number 2 Phillips screwdriver.

1. With the power cord and interface cable removed (Figure 4-1), carefully place the TU58-DA upside-down on a flat working surface so the front of the device is facing you.

2. Remove the rubber feet if attached by removing the screws that hold them in place (Figure 4-1). Refer to Figure 4-2 and align the two support brackets on the bottom of the TU58-DA so they are flush with the left side. Fasten down the right side of each support bracket with two screws and two lock washers.

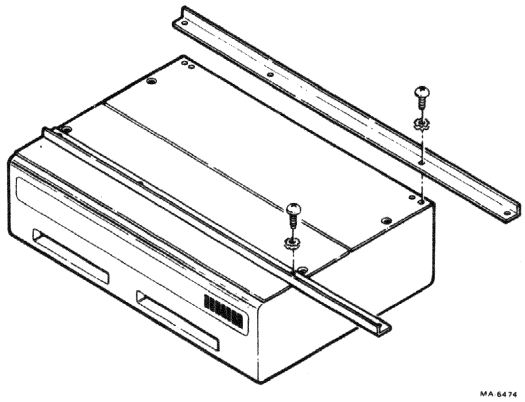

Figure 4-2 Installing Support Brackets

3. Fasten the mounting bracket to the left side of the TU58-DA with two screws and two lock washers so the ball stud faces forward (bend in bracket toward center of TU58-DA). Install the other mounting bracket in the same manner using two screws, two lock washers, and two lock nuts to secure it to the support brackets (Figure 4-3).

4. Attach four U-Nut retainers (two per side) to the front vertical rails (Figure 4-4). Refer to Figure 4-5 and position U-Nut retainers at the desired height for the TU58-DA.

5. Open the back of the rack. Attach four U-Nut retainers to the rear vertical rails (Figure 4-6).

6. If a non-DIGITAL rack is used, fasten the support bracket extenders to the rear vertical rails with four screws, four lock washers, and four lock nuts. Use two per side in the top and bottom holes (Figure 4-7).

7. Turn the TU58-DA rightside-up and while supporting it with one hand, place it into position in the rack.

**NOTE**

**Be sure the mounting brackets are to the inside of the rear vertical rails.**

8. Fasten the mounting brackets to the front vertical rails with four screws and four lock washers (two per side in the top and bottom holes).

**CAUTION**

**Install the two bottom screws first to avoid bending the mounting ears.**

9. For DIGITAL racks, fasten the mounting brackets to the rear vertical rails (Figure 4-6). For non-DIGITAL racks, fasten the mounting brackets to the support bracket extenders (Figure 4-7).

10. Attach the power cord and interface cable and connect to the appropriate device or receptacle. Close the back of the rack.

11. Install the bezel by pushing into place over the ball studs (Figure 4-8).

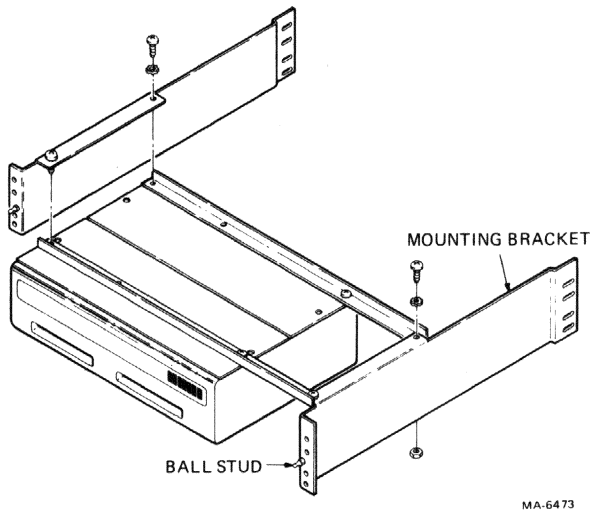


Figure 4-3 Installing Mounting Brackets

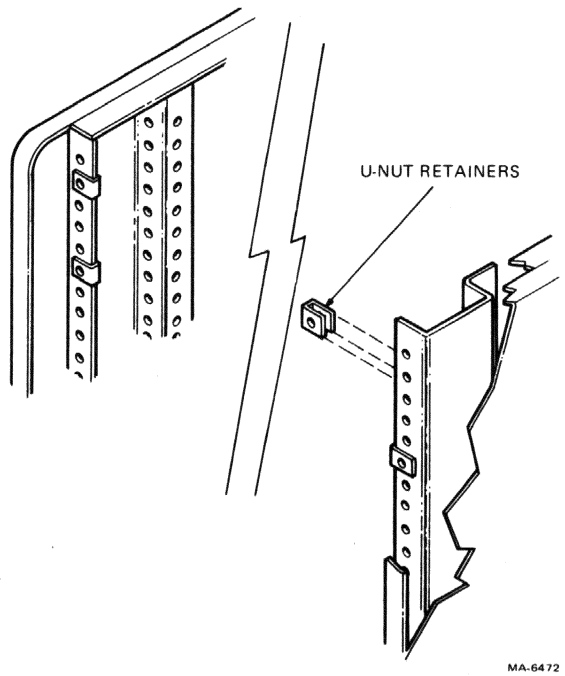


Figure 4-4 Front Vertical Rail U-Nut Retainers

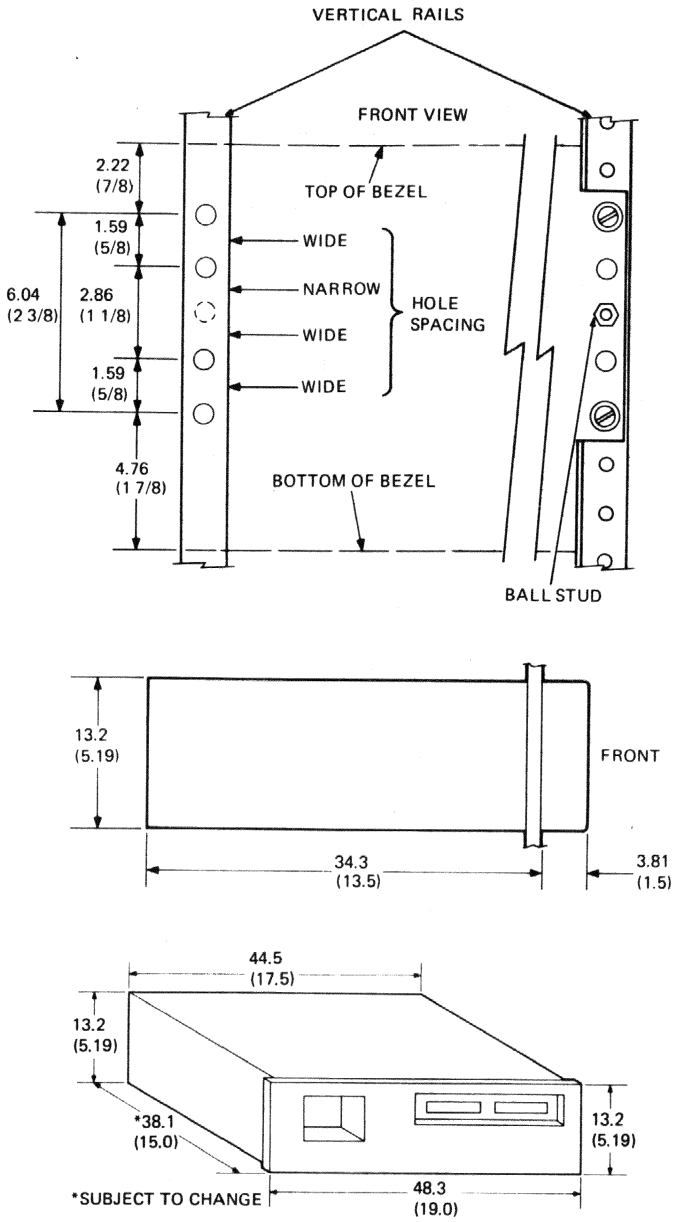


Figure 4-5 Rackmounting the TU58-DA

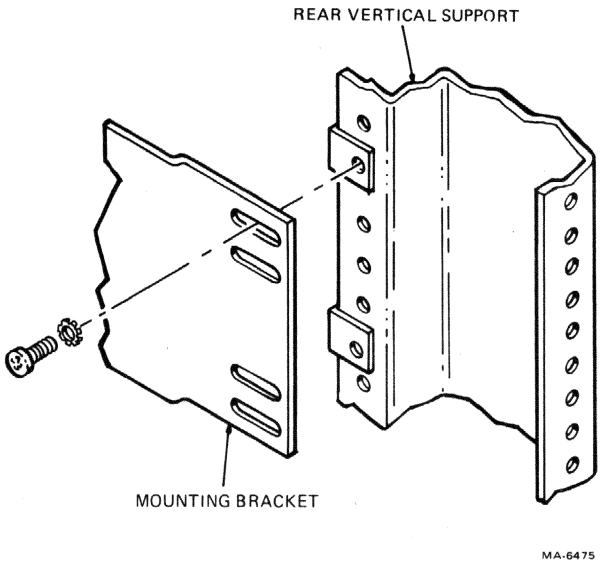


Figure 4-6 Rear Vertical Support U-Nut Retainers

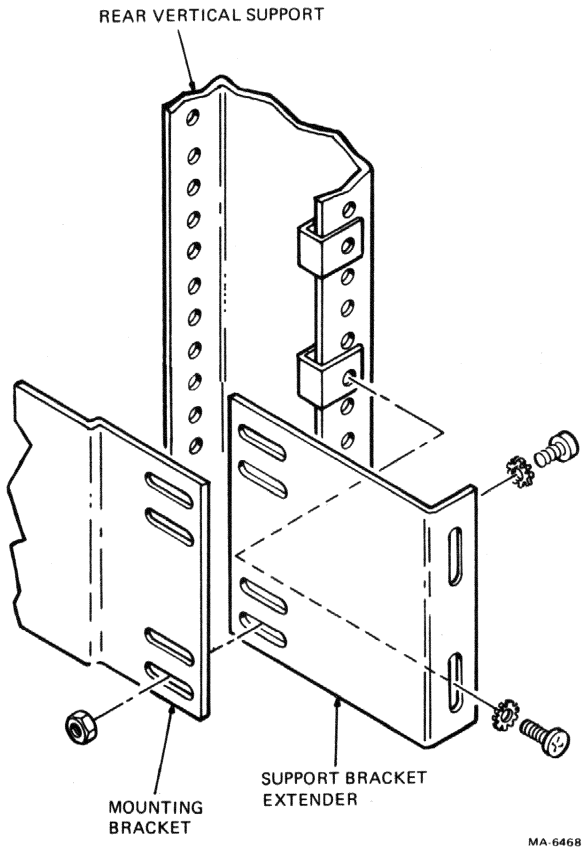


Figure 4-7 Fastening Support Bracket Extenders

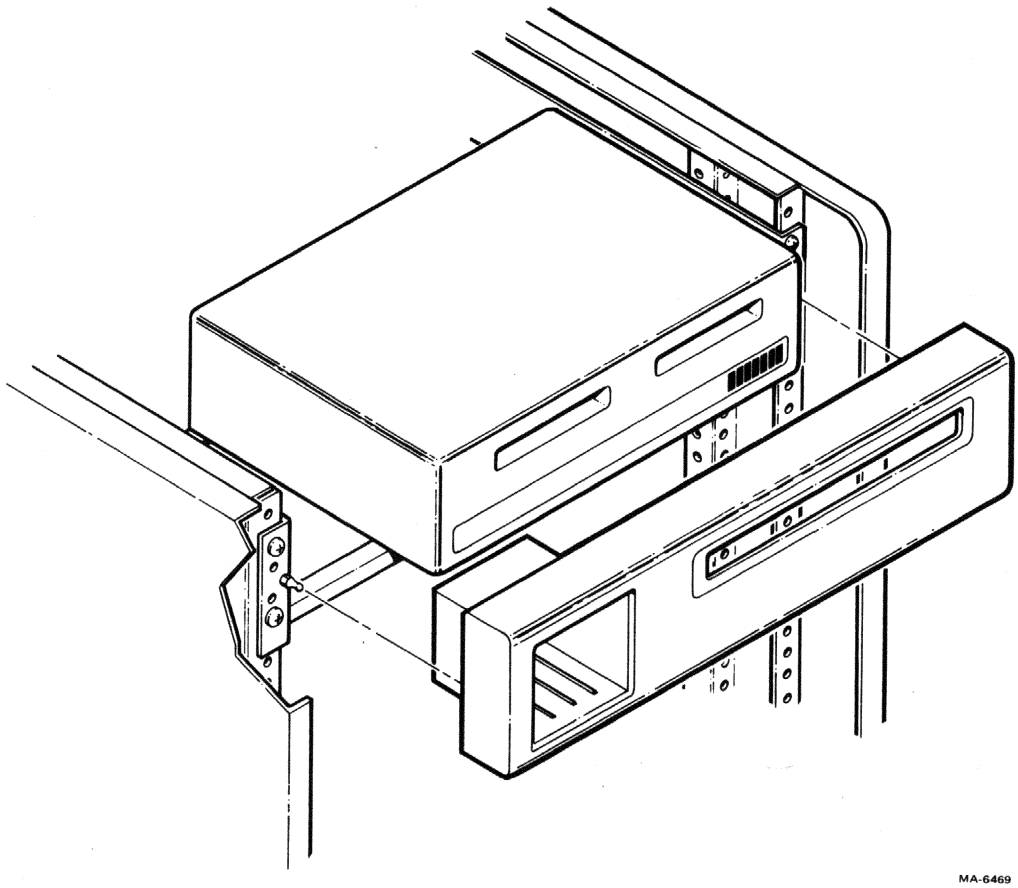


Figure 4-8 Installing the Bezel

## 4.3 RACK INSTALLATION (-CA Version)

### 4.3.1 Rackmount

The TU58-CA rackmount unit mounts in 13.2 cm (5.2 in) of standard 48.3 cm (19 in) width rack. It should be located so that the 2 m (6 ft) power cord can reach a power controller outlet box such as the DIGITAL 861 or any power outlet.

To get to the mounting holes, remove the bezel (Figure 4-9) by gripping it at the top and bottom with both hands. Rotate it out from the bottom and lift it away. If the unit is installed in a recessed rack, the bezel may be removed by gripping it with both hands on the left edge with fingers or thumbs inside the storage well. Pull sharply out and swing the bezel away.

**WARNING**
**Metal bezels are heavy!**

If the rack requires them, install four U-Nut retainers at the holes spaced according to Figure 4-10. The TU58 is light enough for one person to install. Put the two bottom screws in first to avoid bending the mounting ears.

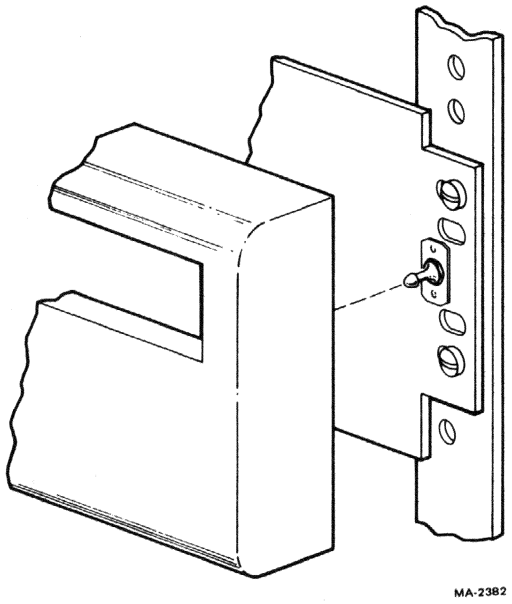


Figure 4-9 Bezel and Ball Stud

Four brackets and hardware are included with the TU58-CA to support the back end of the chassis in a rack. Use the two long brackets for DIGITAL cabinets. The short brackets are extenders for the long brackets used in non-DIGITAL cabinets. Attach the long brackets to the chassis with the existing power supply screws, and attach to the side rails of the rack with the supplied clipnuts and screws. Hardware is also provided to fasten the extender to the long bracket if required. The bend on the long bracket should point to the center of the rack while the bend on the extender should point to the outside of the rack.

### 4.3.2 Power Selection for the Rack Version

Line cords for 110 V and 220 V and two fuses are supplied with the TU58-CA. The chassis power receptacle meets European IEC standards. A switch on the back of the rackmount cabinet selects 110 V or 220 V (Figure 4-11).

1. Set the switch to the correct value using a small screwdriver.

**CAUTION**
**If the unit is plugged into a 220 V circuit while set for 110 V, it may be severely damaged.**

2. Install a fuse in the fuse post.

**NOTE**
**A 3/8 Amp slow-blow fuse is required for 220 V, a 3/4 Amp slow-blow fuse for 110 V.**

3. Insert the appropriate power cord into the receptacle. Do not plug it into an outlet until the installation is complete.

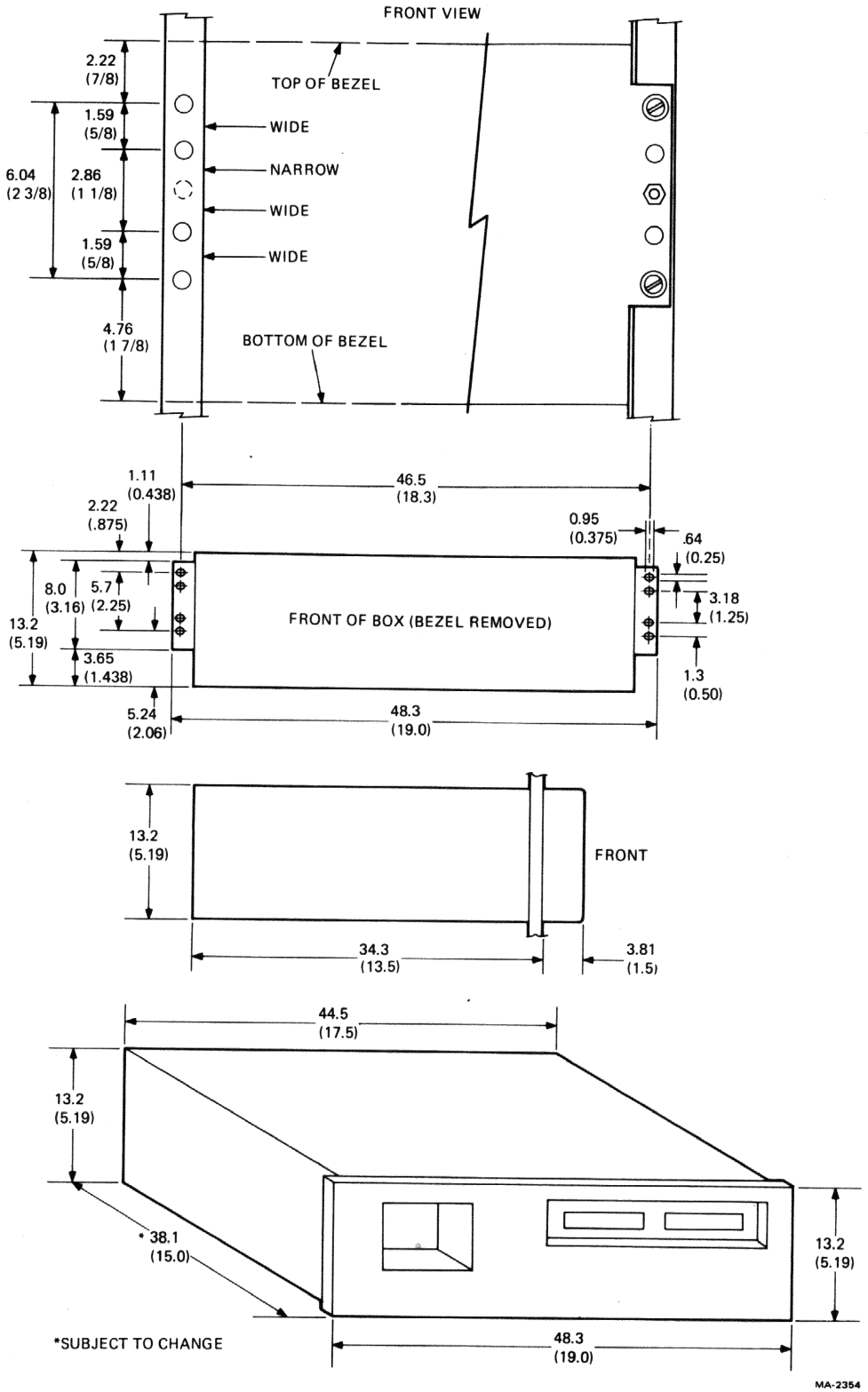


Figure 4-10 Rackmounting the TU58-CA

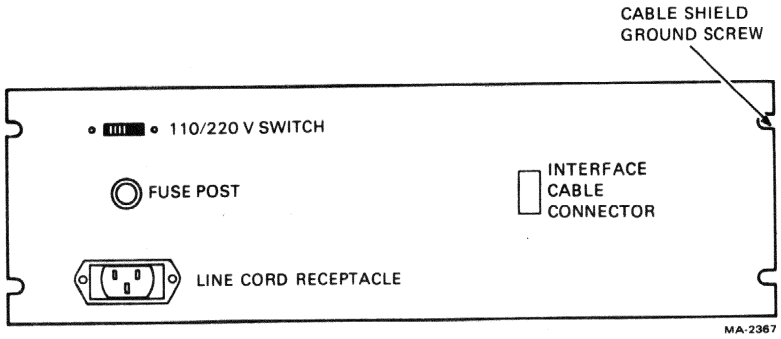


Figure 4-11 TU58-CA Rear Panel

### 4.3.3 Removing Module from Chassis

Refer to Figure 4-12 and perform the following steps.

1. Disconnect the power cord.
2. Remove the bezel.
3. Twist a coin or screwdriver in the gap between the retainer bar and the lip of the chassis. Lift the bar out of the chassis and set it aside.
4. Pull the cage toward you a few inches and turn it to the right. Slide the module out an inch or two and reach in at the back of the cage to remove the power and communication cables from the module connectors. Remove the cage entirely from the chassis and put it on a stable work surface.
5. Reach in again at the back of the cage and remove the drive cables from their connectors on the module.
6. Now slide the module out of the cage.

**CAUTION**

**Be careful around the thin tachometer disk. It is easily bent (and its edge is sharp). If the disk gets bent without creasing, it might be straightened with pliers. Alignment is not critical, but it is better if the disk does not rub against the optical sensor block. If it cannot be aligned, or if it is creased, it must be replaced.**

### 4.3.4 Reinstalling the Module

1. With the connector edge facing into the cage, slide the module partially into the cage along the card guides.
2. Install the drive cables onto their connectors. Note that the drive cables cross each other, with the left drive cable going to the right connector (as you look into the open end of the cage).

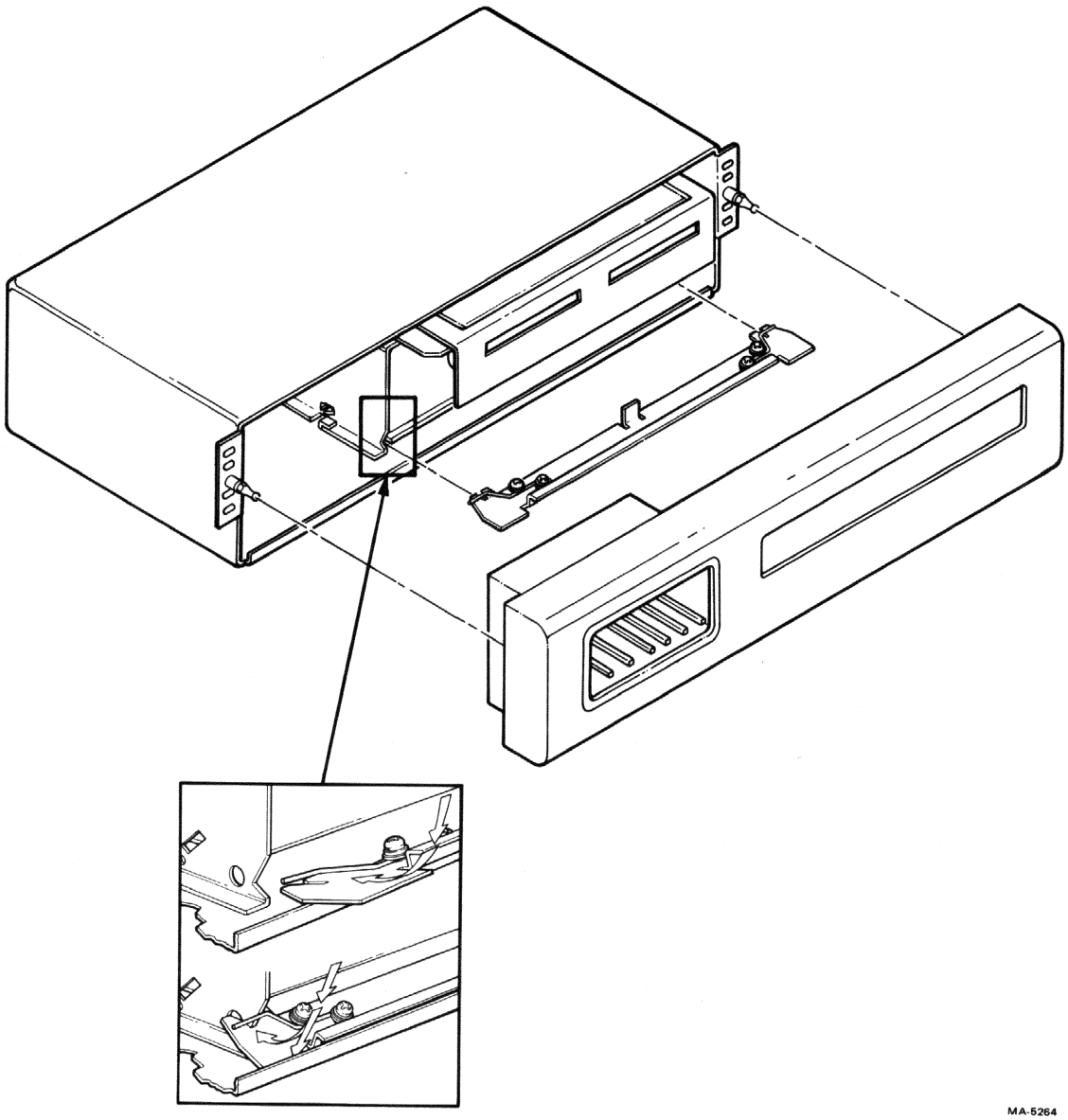


Figure 4-12 Installing Cage and Retainer Bar

3. Place the cage partially in the chassis and run the power and communication cables up to the module onto their connectors.

4. Slide the module all the way into the cage and set the cage into the hooking tabs stamped into the bottom of the chassis.

5. Align the retainer bar parallel to the floor of the chassis, with the spring on top. Engage the two slots with the vertical sheet metal of the cage at the middle of the cutaways. Press each end of the bar away and down, one at a time, so that the ends catch the lip of the chassis and the bar holds the cage in place in the chassis. The module should sit in the cage with its edge just clear of the retainer bar springs.

6. Replace the bezel and power cord.

## 4.4 INSTALLATION (-EA AND -EB VERSIONS)

The TU58-EA and -EB are tabletop units that require a minimum amount of space. Detachable line cords for 115 V and 230 V and two fuses are supplied with the TU58-EB; only the 115 V cord and fuse are supplied with the -EA. The cords are 6 ft long, enabling you to place the TU58 on a desk, tabletop, or a convenient location within reach of a power outlet. See Paragraph 4.2.3 for the correct power selection information and Paragraph 4.2.4 for controller board configuration.

### 4.4.1 Tabletop Installation

1. Disconnect the power cord and interface cable from the rear of the TU58 (Figure 4-1).
2. Place the TU58 upside-down on a flat working surface.
3. Install the four rubber feet using the four 1.3 cm (1/2 in) Phillips head screws to secure them to the bottom plates (Figure 4-13).
4. Turn the TU58 rightside-up and place it in the desired location.
5. Connect the power cord, interface cable, and cable shield wire to the rear panel.

### 4.4.2 Solid Mounting Installation

1. Perform steps 1 and 2 as above.
2. Install the four mounting brackets (bend facing the side of the TU58 housing) using the four 1.3 cm (1/2 in) Phillips head screws and lock washers to secure them to the bottom plates (Figure 4-13).
3. Turn the TU58 rightside-up and place it in the desired location.

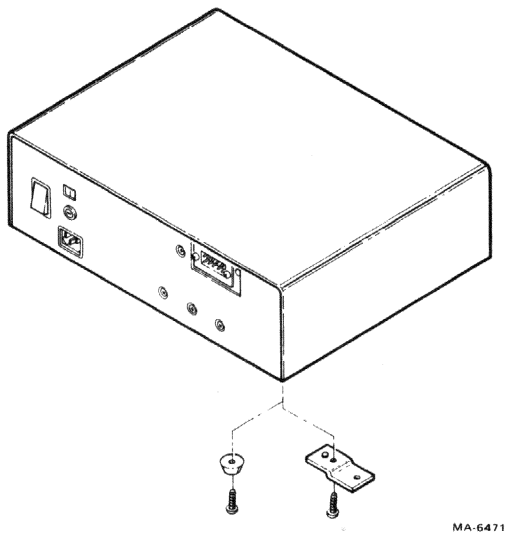


Figure 4-13 Mounting the TU58-EA and -EB

4. Fasten the unit to the mounting surface using four screws through the holes of the mounting bracket bends.

**NOTE**
**The four screws needed to secure the unit to the mounting surface are not supplied with the TU58.**

5. Connect the power cord, interface cable, and cable shield wire to the rear panel.

## 4.5 INSTALLATION (-VA VERSION)

The TU58-VA is a tabletop unit that requires a minimum amount of space and can be placed in a convenient location within reach of a dc power source. (See Paragraph 1.4.2 for power requirements.) In addition, the TU58-VA can mount to the SB11 (or BA11-VA) if so desired.

**NOTE**
**If reconfiguration is necessary, see Paragraph 4.2 before installing the TU58-VA.**

### 4.5.1 Tabletop Installation

See Paragraph 4.4.1 for installation procedure.

### 4.5.2 Solid Mounting Installation

See Paragraph 4.4.2 for installation procedure.

### 4.5.3 Mounting the TU58-VA to the SB11 (or BA11-VA)

1. Attach the four rubber feet to the TU58-VA as described in Paragraph 4.4.1, steps 1 through 4. (If solid mounting is desired, order hardware kit PN 70-16753-00).

2. Place the SB11 (or BA11-VA) upside-down on a flat working surface.

3. Remove the rubber feet from the SB11 (or BA11-VA) if attached by removing the screws securing them to the bottom. Fasten the four brackets to the bottom (bend on the outside edge) using four screws and lock washers (Figure 4-14).

4. Position the SB11 (or BA11-VA) rightside-up over the TU58-VA so the mounting brackets line up with the holes on the side of the TU58-VA. Fasten to the TU58-VA using four screws and lock washers (Figure 4-14).

5. Referring to Figure 4-15, connect the interface cables and power cord to their respective locations.

## 4.6 COMPONENTS

Figures 4-16 and 4-17 provide the mounting dimensions for the circuit board and drive mechanism. The drive has a 19 cm (7.5 in) cable which plugs into the board connector with the wires coming out of the plug toward the center of the board. The plug is keyed to ensure proper orientation. The cartridge extends 1.60 cm (0.62 in) from the front of the drive. If the drive is recessed in a panel, clearance must be provided around the opening for fingers to grip the cartridge. Ideally, the cartridge slot in a front panel is somewhat larger than minimum, to allow easy insertion. The opening should be at least the dimensions of the cartridge, 1.3 cm (0.5 in) $\times$ 8.1 cm (3.2 in), located not more than 0.53 cm (0.17 in) above the bottom mounting surface (line A in Figure 4-16). The drive must be free to float on its mounting screws, so bezels or panels must not touch the drive.

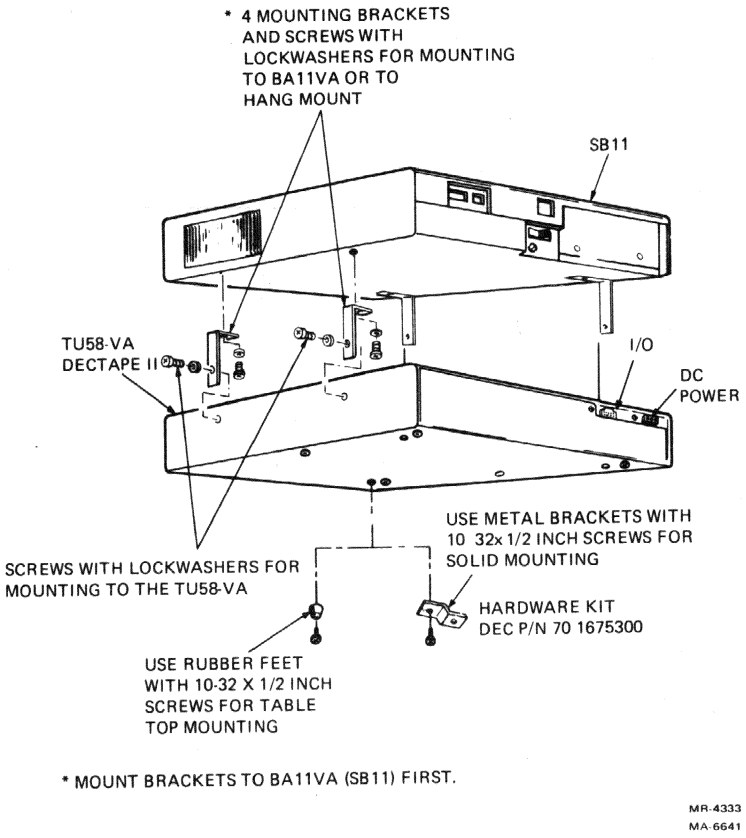


Figure 4-14 Mounting Choices for the TU58-VA

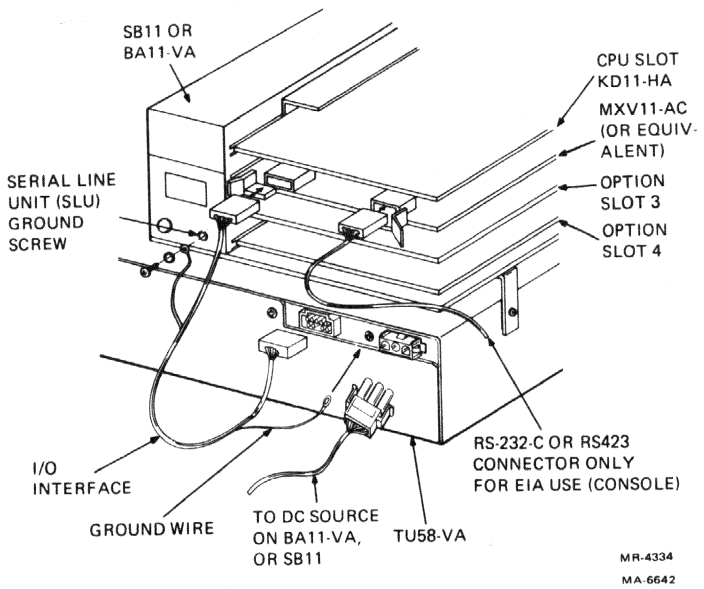


Figure 4-15 Interfacing the TU58-VA

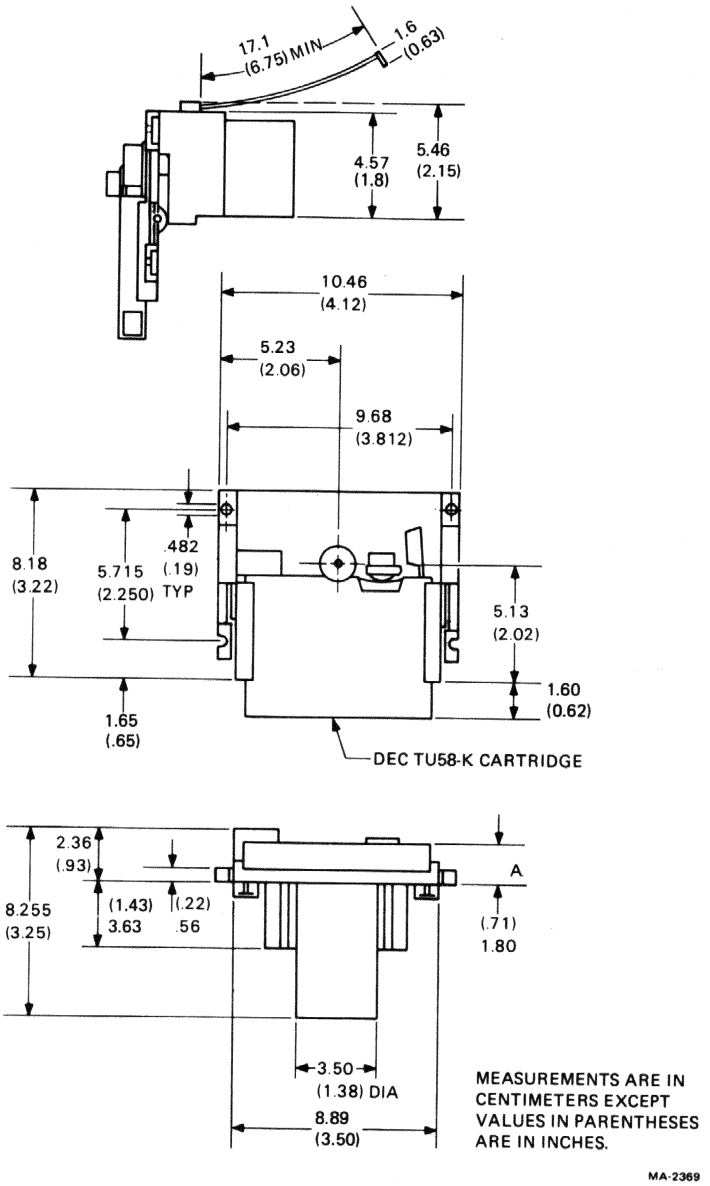


Figure 4-16 Drive Outline Drawings

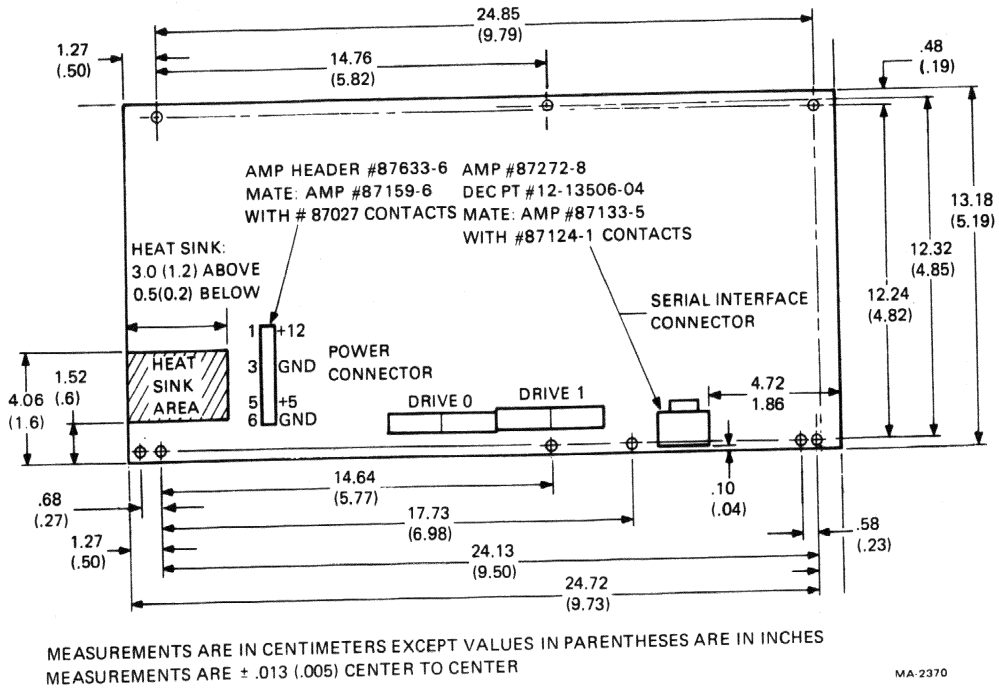


Figure 4-17 Board Outline Drawings

The board should be mounted on a flat surface with 3 mm (4-40) hardware and 1 cm (3/8 in) standoffs. Both the board and the drive may be mounted at any angle. For mounting the drives to a surface above the drives, 1.80 cm (0.71 in) clearance is required; hole spacing is given in the outline drawings. For mounting the drives to a surface below the drives, an 8.18 cm (3.22 in) × 8.89 cm (3.50 in) chassis cutout is required, with the same mounting hole spacing.

**CAUTION**

**The mounting surface for the drives must be flat within 0.64 cm (0.025 in).**

Mounting hardware is included with each drive. There is a shoulder screw, spring, and flat washer for each of the four mounting holes. Figure 4-18 shows one assembly; in addition to the flatness specification for the mounting surface, there is a specification for the depth of the shoulder screw in the mounting surface. To prevent extra compression of the spring, the shoulder screw should meet the top of the mounting surface. Any tapering of the mounting hole must be limited so that at the screw's diameter of 0.419 cm (0.165 in) the edge of the shoulder is not more than 0.076 cm (0.030 in) below surface.

## 4.7 INTERFACE STANDARDS SELECTION AND SETUP

The TU58 is shipped with factory-installed jumpers for a transmission rate of 38.4K baud and the RS-423 unbalanced line interface. A variety of standards and rates may be selected by changing the jumpers on the controller board. Table 4-1 provides a list of all the pins on the board and their functions, including the wire-wrap (WW) pins, interface, and power connectors.

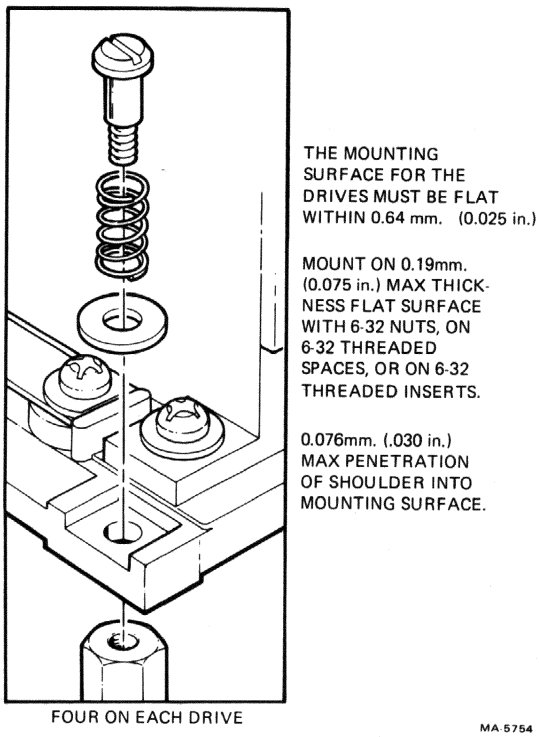


Figure 4-18 TU58 Drive Mounting Hardware

### 4.7.1 Selecting Interface Standards

The serial interface operates on full-duplex, asynchronous, 4-wire lines at rates from 150 baud to 38.4K baud. The transmit and receive rates may be independently set. Each 8-bit byte is transmitted with one start bit, one stop bit, and no parity. The line driver and receiver may be set to operate in accordance with EIA RS-422 balanced or RS-423 unbalanced signal standards. When set to RS-423, the TU58 is compatible with devices complying with RS-232-C.

The TU58 is shipped prewired for operation at 38.4K baud transmit and receive on RS-423. The maximum wire length that may be used at that data rate in an electrically quiet environment like an office is approximately 27 m (90 ft). The wire used with any installation should be no less than 24 AWG diameter.

Longer wire runs may be made if data rates are reduced. RS-422 is considerably more noise-immune than RS-423 and can be used over at least 1200 m (4000 ft) at any TU58 data rate. Figure 4-19, derived from the EIA standards, illustrates the variations in distance needed by RS-423 for different data rates. For more information, consult the standards for RS-422 and RS-423 published by the Electronic Industries Association.

**Table 4-1 TU58 Module Connections**

**Wire-Wrap Pins**

| | |
|---|---|
| WW1 | 150 Baud |
| WW2 | 300 Baud |
| WW3 | 600 Baud |
| WW4 | 1200 Baud |
| WW5 | 2400 Baud |
| WW6 | 4800 Baud |
| WW7 | 9600 Baud |
| WW8 | 19200 Baud |
| WW9 | 38400 Baud |
| WW10 | UART Receive Clock Input |
| WW11 | UART Transmit Clock Input |
| WW12 | Auxiliary A (to interface connector pin 1)* |
| WW13 | Auxiliary B (to interface connector pin 10)* |
| WW14 | Factory Test Point |
| WW15 | Ground } Connect together for auto-boot |
| WW16 | Boot } on power-up. |
| WW17 | RS-423 Driver |
| WW18 | RS-423 Common (Ground) |
| WW19 | Transmit Line+ |
| WW20 | Transmit Line− |
| WW21 | RS-422 Driver+ |
| WW22 | RS-422 Driver− |
| WW23 } | Receiver Series Resistor |
| WW24 } | (Jump for RS-422) |

**Serial Interface Connector**

| | | | |
|---|---|---|---|
| J2 – 10 | Auxiliary B | J2 – 5 | Ground |
| J2 – 9 | Ground | J2 – 4 | Transmit Line− |
| J2 – 8 | Receive Line+ | J2 – 3 | Transmit Line+ |
| J2 – 7 | Receive Line− | J2 – 2 | Ground |
| J2 – 6 | Key (no connection) | J2 – 1 | Auxiliary A |

**Power Input Connector**

| | |
|---|---|
| J1 – 1 | +12 V |
| J1 – 3 | Ground |
| J1 – 5 | +5 V |
| J1 – 6 | Ground |

**Drive Cable**

| | | | |
|---|---|---|---|
| J3,4-1 | Cart L | J3,4-9 | LED |
| J3,4-2 | No Connection | J3,4-10 | Head Shield Ground |
| J3,4-3 | Permit L | J3,4-11 | Erase Return |
| J3,4-4 | Signal Ground | J3,4-12 | Erase 1 |
| J3,4-5 | Motor+ | J3,4-13 | Erase 0 |
| J3,4-6 | Motor− | J3,4-14 | Head Return |
| J3,4-7 | +12 V | J3,4-15 | Head 0 |
| J3,4-8 | Tachometer | J3,4-16 | Head 1 |

* For optional use, such as timing signals from baud clocks.

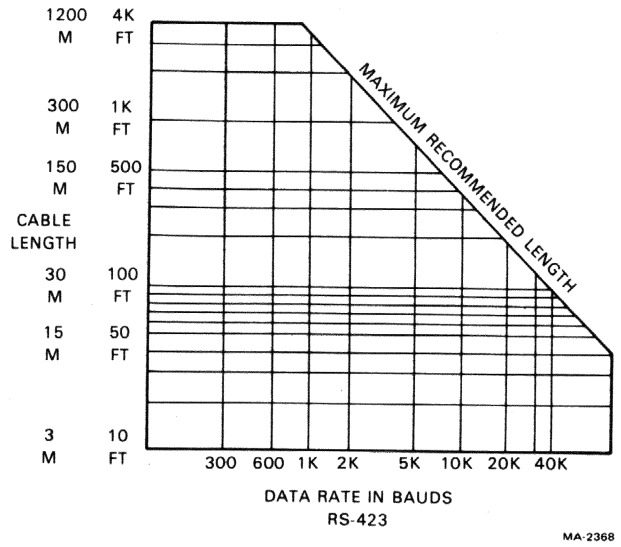


Figure 4-19 Data Rate and Cable Length for RS-423

### 4.7.2 Connecting Standards Jumpers

The jumper pins are standard 0.635 mm (0.025 in) wire-wrap posts which may be connected using 30 AWG wire and a hand tool. Other techniques that may be used include slip-on connectors, such as DIGITAL 915 patchcords, 917 daisy-chain, or soldering.

The baud rates may be set independently for transmission and reception, or both can operate together. Simply connect the pin with the desired baud value to either the XMIT or RCV pins or both. Figure 4-20 illustrates the pin locations, and Figure 4-21 the factory-wired configuration.

The interface standards may be selected by connecting sets of pins together. The connections are listed in abbreviated form in Figure 4-20. The group of pins 17 through 24 are the interface pins. The module is shipped prewired for RS-423 with pin 17 connected to pin 19, and pin 18 connected to pin 20. No other pins in the group are connected.

For RS-422, pin 21 should be tied to pin 19, pin 22 to pin 20, and pin 23 to pin 24. No other pins in the group are connected.

## 4.8 OPERATIONAL CHECKOUT

A confidence check of the operation of the newly installed TU58 may be performed through the console or keyboard console emulator of a host system without the use of an operating system device handler. The light on the TU58 board should be on, indicating a functional processor.

### 4.8.1 Checkout of Interface

To address the serial interface device registers with the console (consult the system manuals for address and codes), perform the following steps.

1. Set the transmit control status register to send Break to the TU58.
2. Remove the Break condition.
3. Transmit INIT: 04 (octal) to the TU58.
4. Transmit a second 04.
5. Examine the receive data buffer to find Continue: 20 (octal)

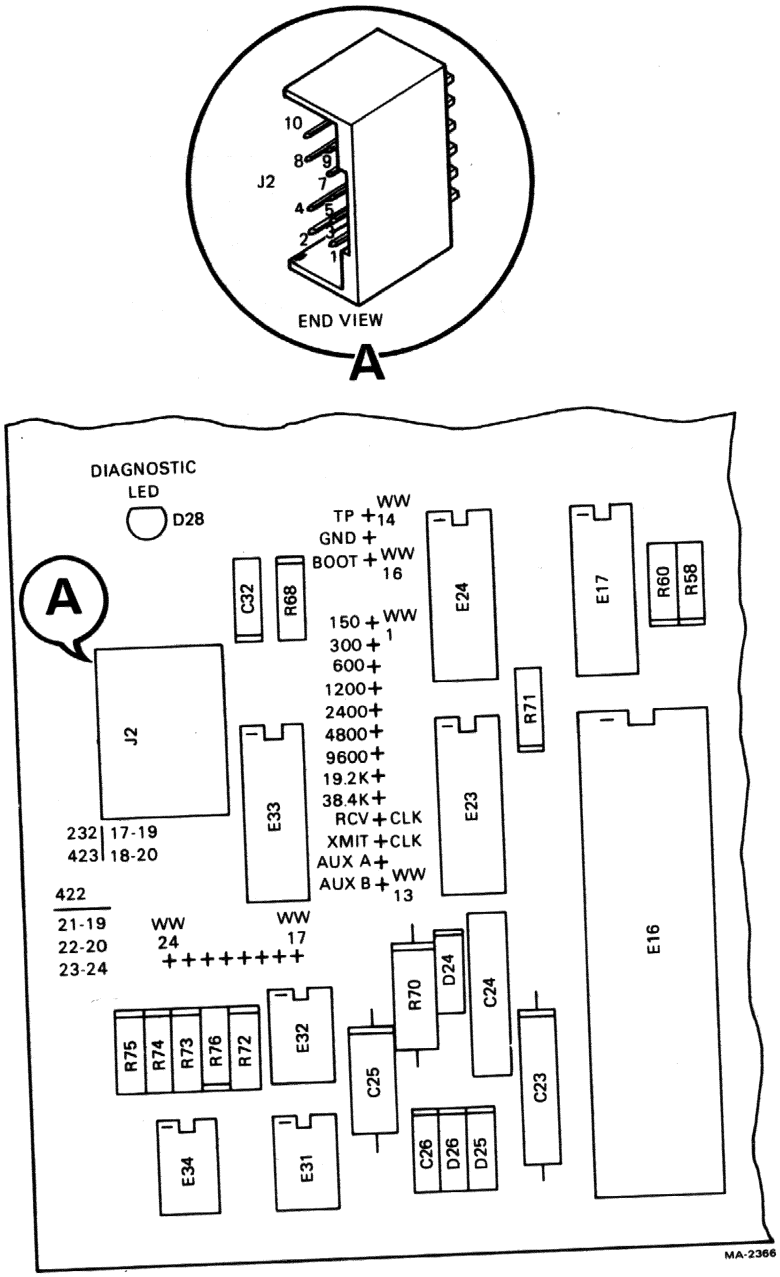


Figure 4-20 Interface Selection Jumper Pin Locations

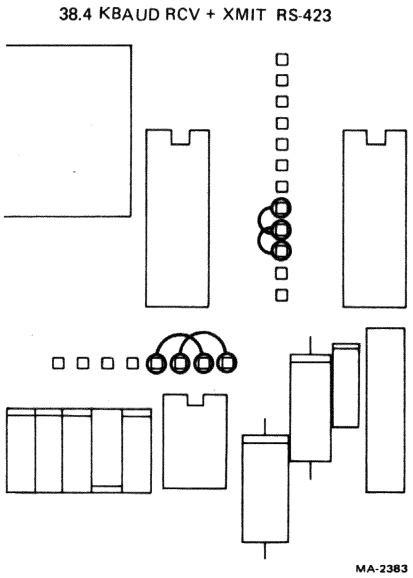


Figure 4-21 Factory Wiring

### 4.8.2 Checkout of Drive Command Function

1. Insert a tape cartridge into drive 0 (left side). (The TU58 should have sent Continue ($20_8$) already.)

2. Transmit the following string of octal numbers to the TU58. (Consult the programming chapter for an explanation of this format.)

```
2
12
2
0
0
0
0
0
0
200
200
0
204
212
```

The TU58 should wind to the beginning of the tape and read about half of the tape. If it does not work, see Table 2-1, under "TU58 does not respond to host."

# CHAPTER 5
# OPTIONS

## 5.1 RUN INDICATOR

Each tape drive may have an LED indicator which lights to show tape motion. Since data loss can occur if a cartridge is removed while the tape is being written, the cartridge should not be touched if the indicator is on.

### 5.1.1 Installation

The indicator (which may be any device capable of handling 30 mA with a forward voltage less than 1.8 V) is wired in series with the tachometer source indicator. Splice the run indicator into the wire from pin 7 of the drive connector. (Count from the end with the missing pin; that pin is number 2.) The anode should be on the board side of the wire (symbol arrow pointing away from pin 7, Figure 5-1). The indicator is available from DIGITAL (PN 11-10324), and wires with slip-on connectors are available to join the indicator to the tach (cable number 70-16526) and to extend the board connector end to the indicator at the front of the drive (cable number 70-16525).

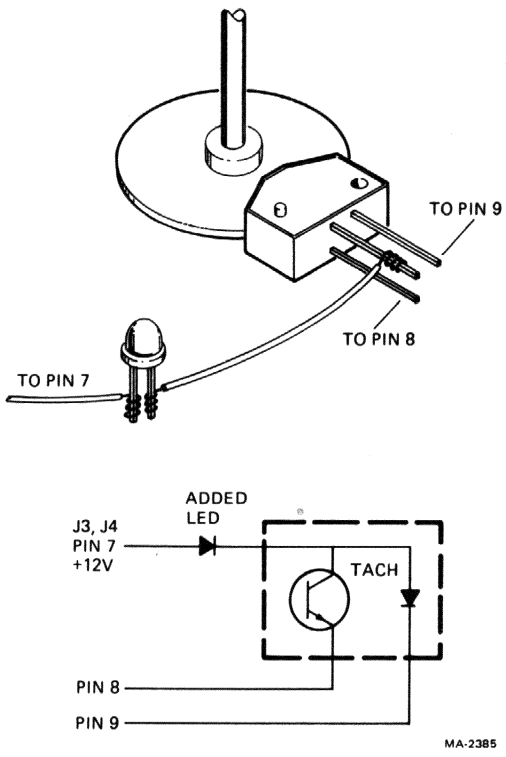


Figure 5-1 Installation of Run Indicator

## 5.2 BOOT SWITCH

### 5.2.1 General

Special provision has been made for interfacing the TU58 to the LSI-11 ODT keyboard interpreter. The TU58 is placed at the system console address and vector, permitting it to "type" in a program using keyboard ODT. This means that a keyboard cannot be connected at the same time. This arrangement is useful in an unattended control system, where the TU58 can automatically load and start or reload and restart an unsupervised process controller or similar application. The Boot switch allows a manual reboot without powering down to cycle the automatic sequence.

**NOTE**

**Boot mode does not work in any DIGITAL operating system environment.**

### 5.2.2 Operation

When the boot switch is connected according to Paragraph 5.2.3, the TU58 operates in the following manner.

1. On power-up, the TU58 checks for the presence of the closed switch. It then delays one second and begins the boot procedure.

2. When the TU58 is in the idle state, it monitors the Boot switch. Any switch contact open-close sequence causes a one-second delay (to allow for contact settling or to allow the host processor to enter the halt mode), and then the TU58 begins the boot procedure.

The boot procedure positions the tape in drive 0 to block 0, sends Break to the host, and transfers ASCII characters from the tape to the host. A delay is inserted between characters to allow for the echo from the LSI-11. If the character sent is ASCII 0 – 7, this delay is one character time at 9600 baud. Any other character is interpreted as a control character, and time is allowed for 15 characters of echo. The TU58 exits the boot mode following the transfer of the terminating character ASCII G ($147_8$) and enters the idle state. Because of the timing requirement, only rates of 9600, 19.2, and 38.4K baud may be used with boot.

### 5.2.3 Installation

The boot pin on the board (WW16) may be connected to ground through a normally closed momentary action switch. Wires may be wire-wrapped, DIGITAL 915 patch-corded, or soldered to the pins. Placement of the switch and lead dress are not critical if adequate clearance is provided around moving parts of the drive and the heat sink and power resistors on the module.

The boot tape contents are formatted to appear to the LSI-11 as output from a console (keyboard) operating under the ODT keyboard interpreter.

# APPENDIX A
# TU58/PDP-11 TOGGLE-IN BOOT

This boots drive 0 only.

```
1000/012701
1002/176500
1004/012702
1006/176504
1010/010100
1012/005212
1014/105712
1016/100376
1020/006300
1022/001005
1024/005012
1026/012700
1030/000004
1032/005761
1034/000002
1036/042700
1040/000020
1042/010062
1044/000002
1046/001362
1050/005003
1052/105711
1054/100376
1056/116123
1060/000002
1062/022703
1064/001000
1066/101371
1070/005007
```

# APPENDIX B
# RSP SEQUENCE

"NEWTAPE" RSP Sequence

```
     2
    12
     5
     0
     0
     0
     0    SEEK TO
     0    BLOCK 777
     0
     0
   377
     1
     6
    14

     2
    12
     5
     0
     0
     0
     0    SEEK TO
     0    BLOCK 0
     0
     0
     0
     0
     7
    12
```

Checksum Calculation Example

```
carry> 1
       12    2
        0    5
        0    0    Octal
        0    0    Addition
        0    0
   +    1  377
      ---  ---
       14    6
```

```
0000101000000010
0000000000000101
0000000000000000    Binary
0000000000000000    Addition
0000000000000000
+ 0000000111111111
------------------
0000110000000110
    14       6
```

# APPENDIX C
# SAMPLE DEVICE HANDLERS

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                             C
C       TTTTTTT U     U 5555555  88888                                       C
C          T    U     U 5       8     8         F O R T R A N               C
C          T    U     U 5       8     8                                     C
C          T    U     U 555555   88888          S U P P O R T               C
C          T    U     U       5 8     8                                     C
C          T    U     U       5 8     8         P A C K A G E               C
C          T    UUUUUUU 555555   88888                                       C
C                                                                             C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C       The following program listing contains a complete TU58 device
C       handler package written entirely in PDP-11 FORTRAN IV.  When
C       used with the RT-11 FORTRAN IV compiler and object-time system,
C       it can be built into ROM/PROM/EPROM based LSI-11 microcomputer
C       applications.
C
C       The program implements four user-program callable entry points:
C
C       ierror = TUREAD( unit, block, buffer, bytecount )
C
C               Read "bytecount" bytes from the TU58 drive specified by
C               unit "unit" into the data area specified by "buffer",
C               starting at random-access block "block" on the cartridge.
C
C       ierror = TUWRIT( unit, block, buffer, bytecount )
C
C               write "bytecount" bytes from the data area specified by
C               "buffer" onto TU58 drive "unit", starting at the random-
C               access block noted by "block".
C
C       ierror = TUSEEK( unit, block )
C
C               Position the TU58 cartridge located in drive "unit" at
C               the random-access block specified by "block".
C
C       ierror = TUDIAG( )
C
C               Run the TU58 internal controller diagnostic function.
C
C       In all four cases, the functions return a standard set of error
C       codes, as follows:
C
C       ierror value    meaning of error code
C
C               0       Normal success
C               1       Success, but retries were required
C               -1      TU58 failed self-test (TUDIAG)
C               -2      Partial operation (end-of-medium encountered)
C               -8      Invalid unit number was specified
C               -9      No cartridge is mounted in specified drive
C               -11     Specified cartridge is write-protected (TUWRIT)
C               -17     Data check error on cartridge
```

```
C               -32     Seek error (block not found)
C               -33     Motor stopped (TU58 hardware error)
C               -48     Invalid operation code (error in this program)
C               -55     Invalid record number (bad block number passed)
C               -127    Communications error between host and TU58
C
C       This software assumes that the TU58 controller is interfaced
C       through a DLV11, DLV11-J, DLV11-E, DLV11-F, or MXV11 interface
C       which has a receiver CSR address assignment of 176500(8).
C
C       The software operates the interface in a non-interrupt-driven
C       mode only for simplicity.
C
C       This program is neither licensed nor supported by Digital
C       Equipment Corporation, and may be copied or modified for use
C       on any computer system.  Digital assumes no responsibility for
C       its reliability on any hardware, Digital-supplied or otherwise.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
        INTEGER FUNCTION Tudiag
C
C       Runs TU58 controller diagnostic function, returning success
C       code as function result.
C
        IMPLICIT INTEGER (a-z)
        COMMON /cmdpkt/ packet(6)
C
C       Initialize a command packet with the DIAGNOSE operation code.
C
        Tudiag = Init(7)
        IF (Tudiag.ne.0) GOTO 100
C
C       Now transmit the packet to the TU58 controller.
C
        CALL Sndpkt(packet,12,0)
C
C       And ask for the results of the diagnostic in the end packet.
C
        Tudiag = Getend(0)
100     RETURN
        END


        INTEGER FUNCTION Tuseek( unit, block )
C
C       Positions cartridge in "unit" to random-access block specified
C       by "block", returning the success code from the TU58 controller
C       as the function result.
C
        IMPLICIT INTEGER (a-z)
        COMMON /cmdpkt/ packet(6)
C
C       Initialize a command packet with the POSITION operation code
C
        Tuseek = Init(5)
        IF (Tuseek.ne.0) GOTO 100
        packet(3) = unit
        packet(6) = block
C
C       Transmit the command to the TU58 controller
C
        CALL Sndpkt(packet,12,0)
C
C       And ask for the status of the operation from the end packet
C
        Tuseek = Getend(0)
100     RETURN
        END
```

```
      INTEGER FUNCTION Turead( unit, block, buffer, bytcnt )
C
C     Reads "bytcnt" bytes from the TU58 drive selected by "unit"
C     into the data area specified by "buffer", starting at random-
C     access block "block" on the cartridge.  Returns the success
C     code from the TU58 controller as the function result.
C
      IMPLICIT INTEGER (a-z)
      BYTE buffer(bytcnt), bword(2)
      COMMON /cmdpkt/ packet(6)
      EQUIVALENCE (iword,bword)
C
C     Build a READ command packet and send it to the TU58 controller
C
      Turead = Init(2)
      IF (Turead.ne.0) GOTO 500
      packet(3) = unit
      packet(5) = bytcnt
      packet(6) = block
      CALL Sndpkt(packet,12,0)
      index = 1
C
C     Read the next data packet from the TU58
C
10    chksum = Getbyt()
      IF (chksum.ne."001) GOTO 400  !Proceed to 400 if not data packet
      datcnt = Getbyt()
      chksum = chksum + datcnt*256
      odd = 0
      DO 100, i=1,datcnt
      odd = .not.odd
      IF (odd.eq.0) GOTO 50
      bword(2) = 0
      bword(1) = Getbyt()
      buffer(index) = bword(1)
      GOTO 100
50    bword(2) = Getbyt()
      chksum = Check(chksum,iword)
      buffer(index) = bword(2)
100   index = index + 1
      IF (odd.ne.0) chksum = Check(chksum,iword)
      bword(1) = Getbyt()
      bword(2) = Getbyt()
      IF (chksum.ne.iword) GOTO 500
      GOTO 10
C
C     Found packet which is not data packet; try for end packet
C
400   Turead = Getend(chksum)
500   RETURN
      END

      INTEGER FUNCTION Tuwrit( unit, block, buffer, bytcnt )
C
C     Writes "bytcnt" bytes from data area specified by "buffer"
C     to TU58 drive "unit", starting at random-access block "block".
C     Returns success code from TU58 controller as function result.
C
      IMPLICIT INTEGER (a-z)
      BYTE buffer(bytcnt), bword(2)
      COMMON /cmdpkt/ packet(6)
      EQUIVALENCE (iword,bword)
C
C     Construct a WRITE command packet and transmit it to the TU58
C
      Tuwrit = Init(3)
      IF (Tuwrit.ne.0) GOTO 500
      packet(3) = unit
      packet(5) = bytcnt
      packet(6) = block
      CALL Sndpkt(packet,12,0)
```

```
C
        datcnt = bytcnt
C
C       Prepare to transmit a new data packet by checking for CONTINUE
C       flag
C
10      chksum = Getbyt()
        IF (chksum.ne."020) GOTO 400    !Proceed to 400 if not CONTINUE
        CALL Putbyt("001)               !Transmit data packet flag
        count = Min0(datcnt,128)        !Make packets maximum 128 bytes
        CALL Putbyt(count)              !Output byte count for packet
        CALL Sndpkt(buffer(bytcnt-datcnt+1),count,count*256+"001)
        datcnt = datcnt - count
        GOTO 10
C
C       Received flag was not a CONTINUE.  Try for END packet.
C
400     Tuwrit = Getend(chksum)
C
500     RETURN
        END


        INTEGER FUNCTION Check( i, j )
C
C       Computes 16-bit checksum of i and j, using end-around carry
C       technique for TU58.  Sum is returned as function value.
C
        IMPLICIT INTEGER (a-z)
C
        check = i + j
C
C       If neither input operand has high-order bit set, no carry
C       is possible
C
        IF ( (i.or.j).ge.0 ) GOTO 200
C
C       If both inputs have high-order bit set, a carry is always
C       generated
C
        IF ( (i.and.j).lt.0 ) GOTO 100
C
C       If only one has high-order bit set, then a carry occurred
C       only if the sum does not have its high-order bit set
C
        IF ( check.lt.0 ) GOTO 200
C
C       Perform end-around carry if required
C
100     check = check + 1
200     RETURN
        END


        INTEGER FUNCTION Getbyt
C
C       Gets next byte from the serial interface connecting the
C       TU58 controller to the host machine.
C
        IMPLICIT INTEGER (a-z)
C
C       Wait for input character available (DONE flag set)
C
10      IF ( (Ipeek("176500).and."200).eq.0 ) GOTO 10
C
C       Get character from data buffer and remove extraneous bits
C
        Getbyt = Ipeek("176502).and.255
C       WRITE(6,222)GETBYT
222     FORMAT(' R',O3)
        RETURN
        END
```

```
        SUBROUTINE Putbyt( outbyt )
        BYTE OUTBYT
C
C       Writes a byte to the serial interface connecting the TU58
C       controller to the host machine.
C
        IMPLICIT INTEGER (a-z)
C
C       Wait for output interface ready (DONE bit set)
C
10      IF ( (Ipeek("176504).and."200).eq.0 ) GOTO 10
C
C       Transmit character by moving it to data buffer register
C
        CALL Ipoke("176506,outbyt)
C       WRITE(6,222)OUTBYT
222     FORMAT(' T',03)
        RETURN
        END

        INTEGER FUNCTION Init( opcode )
C
C       Initializes TU58 controller for operations.
C       Returns 0 if properly initialized, -127 otherwise.
C
        IMPLICIT INTEGER (a-z)
        COMMON /cmdpkt/ packet(6)
C
C       Initialize command packet area with opcode specified
C
        Init = 0
        packet(1) = 10*256 + "002
        packet(2) = opcode
        packet(3) = 0
        packet(4) = 0
        packet(5) = 0
        packet(6) = 0
C
C       Loop to allow eight retries of entire initialization
C       procedure
C
        DO 50, J=1,8
C
C       Set BREAK bit in output serial interface (XCSR)
C
        CALL Ipoke("176504,"000001)
C
C       Wait for BREAK to cause framing error in TU58 by
C       transmitting eight NUL characters under it
C
        DO 10, I=1,8
10      CALL Putbyt("000)
C
C       Remove BREAK condition on output interface
C
        CALL Ipoke("176504,"000000)
C
C       Discard any (possibly erroneous) character in input buffer
C
        idummy = Ipeek("176502)
C
C       Output two INIT commands to TU58 controller
C
        CALL Putbyt("004)
        CALL Putbyt("004)
C
C       Wait for & check for CONTINUE flag in response
C
        IF ( Getbyt().eq."020 ) GOTO 100
50      CONTINUE
C
```

```
C       If no success after eight retries, report error
C
        Init = -127
100     RETURN
        END

        SUBROUTINE Sndpkt(buffer,bytcnt,chkini)
C
C       Transmits a command or data packet to the TU58 controller,
C       sending "bytcnt" bytes from "buffer", followed by the updated
C       checksum.  Checksum is initialized from the "chkini" argument.
C
        IMPLICIT INTEGER (a-z)
        BYTE buffer(bytcnt), bword(2)
        EQUIVALENCE (iword,bword)
C
        chksum = chkini
        odd = 0
C
C       Loop to transmit packet contents, one byte at a time
C
        DO 100, i=1,bytcnt
        odd = .not.odd
        IF (odd.eq.0) GOTO 50
C
C       If an odd-numbered byte, remember it for checksum calculation
C
        bword(2) = 0
        bword(1) = buffer(i)
        GOTO 100
C
C       Perform checksum calculation for each byte pair on even-
C       numbered byte
C
50      bword(2) = buffer(i)
        chksum = Check(chksum,iword)
C
C       In either case, output byte to interface
C
100     CALL Putbyt(buffer(i))
        IF (odd.ne.0) chksum = Check(chksum,iword)
C
C       Output computed checksum for packet
C
        iword = chksum
        CALL Putbyt(bword(1))
        CALL Putbyt(bword(2))
        RETURN
        END
        INTEGER FUNCTION Getend(chkini)
C
C       Reads an end packet from the TU58 controller, returning
C       as the function value the success code from the packet.
C
        IMPLICIT INTEGER (a-z)
        BYTE bword(2)
        EQUIVALENCE (iword,bword)
C
        chksum = chkini
C
C       If input checksum is not zero, first byte of packet has
C       already been read by the caller.  It is the "chkini" value.
C
        IF (chksum.ne.0) GOTO 10
        chksum = Getbyt()
C
C       Check for valid END packet structure.
C
10      IF (chksum.ne."002) GOTO 500     !Must have flag = COMMAND
        IF (Getbyt().ne.10) GOTO 500     !Must have byte count = 10
        IF (Getbyt().ne."100) GOTO 500   !Must have opcode = END
```

```
C
C       If a valid packet is found, read success code byte and
C       store it
C
        Getend = Getbyt()
        chksum = Check(10*256+"102,Getend*256)
C
C       Read and discard remainder of packet, updating checksum
C
        DO 100, i=1,4
        bword(1) = Getbyt()
        bword(2) = Getbyt()
100     chksum = Check(chksum,iword)
C
C       Read transmitted checksum and compare with computed value
C
        bword(1) = Getbyt()
        bword(2) = Getbyt()
        IF (iword.eq.chksum) GOTO 600
C
C       Indicate checksum or transmission error
C
500     Getend = -127
600     RETURN
        END
```

```
        .TITLE  TU58    NON-INTERRUPT DRIVEN TU58 HANDLER
        .IDENT  /0.1/

;+
; The following program listing contains a complete TU58 device
; handler package written in PDP-11 MACRO assembly language.
;
; The program implements four FORTRAN-callable entry points:
;
;       IERROR = TUREAD( UNIT, BLOCK, BUFFER, BYTECOUNT )
;
;       Read "BYTECOUNT" bytes from the TU58 drive specified by
;       unit "UNIT" into the data area specified by "BUFFER",
;       starting at random-access block "BLOCK" on the cartridge.
;
;       IERROR = TUWRIT( UNIT, BLOCK, BUFFER, BYTECOUNT )
;
;       Write "BYTECOUNT" bytes from the data area specified by
;       "BUFFER" onto TU58 drive "UNIT", starting at the random-access
;       block noted by "BLOCK".
;
;       IERROR = TUSEEK( UNIT, BLOCK )
;
;       Position the TU58 cartridge located in drive "UNIT" at the
;       random-access block specified by "BLOCK".
;
;       IERROR = TUDIAG( )
;
;       Run the TU58 internal controller diagnostic function.
;
;
; In all cases, the functions return a standard set of status
; codes, as follows:
;
;       IERROR VALUE            MEANING OF STATUS CODE
;
;            0          Normal success
;            1          Success, but retries were required
;           -1          TU58 failed self-test (TUDIAG)
;           -2          Partial operation (end-of-medium encountered)
;           -8          Invalid unit number was specified
;           -9          No cartridge is mounted in specified drive
;          -11          Specified cartridge is write-protected (TUWRIT)
;          -17          Data check error on cartridge
;          -32          Seek error (block not found)
;          -33          Motor stopped (TU58 hardware error)
```

```
;          -48          Invalid operation code (error in this program)
;          -55          Invalid record number (bad block number passed)
;         -127          Communications error between host and TU58
;
;
; This software assumes that the TU58 controller is interfaced
; through a DLV11, DLV11-J, DLV11-E, DLV11-F, or MXV11 interface
; which has a receiver CSR address assignment of 176500(8).  These
; subroutines were written with a goal of readability: time and
; space performances are not optimal.
;
; This program is neither liscensed nor supported by DIGITAL
; Equipment Corporation, and may be copied or modified for use
; on any computer system.  Digital assumes no responsibility
; for its reliability on any hardware, Digital-supplied or
; otherwise.
;
;-
        .SBTTL  Symbol Definitions and Data Area

; Define packet flag byte codes
;
        F.DATA  =       1               ; Data packet
        F.CTRL  =       2               ; Control packet
        F.INIT  =       4               ; INIT packet
        F.CONT  =       20              ; CONTINUE packet
        F.XOFF  =       23              ; XOFF packet

; Define Control packet op-codes
;
        O.READ  =       2               ; perform read operation
        O.WRIT  =       3               ; perform write operation
        O.POS   =       5               ; perform seek operation
        O.DIAG  =       7               ; perform self-test
        O.END   =       100             ; packet is an End packet

; Define controller interface address assignment
;
        DLRCSR  =       176500
        DLRBUF  =       DLRCSR+2
        DLXCSR  =       DLRCSR+4
        DLXBUF  =       DLRCSR+6

; Macro to send a byte to TU58
;
        .MACRO  PUTBYT  ARG, ?LAB
LAB:    TSTB    @#DLXCSR
        BPL     LAB
        MOVB    ARG, @#DLXBUF
        .ENDM   PUTBYT

; Macro to wait for a byte from TU58
; NOTE: If ARG is a register, the parity bit will be sign-extended
; into the high-byte of the register.
;
        .MACRO  GETBYT  ARG, ?LAB
LAB:    TSTB    @#DLRCSR
        BPL     LAB
        MOVB    @#DLRBUF, ARG
        .ENDM GETBYT

; Data area
;
        .PSECT  USERSD  RW, D, LCL, REL, CON

; Area used to build Control packets
;
PACKET: .BYTE   F.CTRL, 10.             ; Control packet, length = 10.
        .WORD   0, 0, 0, 0, 0

; Word used to collect byte-pairs for 16-bit checksum calculation
```

```
;
WORD:   .BLKW

; Pure (ROM-able) code follows
;
        .PSECT  USER$I  RW, I, LCL, REL, CON

        .SBTTL  TUREAD  Read Data From TU58

;+
; TUREAD
;
; Description:
;       See module heading
;
; Inputs:
;       R5 points to a five word standard FORTRAN argument block
;       0(R5) = 4
;       2(R5) = address of unit number byte
;       4(R5) = address of block number word
;       6(R5) = address of callers input buffer
;       8.(R5) = address of bytecount word
;
; Outputs:
;       R0 = 16-bit status code
;-

TUREAD::
        CALL    INIT                    ; initialize the TU58
        TST     R0                      ; if error occured,
        BNE     50$                     ; return error to caller

; Build a Control packet
;
        MOVB    #0.READ, PACKET+2       ; operation is "READ"
        MOVB    @2(R5), PACKET+4        ; store unit number
        MOV     @8.(R5), PACKET+8.      ; store byte count
        MOV     @4(R5), PACKET+10.      ; store block number

; Send the Control packet to the TU58
;
        MOV     #PACKET, R0             ; R0 points to packet
        MOV     #12., R1                ; R1 = number of bytes to send
        CLR     R2                      ; R2 = initial checksum
        CALL    SNDPKT                  ; send the Control packet

; Receive zero or more Data packets from TU58, followed by an End packet.
;
        MOV     6(R5), R1               ; R1 = address of callers buffer
5$:     GETBYT  R0                      ; get flag byte from TU58
        CMPB    R0, #F.DATA             ; if not a Data packet,
        BNE     90$                     ; check for End packet
        GETBYT  R2                      ; get byte count from TU58
        BIC     #^C<377>, R2            ; clear hi-byte
        SWAB    R2                      ; form initial checksum in R0
        BIS     R2, R0                  ; R0 = current checksum
        SWAB    R2                      ; restore byte count
        CLR     R3                      ; R3 is even/odd byte flip-flop
; Receive all characters in the Data packet, updating checksum as
; each pair of characters is received.
;
10$:    COM     R3                      ; flip flag
        BEQ     20$                     ; branch if second character of pair
        CLR     WORD                    ; build character pair
        GETBYT  WORD                    ; with next data character
        MOVB    WORD, (R1)+             ; store data byte in callers buffer
        BR      30$

20$:    GETBYT  WORD+1                  ; get second byte of pair
        ADD     WORD, R0                ; update checksum
        ADC     R0                      ; with end-around carry
        MOVB    WORD+1, (R1)+           ; store data byte in callers buffer
```

```
30$:    SOB     R2, 10$                 ; loop until byte count is zero

        TST     R3                      ; if odd number of data bytes in
; packet,
;
        BEQ     40$
        ADD     WORD, R0                ; finish checksum
        ADC     R0

; Receive checksum from TU58 and compare to computed value
;
40$:    GETBYT  WORD                    ; get 16-bit checksum
        GETBYT  WORD+1
        CMP     WORD, R0                ; compare with computed checksum
        BEQ     5$                      ; if the same, get next Data packet

; Checksum error.  Return error code.
;
        MOV     #-127., R0              ; set function value to error code
50$:    RETURN                          ; and return to caller

; A packet was received from the TU58 that was not a Data packet.
; Check to see that it is an End packet.
;
90$:    JMP     GETEND                  ; check End packet and return to
;caller
;
        .SBTTL  TUWRIT  Send Data to TU58

;+
; TUWRIT
;
; Description:
;       See module heading
;
; Inputs:
;       R5 = address of a five word standard FORTRAN argument block
;       0(R5) = 4
;       2(R5) = address of unit number byte
;       4(R5) = address of block number word
;       6(R5) = address of output buffer
;       8.(R5) = address of bytecount word
;
; Outputs:
;       R0 = 16-bit status code
;-

TUWRIT::
        CALL    INIT                    ; initialize the TU58
        TST     R0                      ; if unsuccessful,
        BNE     99$                     ; return to caller

; Build a Control packet to send to the TU58.
;
        MOVB    #0.WRIT, PACKET+2       ; op-code is "WRITE"
        MOVB    @2(R5), PACKET+4        ; set unit number
        MOV     @8.(R5), PACKET+8.      ; set byte count
        MOV     @4(R5), PACKET+10.      ; set block number

; Send the Control packet to the TU58.
;
        MOV     #PACKET, R0             ; R0 = address of packet
        MOV     #12., R1                ; R1 = number of bytes to send
        CLR     R2                      ; R2 = initial checksum
        CALL    SNDPKT                  ; send the packet

        MOV     @8.(R5), R3             ; R3 = number of bytes left to send
        MOV     6(R5), R5               ; R5 = address of callers data buffer
```

```
; Send one or more Data packets to the TU58.
;
10$:    GETBYT  R0                      ; get a byte from TU58
        CMPB    R0, #F.CONT             ; if not CONTINUE,
        BNE     90$                     ; check for End packet
        PUTBYT  #F.DATA                 ; send Data packet flag
        MOV     #128., R4               ; assume 128 or more bytes left
; to send
;
        CMP     R3, R4                  ; if less than 128. bytes left
; to send,
;
        BHIS    20$
        MOV     R3, R4                  ; RESET R4 to actual number left
20$:    PUTBYT  R4                      ; send byte count
        MOV     R5, R0                  ; R0 = address of packet to send
        MOV     R4, R1                  ; R1 = length of packet
        MOV     R4, R2
        SWAB    R2
        ADD     #F.DATA, R2             ; R2 = initial checksum
        CALL    SNDPKT                  ; send the packet to TU58
        SUB     R4, R3                  ; update number of bytes left to send
        ADD     R4, R5                  ; update data buffer pointer
        BR      10$                     ; wait for acknowledgment from TU58

; A byte was received from the TU58 that was not a CONTINUE.
; See if the packet is an End packet.
;
90$:    CALL    GETEND
99$:    RETURN                          ; return to caller

        .SBTTL  TUSEEK  Position the TU58

;+
; TUSEEK
;
; Description:
;
;       See module heading
;
; Inputs:
;       R5 = address of three word standard FORTRAN argument block
;       0(R5) = 2
;       2(R5) = address of unit number byte
;       4(R5) = address of block number word
;
; Outputs:
;       R0 = 16-bit status code
;-

TUSEEK::
        CALL    INIT                    ; initialize the TU58
        TST     R0                      ; if initialization failed,
        BNE     99$                     ; return error to caller

; Build Control packet to send to TU58.
;
        MOVB    #O.POS, PACKET+2        ; set op-code
        MOVB    @2(R5), PACKET+4        ; set unit number
        MOV     @4(R5), PACKET+10.      ; set block number

; Send the Control packet to the TU58.
;
        MOV     #PACKET, R0             ; R0 points to packet
        MOV     #12., R1                ; R2 = length of packet
        CLR     R2                      ; R2 = initial checksum
        CALL    SNDPKT                  ; send the packet

; Get an End packet from the TU58 and return to caller.
;
        CLR     R0                      ; indicate no byte pre-read
        CALL    GETEND
99$:    RETURN                          ; return to caller
```

```
        .SBTTL  TUDIAG  Run TU58 Local Diagnostic

;+
; TUDIAG
;
; Description:
;       See module heading
;
; Inputs:
;       none
;
; Outputs:
;       R0 = 16-bit status code
;-

TUDIAG::
        CALL    INIT                    ; initialize the TU58
        TST     R0                      ; if initialization failed,
        BNE     99$                     ; return error code to caller

; Build Control packet.
;
        MOVB    #0.DIAG, PACKET+2       ; set op-code

; Send the Control packet to the TU58.
;
        MOV     #PACKET, R0             ; R0 = address of packet
        MOV     #12., R1                ; R1 = length of packet
        CLR     R2                      ; R2 = initial checksum
        CALL    SNDPKT                  ; send the packet

; Receive an End packet and return status code to caller.
;
        CLR     R0                      ; indicate no byte pre-read
        CALL    GETEND                  ; get the End packet
99$:    RETURN                          ; return to caller

        .SBTTL  INIT    Initialize the TU58

;+
; INIT
;
; Description:
;       INIT is called by TUREAD, TUWRIT, TUSEEK, and TUDIAG
;       to initialize the TU58 to its power-up state.
;       This is done by causing a framing error in the TU58,
;       followed by sending two INIT packets to the TU58.
;       The TU58 should reply with a CONTINUE packet.
;
; Inputs:
;       none
;
; Outputs:
;       R0 = 16-bit status code
;-

INIT:
        CLR     R0                      ; assume success
        MOV     #8., R1                 ; retry eight times before failure

; Send BREAK.
;
10$:    MOV     #1, @#DLXCSR            ; start sending a break
        MOV     #8., R2                 ; send nulls for 8 character times
20$:    PUTBYT  #0
        SOB     R2, 20$
        CLR     @#DLXCSR                ; stop sending the break
        TSTB    @#DLRBUF                ; remove any extraneous input byte
```

```
; Send two INIT packets.
;
        PUTBYT  #F.INIT                 ; send two INIT packets
        PUTBYT  #F.INIT

; See if TU58 sent a Continue packet
;
        GETBYT  R2                      ; get a byte from TU58
        CMPB    R2, #F.CONT             ; if CONTINUE packet,
        BEQ     99$                     ; return to caller
        SOB     R1, 10$                 ; else retry

; Retry failed.  Return error status to caller.
;
        MOV     #-127., R0
99$:    RETURN


        .SBTTL  SNDPKT  Send a Packet to TU58

;+
; SNDPKT
;
; Description
;       SNDPKT is called by TUREAD, TUWRIT, TUSEEK, and TUDIAG to
;       send a multi-byte packet (either Control or Data) to the TU58.
;       SNDPKT will also compute the packets checksum and send the
;       checksum to the TU58.
;
; Inputs:
;       R0 = address of bytes to send
;       R1 = number of bytes to send
;       R2 = the initial checksum (non-zero if part of the packet has
;            already been sent)
;
; Outputs:
;       None
;-

SNDPKT:
        CLR     -(SP)                   ; make odd/even byte flip-flop

; Send bytes to TU58.  Update 16-bit checksum after each byte pair.
;
10$:    COM     (SP)                    ; flip odd/even flag
        BEQ     20$                     ; branch if second byte of pair
        CLR     WORD                    ; set up byte pair word
        MOVB    (R0), WORD              ; with first byte of pair
        BR      30$

20$:    MOVB    (R0), WORD+1            ; store second byte of pair
        ADD     WORD, R2                ; update checksum
        ADC     R2                      ; with end-around carry
30$:    PUTBYT  (R0)+                   ; send the data byte
        SOB     R1, 10$                 ; loop until byte count is zero

        TST     (SP)+                   ; if odd number of data bytes,
        BEQ     40$
        ADD     WORD, R2                ; update computed checksum
        ADC     R2
40$:    PUTBYT  R2                      ; send 16-bit checksum
        SWAB    R2
        PUTBYT  R2
        RETURN                          ; return to caller

        .SBTTL  GETEND  Check for End Packet

;+
; GETEND
;
; Description:
;       GETEND is called by TUREAD, TUWRIT, TUSEEK, and TUDIAG to accept
;       an End Control packet from the TU58.  GETEND verifies that the
```

```
;       received packet is indeed an End packet, and that the checksum
;       word in the packet equals the computed checksum.
;
; Inputs:
;       R0 = a byte already read by the caller or zero
;
; Outputs:
;       R0 = 16-bit status code
;-

GETEND:

; If flag byte was not pre-read by caller, read it now.
;
        TSTB    R0                      ; if not pre-read,
        BNE     10$
        GETBYT  R0                      ; read it
10$:

; Verify that the packet coming from the TU58 is indeed an End Control packet.
;
        CMPB    R0, #F.CTRL             ; check Control packet
        BNE     90$
        GETBYT  R0
        CMPB    R0, #10.                ; with length = 10.
        BNE     90$
        GETBYT  R0
        CMPB    R0, #O.END              ; and op-code = End
        BNE     90$
        GETBYT  R0                      ; get success code
        BIC     #^C<377>, R0            ; clear hi-byte
        MOV     R0, R1                  ; form 16-bit checksum in R1
        SWAB    R1                      ; form 16-bit checksum
        ADD     #10.*256.+O.END+F.CTRL, R1
        ADC     R1                      ; from bytes already received

; Receive and ignore rest of End packet, updating checksum.
;
        MOV     #4, R2
20$:    GETBYT  WORD
        GETBYT  WORD+1
        ADD     WORD, R1
        ADC     R1
        SOB     R2, 20$                 ; loop four times to get 8. bytes

; Receive checksum transmitted from TU58 and compare with computed checksum.
;
        GETBYT  WORD
        GETBYT  WORD+1
        CMP     WORD, R1
        BEQ     100$

; Return error code for transmission error.
;
90$:    MOVB    #-127., R0
100$:   MOVB    R0, R0                  ; return status as 16-bit value
        RETURN                          ; return to caller


        .END
```

# APPENDIX D
# CARTRIDGE REPAIR

## D.1 INTRODUCTION

Under unusual circumstances of controller failure or cartridge mishandling, the tape might come free of the hub. The tape is not fastened to the hub but is held in place by the elastomer belt and by the tape's wrap around itself. The procedures for looping the tape back onto the hub are given here to help the user prevent the loss of important data. They are not a substitute for the customary precautions of proper handling and backup copying. Two procedures are given here. One is for the metal-base cartridge and the other is for the plastic-base cartridge.

These are moderately difficult procedures requiring the use of small tools. Minimum tools are a number 1 Phillips head screwdriver and a small probe (a straightened paper clip can be used). Tweezers are helpful.

**NOTE**

**Keep magnetized tools away from the bulk of the tape and do not touch the tape surface except at the ends because fingerprints cause errors. (If staples or paper clips stick to a tool, it is magnetized.)**

## D.2 METAL-BASE CARTRIDGE

1. Open the cartridge by removing the four baseplate Phillips head screws (Figure D-1) and set it upright on the work surface with the cover still on.

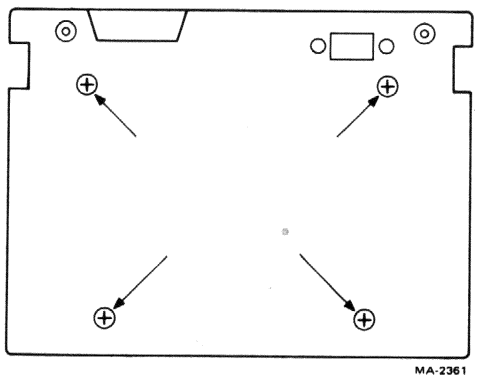


Figure D-1 Baseplate Screw Locations

2. Lift the cover off.

**NOTE**

**To remove the head gate, swing it out to clear the tape before lifting it up. Its replacement is optional.**

**A spring is in the bottom of the gate. (Figure D-3).**

3. Thread the end of the tape around the tape guides (Figure D-2).

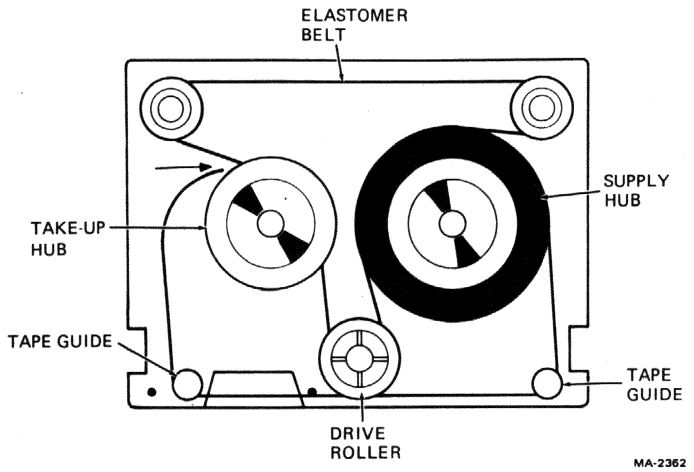


Figure D-2 Threading the Metal-Base Cartridge

4. Moisten the end of the tape with water to get it to stick to the hub.

5. With a small amount of slack at the free end, insert the end between the hub and belt and operate the drive roller with a finger to take up the tape. As soon as the tape is grabbed, keep some back tension on the tape. This keeps it feeding straight into the hub.

6. Continue to wind. Watch for the loose ends as it comes around. If it separates from the hub, tuck it under the next turn of tape with the probe. (Back up if the end is too long.)

7. Continue to wind a few more turns with the drive roller while applying tension to the tape.

8. Hold the takeup hub and drive roller fixed, and rotate the supply hub to take up the slack.

9. Continue winding the tape about 20 turns before reassembling.

10. To reassemble the cartridge, reinstall the gate (if desired) by aligning the long and short ends of the spring with the long and short ends of the gate, as in Figure D-3.

11. Drop the spring into the well in the gate. Holding the spring down with a thumbnail or probe, rotate the long end of the spring around to the slot that is at a right angle to the long dimension of the gate. Push the end of the spring into the slot; it should stay there by itself.

12. Hold the gate halfway out so that the gate and the spring end do not touch the tape. Slowly press the gate down onto its pin on the cartridge baseplate. Reach in with the probe and press the spring down. It will clear its holding slot and snap into position, closing the gate.

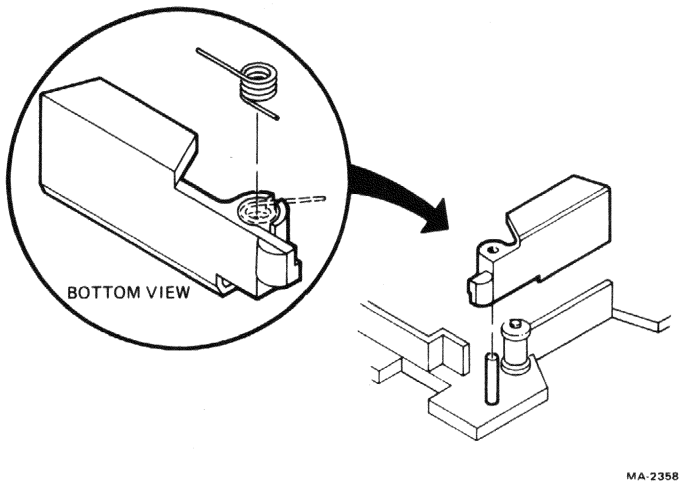


Figure D-3 Head Gate and Spring

13. Carefully lower the cartridge cover into place and reinstall the screws.

## D.3 PLASTIC-BASE CARTRIDGE

Open the plastic-base cartridge case by removing the four baseplate Phillips head screws (Figure D-1). Carefully remove the top.

### D.3.1 Preparation for Threading

The four rollers and tape hubs in the plastic-base cartridge are held in their operating plane by the top and bottom of the case together. When the top is off, the various parts tend to creep out of position, and the elastomer belt can get folded under the hubs.

1. To organize the parts for threading, remove and discard the head gate and spring. Take the empty tape hub from the case and set it aside.

2. Remove the floating roller (Figure D-4).

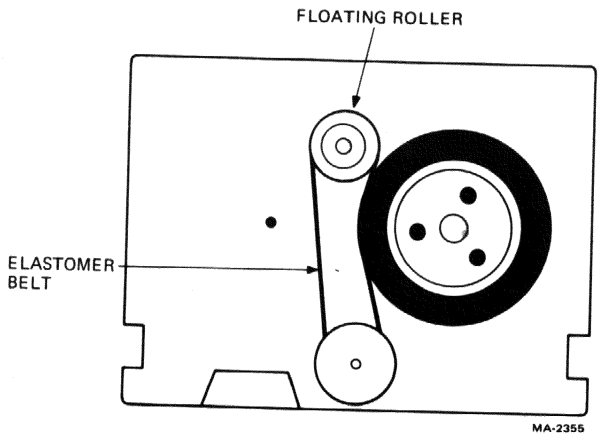


Figure D-4 Stretch the Belt with the Floating Roller

3. Rearrange the elastomer belt around the drive roller and the supply hub.

4. Put the takeup hub on its pin.

5. Put the empty tape hub on its pin.

6. Using the top to hold the floating roller, belt and supply hub down, use a straightened paper clip or pencil to guide the elastomer belt around the hub. The hub should seat against the base with the belt around it.

### D.3.2 Threading the Cartridge

1. Pull several centimeters (a few inches) of tape off the supply hub and through the tape guides (Figure D-5).

**NOTE**

**Hold all parts down when moving them. Otherwise, the hubs will creep up the pins and cause the belt to slip. Then the procedure must be restarted at Paragraph D.3.1.**

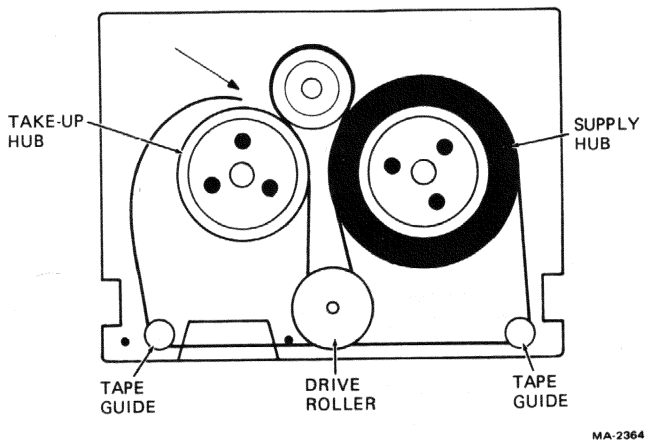


Figure D-5 Threading the Plastic-Base Cartridge

2. Moisten the end of the tape with water to get it to stick to the hub.

3. With a small amount of slack at the free end, insert the end between the hub and belt, and operate the floating roller to take up the tape.

4. As soon as the tape is grabbed, keep some back tension on the tape. This keeps the tape feeding straight into the hub.

5. Continue to wind. Watch for the loose end as it comes around. If it separates from the hub, tuck it under the next turn of tape with the paper clip. (Back up if the end is too long.)

6. Continue to wind a few more turns with the floating roller while applying tension to the tape.

7. Now hold the takeup hub, drive roller, and floating rollers fixed and rotate the supply hub to take up the slack.

### D.3.3 Closing the Cartridge

Place the top back on the cartridge. Do not reinstall the head gate. The mirror window may need to be pressed in slightly to clear the bottom. Reinstall the four baseplate screws.

Now use a finger to operate the drive roller and wind the tape about 20 turns onto the takeup hub before inserting the cartridge into a drive.

**NOTE**

**The only reason for performing this exercise is to copy the data from the injured tape as soon as possible. Discard the cartridge after copying.**

# APPENDIX E
# FIELD REPLACEABLE UNIT SPARES LIST

| Module | DIGITAL P/N | Option Name |
|---|---|---|
| Serial Controller Board | 54-13489 | TU58 – XB |
| Regulator Module | 54-13609 | |
| Drive | 70-15510 | TU58 – XA |
| Tachometer Encoder Wheel | 74-20649 | |
| Tape Cartridge | 36-15809 | TU58 – K |

**TU58 DECtape USER GUIDE**

**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?

| | Excellent | Very Good | Good | Fair | Poor |
|---|---|---|---|---|---|
| Accuracy | ☐ | ☐ | ☐ | ☐ | ☐ |
| Completeness | ☐ | ☐ | ☐ | ☐ | ☐ |
| Organization | ☐ | ☐ | ☐ | ☐ | ☐ |
| Format | ☐ | ☐ | ☐ | ☐ | ☐ |

What features are most useful?
(Notes, Tables, Illustrations, etc.) _____

_____

_____

_____

_____

Does the publication satisfy your needs? ☐ Yes ☐ No

What errors have you found?
(Ref. page no., table no., figure no.) _____

_____

_____

_____

_____

Additional Comments _____

_____

_____

Name _____ Street _____
Title _____ City _____
Company _____ State/Country _____
Department _____ Zip _____

The Documentation Products Directory and the DECdirect Guide contain information on the remainder of DIGITAL's technical documentation.

Copies of the above Directory and Guide, as well as additional copies of this document, are available by writing or calling:

Digital Equipment Corporation
Accessories and Supplies Group
P.O. Box CS2008
Nashua, New Hampshire 03061

Attention: Documentation Products
Telephone: 1-800-258-1710

Order No. **EK-0TU58-UG-003**

MY

Fold Here

Do Not Tear – Fold Here and Staple

digital

No Postage
Necessary
if Mailed in the
United States

**BUSINESS REPLY MAIL**

FIRST CLASS PERMIT NO. 33 MAYNARD, MA.

POSTAGE WILL BE PAID BY ADDRESSEE

Digital Equipment Corporation
Educational Services Development and Publishing
129 Parker Street, PK3-1/T12
Maynard, MA 01754

Digital Equipment Corporation • Maynard, MA 01754