List of Software Versions

BOOT	1.2
FDOS	1.3
CONMON	1.1
BASIC	1.2
FUP	1.1
SET	2.0
TIME	1.0

NOTICE

This manual documents this set of software versions only. Software modules will function properly only in compatible versions. Refer to the software compatibility check procedure at the end of the installation section for additional information.

1720A

Instrument Controller

User Manual

P/N 518654
April 1980
Rev 1 11/80
Rev 2 11/81

FLUKE®

WARRANTY

Notwithstanding any provision of any agreement the following warranty is exclusive:

The JOHN FLUKE MFG. CO., INC., warrants each instrument controller it manufactures to be free from defects in material and workmanship under normal use and service for the period of 90 days from date of purchase. This warranty extends only to the original purchaser. This warranty shall not apply to fuses, floppy disks or contents thereof, or any product or parts which have been subject to misuse, neglect, accident or abnormal conditions of operations.

In the event of failure of a product covered by this warranty, John Fluke Mfg. Co., Inc. will repair an instrument controller returned to an authorized Service Facility within 90 days of the original purchase; provided the warrantor's examination discloses to its satisfaction that the product was defective. The warrantor may, at its option, replace the product in lieu of repair. With regard to any instrument controller returned within 90 days of the original purchase, said repairs or replacement will be made without charge. If the failure has been caused by misuse, neglect, accident or abnormal conditions of operations, repairs will be billed at a nominal cost. In such case, an estimate will be submitted before work is started if requested.

THE FOREGOING WARRANTY IS IN LIEU OF ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS OR ADEQUACY FOR ANY PARTICULAR PURPOSE OR USE. JOHN FLUKE MFG. CO., INC., SHALL NOT BE LIABLE FOR ANY SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES, WHETHER IN CONTRACT, TORT OR OTHERWISE.

If any failure occurs, the following steps should be taken:

1. Notify the JOHN FLUKE MFG. CO., INC., or the nearest Service Facility, giving full details of the difficulty, and include the model number, type number, and the serial number. On receipt of this information, service data or shipping instructions will be forwarded to you.

2. On receipt of the shipping instructions, forward the instrument controller transportation prepaid. Repairs will be made at the Service Facility and the instrument controller returned, transportation prepaid.

SHIPPING TO MANUFACTURER FOR REPAIR OR ADJUSTMENT

All shipments of JOHN FLUKE MFG. CO., INC., instrument controllers should be made via United Parcel Service or "Best Way" prepaid. The instrument controller should be shipped in the original packing carton. If this original carton is not available a suitable container that is rigid and of adequate size may be used. However, the JOHN FLUKE MFG. CO., INC., does not recommend shipment of instrument controllers in substitute containers. In the event a substitute container must be used, the instrument controller should be wrapped in paper and surrounded with at least four inches of excelsior or other similar shock absorbing material. JOHN FLUKE MFG. CO., INC., shall assume NO risk for intransit shipment damage.

CLAIM FOR DAMAGE IN SHIPMENT TO ORIGINAL PURCHASER

The instrument controller should be thoroughly inspected immediately upon original delivery to purchaser. All material in the container should be checked against the enclosed packing list. The manufacturer will not be responsible for shortages against the packing sheet unless notified immediately. If the instrument controller is damaged in any way, a claim should be filed with the carrier immediately. (To obtain a quotation to repair shipment damage, contact the nearest Fluke Technical Center.) Final claim and negotiations with the carrier must be completed by the customer.

The JOHN FLUKE MFG. CO. INC., will be happy to answer all applications or use questions, which will enhance your use of this instrument. Please address your requests or correspondence to: JOHN FLUKE MFG. CO. INC., P.O. BOX C9090, EVERETT, WASHINGTON 98206, ATTN: Sales Dept. For European Customers: Fluke (Holland) B.V., P.O. Box 5053, 5004 EB, Tilburg, The Netherlands. *

*For European customers, Air Freight prepaid.

Table of Contents

(continued on page ii)

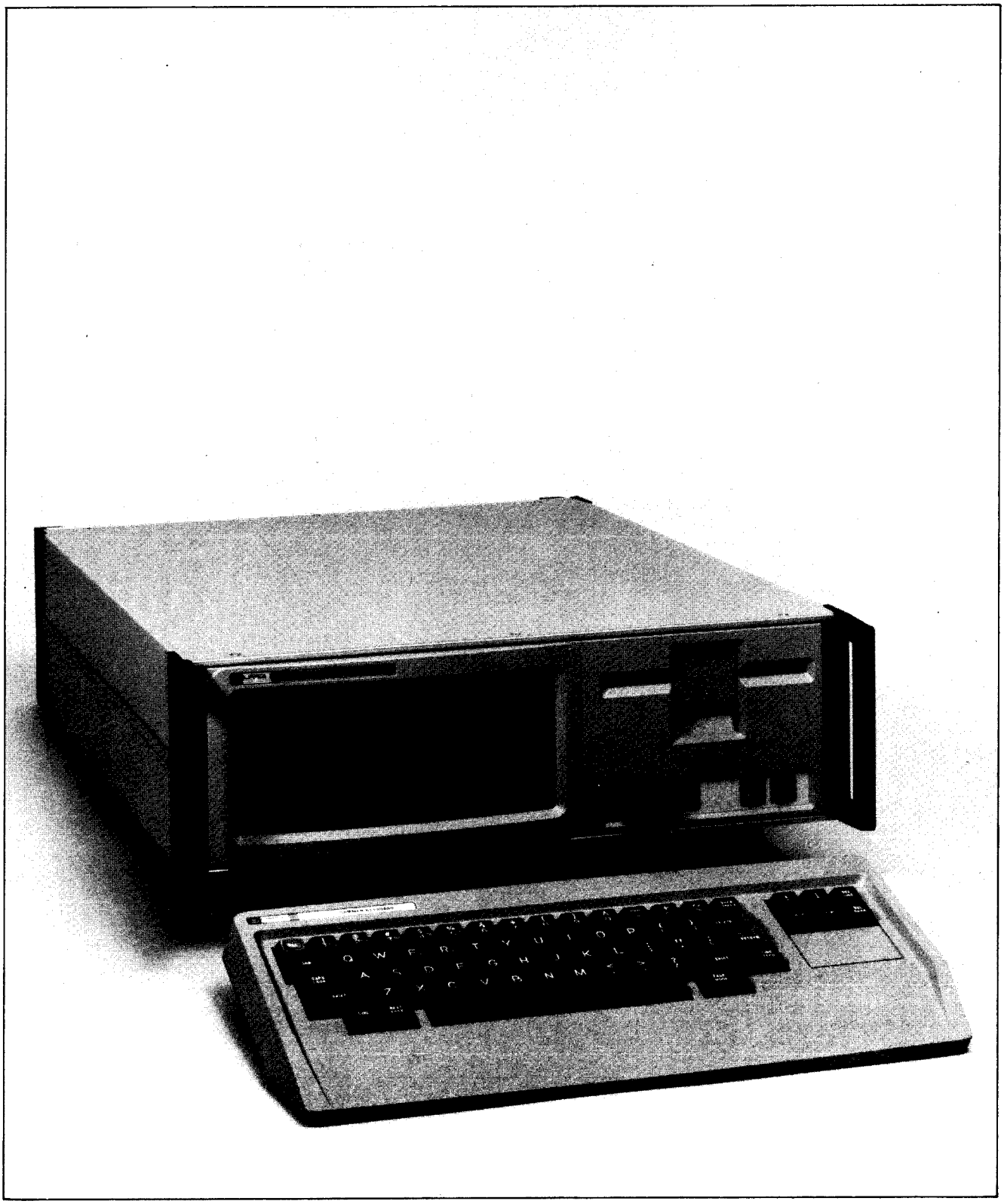
TABLE OF CONTENTS, *continued*

List of Tables

List of Illustrations

Note

This equipment generates, uses, and can radiate radio frequency energy and if not installed and used in accordance with this User Manual, may cause interference to radio communications. As temporarily permitted by regulation it has not been tested for compliance with the limits for Class A computing devices pursuant to Subpart J Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference. Operation of this equipment in a residential area is likely to cause interference in which case the user at his own expense will be required to take whatever measures may be required to correct the interference.



1720A Instrument Controller

Section 1
Introduction

1-1. DESCRIPTION OF 1720A DOCUMENTATION

1-2. Several documents are available to serve the needs of a variety of users. Figure 1-1 illustrates the manual set. To obtain additional copies of these, or other manuals when developed, check with a local Fluke Sales Office, listed at the end of this manual. A summary description of each manual is given below:

1720A User Manual	An introductory manual for the programmer or system designer who is using the 1720A to set up an instrumentation system. It presents both the purpose of and the interaction between the various software and hardware resources in the 1720A. In addition, it serves as a reference guide for FDOS, the Console Monitor, and the utility programs. Fluke Part Number 518654.
1720A Fluke BASIC Programming Manual	A description of the Fluke-enhanced ANSI-standard BASIC language developed for the 1720A Instrument Controller. It is arranged by subject, and combines syntax diagrams with examples. Emphasis is on instrumentation system control. This manual presumes familiarity with the 1720A User Manual. Fluke Part Number 518670.
1720A BASIC Reference Guide	A pocket-size quick reference guide that summarizes the most often used contents of the User and BASIC Programming manuals. Fluke Part Number 526210.
HPL to Fluke BASIC Handbook	A specialized manual for the programmer who is familiar with the Hewlett-Packard 9825 Calculator using the HPL language. Building upon this familiarity, the handbook describes functional similarities and differences and leads the programmer through the task of converting existing HPL programs for use with the 1720A Instrument Controller. Fluke Part Number 546341.

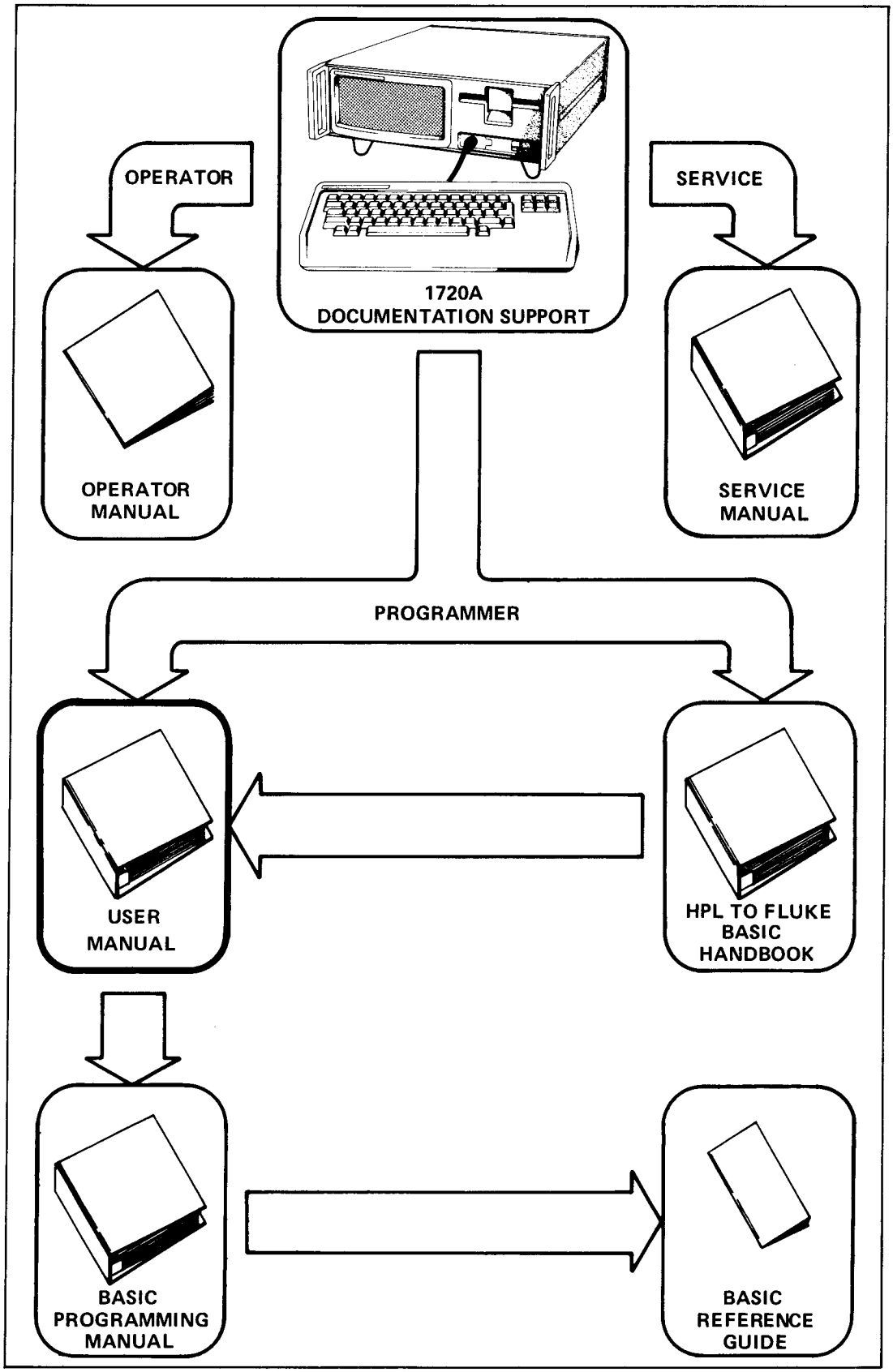


Figure 1-1. 1720A Manual Set

1720A Display Worksheet Pads	A grid-pattern worksheet that shows both the normal and double size character positions and the touch sensitive areas. This aids the programmer in the design of effective display layouts. Available in pads of 50. Fluke Part Number 533547.
1720A Operator Manual	A brief manual for the operator of a programmed 1720A-controlled instrument system. It discusses proper disk handling and error interpretation and includes an error and trouble-incident log. Fluke Part Number 518647.
1720A Service Manual	A component level theory of operation for each hardware module, with diagnostic procedures to resolve failures to the modular level. Schematics and parts lists are included. Fluke Part Number 518662.

1-3. INTRODUCTION TO THIS MANUAL

1-4. This manual introduces the 1720A and is a reference for FDOS, the Console Monitor, and the other utility programs provided. It establishes familiarity with both software and hardware capabilities. The Fluke BASIC Programming Manual and other language manuals then build upon this system familiarity describing the use of a programming language and its associated software tools. The subject matter is divided into sections as follows:

Section 1	Introduction. Describes the documentation set, the User Manual, syntax diagrams, and the 1720A Instrument Controller.
Section 2	Installation. Covers unpacking and set-up. Identifies controls, indicators and connectors. Includes start-up procedures and a software module compatibility check.
Section 3	Software. Describes use of FDOS, the Console Monitor, and each system-level utility program. Divided into modular subsections.
Section 4	Hardware. Describes use of each of the hardware modules as a system resource. Divided into modular subsections.

1-5. Programming languages are covered in separate manuals that presume familiarity with this user manual. Where programming examples are required, Fluke Enhanced BASIC is used unless otherwise specified.

1-6. Command Definitions

1-7. The length of any command line cannot exceed 80 characters. Commands are introduced with a syntax diagram that defines legitimate command constructs. A complete definition of the command follows the syntax diagram. The short indented paragraphs (with a "bullet" in place of a paragraph number) following the syntax diagram are part of the definition.

1-8. Syntax Diagrams

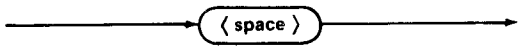
1-9. Syntax diagrams are used throughout this manual to define correct spelling, punctuation, sequences of words, symbols, and expressions for system and utility commands. The following guidelines define proper use of these diagrams:

- Any path through a diagram starting from the left that does not run contrary to an arrowhead forms a legitimate command construct. The text accompanying the diagram explains legal usage.

- Boldface words in a circular enclosure are to be entered exactly as shown. Example:

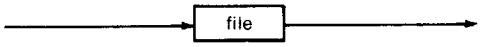
- Key entries with names, such as ESC or RETURN, are shown in a box with rounded corners. Example:

- A required space character entry is always shown as:

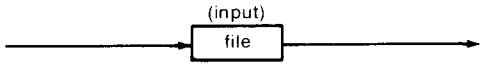


- Control character entries are shown in circular enclosures within angle brackets. The representation CTRL/ means to hold the CTRL key depressed while typing the letter that follows. Example:

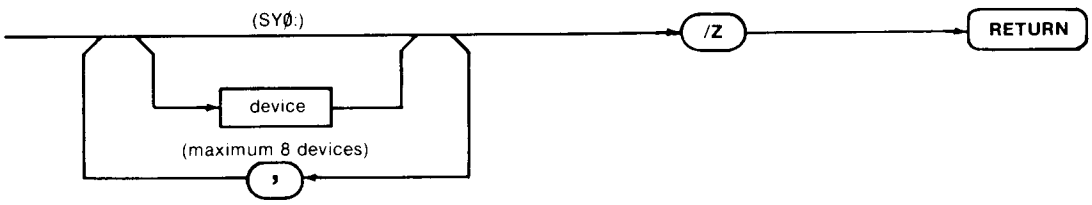
- Lower case words enclosed in a box represent other information to be supplied. Example:



- Words outside the path of the diagram, usually in parentheses, provide supplementary information. These words are normally not part of the definition of the statement. Example:



1-10. The following syntax diagram of the zero directory command of the File Utility Program illustrates these points. The diagram shows that the basic construction of the command is /Z. The supplementary note shows that the default system device (SY0:) is assumed when one is not specified. The diagram also shows that one or more devices may be specified, separated by commas.



1-11. THE 1720A INSTRUMENT CONTROLLER

1-12. The Fluke 1720A Instrument Controller was designed for the task of building and controlling instrument systems. Because it has a removable Programmer Keyboard, a Touch-Sensitive Display for operator interaction, and a rack-mountable package, it can be used on the programmer's desk or rack mounted in an industrial instrumentation system.

1-13. The 1720A includes a 16-bit microcomputer and is supplied with a System Disk containing Fluke-designed system software, utility programs, and one or more programming languages. Dual IEEE-488-1978 standard ports allow the 1720A to control up to 28 compatible instruments or peripheral devices. In addition, two RS-232-C ports permit the 1720A to interface with a wide variety of general purpose peripherals and data communications equipment.

1-14. The 1720A is not a business computer or calculator that has been adapted to the job of an industrial controller. Instead, the 1720A is a special purpose machine that has a variety of standard features specifically designed for controlling industrial instruments. These standard features include:

- A rack-mount package that occupies a minimum of valuable rack space.
- A full-width touch-sensitive display that allows the design of operator control interfaces that are flexible, simple to use, and specifically adapted to the process.
- A detachable, ASCII keyboard that allows separation of programming functions from daily operation.
- A two-port IEEE-488-1978 interface that allows the 1720A to function as two controllers with up to 14 instruments in each system.
- A software development system that includes an operating system, a monitor, utility programs, and one or more programming languages with an editor and other supporting software tools.
- A double-density floppy disk drive that allows storage of programs and files on a permanent media. The floppy disks provided with the 1720A contain the standard software development programs as well as a complete set of diagnostic software for isolation of failures to the module level.
- A soft-loaded software architecture that allows field upgrading of software as new capabilities and languages are developed.
- A 60K byte memory module that provides the main memory area for program execution, file transfer, and the execution area for all system software.
- An optional electronic disk memory module, E-Disk™, that provides an additional 128K or 256K bytes of storage. It is intended for use as a fast access operation media. Internal battery backup can sustain the contents of the electronic disk for a short time when AC power is removed. This backup time may be extended indefinitely by using an external battery. (E-Disk is a trademark of the John Fluke Mfg. Co., Inc.)

- Two independent software-controlled RS-232-C ports that provide serial access to a wide variety of general-purpose peripherals and data communications equipment. Software control of the ports eliminates mechanical switches for baud rate selection, and gives flexibility in protocol.

- A real-time clock that may be used as a determining program variable during program execution. It also allows application programs to date- and time- stamp data files as they are collected.

Section 2
Installation and Start-Up

2-1. INTRODUCTION

2-2. This section covers unpacking and installation. In addition, it provides a description of the controls, indicators, and connectors. Following this, it presents start-up procedures for putting the 1720A into operation.

2-3. UNPACKING

2-4. The 1720A Instrument Controller is carefully packed for shipping to ensure that the unit arrives in good condition. Unpack all containers carefully and check all packing materials for accessories, cables, and manuals. Do not dispose of the packing materials before inspecting for shipping damage. If this inspection reveals damage or indicates that damage might have occured, notify the shipper immediately. Then call a Fluke Sales Office or Customer Service Office. A checklist of the standard and optional items included with the 1720A Instrument Controller is presented in paragraph 2-6.

2-5. If the 1720A Instrument Controller needs to be shipped again at a later date, the original packing carton should be used with all fillers properly in place. Fluke does not recommend shipping the 1720A in a substitute container. To obtain an approved shipping container, call any Fluke Sales Office.

2-6. Unpacking Checklist

1. Controller Mainframe
2. Programmer Keyboard
3. Power Cord
4. System Disk
5. Diagnostics Disk
6. Data Disk
7. This User Manual
8. 1720A Fluke-Enhanced BASIC Programming Manual
9. 1720A Fluke BASIC Reference Guide (pocket-size)
10. 1720A Operator Manual
11. 1720A Service Manual
12. 1720A Controller Programming Work Sheet (Pad of 50)

2-7. Options and Accessories

2-8. These items are not included in the shipment unless ordered separately. They may or may not be separately packaged.

1. 128K Byte Electronic Disk Module, Option 1720A-001, maximum two per unit. These will be installed in the mainframe if ordered with the Instrument Controller.

2. Extra Programmer Keyboard (accessory Y1720).

3. Ten-pack of blank floppy disks, unformatted (accessory Y1706).

4. IEEE-488-1978 Cables, one-meter length (accessory Y8001), two-meter length (accessory Y8002), four-meter length (accessory Y8003).

5. RS-232-C Standard Interface Cable, allows connection to standard telephone data sets and modem-type devices. Two-meter length (accessory Y1707). Ten-meter length (accessory Y1708).

6. RS-232-C Null Modem Cable, 0.3 meter length. This cable allows connection to standard data terminal devices (accessory Y1705).

7. RS-232-C Printer Cable, two-meter length. This cable is designed for connecting the 1720A to a Fluke 1776A Serial Printer (accessory Y1709).

8. Rack Mounting Kit, includes 61 cm (24 inch) slides (accessory number Y1790).

9. Shipping Case (accessory Y1711).

10. Customer Maintenance Kit, includes the tools and accessories that are required by the diagnostic and troubleshooting procedures in the 1720A Service Manual. This kit is included in the Module Service Kits below. Order through Fluke Service Centers as part number 577791.

11. Module Service Kit, includes one of each of the functional modules, the Maintenance Kit of required tools, and a selection of additional parts and tools. This is the basic spares kit for modular servicing. Order through Fluke Service Centers as part number 539379.

12. Module Service Kit with E-Disk™, includes one of each of the functional modules, the Maintenance Kit of required tools and a selection of additional parts and tools. This kit is identical to the 539379 kit except for the addition of an E-Disk™ module and associated spares. Order through Fluke Service Centers as part number 537506.

13. CRT Display Service Kit, includes the CRT display tube with factory aligned yoke, the high voltage transformer, and the horizontal and vertical drive electronics. Order through Fluke Service Centers as part number 537472.

14. 70% Level Service Kit, includes a minimum number of functional modules that have been calculated or shown by experience to have the lowest mean time between failure (MTBF). This kit must be supplemented by a Maintenance Kit. Order through Fluke Service Centers as part number 539361.

2-9. INSTALLATION

2-10. Installation of the 1720A requires locating the 1720A on a suitable table, attaching the keyboard and line cord, and turning on the power. The Power Switch is located on the rear panel. The line voltage has been factory set as ordered. This voltage setting is indicated on the rear panel decal.

2-11. Line Voltage Range Selection

2-12. The 1720A Instrument Controller uses a high-efficiency switching power supply that will operate from any of the inputs shown in Table 2-1.

Table 2-1. Power Supply Inputs

VOTAGE RANGE	LINE FREQUENCY	SWITCH SETTING	FUSE
90V to 132V	47 Hz to 63 Hz	115	3A SB
192V to 250V	47 Hz to 63 Hz	230	1.5A SB
104V to 126V	380 Hz to 420 Hz	115	3A SB

2-13. If line voltage requirements change, it will be necessary to set the power supply line voltage switches. It may also be necessary to change the fuse to match the power source and install an appropriate power plug.

CAUTION

Do not operate the 1720A from a 230V-range source with the 3 amp fuse installed. This size fuse may be unable to protect the unit from internal damage if a failure-caused overload occurs.

WARNING

TO AVOID ELECTRICAL SHOCK HAZARD, REMOVE THE POWER CORD FROM THE 1720A AT LEAST FIVE (5) MINUTES PRIOR TO PERFORMING THE FOLLOWING PROCEDURE. THIS WILL ALLOW TIME FOR THE INTERNAL CAPACITORS TO DISCHARGE TO A SAFE LEVEL.

2-14. To gain access to the voltage range setting, use a #1 Phillips screwdriver to remove the three screws holding the Power Supply Module to the chassis and slide the power supply straight out the rear. Facing the rear of the unit, the power supply is the right half-panel. The two slide switches are located under the right side of the top circuit board as illustrated in Figure 2-1, and are labeled with the numbers 115 and 230.

2-15. The fuse is accessible without removing the power supply, and is on the right side of the rear panel. Be sure that the fuse value corresponds to the setting of the voltage range switches as given in Table 2-1.

2-16. Environment

2-17. The 1720A requires adequate ventilation to maintain the correct operating temperature. Keep the rear panel at least 10 cm. (4 inches) from the wall so that the fan can be effective. There is no minimum distance for top or bottom placement, as the controller is designed to be mounted in a rack with other instruments.

2-18. The 1720A operates reliably within the temperature range of 10°C to 40°C (50°F to 104°F). The controller may be turned on at lower temperatures, down to -10°C (14°F), to help warm it up to room temperature. The relative humidity should be within the range of 8% to 80%, non-condensing.

CAUTION

Low humidity environments can induce electrostatic discharges within the rotating floppy disk, and between the touch-sensitive display and the operator. Such discharges can permanently damage high-density MOS microcircuitry within the 1720A.

2-19. The 1720A must be stored (non-operating) within the temperature range of -10°C to 60°C (14°F to 140°F), and within a humidity range of 5% to 95%, non-condensing.

2-20. Floppy disks are more sensitive to storage environments than the instrument controller. They must not be allowed to become colder than 10°C (50°F), or warmer than 52°C (126°F). Relative humidity must be kept within the range 8% to 90%, non-condensing. If a disk has been stored outside these limits, the best way to attempt recovery of data is to allow it to slowly reach room temperature and humidity at least 3 to 4 hours before placing it in the controller. The Start-up Procedures include other disk handling precautions.

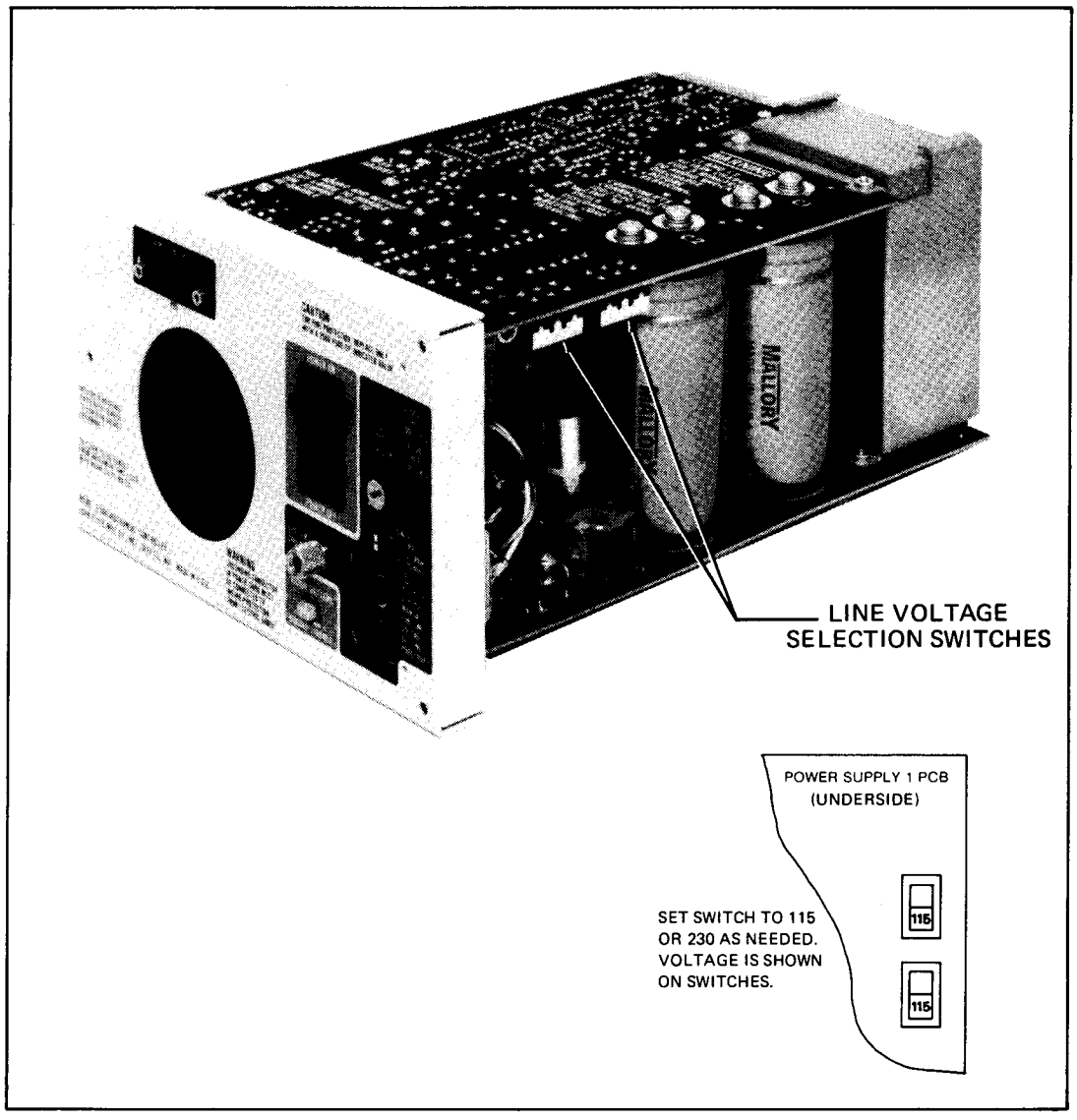


Figure 2-1. Power Voltage Range Selection

2-21. CONTROLS, INDICATORS, AND CONNECTORS

2-22. Figure 2-2 and Table 2-2, illustrate and describe the front and rear panels, and Figure 2-3 and Table 2-3, illustrate and describe the programmer keyboard.

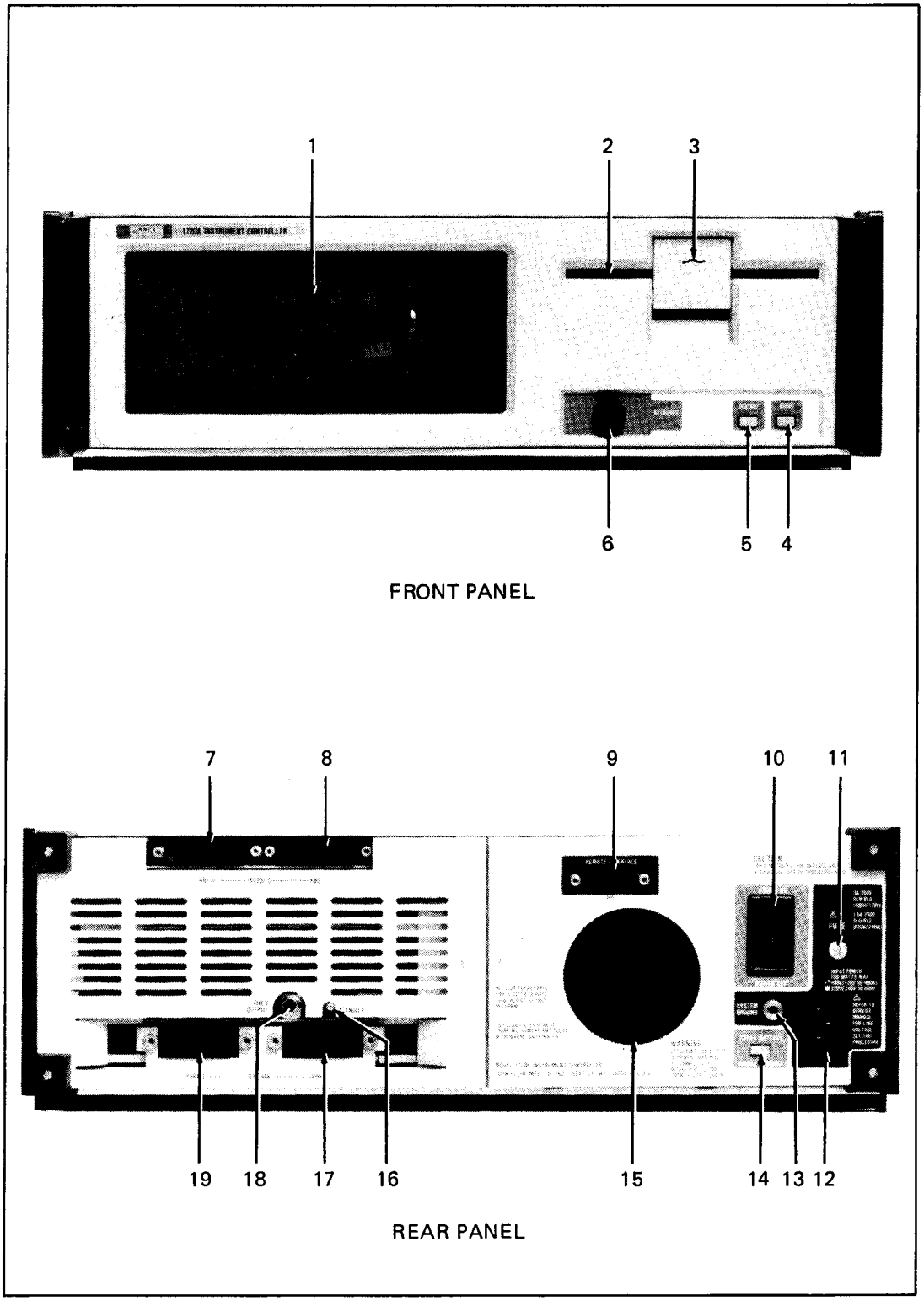


Figure 2-2. Front and Rear Panel Controls, Indicators, and Connectors

Table 2-2. Front and Rear Panel Controls, Indicators, and Connectors

REF. NO.	NAME	FUNCTION
1	Touch-Sensitive Display	A display with touch sensitive areas which allows operation of a program by touch according to commands shown on the screen. The commands are program controlled and generated.
2	Disk Loading Slot	The floppy disk is loaded into the disk loading slot for program execution and data storage.
3	Disk Entry Latch	Seats disk for operation.
4	ABORT Push Button Switch	Clears devices on the IEEE-488 buses and either stops the program or allows automatic program continuation, depending on the application program.
5	RESTART Push Button Switch	Initiates self test, clears main memory, and reloads program.
6	Keyboard Connector	For connecting the detachable keyboard.
7	RS-232-C Connector #1	Standard I/O port for peripheral devices. (KB1:)
8	RS-232-C Connector #2	Standard I/O port for peripheral devices. (KB2:)
9	Remote Interface Connector	Provides for external E-Disk battery back-up, external control of ABORT and RESTART switches, and two software controlled output lines.
10	Main Power Switch	Applies and removes line power to the instrument controller.
11	Fuse	Power supply line fuse.
12	Power Connector	Input line power connector.
13	System Ground	A common ground for system components.
14	E-Disk Enable Switch	Enables battery backup for E-Disk.
15	Air Filter	Filters air used to cool power supply. See maintenance procedures.
16	Display Intensity Control	A screwdriver adjustment which controls the intensity of the display.
17	IEEE-488 Port 1 Connector	Standard IEEE-488-1978 connector for instrumentation.
18	Composite Video Connector	Drives standard data display monitor.
19	IEEE-488 Port 0 Connector	Standard IEEE-488-1978 connector for instrumentation.

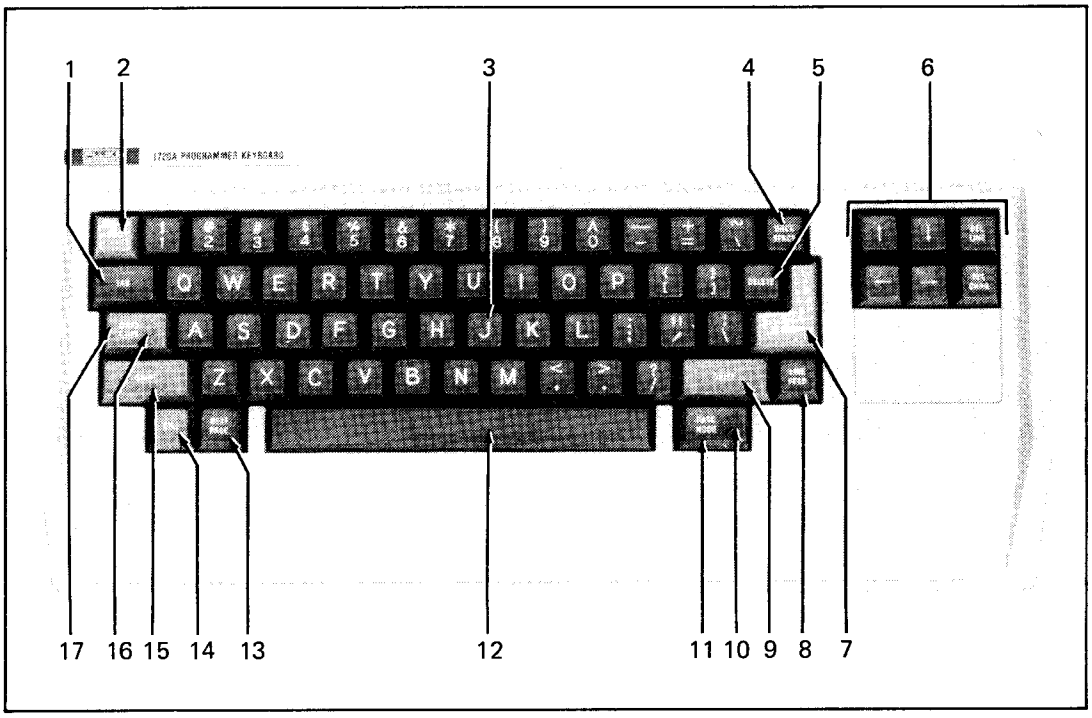


Figure 2-3. Programmer Keyboard

Table 2-3. Programmer Keyboard

REF. NO.	NAME	FUNCTION
1	TAB Key	Not used at this time (Horizontal Tab code).
2	ESCape Key	Escape code.
3	Character Entry Keys	In standard typewriter layout, for entry of characters and numbers into the 1720A.
4	BACK SPACE Key	Functional only in Edit Mode.
5	DELETE Key	Delete the character just entered and backspace to the previous position.
6	Editing Keypad	Functional only in EDIT Mode. Arrows indicate direction of cursor movement. DELete LINE and DELete CHARacter act upon the current cursor position.
7	RETURN Key	General-purpose entry key. Command or statement entries in all modes are stored in temporary buffer unit RETURN is pressed. The display then moves the cursor to the start of the next line.
8	LINE FEED Key	Functional only in Edit Mode.
9	SHIFT Key	A key modifier that must be held down while another key is pressed to select the upper character of any key that has two characters on it, or the CAPITAL of any alpha character.

Table 2-3. Programmer Keyboard (cont)

REF. NO.	NAME	FUNCTION
10	Page Mode Indicator	ON when Page Mode has been selected. See Reference Number 11.
11	PAGE MODE	Alternate action switch that selects or cancels Page Mode. In Page Mode, the indicator is lighted, and display output stops at the bottom without scrolling. See Reference Number 13.
12	Space Bar	Generates space characters.
13	NEXT PAGE Key	Functional in Page Mode only. Clears the screen and allows an additional 16 display lines.
14	CTRL (Control) Key	A key modifier that must be held down while another key is pressed.
15	SHIFT Key	A key modifier that must be held down while another key is pressed to select the upper character of any key that has two characters on it, or the CAPITAL of any alpha character.
16	Caps Lock Indicator	ON when Caps Lock Mode has been selected. See Reference Number 17.
17	CAPS LOCK	Alternate action switch that places the controller into and out of Caps Lock Mode. In Caps Lock Mode the indicator is lighted, and all alpha characters are automatically capitalized. Other characters remain unchanged.

2-23. START-UP PROCEDURE

2-24. Following is a step by step procedure for starting up the 1720A Instrument Controller. This procedure is for the standard 1720A, supplied with Fluke-enhanced BASIC.

1. Check for proper line voltage selection as discussed under Installation earlier in this section. Plug the power cord into a properly grounded outlet.

WARNING

SAFE OPERATION OF THE 1720A, AS WELL AS RELIABLE SYSTEM OPERATION REQUIRES A GROUNDED POWER CONNECTION.

2. Apply power to the system by depressing the top of the power switch located on the rear panel.

3. Release the door latch on the floppy disk drive by pressing in on the top of the door, as shown in Figure 2-4.

4. Read the disk handling precautions in Table 2-4. Gently insert the System Disk into the disk drive with the label upward, as shown in Figure 2-4, until the disk is fully seated. Then latch the drive closed by pressing the bottom of the door.

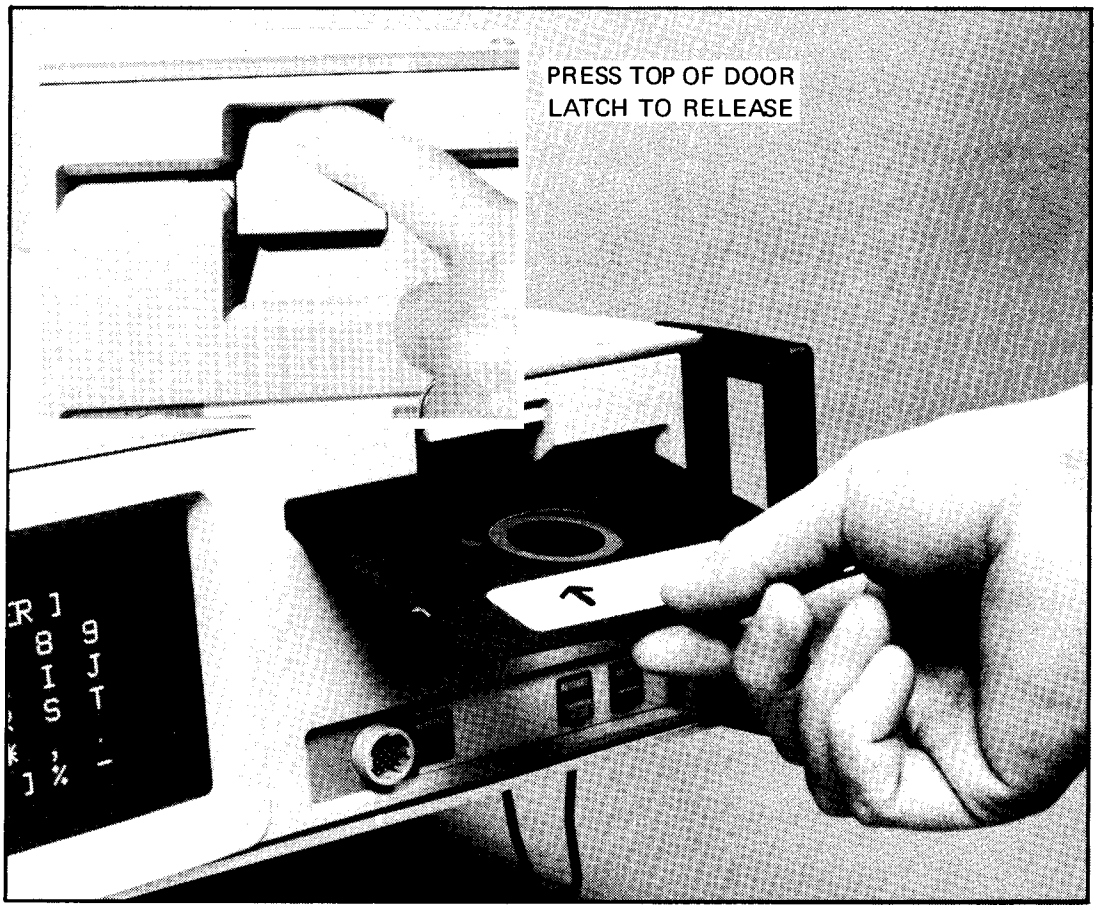


Figure 2-4. Inserting the Disk

Table 2-4. Disk Handling Precautions

1.	Keep the disk in its protective folder when not in use.
2.	The disk surface can be easily damaged. Do not touch the exposed disk surface with hands or any material which may damage it.
3.	Careless handling may damage the disk. Avoid dropping, throwing, or twisting the disk.
4.	Store disks vertically and support them so they will not bend. Stacking may distort them and affect their contents. Avoid placing heavy objects on the disks.
5.	Exposure to heat and sunlight may warp the disks.
6.	Magnetic sources may erase data on the disks. Keep the disks away from electric motors, generators, and transformers.
7.	Mark disk labels before placing them on the disk envelope or use a felt tip pen. Pressure from a ball point pen or pencil may destroy data on the disk.
8.	Temperature affects the performance of the disk. Do not use a disk that is overheated or overcooled until it has reached room temperature.

5. Press RESTART. This step may be avoided by inserting the disk before turning power ON; however, some of the following messages would not appear as the display system warms up to normal intensity.

6. Observe the display. It should read:

```
FLUKE 1720A CONTROLLER
HELLO
BOOT V1.2
```

7. Verify that the version number is 1.2. If it is not, check the front of this manual for supplemental information, or call a Fluke Customer Service Center for advice.

8. Following this display, the 1720A then performs a self-test procedure from permanent (ROM) memory located on the CPU module. The display changes the `BOOT V1.2` to `SELF TEST`. The Self Test functionally tests the following modules:

a. CPU Module

b. Floppy Disk Interface

c. Memory Module

d. Electronic Disk (if present)

e. Keyboard/Video Interface

f. IEEE-488 Interface

9. Refer to Table 2-5 if any error messages occur. This table presents the error messages that the self-test program can generate.

10. After the self-test, the 1720A loads the Floppy Disk Operating System software (FDOS) into main memory from the floppy disk. During the loading process, the display reads:

```
FLUKE 1720A CONTROLLER
LOADING
```

11. When loading is complete, FDOS takes control and attempts to load a System Command File from the disk by looking for a file named COMMND.SYS. A command file contains stored keyboard commands that would otherwise have to be typed in. Section 3 explains these concepts.

a. If the file COMMND.SYS is not found, FDOS loads the Console Monitor (CONMON) directly and transfers control to it. This may be the case if the disk being loaded is not the system disk supplied with the 1720A, or the COMMND.SYS file has been renamed or deleted. The Console Monitor displays a # prompt character. Section 3 explains the use of CONMON.

b. If the file COMMND.SYS is found, it is loaded and command file status is designated active. The display then reads:

```
Command File Active
```

12. The next action depends on the contents of the command file. The System Disk supplied with the 1720A contains a short system command file that first checks the TIME utility to see if the time has been set. It then loads the BASIC interpreter and transfers control to it.

Table 2-5. Self Test Errors

MESSAGE	ACTION
!Memory Error	Indicates a problem with the memory board. Service may be required.
!Video Error	Indicates a problem with the video board. Service may be required.
!IEEE Missing or Faulty	Indicates a problem with the IEEE Bus Interface board. Check that all bus instruments connected to the controller are ON and that all cables are securely in place. If everything seems correct and the message still appears when RESTART is again pressed, service may be required. *NOTE* *Disconnect the IEEE Bus cables from the controller to confirm that the instruments are the source of trouble.*
!Floppy Error	Indicates a problem with the floppy disk drive or its control board. Service may be required.
?No System on Device	Check that the disk in the drive is a system disk. Try another system disk. The wrong disk may be inserted or it may be inserted incorrectly.
?Disk not Mounted	Insert or reinsert a system disk. Either there is no disk inserted or it is improperly inserted. Check to see if the disk is upside down or the entry latch has not been closed.
?Floppy Error	The inserted disk is faulty. Try another system disk.
?Bad Directory on Device	The directory on the disk is incorrect. Use another disk.

13. If you are using the System Disk, and the time clock has not previously been set, the display will next read:

```
Enter Date:  DD-MM-YY
```

14. Type in today's date in numeric form, starting with the day, then the month, and then the year. The entries must be separated by any non-numeric character. Use DELETE to correct any mistakes. Press RETURN and the display reads:

```
Enter Time:  HH-MM
```

15. Type in the time, in 24-hour form, starting with the hour, followed by the minutes. The entries must be separated by any non-numeric character. Press RETURN and the display reads:

```
BASIC  (time)  (date)

Basic Version 1.3

Ready
```

NOTE

The 24-hour day starts at 00:00 (midnight) and ends at 23:59 (just before midnight). If the time is at or beyond midnight and before 1:00 A.M., subtract 12 hours. If the time is at or beyond 1:00 P.M., and before midnight, add 12 hours. Otherwise, use the time unchanged.

Three examples:

TIME	24-HOUR TIME
12:30 A.M.	00:30
9:45 A.M.	09:45
4:20 P.M.	16:20

16. **The 1720A is now in Immediate Mode BASIC. Check that the version number agrees with the title page. If it is not, check the front of this manual or the Fluke BASIC Programming Manual for supplemental information, or call a Fluke Customer Service Center for advice. Refer to the 1720A Fluke BASIC Programming Manual for details on the use of BASIC in the 1720A Controller. Information on loading and the use of other languages is presented in separate manuals.**

2-25. SOFTWARE COMPATIBILITY CHECK

2-26. Software modules are supplied with the 1720A Instrument Controller as files on the System Disk. These machine-language program files are interdependent and are compatible only in the combinations of versions supplied by Fluke and documented in published Fluke manuals, including addendum inserts. The portability and copying ease of Fluke system software allows the user to take advantage of the continuing program of software development at Fluke. The user should make sure however in copying new or updated Fluke system software that all of the software modules are compatible. This can be verified by obtaining the appropriate Fluke manuals and then following the procedure below.

2-27. Because software modules are easily erased and copied, do not assume that the label on a Fluke System Disk is valid unless the disk is new. The use of incompatible system software modules will produce unpredictable results for which Fluke cannot provide analytical, diagnostic, or other software support, except for identification of compatible combinations.

2-28. Compatible Versions of System Software

2-29. The versions of Fluke system software modules that are compatible and supported in this manual are listed in table 2-6. If subsequent software releases are also supported by this manual, an addendum insert is available that identifies the compatible versions and any functional changes and additions.

Table 2-6. Compatible Versions of System Software

NAME	VERSION	NAME	VERSION
BOOT	SEE	TIME.CIL	SEE
FDOS.SYS	TITLE	SET.CIL	TITLE
CONMON.SYS	PAGE	FUP.CIL	PAGE

NOTE

BOOT is contained in PROM memory on the CPU module. All other system software modules are supplied as disk files.

2-30. Procedure

1. With power ON, insert the System Disk and press RESTART.

2. Check that the initial display includes `BOOT V1.2` .

3. If the final display includes `Command File Active` the version of FDOS will not be displayed unless the system command file is temporarily renamed as follows:

   a. Enter ⟨CTRL/P⟩ (Hold [CTRL] down while typing P.)

   b. After the prompt #, type `FUP` [RETURN].

   c. After the prompt *, type `MFϕ:/A` [RETURN].

   d. After the prompt *, type `COMMND.CMD=COMMND.SYS/R` [RETURN].

   e. Press RESTART

4. Check that the final display includes `FDOS Version 1.5` and `Console Monitor Version 1.2` .

5. After the prompt #, type `TIME` [RETURN]. Check that the resulting display includes `Time Version 1.0`. Then enter ⟨CTRL/P⟩.

6. After the prompt #, type `SET` [RETURN]. Check that the resulting display includes `Set Version 2.0` . Then enter ⟨CTRL/P⟩.

7. After the prompt #, type `FUP` [RETURN]. Check that the resulting display includes `File Utility Program Version 1.1`.

8. If the system command file was temporarily renamed in step 3, restore it as follows:
   After the prompt *, type `COMMND.SYS=COMMND.CMD/R` [RETURN]

9. Press RESTART.

2-31. If any of the displays do not agree with these software versions, check the front of this manual for supplemental information, or call a Fluke Technical Service Center for advice.

CAUTION

Do not continue to use the 1720A with system software modules that are not known to be compatible and supported by a published Fluke manual or manual addendum. To do so will produce unpredictable results for which Fluke cannot provide analytical, diagnostic, or other support, except for identification of compatible combinations.

Section 3
System Software

3-1. INTRODUCTION

3-2. System Software developed for the 1720A Instrument Controller consists of distinct modules that serve specific purposes. This section provides a brief description of each software module.

3-3. Detailed functional descriptions of each software module, Fluke-Enhanced BASIC, and other programming languages are covered in separate manuals. The software modules are summarized below.

3-4. FDOS

3-5. The Floppy Disk Operating System (FDOS) performs file management for all programs and responds to interrupts from input and output (I/O) devices, such as the keyboard and the floppy disk. Whenever the 1720A is operating under normal conditions, FDOS is present in Main Memory. FDOS also includes a processor that allows command files of stored user defined keyboard inputs to be processed.

3-6. CONMON

3-7. The Console Monitor (CONMON) is a system program that works through FDOS to simplify the process of loading and transferring control to the other software tools available to the programmer. This software, including CONMON, remains on the system disk, or when configured, in the optional electronic disk until needed. This arrangement allows the 1720A to make efficient use of the Main Memory.

3-8. TIME

3-9. The Time and Date Set utility (TIME) enables the user to set the time and date into the system clock where it is accurately maintained for use by other programs.

3-10. SET

3-11. The Set Serial Port utility (SET) enables the user to independently set the baud rate, number of data bits, stop bits, parity, and line and file terminator characters for each RS-232-C Serial Port.

3-12. FUP

3-13. The File Utility Program (FUP) is a file transfer and management program that gives the programmer flexible control over the files on floppy disks and on the optional electronic disk. FUP has many other capabilities which are explained fully.

Section 4
Hardware

4-1. INTRODUCTION

4-2. This section describes each of the hardware modules individually from a user point of view. Beginning with an overview of the hardware architecture, a separate subsection is then devoted to each module. The reader who needs additional technical detail should consult the 1720A Instrument Controller Service Manual.

4-3. SYSTEM ARCHITECTURE

4-4. Refer to Figure 4-1 for a functional view of the 1720A Instrument Controller. Using system software, the CPU Module processes information to and from the other modules while directing their functions. The actions of the CPU are direct responses to a variety of conditions in the sequence that they occur. Among these are the statements in a user program, inputs from the programmer keyboard or the touch sensitive display, and interrupts from the disk drive or an external device.

4-5. The modular design of the 1720A Instrument Controller results in a segregation of tasks among the modules. These modules are internally connected by a high speed parallel bus similar to that found in computer systems. In addition to ease of servicing, this design approach enhances functional and processing throughput. The subsections that follow discuss the functions of each module independent of the programming language or utility program in use. Fluke Enhanced BASIC is used wherever programming examples are provided unless specified otherwise.

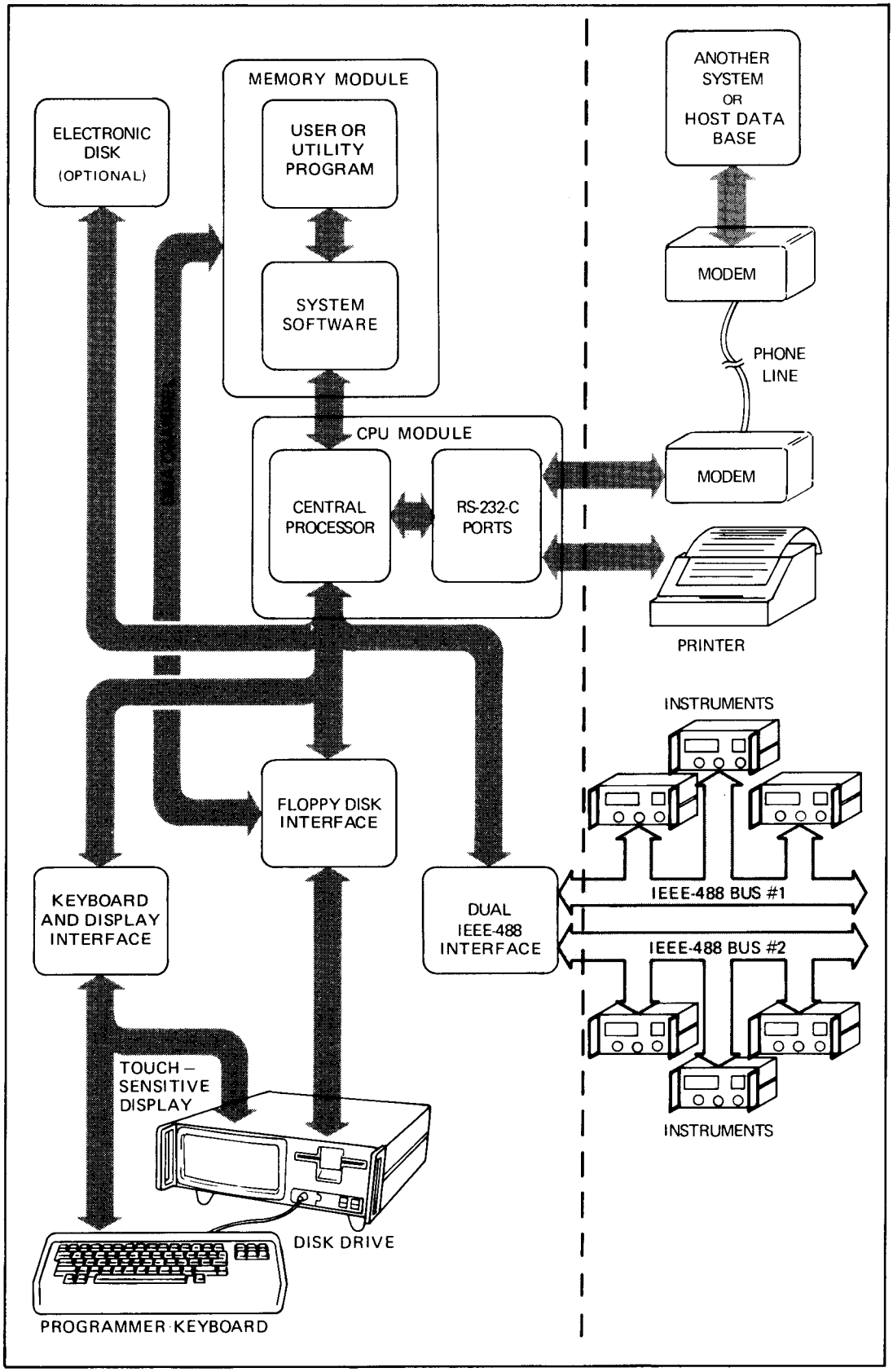


Figure 4-1. Functional View of the 1720A Instrument Controller

Section 4A
CPU MODULE

4A-1. INTRODUCTION

4A-2. The CPU Module gives the 1720A Instrument Controller the ability to accept and carry out user programs and control an instrument system precisely as the user intended. Through system software residing in the Main Memory Module, the user's program is processed into control actions, data collection and processing, and event-driven decisions. The system software may include processing tools such as a language interpreter as well as programming tools such as a compiler, assembler, linker, or editor. These software tools, covered in separate manuals, are placed in Main Memory by FDOS only when they are in use.

4A-3. The instructions that the CPU Module receives from memory are in 16-bit binary coded machine language corresponding to the microprocessor instruction set. Since these codes are not easily recognizable to the user,one of the functions of system software modules, themselves in binary form, is to provide the CPU with the necessary instructions to change the user recognizable utility commands and program statements into binary machine codes.

4A-4. LOCAL MEMORY

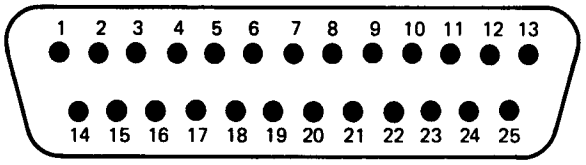
4A-5. The CPU Module intercepts memory addresses generated by its microprocessor within the upper 2048 words of address space for use on the module. Half of this is assigned to permanent read-only memory (ROM) located on the CPU module. The standard program factory recorded in this memory performs a functional test of each hardware module at start-up and then loads FDOS from the System Device into Main Memory. Since this ROM memory is in sockets, special purpose ROM components can be substituted to augment troubleshooting procedures. Refer to the 1720A Instrument Controller Service Manual for further detail.

4A-6. Of the remaining 1024 words, 256 are assigned to a local scratchpad memory, and 768 are set aside for directly addressing other modules in the system. This technique of addressing a hardware module through a memory location is called memory-mapped I/O.

4A-7. THE SERIAL PORTS

4A-8. The CPU Module includes two serial interfaces that conform to EIA Standard RS-232-C for Data Terminal Equipment. These ports are assigned the names KB1: and KB2: respectively by FDOS. Table 4A-1 lists the pin connections for these ports with a summary description of each pin function.

Table 4A-1. RS-232-C Serial Data Ports (KB1: and KB2:)



(MALE PINS)

PIN	NAME	DIRECTION	MEANING
1	GND	---	Ground. See below.
2	$\overline{\text{XMT}}$	OUT	Transmitted Data. (Low = 1)
3	$\overline{\text{REC}}$	IN	Received Data. (Low = 1)
4	RTS	OUT	Request To Send. Asserted (high) during data transmission.
5	CTS	IN	Clear To Send. Must be high or open for data to be transmitted. May be used to stop transmission on character boundaries by switching to low (nominal -12V).
6	DSR	IN	Data Set Ready. Not required by current driver software.
7	RET	---	Signal Return. Zero-voltage reference point for all other signals.
11	---	OUT	Unassigned. Always high. (+12V through 4700 ohms)
12	SCD	IN	Secondary Carrier Detect. Must be high or open for data to be transmitted. May be used to stop transmission on character boundaries by switching to low (nominal -12V).
19	SRTS	OUT	Secondary Request To Send. Always high (+12V through 4700 ohms)
20	DTR	OUT	Data Terminal Ready. Always high. (+12V through 4700 ohms)

4A-9. Grounding

4A-10. Pin 1 of an RS-232-C cable supplied by Fluke is connected to the cable shield. This pin is not a reference point for signals. Its purpose is to connect the cable shield to a system ground. Make sure that use of this connection does not result in circular grounding paths (loops) through the system. Such ground loops can conduct current of enough magnitude to interfere with data transmission.

4A-11. In the 1720A, Pin 1 of Serial Port 1 is connected to Pin 1 of Serial Port 2, and then to a jumper point on the board just to the right of Serial Port 2. This jumper connects Pin 1 of both ports to the system chassis ground. Cutting the jumper disconnects both cable shields from the 1720A chassis, but they remain interconnected. In general it is good practice to ground only one end of each cable shield to avoid circular ground paths.

4A-12. Signal Characteristics

4A-13. RS-232-C uses larger voltages than standard logic levels, and they swing both positive and negative. Data 0 and logic true are represented by a range from +3V to +25V, nominally +12V. Data 1 and logic false are represented by a range from -3V to -25V, nominally -12V. The rate of change of these voltages is limited to 30 volts per microsecond,maximum.

4A-14. The 1720A serial port output lines are nominally ±12V. The can withstand shorting to ground or being connected to an opposing output without damage. They are designed for a DC load between 3000 ohms and 7000 ohms, with a capacitive load of less than 2500 picofarads. They are not designed for inductive loads, and cannot drive a relay coil. The capacitive loading restriction generally limits cable lengths to about fifteen meters. Special low capacitance cables, however, can extend this limit to 50 meters or more.

4A-15. The 1720A serial port input lines can withstand ±30V inputs and are set for an input threshold level of -0.75V. An open input line is interpreted as high.

4A-16. Parameters

4A-17. The start-up condition of the serial ports, when not modified by SET commands in a system command file, is as follows:

Baud Rate	S1 setting
Data Bits per Character	8
Parity	None
Stop Bits per Character	1
End-of-Line Code	10 (line feed)
End-of-File Code	26 (⟨CTRL/Z⟩)

4A-18. Rotary switch S1 is located on the CPU module near the connector for Serial Port 2. This switch is factory set for 300 baud. It is relatively simple to prepare a system command file containing SET commands that change any of the port parameters, independently of each other, automatically at start-up time. For this reason, the setting of S1 is normally of no concern to the user.

4A-19. External Terminal

4A-20. When the Video Interface Module is removed, the console device (KBØ:) defaults to Serial Port 2. This feature allows an external terminal to be used as console if the Video Interface Module, the programmer keyboard, or the display fails. This procedure should be considered a temporary remedy only while the failure is being repaired. In this case, the setting of switch S1 becomes important.

CAUTION

The internal modules contain MOS circuitry that is easily damaged by static electricity. The following procedure should be performed only by qualified service personnel familiar with the handling of MOS circuits. Specifically, the case and the technician must be grounded and interconnected. The modules must be handled only by the edges and wrapped in a conductive protective wrap when not installed. This procedure must not be performed in a low humidity environment. Failure to observe these precautions will, at the discretion of John Fluke Mfg. Co., Inc., violate the terms of the warranty as well as any service contract. Refer to the 1720A Instrument Controller Service Manual for additional information.

4A-21. The procedure for using an external terminal as the 1720A console device (KBØ:) is:

1. Turn power OFF.

2. Remove the rear card cage cover (3 screws) using a #1 Phillips screwdriver. Facing the rear, this is the left half-panel.

3. Remove the Video Interface Module and place it in a conductive protective wrap.

4. Slide the CPU Module out approximately 3 cm.

5. Set rotary switch S1 to a baud rate that is compatible with the terminal. See Table 4A-2.

6. Press the CPU Module fully back into place.

7. Install the rear card cage cover.

8. Connect the terminal to serial port 2.

9. Turn Power ON.

10. The terminal will now be addressed by FDOS as KBØ.

Table 4A-2. CPU S1 Settings

POSITION	BAUD RATE	POSITION	BAUD RATE
0	110	4	1200
1	150	5	2400
2	300	6	4800
3	600	7	9600

Section 4B
Main Memory Module

4B-1. INTRODUCTION

4B-2. The Main Memory Module provides a high speed direct access storage medium for system software, user programs, and data. It is constructed of 36 dynamic RAM IC's that each store 16,384 bits, for a total capacity of 65,536 bytes or 32,768 words. Each byte includes an additional parity bit for error detection. Refreshing of the volatile dynamic storage cells is performed by on-board refresh timing circuitry that is transparent to the rest of the system. The module can transfer data in either a byte (8-bit) or word (16-bit) mode. Processor instruction fetches are in word mode, while some types of data are transferred in byte mode.

4B-3. MEMORY ALLOCATION

4B-4. The allocation of memory space is application dependent. The following discussion using the Fluke Enhanced BASIC Interpreter provides a useful perspective.

4B-5. Figure 4B-1 illustrates the general allocation of memory space during a running BASIC program. After studying the memory map, the trade-offs in optimizing use of this resource become apparent. Memory overflow occurs when either the data being collected by the program or an editing step attempts to use memory area in use by the interpreter stack. This stack stores return location and status information needed by the BASIC interpreter. It grows downward as required and retreats upward as tasks are completed. Variables and their data contents being used and collected by the program grow upward from the program area as needed. Should an overflow condition occur during a program run, the following may offer a solution. This discussion covers two possible causes of overflow: program size, and data overflow.

4B-6. Program Size

4B-7. Without allowing for variable data storage, there is room for about 24,000 characters (bytes) in a program. For an average statement length of between 10 and 15 characters, this works out to be somewhere between 1500 and 2400 program statements. If the program is that large, try restructuring it to make greater use of subroutines, or breaking it into separable tasks that can be chained together with RUN statements. Refer to the 1720A Fluke-Enhanced BASIC Programming Manual for techniques.

4B-8. Data Overflow

4B-9. Several options exist for the handling of data. Even if the program is going to modify the data after collection, try opening a channel to a virtual array on the floppy disk, or the optional high speed electronic disk. This will keep the data from accumulating in memory. Movements of large strings (e.g. A$=B$) will temporarily occupy the interpreter stack, an occasional cause of overflow condition during a program run. Also, see that programs do not dimension data arrays larger than the application really requires, and that space consuming string variables are de-allocated when no longer needed.

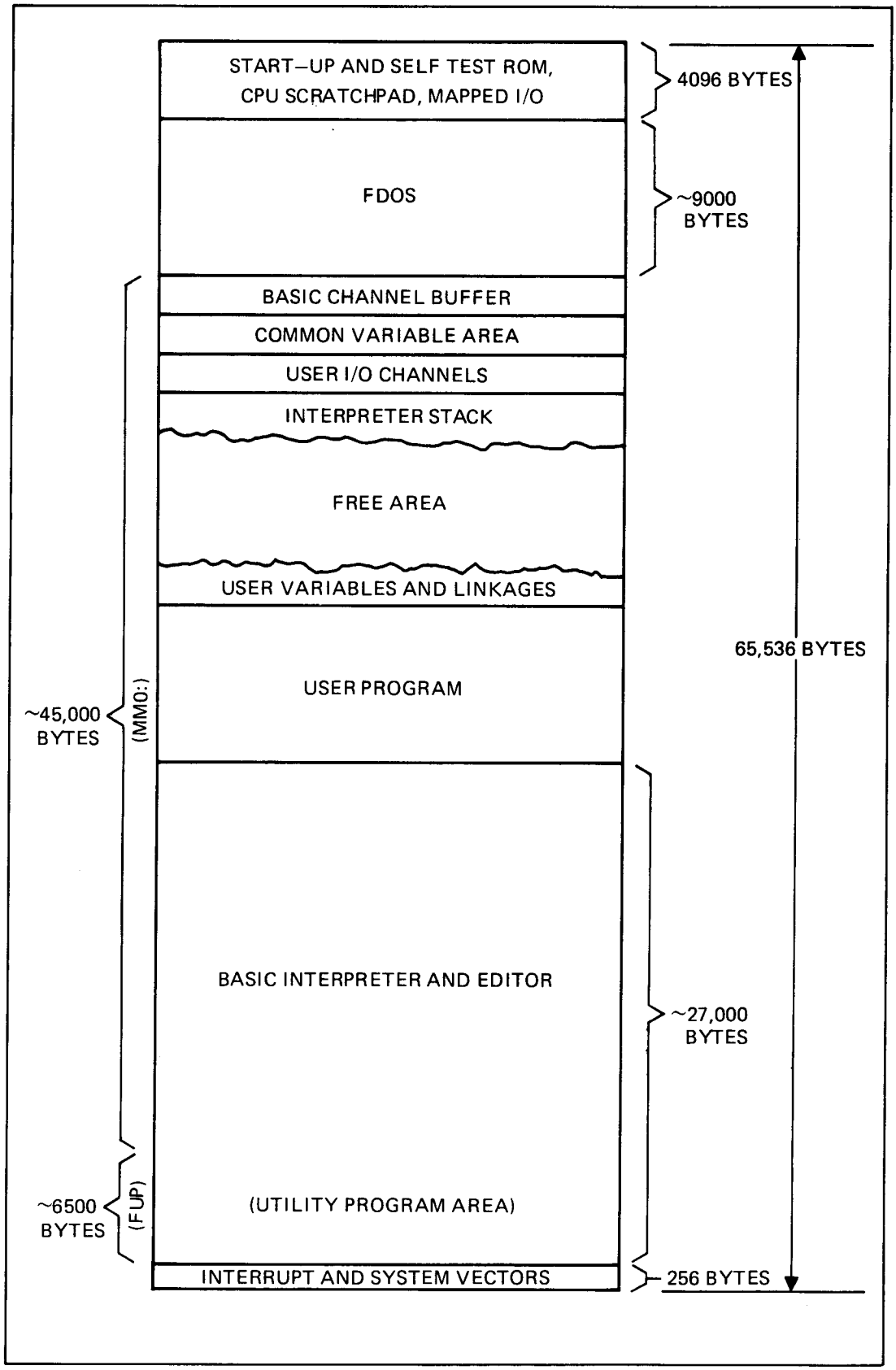


Figure 4B-1. Memory Allocation

Section 4C
Video Interface and Keyboard Module

4C-1. INTRODUCTION

4C-2. The Video Interface Module performs two main functions within the 1720A Instrument Controller:

1. Generates and controls all display functions in response to instructions from the CPU Module.

2. Receives inputs from the programmer keyboard and the touch sensitive panel overlaying the display, and interrupts the CPU module to pass them on.

4C-3. This module runs independent of the CPU Module, except for the passing of data and instructions. The design includes a separate microprocessor with local program and data memory, an LSI display controller, and a high speed on-board display refresh memory. The microprocessor performs a parallel scan of the programmer keyboard and the touch-sensitive display while handling screen formatting and all communications with the CPU module. The LSI display controller scans an on-board display memory, generates dot-patterns from a character ROM, and keeps the screen continually refreshed with information stored in an on-board memory. In graphics mode, character pattern information is supplied to the LSI display controller from alternate circuitry.

4C-4. On-board ROM programming for the microprocessor defines the characteristics and responses of the display to inputs from the user program. In general, these conform to subsets of recognized industrial standards. The discussion that follows covers these standards.

4C-5. ASCII 3.4-1968

4C-6. Table 4C-1 is a chart of ASCII codes as used by the 1720A Instrument Controller display. The left column depicts the actual display response for each ASCII code. Codes numbered decimal 0 through 31 that come from the programmer keyboard (by use of the CTRL key) are passed on to the CPU module unchanged, but are not displayed. In their place is displayed a block character (number 127) unless the code caused the CPU Module to return instructions to clear the screen. Exceptions are control codes C, P, and Z, which are displayed as ^C, ^P, and ^Z respectively.

4C-7. To send the display a code in the range of 0 through 31, it must come from the CPU Module. For example, using BASIC, the statement PRINT CHR$(27) will send the display an ESCape code. Note that the symbols available are different from the graphics mode characters presented in this section. These symbols can be enhanced with such things as high intensity or blinking just as any other character. Graphics mode characters cannot be enhanced.

Table 4C-1. ASCII Display Chart

1720A DISPLAY	DECIMAL	OCTAL	HEX	1720A DISPLAY	DECIMAL	OCTAL	HEX	1720A DISPLAY	DECIMAL	OCTAL	HEX
NUL	0	000	00	,	44	054	2C	X	88	130	58
β	1	001	01	-	45	055	2D	Y	89	131	59
γ	2	002	02	.	46	056	2E	Z	90	132	5A
δ	3	003	03	/	47	057	2F	[	91	133	5B
ε	4	004	04	0	48	060	30	\	92	134	5C
ζ	5	005	05	1	49	061	31	]	93	135	5D
η	6	006	06	2	50	062	32	^	94	136	5E
BEL	7	007	07	3	51	063	33	—	95	137	5F
BS	8	010	08	4	52	064	34	`	96	140	60
HT	9	011	09	5	53	065	35	a	97	141	61
LF	10	012	0A	6	54	066	36	b	98	142	62
VT	11	013	0B	7	55	067	37	c	99	143	63
FF	12	014	0C	8	56	070	38	d	100	144	64
CR	13	015	0D	9	57	071	39	e	101	145	65
◇	14	016	0E	:	58	072	3A	f	102	146	66
π	15	017	0F	;	59	073	3B	g	103	147	67
ρ	16	020	10	<	60	074	3C	h	104	150	68
σ	17	021	11	=	61	075	3D	i	105	151	69
τ	18	022	12	>	62	076	3E	j	106	152	6A
υ	19	023	13	?	63	077	3F	k	107	153	6B
φ	20	024	14	@	64	100	40	l	108	154	6C
χ	21	025	15	A	65	101	41	m	109	155	6D
ψ	22	026	16	B	66	102	42	n	110	156	6E
ω	23	027	17	C	67	103	43	o	111	157	6F
Ω	24	030	18	D	68	104	44	p	112	160	70
↓	25	031	19	E	69	105	45	q	113	161	71
→	26	032	1A	F	70	106	46	r	114	162	72
ESC	27	033	1B	G	71	107	47	s	115	163	73
↑	28	034	1C	H	72	110	48	t	116	164	74
±	29	035	1D	I	73	111	49	u	117	165	75
Σ	30	036	1E	J	74	112	4A	v	118	166	76
≈	31	037	1F	K	75	113	4B	w	119	167	77
SPACE	32	040	20	L	76	114	4C	x	120	170	78
!	33	041	21	M	77	115	4D	y	121	171	79
"	34	042	22	N	78	116	4E	z	122	172	7A
#	35	043	23	O	79	117	4F	{	123	173	7B
$	36	044	24	P	80	120	50	|	124	174	7C
%	37	045	25	Q	81	121	51	}	125	175	7D
&	38	046	26	R	82	122	52	~	126	176	7E
'	39	047	27	S	83	123	53	■	127	177	7F
(	40	048	28	T	84	124	54				
)	41	049	29	U	85	125	55				
*	42	050	2A	V	86	126	56				
+	43	051	2B	W	87	127	57				

4C-8. Figure 4C-1 is a short program that will display all of the displayable characters in double-size format. Type the program while in BASIC, and then type RUN followed by (RETURN) . The block, character number 127, is displayed in place of all control characters that are not displayable. The space character is just left of the !. Touch the screen to clear the display.

```
10   E$=CHR$(27)+"["\BL$=CHR$(127)
20   PRINT E$+"1p";CPOS(2,5);BL$:
30   FOR I=1 TO 6\PRINT CHR$(I);\NEXT I
40   FOR I=7 TO 13\PRINT BL$;\NEXT I
50   FOR I=14 TO 26\PRINT CHR$(I);\NEXT I\PRINT BL$;
60   FOR I=28 TO 31\PRINT CHR$(I);\NEXT I
70   PRINT CPOS(4,5);\FOR I=32 TO 63\PRINT CHR$(I);\NEXT I
80   PRINT CPOS(6,5);\FOR I=64 TO 95\PRINT CHR$(I);\NEXT I
90   PRINT CPOS(8,5);\FOR I=96 TO 127\PRINT CHR$(I);\NEXT I
110  END
```

Figure 4C-1. Character Display Program

4C-9. ANSI X3.64-1977

4C-10. This document of the American National Standards Institute defines some control actions to be associated with certain sequences of ASCII codes. These control sequences are always introduced by an ESCape code, usually followed by a "[" character. When sent to the Video Interface Module from the CPU Module, a control action is initiated by the Video Interface. Table 4C-2 summarizes these control actions by type, and shows the associated code sequence.

Table 4C-2. ANSI Control Sequences

TYPE	ACTION	CODE SEQUENCE
CURSOR CONTROLS	Up n lines	⟨ESC⟩ [n A
	Down n lines	⟨ESC⟩ [n B
	Right n columns	⟨ESC⟩ [n C
	Left n columns	⟨ESC⟩ [n D
	Direct to line, column	⟨ESC⟩ [l ; c H
	Scroll down 1 line	⟨ESC⟩ D
	Scroll up 1 line	⟨ESC⟩ M
	Scroll to start next line	⟨ESC⟩ E
ENHANCEMENTS	Off	⟨ESC⟩ [m
	High intensity	⟨ESC⟩ [1 m
	Underline	⟨ESC⟩ [4 m
	Blinking	⟨ESC⟩ [5 m
	Reverse image	⟨ESC⟩ [7 m
MODES	Normal size	⟨ESC⟩ [p
	Double size	⟨ESC⟩ [1 p
	Graphics off	⟨ESC⟩ [2 p
	Graphics on	⟨ESC⟩ [3 p
	Enable keyboard	⟨ESC⟩ [4 p
	Disable keyboard	⟨ESC⟩ [5 p

Table 4C-2. ANSI Control Sequence (cont)

TYPE	ACTION	CODE SEQUENCE
ERASING	To end of line	⟨ESC⟩ [K
	To start of line	⟨ESC⟩ [1 K
	All of line	⟨ESC⟩ [2 K
	To end of display	⟨ESC⟩ [J
	To start of display	⟨ESC⟩ [1 J
	All of display	⟨ESC⟩ [2 J
CURSOR STATUS	Request cursor position	⟨ESC⟩ [6 n
	Cursor position report	⟨ESC⟩ [l ; c R
EXTERNAL LINE CONTROL	All lines off	⟨ESC⟩ [q
	Line 1 on	⟨ESC⟩ [1 q
	Line 2 on	⟨ESC⟩ [2 q
	Line 3 on	⟨ESC⟩ [3 q
	Line 4 on	⟨ESC⟩ [4 q
	Line 5 on	⟨ESC⟩ [5 q
	Line 6 on	⟨ESC⟩ [6 q
	Line 7 on	⟨ESC⟩ [7 q

4C-11. Cursor Controls

4C-12. Cursor control sequences allow the user to move the display cursor to any position on the screen, either relative to its current position, or absolutely, by designating line and column. Scrolling commands allow movement of the entire display up or down one line at a time when movement goes beyond an edge. A cursor control sequence produces immediate results, and does not occupy a display location. Cursor controls will not move the cursor beyond the display boundaries, except that scrolling commands will cause a one-line scroll when the cursor is already at a top or bottom boundary.

4C-13. The following Fluke BASIC statement will move the cursor right 3 columns. Note the final semicolon to inhibit the usual Carriage Return and Line Feed codes following a BASIC PRINT statement.

```
PRINT CHR$(27)+"[3C";
```

4C-14. Enhancements

4C-15. Display enhancements are used to draw attention to a word or portion of the displayed message. Four different enhancements are available: high intensity, underlining, blinking, and reverse image. Enhancement commands occupy one character position on the display screen and define a display mode for all characters that follow until another enhancement command is encountered. Overwriting a display location containing an enhancement command will delete the enhancement. Each enhancement command cancels all previous enhancements.

4C-16. Multiple enhancements can be used simultaneously by separating the enhancement numbers with semicolons. A multiple enhancement command occupies one display location. For example:

```
PRINT CHR$(27)+'[1;4;5m";
```

4C-17. Since the refresh scanning rate of the display exceeds the 4800 baud rate at which characters are written into the display memory, underlining and reverse image commands

will momentarily cause the entire remaining display to have underlines or light background until the enhancements-off command is written. This problem can be avoided by positioning the cursor to place the enhancements-off command, then backing up to place the desired enhancement command before the start of the words to be enhanced. The use of the Display Worksheet (Part Number 533547, pad of 50) will make positioning the cursor easier. When planning an enhanced display that uses this technique, remember that the cursor spaces forward one place after each enhancement command, just as it does for any other character.

4C-18. Modes

4C-19. Mode selections cover three different areas: character size, the alternate graphics characters, and disabling of keyboard inputs. They may be intermixed with other display control commands, but, as explained below, the sequence can sometimes make a difference. They do not occupy a character position on the display.

4C-20. Character size is changed by controlling the display raster scan timing. For this reason it is available only as an attribute of the entire display. Character size commands clear the screen, set the entire screen to the selected character size, and move the cursor to the upper left (home) position. Normal size is 16 lines of 80 characters each. Double size is 8 lines of 40 characters each.

4C-21. If it is planned to both position the cursor and change the character display size, the program should first change the display size, and then position the cursor. This sequence is necessary because these commands return the cursor to the home position, and because cursor positioning relates to the current size mode.

4C-22. Graphics mode substitutes block graphics characters generated from alternate circuitry for the numbers 0 through 9, and the : character. Table 4C-3 shows the graphics mode characters in normal and double size. When graphics mode is enabled, these characters are displayed from alternate pattern generation circuitry whenever the numbers 0 through 9, or the : are sent to the screen. Because they are not part of the character set in the ROM, they cannot be enhanced.

4C-23. Note the names assigned to each character. Although it is not immediately obvious from looking at the character shapes, there is a correlation between single and double size characters that allow a graphics display to be set up that will change only in size and not in shape as the display mode is changed between normal and double size. This means, for example, that a display subroutine should use a 2 instead of a 6 for a normal size lower right corner. That way if the display is later changed to double size it will still be a lower right corner. Some attention to this detail will produce graphics code subroutines that are useful later for other projects.

4C-24. The program presented in Figure 4C-2 will display all of the graphics characters, first in normal size, and then, when the screen is touched, in double size. Touching the screen again will clear the display. Type it while in BASIC and then type RUN followed by RETURN.

```
10    E$=CHR$(27)+"["\CL$=E$+"2J"
20    PRINT CL$;E$+",3p";CPOS(8,24);\GOSUB 50
30    PRINT CL$;E$+"1;3p";CPOS(4,5);\GOSUB 50
40    PRINT E$+";2p";CL$\GOTO 70
50    PRINT "0 1 2 3 4 5 6 7 8 9 :"
60    WAIT FOR KEY\K$=KEY\RETURN
70    END
```

Figure 4C-2. Graphics Character Display Program

Table 4C-3. Graphics Mode Characters

CHARACTER	NORMAL SIZE	DOUBLE SIZE	CHARACTER NAME
0			Top Right Corner
1			Top Left Corner
2			Bottom Right Corner
3			Bottom Left Corner
4			Top Intersect
5			Right Intersect
6			Left Intersect
7			Bottom Intersect
8			Horizontal Line
9			Vertical Line
:			Crossed Line

NOTES:
1. To enable Graphics Mode, send the display ESC [3p
2. To disable Graphics Mode, send the display ESC [2p
3. In Graphics Mode, characters in the left column are displayed as shown.
4. Use the character names as defined to construct illustrations that do not change form between normal and double size.

4C-25. Erasing

4C-26. Commands are provided to allow a program to erase part or all of any line or of the entire display. Partial erase commands are relative to the current cursor position. An erase control sequence produces immediate results, and does not occupy a display location. For example, the BASIC statement `PRINT CHR$(27)+"[2J"` will clear the display.

4C-27. Cursor Status

4C-28. Two control sequences are implemented. The sequence ⟨ESC⟩[6 n sent from the CPU Module to the Video Interface requests a report of the current cursor position. The Video Interface returns that report to the CPU Module in the form ⟨ESC⟩[l ; c R where l = line number and c = column number.

4C-29. External Line Control

4C-30. Seven external lines are available for software control through control sequences from the CPU Module to the Video Interface. Two of these are located on the

Remote Interface connector on the rear of the Power Supply Module. Five are located in the connector for the programmer keyboard. The drivers of these lines are industry standard TTL, capable of sinking 24 milliamperes, or sourcing 3 milliamperes. The two lines available at the rear Remote Interface connector have an additional 390 ohm series resistor, limiting sink current to approximately 10 milliamperes when used with a red light-emitting diode that is sourced from +5 volts. Table 4C-4 shows the location of each external line by connector and pin number.

Table 4C-4. External LED Lines

LINE #	CONNECTOR	PIN	LINE #	CONNECTOR	PIN
1	Keyboard	7	5	Keyboard	13
2	Keyboard	2	6	Remote Interface	11
3	Keyboard	21	7	Remote Interface	13
4	Keyboard	8			

4C-31. THE TOUCH SENSITIVE DISPLAY

4C-32. The touch-sensitive panel overlaying the face of the display is constructed of two layers of transparent flexible material. The two surfaces of this material that face together are slightly metalized for conductivity, and etched into horizontal and vertical bands. The result is a resistive switch contact matrix that responds to touch pressure. Decoding circuitry in the Video/Keyboard Interface can separately identify any of 60 different areas on the display. The size of each touch sensing area or block is three double-size characters. These blocks are arranged in 6 rows of 10 each, numbered from upper left 1 through 60. The lines that the user sees on the overlay are the etch lines that form the boundaries of each touch sensitive block. A proper touch target is the center of a block.

4C-33. Operating System Software (FDOS) is notified by the Video Interface whenever a touch key has been pressed. The system variable KEY is maintained for use by other software modules, such as BASIC. This variable contains the number of the last touch key pressed, and remains set until it is read. For example, in BASIC, `K%=KEY` reads the system variable KEY.

4C-34. When a touch key is held down, it will repeat the same as a key on the programmer keyboard. The repeat rate is 15 per second. Unlike the programmer keyboard, touch sense keys have no provision for key rollover. This is the feature of a keyboard that allows a second (or more) key to be pressed before lifting the current one. Because touch keys do not have defined edges like a keyboard, multiple key rollover can cause erroneous multiple entries. Consequently, the Video Interface recognizes only one key when multiple keys are pressed.

4C-35. Figure 4C-3 illustrates the display worksheet, available in pads of 50 as Fluke part number 533547. Note that the touch sense blocks are shown overlaying both single and double-size display locations with the corresponding KEY number in the lower right corner of each block.

4C-36. THE PROGRAMMER KEYBOARD

4C-37. The keyboard provided with the 1720A Instrument Controller is designed primarily as a programming tool. The controller can be rack mounted easily because the keyboard is removable and there is an alternate method of operator response input through the touch-sensitive display.

4C-38. The keyboard generates the full 128-character ASCII code set. Not all of the keys are used at this time by the 1720A. This has been done to provide flexibility in future

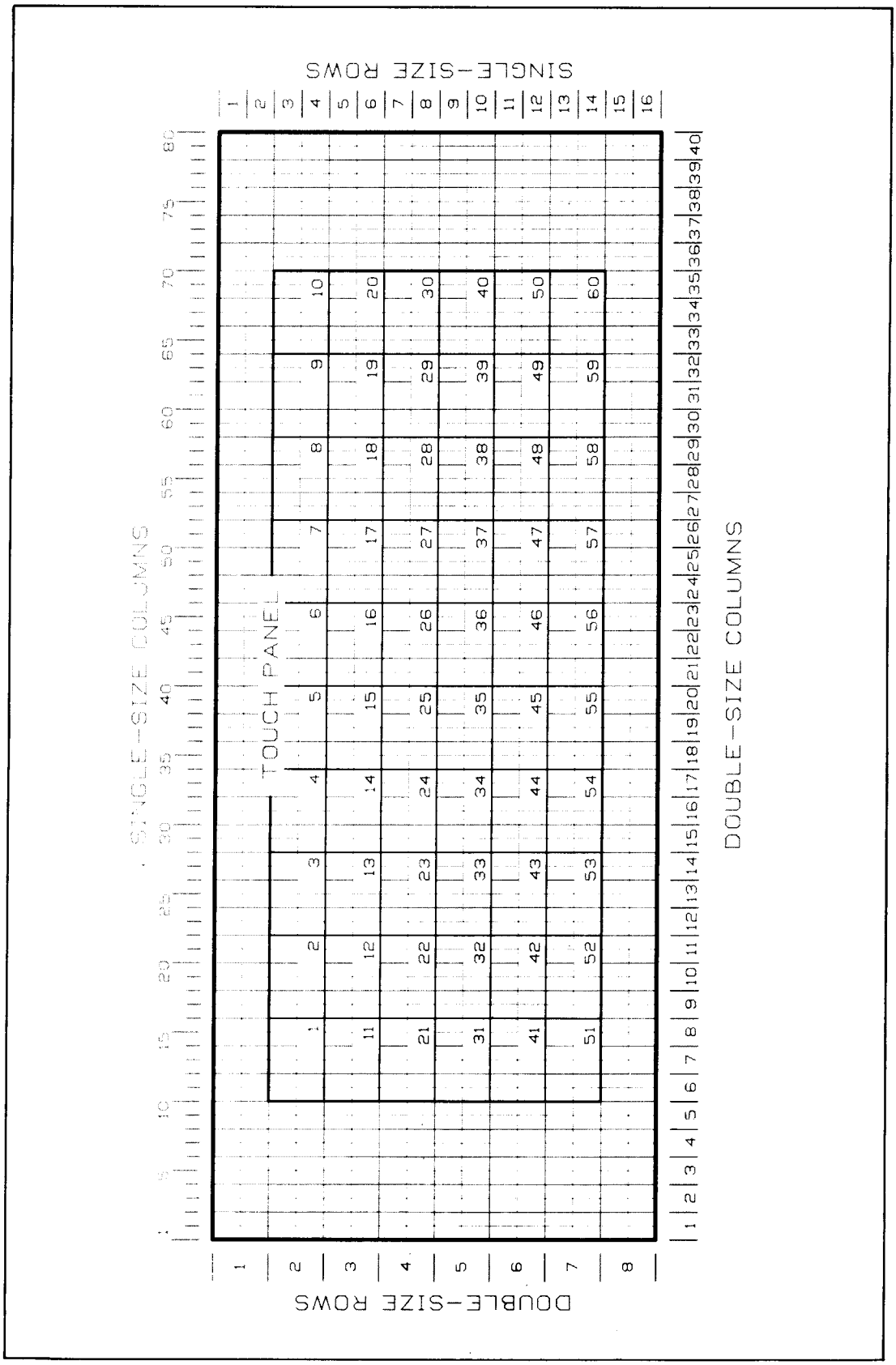


Figure 4C-3. Display Worksheet

applications. Two-key rollover and multiple key lockout is provided for unambiguous key response. All keys (except SHIFT and CTRL) will repeat at 15 per second when held down. An audible clicker identifies acceptance of each keystroke. The back of the keyboard case has provisions for holding its connecting cable. A cutout under the front edge provides a convenient carrying handle. Protective ridges moulded into the case allow the keyboard to be turned face down without damage to the keys.

4C-39. CTRL/SHIFT Keys

4C-40. The Video Interface Module interprets many keys that are modified by both CTRL and SHIFT simultaneously as additional system variable KEY inputs. These inputs are handled by FDOS in the same manner as a touch-sense key input, using the same system variable KEY. This variable contains the number of the last key input received by FDOS, and remains set until read. Table 4C-5 lists the CTRL/SHIFT key inputs and their associated KEY values. Note that touch-sense KEY values are numbered 1 through 60, and CTRL/SHIFT key inputs are numbered 61 and up. For example, in BASIC the statement `IF KEY > 60 THEN GOSUB 5000` will branch only if the last KEY entry was from a CTRL/SHIFT keyboard operation.

Table 4C-5. CTRL/SHIFT Keys

CTRL/SHIFT KEY	KEY NUMBER	CTRL/SHIFT KEY	KEY NUMBER
NEXT PAGE	61	6	91
Z	62	7	92
CAPS LOCK	64	(	93
TAB	65	)	94
Q	66	J	95
ESC	67	K	96
1	68	I	97
X	69	O	98
C	70	8	99
A	71	9	100
S	72	?	101
W	73	PAGE MODE	102
E	74	L	103
2	75	:	104
3	76	P	105
V	77	{	106
B	78	Ø	107
D	79	SPACE	109
F	80	"	111
R	81	\	112
T	82	}	113
4	83	DELETE	114
5	84	+	115
N	85	~	116
M	86	LINE FEED	118
G	87	RETURN	119
H	88	←	122
Y	89	BACK SPACE	123
U	90	↑	124

4C-41. COMPOSITE VIDEO OUTPUT

4C-42. The video signal that drives the 1720A display is also made available through a rear-mounted connector. This signal is intended to drive a standard data display video monitor that conforms to EIA standard RS-170. This signal complies with the timing specifications of EIA RS-170, except that equalization and serration pulses are not provided during vertical retrace. Data display monitors normally do not require this.

Section 4D
Floppy Disk and Interface

4D-1. INTRODUCTION

4D-2. The Floppy Disk Interface Module, together with the Floppy Disk Drive, provide the user with a file structured bulk storage medium that can be removed for a permanent record. Since system software is initially loaded from disk, updating a system to take advantage of newer software is simply a matter of inserting a different System Disk. The portability afforded to user programs allows the development ofstandardized test procedures that work identically for every instrument controller. The non-volatility of a disk allows the creation of permanent records for such applications as a calibration laboratory.

4D-3. In addition, the Floppy Disk Interface Module includes the Real-Time Clock used in numerous ways where time-of-day is required. The clock is discussed later in this subsection.

4D-4. DATA ORGANIZATION

4D-5. Programs and data on a floppy disk are organized as a sequence of 350 (512-byte) blocks. A directory is maintained in the first two blocks that enables FDOS to locate each file by its assigned name. The disk interface converts the sequential block number to a track and sector address.

4D-6. A track is one of 35 distinct radial distances from the center of the disk and is accessed by step movements of the read/write head. Data on the disk is therefore organized into concentric circles. Each track contains ten blocks or 5120 bytes of data.

4D-7. There are 10 pie-shaped radial subdivisions on a disk that divide each track into 10 sectors. An index hole punched in the disk identifies the start point for numbering the sectors. The 10 sectors on each track each contain one block or 512 bytes or data.

4D-8. Any block of data on a floppy disk can be located by positioning the head to the required track and reading sector numbers until the required one is found. Formatting a disk, one of the FUP functions presented earlier, is the process of preparing a disk for data by writing track and sector identification numbers, followed by a standard data pattern, throughout the disk surface. When reading or writing data on the disk, the DMA Floppy Interface always positions the head in or out the necessary number of steps. It then compares both track and sector identification from the disk with the desired track and sector address. When a match is found, it proceeds with the read or write operation.

4D-9. The largest file that can be put on a disk is the full capacity less two blocks for the directory. That works out to (348 blocks) X (512 bytes/block) or 178,176 bytes maximum file size. The smallest file that can be put on a disk is one block, or 512 bytes. However, the maximum number of files is limited by the size of the directory to 72.

4D-10. FILE ACCESS TIME

4D-11. The Floppy Disk Interface Module uses a direct memory access (DMA) technique to transfer programs and data to and from the Main Memory Module. Processor control of the internal bus is intermittently suspended in favor of the disk interface for a direct memory byte transfer. The result is a block transfer rate of 15,625 bytes per second, and an average transfer rate over a file of 25 blocks per second. This is the full transfer rate of the disk drive itself, as the DMA process easily outpaces the drive.

4D-12. If 16 seconds elapse during which no disk access has been made, the drive motor is turned off to extend its life as well as that of the head and the disk. The first access after a turn-off is delayed approximately one second while the disk comes up to speed.

4D-13. In addition, there is an average file seek time before data transfer starts, as the head moves to the proper track and locates the desired data block. Based upon an average head movement of 17.5 tracks, and an average rotation of 180 degrees, this time is about 0.55 second.

4D-14. FLOPPY DISKS

4D-15. Fluke makes available ten-packs of unformatted blank floppy disks which have been manufactured and certified to Fluke standards.

NOTE

Performance of the 1720A Controller to published specifications can be ensured only by using Fluke Accessory Number Y1706 floppy disks, or a Fluke-approved equivalent. Contact any Fluke Sales or Service Office for information.

4D-16. Refer to Figure 4D-1. The disk drive has a sense switch along each side of the disk slot. When fully inserted, one of them falls into the write enable notch, unless it has

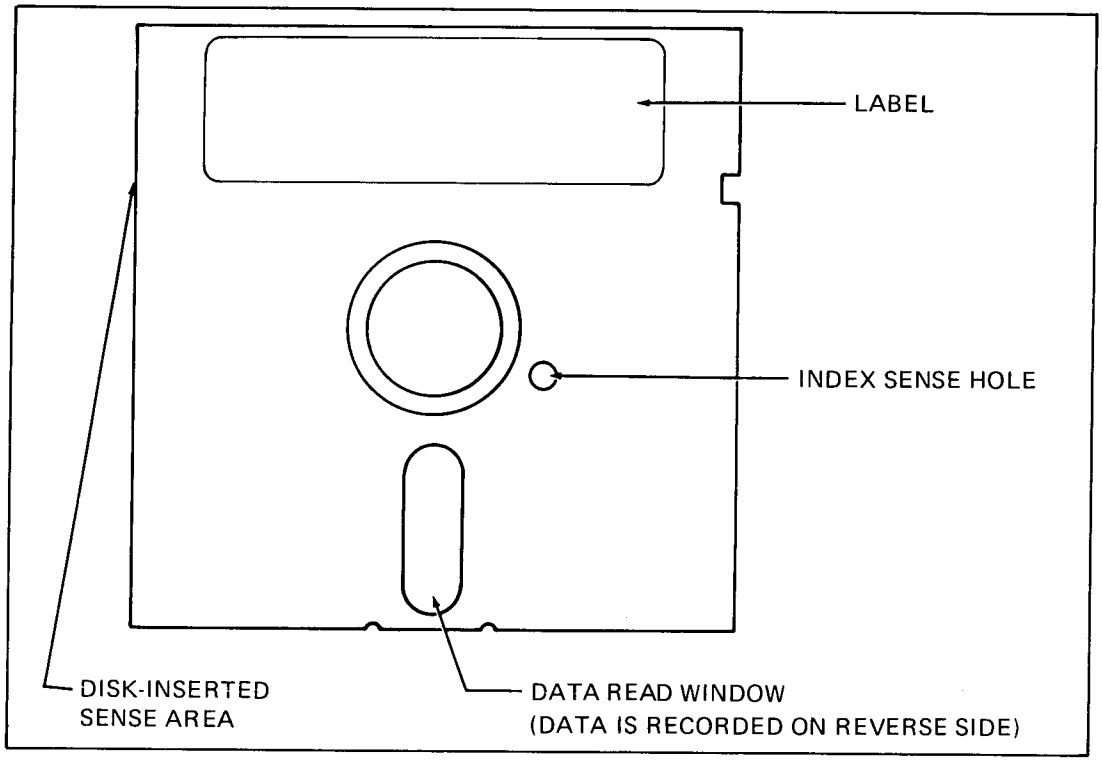


Figure 4D-1. Floppy Disk

been covered over with a write protect tab. The other switch along the opposite side is used to sense the presence of the disk.

4D-17. Three floppy disks are supplied with the 1720A Instrument Controller. The System Disk contains FDOS, a programming language, and the standard set of utility programs. The Diagnostic Disk contains the programs that are used by the diagnostic procedures in the 1720A Service Manual. The Data Disk is a formatted blank disk.

4D-18. THE REAL-TIME CLOCK

4D-19. The Floppy Disk Interface Module includes a Real-Time Clock that allows a program to label data files with date and time-of-day, to measure elapsed time, and to display or use this information in other ways. In addition, FDOS labels each directory entry with the date the file was created.

4D-20. Refer to the discussion in Section 3 on the TIME utility program for setting this function with the proper date and time. Once set, it is correct until February 29 of the next leap year, when it will have to be manually set. The internal battery will keep the clock running at least two months before requiring recharge, if the power is turned off for that long.

4D-21. Note that if an external battery is used the internal battery should be disconnected from the system. Whenever the external battery is disconnected, the clock will turn off and require resetting.

4D-22. The system can operate with the internal battery disconnected and without an external battery. In this case, it is necessary to reset the time and date, using the TIME utility, whenever the system is tuned on.

4D-23. An added feature of the Real Time Clock is a status flag that is set whenever power is first applied, and reset whenever the time and date are set. The flag is checked whenever the TIME utility is entered from within an active system command file. If it is reset, control is returned directly to CONMON, and to the next system command file line, without requiring any input. This allows a system command file to be set up with a T command that will not require a keyboard response unless power to the clock had been interrupted.

Section 4E
Optional Electronic Disk Modules

4E-1. INTRODUCTION

4E-2. The optional electronic disk is a one or two module memory system designed to provide fast access file storage. It functions just like the floppy disk, differing only in capacity and speed. Electronic disk files must be transferred to the floppy disk for permanent storage. Like a disk, it contains a file directory and must be formatted (using FUP) before it can be used. Program files must be transferred to the Memory Module before they can be executed. Data files can be directly accessed through virtual array channels.

4E-3. DATA ORGANIZATION

4E-4. Programs and data on the electronic disk are organized as a sequence of 256 512-byte blocks. Adding the second module, the maximum allowable, expands this to 512 blocks. This second module results in a larger electronic disk, rather than two of them. This relieves concern over which electronic disk contains a given file, and allows the storage of files larger than one electronic disk. A two-module system can handle a file larger than the capacity of the floppy disk. A single directory in the first two blocks of the first module functions as the directory for both modules.

4E-5. The largest file that can be put on the electronic disk is the full capacity less two blocks for the directory. For one module, that works out to (254 blocks) X (512 bytes/block) or 130,048 bytes. A two-module system has a maximum file capacity of (510 blocks) X (512 bytes/block) or 261,120 bytes.

4E-6. The smallest file that can be put on the electronic disk is one block, or 512 bytes. However, the number of files is limited by the size of the directory to a maximum of 72.

4E-7. Although a single module is 73% of the capacity of the floppy disk, it can be used to create back-up copies of floppy disks as long as no single file exceeds 254 blocks (130,048 bytes) in size. See the discussion of the FUP /W command in Section 3.

4E-8. An electronic disk is needed to create back-up floppy disks whenever any single-file size exceeds the memory capacity available for this function (approximately 85 blocks). A two-module electronic disk is needed to create back-up floppy disks whenever any single-file size exceeds 254 blocks (130,048 bytes).

4E-9. FILE ACCESS TIME

4E-10. Access time to an electronic disk file before data transfer can proceed is 18 microseconds. Thereafter data transfer proceeds at the rate of 260 blocks (133,120 bytes) per second, or at the rate the other device can accept the transfer. For a program load, which proceeds at full speed, that is equivalent to filling the entire contents of the Main Memory Module in less than 0.5 second.

4E-11. BATTERY BACK-UP

4E-12. The electronic disk will hold its contents intact when the 1720A Controller is powered down, provided the E-DISK BATTERY switch on the power supply is set to ENABLE and there is an internal or external source of battery power. A fully charged internal battery will support a single electronic disk module for approximately 1 hour or two electronic disk modules for about 30 minutes. Voltage sensing circuitry in the power supply determines when the battery has significantly discharged, and disconnects it from electronic disk back-up so that the internal time clock may continue to run.

4E-13. If battery back-up of the electronic disk will not be used, battery life can be extended by leaving the feature disabled. This prevents the battery from having to discharge and recharge regularly.

4E-14. SWITCH SETTINGS

4E-15. Electronic disk modules are identical except for a switch that identifies the module as first or second. They may be installed in either or both of the spare circuit board slots. The only requirement is that a single module must be designated as first, and that two modules must not be designated the same. Refer to Figure 4E-1, E-Disk Switch Settings.

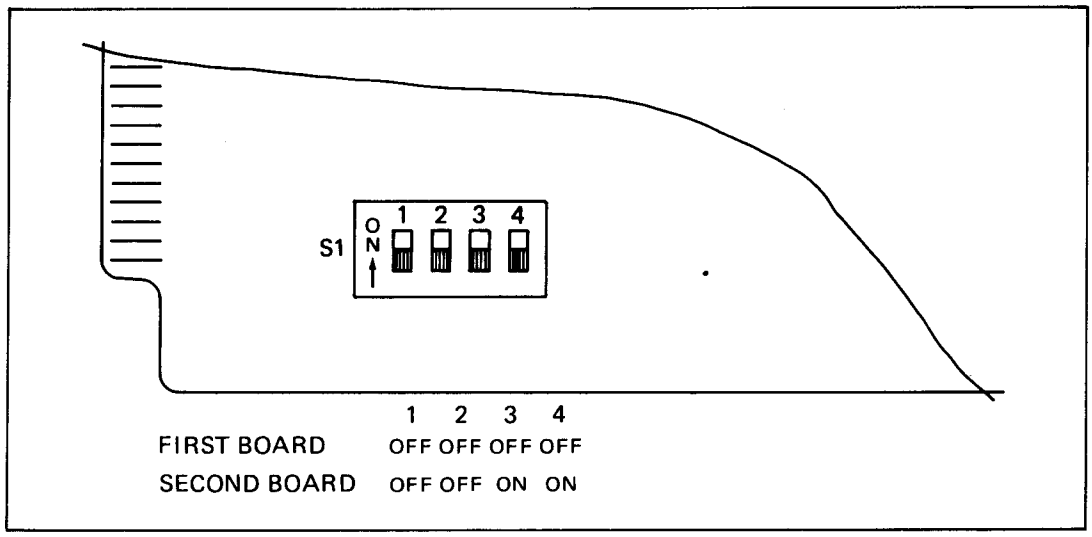


Figure 4E-1. E-Disk Select Switch Position

Section 4F
Dual IEEE-488 Interface Module

4F-1. INTRODUCTION

4F-2. Industrial instrumentation control is the primary design focus of the 1720A Instrument Controller. The dual IEEE-488-1978 (GPIB) interface supplied standard with each controller enables the 1720A to effectively function as two independent controllers for as many as 28 instruments on different IEEE-488 busses.

4F-3. FEATURES

4F-4. The availability of two separate IEEE-488 buses allows instruments to be isolated from each other. This can be a significant advantage in a number of ways. The major advantage is when the bus must connect to a device being tested. Separate buses allow the suspect device to be functionally separated from other instrumentation participating in the test. In addition, there is often a noticable speed advantage in separating devices that respond slowly to bus protocols from higher speed instruments.

4F-5. All standard controller functions defined in subsets C1 through C4 and C6 of the IEEE-488-1978 standard are implemented. Data transfer can be in any code acceptable to system software. Each port is switch set to System Controller, able to assert Interface Clear. Each port also includes readback circuitry for diagnostic self-testing.

4F-6. With current system software, this interface is designed to function as System Controller. The 1720A will not function as a listener or talker without also being System Controller. The 1720A also will not pass control to another controller on the same bus.

4F-7. Table 4F-1 lists ASCII codes and their corresponding Bus messages. Refer also to the appropriate programming language manual for Bus message sequences that have been standardized for instrument control.

Table 4F-1. ASCII/IEEE-488-1978 Bus Codes

ASCII	DECIMAL	OCTAL	HEX	BINARY 7654 3210	MESSAGE (ATN = TRUE)		DEV. NO.	ASCII	DECIMAL	OCTAL	HEX	BINARY 7654 3210	MESSAGE (ATN = TRUE)		DEV. NO.
NUL	0	000	00	0000 0000	ADDRESSED COMMANDS	——		@	64	100	40	0100 0000	TALK ADDRESSES	MTA	0
SOH	1	001	01	0000 0001		GTL		A	65	101	41	0100 0001		MTA	1
STX	2	002	02	0000 0010		——		B	66	102	42	0100 0010		MTA	2
ETX	3	003	03	0000 0011		——		C	67	103	43	0100 0011		MTA	3
EOT	4	004	04	0000 0100		SDC		D	68	104	44	0100 0100		MTA	4
ENQ	5	005	05	0000 0101		PPC		E	69	105	45	0100 0101		MTA	5
ACK	6	006	06	0000 0110		——		F	70	106	46	0100 0110		MTA	6
BEL	7	007	07	0000 0111		——		G	71	107	47	0100 0111		MTA	7
BS	8	010	08	0000 1000		GET		H	72	110	48	0100 1000		MTA	8
HT	9	011	09	0000 1001		TCT		I	73	111	49	0100 1001		MTA	9
LF	10	012	0A	0000 1010		——		J	74	112	4A	0100 1010		MTA	10
VT	11	013	0B	0000 1011		——		K	75	113	4B	0100 1011		MTA	11
FF	12	014	0C	0000 1100		——		L	76	114	4C	0100 1100		MTA	12
CR	13	015	0D	0000 1101		——		M	77	115	4D	0100 1101		MTA	13
SO	14	016	0E	0000 1110		——		N	78	116	4E	0100 1110		MTA	14
SI	15	017	0F	0000 1111		——		O	79	117	4F	0100 1111		MTA	15
DLE	16	020	10	0001 0000	UNIVERSAL COMMANDS	——		P	80	120	50	0101 0000		MTA	16
DC1	17	021	11	0001 0001		LLO		Q	81	121	51	0101 0001		MTA	17
DC2	18	022	12	0001 0010		——		R	82	122	52	0101 0010		MTA	18
DC3	19	023	13	0001 0011		——		S	83	123	53	0101 0011		MTA	19
DC4	20	024	14	0001 0100		DCL		T	84	124	54	0101 0100		MTA	20
NAK	21	025	15	0001 0101		PPU		U	85	125	55	0101 0101		MTA	21
SYN	22	026	16	0001 0110		——		V	86	126	56	0101 0110		MTA	22
ETB	23	027	17	0001 0111		——		W	87	127	57	0101 0111		MTA	23
CAN	24	030	18	0001 1000		SPE		X	88	130	58	0101 1000		MTA	24
EM	25	031	19	0001 1001		SPD		Y	89	131	59	0101 1001		MTA	25
SUB	26	032	1A	0001 1010		——		Z	90	132	5A	0101 1010		MTA	26
ESC	27	033	1B	0001 1011		——		[	91	133	5B	0101 1011		MTA	27
FS	28	034	1C	0001 1100		——		\	92	134	5C	0101 1100		MTA	28
GS	29	035	1D	0001 1101		——		]	93	135	5D	0101 1101		MTA	29
RS	30	036	1E	0001 1110		——		^	94	136	5E	0101 1110		MTA	30
US	31	037	1F	0001 1111		——		—	95	137	5F	0101 1111		UNT	
SP	32	040	20	0010 0000	LISTEN ADDRESSES	MLA	0	`	96	140	60	0110 0000	SECONDARY ADDRESSES	MSA	0
!	33	041	21	0010 0001		MLA	1	a	97	141	61	0110 0001		MSA	1
"	34	042	22	0010 0010		MLA	2	b	98	142	62	0110 0010		MSA	2
#	35	043	23	0010 0011		MLA	3	c	99	143	63	0110 0011		MSA	3
$	36	044	24	0010 0100		MLA	4	d	100	144	64	0110 0100		MSA	4
%	37	045	25	0010 0101		MLA	5	e	101	145	65	0110 0101		MSA	5
&	38	046	26	0010 0110		MLA	6	f	102	146	66	0110 0110		MSA	6
'	39	047	27	0010 0111		MLA	7	g	103	147	67	0110 0111		MSA	7
(	40	048	28	0010 1000		MLA	8	h	104	150	68			MSA	8
)	41	049	29	0010 1001		MLA	9	i	105	151	69	0110 1001		MSA	9
*	42	050	2A	0010 1010		MLA	10	j	106	152	6A	0110 1010		MSA	10
+	43	051	2B	0010 1011		MLA	11	k	107	153	6B	0110 1011		MSA	11
,	44	054	2C	0010 1100		MLA	12	l	108	154	6C	0110 1100		MSA	12
-	45	055	2D	0010 1101		MLA	13	m	109	155	6D	0110 1101		MSA	13
.	46	056	2E	0010 1110		MLA	14	n	110	156	6E	0110 1110		MSA	14
/	47	057	2F	0010 1111		MLA	15	o	111	157	6F	0110 1111		MSA	15
0	48	060	30	0011 0000		MLA	16	p	112	160	70	0111 0000		MSA	16
1	49	061	31	0011 0001		MLA	17	q	113	161	71	0111 0001		MSA	17
2	50	062	32	0011 0010		MLA	18	r	114	162	72	0111 0010		MSA	18
3	51	063	33	0011 0011		MLA	19	s	115	163	73	0111 0011		MSA	19
4	52	064	34	0011 0100		MLA	20	t	116	164	74	0111 0100		MSA	20
5	53	065	35	0011 0101		MLA	21	u	117	165	75	0111 0101		MSA	21
6	54	066	36	0011 0110		MLA	22	v	118	166	76	0111 0110		MSA	22
7	55	067	37	0011 0111		MLA	23	w	119	167	77	0111 0111		MSA	23
8	56	070	38	0011 1000		MLA	24	x	120	170	78	0111 1000		MSA	24
9	57	071	39	0011 1001		MLA	25	y	121	171	79	0111 1001		MSA	25
:	58	072	3A	0011 1010		MLA	26	z	122	172	7A	0111 1010		MSA	26
;	59	073	3B	0011 1011		MLA	27	{	123	173	7B	0111 1011		MSA	27
<	60	074	3C	0011 1100		MLA	28	|	124	174	7C	0111 1100		MSA	28
=	61	075	3D	0011 1101		MLA	29	}	125	175	7D	0111 1101		MSA	29
>	62	076	3E	0011 1110		MLA	30	~	126	176	7E	0111 1110		MSA	30
?	63	077	3F	0011 1111		UNL		DEL	127	177	7F	0111 1111			

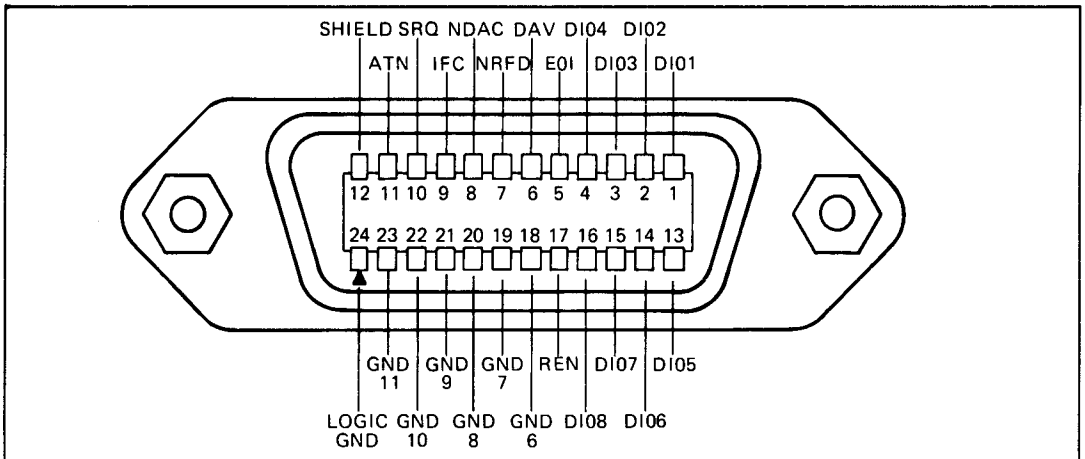
4F-8. Table 4F-2 lists the pin connections for the IEEE-488 instrument ports. These connections can be functionally divided into three groups: the Data Bus, the Management Bus, and the Transfer Bus.

4F-9. The Data Bus contains eight bidirectional active-low signal lines, DIO1 through DIO8. One byte of information (eight bits) is transferred over the bus at a time. DIO1 is least significant; DIO8 is most significant. Each byte represents a peripheral address (either primary or secondary), a control word, or a data byte.

4F-10. The Management Bus is a group of five signal lines which are used to control data transfers over the Data Bus. The Transfer Bus performs a handshake sequence executed by the talker and the listeners each time a byte is transferred over the Data Bus.

4F-11. Table 4F-3 lists the memonic abbreviations that are used in this subsection.

Table 4F-2. IEEE-488-1978 Instrument Ports (Port 0 and Port 1)



BUS	PIN	SIGNAL	DEFINITION
DATA BUS	1 2 3 4 13 14 15 16	DIO1 DIO2 DIO3 DIO4 DIO5 DIO6 DIO7 DIO8	Data input and output lines, bidirectional and active-low, DIO8 is most significant. Data transfers are 8-bit parallel and byte serial.
MANAGEMENT BUS	11 23	ATN ATN Return	Attention. Activiated by the 1720A when peripheral devices are being assigned as listeners and talkers. The 1720A assumes it is the only source of this signal.
	10 22	SRQ SRQ Return	Service request. Any peripheral device on the Instrument Bus can request the attention of the 1720A Controller by setting SRQ active low.
	9 21	IFC IFC Return	Interface clear. Set by the 1720A to place all instruments on the bus in a predetermined reset state. The 1720A assumes that it is the only source of this signal.
	17	REN	Remote enable. Causes all responding instruments on the Bus to ignore their front panel controls and operate under remote control via signals and control messages received over the Bus.

Table 4F-2. IEEE-488-1978 Instrument Ports (Port 0 and Port 1) (cont)

BUS	PIN	SIGNAL	DEFINITION
MANAGEMENT BUS (cont)	5	EOI	End or identify. Can be used by a talker to identify the end of a data transfer sequence, or with ATN by a Controller, to execute a polling sequence.
TRANSFER BUS	7 19	NRFD NRFD Return	Not ready for data. An active low signal line to indicate that one or more assinged listeners are not ready to receive the next data byte. When all of the assigned listeners for a particular data transfer have released NRFD, the NRFD line goes inactive high. The talker can then place the next data byte on the Data Bus.
	6 18	DAV DAV Return	Data valid. Activated by the talker shortly after the placing a valid data byte on the Data Bus. An active low DAV signal tells each listener to capture the data byte presently on the Data Bus. The talker should be inhibited from activating DAV when NRFD is active low.
	8 20	NDAC NDAC Return	Not data accepted. Held active low by each listener until the listener captures the data byte currently being transmitted over the Data Bus. When all listeners have captured the data byte, NDAC goes inactive high. This tells the talker the transfer is complete.

Table 4F-3. ASCII and IEEE-488 Mnemonic Abbreviations

ACK	Acknowledge	DLE	Data Link Escape
ASCII	American Standard Code for Information Interchange	ENQ	Enquiry
		EOF	End of File
ATN	Attention	EOI	End or Indentify
BEL	Bell	EOT	End of Transmission
BS	Backspace	ESC	Escape
CAN	Cancel	ETB	End of Transmission Block
CR	Carriage Return	ETX	End of Text
DCL	Device Clear	FF	Form Feed
DCn	Device Control 1, 2, 3, or 4	GET	Group Execute Trigger
DEL	Delete	GND	Ground
DIOn	Data Input/Output 1 through 8	GTL	Go To Local

Table 4F-3. ASCII and IEEE-488 Mnemonic Abbreviations (cont)

HT	Horizontal Tab	PPRn	Parallel Poll Response 1 through 8
IEEE	Institute of Electrical and Electronic Engineers	PPU	Parallel Poll Unconfigure
		REN	Remote Enable
IFC	Interface Clear	RS	Record Separator
LF	Line Feed	SDC	Selected Device Clear
LLO	Local Lockout	SI	Shift In
MLA	My Listen Address	SO	Shift Out
MSA	My Secondary Address	SOH	Start of Heading
MTA	My Talk Address	SP	Space
NAK	Negative Acknowledge	SPD	Serial Poll Disable
NDAC	Not Data Accepted	SPE	Serial Poll Enable
NRFD	Not Ready For Data	SRQ	Service Request
NUL	Null	STB	Status Byte
OSA	Other Secondary Address	STX	Start of Text
OTA	Other Talk Address	TCT	Take Control
PCG	Primary Command Group	UNL	Unlisten
PPC	Parallel Poll Configure	UNT	Untalk
PPD	Parallel Poll Disable	US	Unit Separator
PPE	Parallel Poll Enable		

Section 4G
Remote Interface

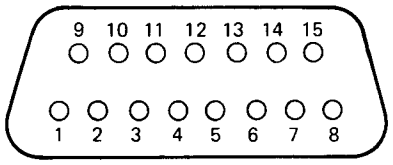
4G-1. INTRODUCTION

4G-2. The Remote Interface Connector provided on the rear power supply panel allows the user to do any of the following:

- Disable the front panel controls.
- Remotely perform front panel switch functions.
- Connect a larger capacity external battery.
- Halt the processor at the completion of the current machine instruction.
- Gain access to two software controlled external lines.
- Separate the battery back-up function so that it responds only to power failures and not to power turn-off by an exernal system power switch.

4G-3. Table 4G-1 lists these functions and their associated connector pin numbers.

Table 4G-1. Remote Interface Connector (Rear Power Supply)



PIN	NAME	MEANING
1	ABORT SW	The front panel ABORT switch applies a ground to this pin when pressed. If the internal jumper connecting pins 1 and 2 is cut, the front panel ABORT switch is enabled by connecting this pin to pin 2.
2	ABORT ENABLE	The ABORT function is available on this pin. If the internal jumper connecting pins 1 and 2 is cut, the front panel ABORT switch is enabled by connecting this pin to pin 1. Applying ground (pin 7, 8, or 15) to this pin will cause an ABORT.

Table 4G-1. Remote Interface Connector (Rear Power Supply) (cont)

PIN	NAME	MEANING
3	RESTART SW	The front panel RESTART switch applies a ground to this pin when pressed. If the internal jumper connecting pins 3 and 4 is cut, the front panel RESTART switch is enabled by connecting this pin to pin 4.
4	RESTART ENABLE	The RESTART function is available on this pin. If the internal jumper connecting pins 3 and 4 is cut, the front panel RESTART switch is enabled by connecting this pin to pin 3. Applying ground (pin 7, 8, or 15) to this pin will cause a RESTART.
5	EXT. BATT.	Connect the positive terminal of an external battery here, if you need to extend the E-Disk back-up time provided by the internal battery. Any DC voltage between 6 and 12 may be used. Before using this function: remove the power cord, wait 5 minutes, remove the power supply, and unplug the internal battery.
6	HALT	CPU Halt. Applying ground (pin 7, 8, or 15) to this pin will halt the processor at the completion of the current machine instruction, and execute a LOAD sequence. Unless HDT software is available (see the 1720A Service Manual), an error message will result.
7	GROUND	One of three available ground pins.
8	GROUND	One of three available ground pins.
9	RESERVED	
10	N/C	
11	LED #6	A software-controlled output line designed to drive a Light-Emitting Diode. See section 4C.
12	+5V	Five volts from the power supply. Use this to supply positive voltage for LEDs on pins 11 and 13.
13	LED #7	A software-controlled output line designed to drive a Light-Emitting Diode. See section 4C.
14	BATTERY DISABLE	A ground applied to this line will disable the battery back-up function. Use this to prevent regular discharage of the battery when the power is applied and remove externally by a common power switch.
15	GROUND	One of three available ground lines.

List of Software Versions
SET 2.0

1720A
Instrument Controller

Set RS-232 Utility User Manual

P/N 633131
December 1981

FLUKE®

Table of Contents

INTRODUCTION

The Set RS-232 Utility (SET) is a machine-language utility program on the System Disk with the file name SET.CIL. It allows the user to reconfigure the parameters used by FDOS in communicating through each of the two RS-232-C serial ports.

NOTE

The software supporting the RS-232 serial ports is not intended to handle binary data. Line and file terminators, as well as other control codes, are tested for and used to identify specific operations.

CONVENTIONS USED IN THIS MANUAL

The following paragraphs describe conventions used in this manual. Individual sections of this manual may define additional conventions.

Command Definitions

Each 1720A command is defined in a standard format that allows a maximum of 80 characters in any command line.

- The command is named on a title line.
- A syntax diagram or syntax statement follows the title line.
- The command definition follows the syntax diagram or syntax statement.
    1. The body of the command definition is a short paragraph that describes the basic function of that command.
    2. Amplifying information is separated by item and organized in a top down outline format similar to this description.
    3. A cross-reference to any associated information or documents ends the command definition.

Syntax Diagrams

Syntax diagrams define correct spelling, punctuation, and sequence of words, symbols, and expressions for system and utility commands. The following guidelines define proper use of these diagrams:

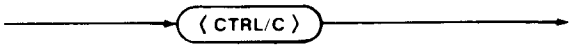
- Any path through a diagram starting from the left that does not run contrary to an arrowhead forms a legitimate command construct. The text accompanying the diagram explains legal usage.

- Boldface words in a circular enclosure are to be entered exactly as shown. Example:

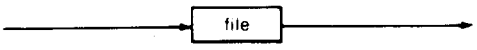
- Key entries with names, such as ESC or RETURN, are shown in a box with rounded corners. Example:

- A required space character entry is always shown as:

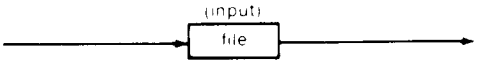
- Control character entries are shown in circular enclosures within angle brackets. The representation CTRL/ means to hold the CTRL key depressed while typing the character that follows. Example:



- Lower case words enclosed in a box represent other information to be supplied.



- Words outside the path of the diagram, usually in parentheses, provide supplementary information. These words are normally not part of the definition of the statement. Example:

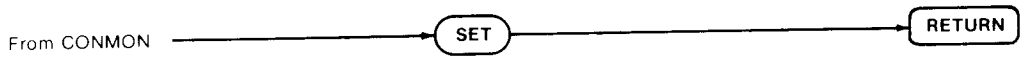


Notation Conventions

There are several notation conventions used in the text and in the examples.

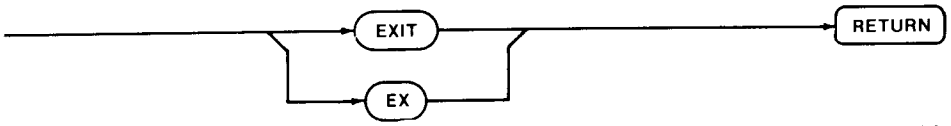
- 0 <XX> is understood to mean press key XX.
  1. <cr> is understood to mean a carriage return (the RETURN key).
  2. <lf> is understood to mean line feed.
- 0 [xxx] is understood to mean that xxx is an optional input.
- 0 {xxx} is understood to mean that xxx is a required input.

ENTERING THE SET RS-232 UTILITY



SET is accessible through CONMON by typing SET.

EXITING THE SET RS-232 UTILITY



SET may be exited at any time by entering EXIT or EX in response to the SET prompt *.

USING THE SET RS-232 UTILITY

SET displays the message:

```
SET (time)(date)
Set Version <x.y>

*
```

NOTE

Verify that the version number <x.y> is the same as on the front of this manual. If it is not, contact a Fluke Customer Service Center for advice.

After first specifying a port, SET allows the user to set any of the following six characteristics for the port:

Baud Rate	Number of Stop Bits
Number of Data Bits	End of Line Character
Parity	End of File Character

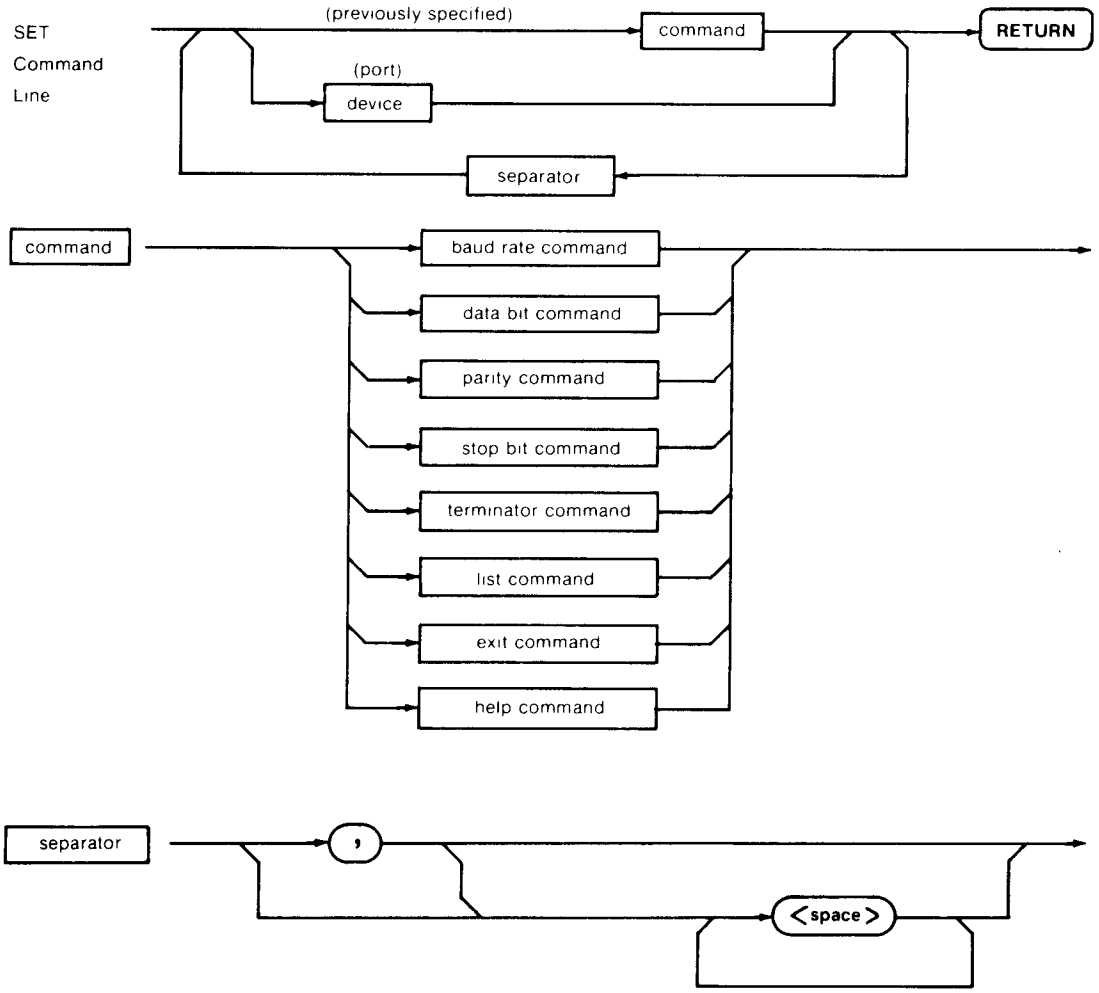
In addition, SET displays the current configuration of the selected port or a summary of SET commands.

SET Command Structure

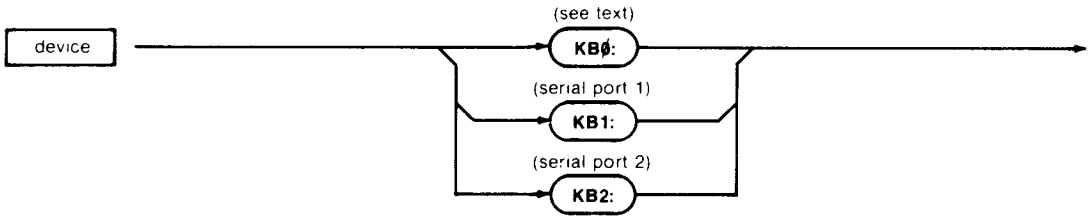
The SET utility program uses a flexible and straightforward command structure. After selecting a device, the user may list the current settings and change selected ones in any order.

- Commands may be entered singly or may be combined into a multiple command line.
- Both upper and lower case entries are accepted.
- Commands become effective when the command line is terminated by <cr>.
- All SET commands may be automated through command files.
- Parameters controlled by SET may be independently set for each serial port.

The following syntax diagrams define the structure of SET commands:



Device Specification



SET requires that a port device be selected prior to accepting other commands.

- Once selected, subsequent commands affect the specified port until a different port is selected.

- When a device is selected other than KB0:, KB1:, or KB2:, the following display appears:

  ?Unknown command

- When KB2: is selected while an external terminal is in use, the following display appears (see the discussion of KB0 that follows):

  ?Illegal device

- When a command is entered before a port is selected, the following display appears:

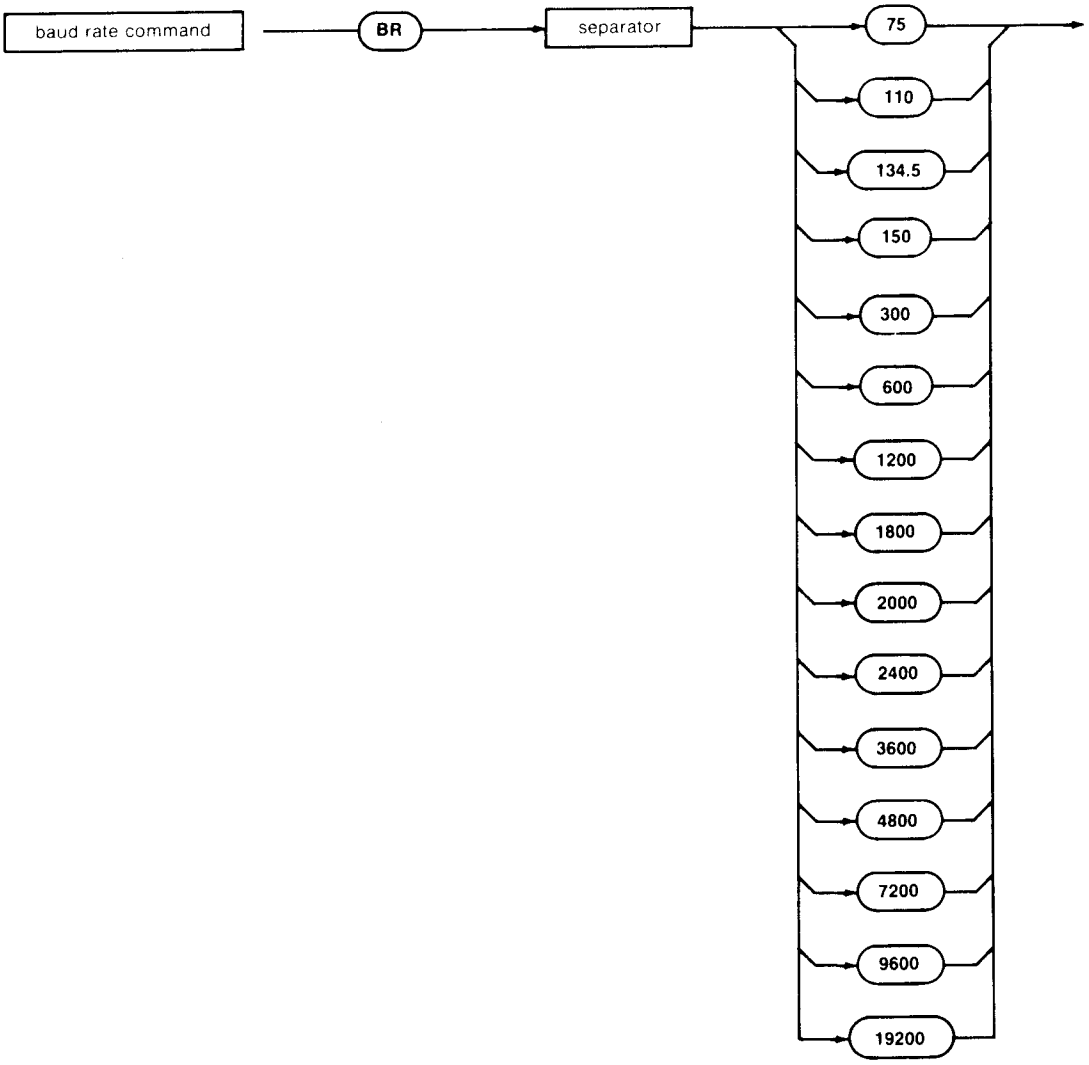
  ?No device specified

KB0: identifies the console device. All command inputs except baud rate are ignored for KB0:. Normally, the console device is the Programmer Keyboard for input, and the display for output.

NOTE

The baud rate of the 1720A Programmer Keyboard and display is 4800. Setting KB0: to any other baud rate immediately disables both the programmer keyboard and the display. This condition is remedied by pressing RESTART.

Setting Baud Rates



The baud rate command changes the baud rate of the selected port to that specified by the command argument. Baud rate is the bit transfer rate in bits per second, including start, stop, parity, and unused bits.

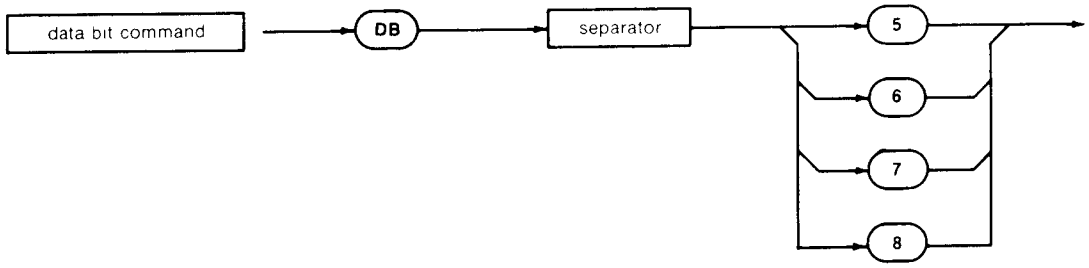
- The baud rate change takes effect when RETURN is pressed.
- If a baud rate is entered that is not one of the 15 values given in the syntax diagram, the following display appears:

  ?Bad argument

The following example sets 2400 baud for Serial Port 2, and 300 baud for Serial Port 1:

```
*KB2:  BR 2400, KB1:  BR 300 <cr>
```

Setting Character Length



The data bit command sets the number of data bits that will be included in each character.

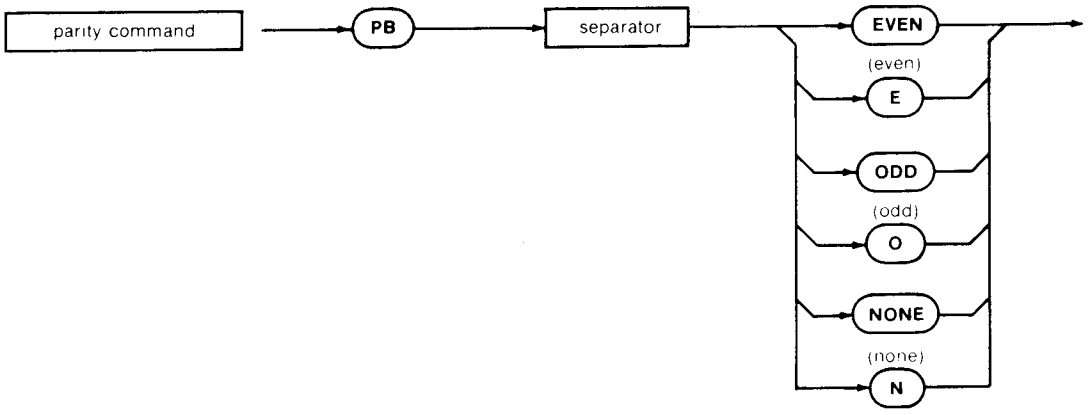
- When character length is shorter than the actual data, the lower (least significant) data bits are used.
- When character length is longer than the actual data, the remaining (most significant) bits are set to zero.
- If a word length is entered that is not one of the four values shown in the syntax diagram, the following display appears:

  ?Bad argument

The following example selects Serial Port 2, sets its baud rate to 1200, and sets the character length to 7 bits:

```
*KB2:  BR 1200, DB 7 <cr>
```

Defining a Parity Bit



The parity command defines a parity bit to be generated and checked for the selected port. Parity is an error detection scheme that uses an extra bit appended after the last data bit of each word. This bit is set so that the total number of 1-bits in each word is always even or odd.

- The command argument NONE or N eliminates parity generation and checking.
- If a command argument is entered that is not one of the six shown in

the syntax diagram, the following display appears:

?Bad argument

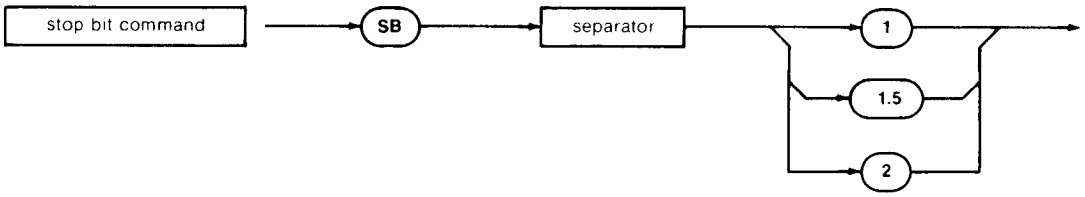
During input, parity is checked as defined by SET for each character. If an error is detected, it is identified to FDOS as a device error. In a BASIC program, this returns error number 311. This error can only be cleared by closing the channel associated with the serial port.

During output, parity is generated as defined by SET, and appended to each character. Parity is always in addition to the data bits.

The following example selects Serial Port 2, sets its baud rate to 1200, sets the word length to 7 data bits, and defines even parity:

*kb2: br 1200, db 7, pb even <cr>

Defining Character Spacing



The stop bit command defines the number of bit-cell time periods to be allocated as a minimum spacing between characters transmitted to external equipment that requires additional settling or synchronization time.

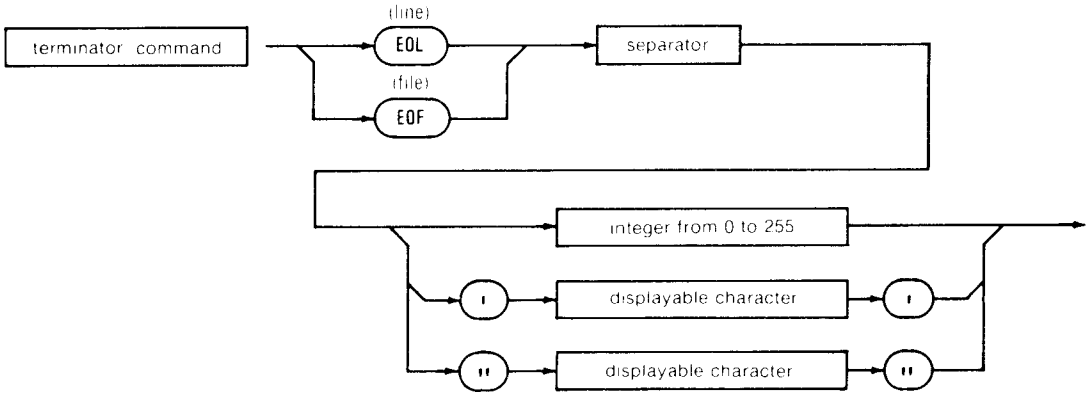
- This command does not affect incoming data.
- If an argument is entered that is not one of the three given in the syntax diagram, the following display appears:

  ?Bad argument

The following example defines a minimum transmission word spacing of 1.5 bit-cell time periods for Serial Port 1:

*KB1: SB 1.5 <cr>

Defining Terminator Characters



The terminator command defines a character that will be used to identify the end of a line or file. When received with incoming data, a terminator will generate an interrupt if enabled by the user program.

- If an integer argument greater than 255 is entered, the following display appears:

  ?Arguement out of range

- If an arguement is entered that otherwise does not meet the requirements of the syntax diagram, the following display appears:

  ?Bad arguement

- Line and file terminator characters are handled as described in the following paragraph.

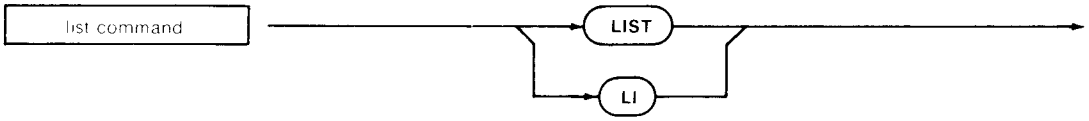
The end-of-line character defined by SET only affects data input:

- All Carriage Return and Line Feed characters are deleted.
- A Carriage Return, Line Feed sequence is then appended after each occurence of the line terminator character.
- If Line Feed or Carriage Return is the terminator, this process does not duplicate it.
- Resulting data is compatible with 1720A internal format.
- During output, data is not changed by the line terminator defined by SET.

The end-of-file character defined by SET affects both input and output:

- During input, each occurence of the file terminator defined by SET is deleted, and <CTRL/Z> (character 26) is put in its place.
- During output, each occurence of a <CTRL/Z> character is deleted, and the file terminator defined by SET is put in its place.

Listing Port Configuration



The list command displays the configuration of the currently selected port. Displayable terminator characters are shown as their decimal value and as the character.

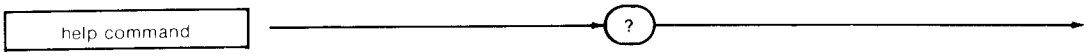
The following example lists the configuration of Serial port 2:

```
*KB2: LIST <cr>
```

The resulting display from this command might appear as follows:

```
Device          KB1:
Baud Rate       2400
Data Bits       7
Parity          even
Stop Bits       1
End of Line     10
End of File     26
```

The SET Help Display



The help command clears the screen and displays a summary of all SET commands with examples.

SET ERROR MESSAGES

SET provides error messages that aid the user in recognizing faulty command inputs. Table 1 summarizes and defines these errors.

Table 1. SET Error Messages

MESSAGE	MEANING
?Argument missing	A command was entered without the argument necessary to complete its meaning.
?Argument out of range	A command argument was entered which was beyond the range of acceptable values.
?Bad argument	A command argument was entered which was not in the list of acceptable arguments for that command.
?Illegal device	KB2: was selected when an external terminal was in use. (See Section 4A of the 1720A User Manual.)
?No device specified	A command was entered before specifying a device.
?Unknown command	A command was entered that was not recognized.

SET COMMAND SUMMARY

Table 2 summarizes SET commands and command arguments.

Table 2. SET Commands and Arguments

FUNCTION	COMMAND	ARGUMENTS
Select Port Device	KB0: KB1: KB2:	
Define Baud Rate	BR	75 600 3600 110 1200 4800 134.5 1800 7200 150 2000 9600 300 2400 19200
Define Data Bits Per Character	DB	5 6 7 8
Define Parity Bit	PB	EVEN E ODD O NONE N
Define Stop Bits For Character Spacing	SB	1 1.5 2
Define End of Line Terminator	EOL	Integer 0 through 255 ' character ' " character "
Define End of File Terminator	EOF	Integer 0 through 255 ' character ' " character "
Return to CONMON	EXIT EX <CTRL/P>	
List Port Configuration	LIST LI	
Ignore this Command Line	<CTRL/C>	
Display Command Summary	?	

List of Software Versions
FUP 1.1

1720A

Instrument Controller

File Utility User Manual

P/N 633149
December 1981

FLUKE®

Table of Contents

INTRODUCTION

The File Utility Program (FUP) is a machine-language utility program on the System Disk with the file name FUP.CIL. It gives the user control over files on floppy disks and on the optional electronic disk. The flexible structure of FUP also provides other useful capabilities. The examples provided in this section illustrate only some of the ways this multi-purpose tool can be used.

CONVENTIONS USED IN THIS MANUAL

The following paragraphs describe conventions used in this manual. Individual sections of this manual may define additional conventions.

Command Definitions

Each 1720A command is defined in a standard format that allows a maximum of 80 characters in any command line.

- The command is named on a title line.
- A syntax diagram or syntax statement follows the title line.
- The command definition follows the syntax diagram or syntax statement.

    1. The body of the command definition is a short paragraph that describes the basic function of that command.
    2. Amplifying information is separated by item and organized in a top down outline format similar to this description.
    3. A cross-reference to any associated information or documents ends the command definition.

Syntax Diagrams

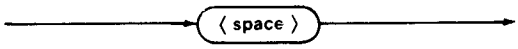
Syntax diagrams define correct spelling, punctuation, and sequence of words, symbols, and expressions for system and utility commands. The following guidelines define proper use of these diagrams:

- Any path through a diagram starting from the left that does not run contrary to an arrowhead forms a legitimate command construct. The text accompanying the diagram explains legal usage.

- Boldface words in a circular enclosure are to be entered exactly as shown. Example:

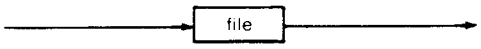
- Key entries with names, such as ESC or RETURN, are shown in a box with rounded corners. Example:

- A required space character entry is always shown as:

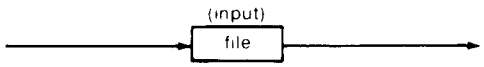


- Control character entries are shown in circular enclosures within angle brackets. The representation CTRL/ means to hold the CTRL key depressed while typing the character that follows. Example:

- Lower case words enclosed in a box represent other information to be supplied.



- Words outside the path of the diagram, usually in parentheses, provide supplementary information. These words are normally not part of the definition of the statement. Example:

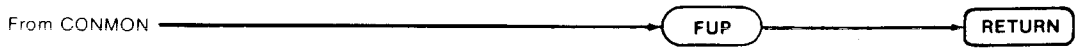


Notation Conventions

There are several notation conventions used in the text and in the examples.

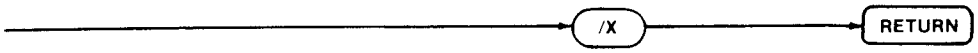
- <XX> is understood to mean press key XX.
    1. <cr> is understood to mean a carriage return (the RETURN key).
    2. <lf> is understood to mean line feed.
- [xxx] is understood to mean that xxx is an optional input.
- {xxx} is understood to mean that xxx is a required input.

ENTERING THE FILE UTILITY



FUP is accessible through CONMON by typing FUP.

EXITING THE FILE UTILTIY



FUP may be exited at any time by entering /X in response to the FUP prompt *.

USING THE FILE UTILTIY

FUP displays the message:

```
FUP(time)(date)
File Utility Progrm Version <x.y>

*
```

NOTE

Verify that the version number <x.y> is the same as on the front of this manual. If it is not, contact a Fluke Customer Service Center for advice.

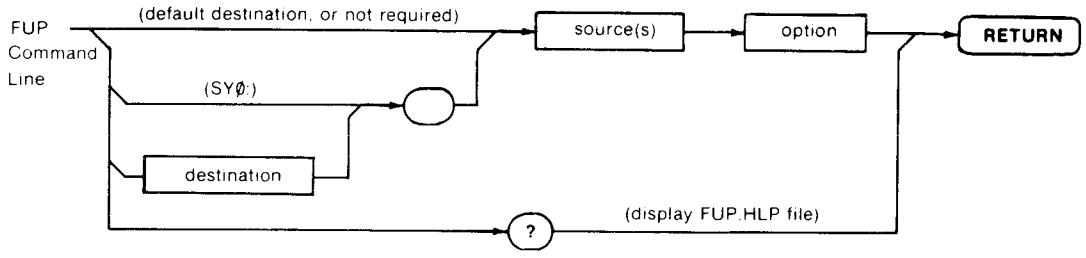
FUP Command Structure

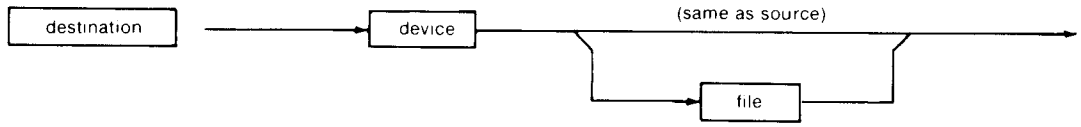
A FUP command sets up a communication channel between a specified destination device or file, and one or more source devices or files. A variety of command options are available for specific additional actions.

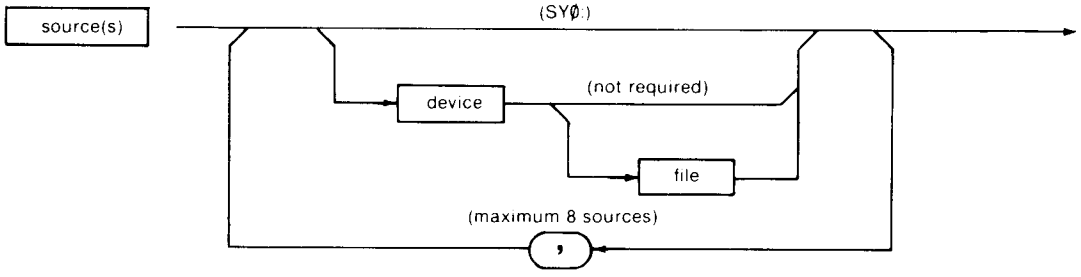
- Each command line must contain only one FUP command.
- Both upper and lower case entries are accepted.

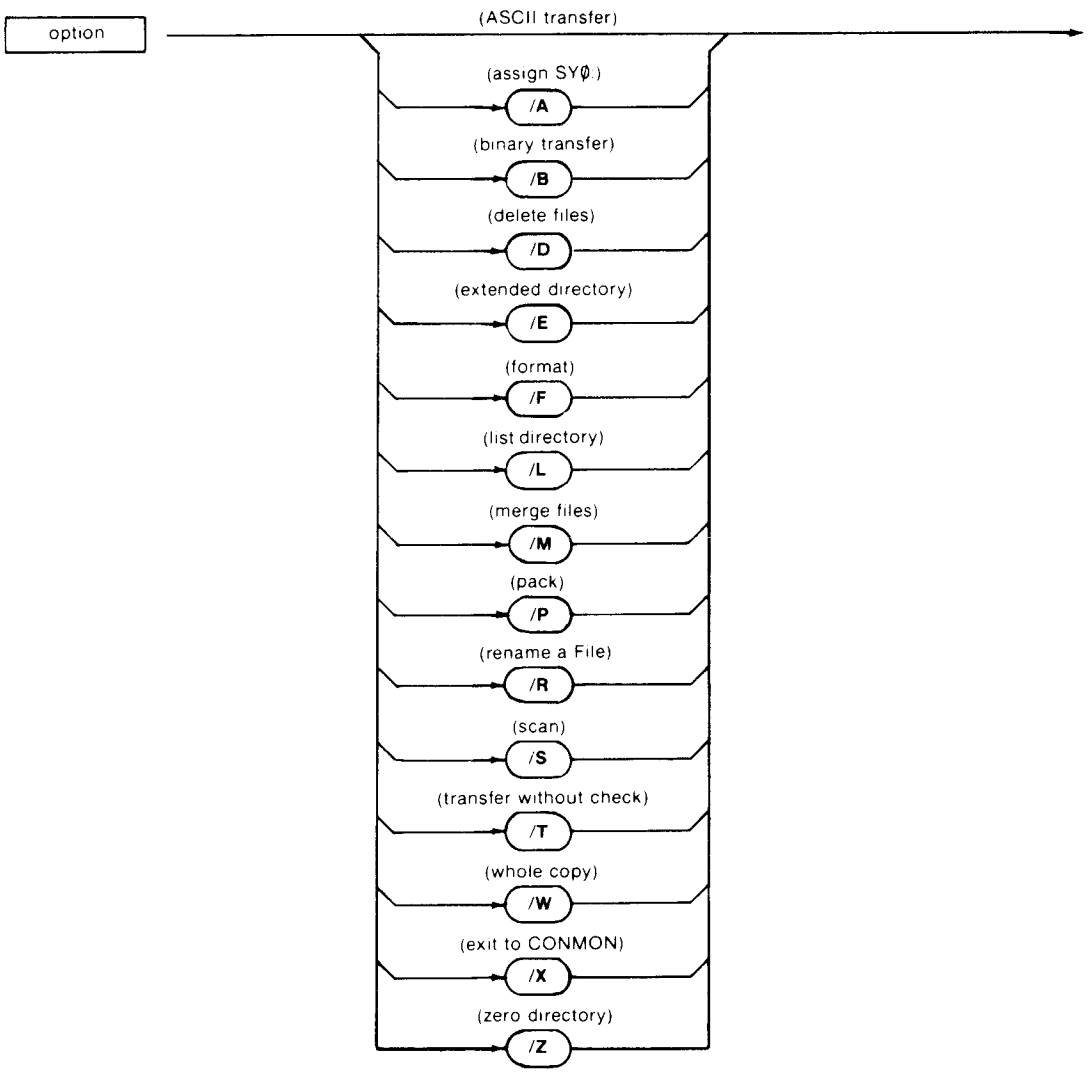
- Commands become effective when the command line is terminated by <cr>.
- The pseudo-device MM0: (main memory) can be used in FUP as temporary file storage for copying without the optional electronic disk.
- All FUP commands may be automated through command files.

The following syntax diagrams define the structure of FUP commands:

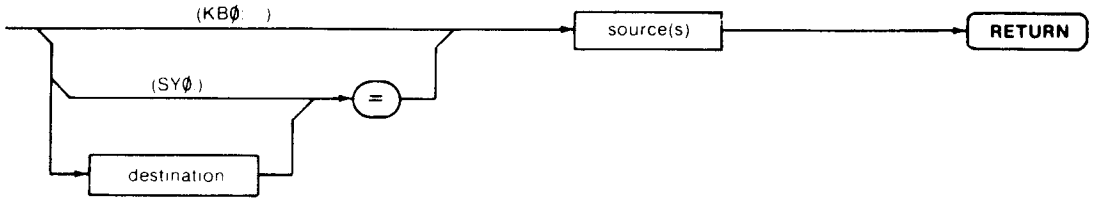








ASCII Transfer (no option)



A FUP command without a specified option sets up an ASCII communication channel between the specified destination and source(s).

- Up to eight sources may be specified.
- Transfer of a <CTRL/Z>, ASCII character number 26, identifies the end of each source. Transfer then proceeds with the next source on the list, or terminates.

- When the destination is a named file, each source overlays the previous one. The resulting destination file contains only the last specified source. To merge multiple files, refer to the discussion of the FUP /M option.

- When the destination is a file device (MF0:, ED0:, or MM0:) and no destination file name is specified, the names of the source files are used. If any source is not a file device (no name can be identified), the resulting destination file will have no name. (See the Floppy Disk Operating System User Manual for a discussion of the no-name file.)

- When the source is a file device and a source file is not specified, the source device is searched for a no-name file.

- Once a system device is specified in the source list, that device is assumed until a different device is specified.

CAUTION

If an existing file is specified as destination, it will be deleted without warning and replaced by information from the specified source.

The examples given below show the variety of ways in which this FUP command construct is useful. A short description precedes each example.

In the following example, the display (KB0:) is specified as destination, and T44.BAS on the floppy disk (MF0:) as source. This displays the file. Use Page Mode if the file is longer than 16 lines.

```
KB0:=MF0:T44.BAS<cr>
```

The following example is equivalent to the previous one if the floppy disk is the System Device. The use of defaults eliminates 13 of the 17 keystrokes.

```
T44<cr>
```

The following example copies PROGRM.38 from the floppy disk into temporary storage in main memory using the same file name.

```
MM0:=MF0:PROGRM.38<cr>
```

The following example completes the copy operation started above, after exchanging floppy disks. PROGRM 38 is copied onto the floppy disk (MF0:) from temporary storage in main memory (MM0:) using the same file name.

```
MF0:=MM0:PROGRM.38<cr>
```

The following example produces a duplicate copy of TEST.C41 on the System Device under the name TEMP.BAS.

```
TEMP=TEST.C41<cr>
```

In the following example, a file named MEMO.23 is created on the floppy disk (MF0:) from keyboard (KB0:) inputs. A <CTRL/Z> input is used to terminate the file.

```
MF0:MEMO.23=KB0:<cr>
```

The following example transfers ASCII data to the device, such as a printer, connected to RS-232 serial port 2 (KB2:) from keyboard (KB0:) inputs. A <CTRL/Z> input terminates this mode.

```
KB2:=KB0:<cr>
```

The following example sends to the device connected to Serial Port 1 a copy of TEST.M6 on the System Device.

```
KB1:=TEST.M6<cr>
```

The following example sends to the device connected to serial port 2, copies of TEST.Q4 on the System Device, and TEST.Q3 and TEST.Q2 from the floppy disk (MF0:). Note that MF0: is not specified for TEST.Q2 since it was previously specified.

```
KB2:=TEST.Q4,MF0:TEST.Q3,TEST.Q2<cr>
```

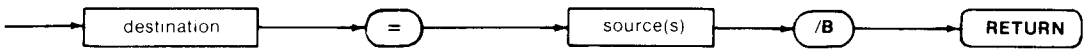
Assign the System Device (/A)

The /A FUP command option is used to assign the default file device, called the System Device. Refer to the FDOS manual for information on the System Device.

The following example assigns the optional electronic disk as System Device.

```
ED0:/A<cr>
```

Binary Transfer (/B)



The /B FUP command option is used to transfer data that contains binary code. This includes system and utility software files, lexical form BASIC programs (.BAL extension), and virtual array files.

- Up to eight source files may be specified.
- When the destination is a named file, the resulting filw will contain only the last specified source file. Binary files cannot be merged.

- When the destination is a file device (MF0:, ED0:, or MM0:) and no destination file name is specified, the names of the source files are used.

- Once a system device is specified in the source list, that device is assumed until a different device is specified.

CAUTION

If an existing file is specified as destination, it will be deleted without warning and replaced by the source file.

The following example transfers the virtual array file T4.DAT from the electronic disk (ED0:) onto the floppy disk (MF0:). A date-coded name is assigned to the file copy by this transfer.

```
MF0:220981.T4D=ED0:T4.DAT/B<cr>
```

The following example transfers some system and utility software files into temporary storage in main memory.

```
MM0:=FDOS.SYS,CONMON.SYS,SET.CIL,FUP.CIL/B<cr>
```

The following example completes the copy operation started above after exchanging floppy disks. The files in temporary storage in main memory (MM0:) are copied onto the floppy disk (MF0:) using the same file names.

```
MF0:=MM0:FDOS.SYS,CONMON.SYS,SET.CIL,FUP.CIL/B<cr>
```

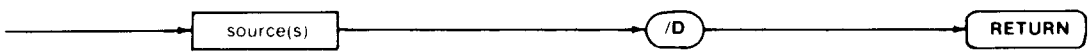
NOTE

A new disk must be formatted (/F option) before copying onto it.

NOTE

Be sure that a new System Disk created through FUP has compatible system and utility software files. Compatible version numbers are identified in the front of published Fluke manuals and manual addendums. Failure to observe this will produce unpredictable results for which Fluke cannot provide analytical, diagnostic, or other software support, except for identifi- cation of compatible combinations.

Deleting Files (/D)



The /D FUP command option is used to delete specified files.

- Up to eight files may be deleted at once.

- A single command may delete files from more than one file device.
- Deleting a file leaves a gap in the file structure. Refer to the Pack (/P) command option.
- When no file is specified, the no-name file is deleted.

The following example deletes the file TEST6.BAS from the System Device.

```
TEST6.BAS/D<cr>
```

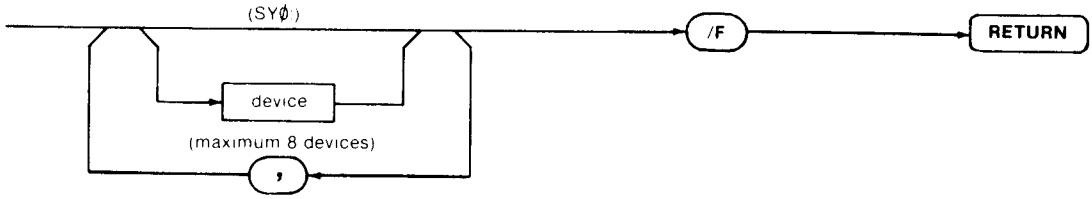
The following example deletes the BASIC Interpreter system software file from temporary storage in main memory to make room for other files.

```
MM0:BASIC.CIL/D<cr>
```

The following example deletes the user program TEST.16 and the virtual array file T16.DAT from the floppy disk (MF0:). Note that MF0: is only specified once.

```
MF0:TEST.16,T16.DAT/D<cr>
```

Format, Verify, and Zero File Device(s) (/F)



The /F FUP command option is used to prepare a floppy or electronic disk to receive files. Formatting is the process of writing identifying codes and standard data patterns throughout the disk in a defined format. Each data block is verified for data integrity during formatting. A message is displayed if a block requires one or more retries, or is unusable. A file directory is then set up in proper format with all entries set to zero.

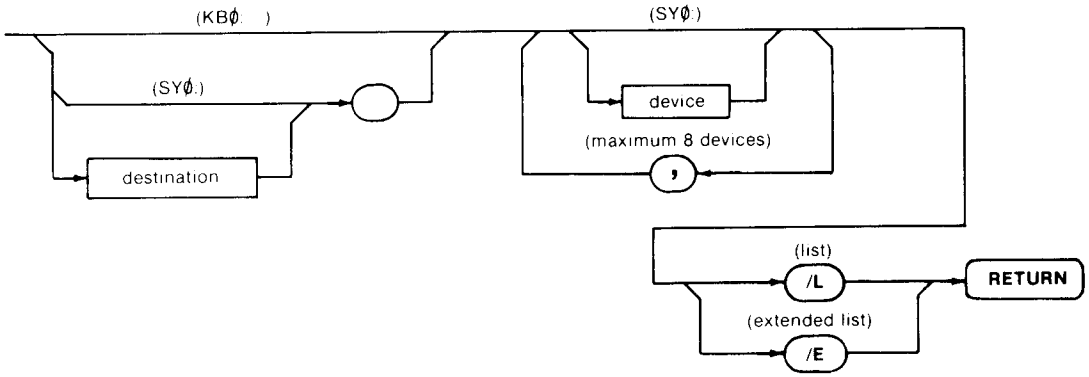
- Because formatting deletes all files from the specified device, FUP requests an affirmative before proceeding. Entries accepted as affirmative are YES, Y, yes, and y.
- The affirmative response must be on a separate line. The affirmative response is required even when /F is used in a command file.
- Formatting a floppy disk (MF0:) also causes the pseudo-device main memory (MM0:) to be zeroed, which deletes any files that were in temporary storage.
- The electronic disk must be formatted before use.
- The main memory temporary storage (MM0:) does not require formatting before use.

This example formats, verifies, and zeros a floppy disk:

```
MF0:/F<cr>
```

The message Really format MF0? will then be displayed. An entry of YES, Y, yes, or y allows formatting to proceed.

Listing a Directory (/L or /E)



The /L and /E FUP command options are used to list a directory of one or more file storage devices.

- The listing is sent to the display when no other destination is specified.
- The extended list includes all empty file areas remaining after deleting files. See the /P FUP command option for a technique of consolidating these empty areas.

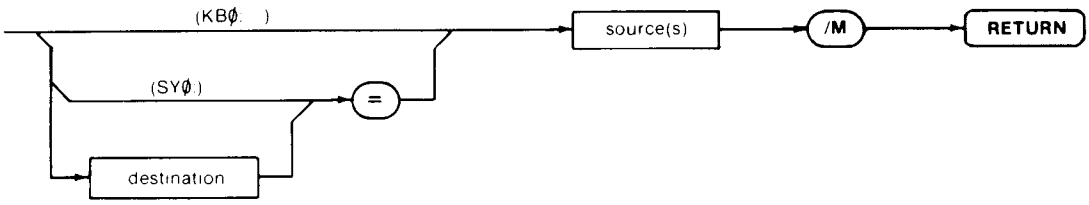
The following example displays an extended directory list of the System Device:

```
/E<cr>
```

The following example sends a directory list of the floppy disk (MF0:) and the electronic disk (ED0:) to serial port 2 (KB2:):

```
KB2:=MM0:,ED0:/L<cr>
```

Merging ASCII Files (/M)



The /M FUP command option merges up to eight ASCII source files into one destination file.

- The source files remain intact unless the destination file uses one of their names.

- Once a file device is specified in the source list, that device is assumed until a different device is specified.
- When the destination file is not given a name, a no-name file is created.
- When the destination is a non-file device (such as (KB2:), the /M option has the effect of removing the <CTRL/Z> character from the end of all but the last file.
- System and utility software files, lexical form BASIC programs (.BAL extension), and virtual array files cannot be merged.

CAUTION

Make sure that the line numbers in BASIC programs are different before merging them together. The BASIC Interpreter allows only one occurance of each line number. When a duplicate line number is encountered by BASIC during a program load into main memory, the previous line of that number is deleted.

The following example creates a file on the floppy disk (MF0:) called PROGRM.T44. The new file will be composed in sequence of the contents of TEST1.BAS and TEST2.BAS on the floppy disk, and TEST4.BAS on the electronic disk. The original files all remain intact.

```
MF0:PROGRM.T44=MF0:TEST1,TEST2,ED0:TEST4/M<cr>
```

The following example merges two files and deletes one of them by reusing the same file name. The new file TEST4.BAS is composed of the old file TEST4.BAS and the file TEST1.BAS. The file TEST1.BAS remains intact.

```
TEST4=TEST4,TEST1/M<cr>
```

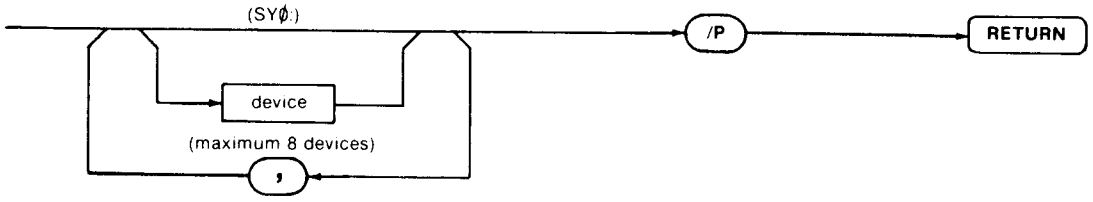
The following example shows a technique that allow the user to directly append keyboard input to the end of an existing file without creating a new file. The destination file DATA$C.832 consists of the old file DATA$C.832 followed by keyboard (KB0:) inputs. A keyboard entry of <CTRL/Z> terminates the keyboard portion of the input. The old file is deleted by the creation of the new file with the same name.

```
DATA$C.832=DATA$C.832,KB0:/M<cr>
```

The following example is similar to the previous one, except that the keyboard input is placed at the beginning of the file. An entry of <CTRL/Z> terminates the keyboard (KB0:) portion, and copies the file DATA.T26 after it. The old file is replaced by the new file with the same name.

```
DATA.T26=KB0:,DATA.T26/M<cr>
```

Packing a File Device (/P)



The /P FUP command option is used to restructure a file device so that the unused areas left by file deletions are collected into a contiguous area at the end of the file structure.

- Since files are contiguous, packing often makes room for a file that previously would not fit.
- The extended list (/E option) shows whether packing will be of benefit.
- Packing has no effect on the integrity or size of files.

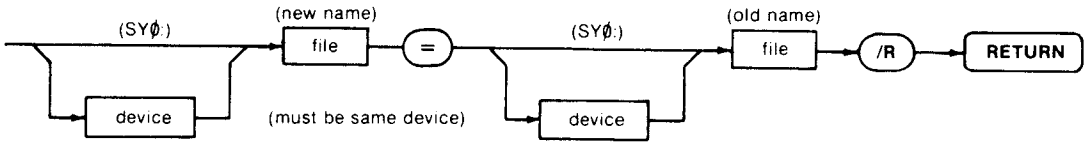
The following example packs the System Device:

```
/P<cr>
```

The following example packs the electronic disk and the floppy disk:

```
ED0:,MF0:/P<cr>
```

Renaming a File (/R)



The /R FUP command option is used to rename a file.

- Renaming has no effect on the size or location of a file. Renaming operates only on the file directory.
- An error message is displayed if the file devices specified for the new and old file names do not match.
- The no-name file is assumed when a file name is not given.
- For convenience, the utility programs SET, FUP, TIME and the BASIC language interpreter can be renamed S, F, T and B, respectively.

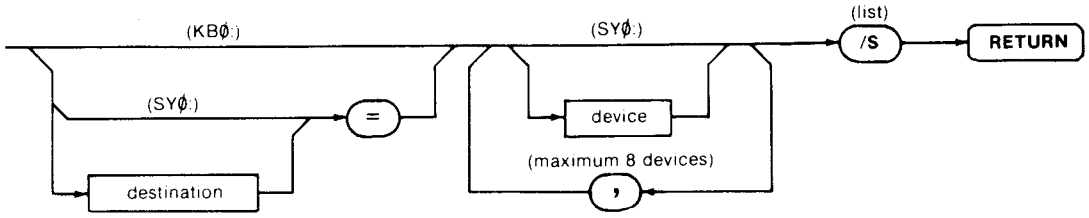
The following example renames the file TEST4.BAS on the System Device to PROG.T4:

```
PROG.T4=TEST4/R<cr>
```

The following example assigns the name FILE84.T26 to a file on the electronic disk that previously has no name:

```
ED0:FILE84.T26=ED0:/R<cr>
```

Scanning For Bad Blocks (/S)



The /S FUP command option is used to scan a file-structured device for bad blocks. Each block is read and a check character is generated. It is then compared with the check character recorded with the block. Result messages are sent to the specified destination.

- Results of the scan are displayed when no destination is specified.
- Scanning a device has no effect on the integrity or location of files.

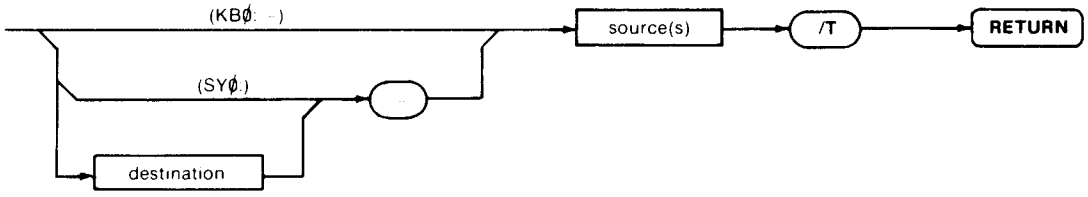
The following example performs a scan of the System Device:

```
/S<cr>
```

The following example scans the floppy disk (MF0:) and sends the results of the scan to serial port 2 (KB2:):

```
KB2:=MF0:/S<cr>
```

Transferring ASCII files Without Error Checks (/T)



The /T FUP command option is used to inhibit the checking for device error and for end-of-file code (<CTRL/Z>) during an ASCII file transfer.

- Except for inhibiting error checks, this command is identical to the no-option FUP command described in this section.
- If an end-of-file code is missing, it will be added to the end of the destination file.
- If a data block check-character fails, causing a device error, the error condition is ignored and the block is transferred unchanged.
- Source files are left unchanged.

The following example will display the file TEST6.BAS from the System Device, including any incorrect data and regardless of a file terminator:

```
TEST6/T<cr>
```

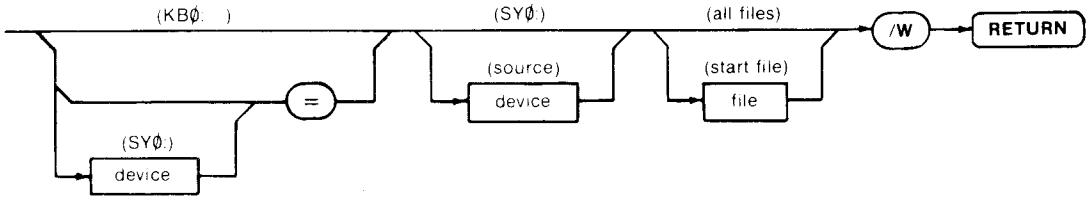
The following example creates a back-up copy on the floppy disk (MF0:) of the System Device file TEST.44 which is suspected to contain an error:

```
MF0:BACKUP.T44=TEST.44/T<cr>
```

The following example copies three files from the electronic disk, which has been assigned as System Device, onto the floppy disk (MF0:) for back-up purposes using the same file names:

```
MF0:=TEST1,TEST2,TEST3/T<cr>
```

Whole Copying a File Device (/W)



The /W FUP command option is used to transfer some or all of the files on a file device at one time using a single command. This option simplifies the task of duplicating a floppy disk, whether or not an optional electronic disk is available.

- The source and destination devices may be the floppy disk (MF0:), the main memory temporary storage (MM0:), or the optional electronic disk (ED0:). They must however be different.

- To duplicate all or part of a floppy disk, first copy files into MM0: or ED0:, insert a formatted disk, and copy from MM0: or ED0: onto the floppy disk (MF0:). More than one pass may be required, especially when using MM0:.

- An electronic disk is needed to create a back-up floppy disk whenever any single file size exceeds the size of MM0:, approximately 85 blocks (42K bytes). This is because the copy process works only with complete files. A two-module electronic disk is needed to create a back-up floppy disk whenever any single-file size exceeds 254 blocks (130,048 bytes).

- Files are copies in the sequence they appear on the source device directory. Copying starts at the beginning of the directory or at the specified start file.

- Binary and ASCII files may be copied within the same command.

- If the destination device already has files on it, the files copied from the source device are added. If this results in two files on the destination device with the same name, the existing file is deleted and replaced with the one copied from the source device.

- During the copy process, the names of the files being copied are displayed. If at any time the destination device does not have room for the next file to be copied, the process stops with all files complete and properly terminated. It is possible to see where the process stopped by looking at the display.

- The Whole Copy process can be properly terminated before all files are copied on the source device by typing <CTRL/C> during the copy of the last file desired. The file copy in progress will be completed before the operation stops.

- If <CTRL/P> is used to terminate a whole copy, the file copy in progress is aborted and the resulting partial file is closed.

The following example begins a whole copy into main memory temporary storage (MM0:) of the files on the floppy disk (MF0:). The display should be noted for the last file copied, in the event a second pass is required.

```
MM0:=MF0:/W<cr>
```

After noting the last file copied, insert a disk that has previously been formatted (see the /F option). The following example then copies the files onto the floppy disk (MF0:) that were in main memory temporary storage (MM0:).

```
MF0:=MM0:/W<cr>
```

NOTE

Formatting a disk deletes the files in main memory temporary file storage.

If a second pass is needed, the main memory temporary storage must first be zeroed (see the /Z option) to make room for additional files. Then if a check of the directory of the original disk (see the /L option) shows the next file to be copied is CTEST.H29, the following example will resume whole copying with that file:

```
MM0:=MF0:CTEST.H29/W<cr>
```

After completing a second pass of copying files into main memory (MM0:), the new disk is reinserted. The following example then copies files onto the floppy disk (MF0:) from main memory temporary storage (MM0:) adding them after the files that are already in place:

```
MF0:=MM0:/W<cr>
```

Use of the optional electronic disk (ED0:) in place of main memory temporary storage (MM0:) will normally speed up the process since its greater capacity will reduce the number of passes. Disk duplication using a two-module electronic disk is always a single-pass operation because the electronic disk capacity is then greater than the floppy disk.

Zeroing a File Directory (/Z)

The /Z FUP command option is used to zero the directory of one or more file devices.

- Because zeroing deletes all files from the specified device, FUP will request an affirmative before proceeding. Entries accepted as affirmative are YES, Y, yes, and y.

- The afffirmative response must be on a separate line. It is required even when /Z is used in a command file.

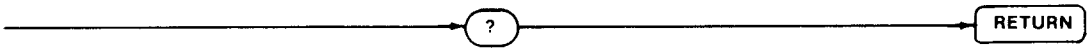
The following example zeroes the main memory temporary storage area to make room for additional files:

```
MM0:/Z<cr>
```

The following example zeros the electronic disk (ED0:) and main memory temporary storage (MM0:).

```
ED0:,MM0:/Z<cr>
```

The FUP Help File (?)



An entry of ? given at any time in respones to FUP prompt * causes the file FUP.HLP to be displayed. As supplied on the System Disk, this file is a one-screen summary of FUP commands options with examples.

- The FUP help file is a convenient quick reference of FUP command options.

- The message "?Help file not available" is displayed when the file FUP.HLP is not found.

FILE UTILITY ERROR MESSAGES

FUP provides error messages that aid the user in recognizing faulty command inputs. Table 1 summarizes and defines these errors.

FILE UTILITY COMMAND SUMMARY

Table 2 summarizes FUP command options for quick reference.

Table 1. File Utility Error Messages

MESSAGE	MEANING
?Device Error	A non-recoverable error was detected during transfer to or from the floppy disk or the electronic disk. This also occurs when writing on an unformatted new floppy or electronic disk.
?Device not ready	The floppy disk is not ready. This usually means that the disk is not inserted, or the disk drive door is not shut.
?Devices do not match	A rename was attempted for files not on the same device.
?File already exists	A rename was attempted using a file name already in use.
?File not found	The file could not be found on the device specified. This is usually caused by mispelling, although the wrong device may have been specified.
?Help file not available	A ? was entered and the file FUP.HLP could not be located on the System Device.
?Illegal option	The command option selected was not recognized. This is usually caused by typing errors.
?No end-of-file	An ASCII source file was not terminated with a CTRL/Z.
?No room for user on device	A copy or merge operation was attempted, but the resulting file would not fit on the specified device.
?Not a valid device	A Device was specified that is not on the list of recognized devices at the beginning of this section. This is usually caused by misspelling.
?Not a valid file name	A file name is the command contains too many characters other than letters, numbers, spaces, or $ signs.

Table 1. File Utility Error Messages (continued)

MESSAGE	MEANING
?System error	This error should not occur under normal use. It indicates an error in FDOS or FUP. The user should make an accurate report detailing the conditions under which the error occurred and contact a Fluke Service Center.
?Syntax error	The form of the command input does not amtch the requirements of a FUP command. This is normally caused by typing errors.
?Too many files	More than eight source files were specified.
?Write protected	A write operation was attempted on the floppy disk, but it has a write protect tab. Either the wrong disk is inserted or the protection tab has to be removed.

Table 2. FUP Command Options

OPTION	MEANING
(omitted)	ASCII File or Data Transfer
/A	Assign the System Device
/B	Binary File Copy
/D	Delete File(s)
/E	Extended Directory List
/F	Format, Verify, and Zero File Device(s)
/L	List Directory
/M	Merge ASCII Files
/P	Pack File Device(s)
/R	Rename a File
/S	Scan File Device for Bad Blocks

Table 2. FUP Command Options (continued)

OPTION	MEANING
/T	Transfer File(s) Without Error Check
/W	Whole Copy, Device to Device
/X	Exit FUP to CONMON
/Z	Zero File Directory

List of Software Versions
TIME 1.0

1720A
Instrument Controller

Time and Date Utility User Manual

P/N 633008
December 1981

FLUKE®

INTRODUCTION

The Time and Date Set Utility (TIME) is a machine-language utility program on the System Disk with the file name TIME.CIL. It requests information from the user to set the time and date of the internal time clock. When called from an active command file, it does not request input unless the clock status shows that the clock has lost power since the last setting by TIME.

CONVENTIONS USED IN THIS MANUAL

The following paragraphs describe conventions used in this manual. Individual sections of this manual may define additional conventions.

Command Definitions

Each 1720A command is defined in a standard format that allows a maximum of 80 characters in any command line.

- The command is named on a title line.
- A syntax diagram or syntax statement follows the title line.
- The command definition follows the syntax diagram or syntax statement.

    1. The body of the command definition is a short paragraph that describes the basic function of that command.
    2. Amplifying information is separated by item and organized in a top down outline format similar to this description.
    3. A cross-reference to any associated information or documents ends the command definition.

Syntax Diagrams

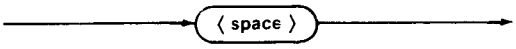
Syntax diagrams define correct spelling, punctuation, and sequence of words, symbols, and expressions for system and utility commands. The following guidelines define proper use of these diagrams:

- Any path through a diagram starting from the left that does not run contrary to an arrowhead forms a legitimate command construct. The text accompanying the diagram explains legal usage.

- Boldface words in a circular enclosure are to be entered exactly as shown. Example:

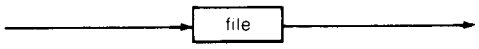
- Key entries with names, such as ESC or RETURN, are shown in a box with rounded corners. Example:

- A required space character entry is always shown as:

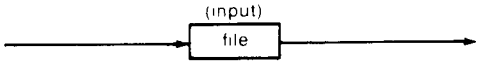


- Control character entries are shown in circular enclosures within angle brackets. The representation CTRL/ means to hold the CTRL key depressed while typing the character that follows. Example:

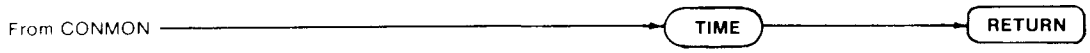
- Lower case words enclosed in a box represent other information to be supplied.



- Words outside the path of the diagram, usually in parentheses, provide supplementary information. These words are normally not part of the definition of the statement. Example:

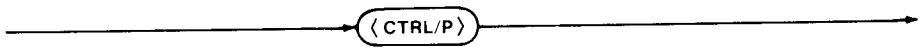


ENTERING THE TIME AND DATE SET UTILITY



TIME is accessable through CONMON by typing TIME.

EXITING THE TIME AND DATE SET UTILITY



TIME may be exited by entering <CTRL/P>. When the requested information is entered, control automatically returns to CONMON without the need for a separate command entry.

USING THE TIME AND DATE SET UTILITY

TIME displays the message:

```
TIME(time)(date)
Time Version <x.y>
Enter date:  DD-MM-YY
```

NOTE

Verify that the version number <x.y> is the same as on the front of this manual. If it is not, contact a Fluke Customer Service Center for advice.

Type in the date, numeric month, and the year (two digits). The entries may be separated by any non-numeric character. Complete the entry with RETURN. TIME responds with:

```
Enter time:  HH-MM
```

Type in the time, in 24-hour form, starting with the hour, followed by the minutes. The entries may be separated by any non-numeric character. Complete the entry with RETURN, and TIME returns control to CONMON.

The 24-hour day starts at 00:00 (midnight) and ends at 23:59 (just before midnight). If the time is at or beyond midnight and before 1:00 A.M., subtract 12 hours. If the time is at or beyond 1:00 P.M., and before midnight, add 12 hours. Otherwise, use the time unchanged. Three examples:

TIME	24-HOUR TIME
12:30 A.M.	00:30
9:45 A.M.	09:45
4:20 P.M.	16:20

The clock does not compensate for leap years. Whenever February 29 occurs, it must be manually entered. It if occurred on a day the 1720A was not being used, the date will be ahead by one until it is corrected.

The clock maintains a status flag that is set whenever power is first applied to it. The clock is supported by the internal battery even when the electronic disk is not. The TIME utility resets this flag after setting time and date information into the clock. Whenever TIME is called from an active command file, this status flag is checked. If it is found to be reset, no input is requested. This feature allows the use of a system command file that checks the clock for interruption without requiring keyboard input, except when the clock information is known to be wrong.

List of Software Versions
CONMON 1.2

1720A
Instrument Controller

Console Monitor User Manual

P/N 638932
December 1981

FLUKE®

INTRODUCTION

The Console Monitor (CONMON) is a machine-language system program on the System Disk with the file name CONMON.SYS. It processes keyboard inputs, giving the user access to utility programs and programming languages. Each utility program and language includes a command to return to CONMON.

CONVENTIONS USED IN THIS MANUAL

Introduction

The following paragraphs describe conventions used in this manual. Individual sections of this manual may define additional conventions.

Command Definitions

Each 1720A command is defined in a standard format that allows a maximum of 80 characters in any command line.

- The command is named on a title line.
- A syntax diagram or syntax statement follows the title line.
- The command definition follows the syntax diagram or syntax statement.

    1. The body of the command definition is a short paragraph that describes the basic function of that command.
    2. Amplifying information is separated by item and organized in a top down outline format similar to this description.
    3. A cross-reference to any associated information or documents ends the command definition.

Syntax Diagrams

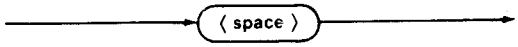
Syntax diagrams define correct spelling, punctuation, and sequence of words, symbols, and expressions for system and utility commands. The following guidelines define proper use of these diagrams:

- Any path through a diagram starting from the left that does not run contrary to an arrowhead forms a legitimate command construct. The text accompanying the diagram explains legal usage.

- Boldface words in a circular enclosure are to be entered exactly as shown. Example:

- Key entries with names, such as ESC or RETURN, are shown in a box with rounded corners. Example:

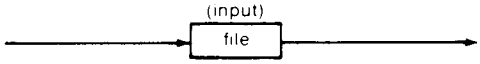
- A required space character entry is always shown as:



- Control character entries are shown in circular enclosures within angle brackets. The representation CTRL/ means to hold the CTRL key depressed while typing the character that follows. Example:

- Lower case words enclosed in a box represent other information to be supplied.

- Words outside the path of the diagram, usually in parentheses, provide supplementary information. These words are normally not part of the definition of the statement. Example:



ENTERING THE CONSOLE MONITOR

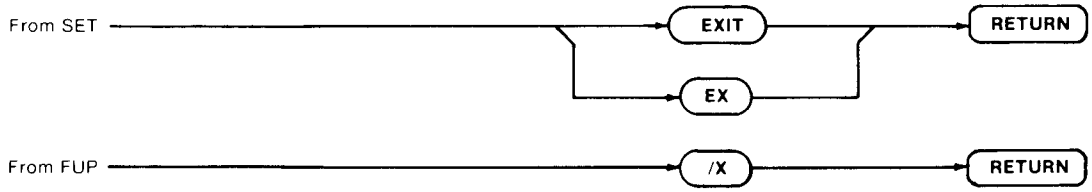
Several different paths are provided for gaining access to CONMON. The start-up sequence of the 1720A Instrument Controller terminates with the loading of CONMON when a system command file is not provided. Each utility program and language includes a command to return to CONMON. Also, a direct access method is provided that overrides all other operations. Each of these is discussed below.

Start-Up Access

Refer to Figure 1. Whenever power is turned on, or RESTART is pressed, the 1720A loads CONMON after completing a self-test, loading FDOS, and checking for a system command file. If the file COMMND.SYS is not found on the system device, the operation stops with CONMON loaded and command file status inactive.

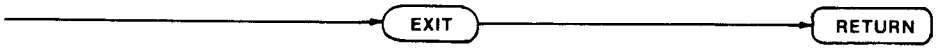
To save a system command file without allowing it to intercept the start-up sequence, use FUP to rename the file to COMMND.CMD. The file will then be ignored at start-up time. However, the file can be used any time COMMND is given as an input to CONMON.

Access From Utility Programs



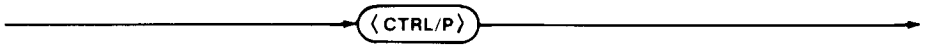
Each utility program includes a command to return to CONMON. When the command is entered, the utility program properly completes and terminates any operation that was in progress before requesting FDOS to load CONMON. This process deletes the utility program from Main Memory.

Access From BASIC



When the BASIC interpreter processes an EXIT statement, all open files are closed and terminated and a request is made to FDOS to load CONMON. This process deletes BASIC and the user program that was in memory. The EXIT statement may be part of a running program, or may be directly executed as an immediate mode command.

Direct Access



Whenever a <CTRL/P> entry from the keyboard is detected by FDOS, any other operation in progress is immediately suspended, the file CONMON.SYS is loaded from the System Device, and control is transferred to it. Other Main Memory contents (except FDOS) are no longer usable. If a file was being written to the floppy or electronic disk, it will appear on the file directory, but may

not be accessible to FDOS if the <CTRL/P> entry interrupted file completion and termination. This direct access method should not generally be used except as a back-up when other methods do not work and when the user is certain that the operation in process is not writing to a file.

USING CONMON

When accessed, CONMON displays the message:

```
Console Monitor Version <x.y>              (time)(date)
#
```

NOTE

Verify that the version number is the same as on the front page of this manual. If it is not, contact a Fluke Customer Service Center for advice.

CONMON processes inputs in the following way:

1. The directory of the system device is searched for a file with the entered name, having a .CMD extension. If found, it is loaded into memory, command file status in FDOS is set active, and the contents of the file are processed as if they were received as inputs from the keyboard.

2. If the .CMD file is not found, the directory of the system device is searched for a file of the entered name, having a .CIL extension. If found, it is loaded into memory as a machine language program and control is transferred directly to it.

3. If neither the .CMD file nor the .CIL file is found, CONMON displays:

?File not found.

List of Software Versions
FDOS 1.5

1720A
Instrument Controller

Floppy Disk Operating System User Manual

P/N 633123
December 1981

FLUKE®

Table of Contents

List of Illustrations

List of Tables

Section 1
Introduction

1-1. INTRODUCTION

1-2. The Floppy Disk Operating System (FDOS) is a machine-language system program supplied on the System Disk with the file name FDOS.SYS. It responds to interrupts from input and output devices, such as the keyboard, the touch-sensitive display, and the floppy disk. In addition, FDOS performs file management on the floppy and electronic disks for all other programs. Whenever the 1720A is operating under normal conditions, FDOS is present in Main Memory. FDOS also includes a processor that allows command files of stored user defined keyboard inputs to be processed automatically when called. Since these command files consist of a series of inputs to the console monitor, this concept is explained in the Console Monitor User Manual.

1-3. The 1720A is designed to be a single-user, single-processor computer so FDOS does not have many of the complexities inherent in operating systems used by large mainframe processors. Even so, FDOS is a sophisticated operating system (O/S), well-suited for its purpose as the 1720A Instrument Controller executive.

1-4. Section 2 of this manual provides user information and Section 3 provides programmer information. Appendix A and Appendix B support the programmer information.

1-5. CONVENTIONS USED IN THIS MANUAL

1-6. Introduction

1-7. The following paragraphs describe conventions used in this manual. Individual sections of this manual may define additional conventions.

1-8. Command Definitions

1-9. Each 1720A command is defined in a standard format that allows a maximum of 80 characters in any command line.

- The command is named on a title line.
- A syntax diagram or syntax statement follows the title line.

- The command definition follows the syntax diagram or syntax statement.

    1. The body of the command definition is a short paragraph that describes the basic function of that command.

    2. Amplifying information is separated by item and organized in a top down outline format similar to this description.

    3. A cross-reference to any associated information or documents ends the command definition.

1-10. Syntax Diagrams

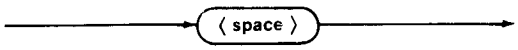
1-11. Syntax diagrams define correct spelling, punctuation, and sequence of words, symbols, and expressions for system and utility commands. The following guidelines define proper use of these diagrams:

- Any path through a diagram starting from the left that does not run contrary to an arrowhead forms a legitimate command construct. The text accompanying the diagram explains legal usage.

- Boldface words in a circular enclosure are to be entered exactly as shown. Example:

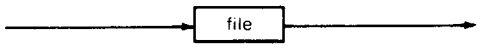
- Key entries with names, such as ESC or RETURN, are shown in a box with rounded corners. Example:

- A required space character entry is always shown as:

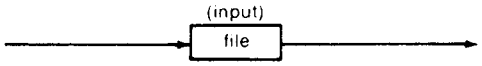


- Control character entries are shown in circular enclosures within angle brackets. The representation CTRL/ means to hold the CTRL key depressed while typing the character that follows. Example:

0 Lower case words enclosed in a box represent other information to be supplied. Example:



0 Words outside the path of the diagram, usually in parentheses, provide supplementary information. These words are normally not part of the definition of the statement. Example:



1-12. Notation Conventions

1-13. There are several notation conventions used in the text and in the examples.

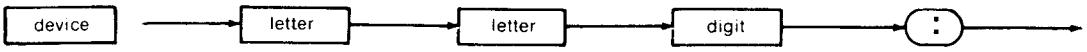
- 0 <XX> is understood to mean press key XX.
  1. <cr> is understood to mean a carriage return (the RETURN key).
  2. <lf> is understood to mean line feed.
- 0 [xxx] is understood to mean that xxx is an optional input.
- 0 {xxx} is understood to mean that xxx is a required input.
- 0 ^ is understood to mean that the characters that follow are a superscript. For example, 6^2 means six raised to the second power or 36.

Section 2
User Information

2-1. INTRODUCTION

2-2. The information in this section provides the user with the information necessary to use FDOS.

2-3. DEVICE NAMES



2-4. FDOS uses symbolic names to identify the source and destination of any information transfer, except for IEEE-488 bus instruments. Table FD-1 summarizes these device names.

- Two letters followed by a numeric digit and a colon are required.
- When a file device is specified (MF0: or ED0:), the device name is used as a prefix to a file name.

  Example: MF0:TEST.Q44.

- When no device is specified, the default for file operations is the system device (SY0:). Other defaults are defined individually.
- SY0: is always redundant and does not need to be specified.
- Usage of device names is identical for any utility program or programming language.

2-5. THE SYSTEM DEVICE

2-6. The System Device is a useful default provided in FDOS for file operations.

- When a file name is specified without a device, the System Device is assumed.
- Normally the System Device is the floppy disk (MF0:).
- The System Device can be reassigned to the electronic disk by a FUP command. This process can be automated at start-up in a command file.

- Table 2-1 lists the available devices and the entry that must be made to specify the System Device.

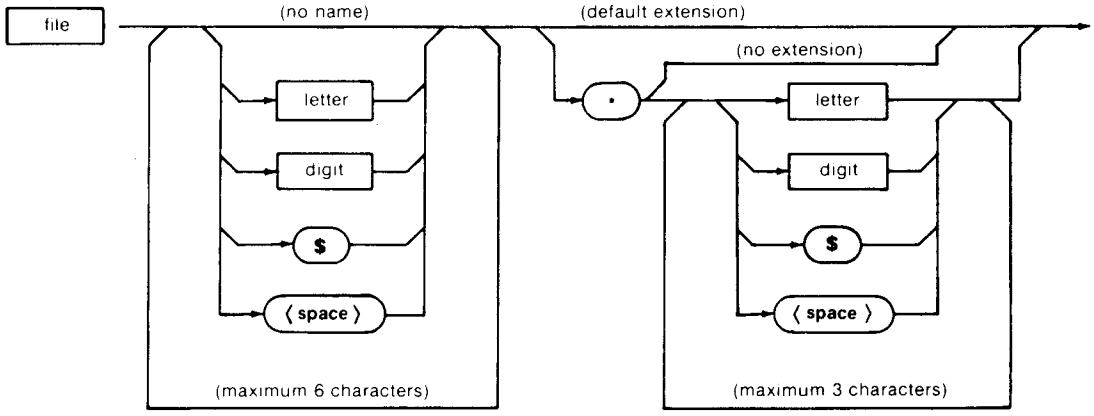
2-7. The System Device is automatically defined as the optional electronic disk at start-up time if the following conditions are met:

1. The file FDOS.SYS is not found on the floppy disk.
2. The electronic disk is installed.
3. The electronic disk contains a valid directory and the file FDOS.SYS.

Table 2-1. Device Names

ENTRY	DEVICE	ENTRY	DEVICE
SY0:	System Device	KB0:	Keyboard and Display
MF0:	Floppy Disk	KB1:	RS-232 Serial Port 1
ED0:	Optional Electronic Disk	KB2:	RS-232 Serial Port 2
MM0:	Main Memory (FUP only)		

2-8. FILE NAMES



2-9. FDOS uses file names to identify programs and data files stored on a file device (floppy or electronic disk).

- A file name is 1 to 6 letters, numeric digits, or $ or space characters, in any combination. Examples:

  TESTB2, $25795

- The optional file name extension is a period followed by 1 to 3 letters, numeric digits, or $ or space characters in any combination. Examples:

  TEST.T43, FILE9.$3

- Since a file is uniquely identified by its name, there can be only one occurrence of any file name on each file device.

- When FDOS is unable to identify a file name from the context of an operation, it searches for or assigns, a file with no name. There can be only one such no-name (all-space) file on each file device.

- A period at the end of a file name without an extension specifies a blank (3-space) extension.

- A file name without an extension specifies a default extension that is defined by the utility program or language in use.

2-10. The following file names are used by FDOS to identify special purpose files:

NAME	MEANING
FDOS.SYS	Floppy Disk Operating System
CONMON.SYS	Console Monitor
TIME.CIL	Time and Date Set Utility
SET.CIL	Set Serial Ports Utility
FUP.CIL	File Utility Program
BASIC.CIL	Fluke Enhanced BASIC Interpreter
COMMND.SYS	System Start-up Command File
FUP.HLP	File Utility Program Help Display File

2-11. The following file name extensions are used by FDOS to identify certain types of files:

EXTENSION	MEANING
.SYS	System-level programs and command file
.CIL	Utility programs (Core Image Load)
.BAS	ASCII text BASIC program
.BAL	Lexically analyzed BASIC programs
.HLP	System level data files
.CMD	Utility command files

2-12. COMMAND FILES

2-13. Introduction

2-14. A command file is a sequence of Programmer Keyboard inputs stored in a file. Command files are identified by the form of the file name. When processed, the contents of a command file are treated as if they were input from the Programmer Keyboard, except for the ? and & characters which identify special operations.

2-15. Two types of command files are available:

1. COMMND.SYS is the system command file. It operates only at start-up time.

2. (File name).CMD is a utility command file. It operates whenever its file name is given as input to the Console Monitor.

2-16. The following characteristics are common to all command files:

- All 1720A keyboard operations are accessible from a command file.
- Command file processing begins from the Console Monitor.
- Each line contains one keyboard input command.
- Command file lines are processed in sequence only.
- The & character identifies a line to be sent directly to the display. It may include display control sequences to format a message.
- If a display line includes either the & or ? characters, each must be preceded by a backslash (\).
- Only one command file can be active at any time.
- Command files can only be one block long.
- When a command file is active, a <CTRL/C> entry from the keyboard is transmitted to the running program. Once the running program has processed the <CTRL/C> entry, the command file is aborted and "Command file aborted" appears on the display.

2-17. A command file remains active until one of the following occurs:

1. Its last line has been processed.

2. Control has been passed to another utility command file.

3. An error has been encountered.

2-18. Whenever a command file is active, it is the only source of keyboard input. This means that an INPUT statement in a user program, called by a command file that has additional lines, will take its input from the next line of the active command file. If the program terminates through an END, STOP, or EXIT statement, the next command file line is taken as keyboard input.

2-19. A command file may transfer control to the first line of any utility command file, including itself, by commanding a return to the Console Monitor, followed by a line containing the name of the utility command file. The .CMD extension is not used. Subsequent command lines in the original command file will not be processed.

2-20. The System Command File

2-21. The system command file is activated only at system start-up time. After power-on, or pressing RESTART, the 1720A runs an internal self-test, loads FDOS, and searches the start-up system device for a file named COMMND.SYS. If it is found, the Console Monitor is then loaded with command file status active, and FDOS processes the command file as keyboard inputs to CONMON and any other mode selected as a result of the inputs. The system command file takes control only at start-up time. Control cannot be subsequently transferred to it. Figure 2-1 illustrates this sequence.

2-22. Utility Command Files

2-23. A utility command file is a sequence of keyboard inputs stored in a file with a .CMD name extension. The effect is that the user can create simplified commands that operate directly from the Console Monitor.

2-24. Control is transferred to a utility command file whenever the Console Monitor processes a file name which is found to have the .CMD extension. In addition, a command argument can be passed to a utility command file at the time it is called. It is generally good practice to include commands that return the system to the Console Monitor after completing the task.

2-25. Utility command files have all of the characteristics listed previously. The following points are true only for utility command files:

- The file name extension .CMD identifies a utility command file.
- Control is transfered to a utility command file from the Console Monitor by entering the name of the command file. The .CMD extension is not used.
- A utility command file may be called directly or from any active command file, including itself.
- An argument can be passed to a utility command file by including additional characters after the file name.
    1. These characters must be separated from the file name by a space and must represent a valid keyboard input for the operation that will use them.
    2. Wherever a ? character appears in the command file, the argument is substituted. The ? character is ignored when an argument is not included.
- The following paragraphs show command file examples.

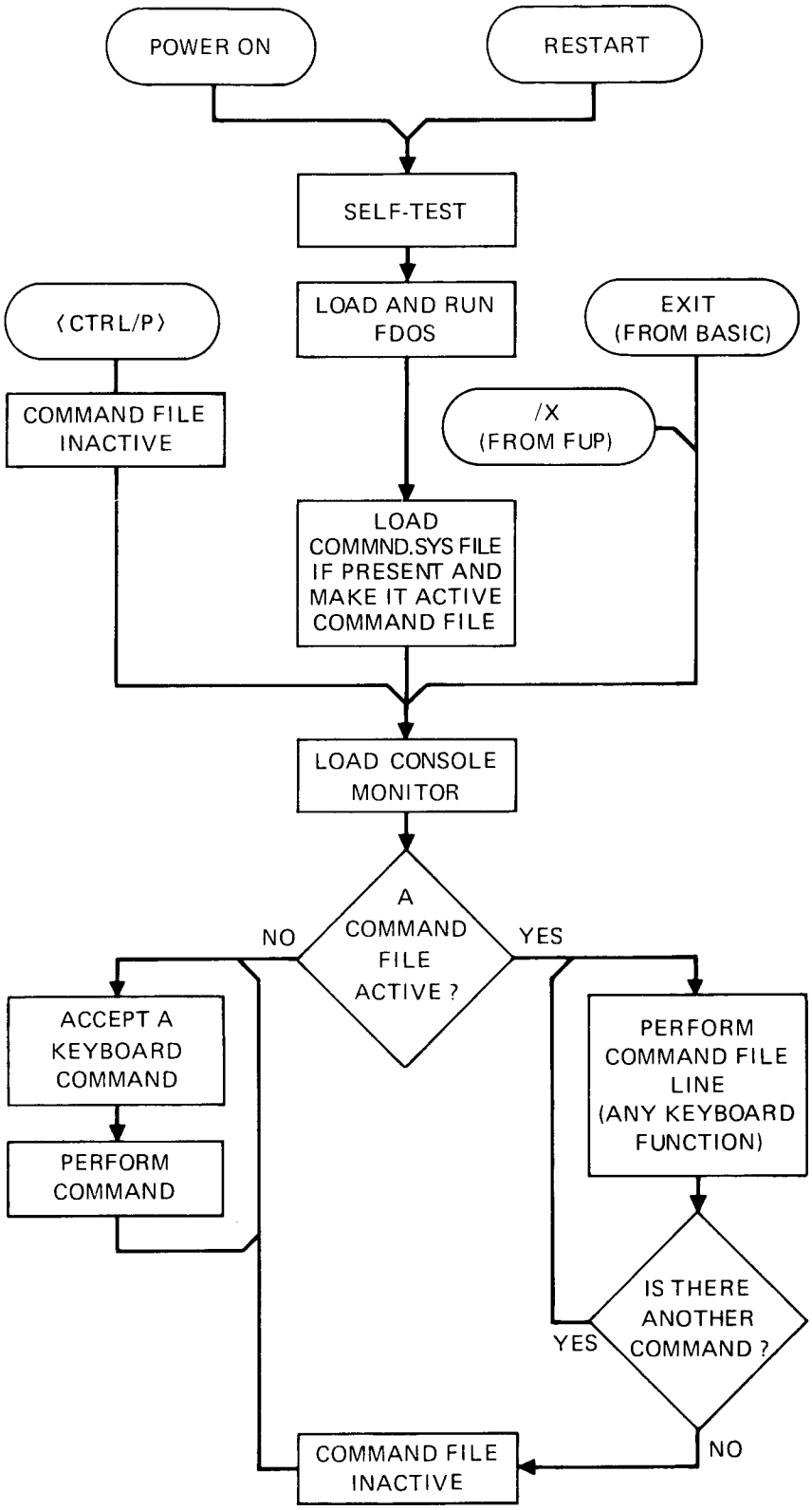


Figure 2-1. Start Sequence

2-26. Command File Examples

2-27. The following command file performs four tasks:

1. Sets some selected parameters on Serial Port 2 using the SET utility.

2. Verifies that the time clock has not experienced a power interrupt (or requests new time information) using the TIME utility.

3. Loads the BASIC interpreter.

4. Loads the program TEST43, remaining in immediate mode.

```
SET
KB2: BR 1200 DB 7 PB EVEN EXIT
TIME
BASIC
OLD "TEST43"
```

2-28. This would be a useful COMMND.SYS system command file during the development of program TEST43. Note that the words SET, EVEN, EXIT, TIME, and BASIC could have been abbreviated S, E, EX, T, and B respectively.

2-29. If the 1720A used for program development contains an optional electronic disk, the COMMND.SYS file can be expanded to speed up operation during programming. In addition to the previous tasks, this command file copies some essential system software and the program TEST43 onto the electronic disk and assigns the disk as the system device. The programmer must then remember to save the revised TEST43, after further development, onto the floppy disk.

```
SET
KB2: BR 1200 DB 7 PB EVEN EXIt
TIME
FUP
ED0:/F
YES
ED0:=FDOS.SYS,CONMON.SYS,FFUP.CIL,BASIC.CIL/B
ED0:=TEST43
ED0:/A
/X
BASIC
OLD "TEST43"
```

2-30. After the program has been developed, the working disk that uses it might contain the COMMAND.SYS system command file (described in the following paragraph). This disk automatically begins running the program TEST43 whenever the 1720A is turned on or RESTART is pressed. A Programmer Keyboard is not required unless the time clock needs to be reset.

```
TIME
BASIC
RUN "TEST43"
```

2-31. The following command file loads the BASIC interpreter, runs program TEST1, and then runs program TEST2. If program TEST1 terminates through an EXIT statement, the command file needs an additional line to reload the BASIC interpreter before attempting to run TEST2.

```
BASIC
&<ESC>[2J<ESC>[1p<ESC>[4;6HStandby for Test 2.
RUN "TEST2"
```

2-32. The message lines in the previous example use display control codes to clear the screen, select double-size characters, position the display cursor, and display a message. This is useful for a large ASCII-format program to inform the operator that Test 1 or Test 2 is being loaded. Display codes are used directly, including the escape key, without the need for quotes or the CHR$ function as in a BASIC PRINT statement. <ESC> represents an ESCape key entry.

2-33. The following command file uses FUP commands to display a directory of the system device whenever it is called and then return to the Console Monitor. If it is given the file name DIR.CMD, the system device directory can then be directly displayed from the Console Monitor by typing DIR.

```
FUP
/L
/X
```

2-34. By adding a ? character to the DIR.CMD file, it gains the ability to accept an argument. In the following example, typing DIR ED0: will produce a directory listing of the electronic disk. Typing DIR will produce a directory of the system device, as in the previous example.

```
FUP
?/L
/X
```

2-35. **Creating Command Files**

2-36. Two ways are presented to create a command file. Using the File Utility Program (FUP), the file can be directly typed and stored. For longer command files, the editing capabilities of BASIC are often useful. A BASIC program is presented that will generate and store a command file and a revised version of itself. Procedures for each of these methods follow:

2-37. For a short command file, use FUP to directly type the following in the file:

1. After a CONMON prompt #, type F to select the File Utility Program.

2. Type MF0: (File name)=KB0:. For the file name, use COMMND.SYS for a system command file, or (command name).CMD for a utility command file.

3. Type each command input to be stored on a separate line. Use <DELETE> to correct errors.

4. Use <ESC> directly to initiate display control sequences.

5. Use <CTRL> with the letters given in Table 2-2 to include some of the characters and control codes below character number 27 in a display string.

NOTE

If the command file requires codes below character number 27 that are not included in Table 2-2, a BASIC program must be used to generate the file.

6. Use two RETURN entries after the last line, when the last line is a display string introduced by &.

7. Terminate the file with <CTRL/Z>.

8. Type /X to return to the Console Monitor.

9. To use the command file, press RESTART on the front panel for a system command file, or type the assigned command name for a utility command file. (Do not include the .CMD extension.)

Table 2-2. Control Codes Available for FUP Generated Command Files

CONTROL CODE	CHARACTER OR FUNCTION	CONTROL CODE	CHARACTER OR FUNCTION	CONTROL CODE	CHARACTER OR FUNCTION
A	β	H	backspace	O	π
B	γ	I	horizontal tab	R	τ
D	ϵ	K	line feed	V	ψ
E	ζ	L	line feed	W	ω
F	η	N	ϕ	Y	Ω
G	beep				

2-38. For a longer command file, use the BASIC program shown in Figure 2-2. Refer to the 1720A Fluke BASIC Programming Manual for information on creating and storing BASIC programs. This program creates two files: the command file and a copy of the program with the name (command name).BAL. Note the following points:

- Use one DATA statement for each command file line.
- In place of (file name) in line 10010 use COMMND.SYS for a system command file, or (command name).CMD for a utility command file.
- In place of (file name) in lines 10120 and 10130, use COMMND.BAL for a system command file, or (command name).BAL for a utility command file.
- If the last command file line is a display string introduced by &, add one more DATA statement with a null string (DATA"").

```
1000     ON ERROR GOTO 10130
1010     DATA   "   "
1020     DATA   "   "
1030     DATA   "   "
10000    DATA "END"
10010    OPEN "(file name)" AS NEW FILE 1
10020    READ A$ \ IF A$= "END" GOTO 10110
10030    C$=""
10040    FOR I%=1% TO LEN (A$)
10050    B$=MID (A$,I%,1%)
10060    IF B$= "^" THEN I%=I%+1%\B$=CHR$(ASCII(MID (A$,I%,1%))-64%
10070    IF B$= "$" THEN B$=CHR$(27)
10080    C$=C$+B$
10090    NEXT I%
10100    PRINT #1, C$\GOTO 10020
10110    CLOSE 1
10120    KILL "(file name)"\SAVEL "(file name)"\GOTO 10140
10130    IF ERL=10120 AND ERR=305 THEN SAVEL "(file name)"ELSE OFF ERROR
10140    END
```

Figure 2-2. Command File Generator Program

- Because DATA statements cannot contain control codes such as escape (ESC), use a ^ character to indicate CTRL, and a $ character to indicate escape.

2-39. CONTROL CODES

2-40. Control character codes are formed by holding the CTRL key depressed while typing another letter, just as when using the SHIFT key. Some of these codes are intercepted by FDOS when detected and used to initiate specific actions. Table 2-3 summarizes these codes and the resulting actions.

Table 2-3. Control Code Summary

CODE	RESULT
<CTRL/C>	Stop program
<CTRL/P>	Abort program and return to CONMON
<CTRL/Q>	Continue display after stop
<CTRL/S>	Stop display
<CTRL/T>	Clear screen, reset display modes, cursor to upper left
<CTRL/U>	Delete current line
<CTRL/Z>	End-of-file code for ASCII files

Section 3
Programmer Information

3-1. INTRODUCTION

3-2. This section provides the information necessary to modify FDOS in order to perform certain specific tasks or operations (for example, modifying the IEEE-488 compatible bus handlers to accomodate instruments that are not IEEE-488 compatible). So, this section is a reference for systems programmers not a users guide. Most 1720A operations and modifications to 1720A programs can be performed in BASIC or assembly language without accessing FDOS.

CAUTION

FDOS routines do not perform error checks on their input parameters. The use of improper function codes or any illegal code may cause the system to terminate unexpectedly, or crash.

CAUTION

It is recommended that the interrupt mask not be altered under any circumstances. To do so may render the FDOS inoperable.

3-3. FDOS CALLS

3-4. Introduction

3-5. The FDOS calls device drivers directly or calls device drivers via the I/O manager (Kernel) using extended operations (XOPs). With the exception of the IEEE-488 device driver (which cannot be called via the I/O manager), device drivers are usually called via the I/O manager. The I/O manager can either be called using XOP15 or using supervisory calls (SVCs). FDOS calls have special requirements for passing arguments into memory and errors may occur when calls are executed. The following paragraphs describe XOP calls, SVC calls, passing arguments into memory and FDOS I/O manager errors.

3-6. XOP Calls

3-7. The XOP calls are software interrupts discussed in the TI Guide section of the 1720A Assembly Language Programming Manual. The TMS 9900 supports sixteen XOP calls.

0 XOP 00 through XOP 12 directly call device drivers. (A device driver is also referred to as a device handler.)

- XOP 13 and XOP 14 are available for use in any user program.

  1. Before using XOP 13, the user program must initialize the XOP 13 vector as follows:

     a. The workspace address of the XOP 13 handler must be placed at location >0074.

     b. The entry address of the XOP 13 handler must be placed at location >0076.

  2. Before using XOP 14, the user program must initialize the XOP 14 vector as follows:

     a. The workspace address of the XOP 14 handler must be placed at location >0078.

     b. The entry address of the XOP 14 handler must be placed at location >007A.

- XOP 15 calls the I/O manager.

  1. The I/O manager performs the primary operations of the FDOS.

  2. Each operation is invoked by XOP 15, but is distinguished by a different register number within the XOP call. The register numbers, operation names, and functions are listed in Table 3-1.

  3. The I/O manager can also be called using supervisory calls (SVCs).

Table 3-1. I/O Manager Functions

REGISTER NUMBER	OPERATION NAME	FUNCTION
R0	OPEN	Open a channel and associate it with a file or device
R1	CLOSE	Close a channel and make a file permanent
R2	READ	Read a block of data from a device
R3	WRITE	Write a block of data to a device
R4	DELETE	Delete a file from a file structured device
R5	RENAME	Rename a file on a file structured device
R6	LOAD	Load a memory image file (CIL) onto memory and start it

Table 3-1. I/O Manager Functions (continued)

REGISTER NUMBER	OPERATION NAME	FUNCTION
R7	EXIT	Give control back to the Operating System
R8	TIME	Return current date and time
R9	STATUS	Return information on whether or not data is available
R10	CONFIG	Tell if device configuration values will be read out of AREA or written into AREA, with changes

- The general format for an XOP call is

```
XOP                 reg,nmbr
```

  1. reg is the register number (also called the argument) of the requested function.
  2. nmbr is the XOP number.

3-8. The following example shows how to call the I/O manager for a Data Write.

```
XOP         R3,R15          ;WRITE A BLOCK
```

3-9. **Supervisory Calls**

3-10. Like XOP 15, supervisory calls (SVCs) call device handlers via the I/O manager. Unlike XOP 15, however, the SVC states the operation name which reduces the possibility of making an input error (this is a critical factor in preventing FDOS call failures). This also helps to write clear, easily understood code (especially important when a large number of calls is required).

- The DXOP pseudo-operation and the EQUATE statement are used to set up SVC statements.

  1. The following subroutine sets up SVC calls for the I/O manager to Open, Close, Read, and Write.

```
            DXOP        SVC,15          ;ASSIGN SUPERVISORY CALL
OPEN        EQU         0
CLOSE       EQU         1
READ        EQU         3
WRITE       EQU         4
```

2. A call for the I/O Manager to write a block of data can now be written directly as a Supervisory Call, containing the word WRITE, instead of as an indirect XOP call. The SVC Call is much easier to understand than a corresponding XOP call would be. This is demonstrated in the two lines of code below.

```
SVC     WRITE       ;WRITE A BLOCK USING AN SVC CALL

XOP     R3,R15      ;WRITE A BLOCK USING AN XOP CALL
```

0 Certain supervisory calls require file and device names to be passed in an argument area. The names must be supplied in RADIX-50, which is defined as a specific numerical format for storing alphanumeric characters in computer words. The procedure for placing characters into RADIX-50 format is detailed below.

1. Three alphanumeric characters can be stored in one 16-bit word.

2. RADIX-50 allows up to decimal 40 (octal 50) characters.

3. All characters are assigned a value from 0 to 39 using one of the two following methods:

   a. Select the correct value using the following rules:

CHARACTER	VALUE
space	0
A-Z	1-26
$	27
0-9	30-39

   b. Compute the correct value as follows:

      (1) Subtract 64 from the ASCII value of each alpha character.

      (2) Subtract 18 from the ASCII value of each numerical character.

4. The first character is given a numerical weight of 40*40, the second character is given a weight of 40, and the third character is given a weight of 1.

5. The weighted values of the characters are then added together to obtain the RADIX-50 equivalent

6. The assembler has a strict left-to-right priority (NOT a hierarchical one) in performing numerical operations.

   a. The expression 8+6*3 equals 42

   b. The expression 8+3*6 equals 66

   c. The expression 3*6+8 equals 26

7. For example, the RADIX-50 equivalent of MFO is computed with the expression:

```
MFO     EQU     'M'-64*40+'F'-64*40+'0'-18  =
                77-64*40+70-64*40+48-18 = 21,070
```

NOTE

The user program must translate file and device names into RADIX-50 format. FDOS does not perform this operation.

3-11. Passing FDOS Arguments

3-12. Arguments are passed in a section of memory whose first location is called AREA.

- The contents of memory location >00A4, which is loaded during BOOTUP, act as the pointer for AREA.
- AREA is physically located in scratch-pad memory on the CPU card.
- Any assembly language routine which calls an I/O Manager (kernel) operation in FDOS first must place the arguments for that operation into AREA.
  1. The first parameter is placed in the memory location called AREA, which is pointed to by the AREA pointer.
  2. The second parameter is placed in memory location AREA+2. The second parameter's location is offset by 2 from that of the first parameter (i.e. The pointer AREA has 2 added to it to point to the next location).
  3. The third parameter is placed in memory location AREA+4.
  4. The nth parameter is placed in memory location AREA+2n.
  5. Figure 3-1 demonstrates the offset technique of loading memory. Table 3-2 lists each SVC, the arguments which must be passed to it, and the offset for each parameter.
- Each supervisory call (SVC) requires different arguments. However, the format of some of the parameters is the same for each SVC which passes that argument. The parameters with identical formats include:
  1. Channel number is an integer between 0 and 7.
  2. Length is an integer specifying the number of blocks required or allocated.

3. File reference is always passed in RADIX-50 format and consists of four parameters:

a. Device name

b. Filename 1, which contains the first three characters of a file name

c. Filename 2, which contains the last three characters of a file name

d. Extension

Table 3-2. SVC Argument Passing

SVC NAME	OFFSET	PARAMETER	DESCRIPTION
Open	(0)	Device name	
	(2)	Filename 1	
	(4)	Filename 2	
	(6)	Extension	
	(8)	Mode	0 -- old file nonzero--new file
	(10)	Channel Number	
	(12)	Required Space	if -1 -- the largest contoguous space in units of blocks if -2 -- half the largest contiguous space in units of blocks
	on return:		
	(12)	Length of file	length in blocks
	(14)	Device type	0--file structured 1--non-file structured
Close	(0)	Channel Number	
Read	(0)	Channel Number	
	(2)	Block Number	1 through max block number of file. if <0 -- next block (relative block numbers in a file are numbered from 1 to max.)
	(4)	Buffer Pointer	anywhere in memory
	(6)	Block Length	length in bytes
	on return:		
	(6)	Block Length	length of block read in bytes (Can be different from length above)

Table 3-2. SVC Argument Passing (continued)

SVC NAME	OFFSET	PARAMETER	DESCRIPTION
Write	(0)	Channel Number	
	(2)	Block Number	1 through max block number of file. if <0 -- next block (relative block numbers in a file are numbered from 1 to max.)
	(4)	Buffer Pointer	can point anywhere in memory
	(6)	Block Length	length in bytes
Delete	(0)	Device name	
	(2)	Filename 1	
	(4)	Filename 2	
	(6)	Extension	
	(8)	Channel Number	
Rename	(0)	Old device name	
	(2)	Old filename 1	
	(4)	Old filename 2	
	(6)	Old extension	
	(8)	New device name	
	(10)	New filename 1	
	(12)	New filename 2	
	(14)	New extension	
	(16)	Channel number	
Load	(0)	Device name	
	(2)	Filename 1	
	(4)	Filename 2	
	(6)	Extension	
			date format
Time	(0)	Date	[0/month/ day /year-1972] 1 bit 5 bits 5 bits 5 bits
	(2)	Upper word of time	(10 millisecond units)
	(4)	Lower word of time	(10 millisecond units)

Table 3-2. SVC Argument Passing (continued)

SVC NAME	OFFSET	PARAMETER	DESCRIPTION
Status	(0)	Channel number	used for RS-232 devices
	on return:		users R0 contains data available status: status=0 implies no data status<>0 implies data available status is zero until at least one line of data has accumulated (i.e. until an end-of-line symbol has been read).
Config	(0)	Device name	
	(2)	Read/Write	R/W<>0 implies that a change in device configuration values will be written into AREA R/W=0 implies that the device configuration values will be read in from AREA
	(3)	baud rate	0 = 75 5 = 600 10 = 4800 1 = 110 6 = 1200 11 = 7200 2 = 134.5 7 = 1800 12 = 9600 3 = 150 8 = 2400 13 = 19200 4 = 300 9 = 3600
	(4)	uart control	universal asynchronous receive/transmit -- described in "TI 9900 Family Systems Design" manual
	(5)	end of line value	
	(6)	end of file value	
Exit	-	No parameters	

3-13. FDOS I/O Manager Errors

3-14. FDOS may encounter an error during the execution of an I/O Manager operation. (For example, a floppy disk could be write protected when a WRITE SVC is called.)

0 If an error is encountered while executing an SVC call, the program continues executing at the first word following the call.

0 If no error is encountered, the program skips the first word after the call and continues at the next location.

MEMORY LOCATION	CONTENTS
>004A	pointer to AREA
AREA	parameter, offset is 0
AREA+2	parameter, offset is 2
AREA+4	parameter, offset is 4
.	. . .
.	. . .
.	. . .
AREA+2n	parameter, offset is 2n

For the SVC call LOAD, the offset for each parameter is

PARAMETER	OFFSET
device name	0
filename 1	2
filename 2	4
extension	6

Figure 3-1. AREA Offsets

0 Whenever FDOS encounters an error, it passes an error number (listed in Table 3-3) in Register R0 of the calling workspace.

1. The user program should contain an error exit, such as a JUMP statement, in the first location following the SVC. It should exit the Call subroutine and jump to an error display subroutine. Failure to exit the SVC call when an error is detected may lead to a sudden termination of system operations.

2. The error numbers passed in Register R0 are NOT displayed or listed. A user subroutine must be written to take recovery action, such as printing an error message.

3-15. The following example shows a program that assigns supervisory calls, moves the Area Pointer from location >00A4 to Register R3, places each parameter into the location pointed to by the contents of R3 (with the appropriate offset), and performs the READ operation.

EXAMPLE:

```
        DXOP    SVC,15              ;ASSIGN SUPERVISORY CALL
READ    EQU     2
AREA    EQU     >A4

(1)  START  MOV     @AREA,R3        ;GET POINTER TO ARGUMENT AREA
(2)         CLR     *R3+            ;SET UP CHANNEL NUMBER 0
(3)         LI      R4,31           ;SET UP THE
(4)         MOV     R4,*R3+             BLOCK NUMBER
(5)         LI      R4,BUFFER       ;SET UP THE POINTER
(6)         MOV     R4,*R3+             TO THE READ BUFFER
(7)         LI      R4,512          ;SET UP THE BLOCK
(8)         MOV     R4,*R3              LENGTH (512 BYTES)
(9)         SVC     READ            ;PERFORM THE READ OPERATION
(10)        JMP     ERROR           ;ERROR EXIT
(11)        ***                     ;CONTINUE ON
```

Explanation:

(1) R3 will point to the first element of the argument area.

(2) The first element is the channel number, which will be cleared in order to indicate channel 0.

(3) The second element is the block number, which is set to 31 in order to access the thirty-first block of a previously opened file.

(5) The third element must point to where the block is to be loaded.

(7) The fourth element indicates how many bytes are going to be read.

(9) This is the SVC call to FDOS.

(10) If there is an error, the program jumps to the subroutine labeled ERROR.

(11) The program comes here if the read was successful.

Table 3-3. FDOS Execution Errors

ERROR NO.	MEANING
0	No error
2	Device hardware error
4	Device not ready
6	Medium write protected
8	Illegal channel number
10	Channel in use
12	Not a valid device name
14	File not found
16	No room on medium
18	Attempt to read/write past end of file
20	Channel not open
22	Devices do not match in 'RENAME'
24	Non-existent memory
26	Non-file structured device interrupt buffer overflow
28	Medium was changed
30	Illegal directory

3-16. DIRECT CALLS TO DEVICE HANDLERS

3-17. Introduction

3-18. Each device on the system has a unique handler which provides an interface between the FDOS and the hardware. Five of the 13 XOP levels (0 through 12) which are allocated to the device handlers are in use: 0, 1, 2, 3, 4, 5, and 12. The mnemonic and XOP number associated with each of the five device handlers are listed in Table 3-4. The I/O manager handles all of the devices except GPIB, so only direct calls to the KB0 and GPIB handlers are recommended. KB0 handles the keyboard and the video display. Since the I/O manager does not handle the GPIB device, GPIB must be called directly.

3-19. XOP Format

3-20. With the exception of GPIB, the XOPs which pass information to and from a handler device all have the following format:

```
XOP          pntr,nmbr
```

- nmbr represents the particular XOP number.
- pntr is a pointer that points to the data area that contains the parameters and has the format:
  1. The first word is the absolute block number.
  2. The second word is the buffer address.
  3. The third word is two bytes. The upper byte is the function code and the lower byte is not used.
  4. The fourth word is the record length in bytes.
  5. The fifth word is the error code.
- These parameters can be passed in both directions.

Table 3-4. XOP Numbers of Device Handlers

DEVICE	MNEMONIC	XOP NUMBERS
Floppy Disk	MF0	0
Keyboard & Video	KB0	1
RS-232 Serial Port #1	KB1	3
RS-232 Serial Port #2	KB2	4
Electronic Disk	ED0	5
IEEE-488 Bus Ports	GPIB	12

3-21. DEVICE HANDLER INTERFACES

3-22. Introduction

3-23. The following paragraphs describe the interfaces to the five handlers.

3-24. Minifloppy Disk Handler (MF0)

3-25. The MF0 device handler has the following characteristics:

- MF0 is called by XOP number 0.

- MF0 provides the interface to the 1720A Instrument Controller's disk hardware.

- MF0 Error codes are:

ERROR NO	DESCRIPTION
0	No error
2	Hard diskette error
4	Diskette not loaded
6	Diskette write protected
24	Non-existent memory
28	Diskette was swapped

- MF0 Function codes are defined as follows:

NUMBER	NAME	ACTIONS
0	Restore Head	The floppy disk read/write head is moved to a position over track 0.
2	Read a Block	N bytes of data (record length) are transferred from the floppy diskette to main memory (buffer address). Prior to the start of the transfer the read/write head is positioned at the specified block (absolute block number).
4	Write a Block	N bytes of data (record length) are transferred from memory (buffer length) are transferred from memory (buffer address) to the diskette on MF0. Prior to the start of the transfer the read/write head is positioned at the specified block (absolute block number).
6	Test Medium Swapped	Test a bit in the floppy interface to determine if the diskette in Drive 0 has been removed and replaced (swapped) since the diskette was swapped. This function is executed prior to accessing a memory resident it describes is still mounted.
8	Step In	Move read/write head to next higher numbered track.
10	Write Track	One full track, both data and format information, is written to the diskette on the designated drive (MF0: or MF1:). Both data and format bytes must appear in the designated buffer (buffer address). See Appendix A of this manual for a description of the format bytes.

3-26. Video/Keyboard Handler (KB0)

3-27. The KB0 device handler has the following characteristics:

- KB0 is called by XOP number 1.
- The video screen and the keyboard together are the primary user-system interface. The handler determines many of the characteristics of that interface.
- KB0 has no Error codes, but does have an error exit for Function code 2 (Read a Record) and Function code 4 (Write a Record).
- KB0 Function codes are as follows:

NUMBER	NAME	ACTIONS
0	Read a Line	Returns a line from the console or a command file starting at the locations pointed to by the buffer pointer. No length is returned.
2	Read a Record	Returns a line (= record) from the console or a command file. The length of the record is returned.
4	Write a Record	Writes a record to the console terminals. Writing is complete when the record length is reached, an end-of-file character is encountered, or when a CTRL/C is given.
6	Read a Character	Returns a character from the console terminal or the command file in R8 of the calling workspace.
8	Write a Character	Writes a character which is in R8 of the calling workspace to the console terminal.
10	Set Input Mode	If the value of the buffer pointer is 0, then the line mode is entered. Otherwise, the driver will go into the single character mode.
12	Write a Line	Writes a line to the screen which is pointed to by the buffer pointer. The last character must be negated. When a command file is active, this function is ignored.
14	No Action	Used by other non-file structured devices to reset their queues.
16	Return Status	Checks to see is at least one line (in Line Mode) or character (in Character Mode) is available for input. If no line or character is available, R0 (the user's register) is set to zero. Otherwise the number of line(s) or character(s) is returned.

NUMBER	NAME	ACTIONS
18	Configure	Checks the R/W flag. If a read operation is requested, the baud rate protion of the port configuration is returned in AREA. If a write operation is requested, only the baud rate of KB0 is updated.

0 The video/keyboard handler can operate in the Line Mode or in the Character Mode.

1. Line Mode

   a. When KB0 is in the Line Mode, a line is not returned to the display until KB0 receives a line terminator (line terminator can be <cr>, CTRL/Z, or CTRL/T).

   b. The following control key operations are active:

      (1) CRTL/U which deletes the current line. CTRL/U echoes (displayed on the screen) as ^U<cr><lf>.

      (2) CTRL/C which deletes all lines in buffer and sets the control/C flag (CC) to a value of 3. CTRL/C echoes as ^C<cr><lf>.

      (3) CTRL/P which deletes all lines in buffer, terminates the current programs and command file (if active), closes all files, and gives control back to CONMON. CTRL/P echoes as ^P<cr><lf>.

      (4) CTRL/S which stalls the output to the display. CTRL/S does not echo.

      (5) CTRL/Q which restarts the stalled output to the display. CTRL/Q does not echo.

      (6) CTRL/Z which terminates the current line. CTRL/Z acts as the End-of-file and echoes as ^Z<cr><lf>.

      (7) CTRL/T which terminates the current line. CTRL/T echoes as <ESC>[;2;4p]<cr><lf>, which clears the screen and sets the screen to no graphics, small size, and no keyboard lock.

      (8) Any control key operations not listed are stored in the input buffer and are echoed as a >7F character, which is an inverse video block.

2. Character Mode

   a. The display handler can accept a signal from the console display (and input buffer) without requiring a line terminator. Echoing is disabled.

   b. The keyboard entries <DELETE>, CTRL/U, CTRL/Z, CTRL/P, CTRL/C, CTRL/T, and <RETURN> are not active, but are returned as are all other entries. CTRL/C and CTRL/P also store their respective values in CC (CTRL/C = 3 and CTRL/P = 16).

0 The operation of the Video/keyboard handler is affected by the status of the Command File Processor (see Appendix A).

3-28. The first of the following examples shows a program that prints a line of test to the console device in Line mode. The second of the following examples shows a program that puts KBO into the Character Mode and accepts a single character from the keyboard.

EXAMPLE 1:

The program is as follows:

```
(1)            BL      @PRINT          ;PRINT MESSAGE
(2)            DATA    T1
     *
(3)  T1        TEXT    -'This is a message'
(4)            DXOP    KBO,1           ;ASSIGN KEYBOARD DRIVER
     *                                    CALL
(5)  PRINT     EQU     $               ;SUBROUTINE ENTRY POINT
(6)            MOV     *R11+,@PP+2     ;STORE TEXT POINTER IN
     *                                    ARGUMENT LIST
(7)            KBO     @PP             ;PRINT THE LINE
(8)            RT                      ;RETURN
(9)  PP        EQU     $-2             ;ARGUMENT LIST
(10)           BSS     2               ;EMPTY SPOT FOR TEXT
     *                                    POINTER
(11)           BYTE    12              ;FUNCTION CODE
(12)           EVEN
```

The explanation of the program is as follows:

LINE — EXPLANATION

(1) This is an example of the use of the subroutine PRINT.

(2) The DATA statement following the subroutine call must point to the beginning of the text to be printed

(3) This is a typical text statement. Note the minus sign, which causes the last byte to be negated so the driver knows when to terminate printing.

(4) This allows a symbolic call to the driver.

(5) This is the required print subroutine.

(6) The word following the call is stored in the second word of the argument list PP.

(7) The call is made here, with a pointer to the argument list.

(8) Return to caller.

(9) This is the start of the argument list. Since the first word is not used with this function code, the label of the argument is set one word backwards.

(10) The pointer to the text is written in this word.

(11) This function code tells the driver to print one line of text.

(12) This directive causes the location counter to start on an even boundary again.

EXAMPLE 2:

The program is as follows:

```
(1)             KB0       @SM             ;GO INTO SINGLE CHARACTER
                                                MODE
(2)             KB0       @ACCEPT         ;ACCEPT A CHARACTER
                ***
(3)  SM         EQU       $-2             ;SINGLE CHARACTER MODE
(4)             DATA      -1              ;SET BOOLEAN TO NON ZERO
(5)             BYTE      10              ;FUNCTION CODE
                EVEN
(6)  ACCEPT     EQU       $-4             ;ACCEPT A CHARACTER
(7)             BYTE      6               ;FUNCTION CODE
                EVEN
```

The explanation is as follows:

LINE EXPLANATION

(1) This call tells the console driver to accept single characters from now on.

(2) This call requests a single character from the console keyboard. The character is returned in R8.

(3) This points to the beginning of the argument list. Only two words are necessary.

(4) This location acts as a boolean, if non-zero character mode is indicated.

(5) Function code 10.

(6) This points to the beginning of the argument list. Only one word is needed.

(7) Function code 6.

3-29. RS-232 Port Handlers (KB1/KB2)

3-30. The KB1 and KB2 device handlers have the following characteristics:

- KB1 is called by XOP number 3.
- KB2 is called by XOP number 4.
- KB1 and KB2 are two identical RS-232 ports
- The error code for KB1 and KB2 is:

ERROR NO	DESCRIPTION
26	Lost data on input

- KB1 and KB2 Function codes are defined as follows:

NUMBER	NAME	ACTIONS
2	Read a Record	Returns a record (line) from the specified port ('KB1'/'KB2') when a line feed character is input or when 512 characters have been input. The record will also be return if a CTRL/C is seen on the console terminal. If the 132-character input interrupt buffer is full when a character is ready to be input, an error 26 occurs ('lost data on input'). The record length is returned.
4	Write a Record	A record is written to the specified port ('KB1'/'KB2'). Writing is complete when the record length is reached, when a CTRL/C is seen on the console terminal, or when an End-of-File character is encountered.
14	Reset Ring Buffers	The interrupt ring buffers are cleared and the lost data flag is cleared. This function is executed by the OPEN XOP handler.

NUMBER	NAME	ACTIONS
16	Return Status	A check is made to see if at least one line of data is available for input. If no lines are available, the user's register R0 is set to zero. Otherwise, the number of lines available is returned in R0.
18	Configure	The R/W flag is checked. If a read operation is requested, the currect configuration of the port is returned in AREA. If a write operation is requested, the information in AREA is used to modify the port's configuration.

3-31. E-Disk Handler (EDO)

3-32. The EDO device handler has the following characteristics:

- EDO is called by XOP number 5.
- EDO is the interface to the electronic disk memory board.
- EDO Error codes are:

ERROR NO	DESCRIPTION
0	No Error
2	Parity Error

- EDO Function codes are defined as follows:

NUMBER	NAME	ACTIONS
2	Read a Block	'N' bytes of data ('record length') are transferred from the E-Disk to main memory ('buffer address'). The Transfer is from the specified block ('absolute block number').
4	Write a Block	'N' bytes of data ('record length') are transferred from main memory ('buffer address') to the E-Disk at a specified block ('absolute block number').

3-33. IEEE-488 Bus Handler (GPIB)

3-34. The IEEE-488 Bus handler has a different structure than the other device handlers. The primary differences include:

- The parameters passed to the handler
- The function codes
- The error codes.

3-35. The GPIB (General Purpose Interface Bus) handler can handle two IEEE-488 standard interface compatible boards with two ports each.

- The GPIB handler is accessed via the XOP call

  XOP pntr,12

  1. pntr is the pointer which points to the data area containing the parameters.

  2. 12 is the XOP number.

- The GPIB handler is accessed directly, rather than going through the I/O Manager as the other handlers do.

- The four ports are numbered from 0 through 3.

- The format of the data area to which the pointer refers is as follows:

first word:	addressed ports mask
second word:	pointer to device list
third word:	
a. upper byte:	unused
b. lower byte:	function code
fourth word:	unused
fifth word:	error return code

  1. The first word is the addressed ports mask.

     a. In the addressed ports mask, the bit settings denote different ports.

     b. The mask addresses port number n when the 2^n bit is set in the mask (the notation 2^n means the nth bit).

     c. A null mask contains all zeros.

     d. The addressed ports mask is usually referred to as the port mask.

  2. The second word is the pointer to device list.

     a. A pointer device list is used for commands which require that a device or set of devices be addressed (such as the commands in a PRINT routine).

     b. A sequence of one-byte integers is used to address the particular port, the primary address, and any secondary addresses required.

     c. A secondary address is distinguished from the primary address by having the 2^6 bit in the address word set.

d. The end of the list is designated by having the 2^7 bit (the SIGN bit) set.

e. The end-of-list pointer is not used for addressing. It means that the last entry has already been processed not that this is the last entry.

f. The format of the device list to which the pointer points to is shown in Figure 3-2.

3. The upper byte of the third word contains the function codes.

a. For each specific call to the GPIB handler, a function code specifies what operation the handler is to perform.

b. The following list provides the function code number for each of the 24 GPIB function codes, describes the function, and refers to a figure in which the function is described in detail.

FUNCTION CODE NUMBER	DESCRIPTION	FIGURE NUMBER
0	Initialize bus	5-3
2	Set REN line	5-4
4	Reset REN line (local)	5-5
6	Send GO TO LOCAL command	5-6
8	Address a group of listeners	5-7
10	Address a talker	5-8
12	Send DEVICE CLEAR	5-9
14	Send SELECTED DEVICE CLEAR	5-10
16	Send LOCAL LOCKOUT	5-11
18	Send GROUP EXECUTE TRIGGER	5-12
20	Send PARALLEL POLL CONFIGURE	5-13
22	Send PARALLEL POLL DISABLE	5-14
24	Read binary data	5-15
26	Write binary data	5-16
28	Input a line of data	5-17

FUNCTION CODE NUMBER	DESCRIPTION	FIGURE NUMBER
30	Print a line of data	5-18
32	Set timeout limit	5-19
34	Set INPUT terminator	5-20
36	Return active SRQ mask	5-21
38	Update and return active SRQ mask	5-22
40	Perform serial poll	5-23
42	Perform parallel poll	5-24
44	Send INTERFACE CLEAR	5-25
46	Re-enable SRQ interrupts on port	5-26

4. The fifth word contains the error return codes. If an error is encountered during processing, an error code is returned in the last parameter location of the function code. The error codes are as follows:

ERROR CODE NO.	DESCRIPTION
2	Input buffer overflow
4	Illegal function code
6	Handshake not completed
8	Too may ports specified
10	No devices attached to port
12	No ports available
14	Illegal or unavailable port specified
16	BUS I/O timeout
18	EOI and ATN specified to be output with a WBYTE command

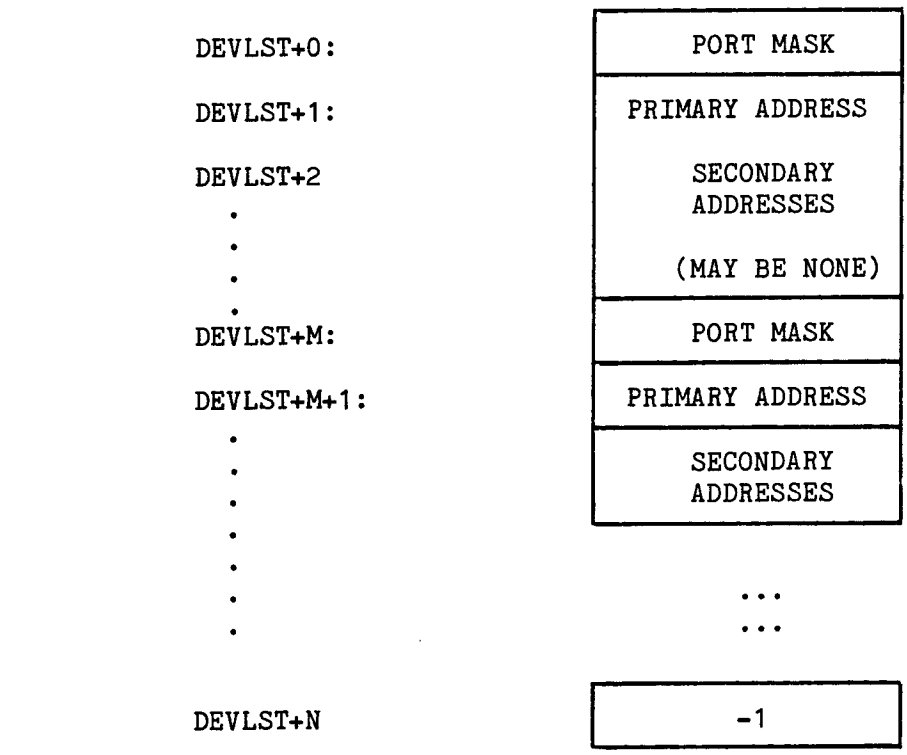


Figure 3-2. Device List Format

FUNCTION CODE NAME: Initialize Bus

FUNCTION CODE NUMBER: 0

DESCRIPTION: Sets the 'IFC' bus line TRUE for 100 microseconds (100 usec) on the designated ports. It then sends a PPU (Parallel Poll Unconfigure) an UNL (Unlisten), and an UNT (Untalk) on the bus.

INPUT PARAMETERS:

LOCATION	CONTENTS
PARMS+0:	Port Mask
PARSM+2:	unused
PARMS+4:	Function Code
PARMS+6:	unused
PARMS+8:	Error Return Word

BUS SIGNALS:

REN
-ATN
IFC
-IFC (after 100 usec delay)
ATN
UNL
UNT
PPU

ERRORS RETURNED:

ERROR NUMBER	DESCRIPTION
10	No devices attached to port
16	Bus timeout

FIGURE 3-3. Initialize Bus Function Code

FUNCTION CODE NAME: Remote

FUNCTION CODE NUMBER: 2

DESCRIPTION: Sets the Remote Enable line (REN) true on the designated ports.

INPUT PARAMETERS:

LOCATION	CONTENTS
PARMS+0:	Port Mask
PARMS+2:	Unused
PARMS+4:	Function Code
PARMS+6:	Unused
PARMS+8:	Error Return Word

BUS SIGNALS: REN

ERRORS RETURNED: NONE

Figure 3-4. Remote Function Code

FUNCTION CODE NAME: Local

FUNCTION CODE NUMBER: 4

DESCRIPTION: Causes the Remote Enable line (REN) on the designated ports to be reset. This is done only after any active transactions on the bus are completed. If they are not completed in the appropriate time, a Timeout error occurs.

INPUT PARAMETERS:

LOCATION	CONTENTS
PARMS+0:	Port Mask
PARMS+2:	unused
PARMS+4:	Function Code
PARMS+6:	unused
PARMS+8:	Error Return Word

BUS SIGNALS: -REN (after completion of last handshake)

ERRORS RETURNED:

ERROR NUMBER	DESCRIPTION
16	Bus Timeout

Figure 3-5. Local Function Code

FUNCTION CODE NAME: Send GO-TO-LOCAL

FUNCTION CODE NUMBER: 6

DESCRIPTION: Addresses a set of devices as listeners and then sends the Go-To-Local (GTL) command on the bus.

INPUT PARAMETERS:

LOCATION	CONTENTS
PARMS+0:	Port Mask
PARMS+2:	Pointer to Devlist
PARMS+4:	Function Code
PARMS+6:	unused
PARMS+8:	Error Return Word

BUS SIGNALS:
ATN
UNL
UNT
MLA (For each device addressed)
GTL

ERRORS RETURNED:

ERROR NUMBER	DESCRIPTION
10	No device attached to port
16	Bus timeout

Figure 3-6. Send GO-TO-LOCAL Function Code

FUNCTION CODE NAME: Address Listeners

FUNCTION CODE NUMBER: 8

DESCRIPTION: Sends a listen address (MLA) to the devices specified in the device list pointed to by the pointer in location PARMS+2.

INPUT PARAMETERS:

LOCATION	CONTENTS
PARMS+0:	Pointer to Devlst
PARMS+2:	unused
PARMS+4:	Function Code
PARMS+6:	unused
PARMS+8:	Error Return Word

BUS SIGNALS:
ATN
UNL
UNT
MLA (For each device addressed)

ERRORS RETURNED:

ERROR NUMBER	DESCRIPTION
10	No devices attached to port
16	Bus Timeout

Figure 3-7. Address Listeners Function Code

FUNCTION CODE NAME: Address Talker

FUNCTION CODE NUMBER: 10

DESCRIPTION: Sends a talk address (MTA) to the device specified in the XOP Call.

INPUT PARAMETERS:

LOCATION	CONTENTS
PARMS+0:	Port Mask
PARMS+2:	Pointer to Devlst
PARMS+4:	Function Code
PARMS+6:	unused
PARMS+8:	Error Return Word

BUS SIGNAL:
ATN
UNL
UNT
MTA

ERRORS RETURNED:

ERROR NUMBER	DESCRIPTION
10	No device attached to port
16	Bus Timeout

Figure 3-8. Address Talker Function Code

FUNCTION CODE NAME: Device Clear

FUNCTION CODE NUMBER: 12

DESCRIPTION: Sends a Device Clear (DCL) command to all ports addressed by the call.

INPUT PARAMETERS:

LOCATION	CONTENTS
PARMS+0:	Port Mask
PARMS+2:	unused
PARMS+4:	Function Code
PARMS+6:	unused
PARMS+8:	Error Return Word

BUS SIGNALS:
ATN
UNL
UNT
DCL

ERRORS RETURNED:

ERROR NUMBER	DESCRIPTION
10	No devices attached to port
16	Bus timeout

Figure 3-9. Device Clear Function Code

FUNCTION CODE NAME: Send Selected Device Clear

FUNCTION CODE NUMBER: 14

DESCRIPTION: Addresses (as listeners) the set of devices specified by the device list. It then sends a Selected Device Clear (SDC) command to the ports on which the listeners reside.

INPUT PARAMETERS:

LOCATION	CONTENTS
PARMS+0:	Port Mask
PARMS+2:	Pointer to Devlst
PARMS+4:	Function Code
PARMS+6:	unused
PARMS+8:	Error Return Word

BUS SIGNALS:
ATN
UNL
UNT
MLA (For each device in list)
SDC
UNL

ERRORS RETURNED:

ERROR NUMBER	DESCRIPTION
10	No devices attached to port
16	Bus timeout

Figure 3-10. Send Selected Device Clear Function Code

FUNCTION CODE NAME: Local Lockout

FUNCTION CODE NUMBER: 16

DESCRIPTION: Sends the Local Lockout (LLO) message to the ports addressed by the call, after 'REN' is set true.

INPUT PARAMETERS:

LOCATION	CONTENTS
PARMS+0:	Port Mask
PARMS+2:	unused
PARMS+4:	Function Code
PARMS+6:	unused
PARMS+8:	Error Return Word

BUS SIGNALS:
REN
ATN
LLO

ERRORS RETURNED:

ERROR NUMBER	DESCRIPTION
10	No devices attached to port
16	Bus timeout

Figure 3-11. Local Lockout Function Code

FUNCTION CODE NAME: Send Group Execute Trigger

FUNCTION CODE NUMBER: 18

DESCRIPTION: Sends a listen address (MLA) to the set of devices specified in the device list pointed to in the XOP call. The Group Execute Trigger MEssage (GET) is then sent to the ports on which the listeners reside.

INPUT PARAMETERS:

LOCATION	CONTENTS
PARMS+0:	Port Mask
PARMS+2:	Pointer to Devlst
PARMS+4:	Function Code
PARMS+6:	unused
PARMS+8:	Error Return Word

BUS SIGNALS:
ATN
UNL
UNT
MLA (For each device in DEVLST)
GET

ERRORS RETURNED:

ERROR NUMBER	DESCRIPTION
10	No device attached to port
16	Bus timeout

Figure 3-12. Send Group Execute Trigger Function Code

FUNCTION CODE NAME: Send Parallel Poll Configure

FUNCTION CODE NUMBER: 20

DESCRIPTION: Sends a listen address (MLA) to the device specified in the device list pointed to in the XOP Call. The device is then configured as specified in call parameter PARMS+6. The SENSE in the low byte is either 0 or 1, while the BIT NO. in the high byte may range from 0 to 7, indicating the DIO line for the response to a parallel poll. The SENSE and the BIT NO. parameters are right-justified in the low and the high bytes, respectively of the Call word.

INPUT PARAMETERS:

LOCATION	CONTENTS
PARMS+0:	Port Mask
PARMS+2:	Pointer to Devlst
PARMS+4:	Function Code
PARMS+6:	Bit No. ¦ Sense
PARMS+8:	Error Return Word

BUS SIGNALS:
ATN
UNL
UNT
MLA
PPC
PPE (ORed with 'BIT NO.' and 'SENSE')
UNL

ERRORS RETURNED:

ERROR NUMBER	DESCRIPTION
10	No devices attached to port
16	Bus timeout

Figure 3-13. Send Parallel Poll Configure Function Code

FUNCTION CODE NAME: Send Parallel Poll Disable

FUNCTION CODE NUMBER: 22

DESCRIPTION: Sends a listen address (MLA) to the device specified on the device list pointed to in the XOP call. A Parallel Poll Disable (PPD) command is then sent on that port. Only one port may be specified.

INPUT PARAMETERS:

LOCATION	CONTENTS
PARMS+0:	Port Mask
PARMS+2:	Pointer to Devlst
PARMS+4:	Function Code
PARMS+6:	unused
PARMS+8:	Error Return Word

BUS SIGNALS:
ATN
UNL
UNT
MLA
PPD
UNL

ERRORS RETURNED:

ERROR NUMBER	DESCRIPTION
10	No devices attached to port
16	Bus timeout

Figure 3-14. Send Parallel Poll Disable Function Code

FUNCTION CODE NAME: Read Binary Data

FUNCTION CODE NUMBER: 24

DESCRIPTION: Moves data from the bus to either word arrays or byte arrays in memory.

0 If the 'OUTPUT FIELD FLAG' is zero, word-aligned output is to be generated. When the data is placed in a word array, the low order byte contains the value and the high order byte indicates whether or not EOI was asserted with that data byte. If EOI was asserted, the 2^8 bit of that word will be set.

0 If the 'OUTPUT FIELD FLAG' is nonzero, byte-aligned output is to be generated. No EOI data is available with the byte array.

INPUT PARAMETERS:

LOCATION	CONTENTS
PARMS+0:	Port Mask
PARMS+2:	Pointer to Array
PARMS+4:	Function Code
PARMS+6:	Number of Bytes to Read
PARMS+8:	Error Return Word
PARMS+10:	Output Field Flag

BUS SIGNALS: NONE -- The array is filled with data bytes.

ERRORS RETURNED:

ERROR NUMBER	DESCRIPTION
10	No devices attached to port
16	Bus timeout

Figure 3-15. Read Binary Data Function Code

FUNCTION CODE NAME: Write Integer Array

FUNCTION CODE NUMBER: 26

DESCRIPTION: Sends the contents of the word-aligned array pointed to in the call parameters onto the bus. The 'EOI' line is set if the 2^8 bit is set; the 'ATN' line is set if the 2^9 bit is set. However, an error is returned if both 'EOI' and 'ATN' are to be set. Each data byte should be right-justified in an array word. Only one port may be addressed by this command.

INPUT PARAMETERS:

LOCATION	CONTENTS
PARMS+0:	Port Mask
PARMS+2:	Pointer to Array
PARMS+4:	Function Code
PARMS+6:	Number of Bytes to Send
PARMS+8:	Error Return Word

BUS SIGNALS: EOI
ATN

ERRORS RETURNED:

ERROR NUMBER	DESCRIPTION
8	More than one port specified
16	Bus timeout
18	EOI and ATN both flagged

Figure 3-16. Write Integer Array Function Code

FUNCTION CODE NAME: Input A Line

FUNCTION CODE NUMBER: 28

DESCRIPTION: Moves data from the bus into the buffer pointed to by the pointer. The function is terminated when an EOI is detected, or when an input termination character is received (see Function Code 34). The number of data bytes actually placed in the buffer is returned.

INPUT PARAMETERS:

LOCATION	CONTENTS
PARMS+0:	Port Mask
PARMS+2:	Pointer to Buffer
PARMS+4:	Function Code
PARMS+6:	Buffer Length (bytes)
PARMS+8:	Error Return Word

OUTPUT: PARMS+6 Word --- Count of characters placed in the buffer.

BUS SIGNALS: Data Received

ERRORS RETURNED:

ERROR NUMBER	DESCRIPTION
2	Input buffer overflow
10	No deviced attached to port
16	Bus timeout

Figure 3-17. Input A Line Function Code

FUNCTION CODE NAME: Print A Line

FUNCTION CODE NUMBER: 30

DESCRIPTION: Sends the contents of a byte-aligned array pointed to by the pointer onto the bus.

0 PARMS+6 gives the number of bytes to be sent

0 PARMS+10 designates the order in which data is stored in the buffer:

1. A value greater than or equal to zero indicates that succeeding data bytes are stored in successively higher memory locations (normal order)

2. A value less than zero indicates that succeeding data bytes are stored in lower memory locations (reverse order)

0 PARMS+12 is set nonzero if EOI is to be asserted true when the last byte of data in the buffer is sent.

INPUT PARAMETERS:

LOCATION	CONTENTS
PARMS+0:	Port Mask
PARMS+2:	Buffer Address (pointer)
PARMS+4:	Function Code
PARMS+6:	Number of Bytes to Send
PARMS+8:	Error Return Word
PARMS+10:	Order of Data in Buffer
PARMS+12:	EOI on Last Byte Flag

BUS SIGNALS: The data is sent to the bus. EOI is sent with the last data byte if requested.

ERRORS RETURNED:

ERROR NUMBER	DESCRIPTION
16	Bus timeout

Figure 3-18. Print A Line Function Code

FUNCTION CODE NAME: Set Timeout Limit

FUNCTION CODE NUMBER: 32

DESCRIPTION:

0 Takes the time limit parameter in milliseconds (msec), converts it to 10-msec ticks by truncation, and stores it in the timer limit word. The time limit parameter (PARMS+2) should be given in msec.

0 A timeout limit of zero (0) disables any check for bus timeout.

0 The maximum time limit is 32767 milliseconds.

INPUT PARAMETERS:

LOCATION	CONTENTS
PARMS+0:	Port Mask
PARMS+2:	Timeout Limit
PARMS+4:	Function Code
PARMS+6:	unused
PARMS+8:	Error Return Word

BUS SIGNALS: None -- TIMLMT is set in 10 msec ticks

ERRORS RETURNED: None

Figure 3-19. Set Timeout Limit Function Code

FUNCTION CODE NAME: Set Terminating Character

FUNCTION CODE NUMBER: 34

DESCRIPTION: Sets the character to be used as the delimiter for an input string (as read by Function Code 28), or can indicate that EOI is to be the only terminator. In PARMS+2, Terminator is the terminating character to be used. Flag indicates that the current terminator, or that no terminator should be used.

0 Flag = 0: Replace current terminator with new one.

0 Flag <>0: Use no terminator except 'EOI' assertion.

INPUT PARAMETERS:

LOCATION	CONTENTS
PARMS+0:	unused
PARMS+2:	Terminator ¦ Flag
PARMS+4:	Function Code
PARMS+6:	unused
PARMS+8:	Error Return Word

BUS SIGNALS: None

ERRORS RETURNED: None

Figure 3-20. Set Terminating Character Function Code

FUNCTION CODE NAME: Test SRQ Status

FUNCTION CODE NUMBER: 36

DESCRIPTION: Causes a poll of each port to see if an SRQ is pending on that port. A port mask is returned in the PARMS+2 indicating the ports with active service requests.

INPUT PARAMETERS:

LOCATION	CONTENTS
PARMS+0:	unused
PARMS+2:	unused
PARMS+4:	Function Code
PARMS+6:	unused
PARMS+8:	Error Return Word

OUTPUT PARAMETERS: PARMS+2, which contains the port mask for ports which have active service requests.

BUS SIGNALS: None

ERRORS RETURNED: None

Figure 5-21. Test SRQ Status Function Code

FUNCTION CODE NAME: Return SRQ Status

FUNCTION CODE NUMBER: 38

DESCRIPTION: Similar to the Test SRQ Status (Function Code 36), except that Return SRQ Status does not cause a test of the ports. It only returns current status as reflected by the occurrence of past SRQ interrupts from the active GPIB ports.

INPUT PARAMETERS:

LOCATION	CONTENTS
PARMS+0:	unused
PARMS+2:	unused
PARMS+4:	Function Code
PARMS+6:	unused
PARMS+8:	Error Return Word

OUTPUT: PARMS+2, which contains the port mask for ports which have active service requests.

BUS SIGNALS: None

ERRORS RETURNED: None

Figure 3-22. Return SRQ Status Function Code

FUNCTION CODE NAME: Serial Poll

FUNCTION CODE NUMBER: 40

DESCRIPTION: Performs a serial poll on the device in the device list pointed to by PARMS+2, by addressing the device as a talker, sending Serial Poll Enable (SPE), and recieving the instrument response. The response is returned to the caller in PARMS+2, in a right-justified, zero-filled to left format.

INPUT PARAMETERS:

LOCATION	CONTENTS
PARMS+0:	Port Mask
PARMS+2:	Pointer to DEVLST
PARMS+4:	Function Code
PARMS+6:	unused
PARMS+8:	Error Return Word

BUS SIGNALS:

ATN
UNL
UNT
MTA
SPE
-ATN
(Accepts Status Byte from Device)
ATN
UNT
SPD

ERRORS RETURNED:

ERROR NUMBER	DESCRIPTION
10	No devices attached to port
16	Bus timeout

Figure 3-23. Serial Poll Function Code

FUNCTION CODE NAME: Parallel Poll

FUNCTION CODE NUMBER: 42

DESCRIPTION: Performs a parallel poll on the port specified in the XOP call. The response is returned to the caller in PARMS+2 in the low byte, with the high byte zeroed.

INPUT PARAMETERS:

LOCATION	CONTENTS
PARMS+0:	Port Mask
PARMS+2:	unused
PARMS+4:	Function Code
PARMS+6:	unused
PARMS+8:	Error Return Word

BUS SIGNALS:
ATN
EOI
(Accepts Poll data from Port)
-EOI

ERRORS RETURNED:

ERROR NUMBER	DESCRIPTION
10	No devices attached to port
16	Bus timeout

Figure 3-24. Parallel Poll Function Code

FUNCTION CODE NAME: Interface Clear

FUNCTION CODE NUMBER: 44

DESCRIPTION: Sets the Interface Clear (IFC) message, on the designated ports for 100 microseconds. No check is made for ongoing bus I/O before 'IFC' is asserted.

INPUT PARAMETERS:

LOCATION	CONTENTS
PARMS+0:	Port Mask
PARMS+2:	unused
PARMS+4:	Function Code
PARMS+6:	unused
PARMS+8:	Error Return Word

BUS SIGNALS: IFC
(Wait for at least 100 Microseconds)
-IFC

ERRORS RETURNED: None

Figure 3-25. Interface Clear Function Code

FUNCTION CODE NAME: Re-enable SRQ Interrupt

FUNCTION CODE NUMBER: 46

DESCRIPTION: Enables the SRQ interrupts on the ports designated by the port mask. Interrupts on the ports are disabled when SRQ is received. This function also resets the "SRQ Received" status for the ports designated in the call.

INPUT PARAMETERS:

LOCATION	CONTENTS
PARMS+0:	Port Mask
PARMS+2:	unused
PARMS+4:	Function Code
PARMS+6:	unused
PARMS+8:	Error Return Word

BUS SIGNALS: None

ERRORS RETURNED: None

Figure 3-26. Re-enable SRQ Interrupt Function Code

Appendix A
1720A Instrument Controller Disk File Structure

A-1. INTRODUCTION

A-2. The information in this appendix summarizes the structure of data and programs on floppy disks as implemented on the John Fluke Model 1720A Instrument Controller.

A-3. FILE-STRUCTURED DEVICE ORGANIZATION

A-4. A file-structured device is usually built out of blocks. Each block is 512 bytes in length. When the disk is zeroed, the File Utility Program (FUP) initializes the directory with the number of available blocks. The first 2 blocks are used for the directory. The remaining space is available for either program files or data files.

A-5. DIRECTORY LAYOUT

A-6. The length of the directory is expressed in segments. Each segment is two blocks. The directory consists of a header and a number of directory entries. Table A-1 lists contenst of the five directory header words and Table A-2 lists the contents of the seven directory segment words.

Table A-1. Directory Header

HEADER WORD	CONTENTS
First Word	Number of avaiable directory segments. Each segment is two blocks long.
Second Word	Current segment number. Currently set to 1.
Third Word	Not used.
Fourth Word	Number of extra words per entry.
Fifth Word	First available block number.

Table A-2. Directory Segment

SEGMENT WORD	CONTENTS
Word 0	Entry Status: 0 >100 is a tentative entry. A file has been opened, but not yet closed. 0 >200 is an empty entry. 0 >400 is a permanent entry. The file is a permanent file. 0 >800 is an end of directory segment.
Word 1	The first three characters of the file name written in Radix-50 notation.
Word 2	The last three characters of the file name written in Radix-50 notation.
Word 3	Extension written in Radix-50 notation.
Word 4	The length of the file in blocks.
Word 5	The channel number associated with a file is stored at this location when a file is tentatively opened as a new file.
Word 6	The date the file was created. If Word 6 is a zero, no date was available when the file was created. The date format is as follows: Bits 0-3 Year minus 1972 Bits 4-9 Day (1 to 31) Bits 10-14 Month (1 to 12) Bit 15 Zero

A-7. RADIX-50 FORMAT

A-8. RADIX-50 Format is descried in detail in Section 5 of this manual. The following material breifly describes RADIX-50 Format.

A-9. In RADIX-50 Format, three alphanumeric characters can be stored in only 16 bits by limiting the number of different characters to 40 (50 octal). Use the following procedure to find the RADIX-50 equivalent of three alphanumeric characters:

0 Assign the correct value to each alphanumeric character one of two ways:

1. Select the appropriate value for each character from Table A-3.

2. Compute the correct value as follows:

   a. Subtract 64 from the ASCII value of each alpha character.

   b. Subtract 18 from the ASCII value of each numerical character.

- The first character is given a numerical weight of 40*40, the second character is given a weight of 40, and the third character is given a weight of 1.
- The weighted values of the characters are then added together to obtain the RADIX-50 equivalent
- The assembler has a strict left-to-right priority (NOT a hierarchical one) in performing numerical operations.

  1. The expression 8+6*3 equals 42.

  2. The expression 8+3*6 equals 66.

  3. The expression 3*6+8 equals 26.

A-10. The following example shows how to find the RADIX-50 equivalent of MF0.

```
MF0     EQU     'M'-64*40+'F'-64*40+'0'-18 =
                77-66*40+70-64*40+48-18 = 21,070
```

NOTE

The user program must translate file and device names into RADIX-50 format. FDOS does not perform this operation.

Table A-3. RADIX-50 Format Value Allocation

CHARACTER	NUMBER	CHARACTER	NUMBER	CHARACTER	NUMBER
Space	0	M	13	Z	26
A	1	N	14	$	27
B	2	O	15	0	30
C	3	P	16	1	31
D	4	Q	17	2	32
E	5	R	18	3	33
F	6	S	19	4	34
G	7	T	20	5	35
H	8	U	21	6	36
I	9	V	22	7	37
J	10	W	23	8	38
K	11	X	24	9	39
L	12	Y	25		

A-10. MODIFIED IBM FORMAT

A-11. The 1720A uases a modified IBM format: MFM, 10 sectors per track, 1 block = 1 sector, and 512 bytes per sector.

NUMBER OF BYTES		HEX VALUE OF BYTE
80		4E
12		00
3		F6
1		FC (Index Mark)
16		4E
14		00
3		F5
1		FE (ID Address Mark)
1		Track Number (0 through 39)
1		Side Number (0 or 1)
1		Sector Number (1 through 10)
1		02
1	*	F7 (2 CRCs written)
22		4E
12		00
3		F5
1		FB (Data Address Mark)
512		DATA
1		F7 (2 CRCs written)
22		4E
598**		4E

* Write the bracketed field 10 times.

** Continue writing until FD 1791 interrupts out (approximately 598 bytes).

A-12. INTERLEAVING

A-13. The sectors in the disk are interleaved in order to increase throughput. Interleaving is accomplished by giving the sectors a number in their ID field which differes from the physical location. Sectors are also skewed from track to track to compensate for step times. Table A-4 shows the relationship between physical and logical sector numbering.

NOTE

The interleaving and skewing is not a requirement for the 1720A to operate properly so sectors can be numbered as they appear in physical order

Table A-4. Physical Versus Logical Numbering

TRACK	PHYSICAL SECTOR NUMBER									
	1	2	3	4	5	6	7	8	9	10
0	1	6	2	7	3	8	4	9	5	10
1	9	5	10	1	6	2	7	3	8	4
2	3	8	4	9	5	10	1	6	2	7
3	6	2	7	3	8	4	9	5	10	1
4	5	10	1	6	2	7	3	8	4	9
5	8	4	9	5	10	1	6	2	7	3
6	2	7	3	8	4	9	5	10	1	6
7	10	1	6	2	7	3	8	4	9	5
8	4	9	5	10	1	6	2	7	3	8
9	7	3	8	4	9	5	10	1	6	2

Appendix B
FDOS Global References

The following globals reside in low memory (>80 to >100) and can be used by application programs to monitor the state of FDOS as well as to find out where certain routines or data areas of FDOS reside.

NAME	LOCATION	USE
RDRECT	>80	This location points to read the directory routine.
WDRECT	>82	This location points to write the directory routine.
DIR	>84	This location points to the beginning of the memory resident directory.
BUFFER	>86	This location points to the read/write buffer.
DEVNAM	>88	This location points to the device name table. This table is built during initialization of the operating system and consists of the RADIX-50 representation of the device name of each linked-in module. A dummy entry is generated as the last item in the device table. This entry can be used by any program to generate a temporary device. When control is given back to FDOS via a SVC EXIT or a CTRL/P, this device will be deleted from the table.
DEVNUM	>8A	This location points to the device number table. This table is built during initialization of the operating system and consists of a byte for each device. The four lower bits of the byte represent the XOP number by which the device is accessed. Bit 4 of this byte is set to indicate that the device is non-file structured. The device numbers appear in the same order as their corresponding device names.
CURDIR	>8C	This location points to the location holding the key for the current directory. This key consists of the XOP instruction by which the driver of the device associated with the current directory can be accessed. The argument of this XOP instruction is always R0. For example, the XOP for driver 3 is XOP R0,3.

NAME	LOCATION	USE
DEVSIZ	>8E	This location points to the device size table. This table is built during initialization of the operating system and consists of a word indicating how many blocks are available on the dveice. The size numbers appear in the same order as their corresponding device names.
AREA	>A4	This location points to the parameter passing area. This location should be read only.
DATE	>A6	This location holds the current date, read only. See Appendix A for the format.
TIME1	>A8	This location contaisn the upper portion of current time, read only.
TIME2	>AA	This location contains the lower portion of current time, read only. Time is expressed as a two word integer that indicates the time elapsed since midnight in 10 msec increments.
CL	>E6	This location points to the first character of the command line. The command line is the user or command file entry following the prompt CONMON.
AA	>EA	This location points to the argument area. This area is used to pass an argument into a command file.
CW	>EC	This location points to the command file processor workspace.
CB	>EE	This location points to the buffer that contains the command file.
AP	>F0	This location points to the first byte of the argument for the command file. If this location is zero, then no argument was given.
KEY	>AC	This location contains the last character (as a word) entered from the console terminal with the parity bit set (overlay). This word may be cleared after reading.
VERIFY	>F2	This location is a flag. This flag indicates to certain device drivers that a read after write has to be performed in order to insure the integrity of written data.
RETRY	>F4	This location displays the number of retries a deviec handler needed to read a block of data. This number is reset every time the handler is entered again.

NAME	LOCATION	USE
CF	>F6	This location is the command file active flag. See the command file interpreter module (Section 5) for a detailed explanation of this flag.
CC	>F8	This location is the CTRL/C flag. This flag is set to a value other than zero every time a CTRL/C or CTRL/P is sent by the console terminal. The user can reset this flag.
OPWORK	>FA	This location points to the first memory location occupied by FDOS. This location can be used to show the top of available memory.