List of Software Versions

System Disk	Rev. 1
Getting Started Disk	Rev. 1
System Diagnostic Disk	Rev. 0

1752A

DATA ACQUISITION SYSTEM

System Guide

P/N 760553
May 1985 Rev 1 10/86

FLUKE ®

1752A Data Acquistion System

Contents

1 **HOW TO USE THIS MANUAL** **1-1**

Introduction 1-3
  The Manual Set 1-3
  Additional Manuals 1-3
Organization 1-4
Usage Guide 1-6
How to Read Syntax Diagrams 1-8
  Sample Syntax Diagrams 1-9
Notation Conventions 1-10

2 **SETTING UP THE 1752A** **2-1**

The 1752A Data Acquisition System 2-3
  Fluke Enhanced BASIC 2-4
  Floppy Disk Operating System 2-4
  Data Acquisition and Control Software 2-5
Features 2-6
Measurement and Control Modules 2-8
1752A Configurations 2-9
Option Configuration 2-9
Physical Layout 2-10
Unpacking 2-12
Installation 2-14
  Environment 2-14
  Pre-Installation Checkout 2-15
  Front Panel Controls 2-20
  Default Switch Settings 2-21
  Configuring the Measurement and Control Modules 2-21

(continued on page ii)

Placement 2-22
Workbench Installation 2-22
Rack Mounting 2-24
Installing the Data Acquisition and Control System 2-26
Installing an IEEE-488 System 2-28
Connecting the IEEE-488 Bus 2-29
Conclusion 2-30

3 SOFTWARE CONFIGURATION 3-1

Introduction 3-2
Loading the System Software 3-3
Bootstrap Loader 3-3
Self-Test Error Messages 3-4
Other Errors 3-5
The Startup Command File 3-6
Setting the Time 3-7
The Operating System 3-9
Command Line Interpreter 3-10
Other Software on the System Disk 3-10
Making a New Operating System 3-12
Introduction 3-12
A Note About Software Compatibility 3-12
Using the System Generation Utility 3-13
The FDOS Configuration Data Program 3-15
Entering CONFIG 3-15
Writing Configuration Data to a File 3-16
CONFIG Program Help Information 3-16
Printing the CONFIG Program Version Number 3-17
The Compatibility Test Program 3-18
Running the Compatibility Program 3-18
Program Help Information 3-19
Printing More Data From Image Files 3-19
Directing Output to a File 3-21
Using OTP 3-21
Conclusion 3-22

4 DEVICES AND FILES 4-1

Introduction 4-2
Devices 4-3
Files 4-5

 (continued on page iii)

System Level Files 4-5
Other Files 4-5
Devices 4-6
RS-232 Ports (KBn:, SPn:) 4-6
IEEE-488 Ports (GPn:) 4-7
Floppy Disk Drives (MFn:) 4-7
Electronic Disk (ED0:) 4-8
Bubble Memory (MBn:) 4-8
Winchester Drive (WDn:) 4-8
Files 4-9
System Files 4-9
Alias File 4-9
Command Files 4-9
The Startup Command File 4-9
Other Command Files 4-10
Machine Executable Files 4-10
Language-Dependent Files 4-10
Source Files 4-10
Object Files 4-10
The File Utility Program 4-11
Introduction 4-11
Entering the File Utility Program 4-11
The Help Command 4-11
Directory Allocation 4-13
Using the File Utility Program 4-14
Wild Cards * and ? 4-15
Protection States + and – 4-16
Switches I, D, and S 4-17
Alphabetical Listing of Commands 4-18
(no option) Transfer 4-18
/A Assign the System Device 4-20
/B Binary Transfer 4-20
/C Configure Electronic Disk Space 4-20
/D Deleting Files 4-21
/E Listing a Directory (Also /L and /Q) 4-21
/F Format, Verify, and Zero a File Device 4-24
/I Interactive Transfer 4-25
/L Listing 4-25
/M Merging ASCII Files 4-26
/P Packing a File Structured Device 4-27
/Q Quick Directory 4-27
/R Renaming a File 4-27
/S Scanning for Bad Blocks 4-27

 (continued on page iv)

/T Transferring Files Without Error Check 4-28
/W Whole Copying a File Device 4-28
/X Exit 4-30
/Z Zeroing a File Directory 4-30
/+ /- Assigning Protection State 4-31
Syntax Diagrams 4-32
Directly Executed Commands 4-32
Directory Listing Commands 4-32
File Transfer (Copy) Commands 4-33
File Rename Command 4-33
File Merge Command 4-34
Whole Copy Command 4-34
File Deletion and Protection Commands 4-34
List Bad Blocks Command 4-35
Device Control Commands 4-35
System Messages 4-36
The Touch Copy Program 4-39
Conclusion 4-49

5 COMMUNICATIONS 5-1

Introduction 5-2
The IEEE-488 Bus 5-3
Bus Functions 5-4
Interface 5-5
Bus Operating Modes 5-5
Command Mode 5-5
Data Mode 5-7
Three-Wire Handshake 5-7
A Typical Instrumentation System 5-7
Sequence 5-8
Multiple Controller Systems 5-9
IEEE-488 Communications Under Program Control 5-11
Example Commands from the BASIC Language 5-12
Sample BASIC Program 5-13
For More Information 5-15
Serial Communications 5-16
Set Utility Program 5-16
Using the Set Utility Program 5-17
The Help Command 5-18
Command Structure 5-19
Syntax Diagram 5-20
Device Selection 5-21

(continued on page v)

Setting Parameters 5-21
Single Command Line Entry 5-25
Error Messages 5-26
Serial Communications Under Program Control 5-27
Sample BASIC Program 5-28
Conclusion 5-31

6 CREATING AND EDITING PROGRAMS 6-1

Introduction 6-2
Selecting a Programming Language 6-3
BASIC 6-3
FORTRAN 6-4
Assembly Language 6-4
File Utility Program 6-5
Command Line Interpreter 6-5
Introduction 6-5
Editing Features of the Command Line Interpreter 6-6
The Edit Program 6-7
Introduction 6-7
Entering the Editor Program 6-8
Exiting the Editor Program 6-10
Operating Modes 6-11
Global Commands 6-12
Most Used Commands 6-12
Cursor Positioning 6-12
Text Insertion and Deletion 6-13
Substitution 6-13
Marking Text 6-14
Searching 6-14
Command Mode 6-17
Markers 6-17
The Yank Buffer 6-17
Search Commands 6-18
Metacharacters 6-19
Command Mode Commands 6-20
Cursor Positioning 6-20
Long Cursor Movements 6-25
Search Commands 6-27
Marker Commands 6-29
Text Insertion 6-30
Text Substitution 6-31
Case Conversion 6-32

 (continued on page vi)

Text Deletion Commands 6-32
Control Commands 6-35
Target Commands 6-37
Global Commands 6-39
Edit Program Messages 6-47
Conclusion 6-48

7 AUTOMATING SYSTEM FUNCTIONS 7-1

Introduction 7-2
Command Files 7-3
Special Characters 7-4
Sample Command Line 7-5
The Startup Command File 7-6
Linking to Other Command Files 7-7
Establishing the Environment-
The BASIC SET SHELL Statement 7-8
Alias File 7-9
Creating Aliases 7-9
Error Messages 7-11
Standard Aliases 7-11
Automating Utility Programs 7-15
The Time and Date Utility 7-15
Using the Time and Date Clock 7-15
Programming Language Commands 7-16
Set Utility Program 7-17
File Utility Program 7-17
Sample Instrumentation System 7-18
Controlling the Sample System 7-20
Step 1: Start With a Flowchart 7-20
Step 2: Establish Bus Addresses 7-24
Step 3: Program the Modules 7-24
Step 4: Concatenate 7-27
Step 5: Debugging 7-27
Step 6: Document the Program 7-28
Sample Program Listing 7-29
The Startup Command File 7-32
A Caution to Systems Programmers 7-32
Conclusion 7-34

(continued on page vii)

8 DISPLAY 8-1

Introduction 8-2
The Character Plane 8-3
Character Sets 8-3
Custom Character Sets 8-3
Character Graphics 8-4
Programming a Character Graphics Display 8-6
Program to Display Graphics Characters 8-6
Program to Display One Touch-Sense Keypad 8-7
Introduction to ANSI Standards 8-8
Special Display Control Characters 8-9
Escape Sequences 8-10
Numerically Defined Control Sequences 8-11
Selective Parameters 8-14
Field Attributes 8-15
Character Attributes 8-15
Non-Destructive Display Character 8-17
The Graphics Plane 8-18
Introduction to Graphics Routines 8-18
Addressing the Pixel Locations 8-21
Graphics Routines 8-21
Summary of Commands 8-22
DOT 8-24
DRAW 8-25
ERAGRP 8-26
GRPOFF, GRPON 8-27
MOVE 8-28
MOVER 8-29
PAN 8-30
PLOT 8-31
PLOTR 8-32
LABEL 8-33
Printing the Graphics Plane 8-40
Graphics Print Routines 8-41
Summary of Graphics Print Commands 8-41
GPOUT 8-42
GPRINT 8-43
GRBYTE 8-45
Printing the Graphics Plane on Unsupported Printers 8-49
Conclusion 8-53

(continued on page viii)

9 OPTIONS 9-1

Introduction 9-3
Peripherals 9-4
Accessories 9-5
Options 9-6
User Information 9-9
Installing Hardware Options 9-9
-002 Parallel Interface Board 9-13
-004/-005 Magnetic Bubble Memory 9-25
-006/-007 Memory Expansion Modules 9-29
-008 IEEE-488/RS-232-C Interface 9-33
-009 Dual Serial Interface 9-37
-010 Analog Measurement Processor 9-55
-011 Analog Output 9-63
-012 Counter/Totalizer 9-69
1760A and 1761A Disk Drive Systems 9-81
1765A Winchester Disk Drive 9-89
1702A Extender Chassis and Option
-013 Mainframe Interface 9-97

APPENDICES

A Specifications A-1
B RS-232 Reference B-1
C IEEE-488 Reference C-1
D Glossary D-1
E Custom Character Sets E-1
F Primary Character Set F-1
G System Diagnostic Software G-1
H Fluke Sales and Service Centers H-1

INDEX

Section 1
How To Use This Manual

CONTENTS

Introduction 1-3
The Manual Set 1-3
Additional Manuals 1-3
Organization 1-4
Usage Guide 1-6
How to Read Syntax Diagrams 1-8
Sample Syntax Diagrams 1-9
Notation Conventions 1-10

INTRODUCTION

The 1752A Data Acquisition System is supported by a complete manual set. This manual, the 1752A System Guide, is the primary reference source.

The Manual Set

The manual set consists of the following manuals:

- 1752A Getting Started Manual: First-time users should read the Getting Started manual. It is designed to help you set up the 1752A and begin using it.
- 1752A System Guide: The System Guide provides an easy-to-use source of information for a variety of users. Whether this is your first exposure to programmable instrumentation, or whether you already have extensive programming experience, our intention has been to anticipate and meet your needs for accurate, well organized information. All support manuals are referenced here.
- 1752A Data Acquisition and Control Manual: This manual presents complete documentation for the optional measurement and control modules, including the 17XXA-002 Parallel Interface, the 1752A-010 Analog Measurement Processor, the 1752A-011 Analog Output, and the 1752A-012 Counter/Totalizer.
- BASIC: This manual is a comprehensive tutorial for learning Fluke BASIC.
- BASIC Reference: This is a complete reference manual for Fluke BASIC.

Additional Manuals

The following manuals are also available:

- 17XXA-002 Parallel Interface Manual: This manual provides detailed documentation for the 17XXA-002 Parallel Interface (PIB). The manual is shipped with the PIB and is included with the manual set if the PIB is ordered with the 1752A Data Acquisition System.
- 1722A/1752A Service Manual: This manual provides comprehensive service information for the 1752A Data Acquisition System and its options. The manual can be ordered using part number 732156.

ORGANIZATION

This manual is organized into the following sections. These sections represent categories of tasks. The sections are presented in the order that most persons would perform those tasks.

Section 1 How to Use This Manual

Describes the organization of the System Guide and the conventions used in the manual.

Section 2 Setting up the 1752A

Provides a first look at the 1752A and contains unpacking and set-up information. Section 2 illustrates controls, indicators, and connectors, and includes start-up procedures.

Section 3 Software Configuration

Describes the Operating System software and the other programs on a new System disk.

Section 4 Devices and Files

Describes the system's resources and how to use them. This section contains a complete description of the File Utility Program.

Section 5 Communications

Tells how to use the 1752A to send and receive information using the ports for the IEEE-488 bus and the RS-232 interface.

Section 6 Creating and Editing Programs

Explains how to use the Editor program to write and modify programs.

Section 7 Automating System Functions

Explains how to use the various software and hardware resources to automate the functions of the 1752A.

Section 8 Display

Explains how to use the 1752A's graphics capabilities to design detailed and informative displays.

Section 9 Appendices

Contains useful reference material, including a list of options and accessories and a glossary of terms.

If more information about a specific topic is needed, consult the Index.

USAGE GUIDE

Evaluators

If you are evaluating the 1752A for a particular application, the section titled Setting Up the Controller describes the general capabilities and functions of the unit. You might also read the introductions to each of the other tab-divided sections to assess the software packages and hardware configuration. The Specifications are given in Appendix A.

Beginning System Designers

For those with little or no experience in designing a programmable instrumentation system, the Getting Started manual is the best place to begin. This handy, stand-alone volume and accompanying disk will familiarize you with the basic operation and layout of the 1752A. Then you can use the System Guide as a reference for a variety of topics, or consult one of the language manuals (e.g., BASIC). If you need more familiarity with the IEEE-488 bus, see Section 5, Communications, or Appendix C, IEEE-488 Interface References. More information is available in Fluke Application Bulletin AB-36 ("IEEE Standard 488 Digital Interface for Programmable Instrumentation") and Fluke Technical Data Bulletin B0079 ("Communication Over the IEEE-488 Bus").

Programmers

If you already have some experience in programming, the sections on Software Configuration or Automating System Functions are good places to start. Section 4, Devices and Files, can help you become familiar with the file conventions used in the 1752A. You may also wish to refer to the Communications section for information about the IEEE-488 bus and the RS-232 port.

Operators

An Operator's Quick Reference Guide has been included following the appendices in this manual. That Guide shows the location and operation of all controls, care of the floppy disk, and what to do if things don't go as expected. Additional copies can be ordered using Fluke Part Number 760637.

Maintenance Personnel

Appendix G contains information necessary to diagnose and troubleshoot problems at the module level. Using the System Diagnostic Disk, a Service Technician can identify a problem and then contact his local Fluke Technical Service Center for information about how to replace the defective module using the Module Exchange Program. Complete service information is available in the 1752A Service Manual (Fluke Part Number 762567). For parts ordering information call the parts department at 1-800-526-4731 (U.S.A. only).

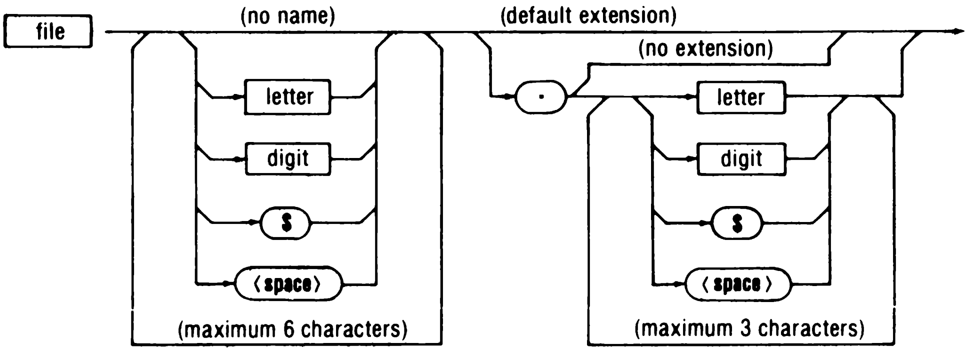
HOW TO READ SYNTAX DIAGRAMS

A syntax diagram is a graphical representation of how to construct a valid command or statement in a programming language. It is a kind of "shorthand" way of writing down all the rules for using the elements of a language. Since they are used throughout this manual, learning how to read them can be a great time saver.

WORD	Words inside ovals must be entered exactly as they are shown.
RETURN	Words inside boxes with rounded corners indicate a single key must be pressed, such as RETURN or ESC.
⟨space⟩	This indicates a space in the statement. (Press the spacebar.)
⟨CTRL⟩/C	To create a control character, hold down the control key (CTRL), then press the other key. This one is a Control C; it causes a break in the program.
filename	A box with lower-case words inside means that you supply some information. In this case, you would enter a filename.
(explanation)	Words in parentheses are explanations of some kind. They give added information about the nearest block or path.

Sample Syntax Diagram

From the left, any path that goes in the direction of the arrows is a legitimate sequence for the parts of a statement. This sample shows the correct syntax for naming a file. The translation is given below.



- A line exits the top of this diagram with no keyboard input. This indicates that it is possible to not specify the filename or its extension. In this case, the file would have "no name", and the system would assign a "default extension".

- Further down the diagram, you can see that there are other possibilities. They are explained by the remarks, "maximum of 6 characters" for the name, and "maximum of 3 characters" for the extension.

- The filename can be any combination of letters, digits, the $ sign, and spaces (up to six characters), and the extension can be up to three of those characters.

- The filename and extension must be separated by a period, as shown in the oval block at the top center.

- The remark "no extension" means that it is not necessary to specify an extension, even though a file name is given. Notice however, that this remark occurs after the period, so the period is necessary if a name is specified.

- Here are some examples of valid filenames according to the syntax illustrated in the diagram:

  TESTIN.$3A   1752A.RAC $$$$$$.$$$

NOTATION CONVENTIONS

The conventions listed here are used for illustrating keyboard entries and to differentiate them from surrounding text. The braces, { } ; brackets, [ ]; and angle brackets, ⟨ ⟩ ; are not part of the keystroke sequence, but are used to separate parts of the sequence. Do not type these symbols.

⟨xxx⟩	Means "press the xxx key". Example: ⟨RETURN⟩ indicates the RETURN key.
⟨xxx⟩/y	Means "hold down key xxx and then press y". Example: ⟨CTRL⟩/C means to hold down the key labeled CTRL and then press the key labeled C.
[xxx]	Indicates an optional input. Example: [input filename] means to type the name of the input filename if desired. If not, no entry is required, and a default name will be used.
xxx	Means to type the name of the input as shown. Example: BASIC means to type the program name BASIC as shown.
{xxx}	Indicates a required user-defined input. Example: {device} means to type a device name of your choice, as in MF0: for floppy disk drive 0.
(xxx)	This construction has two uses: 1. As a separate word, (xxx) means that xxx is printed by the program. Example: (date) means that the program prints today's date at this point. 2. Attached to a procedure or function name, (xxx) means that xxx is a required input of your choice; the parentheses are a required part of the input. Example: TIME(parameter) means that a procedure specification is the literal name TIME followed by a parameter that must be enclosed in parentheses.

Section 2
Setting up the Data Acquisition System

CONTENTS

The 1752A Data Acquisition System 2-3
Fluke Enhanced BASIC 2-4
Floppy Disk Operating System 2-4
Data Acquisition and Control Software 2-5
Features 2-6
Measurement and Control Modules 2-8
1752A Configurations 2-9
Option Configuration 2-9
Physical Layout 2-10
Unpacking 2-12
Installation 2-14
Environment 2-14
Pre-Installation Checkout 2-15
Front Panel Controls 2-20
Default Switch Settings 2-21
Configuring the Measurement and Control Modules 2-21
Placement 2-22
Workbench Installation 2-22
Rack Mounting 2-24
Installing the Data Acquisition and Control System 2-26
Installing an IEEE-488 System 2-28
Connecting the IEEE-488 Bus 2-29
Conclusion 2-30

THE 1752A DATA ACQUISITION SYSTEM

The Fluke 1752A Data Acquisition System is a microcomputer-based system for use in analog and digital measurement and control applications. The 1752A, like the Fluke 1722A, also functions as a powerful instrument controller designed to support IEEE-488 instrumentation systems.

The standard 1752A comes with one 1752A-010 Analog Measurement Processor. Through the use of optional additional plug-in modules, the system can measure or output dc currents and voltages; measure time and frequency of analog and TTL signals; count events with a bi-directional totalizer; and interface with digital signals. The optional modules available to provide these functions are:

- 1752A-010 Analog Measurement Processor
- 1752A-011 Analog Output
- 1752A-012 Counter/Totalizer
- 17XXA-002 Parallel Interface

The 1752A also features a touch-sensitive display, which can replace the keyboard for command entry. This feature allows semi-skilled or non-programming personnel to control complex tasks with ease. The operator can respond to screen prompts one step at a time, making inputs by simply touching the screen.

Fluke Enhanced BASIC

The standard 1752A includes Fluke Enhanced BASIC, which contains a variety of single-word commands for measurement and control. These commands simplify the task of writing programs for both data acquisition and IEEE-488 compatible instrument control. FORTRAN and Assembly Language modules may be linked to any BASIC program for intensive data analysis. A text editor is included with the 1752A for creating and editing programs.

There are three forms of the BASIC language available to suit your software development needs for data acquisition and control:

- Interpreted BASIC, for quick start up.
- Compiled BASIC, for applications requiring advanced structure and higher execution speed.
- Extended BASIC, for very large, sophisticated programs.

Interpreted BASIC is included with the 1752A. Compiled BASIC and Extended BASIC are available as options.

Floppy Disk Operating System

The 1752A's Floppy Disk Operating System (FDOS) combines the best features of both bench-top computers and minicomputers. And, because the 1752A is a soft-loaded system, you can load new versions and languages by simply inserting a disk. Like a bench-top unit, the user can power-up the 1752A and immediately begin programming in BASIC.

The task of developing application programs is simplified by advanced operating system features such as the alias file, command files, system shell command, an advanced editor, file utilities, the powerful File Utility Program, and TCOPY. For example, you can copy an entire disk by making a couple of touch selections displayed on the screen. You can also set up command files to automate repetitive tasks, such as multiple module compilation, or partial disk duplication. A system command file can define a start-up sequence for turnkey operation.

Data Acquisition and Control Software

The flexible measurement and control capabilities of the 1752A are available to the programmer through a library of measurement and control subroutines provided with the 1752A. The 1752A Data Acquisition and Control manual provides detailed programming information for these subroutines.

Since these subroutines are directly accessible from the BASIC program, data acquisition and control tasks are greatly simplified. Subroutines contain the minimum number of parameters required to perform a basic data acquisition function. Subroutine names easily identify the module they address and the function they perform. All subroutines report errors in a standard BASIC format, allowing the programmer to develop specialized error handling routines. The features make programs easier to read and more self-documenting.

An analog measurement configuration program is also provided on the Getting Started Disk that allows the user, through the touch-sensitive display, to program each Analog Measurement Processor and each of its channels. The program also monitors any single channel on a bar graph, or displays all the data on all channels in a table.

FEATURES

Standard features of the 1752A are:

- Analog Measurement Processor with 32 single-ended or 16 differential analog input channels
- Analog-to-digital conversion rates up to 1,000 per second
- Measurement accuracy of 0.02% + 1 count
- High-performance processor with Macrostore (TM) high-speed floating point processing
- 400K-byte floppy disk drive
- 136K-bytes of read/write memory
- E-Disk (TM) with memory allocatable in variable sizes
- Fluke Enhanced BASIC Interpreter
- Detachable programmer's keyboard
- Rack-mountable industry-standard packaging
- Touch-sensitive, high-resolution graphics display
- Real-time non-volatile calendar and clock
- IEEE-488 (1980) Standard controller interface port
- RS-232-C interface port
- Four expansion slots for options (see below)

Options and accessories include:

- Analog Measurement Processor, maximum three additional per system
- Line frequency sync transformer to increase normal mode rejection to 50 dB (at 50 to 60 Hz)

- 32-bit Parallel Interface module (a maximum of three per system)
- Four-channel Analog Output module (a maximum of two per system)
- Counter/Totalizer input module (a maximum of two per system)
- Memory expansion to over 2.0M bytes
- Non-volatile bubble memory storage up to 1.3M bytes
- Compiled BASIC software development system
- Extended BASIC software development system
- Extended FORTRAN IV subroutine development system
- Assembly Language subroutine development system with structured preprocessor and extended floating-point instructions
- IEEE-488 (1980) and RS-232-C Interface modules
- High performance Dual Serial Interface modules (a maximum of three per system) configurable as RS-232-C, RS-422, or current loop
- Dual 400K-byte (800K-byte total) external disk drive system
- 10 M-byte Winchester Hard Disk Drive System

MEASUREMENT AND CONTROL MODULES

The 1752A features a selection of optional measurement and control modules. Each is briefly described below. For more information refer to Section 9 under the appropriate option heading, or refer directly to the 1752A Data Acquisition and Control manual.

- Analog Measurement Processor (Option 1752A-010)

  The Analog Measurement Processor converts voltages or currents to digital readings via 32 individually addressable input channels. The input channels may be configured as 32 single-ended channels, 16 differential input channels, or combinations of single and differential. Measurements are made in two voltage ranges (1V and 10V) and two current ranges (65 mA and 4 to 20 mA).

- Analog Output (Option 1752A-011)

  The Analog Output efficiently provides user-configured voltage or current outputs to external control points. The outputs are individually isolated. Output voltage is programmable over -10.23V to +10.23V range. Output current is programmable over a 0 to 20.47 mA range.

- Counter/Totalizer (Option 1752A-012)

  The Counter/Totalizer performs frequency, time, and totalizing measurements. Frequency measurements are performed on TTL or analog inputs at four different programmable gate times. Time measurements are performed on TTL, analog, or gate inputs. Intervals and periods can be measured by programming the slope of the signal that starts and stops measurement.

- Parallel Interface (Option 17XXA-002)

  The Parallel Interface is a versatile option that provides 32 lines for bidirectional data transmission between the 1752A and compatible external devices. The Parallel Interface may be programmed for either parallel data transfers or individual bit input and output.

1752A CONFIGURATIONS

The standard 1752A includes one Analog Measurement Processor. The 1752A is also available without the Analog Measurement Processor, as Model 1752A-1. Model 1752A-1 is appropriate for users that do not intend to measure voltage or current.

OPTION CONFIGURATION

The following table lists the slots available for the standard and optional plug-in modules. The items in the table are included with the 1752A if they are ordered at the same time as the 1752A. In this table, take particular notice of the constraints with module insertion; these are indicated by either a † or an *.

OPTIONS

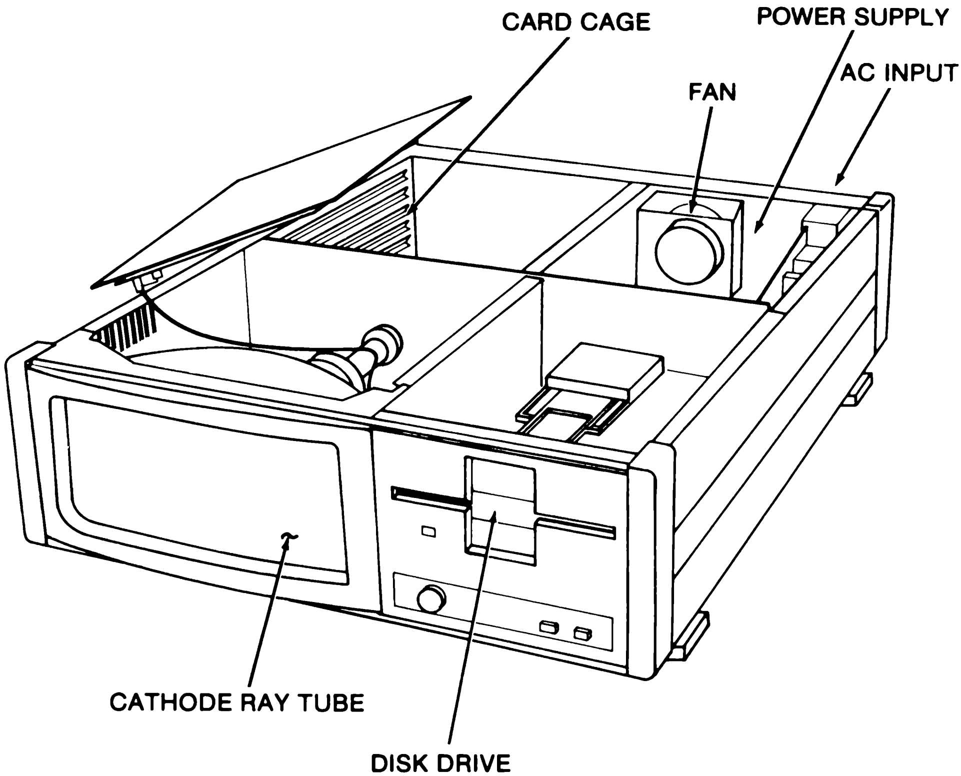
Counter/Totalizer -012	Analog Output -011	Analog Measurement Processor -010	Dual Serial Interface -009	IEEE-488/RS-232C Interface -008	512K byte RAM Expansion -007	256K byte RAM Expansion -006	512K byte Bubble Memory -004	256K byte Bubble Memory -003	Parallel Interface -002	SLOTS
•	•	•	•	•	•	•	•	•	•	1
Reserved for Video/Graphics/Keyboard Interface										2
		•	•		•	•	•	•	•	3
• *	• *	•	•	• †	• †	•	•	•	•	4
• *	• *	•	•	• †	• †	•	•	•	•	5
						•	•	•		6
Reserved for Single Board Computer										7

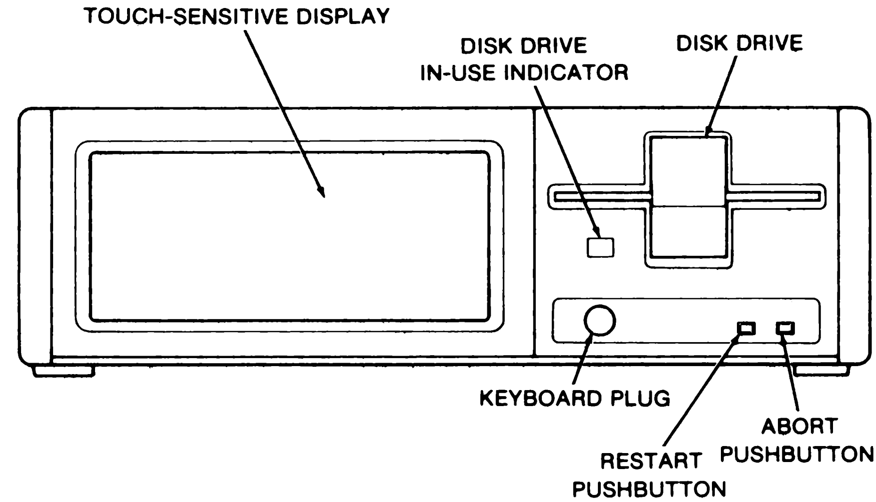
- • Allowable Slot for Option
- † Non I/O must be used in slot above
- * Takes up two slots. No board in slot above

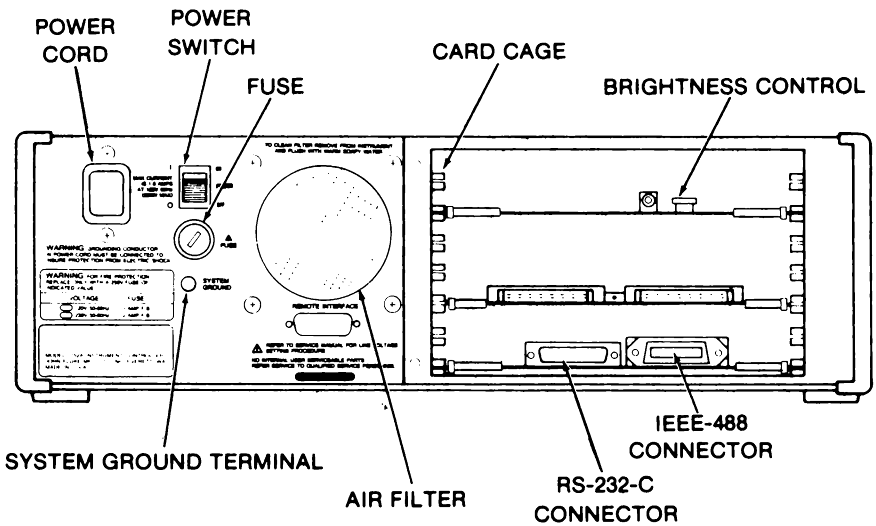
Note:

Analog Measurement Processor is shipped in slot 5. Normally it may be used in any of the Input/Output Options slots. Slot 6 has no Input/Output access.

PHYSICAL LAYOUT







Slots	
Single Board Computer	7
Video/Graphics/Keyboard Interface	2
Analog Measurement Processor	5
Options	1, 3, 4, 6
Memory Expansion Options	1, 3, 4, 6
Measurement and Control Options	1, 3, 4
Input/Output Options	1, 3, 4

UNPACKING

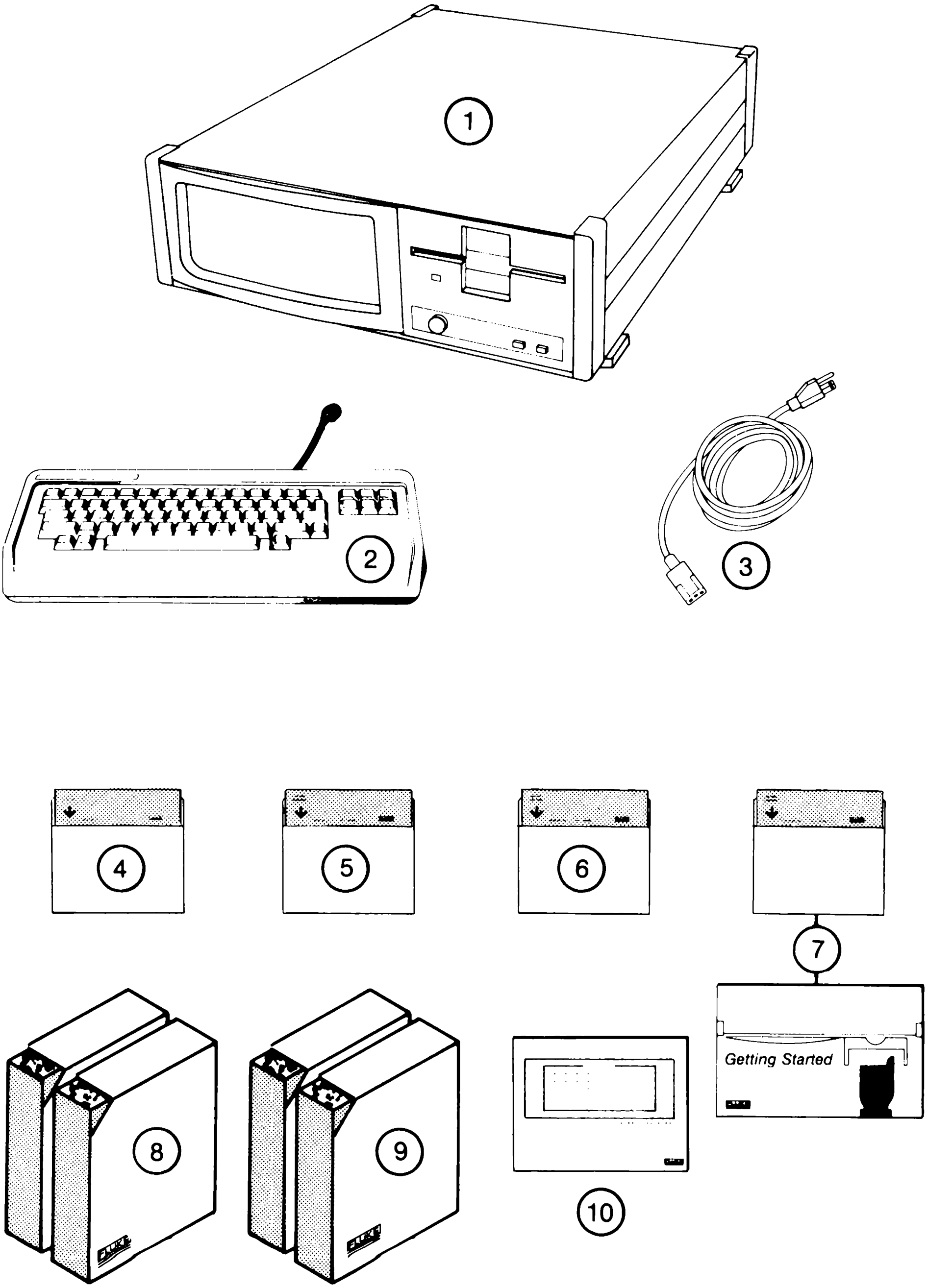
The 1752A is carefully packed for shipping to ensure that it arrives in good condition. Unpack all the containers and check the packing materials for accessories, cables, and manuals. Do not dispose of the packing materials before inspecting them for shipping damage. If this inspection reveals possible damage, notify the shipping agent immediately. Then call a Fluke Sales Office or Customer Service Center (see Appendix H). Use this checklist to be sure the shipment is complete:

Unpacking Checklist

1. System Unit Mainframe
2. Keyboard
3. Power Cord
4. System Disk
5. Data Disk
6. Diagnostic Disk
7. Getting Started Disk and Manual
8. System Guide and Data Acquisition and Control Manual
9. BASIC Programming Manual Set
10. Programming Worksheets (pad of 50)

Other options or accessories may be included with the shipment. Check the contents against your original order to ensure that all items have arrived. A list of options and accessories is given in Appendix B.

If the 1752A needs to be shipped again at a later date, use the original packing carton with all fillers properly in place. Fluke does not recommend shipping the 1752A in a substitute container. To obtain an approved container, call any Fluke Sales Office or Service Center.


1
2
3
4
5
6
7
Getting Started
8
9
10

INSTALLATION

Any microprocessor-based equipment is made up of two parts: hardware and software. Installation usually involves a physical installation and some sort of software configuration. This section describes the physical installation of the 1752A. Software configuration is covered in Section 3.

Before using this section to install and check out the 1752A, be sure you are familiar with the location of all the connectors and controls. This will assist you in setting up your system.

Environment

It is important to ensure that the location for the 1752A meets the environmental requirements listed in the Specifications. Heat and humidity are two of the worst enemies of electronic equipment, particularly the floppy disk and its drive. Allow at least 10 cm (4 inches) between the back of the unit and the wall to allow sufficient air flow for the fan to cool the unit adequately.

Floppy disks are more sensitive to storage environments than the 1752A. If a disk becomes colder than 10°C (50°F), or warmer than 50°C (122°F), allow it to reach room temperature and humidity before placing it in the 1752A. The checkout instructions below include other disk handling precautions.

CAUTION

Low-humidity environments can contribute to static build-up. Static discharges can permanently damage circuitry within the 1752A and erase information recorded on the floppy disk. To prevent such damage, always make sure that the humidity is above the minimum specified level and that the instrument is properly grounded.

Pre-Installation Checkout

1. Place the 1752A on a suitable table. Check the label on the rear panel to ensure that the unit is set up for the proper line voltage, and that the proper fuse is in place. If either the line voltage or fuse are incorrect, ask your Fluke Technical Service Center for assistance.

2. Check to see that any ordered options are installed. If "installed" was specified on your order, the options should already be in the 1752A. Otherwise they will be packed separately.

3. Install any options that were not specified to be factory-installed. Refer to the 1752A Data Acquisition and Control manual or the manual shipped with the option for installation instructions.

4. Release the door latch on the floppy disk drive by pressing in on the top. Remove the protective shipping insert from the disk drive.

CAUTION

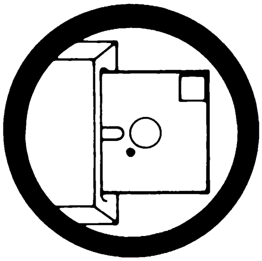
Do not attempt to operate the 1752A before removing the protective shipping insert. Doing so can damage the disk drive.

5. Attach the keyboard and line cord. Plug in the line cord and turn on the power switch (rear panel).

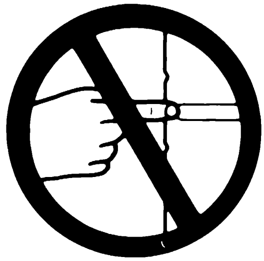
6. Following the disk handling precautions on the next page, gently insert the System Disk (label up) into the disk drive.

NOTE

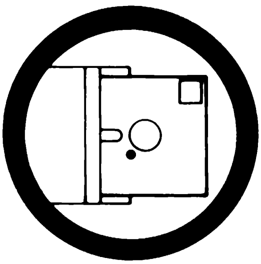
Floppy disks supplied by Fluke use reinforced center rings to help seat the disk on the spindle. If floppy disks without such rings are used, first insert the disk, and gently close the door without latching it. Reopen the door slightly, then close and latch it. This ensures that the disk seats on the spindle, and improves the reliability of reading and recording data.



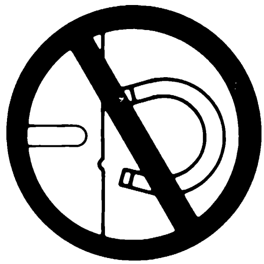
Insert Carefully
Insertar
Inserer avec soin
Sorgfaltig Einsetzen
挿入注意



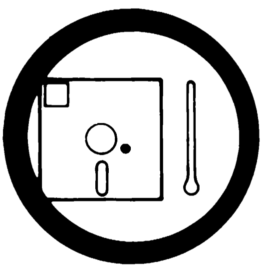
Never
Nunca
Jamais
Nie
絶対禁止



Protect
Proteger
Proteger
Schutzen
保護



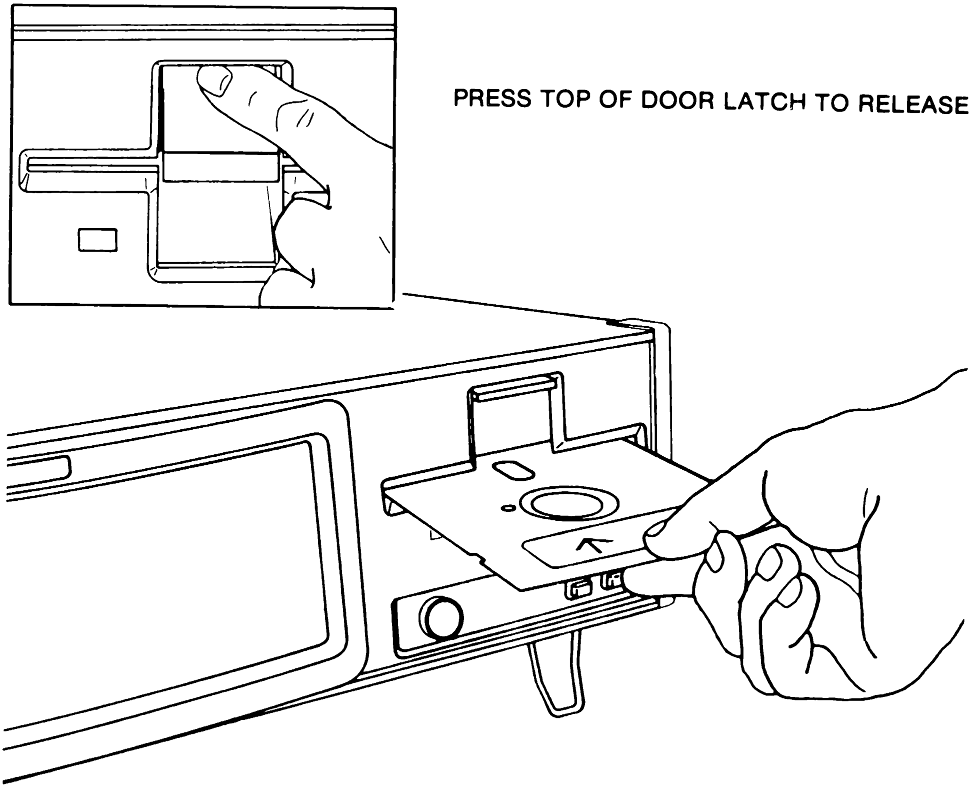
Never
Nunca
Jamais
Nie
絶対禁止



10°C - 50°C
50°F - 122°F

No
No
Non
Falsch
注意

7. When the disk is fully seated, close the drive by pressing in on the bottom of the latch.



8. Press RESTART and watch the display. It should read:

```
FLUKE 1752A DATA ACQUISITION SYSTEM
HELLO
BOOT Vn.n  _
```

9. Following this display, the 1752A performs a self-test that checks out the internal circuitry. The display changes to read:

```
FLUKE 1752A DATA ACQUISITION SYSTEM
SELF TEST IN PROGRESS  _
```

10. If the self-test is not successful, an error message is displayed on the screen. If this happens, recheck the previous steps. For a continuing failure, see the list of Self Test error messages in Section 3. If everything seems to be in order, perform the System Diagnostics as discussed in Appendix G. If further assistance is required, contact your Fluke Service Center.

11. After the self-test is successfully completed, the 1752A loads the Operating System software into main memory from the System Disk. The display again changes to read:

```
FLUKE 1752A DATA ACQUISITION SYSTEM
LOADING _
```

12. When the Operating System has been loaded from the floppy disk into memory, the system asks for you to set the correct time and date. When you have done this, the prompt for the Fluke Disk Operating System is displayed:

```
FDOS Version x.y
Total System Memory - nnn Kbytes
E-Disk - n Kbytes, free - nnn Kbytes  (nn blocks)
FDOS >
```

13. If all these steps have been successful, the 1752A is fully functional, and the Pre-Installation Checkout is complete. There is a more complete description of the system's power up activities in the next section, Software Configuration.

NOTE

Altered system software may not ask for the time and date, nor show the FDOS prompt. The programmer can tell the installer how the successfully loaded software will appear.

Front Panel Controls

Refer to the figure on page 2-11 for the following discussion.

RESTART reloads the system software and restarts the user application program (if a STRTUP.CMD file is present), without disturbing any programs or data on Electronic Disk.

ABORT interrupts the current user application program. This function is programmable, and the action taken is dependent on the application program. If the application program does not specifically handle this situation, the program is stopped.

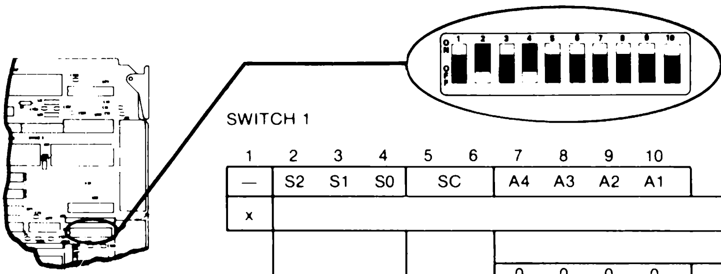
Pressing the RESTART and ABORT keys simultaneously is the same as turning the power off, then on again. The system software and user application program are automatically reloaded from the disk.

CAUTION

The RESTART and ABORT keys only affect programs running on the 1752A. Pressing these keys will not have any effect on the status of other equipment or instrumentation connected to the 1752A. If required, the user program must take specific action to place other equipment in the system in standby mode. For more information, see Section 7, A Caution to Systems Programmers.

Default Switch Settings

The Single Board Computer module in the 1752A has a set of switches that determine the default system configuration on power-up. These switches define the baud rate of the RS-232-C port, the IEEE-488 device address, and whether the 1752A is to be designated "system controller" or an "idle controller" in an IEEE-488 instrumentation system with multiple controllers. The drawing below illustrates the switch settings as configured at the factory. If you need to alter the switch settings, consult the information in Section 9, Installing Hardware Options, for instructions on removing the Single Board Computer. Refer to Section 5, Communications, for complete information on RS-232-C communications and multiple controller systems.



SINGLE BOARD COMPUTER MODULE

SWITCH 1

1	2	3	4	5	6	7	8	9	10	
—	S2	S1	S0	SC		A4	A3	A2	A1	
x										unused
										IEEE-488 ADDRESS
						0	0	0	0	0
						0	0	0	1	1
						"				
						"				
						"				
						1	1	1	1	15
										IEEE-488 CONTROLLER
				0	0					System Controller
				1	1					Idle Controller
										BAUD RATE
	0	0	0							110
	0	0	1							300
	0	1	0							600
	0	1	1							1200
	1	0	0							2400
	1	0	1							4800
	1	1	0							9600
	1	1	1							19200

NOTE: 1 = on; 0 = off

Configuring the Measurement and Control Modules

See the 1752A Data Acquisition and Control manual for detailed instructions for installing and configuring the measurement and control modules.

PLACEMENT

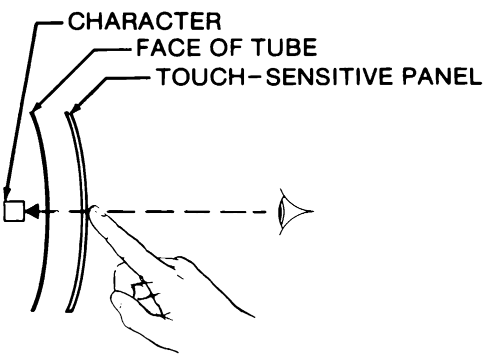
The 1752A can either be rack-mounted or used on a bench. Rack mounting is preferred for more permanent installations or when the system will be set up in an assembly-line application. In research and development facilities, scientific or engineering laboratories, or other locations where the 1752A will be portable, mount the system on a rack.

Wherever the installation site is, choose a location where the display will not be subject to glare from overhead lights or windows.

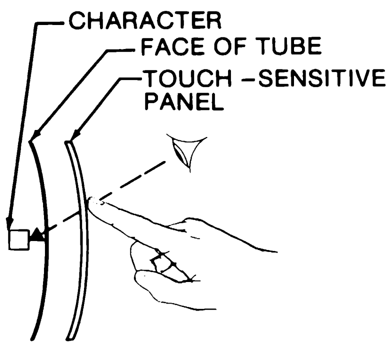
Workbench Installation

Normally, little thought is given to planning how to set up a test fixture on a workbench; wherever things fit is usually where they are put. However, a little planning can make the installation much more versatile, efficient, and pleasant to work with.

1. Find a location for the 1752A where the touch-sensitive display will be as close to eye level as possible. This prevents long reaches for the operator that can become tiring, and ensures that a parallax error will not cause the operator to touch the screen at the wrong location.

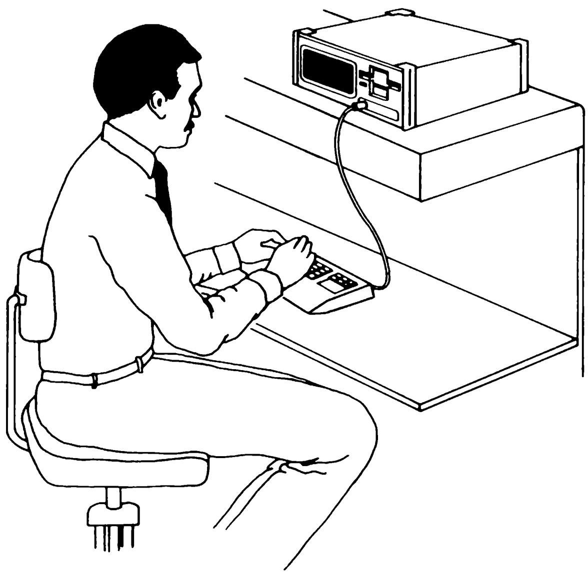


RIGHT



WRONG

2. Position the keyboard so that the programmer is able to sit directly in front of the screen, rather than off to one side.



3. If instruments will be repeatedly connected and disconnected, make sure the back of the workbench is open and not against a wall. This makes connectors on the rear panel more easily accessible.

4. Arrange other instruments on the bench so they are easy to see and operate.

5. To protect information on the floppy disks, never place an oscilloscope, soldering iron, or other sources of high voltage or electromagnetic fields near the disk or disk drives.

6. Keep the equipment cables neatly organized. Dressing the cables ensures that long cables will not degrade signal quality among the instruments, keeps the work area neat, and helps trace any problems that might occur.

Rack Mounting

Installing the 1752A in an equipment rack requires more planning than bench-top installation, because it is more difficult to change if things don't work out. Here are a few things to keep in mind as you plan a rack-mounted installation:

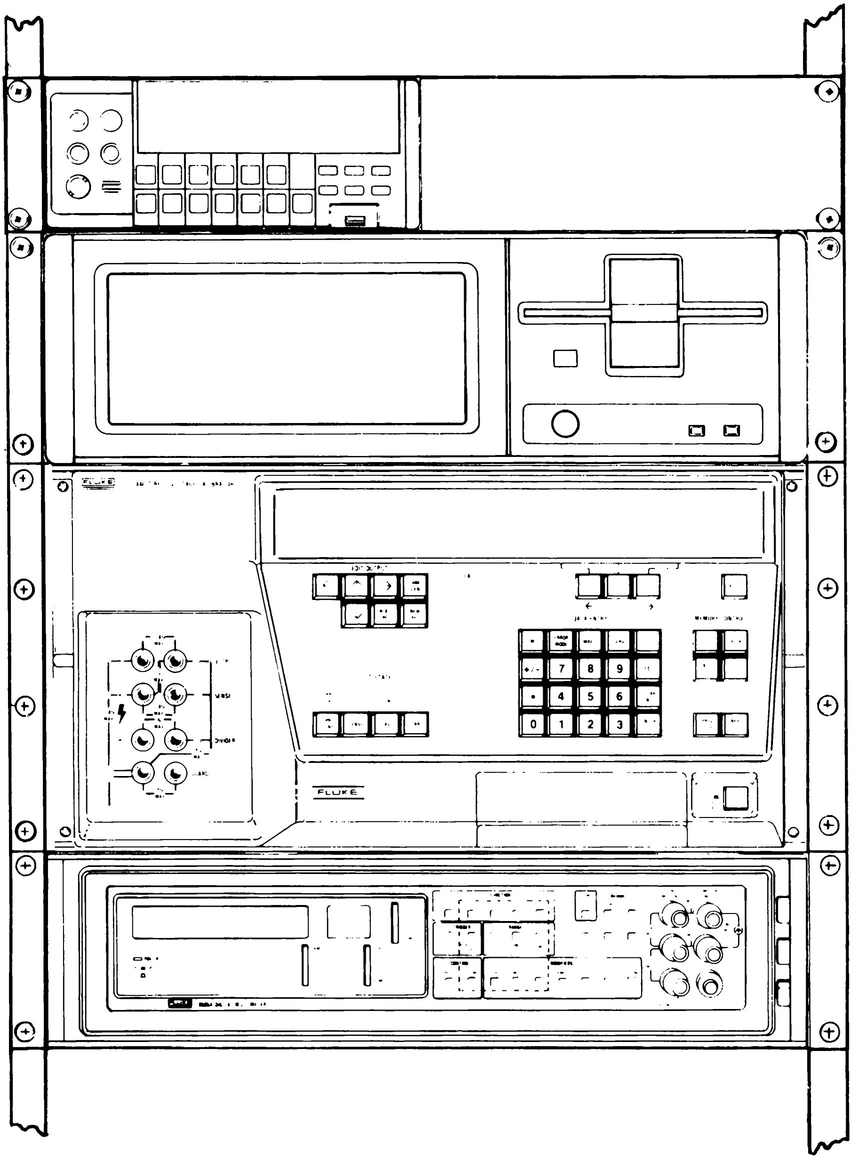
1. Plan for a way that the keyboard can be easily used after the installation. Even though the keyboard will not be attached during normal operation, it may be needed later to change programs. If the 1752A can be left in the rack, you (or the operator) can save time and effort.

2. If the programs will have a high degree of interaction between the operator and the touch-sensitive display, consider mounting the 1752A so that it will be at eye level from the operator's normal working position. If the operator stands, the 1752A should be mounted 1.5 meters (5 feet) above the floor. If the operator sits, then it can be mounted lower. This will avoid long reaches and parallax errors.

3. If programs will not require much interaction, place the instrument with which the operator will be working most frequently directly at eye level, and fill the rack outward from there with those instruments used less frequently.

4. To protect the information on the floppy disks, position any instruments that radiate electromagnetic fields as far away as practical from the 1752A.

5. Dress the cables well to ensure good signal quality and to help trace any problems that might occur.



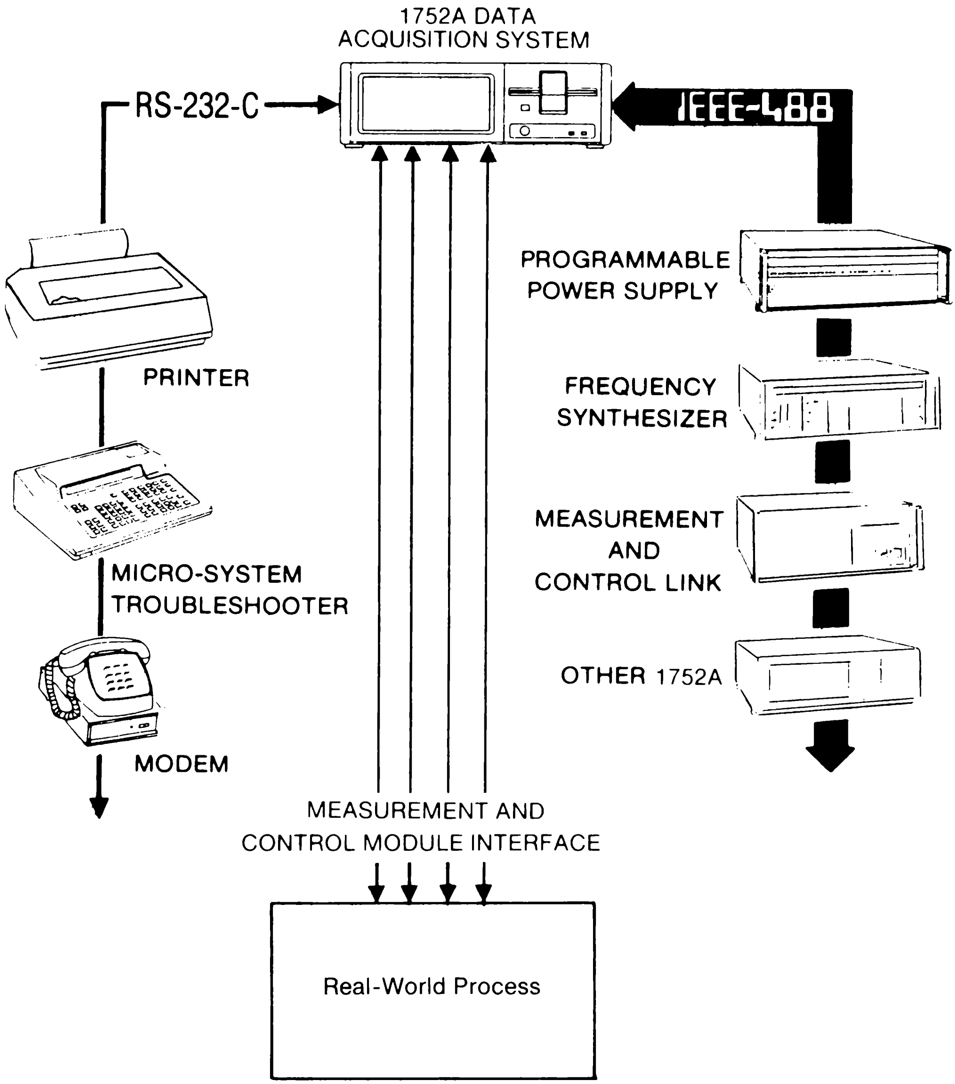
INSTALLING THE DATA ACQUISITION AND CONTROL SYSTEM

The 1752A has three types of interfaces: the optional measurement and control modules, the IEEE-488 interface, and the RS-232-C interface.

The optional measurement and control modules provide a variety of interfaces to real-world measurement and process control. For complete installation and programming instructions, refer to the 1752A Data Acquisition and Control manual.

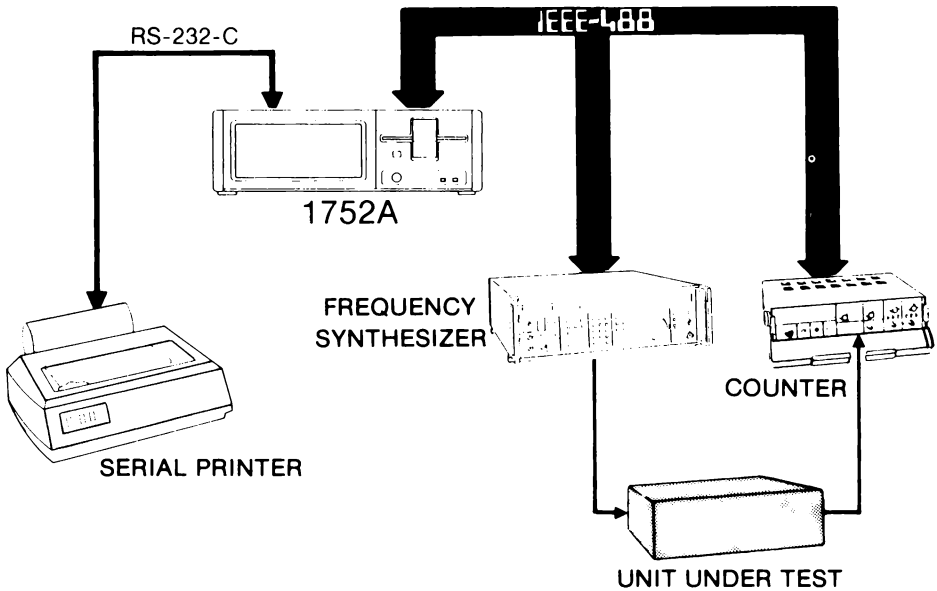
The IEEE-488 interface allows the 1752A to function as a powerful Instrument Controller in an IEEE-488 system. Installation instructions are provided below.

The RS-232-C interface lets you connect the 1752A to devices that use the Electronic Industry Association's RS-232-C Data Communications Interface Standard. Installation instructions are presented in Section 9, Options 17XXA-008 and 17XXA-009.


1752A DATA
ACQUISITION SYSTEM
RS-232-C
IEEE-488
PRINTER
MICRO-SYSTEM
TROUBLESHOOTER
MODEM
PROGRAMMABLE
POWER SUPPLY
FREQUENCY
SYNTHESIZER
MEASUREMENT
AND
CONTROL LINK
OTHER 1752A
MEASUREMENT AND
CONTROL MODULE INTERFACE
Real-World Process

INSTALLING AN IEEE-488 SYSTEM

In addition to the 1752A's data acquisition capabilities, the 1752A can also function as a powerful instrument controller in an IEEE-488 instrumentation system. Because of the versatility of the 1752A, it is not possible to give more than general guidelines on how to configure it into a system. The drawing below illustrates an example system.



Connecting the IEEE-488 Bus

One of the features of the IEEE-488 connector is that it has both male and female connectors. Because of this, it is possible to stack all of the connections (up to the maximum of 14) into one location, or to arrange them in any other configuration desired. However, it is probably a better idea to distribute the IEEE-488 connectors among the instruments for two reasons:

1. In such an arrangement, the connectors do not extend as far, and connector stress that could cause intermittent problems later is eliminated.

2. Using a distributed connector pattern, it is easier to change any connector's position than if the bus connections were all at the same place.

The IEEE-488 standard states that total cable length in a system should not exceed 20 meters (about 60 feet), and that no single cable should exceed 4 meters (about 12 feet). Allow 2 meters (6 feet) of cable per piece of equipment.

CONCLUSION

This section has provided the necessary information for identifying the features and getting familiar with the 1752A. Emphasis has been on introducing the 1752A into its working environment and describing briefly the capabilities of the unit. To complete your system, you must still configure the system software and write test programs to verify hardware installation.

The next section takes you step by step through the necessary procedures that define the basic operating parameters of the 1752A. Refer to the 1752A Data Acquisition and Control manual for complete information about installing and operating the optional measurement and control modules.

System Diagnostic information is included in Appendix G of this manual, and complete service information is available in the 1752A Service Manual (Fluke Part Number 762567).

Section 3
Software Configuration

CONTENTS

Introduction 3-2
Loading the System Software 3-3
  Bootstrap Loader 3-3
  Self-Test Error Messages 3-4
  Other Errors 3-5
  The Startup Command File 3-6
  Setting the Time 3-7
  The Operating System 3-9
  Command Line Interpreter 3-10
  Other Software on the System Disk 3-10
Making a New Operating System 3-12
  Introduction 3-12
  A Note About Software Compatibility 3-12
  Using the System Generation Utility 3-13
The FDOS Configuration Data Program 3-15
  Entering CONFIG 3-15
  Writing Configuration Data to a File 3-16
  CONFIG Program Help Information 3-16
  Printing the CONFIG Program Version Number 3-17
The Compatibility Test Program 3-18
  Running the Compatibility Program 3-18
  Program Help Information 3-19
  Printing More Data From Image Files 3-19
  Directing Output to a File 3-21
Using OTP 3-21
Conclusion 3-22

INTRODUCTION

The last section discussed how to set up the 1752A Data Acquisition System from a physical point of view. This section describes the software, which, together with the hardware, make up a functional 1752A. The section includes a description of the software that is shipped with each new 1752A.

LOADING THE SYSTEM SOFTWARE

Bootstrap Loader

When the 1752A is powered up, a small program that is permanently recorded in memory has control of the internal microprocessor. This program, the Bootstrap Loader, first says "HELLO", then performs two very important functions:

- It checks out the 1752A with a self-test to make sure all the hardware is operating properly.
- It loads the system software.

The self-test checks the memory, processor, and the interfaces.

```
FLUKE 1752A DATA ACQUISITION SYSTEM
     SELF TEST IN PROGRESS _
```

The benefit of this automatic test at power-up is that if it is successful, the 1752A's hardware is verified as operational, and if a problem occurs later, the hardware can be eliminated as the fault. If the test fails, press RESTART and try again. If the problem persists, contact your Fluke Customer Service Center.

Self-Test Error Messages

If an error occurs during the self test, a message will be displayed that takes the form:

```
FAILED: - xxx Test.
```

The xxx will indicate the test that failed, which may be any of the following.

- ROM Test
- RS-232 Port Test
- Memory Mapper Test
- Macrostore Memory Test
- On-board Memory Test
- IEEE Controller Test
- Floppy Disk Controller Test

All of these messages indicate a non-recoverable hardware failure. Try resetting the 1752A first, but if the error continues, make a note of the test that failed, and contact your Fluke Service Representative.

After the Self test, the Bootstrap loader attempts to load the operating system.

The Bootstrap loader searches for the system software on the floppy disk, Electronic disk, then in the bubble memory if one is installed. If the system software is found, it is loaded into main memory. The device from which the system software was loaded is made the system device. (For more information about the system device, see Section 4, Devices and Files.)

As soon as the operating system is loaded, it takes over from the Bootstrap Loader, and the instructions recorded on the software disk direct the controller's activities from that point.

Other Errors

Some errors can occur during the power-up self-test, or any time during the operation of the 1752A. The errors are always preceded by a question mark indicating that they are not recoverable; the 1752A continues to return the same error unless you take some corrective action. These errors and the corrective action required are:

Message	Meaning
`?Disk Not Ready`	Insert or reinsert the System disk. Either there is no disk in the drive, or it has been inserted incorrectly. Make sure the disk drive door is latched.
`?Illegal Directory`	The disk is faulty and must be replaced before the 1752A will operate properly. It may be possible to save the files from the bad disk by using the File Utility Program.
`?Device Error`	The system is having difficulty reading the floppy disk. Check to be sure it is a System disk, and that it is inserted properly. If so, RESTART and try again. If the failure continues, try another System disk.
`?No System On Device`	The 1752A does not recognize the disk in the drive as a System Disk. Try another System Disk. The wrong disk may be inserted, or the disk may be inserted incorrectly.

The Startup Command File

On a standard System disk, the operating system loads a special command file called STRTUP.CMD. A command file is a collection of keyboard commands that would otherwise have to be typed in. Command files serve to automate commonly performed functions. For more details, see section 4.

The STRTUP.CMD file on a standard disk loads two more programs: The Time and Date Utility and the BASIC Interpreter program.

The STRTUP.CMD file is easily changed. After gaining some familiarity with the 1752A, you may want to modify the startup file to customize the 1752A's functions at power-up. There is a complete description of how to do that in Section 7, Automating System Functions.

- If the file STRTUP.CMD is found, it is loaded, and the FDOS prompt will not have been displayed. Instead, the display reads:

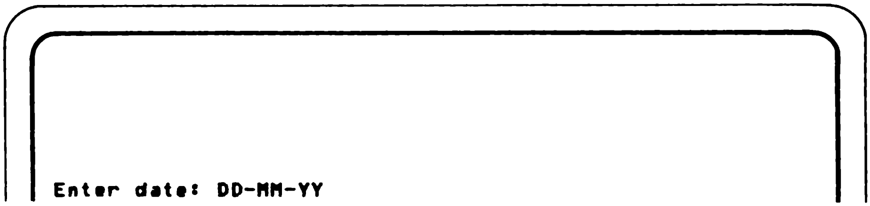
```
FDOS Version x.y
Total System Memory - nnn Kbytes
E-Disk - n Kbytes  free - nn Kbytes  (nn blocks)

Startup Command File Execution in Progress

Please Standby...
```

- If the file STRTUP.CMD is not found, the Operating System takes control. This would happen if the disk being loaded is not the System Disk supplied with the 1752A, or if the STRTUP.CMD file has been renamed or deleted.

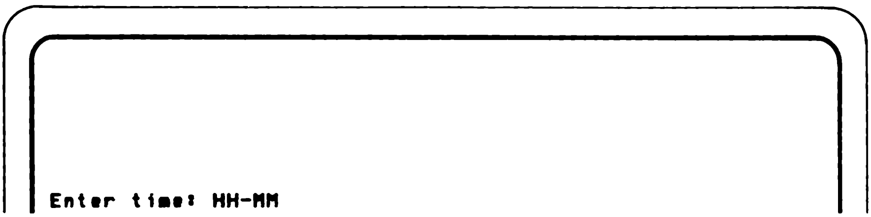
- The STRTUP.CMD file on the System Disk supplied with the 1752A first checks the Time and Date Utility to see if the time has been set. If it has, it loads the BASIC Interpreter program and transfers control to it.

Setting the Time

- If the time clock has not previously been set, the display will next read:

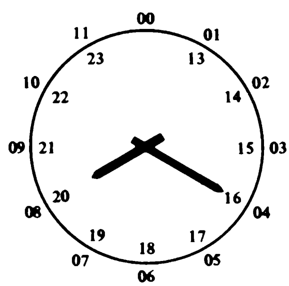


- Type in today's date in numeric form, starting with the day, then the month, and then the year. The entries must be separated by a hyphen or other non-numeric character. Use the DELETE key to correct any mistakes. Press ⟨RETURN⟩ and the display reads:



- Enter the time in 24-hour format: first the hour, then the minutes. Separate the two by any non-numeric character. Press ⟨RETURN⟩ to complete the operation.

The time on this clock is 8:20.
If it is before noon, enter 08 20;
if it is evening, enter 20 20.



- Once the date and time have been set, the BASIC Interpreter program is loaded. The display reads:

```
Welcome to Fluke 1752A BASIC!
Ready
```

- The "Ready" prompt indicates that the BASIC Interpreter program is running in the Immediate Mode.

If all these things have happened as described, the 1752A is operating properly. The "Ready" prompt indicates that the 1752A has passed the Self Test, that the STRTUP.CMD program has run properly, and that the 1752A is now ready to receive commands in the BASIC language.

If you do not want to begin by programming in BASIC, it is a simple matter to exit the BASIC Interpreter program, and begin working with the Command Line Interpreter. To do that, type EXIT ⟨RETURN⟩ . The prompt for the operating system is FDOS⟩

To get back to BASIC, just enter the word BASIC, and then ⟨RETURN⟩

Below is a description of the software modules that make up the operating system on a new system software disk.

THE OPERATING SYSTEM

Because the 1752A is a programmed instrument, its functions are controlled by a master operating program. In the 1752A, this program is the Floppy Disk Operating System (filename FDOS2.SYS). This program is often referred to as "FDOS" in the 1752A Manual Set.

FDOS controls the hardware components of the 1752A. It takes instructions from the keyboard or from a program, and directs the functions of ports, manages the memory, and manipulates files to convert the instructions into action.

FDOS is soft-loaded, which means that it is recorded on a disk, rather than being permanently in the memory. The advantage of making FDOS soft-loaded is that it can be easily maintained and updated. Also, new software can easily be added without having to install special hardware. FDOS is loaded into memory whenever the the instrument is turned on.

For this reason, a floppy disk containing the Operating System file (FDOS2.SYS) must be in place when the power is turned on or when the RESTART button on the front panel is pressed. This file alone is not sufficient for proper operation, however. Another file, MACRO.SYS, which is also supplied on the 1752A System Disk must be included on any disk which is to be used to start up the 1752A. This file contains machine language subroutines used by the system software and other programs.

One more file on the System Disk, ALIAS.SYS, is also loaded into memory when the 1752A is booted up. This file is not required for proper operation of the 1752A, but it contains abbreviations for commonly used commands that can save time during programming. The alias file is explained in more detail in Section 7.

Since it is possible to relocate the Operating System into Electronic Disk or an optional Bubble Memory, and since the Bootstrap loader looks at these devices when it cannot locate FDOS on the floppy disk, it is not necessary to have the System Disk physically in place to load the Operating System. In fact, the loading process can be considerably speeded up by recording the Operating System into memory, and loading from there rather than from the floppy disk.

Command Line Interpreter

The part of the operating system that receives instructions from the keyboard is called the Command Line Interpreter. The Command Line Interpreter can accept either single command lines (instructions) from the keyboard, or the instructions contained in a Command file. Instructions can be entered either in lower case or upper case. To distinguish commands from other text, this manual shows only upper case.

Automating the 1752A using Command Files is described in detail in Section 7, Automating System Functions.

Other Software on the System Disk

The following programs are provided to assist you in configuring the 1752A and in developing software:

TIME — The Time and Date Utility. [*filename TIME.FD2*]
This utility program is used to set or read the time and date maintained by the 1752A calendar/clock circuitry. Once the clock is set to the correct time and date, it can be used to imprint programs or data. It can also be used to display the current setting. Battery power keeps it accurate when the power is turned off. The Time and Date Utility program is discussed in more detail in Section 7, Automating System Functions.

SET — The Set Utility. [*filename SET.FD2*]
This program changes the parameters at the RS-232-C port. These parameters govern the way information is sent and received between the 1752A and any devices connected to the serial communications port. One parameter, the length of time out, can also be changed for the IEEE-488 port. The Set Utility program is discussed in more detail in Section 5, Communications.

FUP — The File Utility Program. [*filename FUP.FD2*]
The File Utility is used to to create, delete, rename, and copy files, and to channel them between the various devices in the 1752A. The File Utility program is described at the end of the next section, Devices and Files.

EDIT — The Editor program [*filename EDIT.FD2*] A program for creating and editing other programs, the Editor lets you insert, delete, search for, and replace characters, lines, phrases, and strings in your files. See Section 6 for details.

BASIC — This is the program that enables the 1752A to run BASIC language programs. The Fluke BASIC Programming Manual set that is supplied with the 1752A is a complete reference for the BASIC language.

SYSGEN — The System Generation program. [*filename SYSGEN.FD2*]
This program is used to create operating system software to support configurations that include options.

COMPAT — The Compatibility Test program. [*filename COMPAT.FD2*]
This program prints the names, version stamps, and other data about an executable program file. A complete description of the Compatibility program is included at the end of this section.

CONFIG — The Configuration Data program. [*filename CONFIG.FD2*]
This program prints a list of the device names which the currently executing Operating System is configured to use. The program is described in detail at the end of this section.

OTP — The Object Translator Program. [*filename OTP.FD2*]
OTP converts object files created by the FORTRAN Compiler and the Assembler into a simpler format usable with the BASIC Interpreter's LINK statement. This program is used with the Assembly Language option (17XXA-201) and the FORTRAN option (17XXA-202)

TCOPY — The Touch-Copy program [*filename TCOPY.FD2*] A program that uses the Touch-Sensitive Display for listing and copying files or groups of files. TCOPY has a menu mode and a command mode to serve a variety of applications. See Section 4 for a description of all the TCOPY options.

MAKING A NEW OPERATING SYSTEM

Introduction

The System Generation program is provided on the System Disk with the filename SYSGEN.FD2. This program is a tool for making a new Operating System. In addition to the standard software device drivers, the file FDOS2.SYS that is supplied on the System Disk contains software device drivers for the following optional modules for the 1752A:

17XXA-002 Parallel Interface Modules
17XXA-004 and 17XXA-005 Bubble Memory Module
17XXA-009 Dual Serial Interface Module
1752A-010 Analog Measurement Processor
1752A-011 Analog Output
1752A-012 Counter/Totalizer

If you do not have any of the above optional devices, you may wish to create a new configuration of FDOS2.SYS using the System Generation program. Doing so will free up the memory used by the additional drivers and make it available for programs or Electronic Disk.

Device drivers are also available for the following 1752A peripherals:

1760A and 1761A Disk Drive Systems
1765A/AB Winchester Disk Drive

If you have any of these peripherals, you must use the System Generation program to reconfigure the operating system to include these drivers before using them with the 1752A.

The CONFIG program described later in this section can be used to determine the configuration of the Operating System currently in memory.

A Note About Software Compatibility...

The System programs are software modules supplied with the 1752A as files on the System Disk. These machine-language programs are interdependent and are compatible in the combinations supplied with the 1752A.

System programs are easily copied and erased, since they are treated as any other file. The portability and copying ease of system software allows you to take advantage of Fluke's continuing program of software development. However, it is possible to inadvertently record incompatible modules onto the same disk. Therefore, it is important to keep track of the various software modules on your disks. The COMPAT program described in this section can assist you in keeping track of compatible versions of software.

Use caution when copying new or updated software to make sure that the modules are recorded in the same combinations as the original disk. If a mistake is made, the operating system may not load the incompatible module, and display an error message.

Experienced programmers often suggest keeping a record of any changes made to software disks, to keep mistakes to a minimum and to make it easier to track down any problems that might occur. One way to do this is to use the /L option of the File Utility program to print a listing and keep it with the floppy disk.

Using the System Generation Utility

The System Generation program requires some files in order to build the operating system. Before beginning, use the File Utility Program to make sure you have these files available on your system device:

SYSGEN.FD2	(the program that generates the new FDOS)
FDOS2 .LIB	(library of modules to build FDOS)
FDOS2 .CFG	(used by SYSGEN to generate the prompts)

Also, decide if you want to keep a copy of the original Operating System configuration. If so, make a backup copy of the old version of FDOS2.SYS before running SYSGEN. This can be done by renaming the file using the File Utility Program (refer to Section 4, Devices and Files).

1. From the FDOS> prompt, type SYSGEN ⟨RETURN⟩.
2. When the program has loaded, the screen will display the System Generation Utility program identification, then will list the names of files it is linking to, and then begin asking if you want various drivers.
3. Answer Y to those that are desired, otherwise N. Only include drivers for the devices you will be using.

4. After you answer all the questions, the program reads the required software modules from the file named FDOS2.LIB, and links them together to create the new operating system.

5. Press RESTART, and allow the new Operating System to load. Test each module by using the File Utility Program to scan (/S), format (/F), or zero (/Z) the devices associated with each driver. Or you may run the diagnostic program for the desired module (refer to Appendix G, System Diagnostic Software).

If the file that SYSGEN is writing (FDOS2.SYS is the default) is file protected, the error message ?File protected is displayed and SYSGEN exits.

If you want SYSGEN to place the new FDOS on a device other than the system device, at step 1 enter "SYSGEN dvc:" where dvc is the name of the device. For example, if MF0: is the system device, but you want SYSGEN to put the new FDOS2.SYS file on ED0:, enter SYSGEN ED0:⟨RETURN⟩ from the FDOS⟩ prompt.

If you want the output of SYSGEN to be named differently than "FDOS2.SYS", then enter "SYSGEN filename" from the FDOS⟩ prompt. For example, to name the output of SYSGEN "FDOS2.NEW", enter "SYSGEN FDOS2.NEW⟨RETURN⟩". Only an operating system with the name FDOS2.SYS can be booted and executed on the controller. FUP can be used to rename the file FDOS2.SYS.

THE FDOS CONFIGURATION DATA PROGRAM

The FDOS Configuration Data program is a utility program supplied on the System Disk with the file name CONFIG.FD2. It prints the version number of the Floppy Disk Operating System (FDOS) currently in use, and the device names of all peripheral devices which FDOS has been configured to use.

Section 4 of this manual explains the the difference between serial and file-structured devices.

Entering the FDOS Configuration Program

From the FDOS〉 prompt, type CONFIG 〈RETURN〉. The CONFIG program will print the current FDOS version and configuration as follows:

```
FDOS> CONFIG
Devices configured with this FDOS (Version m.n)

EDO:     File-structured
MFO:     File-structured (SYO:)
KB1:     Serial
KBO:     Serial
GPO:     Serial
GP1:     Serial
MF1:     File-structured
MF2:     File-structured
MF3:     File-structured
MF4:     File-structured
WDO:     File-structured
WD1:     File-structured
WD2:     File-structured
WD3:     File-structured

FDOS>
```

NOTES

1. *If FDOS was configured with a Bubble Memory or a Dual Serial Interface driver, and neither of these devices are installed when the Controller is powered up, they will not appear on the above list.*

2. *If E-Disk was not configured when CONFIG was run the device name will not appear in the above list.*

3. *There are no device names for ports on the Parallel Interface Board (17XXA-002), therefore it also will not appear in the above list.*

Notice that the entry for the floppy disk, MF0:, also contains the name SY0:. This indicates that the "default system device" (SY0:) is currently assigned to device MF0:.

Writing Configuration Data to a File

The CONFIG program can write its data to a file instead of the screen. From the FDOS> prompt simply place a ">" character followed by a legal file name into which the output from CONFIG should be placed:

```
FDOS>CONFIG  >CONFIG.DAT
FDOS>
```

In this case, the configuration data will be written to the file "CONFIG.DAT" on the current "system device", named "SY0:". Any legal FDOS file name may be used.

CONFIG Program Help Information

The CONFIG program prints a short help message in response to an invalid command. A longer help message is printed in response to an explicit request.

An invalid command causes a one-line "usage" message to be printed. This is simply a quick reminder of how the CONFIG program expects to be told to do its job. For example:

```
FDOS>CONFIG +_&&**
Usage: CONFIG [?] [-version] [>file]
FDOS>
```

In this case CONFIG simply prints an example command line. The square brackets surrounding the command line options simply indicate that they may be omitted if desired.

A slightly more involved help message is produced by the question-mark command to CONFIG:

```
FDOS> config ?

The "Config" program prints the devices which may be used with the version
of FDOS executing on your machine.  The FDOS version number appears on the
first line.  A blank line follows.  Succeeding lines contain a device name
starting in the first column, a tab character, and the word "Serial" or
"File-structured" to indicate the device type.  For example, the command:

                FDOS> config

will print the FDOS configuration to the 1752A display.  The command:

                FDOS> config >myfile.cfg

will print the configuration data to the file "myfile.cfg".

FDOS>
```

Printing the CONFIG Program Version Number

The version number of the CONFIG program itself may be printed by using the command:

```
FDOS>CONFIG -VERSION
CONFIG  Version m.n
FDOS>
```

The letters "m.n" will be replaced by the actual 2-digit version number of the CONFIG program.

THE COMPATIBILITY TEST PROGRAM

The Compatibility Test Program is a utility program supplied on the System Disk with the file name COMPAT.FD2. It is a tool which allows the version stamp of executable program files to be printed and optionally checked for compatibility with any version of the Floppy Disk Operating System (FDOS).

Executable programs (all files with extensions .SYS or .FD2) contain a version stamp field when they are stored. This field records the version of FDOS under which the file was created. The version stamp is not necessarily the version number which is displayed when the program is run. For example, a copy of the File Utility Program, called FUP.FD2, has a version stamp of 1.0 if it is meant to be executed with version 1.0 FDOS.

Running the Compatibility Program

From the FDOS› prompt, type COMPAT ‹RETURN›. The COMPAT program will (by default) print the version numbers of all executable programs found on the current System Device (device SY0:). For example:

```
FDOS> COMPAT
List of executable files (all versions)

MFO: MACRO.SYS        1.5
MFO: FDOS2.SYS        1.5
MFO: BASIC.FD2        1.5
MFO: FUP.FD2          1.5
MFO: SET.FD2          1.5
MFO: EDIT.FD2         1.5
MFO: SYSGEN.FD2       1.5
MFO: TIME.FD2         1.5
MFO: TCOPY.FD2        1.5

FDOS>
```

Notice that in this case the System Device (SY0:) was the floppy disk MF0:.

Program Help Information

COMPAT will provide both long and short "help" messages. A short "usage" message is produced in response to a command with invalid syntax. This message is:

```
FDOS >COMPAT :
Usage:  COMPAT [?] [+-version=x.y] [+-fdos=x.y] [+-fdos] [+data]
        [>file] [pattern...]

FDOS >
```

This message is a short reminder about the correct usage of COMPAT's commands.

A longer help message is printed in response to the "?" command:

```
FDOS >COMPAT ?
The COMPAT program (version m.n) has the following options:
    +version=m.n    Print programs created under FDOS version m.n
    -version=m.n    Print programs not created under FDOS version m.n
    +fdos=m.n       Print programs compatible with FDOS version m.n
    -fdos=m.n       Print programs not compatible with FDOS version m.n
    +fdos           Print programs compatible with current FDOS
    -fdos           Print programs not compatible with current FDOS
    +data           Print data about program memory use and privilege
    pattern         Only check files matching pattern
More than one pattern may appear. Pattern characters are:
    dvc:            Check all .SYS and .FD2 files on dvc:
    *               Zero or more of any character (except ".")
    ?               Exactly one of any character (except ".")
    A-Z 0-9 . $ _   Characters to be matched exactly
```

Printing More Data From Image Files

COMPAT can give more information than just the version number about the files selected for printing. The command to do this is +DATA.

```
FDOS>COMPAT MFO: WD1: +DATA
List of executable files (all versions):

File name         Version Load    Entry  Size   Overlay  Space   Mode

MFO:MACRO.SYS     1.2     >0800   >0800  >13F2  No       Kernel  Privileged
MFO:FDOS2.SYS     1.2     >0100   >5D54  >61FE  No       Kernel  Privileged
MFO:BASIC.FD2     1.2     >0000   >82E4  >853E  No       User    Privileged
MFO:FUP.FD2       1.2     >0000   >0000  >1386  No       User    User
MFO:SET.FD2       1.2     >0000   >0000  >11C0  No       User    User
MFO:EDIT.FD2      1.2     >0000   >0000  >5C86  No       User    User
MFO:SYSGEN.FD2    1.2     >E000   >E000  >1886  No       User    User
MFO:TIME.FD2      1.2     >0000   >0000  >0318  No       User    User
WD1:BC.FD2        1.0     >0000   >0000  >A284  Yes      User    User
WD1:BSCRUN        1.0     >AA26   >AA26  >55D8  No       User    User
WD1:HDT.FD2       1.0     >E000   >E000  >0FFE  No       Kernel  Privileged

FDOS>
```

The information printed by the +DATA command is simply the information which FDOS uses to load a machine language program into memory. The File name and Version fields have already been discussed. The other information fields are:

Load — is the first (lowest) memory address to be used when loading the program into memory.

Entry — is the memory address at which the program initialization code begins. FDOS transfers control to this location to start the program's execution.

Size — is the number of bytes which the program requires to hold both instructions and any initialized data. More memory than this may actually be required to execute the program. For overlaid programs this number is the "longest overlay path", or the maximum memory used by any sequence of overlays.

Overlay — is "Yes" if the program uses overlays, otherwise "No".

Space — is "Kernel" if the program must be loaded into the lowest 64K bytes of memory (which is where FDOS is loaded), or "User" if the programs may be loaded into any memory region other than the lowest 64K bytes.

Mode — is either "User" or "Privileged". This indicates whether the program executes in User mode, which prevents the use of certain critical machine instructions, or in Privileged mode, which permits the program to address any memory location and to execute any machine instruction.

Directing Output to a File

The output of COMPAT may be written to a file instead of to the display simply by using a "〉" character followed by the name of the file to be created. For example:

```
FDOS〉 COMPAT 〉COMPAT.DAT
FDOS〉
```

In this case the output of COMPAT would be written to the file COMPAT.DAT.

USING OTP

OTP (the Object Translator Program) converts object files created by the FORTRAN Compiler (Option 17XXA-202) or the Assembler (17XXA-201) into a simpler format that can be used with the BASIC Interpreter's LINK statement.

1. Compile any FORTRAN subroutines by using the FORTRAN Compiler (FC.FD2). If any FORTRAN subroutines are used you use OTP in step 4.

2. Use the Assembler (ASM.FD2) to assemble any Assembly Language subroutines. If any of the Assembly subroutines use DSEG (Data SEGment) or CSEG (Common SEGment) directives you use OTP in step 4.

3. Use the Linkage Editor program (LE.FD2) to do a PARTIAL link of the object files created in steps 1 and 2 with the Parameter Linkage Routine object file F$RGMY.OBJ.

4. This step is optional, but see steps 1 and 2. Run the object file created by step 3 through OTP to create an object file which may be used by the BASIC Interpreter's LINK statement.

5. Use the LINK statement in the BASIC Interpreter to load the object file created in step 4 with an executing BASIC program.

CONCLUSION

This section has described the programs that are recorded on the System Disk supplied with the 1752A. These programs are tools for the system designer to use in setting up an instrumentation system. None of these programs actually control an instrumentation system, but they facilitate writing those programs that do.

To make use of these programs requires using some other tools, referred to as the system's resources: devices and files.

The next section introduces these system resources and explains how to use each device and each type of file.

Section 4
Devices And Files

CONTENTS

Introduction 4-2
  Devices 4-3
  Files 4-5
Devices 4-6
  RS-232 Ports (KBn:, SPn:) 4-6
  IEEE-488 Ports (GPn:) 4-7
  Floppy Disk Drives (MFn:) 4-7
  Electronic Disk (ED0:) 4-8
  Bubble Memory (MBn:) 4-8
  Winchester Drive (WDn:) 4-8
Files 4-9
  System Files 4-9
  Alias File 4-9
  Command Files 4-9
  The Startup Command File 4-9
  Other Command Files 4-10
  Machine Executable Files 4-10
  Language-Dependent Files 4-10
The File Utility Program 4-11
  Introduction 4-11
  Entering the File Utility Program 4-11
  The Help Command 4-11
  Directory Allocation 4-13
  Using the File Utility Program 4-14
  Alphabetical Listing of Commands 4-18
  Syntax Diagrams 4-32
  System Messages 4-36
The Touch Copy Program 4-39
Conclusion 4-49

INTRODUCTION

In Section 3, the 1752A Data Acquisition System was described as a machine having two components: hardware and software. Together they operate as a Data Acquisition System. Both the software and the hardware are necessary parts of a working data acquisition computer. This section expands that view, and introduces a third aspect of the 1752A: the resources with which it will perform the task of controlling a Data Acquisition System.

In this chapter the concepts of devices and files are introduced since they are key ideas for understanding the operation of the 1752A. The resources of the 1752A are its devices and files. A device can be thought of as a place to store information, like a file cabinet. Electronic files are stored in the 1752A's devices just as hardcopy files are stored in file cabinets. Files can also be retrieved from a device just as from a file cabinet. Devices and files are discussed together here because they are so closely related. This section describes how to use the File Utility program to manipulate devices and files. Section 7, Automating System Functions, explains how to manipulate devices and files under program control.

Devices

The term device is used in three different ways: external instrumentation, file-structured devices, and non-file-structured devices. Understanding the distinctions is essential to effective programming.

Every IEEE-488 instrument is called a device, and the 1752A sends information out to a device address when it sends program data to the instrument, or it sends a command to the instrument to take a measurement or to send back measurement data.

Instruments are devices external to the 1752A. The 1752A has a number of internal devices as well. In this context, a device is a hardware resource that can act as a source or destination of data. There are two types of these internal devices. One type is called file-structured, and can be thought of as a location to store programs or data. File-structured devices include the floppy disk, Bubble memory, and E-Disk. In general, only this type of device can be assigned the function of system device (the one the 1752A assumes you mean if you do not specify a device).

The other kind of internal device is usually called a serial device to distinguish it as a pipeline for information, rather than as a location of information. Notice that the term serial has a slightly different meaning than usual when it refers to a device. It may be that a serial device sends serial data, but not necessarily. For example, the RS-232-C port is a serial device, and is used to send serial data; the IEEE-488 port is also a serial device, but the data is sent bit-parallel, byte-serial.

On power-up (a cold start), the bootstrap PROM checks the floppy disk drive for the operating system program (FDOS). If it is not there, the PROM checks the other file-structured devices, first the Electronic Disk, and then bubble memory. If there is no system software resident on one of those devices, an error message indicates that there is "no system on device".

When the RESTART button is pressed, the system performs a warm start. A warm start differs from a cold start in that the memory is not cleared, nor is the self test performed; only the Operating System is loaded. If there are any files stored in the Electronic disk, they remain intact.

The table below defines each of the 1752A's devices. Notice that all device names must have two alphabetic characters, followed by a number, and ended with a colon. The colon must always be included because it is part of the name.

DEVICE NAME	SYSTEM RESOURCE	TYPE
Standard System		
KB0:	Keyboard (Input) Display (Output)	Serial
KB1:	RS-232 Port	Serial
GP0:	IEEE-488 Port	Serial
MF0:	Mini-Floppy Disk Drive	File-structured
ED0:	Electronic Disk	File-structured
Optional Resources		
KB2:	Optional RS-232 Port (Option -008)	Serial
GP1:	Optional IEEE-488 Port (Option -008)	Serial
SP0: - SP9:	Optional Dual Serial Interface (Option -009)	Serial
MF1: - MF4:	Optional Floppy Drives	File-structured
MB0: - MB3:	Optional Bubble Memory	File-structured
WD0: - WD3:	Optional Fixed Disk Drives	File-structured

Files

A file is an organized record of related information. The file type can usually be identified by its extension (the three characters following the file's name, and it is separated from the filename by a period). The 1752A uses several types of files:

System Level Files

The following table lists the types of files used by the 1752A.

FILE TYPE	EXTENSION	DESCRIPTION*
System	SYS	Reserved for 1752A System Operations
Command	CMD	A collection of keyboard commands
FDOS	FD2	Binary machine language
Configuration	CFG	Used by SYSGEN to generate FDOS2
Help	HLP	Data file for Help screens
Place holder	BAD	Indicates bad areas during packing
Source	BAS	Default extension
Lexical	BAL	Results when a file is SAVELed
Backup	BAK	Created by System editor program
Source	FTN	FORTRAN

Other Files

Assembler Source	ASM	Output of ASMPP, Input to ASM
Error File	ERR	Output of BC
Library	LIB	Input to LE, LM, and LL
General List	LST	Output of ASM, FC
Map File	MAP	Output of LL, LE
Object	OBJ	Output of BC, FC, ASM; input to LE, LL, LM
Preprocessor	PRE	Input to ASMPP
Temporary	TMP	Temporary file for BC, FC and LE

* Description abbreviations

ASM = Assembler
ASMPP = Assembler Pre-Processor
BC = BASIC compiler
FC = FORTRAN compiler
LE = Linkage Editor
LL = Linking Loader
LM = Library Manager

DEVICES

RS-232-C Ports

All of the KB devices are serial. They are RS-232-C ports, and they provide an entry and exit point for serial communications between the 1752A and other RS-232-C compatible equipment. RS-232-C is a designation for a standard digital communications interface, and it describes the connector and voltage levels used in bit-serial communications. The standard permits many of the operating characteristics to be changed, to allow the connection of many types of equipment. Section 5, Communications, gives more information about the standard and about how to change the 1752A's RS-232-C port parameters.

KB0:

KB0: is both an entry and exit point for information between the 1752A's program and the outside world. As an input port, it is the Y1700 Keyboard. As an output, it is the display itself. KB0: is sometimes called the Console Device. None of the operating parameters of KB0: can be changed except the baud rate, but it should only be changed when using an external terminal. Otherwise, an error will result.

KB1:

KB1: is the built-in RS-232-C port for connecting the 1752A to other equipment that uses the standard interface. It is set to a standard configuration on power up, and it can be customized to different characteristics using the Set Utility program. See Section 5 for a detailed discussion of this utility program. KB1: does not exist unless the Video/Graphics/Keyboard module is installed in the 1752A.

KB2:

This device name is used for an optional RS-232 port, and is used if Option -008 (IEEE-488 / RS-232 Interface) is installed. It operates exactly like KB1:.

SP0: - SP9:

These are the device names for the ports on the Dual Serial Interface Option -009. The ports are treated similarly to the KBn: devices. Up to three modules can be installed, and each port can be configured for RS-232-C, RS-422, or 20 mA Current Loop electrical interfaces.

IEEE-488 Ports

GP0:

As shipped, the standard 1752A has a single IEEE-488 port. The IEEE-488 port has the device name GP0: when it is used as a serial device (output only), and Port 0 when it is used by a program as an instrument port.

One application for GP0: is to use an IEEE-488 compatible printer as a listing device. Rather than writing a "PRINT" program, information can be sent out simply by specifying GP0: as the destination device.

GP1:

The device name GP1: is used for the optional IEEE-488 port, and is only implemented if Option -008 (IEEE-488 / RS-232-C Interface) is installed. Its purpose and operation are identical to GP0:.

Floppy Disk Drives

MF0:

The integral 5-1/4 inch floppy disk drive provides the 1752A with removable storage media (MF stands for mini-floppy). Using the floppy disk drive, a collection of programs can be built up, so that a different floppy disk would be used for each set of tests.

Floppy disks must be formatted prior to use. Formatting is the process of sectioning off the disk so that information written onto it is allocated to the proper location, and so that the 1752A is able to locate it again after it is recorded. Floppy disks can be formatted either as single- or double-sided (see File Utility program for details).

All floppy disk operations can be simplified by using File Utility commands in a Command file and by using the 1752A's Alias file.

The MF0: device is the default device at power up. It is possible to designate another device as the location of system software, however. See the discussion of the File Utility program's Assign command.

MF1: MF2: MF3: MF4:

Four other disk devices can be connected to the 1752A at the IEEE-488 connector. The Fluke model 1760A is a single 5-1/4 inch unit, and the 1761A has two drives. In this usage, each device acts similar to MF0:. Operation over the IEEE-488 bus is transparent to the user.

When optional floppy disk drives are added, the System Generation Utility program must be used to create new software to include the necessary driver routines.

Electronic Disk

ED0:

The ED0: device designates the Electronic Disk (E-Disk). The programmer can designate portions of memory as "Electronic Disk". Any area of memory not used by the Operating System can be designated as E-Disk. When E-Disk is used, it must first be configured, a process that allocates how much space is to be used for E-Disk. For details, see the File Utility program /C command later in this section.

Because the E-Disk device is implemented in random access memory, it will perform any operation more than 100 times faster than if the floppy disk is used. However, the memory is volatile, so any programs or data stored in E-Disk are lost when the system is turned off.

Bubble Memory

MB0: - MB3:

These are the device names for Bubble Memory Options -004 and -005, 512 and 1024 blocks respectively. A block of memory is 512 bytes.

Regardless of the capacity of the Bubble Memory module(s) installed, each module is a separate device, whose device designation is selected by a switch setting on the board. Up to three Bubble Memory modules can be installed in the 1752A card cage. When Bubble Memory is added to a system, the System Generation Utility program must be used to create new software to include the necessary driver routines.

The optional Bubble Memory modules provide the 1752A with a large amount of additional non-volatile storage capacity.

Winchester Drive

WD0: - WD3:

WD0: through WD3: are the devices associated with the Winchester Disk Drive, an optional 5-1/4 inch hard disk. It is connected to the 1752A at the IEEE-488 connector, just as any optional floppy disk drives would be. If a 5M byte drive is installed, two devices are available: WD0: and WD1:. If the drive is the 10M byte version, all four devices are available for additional on-line storage.

When A Winchester drive is added to a system, the System Generation Utility program must be used to create new software to include the necessary driver routines.

FILES

A file is a structured collection of information that the 1752A can use to hold programs or data for later use. Most of the files supplied with the 1752A contain programs. The program type is usually indicated by the extension (three characters after the filename). Extensions are always separated from the filename by a period.

System Files

System files have the SYS extension. Together, they make up the collection of programs that are the 1752A's system operation programs.

Alias File

The alias file is a special type of system file that makes it possible to condense long commands into shorter, more easily remembered ones. The system alias file (filename ALIAS.SYS) is discussed in geater detail in Sections 6 and 7.

Command Files

A Command file is a collection of keyboard commands. It has the extension CMD. Using Command files makes it possible to automate keyboard commands to the Operating System through the Command Line Interpreter.

The Startup Command File

Command files allow the user to customize the operation during power up and during normal operation of the 1752A. The Startup Command file (filename STRTUP.CMD) is executed whenever the 1752A is powered up or RESTARTed. Other user-named command files (with the .CMD extension) can be used whenever the FDOS prompt is displayed.

The STRTUP.CMD file on the System Disk supplied with the 1752A checks the time and date clock to see if it has been set, then returns control to the Operating System. The user can create a customized STRTUP.CMD file to perform such functions as displaying special messages at startup, setting the baud rates on the serial ports, and loading the BASIC interpreter.

Other Command Files

In addition to the Startup Command file, others can be created to automatically perform any sequence of keystrokes. The real usefulness of a Command file is that it automatically performs commands that otherwise would have to be keyed in individually each time the system was used.

Command files are discussed in more detail in section 7, Automating System Functions.

Machine Executable Files

These are binary machine language programs that can be run directly by the microprocessor. They usually use the FD2 extension. Because they are actual binary machine instructions, FD2 programs do not have to be translated from a higher level language before the microprocessor can perform their operations, as other programs must.

New files of this type are created using an optional linkage editor/loader program.

FD2 files were called "Core Image Load" files in the 1720A, and used the extension CIL.

Language-Dependent Files

Each programming language has unique properties just as human languages do. Among these unique properties are the file types that they use. In an effort to keep this discussion away from individual languages, only generic file types are explained. For detailed explanations of the file types used by a specific language, refer to the individual programming language manual.

Source Files

Source files contain programs that are written in high-level language. An optional assembler or compiler program translates source files into an executable form.

Object Files

An object program is the result of the translation of a source program. It may be an intermediate step to a machine-executable program, or it may be a directly executable form of a program written in a high level-language.

THE FILE UTILITY PROGRAM

Introduction

The File Utility program is a utility software file supplied on the System Disk with the file name FUP.FD2. It gives the user control over the files in any of the devices. A flexible structure provides other useful capabilities. The examples in this section illustrate the many ways that the File Utility program can be used.

Entering the File Utility Program

From the FDOS> prompt, type FUP ⟨RETURN⟩. The screen will display the identification and prompt of the File Utility program:

```
FDOS> FUP

File Utility Program  Version x.y

FUP>
```

The Help Command

From the FUP> prompt, type ? ⟨RETURN⟩ to see a listing of the command options.

This command causes the file FUP.HLP to be displayed. As supplied on the System Disk, this file is a one-screen summary of command options. An error message indicates if the file FUP.HLP is not found.

```
Quick Summary of FUP Commands
-----------------------------
Assign system device ...... <device>/a   Pack .................... [device]/p
Configure E-Disk ........ ed0:/c<size>   Protect ................ <file>/+[i]
Copy a file .... [<device>=]<file>[/i]   Quick directory list ..... [device]/q
Delete .................... <file>/d[i]  Rename ............. <file>=<file>/r
Extended directory list ... [device]/e   Scan for bad blocks ...... [device]/s
Format ........... [device]/f[s][segs]   Defeat error checking <file>=<file>/t
Directory list ............. [device]/l  Unprotect ............... <file>/-[i]
Merge ....... <file>=<file>{,<file>}/m   Whole copy  <device>=[device][file]/w
                                         Zero directory .. [device]/z[s][segs]

Wildcards:"?" matches single character, i.e.: "alias.s?s" matches "alias.sxs"
          "*" matches any characters, i.e.: "*.bas" matches all basic programs
Individual: place an "i" after command, i.e.: *.bak/di, *.fd2/-i
```

All the command options are listed below with examples. There is an explanation of each option later in this section.

	USAGE	EXAMPLE
Options		
no option	Transfer	MF0:TEST.BAK=ED0:TEST.BAS
/A	Assign System Device	ED0:/A
/B	Binary File Copy	MF0:FUP.SYS=MB0:FUP.SYS/B
/C	Configure E-Disk	ED0:/C (block size)
/D	Delete a File	TEST.TMP/D
/E	Extended Directory	MF0:/E or ED0:/E or /E
/F	Format a Disk	ED0:/F or MF0:/F3
/L	List Directory	MF0:/L or KB1:=ED0:/L or /L
/M	Merge ASCII Files	TEST.NEW=TEST.1, TEST.2/M
/P	Pack a Device	MF0:/P or ED0:/P or /P
/Q	Quick Directory	ED0:/Q or MF0:*.BAS/Q
/R	Rename a File	TEST.1=TEST.OLD/R
/S	Scan for Bad Blocks	MF0:/S or /S
/T	Transfer w/o Error Check	ED0:TEST.BAD=MF0:TEST.BAS/T
/W	Whole Copy	ED0:MF0:/W
/X	Exit FUP	/X
/Z	Zero File Directory	ED0:/Z or MF0:/Z or /Z
/+	Protect a File	ED0:1722A.INC/+
/-	Unprotect a File	MF0:RZDZ.WIZ/-
Switches		
/D, /S	Double- or Single-sided disk format	MF0: /FD or MF0: /FS3
/I	Interactive Switch	ED0: =MF0: *.*/I
Wildcards		
*	Match all characters in field	MF0: *.*/-I
?	Match single character	ED0: =MF0: TEST?.BAS

Directory Allocation

Unless otherwise specified during formatting, all of the 1752A's file-structured devices provide a directory that can contain 72 filenames. If more files are needed, the File Utility program can allocate additional segments for directory space. Each additional segment can also contain 72 entries.

This capability can be particularly valuable when large file-structured devices are added (e.g. Winchester drive, RAM Expansion modules), or in those cases where a great many small files are to be recorded on a floppy disk. Notice that when an Extended Directory Listing (/E) is requested for a device with multiple directory segments, the segments may be separated by ⟨not used⟩ entries, even if the device is packed.

The command for allocating additional directory space on the floppy disk, E-Disk, or Bubble Memory is /F, followed by the number of directory segments desired. Formatting Winchester disks requires using the WDUTIL program provided with the hard disk drive.

Using the File Utility Program

If an option affects more than one device or file, the first must be the destination, and the second the source. If no device is specified, the system defaults to the SY0: device. If no filename is specified, the system uses a null filename. Finally, if no extension is specified, the system uses the BAS extension. When both a device and filename are specified, they are separated by a colon (:).

The complete name of a file takes the form:

dev:filename.ext

This is called a *pathname*. Often, a pathname can be specified without directly naming each of its parts. This is done using defaults and wildcards, which will be discussed in more detail later in this section.

CAUTION

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

This manual uses upper case letters to indicate commands; however, both upper- and lower-case entries are allowed. Each command line must contain only one command.

All commands can be automated from FDOS through Command files and aliases.

Wild Cards * and ?

A wild card is a character that can be used in place of another character or string of characters. The File Utility program can use two wild cards in the filename: the ? character, and the * character. Wild cards cannot be used when specifying devices.

* matches all characters from the wild card until another character or the end of the filename or the extension.

Usage: A*.BAS would match AA.BAS, A1.BAS, and AA1.BAS.

*.BAS matches all files with the BAS extension.

. matches all files.

Example: To list all BAS files on the floppy disk:
*/L

To delete everything stored in the E-Disk:
ED0:*.*/DI

? matches any character in that position.

Usage: A?.BAS matches A1.BAS and AA.BAS, but not AA1.BAS.

Example: To print the BASIC files named TEST1, TEST2, TEST3, TEST4, TEST5, TEST6, TEST7, TEST9, and TESTA that are currently recorded on the floppy disk, using a serial printer conected to the RS-232 port:

KB1:=MF0:TEST?.BAS/T

Protection States + and -

All files are assigned a protection state. A protected file (+ state) may not be erased or rewritten. It is as if the protected file had a write protection tab. If an attempt is made to erase or overwrite a protected file, an error message indicates that the file must be unprotected before continuing.

Newly created files are unprotected (- state), to allow them to be changed easily. Once the file is complete, or in final form, it can be made into a protected file to prevent accidental erasure or write-over.

The protection states are shown in extended directory listings, and are changed with the /+ and /- command options.

CAUTION

The protection state is ignored during zeroing (/Z) and formatting (/F) floppy disks, and during configuring E-Disk space (/C). These commands delete all files associated with the device regardless of protection state. Be sure to backup any desired files before using these options.

Example 1.
To change the protection state of a file called DUTCH.PIM on the floppy disk:

```
MF0:DUTCH.PIM/+
```

Example 2.
To unprotect all the files on the system device, using the individual switch to insure a message before proceeding:

```
*.*/-I
```

Switches I, D, and S

A switch modifies a command, and always follows it. Three are available: the Interactive switch and the disk format switches D and S.

The Interactive switch permits individual selection of files to be copied, transferred, deleted, and protected or unprotected. When the Interactive switch is used with a file deletion command, a message indicates which files are protected.

When the Interactive switch is used with the no option command, separate the two device names with an = sign. The system requests confirmation before completing the transfer. The Interactive switch cannot be used with a file merge (/M) command, nor with any commands that apply only to a device, such as /A, /C, /S, or /P.

Example 1.
To delete selected BAK files from the E-Disk:

```
ED0:*.BAK/DI
```

Example 2.
To transfer selected files from the floppy disk to the display:

```
KB0: = MF0: *.*/I
```

The "D" switch selects double- or single-sided format for the floppy disk. For double-sided, use the argument D after the Format or Zero option. For single-sided, use the argument S. See the /F option for more details about formatting floppy disks, the /C option for information about configuring the Electronic disk, and /Z for details about zeroing the directory of any disk device.

Example
To format the floppy disk as single-sided, with 5 segments:

```
MF0:/FS5
```

Alphabetical Listing of Commands

(no option) Transfer

If no option is specified, a communication channel is established between the specified destination and source(s).

- Up to eight sources may be specified.
- When the destination is a file device (MF0:, ED0:) and no destination filename is specified, the names of the source files are used.
- If the source is not a file device (no name can be identified), the resulting destination file will have the null name, unless one is specifically named.
- If a single file is named as the destination for multiple files, only the last file specified will effectively be copied. Use the /M option to merge files.

The examples below show the ways in which the no option Transfer command can be used. A short description precedes each example.

Example 1.
To make a copy called FILE.NEW of FILE.OLD (both on the System Device):

```
FILE.NEW=FILE.OLD 〈RETURN〉
```

The result will be that two identical files exist, one called FILE.OLD, the other called FILE.NEW.

Example 2.
The display (KB0:) is specified as destination, and T44.BAS on the floppy disk (MF0:) as source. This will display the file. Use Page Mode if the file is longer than 16 lines.

```
KB0:=MF0:T44.BAS 〈RETURN〉
```

Example 3.
This command is equivalent to Example 2, if the System Device is the floppy disk, and the Console Device is KB0:. By using defaults, 13 of the 17 keystrokes have been eliminated in transfering T44.BAS from the floppy disk to the display. The default filename extension is BAS.

T44 〈RETURN〉

Example 4.
To transfer ASCII data from the keyboard (KB0:) to a printer connected to an optional serial port 2 (KB2:) In this example, a 〈CTRL〉/Z would terminate the transfer.

KB2:=KB0: 〈RETURN〉

/A Assign the System Device

The /A command option assigns the named device as the the System Device, SY0: (the default file device).

The example assigns the Mini-Floppy Drive (MF0:).

MF0:/A ⟨RETURN⟩

/B Binary Transfer

The Binary Transfer option is only implemented to assist those who have become proficient using the Fluke 1720A Instrument Controller. For normal transfer of 1752A programs and data, use the no option or /T commands. No errors will result.

The 1720A /B option transfers binary-coded data, such as system and utility software files, lexical-form BASIC programs and virtual array files. Up to eight individual source files can be specified, and wild cards can be used to increase the number.

This example uses wild cards to transfer a group of date-coded files (for the month of July) named RAC from the Electronic disk (ED0:) onto a floppy disk (MF0:).

MF0:=ED0:??0783.RAC/B

/C Configure Electronic Disk Space

The /C option is followed by a number to indicate how many blocks of Electronic disk are to be created. An argument of -1 (/C-1) allocates all available memory as E-Disk. If the number argument is left out, the E-Disk will be de-allocated. An argument of zero blocks (/C0) also de-allocates the Electronic disk.

After configuring E-Disk space, use the /F option if more than one segment is desired.

NOTE

Though specified in blocks, space is allocated by the page. Each page (4096 bytes) has 8 blocks, so the actual number of blocks allocated will be a multiple of 8.

/D Deleting Files

The /D option is used to delete up to eight specified files (more using wild cards). Deleting a file leaves a gap in the file structure. Refer to the Pack (/P) command option. When no file is specified, the null file is deleted. A single command line can be used to delete files from more than one device.

This example uses wild cards to delete all files having a .TST extension from the System Device, and all files whose names start RZDZ from the floppy disk.

```
*.TST,MF0:RZDZ.*/D
```

/E Listing a Directory (Also /L and /Q)

There are three ways to list directories. As with all File Utility program options, the device must first be specified unless the directory of the System Device is desired. If no destination device is specified, the 1752A assumes that the directory is to be sent to the display.

Wild cards can be used with all three listings to see only files whose names or extensions match, and the directory entry for a single file can also be obtained by specifying the file in the command line.

/E yields an Extended Listing. The extended listing includes all unused file areas and the protection state of each file.

Packing a File Structured Device, below, tells how to restructure the disk to remove unwanted blank areas.

/L is the normal listing of all files on the specified device. It displays all 5 fields, but does not include the unused areas.

/Q gives a Quick listing. It does not display the file size, nor the date the file was last updated. The filenames and their extensions are displayed, six columns across the screen, rather than one column as in the other two types of directory listings.

Here is a portion of the extended directory for a 1752A System Disk:

```
FUP MF0:/E
Directory of MF0: on 15-Jul-84 at 08:36

name   .ext      size      prot      date

FDOS2 .SYS        60       [+]      15-Mav-84
MACRO .SYS        12       [+]      13-Apr-84
ALIAS .SYS         1       [+]      04-Jan-84
FUP   .FD2        10       [+]      09-Mav-84
TIME  .FD2         2       [+]      09-Mav-84
SET   .FD2        47       [+]      09-May-84
BASIC .FD2        67       [+]      08-Mav-84
EDIT  .FD2        47       [+]      08-May-84
STRTUP.CMD         1       [+]      13-Apr-84
SYSGEN.FD2        15       [+]      09-May-84
FDOS2 .CFG         1       [+]      15-Dec-84
FDOS2 .LIB       102       [+]      15-Mav-84
GRAPH .OBJ         4       [+]      05-Jan-84
PIBLIB.OBJ         4       [+]      05-Jan-84
COMPAT.FD2        30       [+]      08-Mav-84
CONFIG.FD2        19       [+]      08-May-84
```

There are five fields: the **name** and **ext**ension of all the files appear in the first two fields. The size field indicates the number of blocks the file occupies. Each block is 512 bytes.

The **prot** field indicates the protection state of the file, and **date** is the last date the file was updated.

Two entries may appear in the Extended Directory list that do not appear in either the normal list or the quick listing:

<not used> indicates a blank area within the structure of the disk, left when a file was deleted. The Pack command /P packs all unused blocks into the end of the segment.

<temp ent> indicates that some problem occurred when a file was open (being transferred or edited). Some typical examples would be that the power was removed, or the RESTART button was pressed during operation on an open file. The system places a temporary entry in the segment to indicate that the file no longer exists. To delete the temporary entry, pack the disk with the /P command.

Here is what the screen looks like when a Quick Listing is done for the MF0: device:

```
FDOS2 .SYS   MACRO .SYS   ALIAS .SYS   FUP   .FD2   FUP   .HLP   TIME .FD2
SET   .FD2   BASIC .FD2   EDIT  .FD2   STRTUP.CMD   SYSGEN.FD2   FDOS2.CFG
FDOS2 .LIB   GRAPH .OBJ   PIBLIB.OBJ   COMPAT.FD2   CONFIG.FD2   OTP   .FD2
```

In this example, a quick listing is sent to the optional serial port 2 (KB2:) of the directory of the floppy disk (MF0:) and the Electronic disk (ED0:):

KB2:=MF0:,ED0:/Q ‹RETURN›

/F Format a File Device

The /F option prepares a floppy disk (MFx:) or optional bubble memory device (MBx:) to receive files by creating a completely new magnetic structure on them. Because formatting writes new block identification codes and standard data patterns throughout the device, any device that is formatted will also be erased. Take care not to format disks that have data or programs that you want to save.

When formatting the bubble memory or a floppy disk, a number can follow the /F command to indicate the number of directory segments to be established. For more information see the discussion "Directory Allocation" at the beginning of this section. If no number is given, the default is 1 segment, a useable amount for floppy disks, but restrictive for the bubble memory, because it would not take full advantage of the large amount of memory available. Since the directory for any one segment can contain only 72 entries, selecting one segment results in a maximum of 72 files, which is fine for most floppy disk applications, but is not an efficient use of the mass storage available in bubble memory or Winchester hard disks.

If a disk has suffered media damage, a message will display indicating that it is not able to be formatted. If this happens, the disk is not useable and should be replaced.

This option formats either floppy or Electronic disks, or Bubble Memory devices.

Example 1:
This example formats, zeroes, and verifies a floppy disk:

```
MF0:/F ‹RETURN›
```

Example 2:
This example formats a bubble memory device with 12 segments (864 possible files):

```
MB0:/F12 ‹RETURN›
```

Example 3:
This example uses the double-sided switch to format a floppy disk as double-sided, with 6 segments:

```
MF0:/FD6
```

/I Interactive Transfer

Besides its use as a command modifier, /I can be used alone to transfer files interactively. Just as with other transfer commands (no option, /B, and /T), wildcards are allowed. If no destination is specified, the default device is KB0:. If no source pathname is given, the System Device SY0: and the null file are assumed.

Example:
This example transfers selected files from the E-Disk to the floppy disk.

```
MF0: = ED0: *.*/I
```

/L Listing a Directory - See /E for all directory listings.

/M Merging ASCII Files

The /M option merges up to eight ASCII source files into one destination file. Binary files, such as System and utility software files, lexical form BASIC programs (BAL extension), and virtual array files cannot be merged. The source files remain intact, unless the destintion has an identical filename (on the same device). Wild cards and the Interactive switch cannot be used.

When the destination is a non-file device, the /M option removes the ‹CTRL›/Z character (ASCII EOF) from the end of all but the last file.

CAUTION

When merging two BASIC programs, duplicate line numbers can cause problems. When the BASIC Interpreter program encounters a duplicate line number, the latest occurrence of that line number is retained, and the previous occurrence is deleted. Use the REN statement to renumber the programs to different line number ranges before merging.

Here are two examples of how to make efficient use of the /M option:

Example 1.
This example creates a file on the floppy disk (MF0:) called PROGRM.T44. The new file contains TEST1.BAS and TEST2.BAS from the floppy disk. The original files all remain intact.

MF0:PROGRM.T44=MF0:TEST1,TEST2/M ‹RETURN›

Example 2.
This example appends keyboard input directly to the end of an existing file without creating a new one. The destination file DEST.CPT contains the old file DEST.CPT followed by keyboard inputs. A keyboard entry of ‹CTRL›/Z terminates the keyboard portion of the input.

DEST.CPT=DEST.CPT,KB0:/M ‹RETURN›

/P Packing a File-Structured Device

The /P command option reorganizes a file-structured device. When files are deleted, blank areas are left within the file structure. (See /F for more information.) The /P option compacts these areas into one contiguous space. It may be possible to make room for a file that previously wouldn't fit by packing the device. When this option is used, the file structure is maintained. During packing, the keyboard is disabled from display, but keystrokes are buffered. This feature makes it possible, for example, to give the command /E during packing. When packing is complete, the extended directory listing will display.

This example will pack the System Device:

/P ⟨RETURN⟩

This example packs the Electronic disk and the floppy disk:

ED0:,MF0:/P ⟨RETURN⟩

/Q Quick Directory - See /E for all directory listings.

/R Renaming a File

The /R option is used to rename a file. It has no effect on the size or location of a file, because it only operates on the directory.

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

PROG.T4=TEST4/R ⟨RETURN⟩

/S Scanning for Bad Blocks

The /S option scans a file-structured device for bad blocks, and sends a result message to the specified destination. Each block is read and checked for errors. A check character is compared with one recorded with the block. Any mismatches cause an error message.

Scanning for bad blocks is done to check for a faulty floppy disk or Bubble Memory device. If you are having trouble reading or recording on a disk, use the /S option to determine if the disk is useable. If bad blocks are indicated, attempt to transfer the files to another disk, then discard the faulty one. Bad blocks are a result of wear, age, or abuse.

If no destination is specified for the result, any errors found will be displayed on the console.

To scan the System Device and see the results displayed:

/S ‹RETURN›

This example scans the floppy disk and sends the results to serial port 1:

KB1:=MF0:/S ‹RETURN›

/T Transferring Files Without Error Check

Except for inhibiting error checks, this command is identical to the no-option command. It transfers files just as the no-option command does, but does not check for device errors. If errors should occur, they are ignored, and the file is transfered as is. This option can be used to create backup copies of files that are suspected to contain errors.

/W Whole Copying a File Device

The /W option transfers some or all of the files on a file device at one time using a single command. This option simplifies duplicating a floppy disk.

The source and destination devices should be different. (If they are the same, the result would be merely to record a file back to the same place it was read from.) To duplicate all or part of a floppy disk, first copy files into ED0:. Then insert a formatted disk and copy from ED0: to MF0:.

If the disk already contains files having the same name as those being copied, the whole copy command deletes the existing file and replaces it with the one being copied.

If E-Disk space is less than the total size of the files to be copied, break the task into smaller parts by copying only some of the files on several passes.

The display indicates the name of the file being copied. The whole copy process can be terminated before all files are copied by typing 〈CTRL〉/C during the copy of the last file desired. The file copy in progress will be completed before the operation stops.

CAUTION

If 〈CTRL〉/P is used to terminate a whole copy, the file copy in progress is aborted and the resulting partial file is closed. Use 〈CTRL〉/C to terminate whole copies.

In this example, a whole copy is started by temporarily placing the floppy disk contents into E-Disk storage. The wild card character indicates the name of the first file to be copied. Note the display for the last file copied, in case a second pass is needed.

ED0:=MF0:TEST.*/W 〈RETURN〉

In the second part of the example, the floppy disk is exchanged for the one which will contain the copied files. After noting the last file copied, insert a disk that has been formatted (see /F). This command line then copies the files to the disk from the E-Disk.

MF0:=ED0:/W 〈RETURN〉

If a second pass is needed, first zero the E-Disk (see /Z option below). Now use the * wild card to start the whole copy again, beginning at the first file that matches the wild card.

Notice the subtle difference in how the * wildcard character operates in a whole copy from its normal use in transferring, copying, deleting, protecting, and so forth. When used with the Whole Copy option, the * in the filename indicates that the copy is to begin with the first file that matches the extension.

For example, the command MF0:=ED0:*.BAS/W would not necessarily copy all the files with the BAS extension, but would begin the whole copy with the first file that had a BAS extension.

/X Exit

This command exits the File Utility program, and returns control to the shell.

/Z Zeroing a Fi!e Directory

The /Z command zeroes the directory of one or more file devices.

The result of the /Z option is similar to the /F option if the device is already formatted: no files are accessible when the operation is complete. However, formatting is more time-consuming, and is not necessary if the intent is merely to delete all files from a device.

Zeroing is not equivalent to formatting. The files remain after zeroing a directory, but they are not able to be accessed because there is no directory. After zeroing, the device retains the prior format.

Because zeroing deletes the directory of all files from the specified device, the program requests an affirmative before proceeding. Entries accepted as affirmative are YES, Y, yes, and y.

If a floppy disk or bubble memory are zeroed, the number of directory segments does not need to be specified, because the current structure remains intact.

This example zeroes the Electronic disk (ED0:)

```
ED0:/Z ‹RETURN›
```

This example zeroes the directory of the Electronic disk, and creates four segments:

```
ED0:/Z4 ‹RETURN›
```

/+ and /- Assigning Protection State

The /+ and /- commands change the protection state of a file. Directory listings indicate the current state inside brackets, and the system automatically assigns the unprotected state (-) to newly created files. A file must be unprotected before it can be deleted.

This example protects a file called SHDS.PAT that is currently recorded in the bubble memory:

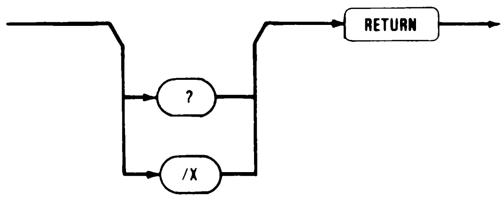
MB0:SHDS.PAT/+ ⟨RETURN⟩

This example uses a wild card to unprotect all files in the System device having a TMP extension. The Interactive switch ensures that confirmation is given before the unprotection.

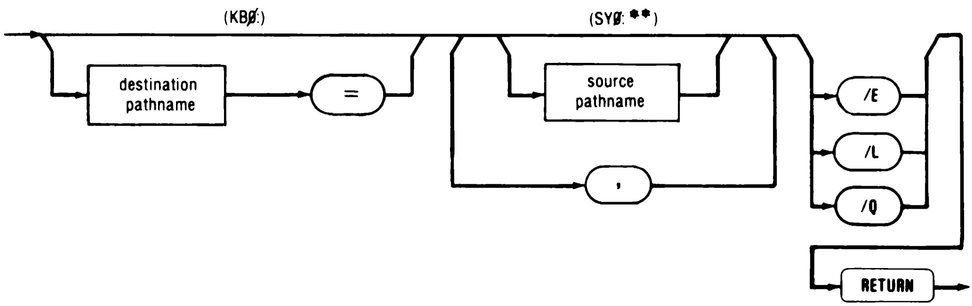
*.TMP/-I RETURN

Syntax Diagrams

Directly Executed Commands

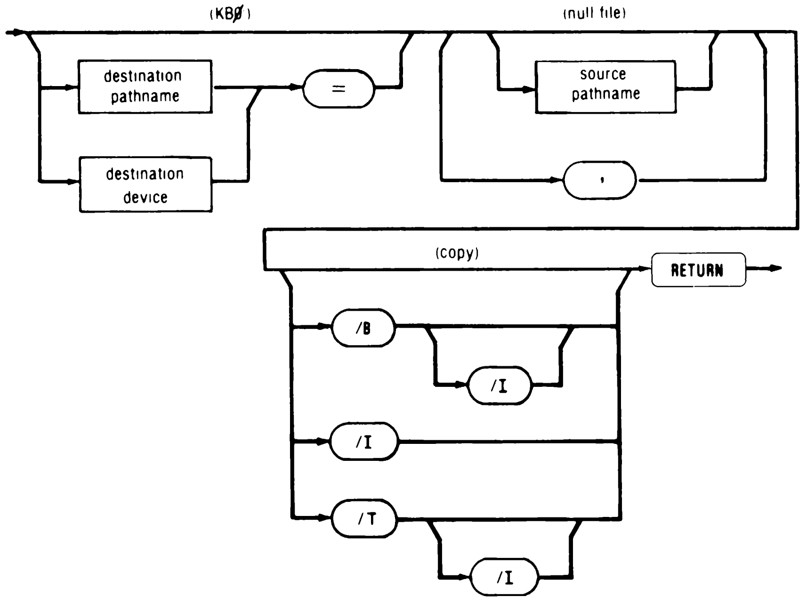


Directory Listing Commands



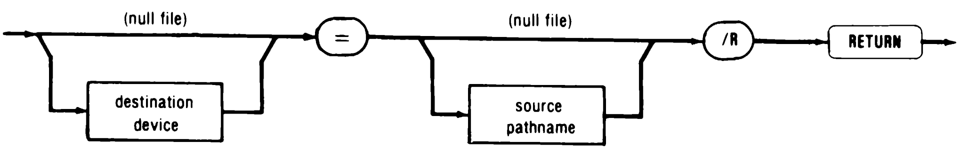
- A pathname can include device + filename + extension.
- All directory listings can use wildcards.
- If the device is not specified in the source pathname, the system device or the last device specified will be used.
- If no destination device is specified, the default is to KB0:.

File Transfer (Copy) Commands

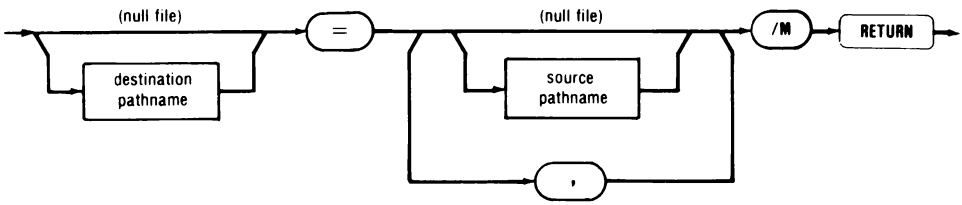


- A pathname can include device + filename + extension.
- All transfer options can use the Individual Switch and wildcards.
- If the device is not specified in the source pathname, the system device or the last device specified will be used.
- If no destination device is specified, the default is to KB0:.

File Rename Command

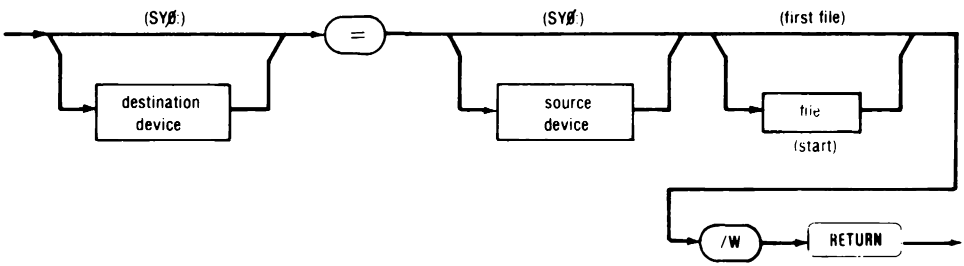


File Merge Command



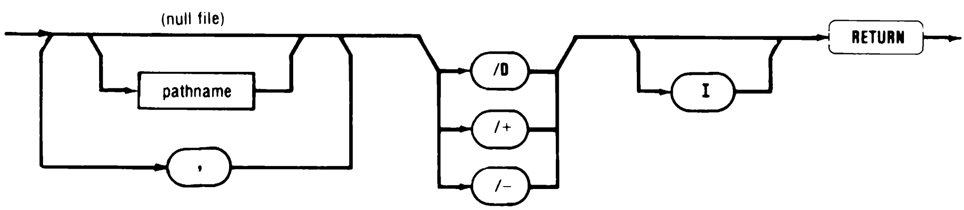
- Wild cards and the Interactive switch are not allowed.
- Each pathname must be separated by a comma.

Whole Copy Command



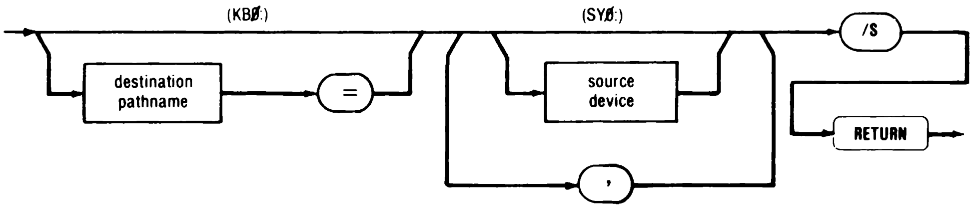
- Wild cards are allowed for the source filenames.

File Deletion and Protection Commands



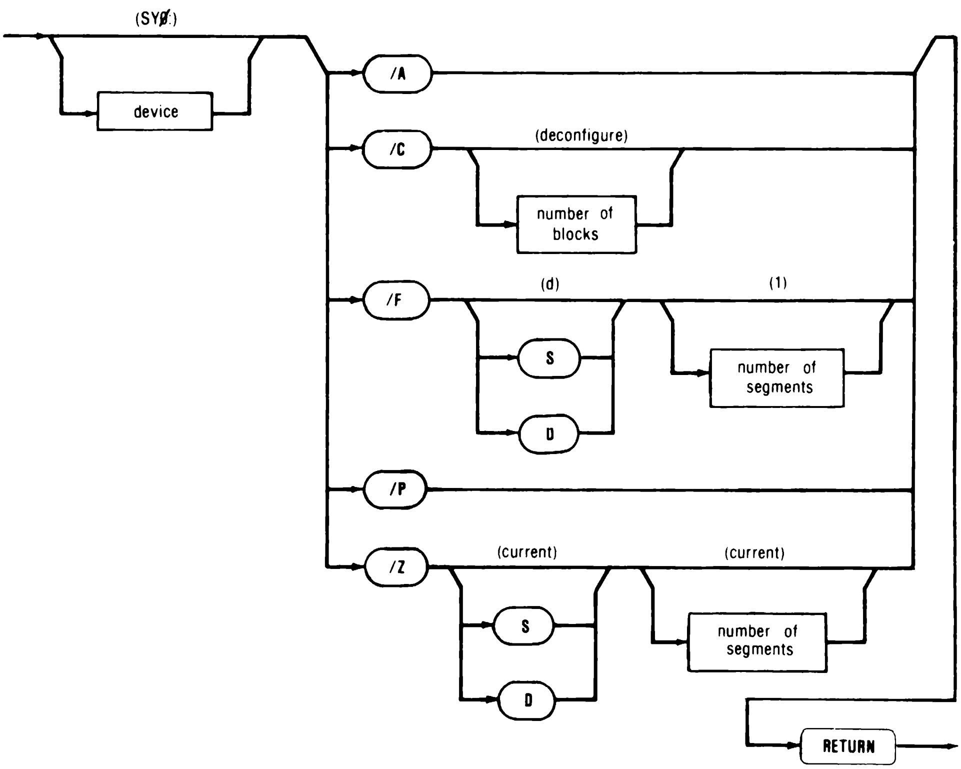
- If device is not specified in the pathname, the system device or the last device specified is used.
- The Interactive switch can be used to insure confirmation is requested before deleting or changing the protection state.
- Wild cards are allowed.

List Bad Blocks Command



- If device is not specified in the pathname, the system device or the last device specified is used.
- Wild cards are allowed.

Device Control Commands



System Messages

Messages from the system are a normal part of operation, and do not always signify that an error has been made, though they are generally referred to as "error messages". Here are the meanings of messages you might see from time to time when using the File Utility program. The messages are listed alphabetically:

MESSAGE	MEANING
`?Device error`	A non-recoverable error was detected during transfer to or from the floppy disk or Electronic disk. This may also occur when writing on an unformatted floppy or Electronic disk.
`?Device not ready`	The device 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.
`?Directory overflow`	Two many files exist for another to be copied or transferred to the device because the drectory is full. To recover, first backup all files, then use the /F option to reformat a disk with more segments.
`?File already exists`	A rename was attempted using a file name already in use.
`?File protected`	The specified file has a + protection state assigned to it. Use the /− command option to unprotect the file before deletion.
`?Help file not available`	A "?" was entered and the file FUP.HLP could not be located on the System Device.

`?Illegal file/device name`	A name in the command contains too many characters or contains characters other than letters, numbers, spaces, or "$" signs.
`?Illegal directory`	The directory on the device is faulty. If the device is a floppy disk, it is damaged and should be replaced. Backup all files first using the /T option.
`?Illegal option`	The command option selected was not recognized. This is usually caused by typing errors.
`?Illegal option for device`	A command was given that would be legal for some other device, but not for the one specified.
`?Incompatible format`	An option was specified for a device that does not accept the format. For example, attempting to zero the E-Disk as double-sided.
`?Input queue overflow`	The RS-232 port was receiving data, and some of it was lost. Use the Set RS-232 Utility program to slow down the baud rate or to enable the Stall Input/Output feature.
`?No end-of-file`	An ASCII source file was not terminated with ‹CTRL›/Z. Can also be caused by running out of storage space before the ‹CTRL›/Z is transferred.
`?No room on device`	A copy or merge operation was attempted, but the resulting file would not fit on the specified device.
`?No such 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.

?No such file	The file could not be found on the device specified. This is usually caused by misspelling, although the wrong device may have been specified.
?Not enough memory	An attempt was made to configure more E-Disk than is available in memory.
?Medium changed	The disk drive door was opened and the disk removed during a read or write operation.
?Number too large	The number specified in the command is too large. This message occurs if the number of directory segments specified is more than the device is able to contain.
?System error	This error should not occur under normal use. It indicates an error in the operating system or the File Utility program. Contact a Fluke Service Center and make an accurate report of the conditions at the time the error ocurred.
?Syntax error	The form of the command input does not match the requirements of a File Utility program command. This is normally caused by typing errors.
?Too many files	More than eight source files were specified. Either break the task into smaller parts, or use wild cards.
?Write protected	A write operation was attempted on the floppy disk, but it has a write protect tab.

THE TOUCH-COPY PROGRAM

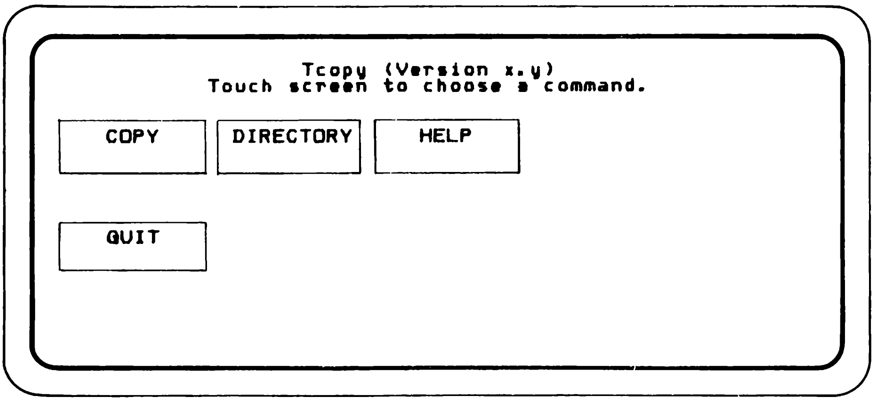
INTRODUCTION

The Touch-Copy Program is called TCOPY, and is provided on the System Disk with the filename TCOPY.FD2. TCOPY lets you use the Touch-Sensitive display to transfer files between any of the 1752A's file-structured devices. TCOPY is easy to learn and use because the program is totally menu-driven, and it provides help information about every command and option.

TCOPY does some of the things that are possible with the File Utility Program, but it is better suited than FUP for many special copying applications. For example, to use FUP to make a copy of an entire floppy disk first requires configuring memory space as E-disk, then using a whole copy command to transfer the files there, then swapping disks, and finally, using whole copy again to transfer the files from E-disk to the new floppy. On the other hand, TCOPY does not require a lot of typing, but presents menu selections to touch, considerably simplifying and speeding up the process.

Using TCOPY

From the FDOS prompt, type TCOPY ⟨RETURN⟩. The first menu is shown here:



TCOPY COMMANDS

TCOPY has only two commands, DIRECTORY and COPY. The directory command prints a listing of the files on a device. The copy command lets you copy various files from one file-structured device to another. Even if you only have one disk drive and no memory expansion options installed, you can still make a copy of a disk.

To choose an option, simply touch a box. To erase a command any time before STARTing it, touch the CANCEL box to return to the main menu. At this point, you can start entering a new command or touch QUIT to leave the program.

HELP

If you ever forget what the different commands and options do, use the help facility to get a description. When you touch the HELP box in the main menu, a list of topics appears. Select the topic by touching a box. Most topics also require the selection of a subtopic.

The DIRECTORY Command

Usually the first thing to do before copying a disk is to check the directory to make sure the proper disk is in the drive. The next few pages describe the DIRECTORY command and its options.

HOW TO LIST A DIRECTORY

With the main menu on the screen, touch the DIRECTORY box. A new menu appears, allowing you to choose the device which you want listed. The two boxes that indicate the system device are highlighted. If this is the device you want to select, just touch the START box. If it is not, first touch a device name box (top row), and then a device number (middle row).

When you press START, a list of the files on the selected device appears on the screen. The list includes the size of the files in blocks, their creation dates, and a plus sign if the file is protected. Touch the CONTINUE box to proceed to the next page or to return to the main menu from the last page.

DIRECTORY OPTIONS

A number of DIRECTORY options let you limit or change the list of files to be displayed. To use them, repeat the steps for displaying a directory listing, but touch OPTIONS instead of START after you choose the device. When an option has been entered, the program returns to the option menu. To begin the directory listing, touch START.

Each of the DIRECTORY options are described in the following paragraphs.

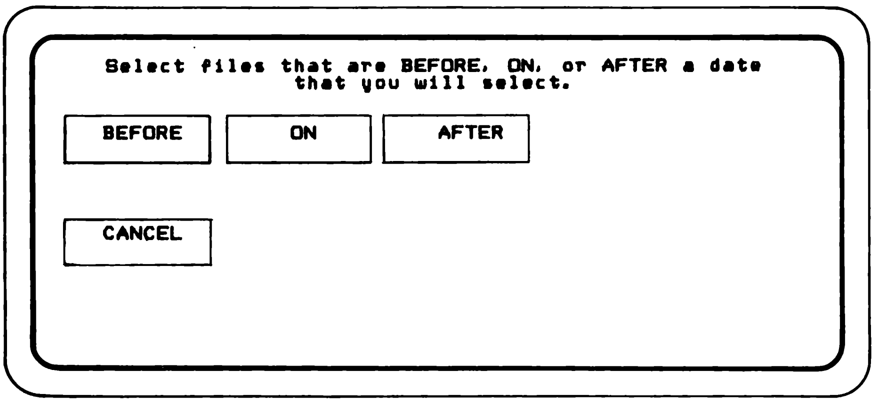
- Displaying a Short Directory List

  The SHORT option gives a more compact directory listing by omitting the file protection, size and creation date fields (like the FUP /Q option.)

- Selecting Files by Creation Date

  The DATE option lets you select only files with particular creation dates. When you touch the DATE box, a new menu displays. There are two steps in choosing a date:

  1. Select the type of limitation needed: whether you want files that were created BEFORE, ON or AFTER a date. Another menu appears to allow you to choose the date.



2. The next menu shows today's date in three boxes on the top row of the screen. To change the month, date, or year, touch one of the three boxes, then use the arrows to change the value in that box. You can then select a different part of the date to change. When the date you want is in the boxes, touch the START box.

NOTE

The selected date is not inclusive for the BEFORE and AFTER options. Files created on that date are not listed. To select files between two dates, combine the BEFORE and AFTER functions. To do that, touch AFTER, then select the earlier date. Now touch OPTIONS (instead of START), then DATE, and select BEFORE. Finally, select the later date and touch START.

- Selecting Files by Size

  Selecting files according to their size works almost the same way as selecting them by creation date. Touch the SIZE box, and then either SMALLER, LARGER or EXACT. A size in blocks appears in the top box on the screen. Use the arrows to change the size. When the desired size is highlighted, touch the START box.

- Selecting Files by File Type

  If you want to limit the listing to particular file types, touch the FILES box on the options menu. Then choose from one of the twelve file types on the screen and touch the START box.

SORTING THE DIRECTORY LIST

Normally the files in a directory list are unsorted. They appear on the screen in the order they occur on the disk. It is possible to sort the files in three ways, though. To use any of the SORT options, select the option by touching the appropriate box in the OPTIONS menu.

SORT NAME   Sorts the directory list alphabetically by filename.

SORT DATE   Sorts the directory list by creation date. The most recently created files are listed first.

SORT SIZE   Sorts the directory by file size. The largest files are listed first.

The COPY Command

The COPY command makes copies of some or all of the files on a device. It reads as many files as it has room for from one device and records them to another. This cycle repeats until all files are copied.

If the source and target devices are the same, and have removable media (such as a floppy disk) then TCOPY will prompt you for the source and target disks when they are needed. If the device does not have removable media, but the source and target devices are the same, TCOPY will report an error and the copy operation will not be performed.

COPYING AN ENTIRE DISK

You can copy an entire floppy disk even if you only have one disk drive. This type of copying may require changing the disk a few times. Follow these steps to copy an entire disk:

1. Make sure that you have the disk you want to copy (the source disk), and a blank, formatted disk (the target disk). See the FUP /F option for directions on how to format disks.

2. Start up TCOPY and when the main menu appears, touch the COPY box. The device menu appears, allowing you to select the device from which you want files copied. If mf0: isn't selected, touch MF and then 0:.

3. Touch CONTINUE. The device menu reappears to permit selection of the device to which the files will be copied. Again, make sure mf0: is selected.

4. Touch START. TCOPY now prompts you to insert the source disk. When you've done that, touch the box on the screen. The program begins reading files into memory. When memory is full (or when the entire disk is in memory), TCOPY prompts you to insert the blank target disk. This cycle repeats a few times until all the files on the source disk have been copied. When the entire copy is complete, the main menu displays again. Use the DIRECTORY command to list the files that were copied.

CAUTION

TCOPY provides a reasonable amount of protection against accidentally overwriting files. However, you may want to place a write-protect tab on the source disk to prevent loss of valuable programs and data in cases where the source and target disks must be swapped several times during a copy operation.

If TCOPY is aborted or if the disk is removed from the drive during a copy, some of the files may be lost or the directory of the target disk may be corrupted. Use the FUP /P option to remove any ⟨temp ent⟩ files from the directory. If the directory is unreadable, the disk must be reformatted using the FUP /F option.

COPY OPTIONS

The COPY command has a number of options. To display a menu that shows them, follow the instructions for the COPY command, but end with OPTIONS instead of START. Three of the options (SIZE, DATE and FILES) work exactly as they do for the DIRECTORY command, limiting the copy to a desired group of files.

- INDIVIDUAL

Use the INDIVIDUAL option if more precise selection is needed than the SIZE, DATE, or FILES options permit. When this option is used, TCOPY requests confirmation before copying each file.

PROTECTED FILES

TCOPY can record onto a disk that is not blank. Unprotected files on the target disk with the same name as a selected file on the source disk will be deleted without warning before the file is transferred.

- If a protected file is found on the target disk, TCOPY asks for permission before overwriting the file. The first time it asks, TCOPY also offers the choice of treating all protected files the same way.

- If you elect to overwrite the first protected file on the disk, the program asks if you want all protected files overwritten. If you respond YES, then TCOPY will overwrite the rest of the protected files without warning. If you answer NO, then TCOPY continues to ask for overwrite permission.

- If you decided not to overwrite the first protected file, then TCOPY asks if you want all protected files skipped. Answering YES causes TCOPY to skip over all files protected on the target disk. By responding NO, you direct TCOPY to continue asking for each protected file.

Additional Features

Besides the DIRECTORY and COPY commands and their options, the TCOPY program has some other features that can help simplify the task of copying files and disks. The next two topics describe these additional features.

COMBINING OPTIONS

Up to 15 options can be chosen for either a DIRECTORY or a COPY command. One example of this feature was given in the description of the DATE option. Here's another example: say you want a directory listing of all the binary machine language files larger than 10 blocks and created after August 30, 1984. Touch the boxes shown below.

```
DIRECTORY
  OPTIONS    FILES    *.FD2
  OPTIONS    SIZE     LARGER (scroll the number of blocks
                      to read 10)
  OPTIONS    DATE     AFTER (scroll the date to read Aug 30
                      1984)
START
```

If no files fit all these conditions, TCOPY displays the message "No files match pattern".

ENTERING TCOPY COMMANDS FROM THE KEYBOARD

Besides using the menus, there are two other ways to give commands to TCOPY.

The first way is to type the entire command from the FDOS prompt. For example, from FDOS, you can type:

TCOPY DIR *.ABC ⟨RETURN⟩

to get a directory listing of all files having the extension ABC. This method returns to FDOS after the TCOPY command has been executed.

The second method of using TCOPY without the menus is to use the -C option. This option puts you in TCOPY command mode. From FDOS, type:

TCOPY -C ⟨RETURN⟩

The TCOPY prompt appears. Now any TCOPY commands can be typed directly. This feature is convenient once you become familiar with the various capabilities of TCOPY, because it can be quicker than working through the various menus.

As an example, if MF0: is the system device and you type:

COPY FUP.FD2 ⟨RETURN⟩

from the TCOPY prompt, you will copy the File Utility Program from one disk to another. The screen still prompts for the source and target disks, just as the menus do.

To use a sorting option with commands entered from FDOS, or from the command mode, the word "sort" must appear, followed by the type of sort. For example, to list all the files on the E-Disk that were created after October 3, 1984, and sort the list alphabetically, use:

DIR ED0: AFTER OCT 3 1984 SORT NAME ⟨RETURN⟩

All words, except for months (which must be at least 3 letters) can be shortened to 2 letters and must be separated from each other by a space.

To exit TCOPY'S command line mode, type:

QUIT ⟨RETURN⟩ or EXIT ⟨RETURN⟩

USING TCOPY IN A COMMAND FILE

When TCOPY is called from a command file, it comes up in command mode by default. This allows subsequent lines of the command file to provide input to TCOPY as if they were typed from the keyboard.

If you want to call TCOPY from a command file and use the menus, you must use the -M option. The line "TCOPY -M" will accomplish this.

NOTE

If TCOPY is called as the last command in a command file, it must be followed by a blank line. Otherwise TCOPY will start up in menu mode if no option is specified.

Error Messages

The TCOPY program returns the same error messages as FUP. For a complete description of these messages, refer to "System Messages" under the File Utility Program earlier in this section.

In addition, the following messages are returned by the TCOPY program:

MESSAGE	MEANING
Can't find help on that topic.	The help file has been changed or overwritten, and the help for that particular subject is missing.
Can't open help file.	The help file (TCOPY.HLP) is missing.
Extra command.	Only one command (DIRECTORY or COPY) can be given at a time.
Illegal date.	An illegal date was entered, or a year before 1972 or after 2003 was specified.
No files match pattern.	No files were found on the source disk that matched the patterns and/or limitations given.
Source and target devices are the same.	If the target device does not have removable media (i.e. floppy disk), then it must not be the same device as the source.
Temporary files on device.	Can only be removed by packing the device. Only a warning. Does not halt execution of the command.

Too many source devices listed.	Only one source device may be listed.
Too many words.	Only 15 words may be entered from command mode.
Unknown option.	TCOPY was invoked with an unknown argument.

CONCLUSION

This section has described the 1752A's devices and files in some detail and explained how to use the File Utility and TCOPY programs to manipulate files. The devices can be thought of as names for the various hardware parts of the system. Files are programs and data.

The next section, Communications, details the way that information can be transferred between the 1752A and other devices by means of two industry-standard connection methods: the IEEE-488 Instrumentation Standard, and the RS-232-C Digital Communications Interface.

Section 5
Communications

CONTENTS

Introduction 5-2
The IEEE-488 Bus 5-3
Bus Functions 5-4
Interface 5-5
Bus Operating Modes 5-5
Command Mode 5-5
Data Mode 5-7
Three-Wire Handshake 5-7
A Typical Instrumentation System 5-7
Sequence 5-8
Multiple Controller Systems 5-9
IEEE-488 Communications Under Program Control 5-11
Example Commands from the BASIC Language 5-12
Sample BASIC Program 5-13
For More Information 5-15
Serial Communications 5-16
Set Utility Program 5-16
Using the Set Utility Program 5-17
The Help Command 5-18
Command Structure 5-19
Syntax Diagram 5-20
Device Selection 5-21
Setting Parameters 5-21
Single Command Line Entry 5-25
Error Messages 5-26
Serial Communications Under Program Control 5-27
Sample BASIC Program 5-28
Conclusion 5-31

INTRODUCTION

The 1752A Data Acquisition System communicates to external devices in two ways: by way of the IEEE-488 bus, and by way of the EIA RS-232-C Data Communications Interface.

These two standards were developed to serve two different purposes: the IEEE-488 as a standard connection between measurement instruments, and the RS-232-C as a standard connection for serial data communications.

THE IEEE-488 BUS

In 1975, the Institute of Electrical and Electronic Engineers (IEEE) published a "Standard Digital Interface for Programmable Instrumentation Systems". This standard was revised in 1978, and a supplement was published in 1980. The IEEE-488 standard has gained acceptance throughout the instrumentation industry because it permits a wide variety of measurement equipment to be connected easily to form a programmable instrumentation system. The 1752A implements the most recent version, including the 1980 supplement.

The IEEE-488 standard describes a bus architecture and defines the timing and handshaking that occurs on the bus. Devices connected to the bus may be *talkers, listeners, or controllers.*

The 1752A is able to control up to 14 instruments directly from the single standard IEEE-488 connector. An additional Interface module can be added to allow the 1752A to control more instruments.

The next few pages describe how the interface operates. Much of this discussion is theoretical, and has been included here to help first time users visualize an instrumentation system. Though it may appear that the operation of an IEEE-488 system is a complicated matter, in fact it is quite easy to use, and most of the details of bus operation are transparent to the user.

In an IEEE-488 system, the 1752A is referred to as the instrument controller.

Bus Functions

The instrument controller (the 1752A) establishes the role of each of the connected instruments by sending commands to them and setting up the correct communication channels. Each piece of equipment recognizes its own address, which is set into it by configuration switches when the system is assembled. Each connected instrument can then respond to polling and receive or send data. Depending on its role in the system, each instrument can also perform these functions:

- Handshaking to establish and confirm the connection
- Single address talking or listening
- Request service to notify the controller that a function is complete
- Respond to poll to answer the controller's request for status
- Clear to return to a default state
- Trigger to respond to the controller's command to perform a function

The controller can perform these functions:

- Command devices to listen, talk, or perform a function
- Trigger devices to perform a pre-programmed function
- Clear devices to an initial state (defined by the device)
- Poll devices for their status serially or in parallel (one device at a time, or all at once)
- Command devices to abort current operations
- Command devices to enter the remote mode of operation
- Pass control to another controller

Interface

There are 16 signal lines on the IEEE-488 bus; all are active low TTL levels. The lines are divided into three categories:

1. Eight data lines

2. Three handshake lines

3. Five bus management lines

For your reference, Appendix C has a pinout diagram of the standard interface connector and a description of each of the lines.

Bus Operating Modes

The bus operates in either command or data mode. A controller uses the command mode to control the various instruments connected to the bus. A talker or a controller uses the data mode to transfer information or device dependent commands on the Data I/O lines. The three handshaking signals officiate transfers on the bus.

Command Mode

The controller places the system into Command Mode by sending an attention signal. All devices on the bus must then interpret the data byte as a command message. Only a controller may issue commands.

There are four types of commands:

1. Addressed Commands (all devices are addressed to listen).
   These are used to control a selected group of devices. The command is preceded by a device address, because only those devices previously commanded to listen must respond.

2. Universal Commands (all devices).
   These commands are used to control all system devices. They do not need to be preceded by an address because all devices that are able to must respond to them.

3. Addresses (all devices).
   The controller uses these commands to designate devices as talkers or listeners.

4. Secondary Commands (all devices are enabled by a primary address or command).
   These are the second byte of a two byte address, and are used by the controller to implement "extended" talk and listen functions. Secondary commands are used to send a second address if the primary address has been accepted as part of the address command.

Each type of command has a specific function in the activity of the bus, and they are all designated by three letter mnemonics. All of these bus functions are implemented in each of the programming languages available for the 1752A. In some cases, a single command performs more than one function; for example, the Fluke Enhanced BASIC command TRIG (Trigger) addresses a set of instruments as listeners and then triggers them.

Appendix C contains a complete listing of the Command Messages. Refer to the appropriate programming language manual for information about how to implement each of the commands in a given language.

Data Mode

The controller places the system into the data mode by setting the attention line false. In this condition, all devices treat the information on the bus as data. This data can originate from either a talker or the controller. Data can flow from device to device on the bus (talker to listener) in any mutually understood code or format, or the 1752A can act as an interpreter accepting data from the talker and sending it out to a listener.

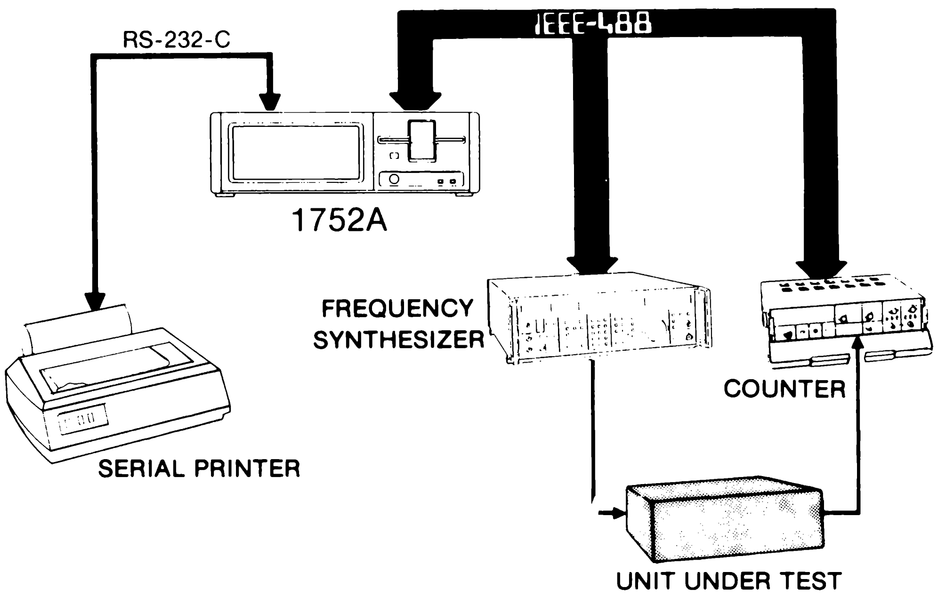
Three-Wire Handshake

All IEEE-488 bus devices use a three-wire handshake to manage the exchange of data. The signals are:

DAV- (Data Valid)
NRFD- (Not Ready For Data)
NDAC- (Not Data Accepted)

A Typical Instrumentation System

An example system is shown here to illustrate how the IEEE-488 interface can handle a variety of tasks. In Section 7, the same sample setup shows how to automate these functions.



In this example system, a sequence of events describes how a specific measurement task is accomplished. In general terms, the controller (the 1752A) programs the instruments and initiates the measurements. The resulting data is returned to the controller, which then routes it to the serial printer for printout.

Sequence

1. The controller initializes the interface devices.
2. The controller commands all devices to set their internal conditions to a predefined state by sending a device clear message.
3. The controller sends the listen address and program data to the frequency synthesizer. The program data tells the frequency synthesizer the output frequency and level.
4. The controller sends the unlisten command to the synthesizer, then sends the listen address and program data for the frequency counter. (Function and range, for example.)
5. The controller sends a program code to trigger the measurement.
6. The controller sends the unlisten command, addresses itself to listen, then sends the talk address of the measurement device.
7. When it completes its internal measurement cycle, the frequency meter sends a request for service to the controller. When the frequency meter's request is recognized, it sends (talks) the measurement data to the addressed listener, the controller.
8. Under program control, the controller gathers the measurement data and sends it via the RS-232 port to the printer (first addressing the printer as a listener, as before).

This example shows how a typical system is connected and describes the way the controller directs the tests. The program required to actually do these steps depends on the programming language used. Each programming language manual includes examples such as this one to assist the programmer. In Section 7, Automating System Functions, a BASIC language program is developed that will show how a test program might be written to test this system.

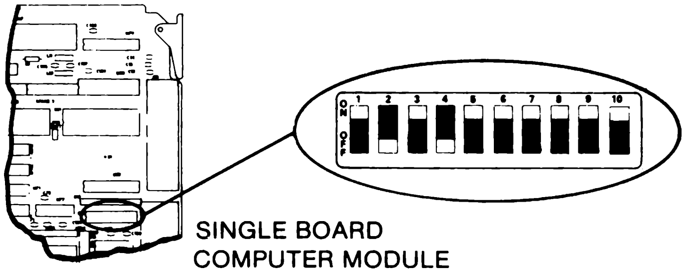
The basic system can be expanded by adding options to the controller so that many more instruments can be connected. These options provide a great deal of flexibility to the system designer. For example, the RS-232 ports could be used to transmit the data via a modem to a larger computer for analysis. Because of this flexibility, it is recommended that you consult your Fluke representative in the early design stages so the final system will perform efficiently in your application.

Multiple Controller Systems

At power up, the 1752A can be designated either as the "system controller" or as an "idle controller" on an IEEE-488 port. There can only be one system controller on a port, and the controller is the only device connected that is able to drive the control lines IFC (interface clear) and REN (remote enable). Likewise, there is only one "controller in charge" on a port at a time and it is the only device that can send interface messages by setting the ATN (attention) line true. The system controller is the controller in charge when it is powered up.

The controller in charge can pass control to an idle controller. When this happens, the idle controller becomes the controller in charge and the former controller in charge becomes an idle controller. If the system controller sets the IFC line true, it gains control on the port and becomes the controller in charge.

A switch on the Single Board Computer module (slot 7) designates the power-up condition of the 1752A either as the system controller or as an idle controller. The other switches of SW1 set the 1752A default baud rate (the rate used unless changed by the Set Utility program), and the 1752A's IEEE-488 bus address. Refer to the following drawing to set the switch.



SWITCH 1

1	2	3	4	5	6	7	8	9	10	
—	S2	S1	S0	SC		A4	A3	A2	A1	
x										unused
										IEEE-488 ADDRESS
						0	0	0	0	0
						0	0	0	1	1
						0	0	1	0	2
						0	0	1	1	3
						0	1	0	0	4
						0	1	0	1	5
						0	1	1	0	6
						0	1	1	1	7
						1	0	0	0	8
						1	0	0	1	9
						1	0	1	0	10
						1	0	1	1	11
						1	1	0	0	12
						1	1	0	1	13
						1	1	1	0	14
						1	1	1	1	15
										IEEE-488 CONTROLLER
				0	0					System Controller
				1	1					Idle Controller
										BAUD RATE
	0	0	0							110
	0	0	1							300
	0	1	0							600
	0	1	1							1200
	1	0	0							2400
	1	0	1							4800
	1	1	0							9600
	1	1	1							19200

IEEE-488 COMMUNICATIONS UNDER PROGRAM CONTROL

Each programming language available for the 1752A includes specific commands to handle the IEEE-488 communications. The standard was developed to permit the easy connection of an instrumentation system, but two important rules should always be kept in mind as you begin programming:

1. There may be any number of listeners at a given time.

2. There may only be one talker at a time.

There are other constraints, but these mostly depend on the capabilities of the connected instruments. An example is the Fluke 8502A Digital Voltmeter, which must be allowed a three-second wait between the time it is reset and the time it is programmed. Be sure to adhere to the requirements of each particular instrument to simplify your programming task.

Example Commands from the BASIC Language

Here is a synopsis of the commands used in Fluke BASIC to control communications over the IEEE-488 bus. For complete definitions and requirements, see the BASIC Programming Manual.

`INIT`	Initializes the bus.
`CLEAR @ n`	Addresses the specified device number as a listener, and issues a selective device clear.
`WAIT`	Followed by a number, suspends program execution for the length of time indicated (milliseconds).
`INPUT @ n`	A command string (in quotes) following this command addresses the instrument as a listener, and programs it.
`PRINT @ n`	A variable follows this command to address the instrument as a talker, and return the measurement data to the controller.

Sample BASIC Program

This sample program shows how the most important bus commands would be included in a BASIC language program to take a reading from the Fluke 8502A Digital Voltmeter. Some features of the program are:

- Between resetting the meter and sending it program data, there is a 5-second wait to allow the meter to stabilize.
- Ten readings are taken to insure accuracy of the returned measurement data.
- A 3-second wait is inserted between the last reading and the display of measurement data.

```
10   !     Program for Fluke 8502A Digital Voltmeter
20   !
30    DIM R (9)                     ! dimension array
40    INIT PORT 0                   ! initialize the bus
50    CLEAR @ 2                     ! clear instrument 2 (DVM)
60    WAIT 5000                     ! let DVM settle
70    PRINT @2,  "VR2T0"            ! program DVM
80    FOR I% = 0% TO 9%             ! set up loop for 10 readings
90         PRINT @ 2,  "?"          ! trigger DVM
100        INPUT @ 2,  R (I%)       ! get reading, send to array
110   NEXT I%                       ! go back to 60 for next reading
120   WAIT 3000                     ! let bus settle
130   FOR I% = 0% TO 9%             ! loop to display readings
140        PRINT I% ; R (I%)        ! print value of I
150   NEXT I%                       ! return for next value
160   END
```

Here is a line-by-line explanation of the sample program:

10 Program identification.

20 Blank line for readability.

30 Dimensions an array called R to hold up to ten readings (0-9).

40 Initializes Port 0, the built-in IEEE-488 bus.

50 Clears and addresses as a listener the DVM whose address is 2. (Set up by switches in the instrument.)

60 Wait for 5 seconds to allow the DVM enough time to respond to the clear signal, and to set up internally as a listener.

70 Programs the instrument with the command "VR2T0", which this particular instrument reads to mean, 'use 10 volts DC scale, and take single readings synchronously with the line (60 Hz)'.

80 Sets up an integer counter (I) within a 'FOR-NEXT' loop that will take ten readings, and return them to the Conroller.

90 Addresses 2 as a listener, and sends a "?" which triggers the measurement.

100 Addresses 2 as a talker, and puts the returned reading into the location specified by I% within array 'R'.

110 Program returns to line 80, which increments the value of I%, so that as the FOR-NEXT loop is repeated, the next reading goes to the next available location in the array R. After ten passes through the loop, the program continues at line 120.

120 All measurements are complete, so this program step waits 3 seconds for the bus to settle.

130 Sets up another FOR-NEXT loop to display all ten readings.

140 Prints each value of I% within the array R.

150 Sends program back to line 130 to increment I%, and display the next reading.

In actual practice, this program could be greatly simplified. It is shown here for illustration purposes only, although it could be used as is. Some of the simplifications might be:

- Trigger the DVM ten times, but then send a command to have it average the readings (many programmable instruments include such mathematical abilities). The DVM would then return the average, rather than ten readings.

- If the meter is unable to perform the mathematics, have it return all ten readings, but finish this subroutine with a branch to an averaging subroutine, and display only the average.

For More Information

System designers and programmers who are unfamiliar with the IEEE-488 standard should obtain a copy of Fluke Application Bulletin 36 (*IEEE Standard 488-1978 Digital Interface for Programmable Instrumentation*), and Technical Bulletin C0076 (*Troubleshooting Information for IEEE-488 Systems*). These publications provide the background needed to set up an IEEE-488 system. Appendix C of this manual provides useful reference material covering the interface connector, handshaking protocol, commands, and message formats.

For an in-depth study of the IEEE-488 standard, a copy can be obtained by writing to the Institute of Electrical and Electronic Engineers, Inc., 345 East 47th Street, New York, NY, 10017.

SERIAL COMMUNICATIONS

The RS-232 ports are connection points for devices that use the Electronic Industy Association's RS-232C Data Communications Interface Standard. Since it was first published, the RS-232 standard has gained wide acceptance among manufacturers because it allows various brands of equipment to use serial data communications to pass information.

The standard describes the physical connector, the signals on each pin of the connector, timing requirements, and the voltage levels of the signals.

The standard allows variations to accommodate different applications. Therefore, the 1752A software includes a program called the Set Utility program that permits changing the values of the parameters at the port.

Set Utility Program

The purpose of the Set Utility program is to configure the 1752A to enable it to communicate with virtually any other piece of equipment that uses the RS-232-C standard. The port parameters are set to default values when the Operating System is loaded, and some applications will not require changing the defaults.

These are the port characteristics that can be changed:

- Baud Rate
- Number of Data Bits
- Number of Stop Bits
- Parity
- End of Line and End of File Terminators
- Stall Input/Output Enable/Disable
- Time Out Value

Using the Set Utility Program

Here is how to use the Set Utility program:

1. From the FDOS⟩ prompt, enter:

   SET ⟨RETURN⟩

2. The screen will display the prompt:

```
FDOS> SET
Set Version x.y

SET>
```

3. Specify the port to be changed, and use the command chart on the next page to change the desired parameter(s).
4. If you would like to see the current parameters of a port, use the List command. As with all Set Utility program commands, specify the device first. Once the device has been specified, the Set Utility program will assume that device until another is specified.

   KB1: LIST ⟨RETURN⟩ -or- KB1: LI ⟨RETURN⟩

   This example illustrates the default parameters:

```
SET> KB1:LI
Device          KB1:

Baud Rate       9600
Data Bits       8
Parity          Even
Stop Bits       1
End of Line     10
End of File     26
Stall Input     disabled
Stall Output    enabled
Time Out        0
```

5. To exit the Set Utility program, type:

   EXIT ⟨RETURN⟩ -or- EX ⟨RETURN⟩ -or- ⟨CTRL⟩/Z

The Help Command

A help command lists all the available parameter selections. Type ?⟨RETURN⟩ to see this display:

```
Command   Argument                                         Function             Example
-------------------------------------------------------------------------------------------
BR        75,110,134.5,150,300,600,1200,1800               Set Baud Rate        BR 2400
          2000,2400,3600,4800,7200,9600,19200
DB        5,6,7,8                                          Data Bits            DB 8
KBn:                                                       Select Device        KB0:
EOF       <0 through 255> (decimal) or '<char>'            End of File          EOF 26
EOL       <0 through 255> (decimal) or '<char>'            End of Line          EOL 10
EX or EXIT                                                 Exit to FDOS         EX
LI or LIST                                                 List Configuration   LI
PB        EVEN, E, ODD, O, NONE, N                         Parity               PB NONE
SB        1, 1.5, 2                                        Stop Bits            SB 1
SI        ENABLE, E, DISABLE, D                            Stall Input          SI E
SO        ENABLE, E, DISABLE, D                            Stall Output         SO E
TO        <0 through 255> (seconds)                        Time Out             TO 5
```

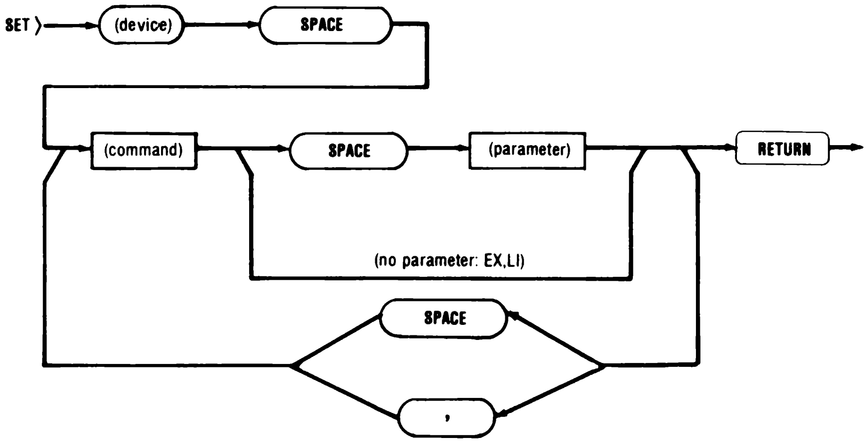
Command Structure

The Set Utility program features a straightforward and flexible command structure. After selecting a device, the current settings can be listed and changed in any order.

- Commands can be entered singly or combined into a multiple command line.
- Both upper- and lower-case entries are accepted.
- All commands are terminated by <RETURN>.
- All commands can be automated by command files.
- Parameters controlled by the program can be set independently for each serial port.
- Plain language messages prevent setting parameters improperly (out of allowable range, incorrect syntax, unspecified device, etc). A table of all messages is given on page 5-26.

Syntax Diagram

The syntax diagram below illustrates the proper syntax for all of the commands available in the Set Utility program.



- Any number of parameters can be specified for a port, as long as each command is separated by a comma or a space.

- The command line cannot exceed 80 characters (the capacity of one line on the screen). If more are needed, use two separate command lines.

- A single command line can be used to set parameters for more than one device as long as a comma precedes each device name after the first one.

- Any number of commands are allowed on the same command line as long as each is separated by a comma or space.

Device Selection

A port device must be selected before the Set Utility program will accept other commands. Device selection can be made a part of the single command line. Once the device is selected, subsequent commands affect that device until another is specified.

If a command is entered before specifying a device, the "no device specified" message is displayed.

Attempting to specify a device other than a serial device (KB0: - KB2:, GP0: - GP1:, SP0 - SP9:) will cause the "illegal device" message to be displayed.

All command inputs except baud rate are ignored for KB0: (the console device).

If the KB1: port is already in use for an external monitor, its parameters's cannot be changed, but will return the "illegal device" message.

Setting Parameters

Baud Rate (BR)

Baud rate is the speed of information transfer. This command changes the baud rate of the selected port to that specified by the command argument.

In the following example, the baud rate of the optional Serial Port 2 is set to 2400 baud:

KB2: BR 2400 ⟨RETURN⟩

NOTE
The baud rate of the keyboard and display is 19.2 Kilobaud. Setting KB0: to any other baud rate immediately disables both the keyboard and the display. This condition can only be remedied by pressing RESTART.

Character Length (DB)

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

When character length is set to a value shorter than the actual data's character length, the lower data bits (least significant) are used.

If the character length is set to a value higher than the actual data, the remaining (most significant) bits are set to zero.

The following example selects Serial Port 1, and sets the character length to 7 bits:

KB1: DB 7 ⟨RETURN⟩

Parity Bit (PB)

Parity is a method of error detection that adds an extra bit 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 parity command defines a parity bit to be generated and checked for the selected port.

The command argument NONE or N eliminates the generation and checking of the parity bit.

During input, parity is checked as defined by the Set Utility for each character. If an error is detected, it is identified to the operating system as a device error. This error can only be cleared by closing the channel associated with the serial port.

During output, parity is generated as defined by the Set Utility program, and appended to each character.

In this example Serial Port 2 is selected, and set up to generate and check for even parity:

KB2: PB EVEN ⟨RETURN⟩

Stop Bit (SB)

The stop bit command defines the number of bit-cell time periods between characters transmitted to external equipment that requires additional settling or synchronization time. This command does not affect incoming data.

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

KB1: SB 1.5 <RETURN>

Stall Characters (SI, SO)

Stall Input only affects data being received at the serial port. When it is enabled, the 1752A sends X-OFF (decimal 19) when its input buffer is 3/4 full. When the buffer has been emptied to the 1/4 full point, the 1752A transmits X-ON (decimal 17).

Stall Output only affects data being sent from the 1752A. When Stall Output is enabled, receipt of an X-OFF character causes the 1752A to suspend transmission until it receives X-ON.

Stall Output only affects data being sent from the Controller. When Stall Output is enabled, receipt of an X-OFF character causes the Controller to suspend transmission until it receives X-ON.

This sample shows how both Stall Input and Stall Output would be enabled for Serial Port 1:

KB1: SI E, SO E <RETURN>

Length of Time Out (TO)

The Time Out parameter affects both input and output data. During input, if no data is received within the length of time specified by the parameter, the 1752A resumes processing, rather than waiting indefinitely.

When the 1752A sends data through a serial port, it does so by filling a buffer, and creating a path between the buffer and the port. If the buffer becomes full, the 1752A waits for the time specified by the Time Out parameter. If the buffer has not started to empty by then, the attempted data transfer is abandoned.

This parameter ensures that the 1752A will not wait indefinitely for a data transfer in either direction. Instead, the 1752A abandons the attempt and continues processing.

Time Out is specified in seconds (0 to 255). The longest time out, 255 seconds, equals 4 minutes, 15 seconds. A time out value of 0 means the 1752A will not time out.

NOTE

The Time Out value can also be used to change this parameter at the IEEE-488 ports. For this usage, specify the GPx: device, and specify the parameter as usual. This is the only IEEE-488 parameter that can be modified using this utility program.

In this example, the Time Out parameter for KB1: is set to 15 seconds.

KB1: TO 15 ⟨RETURN⟩

Terminator Characters (EOL, EOF)

The terminator commands EOL and EOF define characters that will be used to identify the end of a line or file. When received with incoming data, a terminator generates an interrupt if enabled by the user program.

EOL only affects data input: All Carriage Return and Line Feed characters are deleted, but when the terminator character is received, a Carriage Return, Line Feed sequence is appended. (The system does not add a second Carriage Return or Line Feed if that character is the terminator.) The resulting data is in the internal format of 1752A data.

EOF affects both input and output data:

- During input, each file terminator defined by Set is converted to 〈CTRL〉/Z (ASCII 26).
- During output, each 〈CTRL〉/Z character is converted to the file terminator defined by the Set Utility.

This example shows a command line that defines a question mark as line terminator, and the ASCII character 4 (EOT code or 〈CNTRL〉/D) as a file terminator for a port that was previously defined:

```
EOL '?' EOF 4 〈RETURN〉
```

Single Command Line Entry

During experimentation, you normally change the parameters one at a time in order to find the correct combination for the device that is to be communicating with the 1752A. During the development of working software, however, it becomes increasingly important to begin thinking about how to speed things up. Since any keyboard entries can become part of a command file, it is efficient to make one entry in the command file set the parameters needed at the RS-232-C port.

This example illustrates how all of the above commands could be combined into a single command line to change the parameters at a serial port:

```
KB1: BR 2400 DB 7 PB E SB 1.5 EOL '?' EOF 4 SI E SO E TO 15 EX
〈RETURN〉
```

Error Messages

Messages from the system are a normal part of operation, and do not always signify that an error has been made, though they are generally referred to as "error messages". Here are the meanings of messages you might see from time to time when using the Set Utility program:

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.
?No device specified	A command was entered before specifying a device.
?Unknown command	A command was entered that was not recognized.
?Attribute cannot be changed	A parameter was specified that cannot be changed for that port.

Serial Communications Under Program Control

To automatically set the parameters of the serial ports, a command file must be written that uses Set Utility program commands. The parameters cannot be modified by programs written in high level languages (BASIC, Fortran). Once the port parameters are established by the Command file, however, programs written in any language can get information into and out of the port.

Keep in mind that the command file must first enter the Set Utility program, and after parameters are changed, must exit it. Here is a portion of a command file that does just that:

```
SET
KB1: BR 2400,DB 7,PB E,SB 1.5,SI E,SO E,TO 15
EXIT
```

The following table illustrates the commands used in the BASIC and FORTRAN languages to send and receive information via the serial port. Notice that both languages include commands to first open the channel, then either to send or recieve data through it. For more information, consult the manual that covers the programming language you are using.

BASIC

To send:

```
OPEN "KB1:" AS NEW FILE 1
PRINT #1, "TESTING"
```

To receive:

```
OPEN KB1: AS FILE 1
INPUT #1, A$
```

FORTRAN

To send:

```
    CALL OPEN (2, "KB1:", 4, 0, IERR)
    WRITE (2, 10) IERR
10  FORMAT ( 'ERROR CODE: ',I5)
```

To receive:

```
    READ (2,20) A
20  FORMAT (F10.4)
    CALL CLOSE (2, IERR)
```

Sample BASIC Program

This sample program shows how these important commands would be included in a BASIC language program that reads information from a floppy disk in device MF0:, and sends it out to a serial impact printer, like the Fluke model 1776A. Some features of the program are:

- Prompts the operator for the name of the file to be printed.
- Handles the most common error: specifying a file that doesn't exist.
- Regardless of the size of the file, outputs the entire file, and stops when printing is complete.

```
10   !  Program to print a specified file
20   !
30   @DIR                                    !   list files
40   PRINT
50   PRINT 'ENTER FILENAME'                  !  display prompt
60   INPUT A$                                !  filename is variable A$
70   IF A$ = "" THEN GOTO 330                !  graceful exit
80   ON ERROR GOTO 220                       !  error handler
90   CLOSE 1                                 !  make sure channel is closed
100  OPEN A$ AS FILE 1                       !  now open it.
110  CLOSE 2                                 !  make sure channel is closed
120  OPEN 'KB1:' AS NEW FILE 2               !  now open it. KB1: is printer
130  PRINT #2,CHR$(12%)                      !  send form feed to printer
140  INPUT LINE #1, A$                       !  put one line of file into A$
150  PRINT #2, A$                            !  output the line to printer
160  GOTO 140                                !  repeat until done
170  !
180  !            * Error Handler *
190  !     ERR (system variable) = last error
200  !
210  !        - Error 305  File Not Found -
220  IF ERR = 305 THEN RESUME 230 ELSE 270   !  retry if file not found
230  PRINT 'FILE NOT FOUND - TRY AGAIN'
240  WAIT 1000 \ GOTO 30                     !  pause 1 sec, start again
250  !
260  !         - Error 307  End of File -
270  IF ERR = 307 THEN RESUME 280 ELSE RESUME 310        !  halt if EOF
280  PRINT 'TRANSFER COMPLETE'
290  CLOSE 1, 2 \ END                        !  close channels, halt
300  !
310  !            - All Other Errors -
320  PRINT 'ERROR - CANNOT TRANSFER FILE'    !  halt for all other errors
330  PRINT 'RETURNING TO BASIC'
340  CLOSE 1, 2                              !  close channels
350  END                                     !  halt
```

Leaving out the comment lines, here is a line-by-line explanation of how this program operates:

30 The program begins by doing a quick directory listing to display the files that are available for printing.

40 Puts a blank line on the screen for readability.

50 Displays the prompt message.

60 Assigns the response to the prompt as string variable A.

70 If there is a null string (no input) terminated by a ⟨RETURN⟩, this line provides a graceful exit. This is used, for example, if the directory listing indicates the file you want to print is not on this disk.

80 Beginning of the error handler. Any error encountered sends the program to line 220, where the error type is discovered, and various exits are provided depending on the error type.

90 The CLOSE command is insurance that a previously opened channel is closed prior to reopening. To OPEN an already open channel is an error; to CLOSE one that hasn't been opened is not. After this 'insurance' command, the following line opens the file designated as variable A as file 1, for input.

100 Opens the file.

110 Again, a command to insure the channel is closed before opening.

120 Serial port KB1: is opened as an output.

130 The printer (File 2) is sent a Top Of Form command, so that printing will start at the beginning of a sheet.

140 Each line of the file is input and sent to A$.

150 Each line is sent out to #2, the printer.

160 The return to line 140 gets the next line, so it can be PRINTed by line 150.

220 The error handler begins here. Error 305 is returned if there is no file with the name of the one specified as A$. If the error is 305, then a message will be displayed at line 230, and allow a retry.

230 Prints the message that the file was not found.

240 After a one second wait, goes back to line 30 to display the directory again.

260 The error was not 305, so now it is checked to see if it was 307, the End of File. This is the exit when the entire file has been printed. All other errors fall through to ending routine that closes the channels and stops.

270 If the EOF was the error, prints the message that the transfer is complete.

280 Closes the previously opened channels, sends the program to the END statement to stop.

300 If the error was neither End of File or File Not Found, then it is some other error that makes it impossible to transfer the file. In this case, a message is printed, the channels are closed, and the program stops. This line could be enhanced by incorporating the actual error number for debugging purposes if this program is included within some longer one.

This program illustrates how the important BASIC language commands can be used to send data out the Serial port. While it is not comprehensive, it does show some good programming techniques. (And it works!)

Before using the RS-232 port, be sure to confirm that the parameters specified in the SET Utility program match those of the connected device.

CONCLUSION

This section has described the two ways that the 1752A communicates with other pieces of equipment using worldwide standard connection and protocol methods. This section has also shown how to make use of the essential BASIC language commands to send and receive information over its instrumentation bus and serial data communications bus.

The next section, Creating and Editing Programs, will show how to begin writing your own routines with a view to automating the functions of the 1752A.

Section 6
Creating And Editing Programs

CONTENTS

Introduction 6-2
Selecting a Programming Language 6-3
- BASIC 6-3
- FORTRAN 6-4
- Assembly Language 6-4

File Utility Program 6-5
Command Line Interpreter 6-5
- Introduction 6-5
- Editing Features of the Command Line Interpreter 6-6

The Edit Program 6-7
- Introduction 6-7
- Entering the Editor Program 6-8
- Exiting the Editor Program 6-10
- Operating Modes 6-11
- Global Commands 6-12
- Most Used Commands 6-12
- Command Mode 6-17
- Command Mode Commands 6-20
- Edit Program Messages 6-47

Conclusion 6-48

INTRODUCTION

The 1752A Data Acquisition System is a special-purpose computer, and like any computer, its instructions must be given in very precise language. This section is designed to assist you in the task of writing precise instructions.

There are a number of different facilities for writing and editing programs. From Immediate mode BASIC, the command EDIT presents a range of possibilities. This is the Editor of choice for programs written in BASIC, because it is so well adapted to the task, and because it is readily available, once you are "in BASIC".

For creating other types of programs, such as Command files or Alias files, the System Editor program (filename EDIT.FD2) is easier to get at. Also, you will have to use the System Editor if you are programming in FORTRAN or some other language that does not provide editing functions.

This section discusses these main topics:

Selecting a Programming Language
Creating Programs Using the File Utility Program
The Command Line Interpreter
The Edit Program

SELECTING A PROGRAMMING LANGUAGE

There are three languages that can be used for programming the 1752A. The data acquisition and control capabilities for the 1752A's measurement and control options are available only in the BASIC language. FORTRAN and Assembly Language routines may be written and linked into BASIC to provide additional computational capabilities and improved execution speed. Here is a guide to help in selecting the most appropriate language for any particular application.

BASIC

BASIC is an acronym for Beginners All-Purpose Symbolic Instruction Code. The BASIC language is used for about 90% of all programming applications because it is fairly common, is easy to learn, and provides most of the capabilities that are desired for instrumentation systems. There are really two forms of the BASIC language: the normal interpreted version, and a special compiled version.

Interpreted BASIC is an enhanced version of the BASIC language that is common to most computers. It has a built-in editor, Immediate Mode Commands, and runs programs interactively. These features make program development easy and straightforward, even for inexperienced users. Because it is the language of choice for most applications, the interpreted version of BASIC is included with every 1752A.

Compiled BASIC is essentially the same language in a compiled form to increase its execution speed. Compiled BASIC programs are written using the System Editor program. They can be created in a more structured form than conventional line-oriented BASIC programs. The programs are compiled into a form that runs much faster than an equivalent interpreted BASIC program. Compiled BASIC should be used in applications where additional speed or a modular structure is needed while retaining the familiar BASIC language elements. Compiled BASIC is available as an optional accessory software package which contains the software required to create and maintain Compiled BASIC programs.

Extended BASIC is essentially the same as Compiled BASIC with the additional capability for writing programs requiring a large memory space. Extended BASIC allows the programmer to write large programs without chaining program segments or creating overlays. Extended BASIC is available as an optional accessory software package which contains the software required to create and maintain Extended BASIC programs.

FORTRAN

FORTRAN is also an acronym. It stands for *For*mula *Trans*lator. FORTRAN is a useful language for scientific applications because of the ease with which it manipulates numbers. BASIC provides most of the same capabilities, but FORTRAN may be the better choice if many of your current programs are already written in FORTRAN, if you are experienced in programming in FORTRAN, or if the operation requires greater speed. Because it is a compiled language, FORTRAN offers high speed, but, like all compiled languages, it is more complicated to work with.

Assembly Language

Assembly Language provides the programmer with access to all of the capabilities of the TMS-99000 processor used in the 1752A. Assembly Language programs can usually be both faster and shorter than programs written in any other language. In addition, specialized Input/Output and data conversion functions not otherwise available sometimes must be written in Assembly Language. The penalty for this flexibility is that Assembly Language programs usually take longer to design, to write, and to debug than programs written in higher-level programming languages.

When greater performance is required for a program written in a higher-level language it is usually possible to replace time-consuming operations with a faster Assembly language subroutine. This can be a cost-effective solution if the amount of Assembly code is small compared to the total size of the program.

FILE UTILITY PROGRAM

It is possible to create, but not edit, programs using the File Utility program. To implement this capability, use the File Utility program command in the form:

(filename)= KB0: 〈RETURN〉

Now, any keystrokes made will be filed at the System Device, and given the filename specified. The end of the file must be indicated to the File Utility program by the command 〈CTRL〉/Z (EOF).

This facility is not used for creating long or complex programs because there is no way to edit them. If an incorrect keystroke is not seen immediately, it is necessary to either re-write the entire file, or else use an editor program to correct it.

COMMAND LINE INTERPRETER

Introduction

The central program, FDOS, is always present in the system's memory, and its facilities are used by other programs. For example, when a program written in BASIC is running, one is tempted to say "a BASIC program is running", but that would be inaccurate because it is actually the BASIC Interpreter program that is running. It calls upon the Operating System to provide file manipulation and other Input/Output. The execution of a program and the Operating System is interleaved.

In the same way, when the utility programs are in use, they direct the activities of FDOS. The Operating System takes control only when:

1. The utility program requests an I/O operation.
2. A severe hardware or software error occurs.
3. The FDOS〉 prompt is displayed.

An important feature of the Operating System is its Command Line Interpreter. As its name implies, this is the portion of the Operating System that accepts keyboard commands and acts on them. It permits us to type FUP, for example, and be understood as saying, "Hello, Operating System, what I want you to do is go out to the floppy disk and find a program called the File Utility program. Read it into memory, and pass control of the microcomputer to it." The File Utility program returns control when it receives the Exit command /X.

Editing Features of the Command Line Interpreter

When a command line is being written, some rudimentary editing functions are available. These features of the Operating System are great time savers, because they speed up the creation of Command files and access to utility programs. All the commands described are available from from FDOS, BASIC, and the TIME, SET and File Utility programs.

〈CTRL〉/F and 〈CTRL〉/R

These are mnemonically named commands for Forward and Reverse. 〈CTRL〉/R causes the last line entered to be displayed again, and repeats until the first command has been reached. When a former command has been displayed, it is available for editing, thus providing the 1752A with an elegant way to avoid repetitious typing.

Example:
Since the 1752A was turned on, these commands were entered:

```
FUP
RS232.CMD
/X
TIME
03 06 83
14 35
```

The first use of 〈CTRL〉/R would display 14 35, the next 03 06 83, the next TIME, then /X, RS232.CMD, and finally FUP. If 〈RETURN〉 is pressed when any of these are displayed (the cursor can be anywhere on the line), the system accepts the command just as if it had been typed in. If you wanted to get back to the line "RS232.CMD" to run the program, rather than having the File Utility program display it, you would use 〈CTRL〉/R until the display showed that line, then press 〈RETURN〉. If you go all the way back to the line reading "FUP" by mistake, just use 〈CTRL〉/F to go forward.

These commands are circular. If 〈CTRL〉/R were pressed once more after "FUP" was displayed, the next thing to be displayed would be 14 35.

Any time that ⟨CTRL⟩/R or ⟨CTRL⟩/F are used to display a previously entered command, the command can be edited using the arrow keys, DEL LINE and DEL CHAR keys, the DELETE key, or the backspace (back to left margin), or LINE FEED (to end of line). Once the line has been edited, pressing RETURN with the cursor anywhere on the line will cause the command to be executed.

Other editing features of the Command Line Interpreter are:

⟨CTRL⟩/U	Erases the current line.
⟨CTRL⟩/T	Clears the screen, and positions the prompt on the first line.
⟨CTRL⟩/P	Aborts whatever operation is in progress, and returns control to the Operating System (FDOS ⟩ prompt).
⟨CTRL⟩/Z	Used as the End Of File (EOF) command.
⟨CTRL⟩/C	Interrupts an operation, and may abort it depending on the operation in progress.
⟨DELETE⟩	Deletes the character at the cursor position and moves the cursor left one position. This action stops at the prompt.
Key Repeat	All keys, including control characters, repeat when held down.

THE EDIT PROGRAM

Introduction

The Edit program is supplied on the System disk as a file named EDIT.FD2. This editor is a visual one, as opposed to "blind" editors. That is, the file is displayed as you edit it.

The Edit program provides a complete set of commands for performing these functions:

- Inserting Text
- Searching, Replacing, and Marking Text
- Positioning the Cursor
- Changing Editor Modes

Entering the Editor Program

From the FDOS⟩ prompt, type:

EDIT ⟨RETURN⟩ or EDIT (filename) ⟨RETURN⟩

In the first case, no file is specified. If this method is used as the entry to the Editor, a filename must be specified later, when exiting the program. The reverse is true if a filename is specified as you enter; one must not be specified as you exit. If different filenames are given, the result would be that two files would be created; the one specified at the beginning of the editing session will be empty.

As always, if no device name is specified, the default is to the System Device.

In this example, the file named FILE.NEW was specified:

```
EDITING FILE "FILE.NEW" [new file]
```

If the file already exists, the bracketed "new file" would not be included on the display.

If no file has been specified, only the cursor appears on the top line.

The rest of the display depends on the file contents, but if the file is being created, the display has omega symbols on the left, indicating empty lines.

```
EDITING FILE "FILE.NEW"
Ω
Ω
Ω
Ω
Ω
Ω
Ω
```

There are some optional flags when entering the editor:

edit -r [filename]
: Enter the editor and edit the file as "read only". This is useful for viewing a file that you do not wish to change but where using the editor is more convenient than using FUP.

edit -c [filename]
: Starts up the editor in command mode. See the section on operating modes for a description of the different modes of the editor. A useful system alias would be "e edit -c ?" if you prefer the editor to start up in command mode.

edit -t [device] [filename]
: With this option you can specify which device you want the temporary files created with the ":r" command to be written to. If no device follows the "-t" flag, then the system device (SY0:) is assumed. See the section on the global command ":r" for an explanation of this option.

Exiting the Editor Program

There are several ways to exit the Editor program. The most commonly used are:

- Entering ⟨CTRL⟩/C at any time while in the editor writes the file and exits the editor. If there is no output file to be written and changes have been made to the text in the buffer, a confirmation is required to exit the editor.

- First record the changed file, then exit the Edit program:

  ⟨ESC⟩ :w ⟨RETURN⟩ write the file
  ⟨ESC⟩ :q ⟨RETURN⟩ exit the Editor

- Exit without recording the changes to the file:

  ⟨ESC⟩ :q! ⟨RETURN⟩

Other commands permit reading a file into the display while editing another file.

Operating Modes

The Editor has two modes of operation. The Insertion mode is the default, and is primarily used for inserting text, although it does allow a certain amount of cursor and text manipulation. The Command mode is used for more powerful cursor and text manipulation, searching, and exiting the program.

Many Edit program commands result in a temporary return to the Insertion mode. When these commands are used, ⟨ESC⟩ is the return to Command mode. The purpose of these commands is to save the time involved in exiting the Command mode, doing the insertion, then returning to Command mode.

To change from Insertion to Command mode:
⟨ESC⟩:@ ⟨RETURN⟩

To change from Command to Insertion mode:
:@ ⟨RETURN⟩

When either change mode command is given, the new operating mode becomes the default.

On the first entry to the Edit program, Insertion mode is the default. If Command mode is desired, the EDIT command can be modified by using the -c switch. The entire command to enter the Edit program in Command mode and begin editing FILE.NEW would read:

    EDIT -c FILE.NEW ⟨RETURN⟩

Global Commands

Global commands operate in either mode. After they are executed, the Editor returns to the default mode. They are used to read or write a file, and check the status of the Editor. Global commands are always preceded by a colon (:). The most commonly used ones are:

:w	Writes the file.
:q	Quits the Editor and returns to the Operating System.
:q!	Quits the Editor without any changes from the current editing session.
:r	Reads another file in during editing.
:s	Substitutes one string of characters for another.

NOTE

When in the Insertion mode, the Global command prefix is <*ESC*>:.

Most Used Commands

The entire command structure of the Editor program contains some redundancy; also, some commands are used infrequently enough to not require a complete discussion. Most programs can be created and edited using only a few of the available commands. The full capabilities of the Edit program are presented at the end of this section.

Cursor Positioning

In the Insertion mode, the four arrow keys on the six-key auxilliary keypad move the cusor in the direction shown by one space or one line.

In the Command mode, these keys can be preceded by a number to move longer distances. For example, 20→ would move the cursor 20 places right, and 20↓ would move it down 20 lines.

The lower-case letter "w" moves the cursor to the right by one word. If a number precedes the "w", the cursor moves right by the number of words specified.

The lower-case letter "b" moves the cursor back by one word. If a number precedes the letter "b", the cursor moves back by that number of words.

Text Insertion and Deletion

In the Insertion mode, any characters typed are displayed and the cursor is moved right one position. The ⟨DEL LINE⟩ and ⟨DEL CHAR⟩ keys on the auxiliary keypad delete characters or the entire line to the right of the cursor without moving the cursor. The ⟨DELETE⟩ key (just above ⟨RETURN⟩) erases single characters to the left of the cursor, and moves the cursor left one position.

In the Command mode, typing the lower-case letter "a" returns operation to the Insertion mode. Anything typed in strikes over the already existing text, but it is displayed again when the ⟨ESC⟩ key is pressed. The upper-case letter "A" is a command to add whatever text follows, starting at the end of the line the cursor is on. To stop inserting text, press ⟨ESC⟩.

To delete text in the Command mode, position the cursor anywhere on the line and type ⟨CTRL⟩ / U; the entire line is deleted. ⟨DEL CHAR⟩ and ⟨DEL LINE⟩ operate just as they do in Insertion mode. If a number precedes them, that number of characters or lines to the right of or down from the cursor are deleted. Similarly, the ⟨DELETE⟩ key can be preceded by a number. For example, the command 3⟨DELETE⟩ deletes three characters to the left of the cursor.

Substitution

The substitution (s) command replaces whatever text is to the right of the cursor with a specified character. The form is [n]s {char} where [n] is the number of characters to be substituted, and {char} is the character to be substituted.

In this example, a time delay of 200 milliseconds was found to be inadequate for a program, so the number 200 will be substituted for 1000. With the cursor positioned under the 2 of 200 the command 4s1000 changes the 200 to 1000, leaving the cursor at the end of the line.

```
       .                                          .
       .                                          .
       .                                          .
60  WAIT 200   ----->  4s1000  ----->   60  WAIT 1000_
       .                                          .
       .                                          .
       .                                          .
```

All Command mode returns to Insertion mode are terminated by ⟨ESC⟩.

Marking Text

Invisible markers can be placed anywhere in the text from the Command mode. There can be 26 such markers in any file, one for each lower-case letter in the alphabet. The command ma places a marker named "a" at the cursor position. To return to that position after subsequent editing, use the command 'a.

Searching

Searching can only be done from the Command mode. The Find (F) command operates only on the current line, and searches for a character to the left of the cursor. The entire command takes the form [n]F {char}, where n is a number of occurrences prior to the cursor position, and {char} is the character to be searched for. In this example of a line from a BASIC language program, the command F1 would leave the cursor at the line number:

```
10  DIM A$ (5%, 5%)
```

The second type of search looks for patterns rather than single characters. Each of the command characters must be preceded by ⟨ESC⟩. The three command characters that the search command can begin with are:

/ searches forward throughout the buffer.

? searches backward throughout the buffer.

! searches forward to the end of the file.

Whichever character begins the command, search commands always take the form [cc] {pat} where cc is the command character, and {pat} is the pattern to search for. The pattern searches are repeatable by using the lower-case letter n to continue searching in the same direction, and the upper-case N to reverse direction of the last search.

In this example, the programmer wants to find all those occurrences in a program where something is to be printed in normal size letters. The cursor is postioned at the end of line 20. The command !PRINT NS would first locate the PRINT statement in line 400. The lower-case letter n (next) would then locate the next occurance at line 9000.

```
20 NS$ = ES$ + "p"
           .
           .
           .
400   PRINT NS$; NC$;
           .
           .
           .
9000  PRINT NS$
```

INSERTION MODE COMMAND TABLE

ACTION	RESULT	COMMENTS
Cursor Positioning		
→	Move cursor right one position.	Ignored if cursor is at the last position on the line.
←	Move cursor left one position.	Ignored if cursor is at the left margin.
↑	Move cursor up one line.	Cursor remains in same position on next line unless it is shorter; then it goes to the end of the line.
↓	Move cursor down one line.	Acts the same as ↑
⟨BACKSPACE⟩ or ⟨CTRL/H⟩	Move cursor to the beginning of the line.	
⟨LINE FEED⟩ or ⟨CTRL⟩ J	Move cursor to the last position in the line.	
Deletion Commands		
⟨DEL CHAR⟩ or ⟨CTRL D⟩	Delete character at the cursor position.	
⟨DELETE⟩	Delete character to the left of the cursor.	
⟨DEL LINE⟩ or ⟨CTRL E⟩	Delete text from the cursor to the end of the line.	

Command Mode

The Command Mode of the System Editor program provides for rapid cursor placement, complex searches, deletions, and text marking. It includes commands to move sections of a file into a "yank" buffer, so it can be placed back into the file at another location. Many commands provide a temporary return to the Insertion mode. These commands must be terminated by ⟨ESC⟩ to return to the Command mode.

Markers

Each of the lower case letters on the keyboard can be used as a marker. Markers are invisible, so if they will be used extensively, it is probably a good idea to keep a tally sheet handy to aid in remembering where each marker is placed.

To place a marker named "a" into a file, move the cursor to the desired position, then give the command ma.

Now that marker "a" is in place, you can always return to that spot by a search command. The marker is also used with yank buffers and delete commands (see below.)

There is no provision for deleting markers. However, they are not recorded with the file and so do not remain after the current editing session. Also, the same marker can be moved simply by placing it someplace else. The last placement is the one remembered by the Editor program.

The Yank Buffer

A yank buffer is a location in memory available to store information. Information is taken from the cursor location to a specified marker.

The yank buffer is one of the Editor's more powerful features because it can be used to relocate portions of a program as an aid to modifying it. For example, if part of a program has inadvertently been left out, and to include it requires restructuring the program, use the yank buffer to move sections of the program from the screen and into memory, then replace them as the new section is written. Another good use for the yank buffer is as a holding area for a frequently written line of code, such as a tightly formatted PRINT statement or a very long line.

y`a removes text from the cursor to marker a.

p puts the buffer back into the text after the character where the cursor is positioned.

P puts the buffer back into the text before the character where the cursor is positioned.

Search Commands

Searches can be performed from the cursor position either backwards or forwards in a file. The search can be for markers, strings of characters, lines or line positions, or to a string that matches a metacharacter.

To search for the first occurrence in a file of the word PRINT, the command most commonly used would be: !PRINT

Other variations are to use the command characters / and ? to search forward and backward on the current page.

The lower case letter n finds the next PRINT statement.

The upper case letter N locates the previous PRINT statement.

Metacharacters

Metacharacters describe patterns of characters that may be more complex than words or simple strings of characters. Metacharacters are similar to wild cards, but are only used with search commands.

(dot) — Matches any single character in the line.
Example: ta.k matches talk, task, and tank, but not take.

^ (caret) — Matches at the beginning of a line.
Example: ^PRINT would locate all the PRINT statements in a program, as long as there were no line numbers.

$ (dollar) — Matches at the end of a line.
Example: RESUME 580$ would locate all the lines in a program that RESUMEs to line 580.

[ ] (brackets) — Match constructed patterns.
Example: [13579] matches any single-digit odd number. [Pp]rint matches both Print and print.

- (dash) — With the bracket metacharacter, specifies a range of characters (in ASCII order) as a character class.
Examples: [0-9] matches any single digit.
[- ~] matches all printable characters
(ASCII 'blank' to tilde)

! (bang) — Matches any character not in a specified character class.
Example: [!a-zA-Z] matches everthing that is not a letter.

* (star) — The closure character; matches zero or more repetitions of character(s) matched by preceding patterns.
Examples: X* matches zero or more upper-case X's in a row.
(.*) matches anything between parantheses.

\ (backslash) — The escape operator; causes the character immediately following to be treated as a literal character, even if it is a metacharacter.
Example: The pattern \$ matches the dollar sign, not the 'end of line' metacharacter. \\matches the backslash character.

Command Mode Commands

The command mode provides complex cursor positioning, pattern searches, text replacement, deletion, and insertion. In the section that follows, each command is explained, and a five line section of a program is used for the examples. The example program lines are shown double-spaced to better illustrate the movements involved; in actual practice, programs do not allow empty lines as shown here:

```
10  DIM A$ (5%, 5%)

20  TRACE ON 110, A$ ()

30  FOR I% = 0% TO 5%

40  A$ (I%, 0%) = CHR$ (ASCII (' ')) + I%

50  GOSUB 110
```

Cursor Positioning

Moving the Cursor Forward

Command: [n]→ -or- [n]l -or- [n] ⟨SPACE⟩

Purpose: To move the cursor to the right n characters.

Example: The cursor is at the beginning of line 30.
The command 12→ moves it to the equals sign:

```
30  FOR I% = 0% TO 5%
```

Command: [n]|

Purpose: To move the cursor to column n.

Example: The cursor is under 'C' of CHR$ (column 19) on line 40.
The command 38→moves it to column 38 (the I of I%).

```
40  A$ (I%, 0%) = CHR$ (ASCII (' ')) + I%
```

Command: [n]w -and- n[W]

Purpose: Lower-case: To move the cursor forward on a line the specified number of words.
Upper-case: Operates on strings.

A word is made up of alphabetic and numeric characters, and ends with a space, tab, or punctuation mark, or symbols such as $, %, or &. A string can include symbols in addition to the alphanumeric characters. The cursor is left at the beginning of the word or string.

Example: The cursor is at the beginning of line 30.
The command 7w leaves the cursor under the word TO:

```
30  FOR I% = 0% TO 5%
```

Example: The cursor is at the left margin of line 30.
The command 6W moves it to the 5 of 5%

```
30  FOR I% = 0% TO 5%
```

Command: [n]e -and- [n]E

Purpose: Lower-case: To move to the end of the specified word.
Upper-case: To move to the end of the specified string.

Example: The cursor is at the beginning of line 30. The command 2e positions it at the R of FOR.

```
30  FOR I% = 0% TO 5%
```

Command: [n] $ -or- [n] ⟨LINEFEED⟩ -or- [n] ⟨CTRL⟩/J

Purpose: To move the cursor forward to the last position of line n; the current line if n = 1, or if no number is used.

Example: The cursor is at the left margin on line 30.
The command 3$ moves it forward to the end of line 50:

```
10  DIM A$ (5%, 5%)
20  TRACE ON 110, A$ ()
30  FOR I% = 0% TO 5%
40  A$ (I%, 0%) = CHR$ (ASCII (' ')) + I%
50  GOSUB 110
```

Command: [n] + -or- [n] ⟨RETURN⟩ -or- [n] ⟨CTRL⟩/M

Purpose: To move the cursor forward to the left margin n lines down; if n = 1, the next line down.

Example: The cursor is at the end of line 20.
The command 2+ moves it to the beginning of line 40:

```
10  DIM A$ (5%, 5%)
20  TRACE ON 110, A$ ()
30  FOR I% = 0% TO 5%
40  A$ (I%, 0%) = CHR$ (ASCII (' ')) + I%
50  GOSUB 110
```

Moving the Cursor Backward

Command: [n]b -and- [n]B

Purpose: Lower-case: To move the cursor backward to the beginning of a specified word.
Upper-case: To move the cursor backward to the beginning of a specified string.

Example: The cursor is under the R of 'FOR' of line 30.
The command 2b moves it to the 3 of 30:

```
30  FOR I% = 0% TO 5%
```

Command: [n]^ -or- [n] ⟨BACKSPACE⟩ -or- [n] ⟨CTRL⟩/H

Purpose: To move the cursor from the current position to the left margin of the *n*th line up. If n = 1, the cursor moves to the left margin of the current line. The current line is also used if no number or zero are given.

Example: The cursor is at the end of line 50.
The command 3^ moves the cursor to the beginning of line 30:

```
10  DIM A$ (5%, 5%)
20  TRACE ON 110, A$ ()
30  FOR I% = 0% TO 5%
40  A$ (I%, 0%) = CHR$ (ASCII (' ')) + I%
50  GOSUB 110
```

Command: [n] - -or- [n] ⟨RETURN⟩ -or- [n]

Purpose: To move to the left margin of the nth line up.

Example: The cursor is at left margin of line 50.
The command 2- moves it to the left margin of line 30.

```
10  DIM A$ (5%, 5%)
20  TRACE ON 110, A$ ()
30  FOR I% = 0% TO 5%
40  A$ (I%, 0%) = CHR$ (ASCII (' ')) + I%
50  GOSUB 110
```

Command: [n]↑ -or- [n]k

Purpose: Moves the cursor up n lines. Attempts to keep cursor positioned in the same location on the new line. If the target line is shorter, cursor moves to last position.

Example: The cursor is under the 'A' of ASCII in line 40.
The command 3↑ moves the cursor to the last position of line 10:

```
10  DIM A$ (5%, 5%)
20  TRACE ON 110, A$ ()
30  FOR I% = 0% TO 5%,
40  A$ (I%, 0%) = CHR$ (ASCII (' ')) + I%
50  GOSUB 110
```

Moving the Cursor Down

Command:[n]↓ -or- [n]j

Purpose: To move the cursor down n lines. If the cursor cannot be kept in the same position because the target line is shorter, it will be placed at the last position.

Example: The cursor is at the last position of line 20. The command 2↓ moves it to the space after CHR$ in line 40:

```
10  DIM A$ (5%, 5%)

20  TRACE ON 110, A$ ()

30  FOR I% = 0% TO 5%

40  A$ (I%, 0%) = CHR$ (ASCII (' ')) + I%

50  GOSUB 110
```

Long Cursor Movements

Moving to the End of the Buffer

Command: [n]g

Purpose: To move the cursor to line n of the buffer. If n is not specified, the cursor moves to the last line in the buffer.

Example: Line 50 is the last line in the buffer. No matter where the cursor is situated, the command g moves the cursor to the beginning of line 50.

```
01  ANY LINE IN BUFFER
          .
          .
          .
50  GOSUB 110
```

Moving to the End of the File

Command: [n]G

Purpose: Moves the cursor to the specified line of the current file. If not specified, the cursor goes to the last line of the file.

Example: Same as lower-case if the entire file is in the buffer.

Moving By Screenfuls

Command: H

Purpose: To move the cursor to the top line of the screen. (High)

Command: L

Purpose: To move the cursor to the bottom line of the screen. (Low)

Command: [n] ⟨CTRL⟩/F

Purpose: On extremely long files, moves forward by an entire screenful (15 lines). The [n] indicates the number of screenfuls to go forward.

Example: The sample program lines are about 80 lines ahead in the file. The command 5 ⟨CTRL⟩/F will position the cursor in the general vicinity of the program. (5 x 15 75)

Command: [n]⟨CTRL⟩/B

Purpose: To move backwards by screenfuls, as in the prior command.

Search Commands

Searching for a Single Character

Command: [n] f {c}

Purpose: To move the cursor forward to the *n*th occurrence of a character on the same line. If the character does not occur the number of times specified, the command is ignored.

Example: The cursor is at the left margin position on line 40. The command 2f$ leaves it under the $ of 'CHR$'

```
40  A$ (I%, 0%) = CHR$ (ASCII (' ')) + I%
50  GOSUB 110
```

Command: [n] F {c}

Purpose: Same as prior command, except searches backwards for the character.

Command: [n] t {c}

Purpose: Same as [n] f {c} except leaves the cursor at the left of the character specified.

Command: [n] T {c}

Purpose: Same as [n] f {c} except searches backward, and leaves cursor at the right of the specified character.

Searching For a Pattern

Command: / {pat}

Purpose: Moves the cursor forward to the beginning of the specified pattern. The pattern may be anywhere between the current position and the end of the buffer. If the pattern does not occur, the cursor remains, and the message PATTERN NOT FOUND is displayed.

Example: The cursor is at the first position of line 10. The command /5% positions the cursor at the 5 of 5% on the same line:

```
10 DIM A$ (5%, 5%)
```

Command: ? {pat}

Purpose: Same as prior command, but searches from present position backwards to beginning of buffer.

Command: ! {pat}

Purpose: Same as / {pat}, except searches forward to the end of the file.

Command: n

Purpose: To find the next occurence of a pattern specified by the original /, ?, or ! command.

Command: N

Purpose: Same as n(ext), but in opposite direction of the original /, ?, or ! search.

Marker Commands

Placing a Marker

Command: m[x]

Purpose: To place an invisible marker in the text at the cursor position. The [x] must be a lower-case alphabetical character. If the same label is used a second time, the first one is deleted. All markers are deleted if the file is written (w), or when the Editor is quit (q).

Example: Place a marker named c at the string 'CHR$'
Position the cursor at the 'C', and type mc.

Finding a Marker

Command: `[x]

Purpose: To find a previously defined marker. If no marker by that name exists, the message "Mark not set" is displayed.

Example: The cursor is at the beginning of line 10.
The command `c moves the cursor to the previously defined marker at 'CHR$' on line 40.

```
10   DIM A$ (5%, 5%)
20   TRACE ON 110, A$ ()
30   FOR I% = 0% TO 5%
40   A$ (I%, 0%) = CHR$ (ASCII (' ')) + I%
50   GOSUB 110
```

Text Insertion

Text can only be inserted from the insertion mode. However, for convenience, many Command mode commands provide a shortcut entry to and return from the Insertion mode. The return to Command mode is one keystroke: ⟨ESC⟩ Here are the commands:

Command: a -or- A

Purpose: To append text to existing lines. To terminate the insertion, use ⟨ESC⟩. If upper-case, first moves the cursor to the end of the current line.

Example: A comment is to be added to line 10.
Use the upper-case 'A' command to position the cursor at the end of the line, and enter Insertion mode.

```
10  DIM A$ (5%, 5%)_
```

Text can now be inserted. Space over and begin the comment with an exclamation point.

```
10  DIM (10,50)    ! dimension a 5 by 5 array_
```

To terminate the insertion, press ⟨ESC⟩

Command: i -or- I

Purpose: To insert text at the cursor, moving other text off to the right. The upper-case operates the same as lower case, but first moves cursor to beginning of current line. Terminate the insertion with ⟨ESC⟩.

Example: In typing line 40, the word 'ASCII' was inadvertently left out. The remainder of the line is all right. Position the cursor at the space between the parantheses and type the single letter command i.

```
40  A$ (I%, 0%) = CHR$ (_(' ')) + I%
```

Now type the word ASCII. The result will be:

```
40  A$ (I%, 0%) = CHR$ (ASCII_(' ')) + I%
```

Terminate the insertion by typing ⟨ESC⟩

Text Substitution

Command: [n]s

Purpose: The substitute command. The number of characters specified are substituted at the cursor.

Example: The word ASCII is accidentally typed as EBCDIC. With the cursor positioned at the A of ASCII, the command 6s will substitute EBCDIC for whatever characters follow until ⟨ESC⟩ is pressed. The cursor is left at the position following the substitution.

```
40  A$ (I%, 0%) = CHR$ (EBCDIC (' ')) + I%
```

The result will read:

```
40  A$ (I%,0%) = CHR$ (ASCII_(' ')) + I%
```

Command: r[c]

Purpose: The replacement command allows whatever character is at the cursor to be replaced by the specified character. This command differs from the substitute command in not entering the Insertion mode, except for the single character replaced. There is no need to press ⟨ESC⟩ to return to Command mode (you never left it).

Example: In line 10, the array to be dimensioned was given the arguments (7%, 5%). To change the 7 to a 5, position the cursor under the 7 and type the command r5.

```
10  DIM A$ (7$, 5$)
```

The result will be:

```
10  DIM A$ (5$, 5$)
```

Case Conversion

Command: [n]~

Purpose: To change the case of a number of characters, either upper- to lower-case, or vice versa.

Example: On line 40, ASCII was mistakenly typed ascii.

```
40  A$ (I%, O%) = CHR$ (ascii)) + I%
```

Position the cursor at the first character, and give the command 5~. The cursor moves past the inverted text:

```
40  A$ (I%, O%) = CHR$ (ASCII)) + I%
```

Text Deletion Commands

Small amounts of text can be deleted from the Insertion mode. The Command mode expands the possibilities, and permits deletion of text in varying amounts.

Deleting By Character Amounts

Command: [n] ⟨DEL CHAR⟩ -or- [n]x -or- [n] ⟨CTRL⟩/D -or- [n]s ⟨ESC⟩

Purpose: Delete a specified number of characters (not spaces) after the cursor.

Example: In line 20, everything following TRACE ON is to be deleted. Position the cursor at the space following TRACE ON, and give the command 7x.

```
20  TRACE ON_110, A$ ()
```

The new line will read:

```
20  TRACE ON_
```

Command: [n] ⟨DELETE⟩ -or- n[X]

Purpose: Same as[n]x, except deletes from the cursor backwards.

Command: [n] d [^]

Purpose: To delete all text from the cursor to the end of the *n*th line prior to the cursor. If n = 1, this command uses the cursor line.

Command: [n] d [$]

Purpose: Same as prior command, but deletes from the cursor forward.

Command: [n]s ⟨ESC⟩

Purpose: This command is a special side-effect of the substiution command [n]s. It deletes the specified number of characters forward from the cursor. A block marker ▮ is momentarily visible at the location to be deleted to, until ⟨ESC⟩ is pressed. If any keystrokes are made before the ⟨ESC⟩ terminator, the characters will write over whatever is currently at those positions. See also the [n]s substitute command.

Example: The cursor is on line 40. To delete the word ASCII and the space and left paranthesis after it, use the command 7s ⟨ESC⟩

```
40  A$ (I%, O%) = CHR$ (ASCII (▮)) + I%
```

The result will be:

```
40  A$ (I%, O%) = CHR$ (_)) + I%
```

Deleting By Line Amounts

Command: [n]dl -or- [n]⟨CTRL⟩ / U

Purpose: Delete the current line, and forward the number of lines specified.

Example: The cursor is in the middle of line 10.
The command 3dl deletes lines 10, 20, and 30. An epsilon symbol is left on the deleted lines:

```
ϵ

ϵ

ϵ

40  A$ (I%, O%) = CHR$ (ASCII (' ')) + I%

50  GOSUB 110
```

Command: [n]D -or- [n] ⟨CTRL⟩ / D

Purpose: To delete forward from the cursor position to the end of the *n*th line down.

Example: As in the previous example, except that not all of line 10 is deleted. the Command 3D results in:

```
10  DIM A$ (5%, 5%)_

ϵ

ϵ

40  A$ (I%, O%) = CHR$ (ASCII (' ')) + I%

50  GOSUB 110
```

Deleting to a Marker

Command: d`[x]

Purpose: Delete from the cursor foward or backward to the specified marker.

Control Commands

Opening a New Line

Command: o

Purpose: Opens a line below the cursor line, and switches to Insertion mode. Notice that the cursor line is momentarily erased, but returns when the screen repaints on leaving the Insertion mode ⟨ESC⟩.

Example: A line is to be added between lines 10 and 20. Position the cursor anywhere on line 10, and press the letter o key.

```
10  DIM A$ (5%, 5%)


20  TRACE ON 110, A$ ()
```

Command: O

Purpose: Same as the lower-case command, except the line is opened above the cursor.

Joining Open Lines

Command: J

Purpose: Joins the cursor line to the line below.

Example: Lines 40 and 50 are to be combined into one line. Position the cursor anywhere on line 40, and give the command (must be upper case; lower case j is equivalent to the down arrow).

```
40  A$ (I%, O%9 = CHR$ (ASCII (' ')) + I%

50  GOSUB 110
```

The result will be:

```
40  A$ (I%, O%) = CHR$ (ASCII (' ')) + I% 50 GOSUB 110
```

Complete the new line by using ⟨DEL CHAR⟩ to remove the line number 50, then separate the two statements by a backslash (\).

```
40  A$ (I%, O%) = CHR$ (ASCII)) + I% \ GOSUB 110
```

Command: ⟨CTRL⟩/L -or- ⟨CTRL⟩/R

Purpose: These commands eliminate unused lines by repainting the screen. Any epsilon characters that have been generated during an editing session will be dropped.

Translating Upper and Lower Case

Command: ⟨CTRL⟩/A

Purpose: This command is a toggle that maps upper case to lower case and vice versa. It is only active in the Insertion mode, and can be helpful for editing FORTRAN programs or others that must be all upper-case. It allows one to enter lower-case commands (most of the Edit program's commands are lower-case) while entering upper case text.

Target Commands

Target commands have two parts, the command itself and a "target" that specifies the extent of the command. The command determines the action to be taken, such as deleting or yanking. The target specifies the direction and number of characters that the action will be performed on. Targets are the same cursor motions that were presented earlier, "w" for word, for example. The user can delete a word of text by the command "dw". The "w" specifies that the deletion will take place over a cursor motion of one word to the right.

By entering the command twice for the "d" and "c" commands, the action will take place on the entire line. For example, the command "4dd" would delete four lines of text.

The table below explains the three Target commands:

Change text from cursor to target (Enters Insertion mode: terminate with ⟨ESC⟩).	[n]c {target}
Delete from cursor to target.	[n]d {target}
Yank from cursor to target.	y[n] {target}

The table below shows each of the Targets, and indicates the cursor movement for each of them:

CURSOR MOVEMENT	TARGET
Left margin of current line.	^
Last character of current line.	$
One space to the right.	⟨SPACE⟩
To specified column.	[n]|
Forward to the *n*th occurrence of c.	f{c}
Forward to the character prior to the *n*th occurrence of c.	F{c}
Forward to the character after the *n*th occurrence of c.	t{c}
Back to the character after the *n*th occurence of c.	T{c}
Beginning of next word.	w
Beginning of next string.	W
End of next word.	e
End of next string.	E
Beginning of previous word.	b
Beginning of previous string.	B
To a marker.	`[mark]

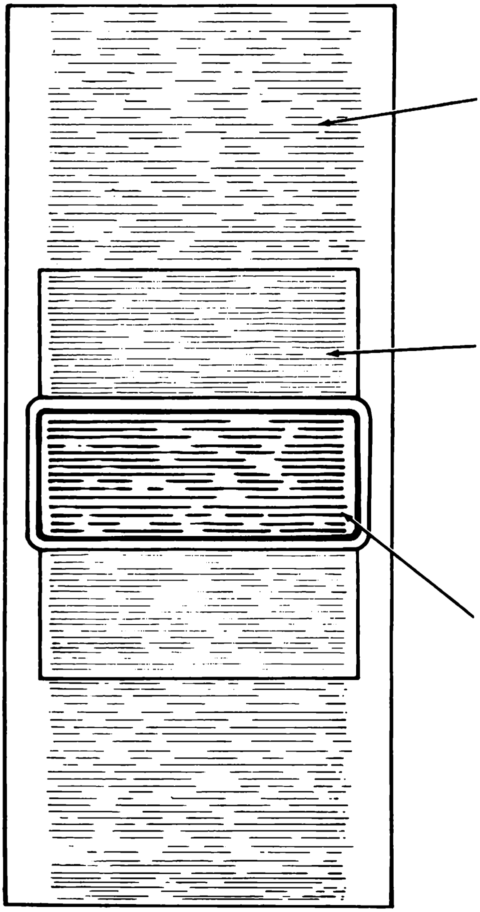
Global Commands

Global commands are available from either the Insertion or the Command mode. They perform these functions:

- Read from files into the buffer.
- Write from the buffer into a file.
- Page through a file.
- Toggle the default mode.
- Check amount of available memory space.
- Display the software version.
- Exit to the the operating system.

All the Global commands must be preceded by a colon (:). From the Insertion mode, they are preceded by ⟨ESC⟩: All Global commands are terminated by ⟨RETURN⟩ or ⟨ESC⟩.

Before using any of the Global commands, be sure you are familiar with these terms:



File: A complete package of information, either data or a program, recorded in a file-structured device. The file may be shorter or longer than the buffer, but is generally longer. If it is longer, that portion that doesn't fit is retained on the disk.

Buffer: An area of memory that the file is brought into for editing. Notice that the buffer may not be large enough to contain the entire file. This is important, because some of the global commands assist you in manipulating portions of the file in and out of the buffer.

Screen: The 15 lines of the buffer that are displayed at any given time.

Editing an Existing File

Command: :e {filename.ext}

Purpose: Reads the named file and opens a temporary file where changes will be recorded. If there is no file by the name specified, a message is displayed. This command is equivalent to exiting the Edit program, and reentering it from FDOS with the command EDIT {filename}.

Example: Work has been finished on one file, and you wish to save it and edit another. File the first one, but do not exit the Edit program. To do this, use the Global command :w to record the first file, then the :e {filename} command to begin editing the next.

Command: :e! {filename.ext}

Purpose: Discards any changes to file being currently edited, then begins editing the named file. This command is equivalent to using the :q! command to exit the editor without changing the file, and re-entering it from FDOS with the EDIT {filename} command.

Example: During editing, it is discovered that the wrong file is being corrected. To exit without incorporating any changes, but stay in the Edit program, re-file the first program, and use the :e! {filename} command to bring in the second.

Paging Through a File

Command: :p

Purpose: Writes out the current contents of a buffer into the output file, then loads the next page into the buffer, displaying the first 15 lines. This command is equivalent to :w followed by :r (see next page).

Reading a File into the Buffer

The commands for reading in another file require temporary files to be created. If the -t option is not specified on the command line then the temporary files will be created on the same device as the input file. Sometimes there is no room for the temporary files and the output file on the same device. It then becomes convenient to use the -t option to specify another device. For example, say you are editing a file on the floppy disk and it is the default system device, SY0:. You know you will be merging some other files into this file using the :r command. You might want to start up the editor with the command:

```
edit -ted0: file.bas
```

This command instructs the editor to create the files needed for the :r command on the device ED0:. This allows the largest amount of disk space to be allocated on the system device for the output of the updated copy of the input file. A temporary file created by a single :r command will never exceed 50 blocks in size. Even in systems without extra memory installed, the E-disk (device ED0:) is a convenient device for creating the temporary files.

Command: :r {filename}

Purpose: To read one file, and write it (or portions of it) out to another file. The :r command is normally followed by a :w command to facilitate merging files.

Example: A subroutine has already been developed, debugged,and incorporated into a file named ACDC.BAS. It has been found to have application within another program currently being developed. While creating the new file (call it SYNCH.BAS, for example), read in ACDC using the :r command, and delete everything except the desired subroutine. Now write the new file using the :w command.

Command: :R {filename}

Purpose: Same as the lower-case :r command, except that :p will work without creating a temporary file.

Example: A program is being created, and it is thought that a particular subroutine can be incorporated. Use the :R command to bring the file to the buffer. This command allows you to page through the file without modifying it.

Command: :r -or- :R

Purpose: Read from the current input file, inserting text at the current cursor position. These commands are the same as the other :r commands, except that no filename is specified, so the file currently being edited is used.

Example: A program in development has been designed with cascaded loops, or with many repetitive lines of code. Write the module to a file, then use the :r command to repetitiously read it back to the screen.

Substituting Patterns of Text

Command: :s/ {oldpat} / {newpat} /

Purpose: Substitute one pattern for another. The patterns may be words, strings, or expressions. The substitution begins at the current cursor position, and continues to the end of the buffer.

Example: During program development, it became necessary to renumber the starting line of a subroutine. All GOSUB statements must be changed to reflect the new line number. If the old line number was 1040, and the new line number is 1060, use the command:

Command: :S/ {oldpat} / {newpat} /

Purpose: Same as the lower case example, except that this command operates on the entire file, not just the buffer contents.

Writing From the Buffer Into a File

Command: :w {filename}

Purpose: Writes the contents of the buffer to the named file. If the output file is the same as the the current input file, it is not necessary to specify a filename. This is the normal command to record a new or revised program to a file, either prior to exiting or before reading (see :r) a new file to the buffer for editing.

Toggling the Default Mode

Command: :@

Purpose: This command changes the default mode between Insertion and Command modes. Use the Insertion mode for most normal keyboard entries of programs, and light editing, like deletions and small cursor movements. Use the Command mode for large movements, searches, replacements, marking text, and other more complicated editing.

Example: To switch to Insertion mode from Command mode:
:@ ⟨RETURN⟩ -or- :@ ⟨ESC⟩

To switch to Command mode from Insertion mode:
⟨ESC⟩:@ ⟨RETURN⟩ -or- ⟨ESC⟩:@ ⟨ESC⟩

Checking Memory Space

Command: :m

Purpose: Displays the size of the text in the buffer, the amount of text in the yank buffer, and the amount of memory space still available. The information is displayed on the top line of the screen, and is given in bytes.

Example: During an editing session in which a long program has been entered, it is desired to check the amount of memory remaining before continuing. If the text size is approaching the size of the memory remaining, it may be necessary to either clear the yank buffer before proceeding (see the Command mode command Y). The other alternative is to write the current buffer to the file, then continue (see the Global command :w).

When the :m command is given, the top line of the display shows:

```
Text size: 437, Yank size: 0, Space remaining: 36392
```

This display would indicate that only 437 bytes of the buffer are in use, the yank buffer is clear, and that nearly 37 Kbytes of buffer space is still available.

Displaying the Software Version

Command: :v

Purpose: To check the Edit program version. This is not used during normal Editing, but can be checked if needed as an aid to tracking down a problem that is suspected to be due to software incompatibility.

Exit to the Operating System

Command: :q

Purpose: Returns to the Operating System when an editing session is complete.

Command: :q!

Purpose: Same as the normal quit command, except that no changes are recorded to the file.

Example: During an editing session, some changes have been made to the wrong program, and the revision would be catastrophic if implemented, but would be a major effort to correct. Use the :q! command to return to the Operating System. Now the File Utility program can be used to locate the correct program. If the filename of the correct program is known, use the :e! {filename} command. (See Editing an Existing Program, above.)

Edit Program Messages

Because of the many commands and options that the Edit program provides, it is likely that some error messages will occur during editing sessions. The list below explains what each of the messages means, and is a guide to corrective actions.

MESSAGE	MEANING
Illegal mark name	A mark name was given that is not a lower-case alphabetic character.
Mark not set	A mark was specified that has not yet been assigned.
Invalid target	The command does not exist, or has been entered incorrectly. Insure that the command is constructed properly.
Illegal substitution syntax	The substitution command was ill-formed in some way. Can be caused by an improper character or added spaces in the command.
Invalid target	A Target command has been given for a target that does not exist. Check that the target is available, and that the command has been constructed properly.
Yank buffer empty	An attempt was made to put onto the screen the contents of an already empty yank buffer.
Can't open new output during 'e'dit	An attempt was made to record the buffer to a file other than the input file.
Can't create temp file	Either the media is write protected or the directory is full.

MESSAGE	MEANING
No output file	A new file is being created, but no name was specified when the editing session was started. To write out the buffer to a file, you must specify a filename using the :w or :q commands.
No previous pattern	A next pattern command (⟨ESC⟩ N) was attempted when no pattern was first searched for.
File medium swapped	The disk drive door was opened, and the floppy disk removed during editing.

CONCLUSION

This section has described how to use the System Editor program as a tool for creating and modifying programs. While the descriptions are accurate and complete, the best way to become familiar with the Edit program is to actually use it. This section can be used as a guide while you are trying out the various commands, and will be a useful reference as you gain experience.

The next section shows how to use the various tools available to automate the functions of the 1752A.

Section 7
Automating System Functions

CONTENTS

Introduction 7-2
Command Files 7-3
  Special Characters 7-4
  Sample Command Line 7-5
  The Startup Command File 7-6
  Linking to Other Command Files 7-7
Establishing the Environment-
The BASIC SET SHELL Statement 7-8
Alias File 7-9
  Creating Aliases 7-9
  Error Messages 7-11
  Standard Aliases 7-11
Automating Utility Programs 7-15
  The Time and Date Utility 7-15
  Using the Time and Date Clock 7-15
  Programming Language Commands 7-16
  Set Utility Program 7-17
  File Utility Program 7-17
Sample Instrumentation System 7-18
  Controlling the Sample System 7-20
    Step 1: Start With a Flowchart 7-20
    Step 2: Establish Bus Addresses 7-24
    Step 3: Program the Modules 7-24
    Step 4: Concatenate 7-27
    Step 5: Debugging 7-27
    Step 6: Document the Program 7-28
  Sample Program Listing 7-29
  The Startup Command File 7-32
A Caution to Systems Programmers 7-32
Conclusion 7-34

INTRODUCTION

The power of the 1752A Data Acquisition System is a direct function of its programmability. This section describes how to program the 1752A to perform its various functions automatically. The major topics in this section are:

Command Files
Establishing an Environment - the BASIC SET SHELL Statement
Alias File
Automating Utility Programs
Sample Instrumentation System

COMMAND FILES

The Operating System recognizes the contents of any file with the extension .CMD as a string of ASCII characters (keyboard commands) to the Command Line Interpreter. This feature provides an advantage to the user by allowing a series of keyboard entries, such as those required by a utility program, to be stored as a file. When such a file is active, it can control the utility program without requiring a long string of keyboard entries each time it is used.

In the following example a command file has been written that presets Port 1 to a state required by a printer connected there. The file is called SETPRT.CMD and is created from FDOS using the edit command. All of the Set Utility commands used in this example are defined in Section 5, and the edit program is described in Section 6.

```
set
kb1:
br 9600
db 8
pb n
sb 1
eol 13
eof 26
si e
so e
ex
```

Any time the command file is active it can be aborted by ⟨CTRL⟩/C, or by pressing the front-panel ABORT switch. The message "Command file aborted" is displayed, and control of the system returns to the shell program.

NOTE

Some programs, like TIME and TCOPY, present a special case when they are used as the last line in a command file. Since they generally require user input, it is necessary to end the command file with a blank line if they are the last command in the command file.

Special Characters

Certain characters take on a new meaning when they occur in a command file. This section explains each of these characters, and then gives an example of a command line using these special characters.

! Substituted by a line entered from the keyboard. Using an exclamation point permits the creation of interactive command files. When an exclamation point occurs in a command file, later commands are not acted on until the ⟨RETURN⟩ that terminates the input.

& Causes the line immediately following to be displayed, until a ⟨RETURN⟩, tilde, or exclamation point. An exclamation point or tilde can be put on the displayed line to cancel the display without a ⟨RETURN⟩. In this way, the Command file can be made to wait for operator input before proceeding.

{ } Any characters between braces is displayed. If the left brace does not have a matching right brace, everything after the left one will be displayed. These characters are used to display portions of the Command file to allow easy debugging.

? A metacharacter that is substituted by an argument passed to the command file either from the keyboard or another command file.

$ A metacharacter that is followed by a single digit (0-9), to be substituted by a portion of the argument passed to the command file either from the keyboard or another command file. A portion is defined as a string of characters between the space and end-of-line delimiters. $0 returns the name of the command file (to invoke it repetitiously); $1 returns the first portion of the argument, $2, the second, and so on. If no digit follows the dollar sign, it is passed through unchanged.

~ The tilde clears any previous entry from the Touch-Sensitive Display, and waits for the screen to be touched. Command files cannot decode the location where the screen was touched.

The backslash is the escape operator. Any character following it is interpreted literally. &\ $5.00, for example, displays as $5.00. There are two special cases of this character:

\e
converts to ⟨ESC⟩

\b
converts to ⟨BEL⟩

Sample Command Line

Assume this command line occured in a Command file:

```
&\e[8;22H\e[5mTOUCH\e[1m SCREEN TO CONTINUE\b\e[m~Thank You.
```

The meaning of this line is:

Print at cursor position 8,22 (line 8, position 22), the word "TOUCH" blinking (escape 5m), and print "SCREEN TO CONTINUE" in high intensity (escape 1m), sound a tone (\ b), and wait for the screen to be touched (~). When the screen is touched, print, "Thank You" on the same line.

The Startup Command File

The file named STRTUP.CMD runs automatically whenever the system is powered up. Like all Command files, the Startup file can be run by typing its name from the FDOS prompt; the extension is unnecessary. The special thing about the Startup Command file is that the Operating System looks for this file whenever the 1752A is powered up or reset.

There are two major results of this feature. First, it allows the initial setup and configuration of the 1752A to be preprogrammed; and second, it permits the keyboard to be disconnected from the 1752A once the programs have been developed to the point that they run properly without it.

NOTE

While developing the Startup Command file, do not name it STRTUP.CMD until it has been tested as a Command file with some other name. If there is a problem e.g., an unending loop is inadvertently created, it is much easier to correct the error if it only occurs when the file is intentionally run, rather than every time the 1752A is turned on.

Linking to Other Command Files

Assume that a Startup Command file looks like this:

```
SET
KB1: BR600, PB E, TO 30
EXIT
TEST1.CMD
BASIC
RUN "RS232.SEL"
```

This Command file would begin by running the Set RS-232 Utility program, establish the baud rate, parity bit, and time out parameters for port KB1:. Then it exits the Utility program and performs the keyboard commands contained in TEST1.CMD, another Command file. When TEST1 is complete (whatever it might be), control reverts to the Startup Command file, which loads the BASIC Interpreter and runs the program RS232.SEL.

Notice that the Command file can bring in another Command file. It is possible, for example, for TEST1.CMD to call still another Command file, say TEST2.CMD. Up to four of these branches are possible. If the fourth Command file calls still a fifth, the first is lost, and any subsequent commands that it contains will not be executed.

Assume that the program named RS232.SEL is a BASIC language program that presents test selections to the operator. By using the BASIC statement SET SHELL, a program can be designed that returns to RS232.SEL when the ABORT button is pressed. Otherwise, the system would return to FDOS, which provides no possibility for operator input other than RESTART, ABORT, or both (a cold start). None of these is of much value, since it means that the test must start again from the beginning, loading the Operating System and the Startup Command file.

ESTABLISHING THE ENVIRONMENT - THE BASIC SET SHELL STATEMENT

Part of the power of the 1752A is the opportunity it affords the programmer to completely structure a programming environment. This capability is a result of the Fluke Enhanced BASIC language statement SET SHELL.

The SET SHELL statement is covered in detail in the BASIC Programming Manual, but deserves mention here because it is so intimately connected with programming the 1752A.

Assume the Startup Command file has these lines:

```
BASIC
SET SHELL"MF0:BASIC"
RUN "PROG1"
RUN "PROG2"
```

On power up, this Command file loads the BASIC Interpreter program, and sets the system to a BASIC Environment. Next, it loads and runs a BASIC language program called PROG1. No matter what happens during execution of PROG1, including a "fatal error", recovery can be made by pressing the ABORT button. The result is that PROG2 would be immediately executed. Upon completion, or when the ABORT button is again pressed, the SET SHELL statement returns control to the BASIC Interpreter program. This is what is meant by establishing the BASIC environment, and is in fact a variation of what the Getting Started disk does.

The Immediate mode BASIC statement SET SHELL (with no arguments) resets the shell to the Operating System. Notice, however, that when the BASIC Interpreter program executes this statement, it does not immediately return to the FDOS› prompt, but to the BASIC Ready prompt. Now, the EXIT command can be used to return to the Operating System. When the shell is set to BASIC, any EXIT commands merely exit the current program, and return you to the BASIC shell, just like pressing RESTART.

Be careful not to use commands like "SET SHELL TIME". Doing so will set the environment to the Time and Date Utility program, which will continually ask you the time of day, rather than doing anything productive.

ALIAS FILE

An alias, as the name implies, is a way to call something by a different name. The purpose of an alias is to provide another programming shortcut to help simplify the creation of programs for the 1752A.

During programming, you may want to shorten frequently used commands (or command lines) by using an alias. Aliases are recorded on the System Disk in a file called ALIAS.SYS. The standard aliases provided with the 1752A can be seen easily by displaying this file. You can use the System Editor program to add your own aliases or to modify the standard ones.

Aliases provide another powerful feature of the Command Line Interpreter. By observing which commands are being entered repetitiously, a collection of shortened commnds can be created and recorded in the file named ALIAS.SYS. The alias file is part of the system software, and its contents become a part of the vocabulary of the Command Line Interpreter during software loading. Since this is true, aliases can be used within the Startup Command file, which does not become active until after the software is loaded.

Aliases are operational whenever the FDOS> prompt is displayed. They can only be used from the FDOS Command Line Interpreter.

Creating Aliases

By adding to the system alias file (ALIAS.SYS), you will be able to abbreviate many commonly used commands into a single keystroke or a short sequence of keystrokes. Be certain to observe the correct syntax. Here is the required syntax for constructing an alias:

key(s) → <space> → command → RETURN

The command may contain the following metacharacters:

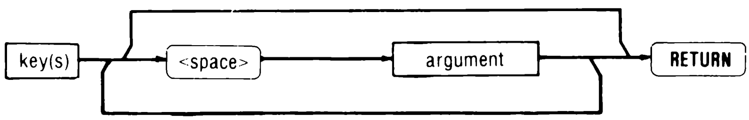
- Use the $ character to pass multiple arguments.
- Use the ? character to pass a single argument line.

According to the required syntax, the alias

cp fup $2=$1

translates into: FUP pathname2 = pathname1 (a useful copy alias).

When the alias is used, the syntax is:



- If the alias takes arguments, then each argument must be separated by a space.

According to this syntax, to use the copy alias in the previous example, you would type:

cp FILE1 FILE2 ⟨RETURN⟩

to copy the contents of FILE1 to FILE2.

- The alphabetic case of an alias is significant, allowing you to use, for example, D to delete files without using the Interactive switch, and d to delete them but asking for confirmation first.
- If the ? or $ characters are used to pass arguments, they can accept any character as the argument.

Error Messages

During software loading, error messages indicate if the alias file is too long, or if an I/O error occurs during the time that the alias table is being built. In either case, the table is valid up to the point of the error. The error messages are:

MESSAGE	MEANING
`?Alias file too long`	Alias file longer than 400 characters
`?Unable to read alias file`	Data in alias file is either corrupted or is non-existent.

Standard Aliases

If the file ALIAS.SYS is displayed, it will look like this:

```
f               fup ?
dir             fup ?/l
qdir            fup ?/q
edir            fup ?/e
protect         fup ?/+
unprotect       fup ?/-
pack            fup ?/p
kill            fup ?/di
list            fup KBO:?
assign          fup ?/a
copy            fup $2=$1
?               fup alias.sys
b               basic
e               edit ?
s               set
t               time
```

Each short expression on the left can be used instead of the longer expression on the right. These standard aliases shorten often-used File Utility program commands into brief, easily remembered keystroke sequences.

Each of the aliases supplied on the System disk are explained here. The form of the alias indicates how it is used, and the equivalent shows the keystrokes that would normally have to be entered instead of the alias.

f Form: f {command} Equivalent: FUP
{command}
/X

The first alias, f, uses the ? metacharacter as an argument to the command fup, so that the effect of the alias is to enter the File Utility program, perform the FUP command indicated, and return to FDOS. If there is no argument, the metacharacter acts as a ⟨RETURN⟩, and the effect is simply to enter the File Utility program.

dir Form: dir {pathname} Equivalent: FUP
{pathname} / L
/X

Enters the File Utility program and lists the contents of a device that is indicated by the ? metacharacter.

qdir Form: qdir {pathname} Equivalent: FUP
{pathname} / Q
/X

The most frequently used directory listing command, this alias is the same as dir, except does the Quick Listing of the named device. Use the * wildcard in the filename field to list only the files with the desired extension, or in the extension field to find all the files with a given filename. Note that wildcards are not allowed in the device field.

edir Form: edir {device} Equivalent: FUP
{device} / E
/X

Yields the Extended Directory listing of the named device.

pack Form: pack {device} Equivalent: FUP
{device} / P
/X

Packs the named device to remove ⟨not used⟩ and ⟨temp ent⟩ entries to make more room in the directory.

protect Form: protect {pathname} Equivalent: FUP {pathname}/+ /X

Protects the named files. Use wild cards and the defaults to extend the capabilities of this alias.

unprotect Form: unprotect {pathname} Equivalent: FUP {pathname}/-

Removes the protection of the named files.

kill Form: kill {pathname} Equivalent: FUP {pathname}/DI /X

Enters the File Utility program, and uses the ? metacharacter to delete the named file. Notice that the Interactive switch is used to be certain that confirmation is requested for each file to be deleted. In aliases of this sort, it is important to make use of the Interactive switch, because otherwise, kill *.* would delete all unprotected files without the "really delete...?" message.

list Form: list {filename} Equivalent: FUP KB0:={filename} /X

Enters File Utility, and uses the ? metacharacter to display a named file.

assign Form: assign {device} Equivalent: FUP {device}/a

Enters File Utility, and assigns as SY0: (the System device) the device that matches the ? metacharacter.

copy Form: copy {source} {destination} Equivalent: FUP {file1=file2} /X

This copy alias accepts two arguments: $2 is the source device:file.ext, and $1 is the destination. To use this alias to copy a file called TEST.BAS from MF0: to ED0:, the construct would be:

COPY MF0:TEST.BAS ED0: ⟨RETURN⟩

Notice that this alias contains an ingenious method of getting around the normal system requirement of first specifying the destination and then the source. The technique used in this alias will probably interest you if you have experience in systems that require specifying the source before the destination.

? Form: ? Equivalent: FUP ALIAS.SYS /X

This alias displays the file which holds its own definition.

b BASIC

This alias saves keystrokes, and simply loads the BASIC Interpreter program.

e EDIT ?

Loads the System Editor program, and displays the first lines of the specified file.

s SET

This alias provides a single-keystroke access to the SET Utility program.

t TIME

This alias provides a single-keystroke access to the Time and Date Utiltiy program.

AUTOMATING UTILITY PROGRAMS

The Time and Date Utility

When a new 1752A arrives, one of the first things that is normally done is to use the Time and Date Utility program to set the internal clock. This clock can later be used by programs to perform an activity at a specified time or to report the time when a condition was met or when a piece of data was gathered.

The Time and Date Utility is a machine-language program supplied on the System Disk with the file name TIME.FD2. When TIME.FD2 is loaded into memory, it requests the user to set the time and date of the internal time clock. (See section 3, Software Configuration, for details about how to set the time and date.) When called from an active command file, the Time and Date Utility program does not request input unless status shows that the clock has lost power since the last time it was set.

Using the Time and Date Clock

The command language of the 1752A permits command files to use the Time and Date Utility program. The TIME command can be modified by the arguments -P, -T, and -F.

TIME -P Prints the current setting of the clock.

TIME -T Allows the clock to be set from the Touch-Sensitive display.

TIME -F Forces the clock to display the current settings and wait for input, just as if the clock had not been previously set.

Any combination of the arguments can be used. For example, this line in a command file:

```
TIME -T -F
```

would display the current setting, and request operator input from the Touch-Sensitive display.

If TIME is called as the last command in a command file, it must be followed by a blank line. Otherwise, TIME will always request user input to set the time whether or not the clock has lost power since the last time it was set.

Programming Language Commands

Each programming language available for use with the 1752A provides commands that can be used to read out the time and date. The table below is a synopsis of these commands. For more information, refer to the individual programming manuals.

BASIC

TIME$	Returns the actual time of day.
STIME$	Same as TIME$, but includes seconds.
TIME	Indicates in scientific notation the number of milliseconds since the previous midnight.
DATE$	Returns the actual date.

FORTRAN

TIME	Returns the number of milliseconds since the previous midnight as a long integer (INTEGER *4).
DATE	Returns the date as an integer in a 16-bit format.
ATIME	The current time as an ASCII string in 24 hour format (hh:mm:ss).
ADATE	Current date as a 10-byte ASCII string (dd-Mmm-yy).
ITIME	Returns the current time as an array of three integers in 24 hour format. (hh:mm:ss).
IDATE	Returns the current date as an array of three integers (dd mm yy).

Set Utility Program

Not all of the functions in the Set Utility program are available as programming language commands, so if you need to change any of the port parameters under program control, you can use a command file. Once a program determines which port parameters have to be changed and what the appropriate value of each parameter should be, the program can write this information out to a new command file and execute it.

For example, if a running program has found that the baud rate at KB1: should be set to 19,200, the following BASIC program lines would create and run a command file to accomplish this:

```
1       !printer!
10      OPEN 'PRNTST.CMD' AS NEW FILE 1
20      PRINT #1, 'SET'
30      PRINT #1, 'KB1: BR 19200'
40      PRINT #1, 'EX'
50      PRINT #1, 'BASIC'
60      PRINT #1, 'RUN MYPROG'
70      CLOSE #1
80      EXEC 'PRNTST'
```

NOTE

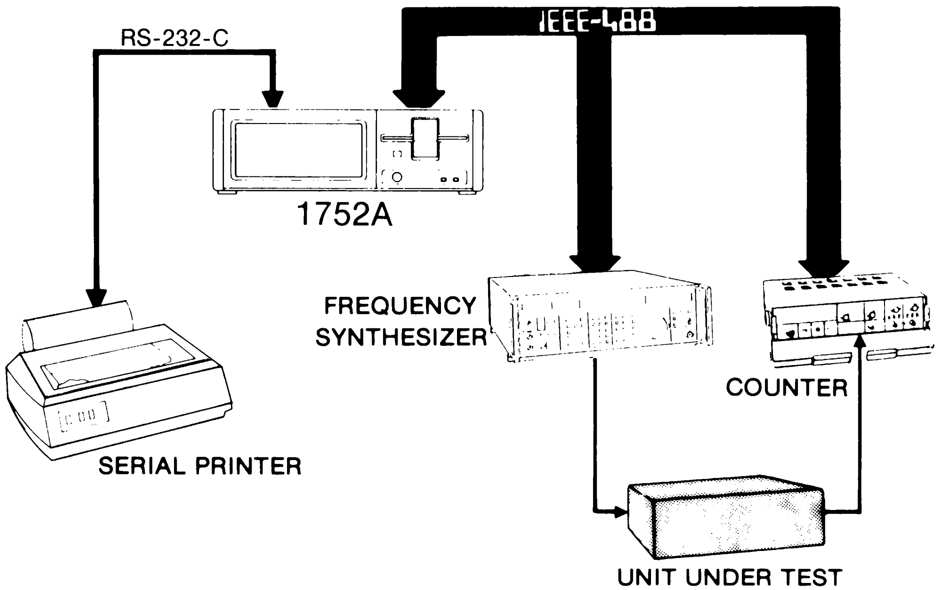
Take some care when using this method of changing port characteristics if simultaneously using the SET SHELL function. If a different shell has been set, it then becomes the job of that shell to interpret all of the commands in an executing command file.

File Utility Program

Like the Set Utility program, not all of the functions of the File Utility program are available as programming language commands. The method described above can also be used for automating File Utilities from a program.

SAMPLE INSTRUMENTATION SYSTEM

In this section, the sample instrumentation system first introduced in section 5 is used once again. Section 5 used it to demonstrate programming techniques using both RS-232-C and IEEE-488 communications. In this section, the sample system is used to demonstrate how a complete program for an IEEE-488 and RS-232-C instrumentation system would be developed.



RS-232 Port:
1776B Serial Impact Printer

IEEE-488 Instrumentation Bus:
1752A (Instrument Controller)
6071A Frequency Synthesizer
1953B Frequency Counter

The Unit Under Test is the undefined subject instrument. It doesn't matter what the UUT is; by its connection we can assume that it will respond to a varying input frequency by varying its output frequency. The Frequency Synthesizer is going to send a signal at some frequency, and the Frequency Counter will read the result, sending it back to the 1752A.

One other instrument is connected in the sample system: the 1776B Serial Impact printer. In this example, the printer will be used to give a hardcopy of the test results. The printer, of course, is not necessary for system operation, but is included to fill out the example program we will be developing.

Controlling The Sample System

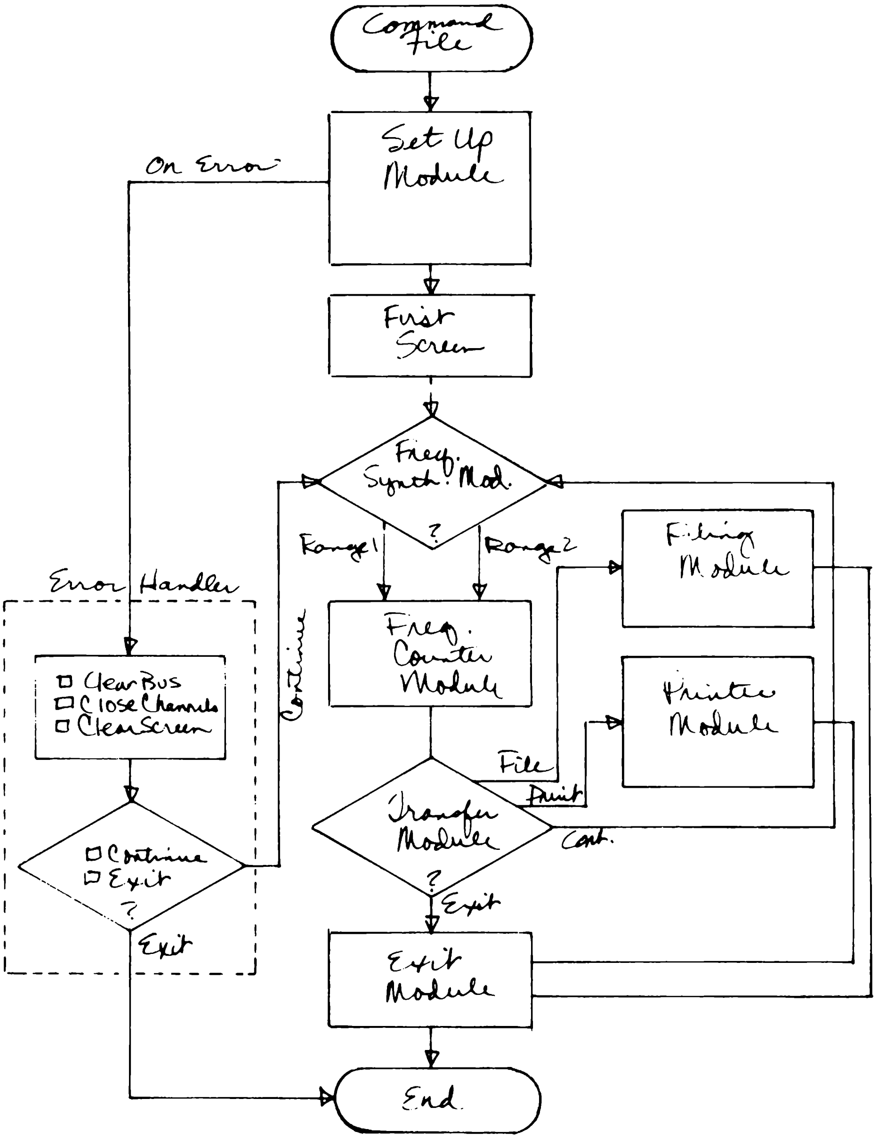
Step 1: Start With a Flowchart

Developing a flowchart is a necessary part of developing software; unfortunately, it is often not done because it sounds like an easy task to string together a few lines of code and get some kind of meaningful result. However, there are several good reasons to write a flowchart before writing a program:

- First, with a flowchart to guide your efforts, you will write more efficient programs, and be less likely to get sidetracked.
- Second, program debugging can be greatly simplified if a flowchart is available that shows what the program is supposed to do.
- Third, the flowchart provides a valuable piece of documentation if you want to modify or use parts of the program later.
- Fourth, the flowchart will help others understand and/or modify your program.

The flowchart is a graphical representation of how the program will proceed. It translates an algorithm into a visual aid, so that the interactions among the various parts of the program can easily be seen. It also provides a valuable first step in the programming task, because it breaks the job down into manageable modules, each of which can be written in order, and then linked together.

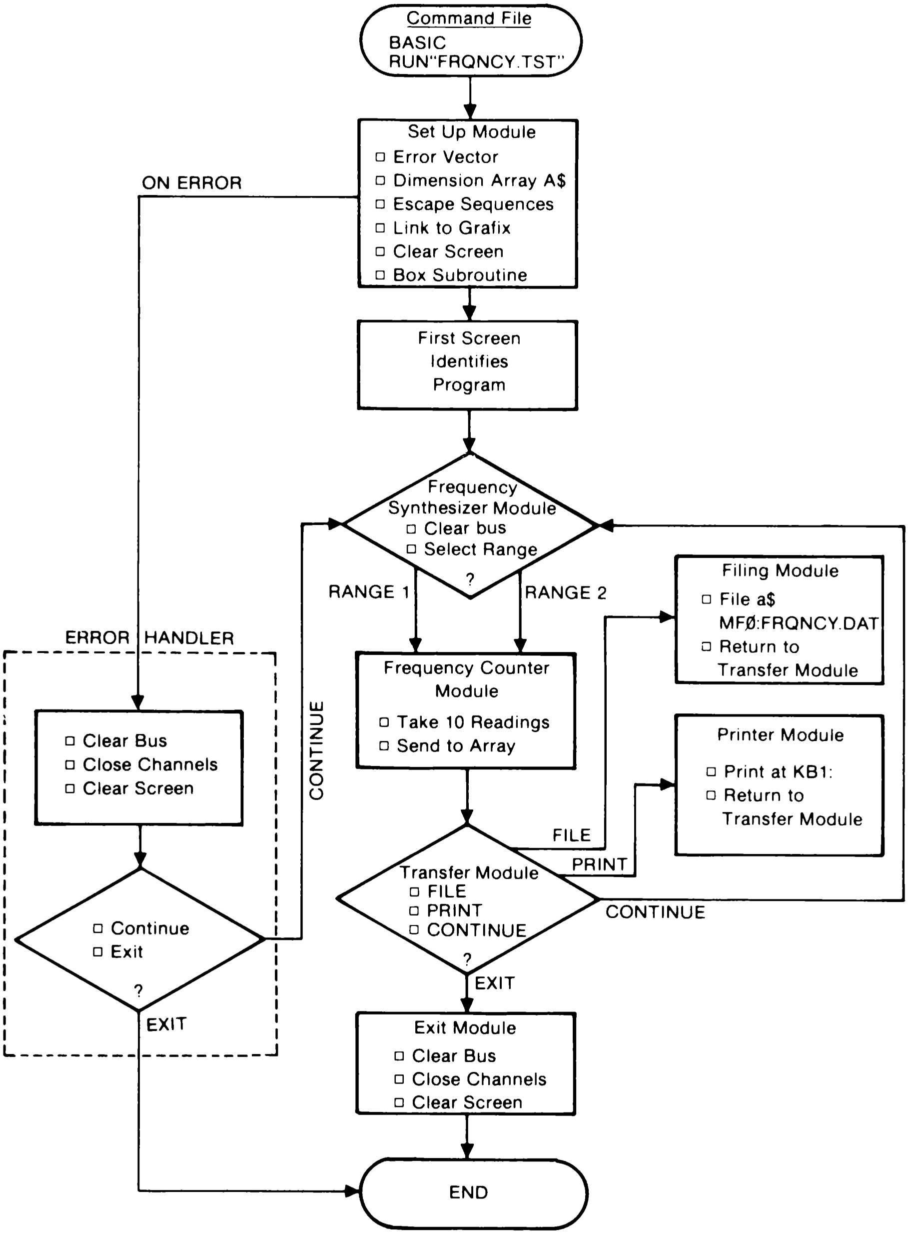
The flowchart for the sample instrumentation system might look something like this:



As program development progresses, update the flowchart to incorporate new ideas and program capabilitites. While the example program was being written, four flowcharts were drawn, each one showing slightly more refinement and detail.

When the program was complete, a final flowchart was drawn to show how the actual program worked. The first flowchart can be thought of simply as a guide for program writing, and the final flowchart as a document that shows how the program operates.

The final flowchart for the sample is shown on the next page.


Command File
BASIC
RUN"FRQNCY.TST"
Set Up Module
□ Error Vector
□ Dimension Array A$
□ Escape Sequences
□ Link to Grafix
□ Clear Screen
□ Box Subroutine
ON ERROR
First Screen
Identifies
Program
Frequency
Synthesizer Module
□ Clear bus
□ Select Range
?
RANGE 1
RANGE 2
Filing Module
□ File a$
MFØ:FRQNCY.DAT
□ Return to
Transfer Module
ERROR HANDLER
Frequency Counter
Module
□ Take 10 Readings
□ Send to Array
Printer Module
□ Print at KB1:
□ Return to
Transfer Module
□ Clear Bus
□ Close Channels
□ Clear Screen
CONTINUE
FILE
PRINT
Transfer Module
□ FILE
□ PRINT
□ CONTINUE
?
CONTINUE
□ Continue
□ Exit
?
EXIT
EXIT
Exit Module
□ Clear Bus
□ Close Channels
□ Clear Screen
END

Step 2: Establish Bus Addresses

The next step in developing a program for this system will be to establish the addresses of all the IEEE-488 instruments. All Fluke instruments are set by rear-panel switches, and usually conform to an easily-remembered scheme. In both the Synthesizer and Counter of this example, the address switches are binary-weighted; the manual provided with the instrument gives complete details on how to set its address.

In the sample system, bus addresses are set up as follows:

01 Frequency Synthesizer
02 Frequency Counter

Step 3: Program the Modules

Programming even this simple system can be too complex without proper planning. Be judicious; program by parts, then link the parts together. In this example, follow the flowchart to program the various modules.

Setup Module

The Setup Module starts the program by establishing variables and strings that will be used later. This module is sometimes thought of as "housekeeping". Here are some of the things that such a module can be used for:

- Establish the location of the error handler routine.
- Assign the name of the array that will hold the test data.
- Assign escape sequences as variables to saving typing later in the program.
- Link to an object file, so that there is no need to access the disk later when the file is used.
- Define subroutines that will be used throughout the program. This program will use such a subroutine to draw the operator selection boxes using the graphics plane.

First Screen

For most programs, it is a good practice to incorporate an introductory screen to identify the program to the operator, and to ask for the screen to be touched when the test setup is ready. This insures that the correct disk is loaded, and gives the operator confidence that everything is under control.

Frequency Synthesizer Module

This module is the first selection screen, and will do these things:

- Present the range selections available.
- Provide an exit selection in case the operator notices that the wrong program has been loaded, or for some other reason wants to quit testing when this screen is displayed.
- Accept the selection, and use it to: 1) Address the Synthesizer as a listener, 2) put it into remote mode so it can be set by command rather than its front panel switches, and 3) program the desired output frequency.
- Ask the operator to wait while the synthesizer is being programmed.

Frequency Counter Module

When the synthesizer has been programmed, the counter will be told to begin reading the frequency output by the UUT and report the resulting measurement data back to the 1752A. This portion of the program will do the following:

- Indicate that the test is in progress.
- Address the Counter as a talker.
- Put the Counter into remote mode so it will be set by 1752A commands rather than by its front panel switches.
- Trigger the readings.
- Collect the resultant data.

Transfer Module

The Transfer Module is included so the operator can select where to transfer the measurement data. It can be filed and/or printed, or discarded by continuing the test or exiting. This module will:

- Display the test results.
- Display selections for the Operator and accept the selection to branch to the File or Print modules, to continue the program at the Frequency Synthesizer Module, or to exit with no further testing.

Filing Module

If the operator elects to file the data, a channel is opened to the floppy disk, and the data is recorded there. When filing is complete, the program returns to the Transfer Module. Notice that in this sample system, the data can only be filed once. In a real application, a virtual array would probably be used in order to increase the amount of test data that could be filed.

Printer Module

If the operator wishes to print the data, the program branches to the Printer Module, and returns to the Transfer Module. In a real application, this module would probably send an entire data file to the printer, rather than just the array that contains the single set of test data.

Error Handler

This module insures that if an error occurs while the program is running, the program itself can handle it, rather than halting. It displays a message to the operator that an error has occurred, some information about the error, and requests a decision whether to continue or exit. If Continue is selected, the program returns to the Frequency Synthesizer Module. If Exit is selected, it goes immediately to the exit module.

Exit Module

The Exit Module can be as short as one line: END. In this program, however, two lines are used that include six separate commands, five of which are "housekeeping". The housekeeping commands clear the bus, close all channels, and erase the display.

Step 4: Concatenate

Now that each portion of the program has been written and found to operate properly, they are linked together, or concatenated. During the writing phase, the line numbers in each of the modules were given the same first digit, and they were assigned in the range that the eventual program would use them. For example, the Setup Module was assigned numbers in the 100 to 199 range; the frequency counter module 200 to 399, and so on.

Each module was recorded to a filename that recalled its place in the order of things: Setup module to NEW1.BAS, First Screen module to NEW2.BAS, Frequency Synthesizer module to NEW3.BAS, and so forth. It is a simple matter to use the File Utility program to merge the modules. Then, when the program is loaded using the BASIC Interpreter program, the line numbers can be renumbered using the REN statement.

Step 5: Debugging

Many programmers write all the code, then start debugging. While this approach may seem to be more efficient, in the long run it only leads to trouble. Fix the little problems as they arise so they are less likely to have larger effects later on.

For example, the program might seem to work, but have you tried asking the Synthesizer to output maximum frequency? Minimum? Before a program for this system is really up and running, we have to be sure that it will accept any parameters. Perhaps the operator can be instructed not to choose the highest range of the Synthesizer, because the program still needs a little work. It is probably a better idea, though, to eliminate those selections if that part of the program is still being tested.

Unfortunately, debugging is something of an arcane art. The techniques used are not easily taught or learned, so most programmers develop techniques of their own through experience. The BASIC Programming Manual includes a section on debugging, and discusses many proven techniques.

One technique is to "comment out" the program lines that cannot be used until the actual system is set up. All that is involved is to put an exclamation point immediately after the line number that you don't want as part of the program. During the development of the Frequency Test program, for example, all the lines containing IEEE-488 commands were commented out because they would cause errors until the IEEE instruments were connected. This technique leaves the program relatively intact, and it is a simple matter to re-include any dropped lines later.

Another technique is to insert GOTO statements to route program execution around known areas. This could be used, for example, to not include the First Screen module to save time. Another frequently used technique is to put a RETURN statement at the first line of an unfinished subroutine.

Step 6: Document the Program

Be sure to adequately document your work. This is one of the last steps, but one which you can work on as the program evolves. No program should be undocumented. This means you! Someday, you will no longer be with the Bitty Widget Corporation, but will have moved on to the Mighty Widget Programming Consortium. What happens to program maintenance when the Author leaves, but his programs don't give a hint as to how they work? As technology advances, all your efforts will be thrown out if someone else is not able to look at the code and see what you had in mind.

Not every line needs a comment, but leave some clues at least. Every programmer, at some time, has needed to re-invent a routine simply because it was more work to figure out what his predecessor had done than it was to start from scratch.

```
10   !   * * *   F R E Q U E N C Y   T E S T   P R O G R A M   * * *
20   !
30   !
40   ! <your name>
50   ! <date written>
60   ! <identification>  [test disk: filename frqncy.tst]
70   !
80   ! This program performs a frequency test in two ranges for the
90   ! 6070/1A frequency synthesizer and a 1953A frequency counter.
100  ! Notes:
110  !  -  The 1953 Counter must have option C installed for Range 2.
120  !  -  Range 1 is 20 MHz output, range 2 is 200 MHz.
130  !  -  The synthesizer is IEEE-488 address 01, the counter is 02.
140  !
150  ! The program takes ten measurements and stores them in an array.
160  ! Then the results are displayed, and the operator decides whether
170  ! to file or print the array, continue testing, or exit.
180  !
190  ! All operator selections are made by touching the screen.
200  !
210  !             - Setup Module -
220  !  Error vector, array for data, escape sequences, link to "graph"
230  !
240  ON ERROR GOTO 1510
250  DIM A(11%)                        ! main memory array to rcv data
260  BP$ = CHR$(7)                     ! beep
270  ES$ = CHR$(27) + "[2J"            ! erase screen
280  PRINT ES$;CHR$(27)+"[?8l";        ! clear the screen, disable cursor
290  LINK "GRAPH.OBJ"                  ! link to graphics object file
300  GRPOFF \ ERAGRP (0%)              ! graphics plane off, then erase
310  GOTO 470                          ! display first screen
320  !
330  !   Selection Box Subroutine
340  !   Draws a box around TSO keys
350  !
360  MOVE (XG%,YG%)                    ! current position is defined
370  MOVE (XG%,YG%)                    ! current position is defined
380  PLOTR (80%, 0%, 1%)               ! prior to the gosub commands
390  PLOTR (0%, -50%, 1%)
400  PLOTR (-80%, 0%, 1%)              ! box is 80 x 50 pixels
410  PLOTR (0%, 50%, 1%)
420  GRPON
430  RETURN
440  !
450  !              - First Screen -
460  !
470  PRINT BP$; ES$; CPOS (6,34);"FREQUENCY TEST";
480  PRINT CPOS (8,20);"Check all connections, and apply power to the";
490  PRINT CPOS (9,20);"test instruments. Touch the screen when ready.";
500  KR% = KEY \ WAIT FOR KEY \ KR% = KEY   ! reset tso \ wait \ get key
510  !
520  !        - Frequency Synthesizer Module -
530  !
540  PRINT ES$; \ GRPOFF \ ERAGRP (0%)
550  XG% = 360% \ YG% = 150% \ GOSUB 360
560  XG% = 360% \ YG% = 100% \ GOSUB 360          ! draw three boxes
570  XG% = 360% \ YG% = 50% \ GOSUB 360
580  PRINT CPOS(2,20);"PLEASE SELECT FREQUENCY SYNTHESIZER RANGE";
590  PRINT CPOS (7,25); "RANGE 1:   20 MHz";
```

```
600   PRINT CPOS (11,25);"RANGE 2:  200 MHz";
610   PRINT CPOS (15,37); "EXIT";
620   KR% = KEY \ WAIT FOR KEY \ KR% = KEY              ! get response
630   IF KR% = 17% OR KR% = 27% THEN 680                ! range 1 selected
640   IF KR% = 37% OR KR% = 47% THEN 710                ! range 2 selected
650   IF KR% = 56% OR KR% = 57% OR KR% = 58% THEN 1470          ! exit
660   PRINT BP$; ES$                                    ! invalid - repeat
670   WAIT 666 \ PRINT BP$; \ GOTO 550
680   R% = 1                                       ! range 1 program data:
690   PD$ = "FR20MZ,AP1V"                          ! 20 MHz @ 1.0 v
700   GOTO 710
710   R% = 2                                       ! range 2 program data:
720   PD$ = "FR200MZ,AP1V"                         ! 200 MHz @ 1.0 v
730   !
740   ! Selection made, so program instruments
750   !
760   GRPOFF \ ERAGRP (0%)
770   PRINT ES$; CPOS (4,32); " - PLEASE WAIT - ";
780   PRINT CPOS (6,23);"Programming Synthesizer for Range ";R%
790   INIT PORT 0                             ! initialize the bus
800   CLEAR @1                                ! clear synthesizer
810   REMOTE @ 1 @ 2                          ! both to remote
820   PRINT @1, PD$                           ! send program data
830   WAIT 666
840   !
850   !          - Frequency Counter Module -
860   !
870   PRINT CPOS (6,15);"                Measurement In Progress        ";
880   CLEAR @2
890   FOR I% = 1 TO 10
900      PRINT @2, "FOR2AOSOM1H1T"   ! chnnl A, 10ms, ac couple, one sample
910   !                              ! sep. output, SRQ, trigger
920      PRINT CPOS(16,40);I%;       ! displays which count is being done
930      WAIT 500                    ! takes a reading every half second
940      INPUT @2, A(I%)             ! puts the measurements into array
950   NEXT I%
960   !
970   !             - Transfer Module -
980   !
990   ERAGRP (0%) \ GRPOFF
1000   PRINT BP$; ES$; CPOS (2,1);"TEST RESULTS:";   ! format for display:
1010   PRINT CPOS (4,0)                              ! 2 rows, 5 columns
1020   PRINT USING "##.####^^^^", A(1..5),           ! array elements 1-5
1030   PRINT USING "##.####^^^^", A(6..10),          ! array elements 6-10
1040   !
1050   XG% =  76% \ YG% = 90% \ GOSUB 360
1060   XG% = 225% \ YG% = 90% \ GOSUB 360
1070   XG% = 376% \ YG% = 90% \ GOSUB 360                 ! draw the boxes
1080   XG% = 530% \ YG% = 90% \ GOSUB 360
1090   PRINT CPOS (12,14); "FILE";CPOS(12,32);"PRINT";CPOS(12,49);"CONTINUE"
1100   PRINT CPOS (12,70); "EXIT"
1110   PRINT CPOS (16,33); "Please Touch Selection";
1120   KR% = KEY \ WAIT FOR KEY \ KR% = KEY
1130   IF KR% = 41% OR KR% = 42% THEN 1210       ! file A in "FRQNCY.DAT"
1140   IF KR% = 44% OR KR% = 45% THEN 1320       ! print the array
1150   IF KR% = 47% OR KR% = 48% THEN 540        ! back to test
1160   IF KR% = 49% OR KR% = 50% THEN 1460       ! exit
1170   PRINT ES$; BP$; \ WAIT 666 \ GOTO 1000    ! invalid - repeat
1180   !
```

```
1190  !                 - Filing Module -
1200  !
1210  PRINT ES$; CPOS (4,30);"One moment, filing";
1220  WAIT 2000
1230  CLOSE 1
1240  OPEN "MFO:FRQNCY.DAT" AS NEW FILE 1       ! filename "frqncy.dat"
1250  PRINT #1, USING "##.####^^^^", A(1..10)   ! store array elements 1-10
1260  PRINT ES$; CPOS (4,20); "Data has been filed - returning to menu";
1270  !
1280  WAIT 750 \ GOTO 990                       ! return to transfer module
1290  !
1300  !                - Printer Module _
1310  !
1320  GRPOFF
1330  PRINT ES$, CPOS (4,30); "One moment, printing";
1340  !
1350  CLOSE 1                       !   -   -   -   note:   -   -   -   !
1360  OPEN 'KB1:' AS NEW FILE 1     !  be sure to check port parameters !
1370  PRINT #1, CHR$(12%)           !  before making this selection.    !
1380  PRINT #1, "TEST RESULTS"
1390  PRINT #1, USING "##.####^^^^", A(1..10)
1400  PRINT ES$; CPOS(4,20); "Data has been printed - returning to menu";
1410  CLOSE 1
1420  WAIT 750 \ GOTO 990
1430  !
1440  !                 - Exit Module -
1450  !
1460  CLEAR @1 \ CLOSE ALL
1470  PRINT ES$; CHR$(27)+"[?8h" \ GRPOFF \  ERAGRP (0%) \ END
1480  !
1490  !               - Error Handler Module -
1500  !
1510  GRPOFF \ ERAGRP (0%) \ PRINT ES$
1520  XG% = 300% \ YG% = 130% \ GOSUB 360               ! draw the boxes
1530  XG% = 300% \ YG% =  60% \ GOSUB 360
1540  PRINT CPOS (3,5); "System Error -";
1550  PRINT CPOS (4,5); "Check instruments and connections";
1560  PRINT CPOS (5,5); "before continuing.";
1570  PRINT BP$; CPOS (10,40); "CONTINUE";
1580  PRINT CPOS (15,42); "EXIT";
1590  KR% = KEY \ WAIT FOR KEY \ KR% = KEY
1600  IF KR% = 26%  OR KR% = 36% THEN 1610 ELSE 1620    ! continue
1610  GRPOFF \ ERAGRP (0%) \ PRINT ES$ \ RESUME 990
1620  IF KR% = 46% OR KR% = 56% THEN RESUME 1470        ! exit
1630  !
1640  PRINT ES$; BP$; \ WAIT 666 \ GOTO 1550            ! invalid - repeat
1650  !
```

The Startup Command File

One final thing is needed to make the resulting program truly stand-alone: the Startup Command file that will make all the rest happen from power up. In this sample programming session, the Command file will be the last thing written. It is a relatively minor portion of the entire task; but without it, it would be necessary to leave the keyboard attached to get the program operating. With it, the keyboard can be detached entirely, and the program will run on its own whenever the disk is loaded and power applied.

To complete the automation of this program, the Startup command file must do two things:

- Load the BASIC Interpreter program.
- Run the Frequency Test program, FRQTST.BAS

Here is what the file STRTUP.CMD should look like:

```
BASIC
RUN "FRQTST"
```

A CAUTION TO SYSTEMS PROGRAMMERS

The 1752A Instrument Controller is sometimes used to control other equipment that can generate potentially hazardous conditions (source a high voltage, for example). Care must be taken by the programmer to insure that, if a program is terminated in the middle of a critical function, the equipment will be programmed to remove the hazardous condition.

Situations that can terminate a program prematurely include, but are not limited to:

1. ⟨CTRL⟩/C or ⟨CTRL⟩/P entered from the keyboard
2. RESTART and/or ABORT buttons pressed on the 1752A front panel
3. Momentary power failure
4. Unexpected error conditions

The ⟨CTRL⟩/C or ABORT can be trapped from within a BASIC program using the ON ⟨CTRL⟩/C GOTO statement. An exception handler can be set up from within an Assembly Language, FORTRAN, or C program, using the FDOS call, to trap an ABORT, ⟨CTRL⟩/C or ⟨CTRL⟩/P. These special handlers can then take whatever action is necessary to remove the hazardous condition. Information about exception handling can be found in Section 11 of the BASIC Programming Manual and Section 6 of the Assembly Language Programming Manual.

Under certain conditions ⟨CTRL⟩/P will not be trapped, and the user program will exit to FDOS. Whenever a user program is terminated with ⟨CTRL⟩/P, the SET SHELL command may be used to cause a special shell program to be executed instead of returning to FDOS. This shell program can then initialize all instruments in the system to a safe condition. Information about the SET SHELL command can be found elsewhere in this section and in the BASIC Reference Manual.

If the RESTART button is pressed, or a power failure occurs, the 1752A will reload the system software and the user program is lost. In this case, a startup command file can be executed to automatically run an initialization program similar to the shell program mentioned above. Information about using command files can be found elsewhere in this section.

A general error handler should be included in the application program to handle unexpected error conditions. Section 11 of the BASIC Programming Manual contains information about error handling techniques.

CONCLUSION

This section has given guidelines on automating the 1752A. It has used a sample instrumentation system to show how to write and develop programs that will be useful for a system of this sort. Larger systems are of course possible; in fact up to 15 instruments can be connected onto the same IEEE-488 bus. If another controller is one of them, 15 more are possible, and so on. No matter how big the final system is to be, the guidelines given here should make the programming task much easier.

Start with a flowchart. It will be a valuable guide once you're down inside all those GOSUBs and FOR-NEXTs. Program in small amounts, then concatenate. Test each module; test each parameter; test each selection. Make sure the program works as you designed it to do. Finally, document your efforts. If you stay with the same company, your task will be much easier if you don't have to relearn the program before you can update it. If you leave the company, your successor will have a lot less trouble figuring out how the program was supposed to operate.

Section 8
Display

CONTENTS

Introduction 8-2
The Character Plane 8-3
Character Sets 8-3
Custom Character Sets 8-3
Character Graphics 8-4
Programming a Character Graphics Display 8-6
Introduction to ANSI Standards 8-8
Special Display Control Characters 8-9
Escape Sequences 8-10
The Graphics Plane 8-18
Introduction to Graphics Routines 8-18
Addressing the Pixel Locations 8-21
Graphics Routines 8-21
Summary of Commands 8-22
Printing the Graphics Plane 8-40
Graphics Print Routines 8-41
Summary of Graphics Print Commands 8-41
Printing the Graphics Plane on Unsupported Printers 8-49
Conclusion 8-53

INTRODUCTION

The 1752A Data Acquisition System features a display that can provide a great deal of visual information to the operator. Dot-addressable graphics, coupled with the state-of-the-art touch-sensitive display, makes it possible to design displays that are meaningful and interesting, and that provide a high degree of interaction between the operator and the 1752A.

The 1752A supports all of the features incorporated in the Fluke 1720A, but it expands the display capabilities by using more of the ANSI Standard controls and by incorporating a set of graphics routines contained in the object files "GRAPH.OBJ" and "GPRINT.OBJ" on the System Disk.

The 1752A includes 256 displayable characters in two character sets. The 128 characters in the standard set are the full ASCII set, and the alternate character set can be customized for characters in languages other than English, or for custom applications such as logos or other special symbols.

THE CHARACTER PLANE

Display information is stored in two separate sections of memory, the character plane and the graphics plane. Each display memory is independent; that is, they can be enabled or disabled separately. When both are enabled, displayed characters can be made either opaque or transparent to the graphics portion of the display.

A pad of 50 Programming Worksheets is provided with the 1752A. The grids printed on the sheets are helpful in the design of displays that use the touch-sensitive display. Columns and rows are indicated for both normal- and double-size characters, and each of the 60 touch-sensitive key locations is clearly marked.

Character Sets

The character set EPROM contains two character sets. The primary character set is ASCII, with some Greek characters and commonly used symbols. The table in appendix F shows the display responses for the primary character set.

Custom Character Sets

Depending on the revision level of the Video-Graphics-Keyboard module, the alternate character set may be a duplicate of the primary set, or may be a selection of non-English characters and additional symbols.

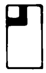
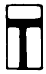
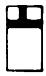
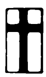
The character EPROM is a readily available 2732 type, which can be programmed with very little effort to display any character set desired. Appendix E of this manual explains how to design a custom character set, and includes a short BASIC program that displays all the characters in both character sets. When the program is run, the primary set is displayed in double size characters. When you touch the screen, the alternate set is displayed. The program toggles between displaying first one set then the other, to allow you to compare them.

Character Graphics

Besides individual characters, straight horizontal and vertical lines can also be included in displays on the character plane. An example of this usage is the program "MASTER.BAS" on the Getting Started disk. This program uses only the character plane. The graphics plane could easily display this grid, but is only needed for displays where diagonal lines or motion simulation are used, as in the program called "WOW" on the Getting Started Disk. In fact, the program titled "TOUCH" on the Getting Started Disk displays all 60 touch sense locations using the Graphics plane. The Graphics plane is discussed in more detail later in the section.

The table on the next page shows the graphics characters that are contained in the character set EPROM. With character graphics enabled, the characters 0 through 9 and the colon (:) result in the display of these symbols.

1722A DISPLAY RESPONSE

CHARACTER	NORMAL SIZE	DOUBLE SIZE	FUNCTION
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 or ESC [?3h
2. To disable Graphics Mode, send the display ESC [2p or ESC [?3l
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.

Programming a Character Graphics Display

Two sample Interpreted BASIC programs are used here to illustrate how to create a display using the 1752A Character Graphics capability. The first program displays all of the graphics characters, first in normal size, then in double size. The second program uses the characters to display an area the size of approximately one touch-sensitive key. Both of the programs make use of escape sequences to clear the screen and put it into character graphics mode. The first program also uses an escape sequence to make use of double-size character mode. These control sequences are described in more detail later in the section.

Program to Display Graphics Characters

```
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 20
50   PRINT " 0  1  2  3  4  5  6  7  8  9  : "
60   PRINT\ PRINT\ PRINT "         Please Touch the Screen";
70   WAIT FOR KEY \ K% = KEY \ RETURN
```

In the first program, the first line establishes an escape sequence variable E$ as ASCII 27 + the left bracket character. This sequence saves typing the entire escape sequence later in the program. The second part of line 10 establishes a variable CL$ (clear) as the escape sequence just defined (〈ESC〉) + "[2J". This sequence will be used later in the program to clear the screen before each of the displays.

Line 20 clears the screen, then enables the graphics display (3p). The line ends by sending the program to the subroutine at line 50, which now prints the characters 0 through 9 and the colon (:), resulting in a display of the 10 graphics characters.

When the subroutine is stopped by touching the screen, the program returns to line 30, where the screen is again cleared, and the escape sequence given the parameters to go to double size, and to once again enable the graphics display (1;3p).

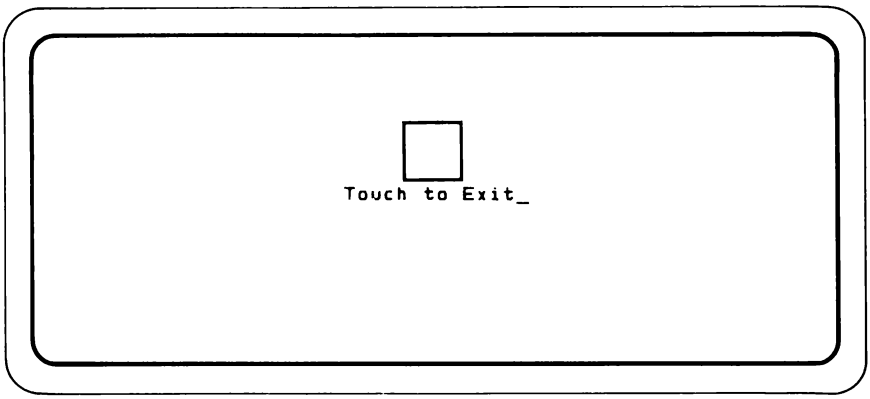
The display toggles between normal- and double-size characters for comparison. The program stops when 〈CTRL〉/C is pressed.

Program to Display One Touch-Sense Keypad

This second sample program uses the graphics characters to draw a box the size of one touch-sense keypad. Notice that the program begins by assigning the escape sequences.

```
10   E$ = CHR$(27) + "[" \ CL$ = E$ + "2J"
20   PRINT CL$; E$ + "3p";
30   PRINT CPOS(4,40); "1888880"
40   PRINT CPOS(5,40); "9     9"
50   PRINT CPOS(6,40); "3888882"
60   PRINT CPOS(8,37); "Touch to exit"
70   WAIT FOR KEY
80   PRINT E$ + "2p"
90   PRINT CPOS(8,37); "  Thank you.  "
100   WAIT 2000 \ PRINT CL$ \ END
```

The display resulting from this sample program looks like this:



INTRODUCTION TO ANSI STANDARDS

The American National Standards Institute publishes ANSI Standard X3.4, which describes the American Standard Code for Information Interchange, or ASCII. Since its initial publication in 1968, ANSI X3.4 has become the industry accepted standard for defining a 7-bit character code.

Another ANSI standard, X3.41, describes recommended code extension techniques for use with ASCII. In essence, the standard specifies how to represent ASCII in an 8-bit environment.

The 1752A Data Acquisition System implements both ANSI standard X3.4 and applicable code extension techniques from ANSI standard X3.41. This compliance assures the upward compatibility of Fluke products as well as the ability of the 1752A to communicate effectively with the products of other manufacturers.

Special Display Control Characters

Eleven of the ASCII characters are interpreted by the Video-Graphics-Keyboard module (VGK) as display control characters. In addition to these eleven control characters, two others (ASCII 24 and 26, CANCEL and SUBSTITUTE) are used by the system's microprocessor to cancel a display control. These codes are not sent on to the VGK module.

The display control characters as summarized in the next table, must be preceeded by an escape sequence.

ASCII DISPLAY CONTROL CHARACTERS

〈CTRL〉	ASCII	MNEMONIC	RESULT
	0	NUL	Null; no action.
G	7	BEL	Sounds a tone.
H	8	BS	Moves the cursor left one column, if it is not already positioned at the leftmost column.
I	9	HT	Moves the cursor to the next tab stop, (every 8 columns).
J K L	10 11 12	LF VT FF	All of these commands move the cursor to the next line down in the same column. The display scrolls up if the cursor is on the bottom line.
M	13	CR	Moves the cursor to the beginning of the current line.
N	14	SO	Selects alternate character set.
O	15	SI	Selects the standard character set.
	24	CAN	Cancels a display control.
	26	SUB	Substitutes a character sequence if sent as part of the sequence.
	27	ESC	Starts a display control character sequence.

Escape Sequences

Besides display control characters, the ANSI Standard also specifies a set of Code Extension Techniques (Escape Sequences) which can be used in controlling the display. These techniques use commands in the form:

⟨ESC⟩ [{parameter 1} ; {parameter 2} ; {parameter n} {terminator}

- The ⟨ESC⟩ [is called the Control Sequence Identifier. All control sequences except scrolling commands begin with this identifier.

- The parameters may be either numeric or selective. If the sequence uses numeric parameters, and no number is given, zero is normally assumed. Cursor controls assume 1 if no number is given.

- The terminator is always an alphabetic character.

Any number of commands can be specified within a given command set as long as each is separated by a semicolon (;). Ill-formed parameters are ignored.

Numerically-Defined Display Control Sequences

The 1752A uses the same display controls as those used for the Fluke 1720A, and some additional ones. Many of the sequences shown in the table on the next page are equivalent to ANSI Standard Selective Parameter Sequences, which are discussed later in this section.

The default conditions are indicated by an asterisk (*).

DISPLAY CONTROL SEQUENCES

FUNCTION	CONTROL SEQUENCE	COMMENTS
Cursor Controls		
Up n lines	⟨ESC⟩[nA	The cursor stops at the edge if the number given as an argument results in movement past the edge of the screen.
Down n lines	⟨ESC⟩[nB	
Right n columns	⟨ESC⟩[nC	
Left n columns	⟨ESC⟩[nD	
Direct to line, column	⟨ESC⟩[l; c H	
Scroll down one line	⟨ESC⟩D	
Scroll up one line	⟨ESC⟩M	
Scroll to start new line	⟨ESC⟩E	
Erasing		
To end of display	⟨ESC⟩[J or ⟨ESC⟩[0J	
To start of display	⟨ESC⟩[1J	
All of display	⟨ESC⟩[2J	
To end of line	⟨ESC⟩[K or ⟨ESC⟩[0K	
To start of line	⟨ESC⟩[1K	
All of line	⟨ESC⟩[2K	
Attributes		
Attributes Off*	⟨ESC⟩[m or ⟨ESC⟩[0m	
High Intensity	⟨ESC⟩[1m	
Underline	⟨ESC⟩[4m	
Blinking	⟨ESC⟩[5m	
Reverse Image	⟨ESC⟩[7m	
Cursor Status		
Request cursor position	⟨ESC⟩[6n	For a program to make use of the report, a logical input channel must exist between the program and KB0:
Cursor position report	⟨ESC⟩[l, c R	
Size of Characters		
Normal	⟨ESC⟩[p or ⟨ESC⟩[0p	These commands affect the entire display.
Double	⟨ESC⟩[lp	
Character Graphics		
Disabled*	⟨ESC⟩[2p	These commands also affect the graphics plane.
Enabled	⟨ESC⟩[3p	

DISPLAY CONTROL SEQUENCES (cont)

FUNCTION	CONTROL SEQUENCE	COMMENTS
Keyboard		
Enabled*	⟨ESC⟩[4p	Even when disabled, the keyboard can respond to control codes. To exit a locked condition, use ⟨CTRL⟩/T to unlock the keyboard, reset the screen to normal-size characters, home the cursor (upper left), and disable the graphics plane.
Disabled	⟨ESC⟩[5p	
Cursor Type		
Blinking Underscore*	⟨ESC⟩[0x	
Steady Underscore	⟨ESC⟩[1x	
Blinking Block	⟨ESC⟩[2x	
Steady Block	⟨ESC⟩[3x	

*Indicates the default conditions.

Selective Parameters

Selective parameters are defined with a number followed by a number. These parameters are always a string with the first character a question mark (?), and the second a numeric character between 1 and 8. The terminator is either the lower case letter 'h' (SET or high), or the lower-case letter 'l' (RESET or low). As with numerically defined parameters, selective parameters are always started with the "Control Sequence Identifier", ⟨ESC⟩[.

- ☐ To SET a mode, the terminator is the lower case letter 'h'.
- ☐ To RESET a mode, terminate with a lower case letter 'l'.

The table below summarizes the ANSI Standard Selective parameters. The defaults are indicated by an asterisk (*).

SUMMARY OF MODE SELECTIONS

MODE	RESET (l)	SET (h)
?1	Field Attributes*	Character Attributes
?2	Single Size*	Double Size
?3	Disable Character Graphics*	Enable Character Graphics
?4	Keyboard Unlocked*	Keyboard Locked
?5	Opaque to Graphics*	Transparent to Graphics
?6	Disable Character Display	Enable Character Display*
?7	Disable Graphics Display	Enable Graphics Display*
?8	Disable Cursor Display	Enable Cursor Display*

*Indicates the default conditions.

Field Attributes

The field attributes are identical to the non-transparent field attributes used on the 1720A. When this mode is RESET, all attributes, such as blinking or inverse video, are defined for a field on the display before the characters themselves are placed there. Both Field and Character attributes use the numeric parameter escape sequences.

NOTE:
The refresh scanning rate exceeds the rate that characters are written to the screen. Therefore, in the field attribute mode, the underlining and reverse image commands will cause the entire remaining display to momentarily exhibit the attribute until the attributes-off command (⟨ESC⟩[m or ⟨ESC⟩[0m) is recieved.

To avoid the "flashing" associated with this phenomenon, first position the cursor to the location for the attributes-off command, then return it to the location for the attributes and the message to be displayed.

The entire display is either in field or character attribute mode; field mode is the default.

Character Attributes

In the character attribute mode, attributes are associated with individual characters rather than with an area of the display. These attributes do not use a display position, so it is possible to highlight a single character within a word, for example.

If the ANSI Standard Mode "?1" is SET (h), then 1720A-type field attributes are disabled, and character attributes are enabled.

In the character attribute mode, displays need not include the leading and trailing spaces on the display associated with field attributes. Additionally, new enhancements can be added without resetting previously set ones. (They are "sticky".) As new enhancements are sent to the display, they are added to an already existing list until they are reset by the reset enhancements command ⟨ESC⟩ [0m. When the set mode command is first received, the screen is cleared, and the cursor is homed.

As shown in the next table, many of the selective parameters have equivalent numerically defined parameter control sequences. The default conditions are indicated by an asterisk (*).

SELECTIVE PARAMETER DISPLAY CONTROLS

FUNCTION	MODE SELECTION	EQUIVALENT NUMERIC SEQUENCE
Attribute Mode		No equivalent.
Field*	⟨ESC⟩[?1l	
Character	⟨ESC⟩[?1h	
Character Size		
Normal*	⟨ESC⟩[?2l	⟨ESC⟩[0p
Double size	⟨ESC⟩[?2h	⟨ESC⟩[1p
Character Graphics		
Disable*	⟨ESC⟩[?3l	Similar to ⟨ESC⟩[2p and ⟨ESC⟩[3p, except that these commands do not affect the graphics plane.
Enable	⟨ESC⟩[?3h	
Keyboard		
Unlocked*	⟨ESC⟩[?4l	⟨ESC⟩[4p
Locked	⟨ESC⟩[?4h	⟨ESC⟩[5p
Opaque to Graphics*	⟨ESC⟩[?5l	When this command is received, any graphics displays that cross display character cells are hidden behind the character cell, an area 8 pixels wide and 14 high. This mode is used to make characters stand out from surrounding graphics displays. There is no equivalent capability for the 1720A Controller.
Transparent to Graphics	⟨ESC⟩[?5h	This mode causes displayed characters to be transparent to the graphics display. Any graphics displays that cross a character cell are not obstructed by the cell. Select this mode to blend characters into the graphics display.
Character Display		
Disable	⟨ESC⟩[?6l	No equivalents.
Enable*	⟨ESC⟩[?6h	
Graphics Plane		
Disable*	⟨ESC⟩[?7l	No equivalents.
Enable	⟨ESC⟩[?7h	
Cursor Display		
Disable	⟨ESC⟩[?8l	No equivalents.
Enable*	⟨ESC⟩[?8h	

*Indicates the default conditions.

Non-Destructive Display Character

Sometimes you may want to call attention to a word or phrase which is already displayed on the screen by changing its display attributes. An example of this would be a menu item which the user has selected by touching the screen. To indicate the selection, it would be desirable to highlight the characters which make up the selected menu item.

When character attributes are selected, there is a means of changing the display attributes of a character on the screen without actually re-writing the displayed character. That is the purpose of the non-destructive display character. Whenever this character sequence is sent to the display, the character at the current position of the cursor will take on the most recently specified character attributes, without changing the actual text on the display. In addition, the cursor moves to the next character position, just as it would if you had sent a normal text character to the display. In this way, any number of text characters can be assigned new display attributes.

The non-destructive character consists of the two character sequence:

⟨ESC⟩=

As an example, suppose the text string "Hello world" has been displayed in the upper left corner of the screen. We wish to assign highlight attributes to each of the characters in this message. The following fragment of a BASIC program could be used to perform this operation.

```
100 ND$ = CHR$(27%) + "="     ! define the non-destructive character
110 HM$ = CHR$(27%) + "[H"    ! define the home cursor function
120 HL$ = CHR$(27%) + "[1m"   ! define highlight attribute
130 PRINT HM$;                ! home the cursor
140 PRINT HL$;                ! select highlight attribute
150 FOR I% = 1% TO 11%        ! use a loop to "touch" each character
160     PRINT ND$;            ! in the text string
170 NEXT I%
```

In this manner, the display attributes of any text string can be modified, without re-writing the original message.

THE GRAPHICS PLANE

Information to be displayed is held in two separate portions of memory: the character plane and the graphics plane. Both of them can be turned on and off using the ANSI Standard Control Sequences just described. Additionally, the graphics plane can be turned on and off using two of the routines in the Object File named "GRAPH.OBJ".

To use the Graphics Routines described in theis section, your program must link to them. For example, using the BASIC Interpreter line

LINK "GRAPH"

early in the program will link the graphics routines and enable them to be used throughout the program.

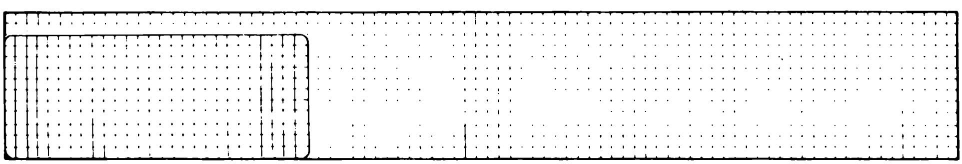
If you will be compiling the program with the BASIC Compiler, do not include the LINK "GRAPH" statement in the program. Instead, use the Linking Loader (LL) to link your program's object file with the graphics routines as shown here:

```
LL> I {program name}, B$LOAD, GRAPH
LL> O {program name}
LL> G
```

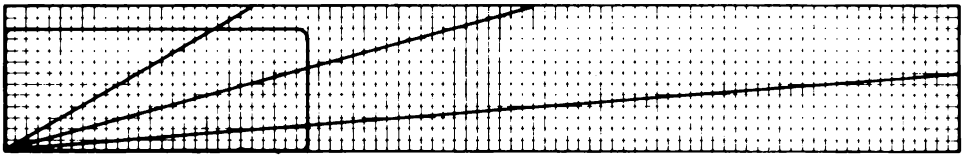
Introduction to the Graphics Routines

The display memory is divided into a character plane, and a graphics plane. Either plane can be enabled or disabled independently.

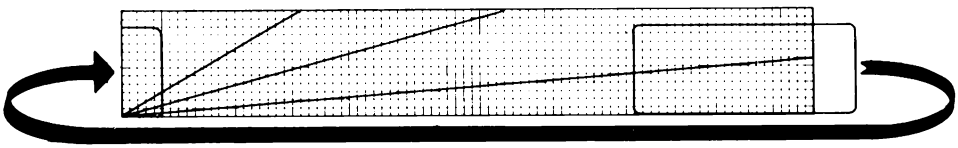
The graphics workspace is an array of dots called pixels, a contraction of the words picture elements. In the horizontal direction, there are 2048 pixels, and in the vertical direction, there are 256. The display screen provides a window into the graphics workspace that is 640 dots wide and 224 dots high. The window can be positioned anywhere over the graphics plane. Here is where it starts out:



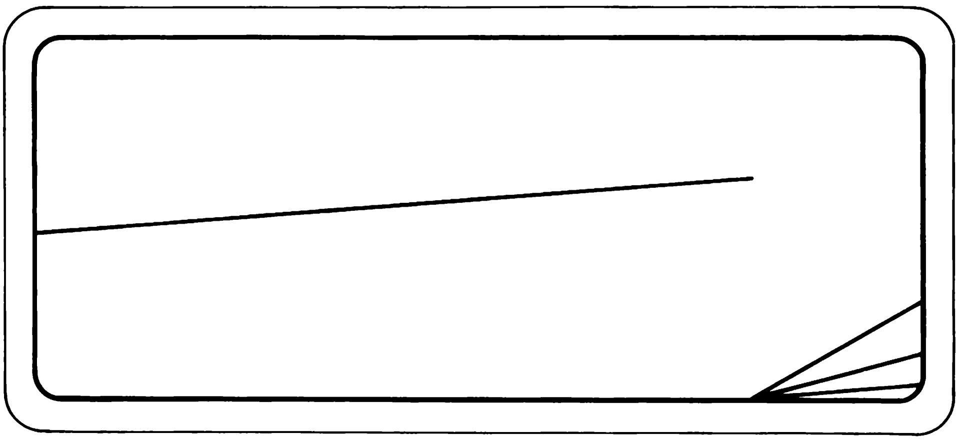
If the window position is moved beyond the edges of the graphics plane, the display wraps around to the opposite edge. For example, assume that these lines have been drawn in the graphics area:



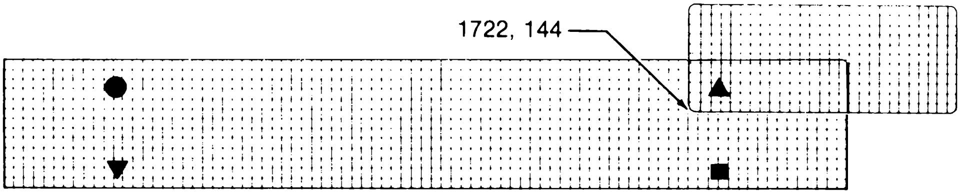
This is what happens when you move the window to overlap the right hand boundary of the workspace:



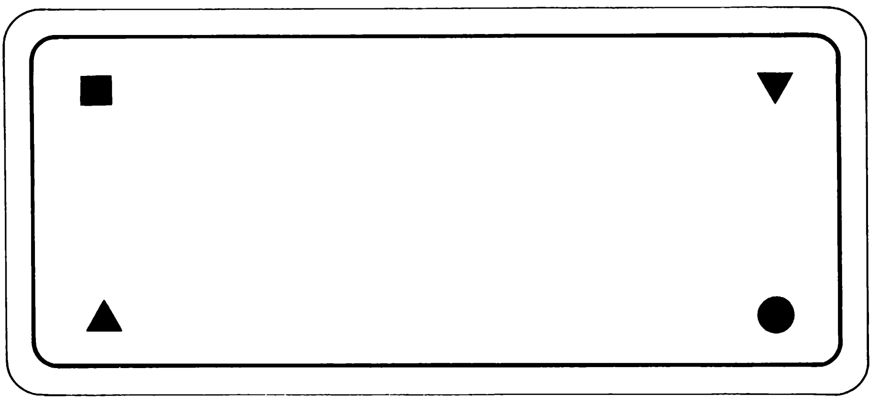
Because of the wraparound effect, the resulting display will be:



The next drawing illustrates what would happen if the window is positioned so that it overlaps both a vertical and horizontal edge of the graphics workspace. First, assume that these symbols have been place in the four corners of the display, and the window positioned as shown:



The resulting display would be:



This result may be either useful or surprising, depending on what you had in mind as you designed the display. Note that the effect does not hold true for the lower left reference corner of the window, which cannot be moved outside the graphics workspace.

Addressing the Pixel Positions

Dot positions are addressed by their X,Y coordinates. In the first drawing, the display window was positioned so that its lower left corner was at position 0,0 (the default starting position); the lower right is at 639,0; upper left is at 0, 223, and the upper right corner is at 639,223. In the last drawing, the current position has been moved so the lower left corner of the display is at 1722, 144.

Pixels can be turned on or off anywhere within the window. The rest of this section describes the set of routines that control turning the pixels on and off, and moving the current position.

Graphics Routines

The graphics routines described in this section are available to BASIC programs. They are recorded on the System disk in a file named GRAPH.OBJ, and in a library file named GRAPH.LIB. FORTRAN programs access these routines by using FLUK22.LIB on the FORTRAN disk. There are additional graphics routines described in the subsequent section, Printing the Graphics Plane.

Some of the routines turn the display on or off, and others move the "current position". Though the current pixel position is not displayed (as a cursor shows the current position in the character plane), the initial current position is always 0,0. As you can see by the drawings, position 0,0 is the lower left corner of the display. The upper right corner of the display window is position 639, 223. The upper right corner of the entire graphics plane is at location 2047, 255.

As a program moves the pixel position, or pans the window around the workspace, the routines keep track of the changing current position.

All the routines can be linked to programs generated by the BASIC Interpreter and the BASIC and FORTRAN Compiler programs.

Summary of Commmands

The table that follows describes each of the graphics routines. Note that all the arguments must be integers except for the first arguments to LABEL and LABELF. This means that in a BASIC language program, for example, the arguments must be followed by the % symbol. In FORTRAN, variable types are determined by the first character in the variable name (Integers I though N), or by using the TYPE statement. See the particular progamming language manual for full details.

All of the following routines can be called by a BASIC or FORTRAN program except for LABEL and LABELF. LABEL is called by a BASIC program and LABELF is called by a FORTRAN program.

SUMMARY OF GRAPHICS ROUTINES

COMMAND	PURPOSE
DOT (X, Y, {type})	Draws a single dot at X, Y, returns to the current position.
DRAW (X1, Y1, X2, Y2, {type})	Draws a line from X1, Y1 to X2, Y2.
ERAGRP ({type})	Erases the entire graphics plane.
GRPOFF	Disables the graphics plane.
GRPON	Enables the graphics plane.
MOVE (X, Y)	Moves to absolute position X, Y.
MOVER (Xoffset, Yoffset)	Moves relative to amount specified by the offset.
PAN (X, Y)	Sets display window position to X, Y.
PLOT (X, Y,{type})	Plots from current position to X, Y.
PLOTR (Xoffset, Yoffset, {type})	Plots relative to amount specified by the offset.
LABEL(S$, D%, T%) LABELF(STR,LENSTR,DIR,TYPE, ERROR)	Place string in graphics memory either horizontal or vertical

*Type: -1 = INVERSE
0 = BLACK
1 = WHITE

- The values of the X and Y arguments may not exceed 2047, and may only be negative in the relative commands MOVER and PLOTR.
- The X argument is the number of pixels in the horizontal direction. Since the screen is 640 pixels wide, the center is 320 pixels from the left edge.
- The Y argument is the number of pixels in the vertical direction. The screen is 224 pixels high, so the center is 112 pixels from the bottom edge.
- The 'Type' argument determines whether the routine paints white on black (1), black onto white (0), or the inverse of the color already at that position (-1).
- The S$ argument is a string variable or string constant that is placed in the graphics plane.
- The STR argument is an array with one character in each byte that is placed in the graphics plane.
- The LENSTR argument specifies how many characters are in the STR array.
- The D% and DIR arguments specify whether the string should be placed in the graphics plane horizontally (0) or vertically (90).

In the pages that follow, each of the routines except LABELF is described in detail, and some suggestions are given about the kinds of things that each of them can be used for. Some of the descriptions include program examples. The example listings are all shown as they would appear in programs written for the BASIC Interpreter.

DOT
GRAPH.OBJ

Usage

DOT(X%, Y%, {type%})

Description

This routine places a single dot at the specified coordinates, then returns to the current pixel position. Because this routine returns to the former pixel position it is useful in the construction of detailed charts or graphs that require pixel resolution and are generated by a mathematical formula that calculates each point from the same position.

Parameters

X An integer 0 to 2047 that specifies the absolute horizontal pixel location for the dot.

Y An integer 0 to 255 that specifies the absolute vertical pixel location for the dot.

type An integer value that specifies whether the dot is painted white on black (1), black on white (0), or the inverse of the color already at that position (-1).

Example

This BASIC program uses DOT to draw a sine wave. It first asks for "Amplitude", and then for "Period". The amplitude is the peak-to-peak pixel amplitude of the sine wave that will be drawn. The period is used for frequency and sampling rate. Notice that the program does not allow a period of less than 1. Selecting periods less than five results in waveforms whose resolution is too coarse for the wave to be observable.

```
10  LINK "GRAPH"                      ! link to graphics
20  ERAGRP (0%) \ GRPON               ! erase, turn on graphics
30  PRINT CHR$(27) + "[2J"            ! clear character plane
40  PRINT CPOS (14,0); "Amplitude";   ! input amplitude
50  INPUT A                           ! amplitude: dots peak-to-peak
60  PRINT CPOS (15,0); "Period";      ! input period (sampling rate)
70  INPUT P                           ! period: no. of dots/cycle
80  IF P = 0 THEN 60                  ! P = 0 is illegal
90  PRINT CHR$(27) + "[2J"            ! clear character plane
100  FOR X% = 1 to 640                ! x across display
110    Y  = A * SIN (X% / P)          ! calculate y
120    Y% = 0.5 * (Y) + 112           ! integerize and offset y
130    DOT (X%, Y%, 1%)               ! dot at calc'ed position
140  NEXT X%                          ! continue calculation
150  GOTO 40                          ! next value
```

Usage

DRAW(X1%, Y1%, X2%, Y2%, {type%})

Description

This routine draws a line from absolute position X1, Y1 to another absolute location, X2, Y2. If the final position is beyond the edge of the graphics plane, the line will end at the edge. The current pixel position is not changed by DRAW.

Parameters

X1	An integer 0 to 2047 that specifies the absolute horizontal pixel location for the start of the line.
Y1	An integer 0 to 255 that specifies the absolute vertical pixel location for the start of the line.
X2	An integer 0 to 2047 that specifies the absolute horizontal pixel location for the end of the line.
Y2	An integer 0 to 255 that specifies the absolute vertical pixel location for the end of the line.
type	An integer value that specifies whether the line is painted white on black (1), black on white (0), or the inverse of the color already at that position (-1).

Example

Current position is 0,0. To draw a white diagonal line across the display, use:

```
DRAW(0%, 0%, 639%, 223%, 1%)
```

ERAGRP
GRAPH.OBJ

Usage

ERAGRP {type%}

Description

This routine erases the entire graphics plane to the color indicated by {type%}, either green, black, or the reverse of the color before erasing. Any data within the plane will be deleted. The character plane is unaffected.

Parameters

type — An integer value that specifies whether the screen is erased with white (1), black (0), or the inverse of the color already at every position in the graphics plane (-1).

Example

At the beginning of a program, use ERAGRP to prepare the Graphics plane for the display.

```
ERAGRP (0%)          ! Erase to black
ERAGRP (1%)          ! Erase to green
ERAGRP (-1%)         ! Create inverse image
```

GRPOFF, GRPON
GRAPH.OBJ

Usage

GRPOFF,GRPON

Description

These routines turn the graphics portion of a display off and on. The memory is left intact; the routines only determine if the graphics plane is displayed or not. The character plane is unaffected.

Example

A selection display has just been presented to the operator. When the selection has been made, a new display is presented that contains new graphics. Rather than using ERAGRP to erase the graphics plane, however, it is desired to leave the contents alone because the test results update the display for the next selection. In this case, use GRPOFF to turn off the graphics display. When the display is updated, the program uses GRPON to display the change.

MOVE
GRAPH.OBJ

Usage

MOVE(X%, Y%)

Description

This routine moves the current pixel location without drawing. If either X% or Y% are outside of the graphics plane, the move stops at the corresponding edge.

Parameters

X An integer 0 to 2047 that specifies the absolute horizontal pixel location to move to.

Y An integer 0 to 255 that specifies the absolute vertical pixel location to move to.

Example

A program has just drawn a diagonal line from the bottom left to the upper right corner of the screen. Now, to "lift the pencil" to get back to 0,0, use the MOVE routine:

```
MOVE(0%, 0%)
```

MOVER
GRAPH.OBJ

Usage

MOVER(Xoffset% Yoffset%)

Description

This is the relative move routine. It moves the current pixel position to a relative position within the graphics plane. The move is done without drawing; if the new position is outside the graphics plane, the move stops at the corresponding edge.

Parameters

Xoffset	An integer -2047 to 2047 that specifies the relative number of horizontal pixel locations to move.
Yoffset	An integer -255 to 255 that specifies the relative number of vertical pixel locations to move.

Example

A program is being designed that draws two figures that may appear any place on the display. The second figure must appear immediately to the right of the first. After the first figure is drawn, use the relative move routine to move the current position relative to the ending location of the first figure.

PAN
GRAPH.OBJ

Usage

PAN(X%, Y%)

Description

The PAN routine moves the window around the graphics workspace. The reference is the lower left corner of the display window. Panning to location (i,j) means that pixel (i,j) of the graphics memory will appear at the lower left corner of the display window. PAN does not affect the current pixel position.

Parameters

X — An integer 0 to 2047 that specifies the horizontal pixel location of the reference corner.

Y — An integer 0 to 255 that specifies the vertical pixel location of the reference corner.

Example

During a measurement session, data has been collected by a program, and has become part of a data file. The operator then elects to view the results of the day. The program inserts the raw data into a subroutine that creates and draws a chart that cannot fit in one window. Use the PAN routine to permit viewing the entire chart. Left and right arrow keys can be made part of the display, to allow positioning the window at any area of interest.

Usage

PLOT(X%, Y%, {type%})

Description

This routine draws a line from the current position to the location indicated by the X and Y arguments. (Also see DRAW.) PLOT uses the current position as the starting place to begin drawing, rather than defining the starting position, as DRAW does. The current pixel position is updated to X,Y.

Parameters

X An integer 0 to 2047 that specifies the absolute horizontal pixel location for the end of the line.

Y An integer 0 to 255 that specifies the absolute vertical pixel location for the end of the line.

type An integer value that specifies whether the line is painted white on black (1), black on white (0), or the inverse of the color already at that position (-1).

Example

Use PLOT rather than DRAW in those instances where the starting position will be unknown, but a line is desired from one place to some other position. This routine can be used in constructing some types of graphs, like pie-charts. As the program collects data, the value of the data would be inserted into a Relative Move statement, and the PLOT statement would draw the line from the starting point to the calculated position (which then becomes the new current position).

PLOTR
GRAPH.OBJ

Usage

PLOTR(Xoffset%, Yoffset%, {type%})

Description

The relative plot routine draws a line from the current position to the location indicated by the Xoffset and Yoffset arguments; it is similar to DRAW, except that as it returns to the starting position, continues drawing; it doesn't "lift the pencil".

Parameters

Xoffset — An integer -2047 to 2047 that specifies the relative number of horizontal pixel locations for the end of the line.

Yoffset — An integer -255 to 255 that specifies the relative number of vertical pixel locations for the end of the line.

type — An integer value that specifies whether the line is painted white on black (1), black on white (0), or the inverse of the color already at that position (-1).

Example

A triangular figure is to be drawn, and it may appear anywhere within the graphics plane. Use the Plot Relative routine to draw the figure relative to any starting position. This example draws a triangle that will be black if the field is green, and green if the surroundings are black:

```
MOVE (0%, 0%)
PLOTR(60%, 60%, -1%)
PLOTR(60%, -60%, -1%)
PLOTR(-120%, 0%, -1%)
```

LABEL
GRAPH.OBJ

Usage

LABEL({string}, {direction}, {type})

Description

This routine places a string of characters in the graphics plane. These characters appear exactly as they would if displayed normal size in the character plane. The character string can be positioned horizontally or vertically. The current pixel position determines the starting position of the string. The current pixel position is updated so that a subsequent LABEL routine call will cause string concatenation. See subsequent examples on string positioning. If the final position of the string is beyond the edge of graphics memory, the string will wrap around and continue at the opposite edge.

The LABEL routine puts a string into the graphics plane at the rate of about 1/60 of a second per character which means that it takes about 1.3 seconds to draw an 80-character string.

If keyboard input occurs during a long series of drawing operations, there could be up to a 6.5 second delay in responding to the input. This delay time also occurs with ⟨CTRL⟩/C, ⟨CTRL⟩/P, and the ABORT.

Parameters

string — This is a string constant or a string variable. Each character of the string is taken from the character EPROM. If access to the alternate character set is desired, the string should contain the ASCII control character SO (shift out, decimal 14). All characters after SO are taken from the alternate character set until the ASCII control character SI (shift in, decimal 15) is encountered. The SI causes selection to revert to the primary character set. The default is the primary set, so all selection is from the primary set unless SO is encountered.

The maximum length of a string that can be placed in the graphics plane with a single LABEL call is 80 characters. Since the control characters SO and SI are not displayed, they are not part of the 80 character string length. If the 80 character string length is exceeded the string will be truncated.

direction	This is an integer 0 or 90. If the direction is 0, then the string will be horizontally oriented in the graphics plane. If a direction of 90 is given, the string will be positioned vertically. See subsequent examples.
type	This is an integer 0, 1, or −1. It determines whether the routine paints a white label on a black screen (1), black label on a white background (0), or the inverse of the color already at that position (−1).

Errors

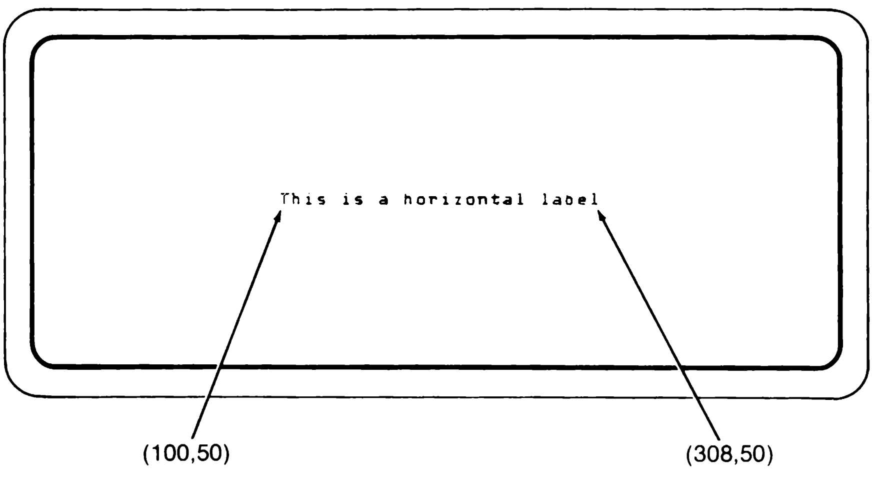
2006 - direction argument not 0 or 90

Example 1

This BASIC program places a 26 character horizontal string in the graphics plane. An initial MOVE changes the current position to 100,150, the place where the bottom left corner dot of the first character cell of the string will be. LABEL updates the current position to the dot immediately to the right of the bottom right corner of the last character cell of the string (308,50). A horizontal character cell is 8 dots wide and 14 dots high, so in this example the current position has been updated in the horizontal direction by 208 dots (26 characters X 8 dots).

```
10 LINK 'graph' \ LINK 'xgraph'              ! link to graphic routines
20 PRINT CHR$(27);"[2J"                       ! clear character plane
30 ERAGRP(0%) \ GRPON \ PAN(0%,0%)            ! erase & turn on graphics plane
40 MOVE(100%,50%)
50 LABEL('This is a horizontal label',0%,1%)
```

The following is what appears on the display screen after the above program is run.



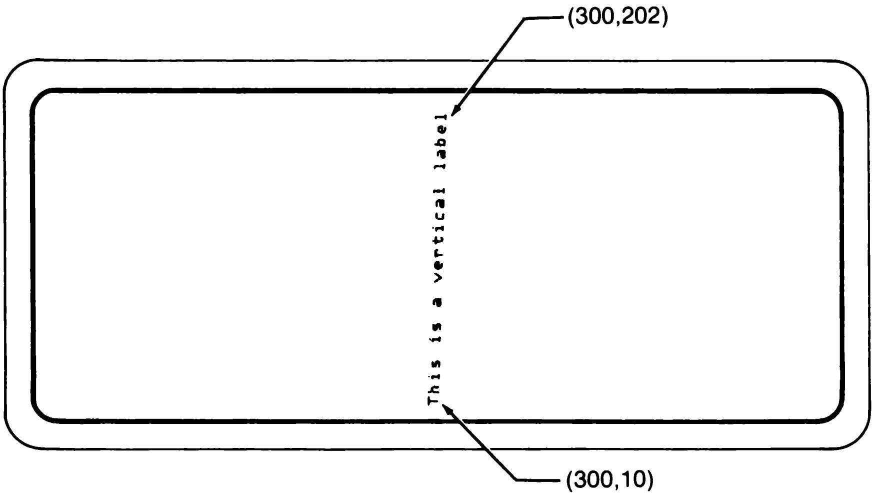
LABEL
GRAPH.OBJ

Example 2

This BASIC program places a 24 character vertical string in the graphics plane. An initial MOVE changes the current position to 300,10, the place where the bottom right corner dot of the first character cell of the string will be. LABEL updates the current position to the dot immediately above the top right corner of the last character cell of the string (300,202). A vertical character cell is 14 dots wide and 8 dots high, so in this example the current position has been updated in the vertical direction by 192 dots (24 characters X 8 dots).

```
10 LINK 'graph'                          ! link to graphic routines
20 PRINT CHR$(27);"[2J"                  ! clear character plane
30 ERAGRP(0%) \ GRPON \ PAN(0%,0%)       ! erase & turn on graphics plane
40 MOVE(300%,10%)
50 LABEL('This is a vertical label',90%,1%)
```

The following is what appears on the display screen after the above program is run.

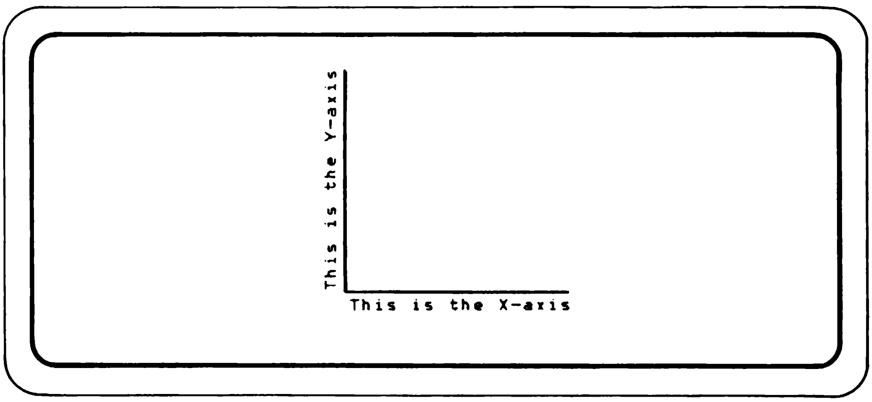


Example 3

This BASIC program draws an X and Y axis with horizontal and vertical labels on the axis. The MOVER and PLOTR commands are used to show how the LABEL routine updates the current pixel position.

```
10 LINK 'graph'                              ! link to graphic routines
20 PRINT CHR$(27);"[2J"                      ! clear character plane
30 ERAGRP(0%) \ GRPON \ PAN(0%,0%)           ! erase & turn on graphics plane
40 MOVE(250%,30%)
50 LABEL('This is the X-axis',0%,1%)
60 MOVER(0%,19%) \ PLOTR(-144%,0%,1%)
70 MOVER(-5%,0%)
80 LABEL('This is the Y-axis',90%,1%)
90 MOVER(5%,0%) \ PLOTR(0%,-144%,1%)
```

The following is what appears on the display screen after the above program is run.



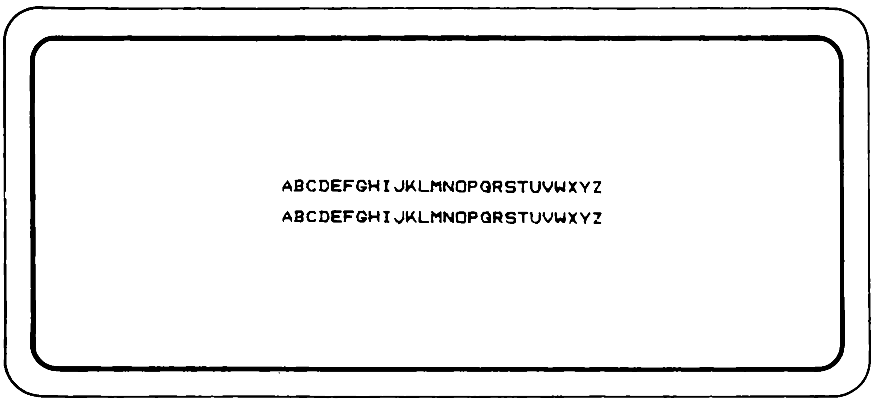
LABEL
GRAPH.OBJ

Example 4

This BASIC program shows how the label command causes the current pixel position to be updated so that strings can easily be concatenated.

```
10 LINK 'graph'                           ! link to graphic routines
20 PRINT CHR$(27);"[2J"                   ! clear character plane
30 ERAGRP(0%) \ GRPON \ PAN(0%,0%)        ! erase & turn on graphics plane
40 MOVE(200%,120%)
50 LABEL('ABCDEFGHIJKLM',0%,1%)           ! some labeling to show concatenatior
60 LABEL('NOPQRSTUVWXYZ',0%,1%)
70 MOVE(200%,100%)
80 LABEL('ABCDEFGHIJKLMNOPQRSTUVWXYZ',0%,1%)
```

The following is what appears on the display screen after the above program is run.

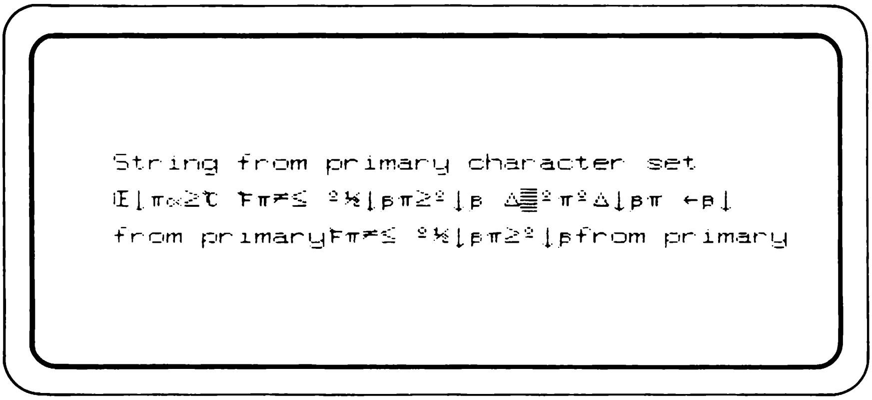


Example 5

This BASIC program shows how the LABEL routine can access the alternate character set.

```
10 LINK 'graph'
20 PRINT CHR$(27);"[2J"                    ! clear character plane
30 ERAGRP(0%) \ GRPON \ PAN(0%,0%)         ! erase & turn on graphics plane
40 SO$=CHR$(14%)
50 SI$=CHR$(15%)
60 MOVE(150%,130%)
70 LABEL("String from primary character set",0%,1%)
80 MOVE(150%,110%)
90 LABEL(SO$+"String from alternate character set",0%,1%)
100 MOVE(150%,90%)
110 LABEL("from primary"+SO$+"from alternate"+SI$+"from primary",0%,1%)
```

The following is what appears on the display screen after the above program is run. This output would be different if a customized (non-standard) character EPROM was being used.



PRINTING THE GRAPHICS PLANE

A printed image of the display screen window into the graphics plane can be obtained by calling special routines from a BASIC or FORTRAN program.

A printer must be attached to a 1752A RS-232-C port, KB1: or KB2:. The Epson FX® and RX® series, the Epson MX-100® with GRAFTRAX®, and the Tally® Model MT1605E are the supported printers. A routine is provided that allows flexibility in printing a screen on other printers or a special assembly language module can be written to support other printers. See the section "Printing the Graphics Plan on Unsupported Printers".

The Epson printers need to have the Epson Intelligent Serial Interface® card (Epson Cat. No. 8148) installed. The Epson is connected to the KB1: or KB2: port with Fluke printer cable Y1709. The Epson printer and Epson serial interface DIP switches should all be left in their factory default settings. Use the Set Utility program to establish the following parameters:

```
SET
KB1: br 19200 db 8 pb n si e so e to 10
exit
```

The Tally® MT1605E printer should be connected to the KB1: or KB2: port with Fluke printer cable Y1709. All DIP switches should be in the OFF position except switch 4 in the I/O panel, which should be ON. Use the set utility program to establish the following parameters:

```
SET
KB1: br 9600 db 8 pb n si e so e to 10
exit
```

It is important to note that only the display screen window into the graphics plane is printed. The character plane is not printed. A convenient way to place characters in the graphics plane is provided by the graphics routine, LABEL.

® Epson FX, RX, Epson MX-100, and GRAFTRAX are Registered Trademarks of Epson America, Inc.
® Tally is a Registered Trademark of Mannesmann Tally

Graphics Print Routines

The routines available to BASIC programs are recorded on the System disk in a file named GPRINT.OBJ, and in a library file named BASIC.LIB. FORTRAN programs access the graphics print routines by using the library file named FLUK22.LIB on the FORTRAN disk.

Summary of Graphics Print Commands

The table that follows describes the graphics print routines. There are two differently named routines for each graphics print capability, one routine is called by a BASIC program and the other is called from FORTRAN. All FORTRAN routine names end in an F; BASIC routine names do not end in an F. All arguments are integers except that the first argument to GRBYTE and GRBYTF is an integer array name.

Summary of Graphics Print Routines

COMMAND	PURPOSE
GPOUT({chan #},{printer type}) GPOUTF({unit #},{printer type},{error})	designate serial port to which to send screen and specify printer type
GPRINT GPRNTF({error})	perform screen dump to printer
GRBYTE({array},{slice},{bit order}) GRBYTF({array},{slice},{order},{error})	read screen slice and store in array

In the pages that follow, each of the BASIC routines is described in detail. Program examples are provided as they would appear in programs written for the BASIC Interpreter. Error numbers given are those that BASIC reports. The corresponding FORTRAN routines are described in the FORTRAN manual.

GPOUT
GPRINT.OBJ

Usage

GPOUT({channel number}, {printer type})

Description

This routine designates a channel for the graphics screen data and the type of printer that is used. This routine must be called before calling the GPRINT routine. The GPRINT routine does the screen dumping and GPOUT is called to prepare for printing.

Parameters

channel number — An integer between 1 and 16 that corresponds to a channel number specified in a previous OPEN statement. The OPEN statement must have assigned a channel to KB1: or KB2:.

printer type — An integer between 0 and n that specifies what type of printer is to be used for the graphics dump.

Printer Type	Printer
0%	Epson FX series
1%	Epson RX series
2%	Epson MX-100 with Graftrax
3%	Tally MT1605E
4% : : n%	for user-defined printer types (See Printing the Graphics Plane on Unsupported Printers section)

Errors

302 - Illegal channel number
308 - Channel is not open
2001 - Undefined printer type

Example

This BASIC program excerpt specifies that the screen to be printed on is an Epson FX printer that is connected to the KB1: port.

```
40 OPEN "KB1:" AS NEW FILE 5        ! assign channel 5 to KB1:
50 GPOUT(5%,0%)                     ! data to channel 5 & Epson FX printer
```

GPRINT
GPRINT.OBJ

Usage

GPRINT

Description

This routine causes the display screen window into the graphics plane to be dumped to a printer. Any images in the character plane are ignored. The BASIC statement after the call to GPRINT is executed when the printing has finished or when an error has been detected.

Almost all input from the keyboard or touch sensitive display is ignored until printing is done or an error is detected. Inputs that are not ignored during the operation of GPRINT are 〈CTRL〉/ P, 〈CTRL〉/ C, and the ABORT button. Any of these cause GPRINT.to stop, but they have different effects than in a normal BASIC program. These effects are:

- If GPRINT is called from the BASIC Interpreter, 〈CTRL〉/ C, or the ABORT button will cause processing to resume at the next BASIC statement. If 〈CTRL〉/ P is done while in character mode (SET NOECHO), control passes to the statement after the GPRINT call. In line mode, 〈CTRL〉/ P causes an exit to FDOS.

- If GPRINT is called from compiled BASIC while in character mode or line mode, 〈CTRL〉/ P, 〈CTRL〉/ C, and the ABORT button will cause an exit to FDOS.

- The BASIC 'ON CTRL/C GOTO ...' statement has no effect during GPRINT execution. This means that while a GPRINT is in operation 〈CTRL〉/ C, the ABORT button, and possibly 〈CTRL〉/ P will not be trapped as they normally would be.

Errors

311 - Non-recoverable device read/write error (timeout)
2002 - GPOUT not called yet
2003 - Printer timeout
2004 - KB0: is not the console device (a switch has been performed)

GPRINT
GPRINT.OBJ

Notes

1. Pressing the ABORT button during GPRINT can cause BASIC error 311 (Non-recoverable device read/write error). A ⟨CTRL⟩/S done before or during GPRINT will always cause a 311 error.

2. If ⟨CTRL⟩/P is done during a GPRINT, it may be necessary to power the printer off and on. This is to avoid extraneous data to be printed at the start of a subsequent screen print.

Example

This BASIC program draws a picture and prints it on an Epson MX-100 printer.

```
10 LINK 'gprint'                      ! link to graphic print routines
20 PRINT CHR$(27);"[2J"               ! clear character plane
30 ERAGRP(0%) \ GRPON                 ! erase & turn on graphics plane
40 MOVE(250%,80%)
50 PLOTR(60%, 60%, -1%)               ! draw a picture
60 PLOTR(60%, -60%, -1%)
70 PLOTR(-120%, 0%, -1%)
80'
90 OPEN "KB1:" AS NEW FILE 4          ! assign channel 4 to KB1:
100 GPOUT(4%,2%)                      ! data to channel 4 & an Epson MX-100
110 GPRINT                            ! print graphics plane
120 CLOSE 4
```

Usage

GRBYTE({array address}, {slice number}, {slice row order})

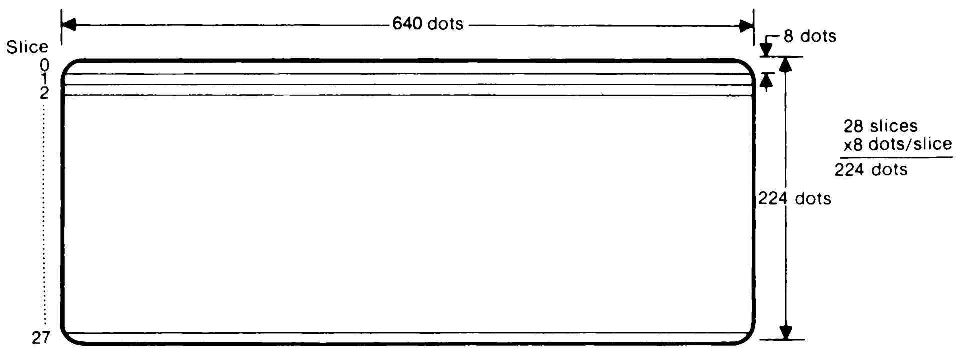
Description

This routine fills an array with a "slice" of the display screen. Any images residing in the character plane are ignored. This routine can be used for printers not supported by the GPRINT routine, or for saving a screen image file to print later.

Parameters

array address — This is the starting address of a 640 element integer array. This array must be a main memory array; it can not be a virtual array. The first address of the array is specified by the array name followed by left and right parentheses, for example A%(). The program can fail if the array does not contain at least 640 elements.

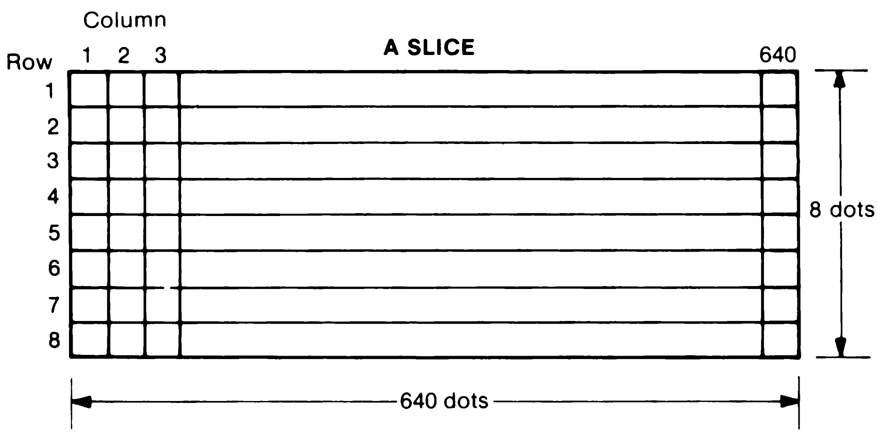
slice number — An integer 0 through 27 which corresponds to an eight dot high horizontal section of the display screen. The following figure shows what is meant by a "slice" of the screen and how the slices are numbered.



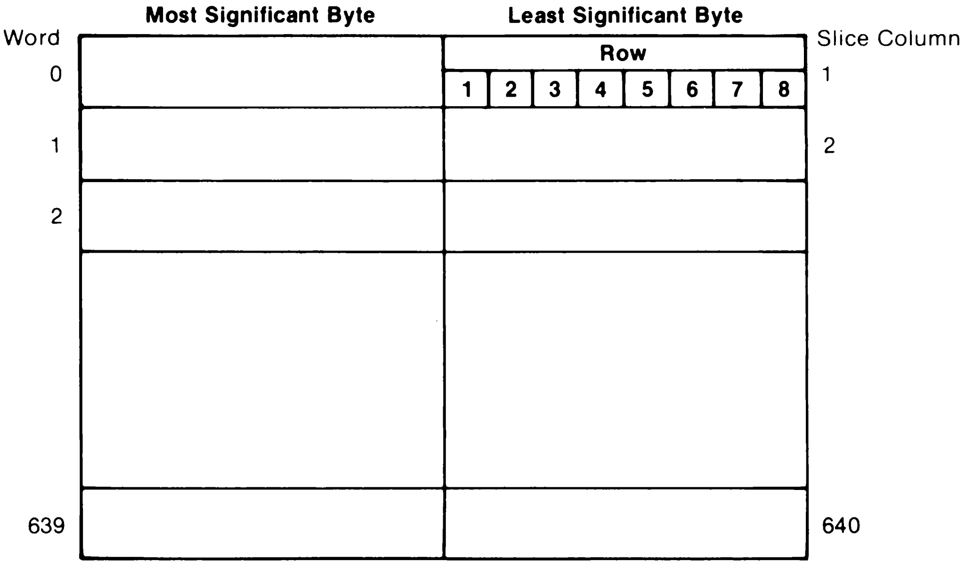
slice row order	An integer that specifies the order in which the bits in each column of a slice are placed into the array. If this argument is 0, the bottom row of a slice is placed in the least significant bit of an array element. If the argument is any other value, the top row of a slice is placed in the least significant bit of an array element. This argument is provided because of different printer model conventions in addressing the print head pins.

The GRBYTE routine takes a slice of the display screen and places the information in the least significant bytes of the user's array. If a dot location is on (illuminated), then the bit in the array corresponding to that dot location is a 1. The following figures show how a slice of the screen is placed into the user's array when the 'slice row order' argument is 0.

SLICE TO ARRAY CORRESPONDENCE

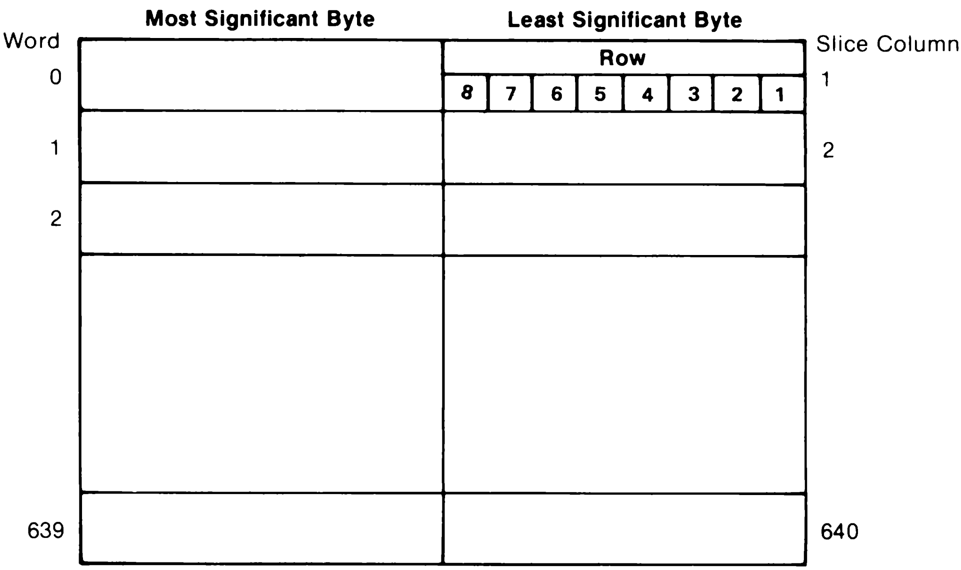


USER ARRAY WHEN SLICE ROW ORDER ARGUMENT IS 0



The following figure shows how a slice of the screen is entered into the user's array when the "slice row order" argument is not 0.

USER ARRAY WHEN SLICE ROW ORDER ARGUMENT IS NOT 0



GRBYTE
GPRINT.OBJ

Errors

311 - Non-recoverable device read/write error (timeout)
2004 - KB0: is not the console device (a switch has been performed)
2005 - Slice number not in 0 to 27 range

Note

Pressing the ABORT button during GRBYTE can sometimes cause BASIC error 311 (Non-recoverable device read/write error). A ⟨CTRL⟩/S done before or during GRBYTE will always cause a 311 error.

Example

This BASIC program draws a picture and prints it on an Epson FX printer using an IEEE port. The program assumes that the Epson's IEEE interface card has been set to device address 6.

```
10 LINK 'grprint' \ LINK 'graph'    ! link to graphic print routines
20 ERAGRP(0%) \ GRPON \ PAN(0%,0%)
30 MOVE(0%,0%)
40 PLOTR(0%,223%,1%)                ! draw a picture
50 PLOTR(639%,0%,1%)
60 PLOTR(0%,-223%,1%)
70 PLOTR(-639%,0%,1%)
80 DIM A%(640%)                     ! array for a screen 'slice'
90 PRINT @6,CHR$(27);'A';CHR$(8);   ! send code for printer line spacing
100 FOR S% = 0% TO 27%
110    GRBYTE(A%(), S%, 0%)         ! a slice of screen to array A
120    PRINT @6,CHR$(27);'*';CHR$(6);! send required code to put Epson FX
130    PRINT @6,CHR$(128);CHR$(2);  !        in graphics mode
140    FOR I% = 0% TO 639%
150       PRINT @6,CHR$(A%(I%));    ! send 8 dot column of slice to printer
160    NEXT I%
170   PRINT @6                      ! carriage return
180 NEXT S%
```

PRINTING THE GRAPHICS PLANE ON UNSUPPORTED PRINTERS

The printers supported by the GPOUT and GPRINT routines are the Epson FX and RX, the Epson MX-100 with GRAFTRAX, and the Tally MT1605E. There are two different ways to print the graphics screen data on other printers.

1. The easiest method of printing the screen on various printers is to use the GRBYTE routine. Flexibility to drive different printers is achieved by making multiple calls to the GRBYTE routine from a BASIC program and using the BASIC PRINT statement. The GRBYTE routine retrieves the screen data and the PRINT statement sends the screen data to a printer. See the previous section that describes the GRBYTE routine.

The following is an example of a BASIC program that calls GRBYTE and uses PRINT statements to print a screen image on a Prowriter Model M8510B® printer.

```
10 LINK 'gprint' \ LINK 'graph'     ! link to graphic print routines
20 ERAGRP(0%) \ GRPON \ PAN(0%,0%)
30 MOVE(0%,0%)
40 PLOTR(0%,223%,1%)                ! draw a picture
50 PLOTR(639%,0%,1%)
60 PLOTR(0%,-223%,1%)
70 PLOTR(-639%,0%,1%)
80 DIM A%(640%)
90 OPEN "KB1:" AS NEW FILE 2        ! assign channel 2 to KB1:
100 PRINT #2,CHR$(27);'T14';        ! send code for printer line spacing
110 FOR S% = 0% TO 27%
120    GRBYTE(A%(), S%, 1%)         ! a slice of screen to array A
130    PRINT #2,CHR$(27);'S00640';  ! put Prowriter M8510B in graphics mode
140    FOR I% = 0% TO 639%
150       PRINT #2,CHR$(A%(I%));    ! send 8 dot column of slice to printer
160    NEXT I%
170  PRINT #2                       ! carriage return
180 NEXT S%
190 CLOSE 1
```

® *Prowriter Model M8510B is a Registered Trademark of C. Itoh Electronics, Inc.*

2. Another method of printing on an unsupported printer is to create an Assembly Language routine and tables modeled after the assembly routines and tables for the supported printers. These printer driver routines and tables are recorded on the Assembly disk (Option 17XXA-201) in a file named PRDRIV.PRE.

   A new printer type value to be specified in the GPOUT routine is defined by supplying a new entry in the file named PTYPE.PRE. The Assembly disk has an image of this Assembly Language source file.

   By writing new routines and tables in PRDRIV.PRE and making the appropriate entry in PTYPE.PRE, a BASIC program only needs to call GPOUT and GPRINT to print the display screen.

   The steps to be taken and excerpts from the above mentioned Assembly Language source are as follows:

   a. Add a new printer type value and entry point name in PTYPE.PRE. The following is the entire source of PTYPE.PRE. Entries would be made before the 'data 0,0' line and after the 'tally' line.

```
        idt     'ptype'
*
        def     ptype           ;table of printer types & module names

        ref     epsonf          ;Epson FX & Epson RX driver
        ref     epsonm          ;Epson MX driver
        ref     tally           ;Tally MT1605E
*
*            type module          printer
*            ---- ------          -------
ptype   data    0,epsonf        ;Epson FX
        data    1,epsonf        ;Epson RX
        data    2,epsonm        ;Epson MX
        data    3,tally         ;Tally MT1605E
        data    0,0             ;null to mark end of table
        end
```

b. Add a def statement for the new printer driver entry point at the start of PRDRIV.PRE

c. Add a new module in PRDRIV.PRE that references a new printer dependency table. The following is the module for the Epson FX printer.

```
*
* module:       epsonf
*
* function:     To send graphics screen data to the Epson FX or RX printer
*
* called via:   bl from GPRINT
*               (all registers available except r11)
*
epsonf  equ     $                   ;entry point
        clr     r1                  ;
*       if      @slicen equ r1      ;if at 1st screen slice
        li      r1,epfdat           ;   get Epson FX & RX printer dependencies table
        mov     r1,@prdptb          ;
*       endif                       ;endif
        b       @driver             ;go to common printer driver
```

d. Create a printer dependency table in PRDRIV.PRE. Most entries in this table are pointers to other tables that contain the character sequences required by a particular printer. The third entry in this table is the time in 10 millisecond ticks that it takes for the printer to print its data buffer. This entry is required for handling CTRL /C and the ABORT button correctly. The order of entries within this printer dependency table is very important. Use the following table for the Epson FX printer as a guide.

```
*
*          Epson FX & RX dependency table
*
epfdat          data    epstch  ;row start characters table address
                data    epstln  ;row start characters table length
                data    200     ;delay in 10ms ticks for ^C during printing
                data    epterm  ;termination characters table address
                data    epterl  ;termination characters table length
                data    0       ;null to mark end of this table
```

e. Create the tables containing the character sequences that are required to put the printer in dot graphics mode. The following are tables used by the Epson FX printer driver.

```
epstch          byte    27      ;Epson FX & RX row start characters
                byte    'A'     ;
                byte    8       ;'ESC A 8' sets line feed spacing to 8/72"
                byte    27      ;
                byte    '*'     ;
                byte    6       ;
                byte    128     ;
                byte    2       ;'ESC * 6 128 2' graphics (90dpi) 640 dot row
epstln          equ     $-epstch
                even
*
epterm          byte    27      ;Epson FX,RX,MX-100 screen dump term characters
                byte    '@'     ;'ESC @' master reset for Epson (power up state)
epterl          equ     $-epterm
                even
```

f. Assemble the modified PTYPE.PRE and PRDRIV.PRE. Create a new GPRINT.OBJ file by issuing the following Linkage Editor commands. The files PRINT.OBJ and F$RGMY.OBJ that are referenced in these LE commands are also provided on the Assembly disk.

```
le gprint.obj
task gprint
form ascii
part
incl print,prdriv,ptype,f$rgmy
end
```

g. Call the GPOUT routine in a BASIC program with the new printer type value and then call the GPRINT routine.

CONCLUSION

This section has described the many features of the 1752A display. The display is specifically designed for ease of use by both the programmer and the operator. While maintaining compatibility with the display capabilities of its predecessor, the Fluke 1720A Instrument Controller, the 1752A incorporates a greatly expanded set of graphics and display-control features.

Taken as a whole, the 1752A boasts one of the most comprehensive display packages in the instrumentation industry. When the possibilities of the software are combined with the unique touch-sensitive screen, the result is a powerful set of tools for the programmer and operator alike.

Section 9
Options

CONTENTS

Introduction 9-3
Peripherals 9-4
Accessories 9-5
Options 9-6
User Information 9-9
  Installing Hardware Options 9-9
  -002 Parallel Interface Board 9-13
  -004/-005 Magnetic Bubble Memory 9-25
  -006/-007 Memory Expansion Modules 9-29
  -008 IEEE-488/RS-232-C Interface 9-33
  -009 Dual Serial Interface 9-37
  -010 Analog Measurement Processor 9-55
  -011 Analog Output 9-63
  -012 Counter/Totalizer 9-69
  1760A and 1761A Disk Drive Systems 9-81
  1765A Winchester Disk Drive 9-89
  1702A Extender Chassis and Option
  -013 Mainframe Interface 9-97

INTRODUCTION

This section of the manual gathers into one area all information about options available at the time of printing. Because new options are always being investigated and released throughout the life of the instrument, this section is expected to grow between printings of the manual.

When a new option becomes available, first shipments are supported by "User Information" sheets, which can be incorporated into this section after the option is installed. The next printing of the manual will then include the new sheets in this section. Be sure to contact your local Fluke representative for the latest information about available options.

The items discussed in this section are not included in the shipment unless they are ordered at the same time as the 1752A. If specified on the order, options will be installed at the factory. Otherwise, they may be packaged separately.

PERIPHERALS

All the peripherals listed here are separate products, and can be ordered by the model numbers shown.

1702A	I/O Extender Chassis
1760A	Disk Drive System, 400K Byte
1761A	Dual Disk Drive System, 800K Byte
1765A/AB	Winchester Disk Drive, 10M Byte
1780A	InfoTouch Display

ACCESORIES

Y1700 Keyboard

Y1706 Ten-pack of Blank Unformatted floppy disks (Certified)

P/N 533547
Pad of 50 Programmers Worksheets

Y1711 Reinforced Shipping Case

Y1704 Circuit Board Extender

IEEE-488 Cables

Y8021 Shielded, 1 meter
Y8022 Shielded, 2 meters
Y8023 Shileded, 4 meters

RS-232C Interface Cables

Standard (For DCE devices)
Y1707 2 meter
Y1708 10 meter

Null Modem (For other DTE devices)
Y1702 2 meter
Y1703 4 meter
Y1705 0.3 meter

Printer Cable

For connecting a serial printer.

Y1709 2 meter

I/O Extender Cable

For connecting the 1752A and 1702A.

2402A-502

Rack Mount Kits

Y1790 Rack Mount Kit with 24-inch slides
Y1794 Rack Mount Kit with 18-inch slides

Side Carrying Handle

Y1795

OPTIONS

Options are listed by a unique three-digit number appended to the product family identifier. For example, the -004 option 256K Byte Bubble Memory option is ordered using model number 17XXA-004.

Memory Expansion

Memory Expansion options greatly increase the available on-line storage capabilities of the 1752A. Memory Expansion Modules can be placed in any of the five unused options slots in the card cage. The maximum dynamic RAM configuration increases the total on-line system memory to about 2.6 megabytes; Bubble Memory can provide up to approximately 1.3 megabytes. Combinations are possible; please consult the Option Configuration Table later in this section for complete details.

-004 256K-Byte Bubble Memory
-005 512K-Byte Bubble Memory
-006 256K-Byte RAM Expansion
-007 512K-Byte RAM Expansion

Interface Additions

Interface options expand the Input/Output possibilities of the 1752A.

-002 Parallel Interface (slots 1,3,4,5)
-008 IEEE-488/RS-232C Interface (slots 4,5)
-009 Dual Serial Interface (slots 1,3,4,5)

Configuration Information

Use this table to determine allowable mainframe configurations of available hardware options. An dot in a column indicates the slots that the option can be placed in. For example, if all available slots are used for 512K byte RAM expansion memory modules, the system has the maximum memory configuration for this type of memory: 2.6M bytes, but no slots are available for other modules. On the other hand, slots 1, 3, 4, and 6 could be used for additional memory, resulting in 2.1M bytes of additonal memory, and slot 5 would still be available for one of the I/O options.

Additional expansion for measurement and control options are provided by the 1702A Extender Chassis. Refer to Option Installation, in the 1702A Instruction Manual for allowable option configurations.

Option Configuration Table

1752A-013 Mainframe Interface Assembly	1752A-012 Counter-Totalizer	1752A-011 Analog Output	1752A-010 Analog Measurement Processor	Dual Serial Interface -009	IEEE-488/RS-232C Interface -008	512K byte RAM Expansion -007	256K byte RAM Expansion -006	512K byte Bubble Memory -005	256K byte Bubble Memory -004	Parallel Interface -002	SLOTS
●		●	●	●		●	●	●	●	●	1
Reserved for Video/Graphics/Keyboard Interface											2
●			●	●		●	●	●	●	●	3
●	● *	● *	●	●†	●†	●	●	●	●	●†	4
●	● *	● *	●	●†	●†	●	●	●	●	●†	5
						●	●	●	●		6
Reserved for Single Board Computer											7

● Allowable Slot for Option

† Non I/O must be used in slot above

* Takes up two slots. No board in slot above

Note:

Analog Measurement Processor is shipped in slot 5. Normally it may be used in any of the Input/Output Options slots. Slot 6 has no Input/Output access.

Software

For increased flexibility, these software options are available to allow programming the 1752A in languages other than Interpreted BASIC, which is supplied as the standard programming language. Each language option is supplied as a floppy disk with an accompanying Programming manual.

-201 Assembly Language Software Development System
-202 FORTRAN Software Development System
-203 Compiled BASIC Software Development System
-205 Extended BASIC Software Development System

1752A USER INFORMATION
Installing Hardware Options

INTRODUCTION

This information is provided to assist you in installing hardware options into the 1752A Data Acquisition System. All options are installed in the same way. The instructions here describe how to install options into the 1752A's card cage and give general directions on how to check the options out.

Some of the options require some initial set-up. Be sure to refer to the appropriate option User Information for a module's unique requirements before you start the installation.

If a new module fails to perform correctly on first applying power, be sure to recheck your work. It may be that a small but important step was missed, resulting in a failure of the new module to operate.

PRE-INSTALLATION CHECKOUT

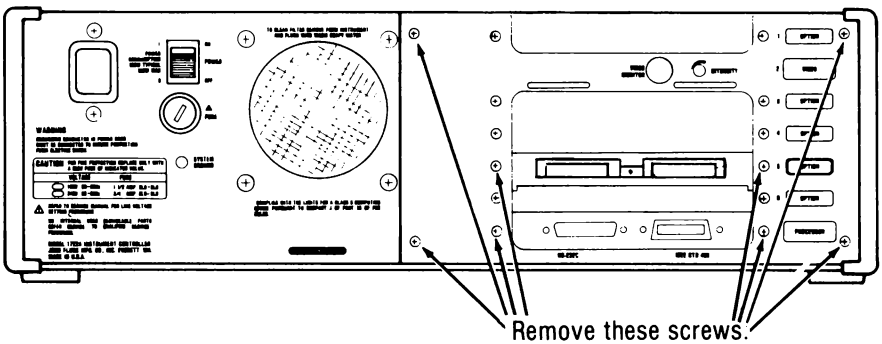
Inspect the shipping carton for damage, and notify the shipper immediately if it appears to have been damaged. Unwrap the module and inspect it for damage. If everything seems to be in order, set any selection switches or jumpers that are unique to the module. This information is available in the individual sections that cover each of the options.

INSTALLATION

1. Refer to the option User Information provided with the module and perform any preliminary set-up steps.

2. Power down the 1752A, and remove the ac line cord.

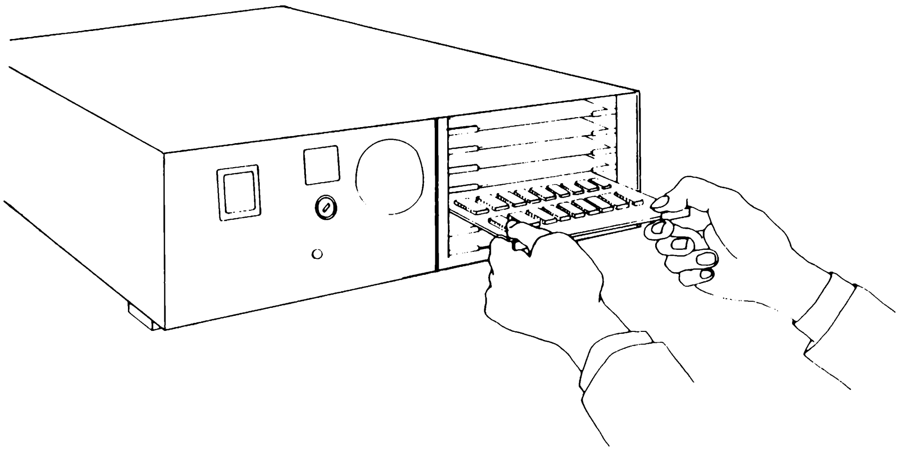
3. Remove the rear card cage cover, illustrated below. Depending on which modules are already installed, more screws than those indicated may also have to be removed. The six screws shown here are those that are removed from a 1752A with no options installed.



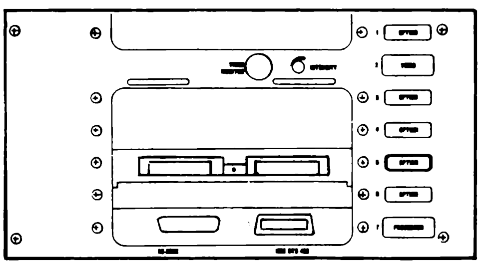
4. Determine which location to use for the option:

Memory Options:	Any open slot
17XX-008 Option:	Slot 4 or 5 only
Other Input/Output Options:	Slots 1, 3, 4, or 5 only

5. Carefully slide the option module into the card cage. Make sure the module is fully seated in the card cage so that it makes solid contact with the card-edge connector.



6. If an input/output option is being installed, remove the two screws holding the solid oval plate onto the card cage cover. After reinstalling the card cage cover, use the two screws to attach the shield plate supplied with the option, to the card cage cover. Afix the identification sticker to the outlined area on the card cage cover.



7. Reinstall the power cord.

8. Once installation is complete, power up the 1752A and test the module. Refer once again to the User Information provided with the module for any special requirements or procedures.

9. In case of problems with any new option, recheck your work to insure that switches are set properly, and that the correct jumpers are in place. Be sure that the board is fully seated in the card cage. If everything is in order, but the failure continues, refer to Appendix G, System Diagnostics, for troubleshooting information, or call your local Fluke Service Center.

1752A USER INFORMATION

OPTION 17XXA-002
PARALLEL INTERFACE BOARD

INTRODUCTION

The 17XXA-002 Parallel Interface (PIB) adds two 16-bit parallel interfaces to the 1752A. A maximum of three PIBs may be installed in one 1752A system, for a total of six 16-bit ports. The PIB may be installed in slots 1, 3, 4, or 5.

The PIB can adapt to some of the most unusual interface requirements of connected devices. Using software drivers supplied by Fluke, the PIB provides bidirectional transfer of bits (for monitoring and controlling status), 16-bit words (for communication with BCD instrumentation), or 512-word blocks (for maximum-speed data transfer of 80K words per second). Handshake or strobe protocols may also be selected under the control of the user's program.

The PIB is controlled by subroutines supplied in the PIBLIB library on the 1752A System Disk. The subroutines may be called from programs written in several programming languages, including Fluke Interpreted BASIC, Compiled BASIC, Extended BASIC, and FORTRAN.

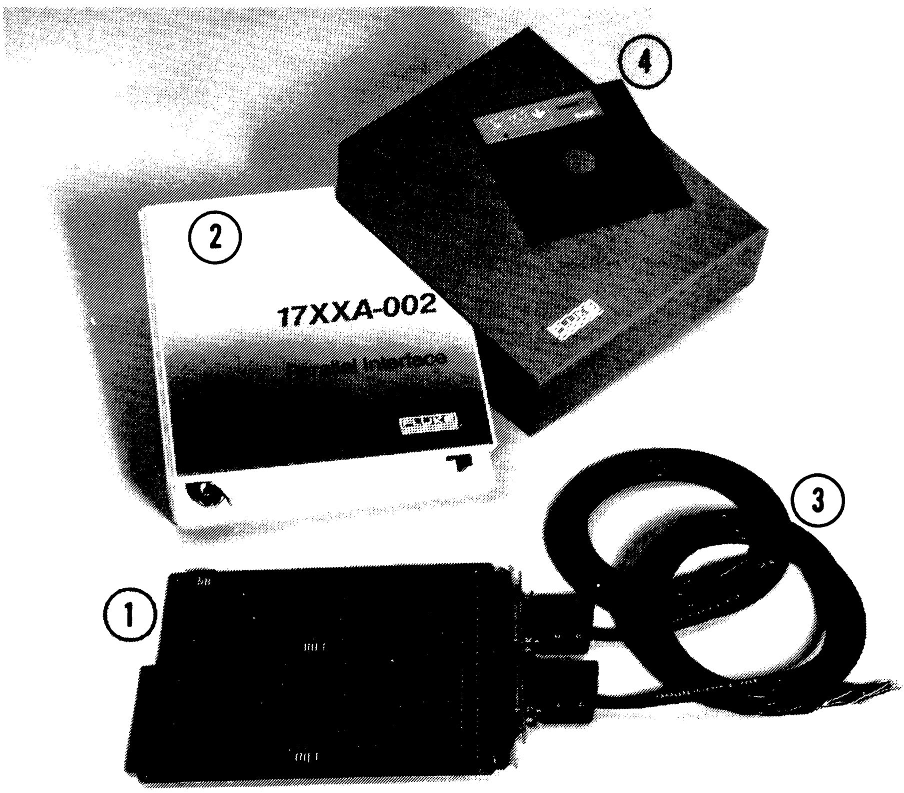
INCLUDED WITH THE OPTION

The photograph and table below show the items included with the 17XXA-002 option. The 17XXA-002 Parallel Interface manual provides complete documentation for the PIB.

The floppy disk supplied with the PIB allows the PIB to operate in the Fluke 1720A Instrument Controller. The disk is not needed when the PIB is used in the 1752A. The PIBLIB software library and drivers are already supplied on the 1752A system disk.

17XXA-002 PARALLEL INTERFACE OPTION CONTENTS

NUMBER	ITEM	JOHN FLUKE PART NUMBER
1	Parallel Interface Module	611947
2	Instruction Manual	732230
3	(2) Parallel Interface Cables	733907
4	1720A PIB Software Disk	630699



SPECIFICATIONS

HARDWARE SPECIFICATIONS

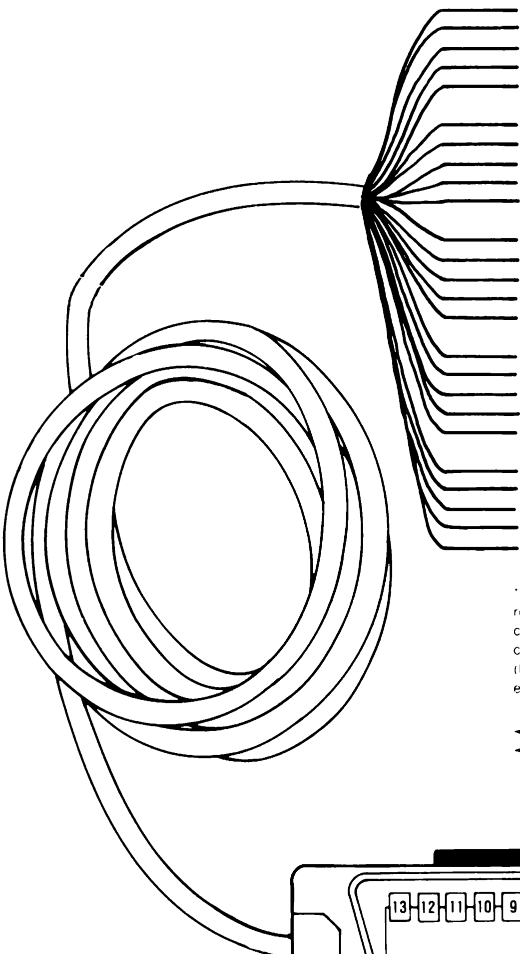
Ports	Two independent 16-bit parallel ports, 25-pin D-type subminiature female pin connectors.
Logical Interface	Memory mapped, two memory locations per port: data and status/control.
Line Characteristics	Data I/O lines are terminated resistively with diode input protection, 2400 ohms to +5V, 5000 ohms to ground.
Line Sense	Independent jumper-configurable active sense level for each control line, and for input and output data lines.
Data Out	Low: <0.4V @ 48 mA High: >2.4V @ -0.4 mA
Data In	Low: <0.8V. High: >2.0V.
Control Out	Low: <0.4V @ 8 mA. High: >2.4V @ 400 uA
Control In	Low: <0.4V High: >2.4V
Control Lines (each port)	PCTRL (output) PFLAG (input) PDIR (output) POEN (input)
Temperature	0 - 55 C storage 10 - 40 C operating
Dimensions	190.5 mm x 254.0 mm x 1.6 mm (7-1/2" x 10" x 1-1/16")
Supplied Cable	1.98 m (6.5 ft) of 25 conductor 26-gauge stranded copper wire.

SOFTWARE SPECIFICATIONS

Drivers	Supplied on System Software Disk: Read/Write Bit Read/Write Words (16 bits) Read/Write Blocks (multiple words)
Subroutine Library	Supplied on System Software Disk: chkbit - Check a bit clrbit - Clear a bit setbit - Set a bit rdwrd - Read a word wtwrd - Write a word rdblk - Read a block wtblk - Write a block frdblk - Fast read a block fwtblk - Fast write a block popen - Open a port pclose - Close a port
Control Modes	Mode 0: No Handshake Mode 1: Input Handshake Mode 2: Full Output Handshake Mode 3: Strobe Output Handshake
Handshake Modes	No Handshake Full Handshake Input Strobe Input Handshake Output Strobe Output

INTERFACE DESCRIPTION

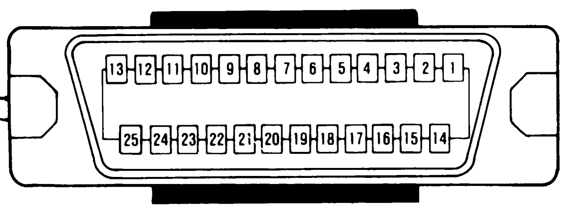
Port lines on the PIB provide bidirectional data transmission between the 1752A and compatible external devices. The drawing of the cable connector (below) names the signals and lists the logic states for each of the lines at the interface.



PIN	MNEMONIC	WIRE COLOR	DIRECTION
1	Shield	Ground*	-
2	DP(13)	Red	↔
3	DP(8)	Yellow	↔
4	DP(11)	Green	↔
5	DP(10)	Violet	↔
6	DP(9)	Grey	↔
7	DP(4)	Red White	↔
8	DP(7)	Yellow White	↔
9	DP(6)	Green White	↔
10	DP(5)	Violet White	↔
11	DP(0)	Grey White	↔
12	DP(3)	Yellow Red	↔
13	DP(2)	Green Red	↔
14	DP(14)	Violet Red	↔
15	DP(15)	Grey Red	↔
16	DP(12)	Red Green	↔
17	RETURN	Red Black	-
18	RETURN	Yellow Black	-
19	RETURN	Green Black	-
20	RETURN	Violet Black	-
21	PFLG	Red Grey	←
22	PIOE-	Yellow Grey	←
23	PCTRL	Green Grey	→
24	PDIR	Violet Grey	→
25	DP(1)	Grey Green	↔

*Pin 1 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 enough current to interfere with data transmission.

↔ indicates bidirectional data lines
← indicates input to PIB
→ indicates output by PIB



SOFTWARE LIBRARY (PIBLIB)

The next few pages summarize the subroutines that control the PIB. The required parameters are described immediately following the descriptions of the subroutines.

CHKBIT Check bit

Parameters: port, bit, bool

Description: This routine checks a particular bit on a Parallel Interface Port. It is equivalent to reading the port and then isolating the specified bit position.

CLRBIT Clear bit

Parameters: port, bit

Description: This routine clears to zero a particular bit on a selected port. It is equivalent to reading the port, ANDing the bit with 0 in the appropriate position, and then writing the data word back out to the port.

SETBIT Set bit

Parameters: port, bit

Description: This routine sets a particular bit (output = high) on a selected port. It is equivalent to reading the data latched at the port, ORing the bit in the specified position with 1, and then writing the data word back out to the port.

RDWORD Read word

Parameters: port, word

Description: This routine reads a selected port and writes the value into a variable.

WTWORD Write word

Parameters: port, word

Description: This routine writes a word from a variable to a specified port.

RDBLK Read block

Parameters: port, block, count

Description: This routine reads multiple words from a port into an array.

WTBLK Write block

Parameters: port, block, count

Description: This routine writes multiple words to a port from an array.

FRDBLK Fast Read block

Parameters: port, block, count

Description: This routine reads a block of data from a port, as fast as possible. It does not perform the error checking that is done in a normal read of a block using RDBLK.

FWTBLK Fast Write block

Parameters: port, block, count

Description: This routine writes a block of data to a port as fast as possible. It does not perform the error checking that is done in a normal block write using WTBLK.

POPEN Open a port

Parameters: port, mode, mask, timeout

Description: POPEN opens a port in preparation for data transfer at the interface.

PCLOSE Close a port

Parameters: port

Description: PCLOSE closes the specified port and returns the hardware to a passive state similar to the power-up condition.

Parameters

When the routines are used in a program, one or more parameters are specified by the programmer. All parameters must be specified as integers.

PORT — The port number for the routine to operate on, expressed as an integer in the range 0 - 15.

MODE — The Parallel Interface module operates in one of four modes: No Handshake, Full Handshake, and Strobe Input and Output Handshakes. With the exception of No Handshake, the handshakes synchronize incoming and outgoing data. The table below lists the handshake names and the mode number that the software recognizes.

HANDSHAKE MODES

NAME	DEFINITION	MODE
—	No Handshake	0
HNDSHKIN	Full Handshake Input	1
STROBEIN	Strobe Input	1
HNDSHKOUT	Full Handshake Output	2
STROBEOUT	Strobe Output	3

TIME OUT — The wait time before an incomplete handshake is terminated.

BIT — The bit number to be checked, read, or written to, expressed as an integer.

BOOL — When a bit or word is read or checked, as in the CHKBIT routine, the routine places a value into the variable specified as BOOL.

DIRECTION MASK — An integer that indicates the desired transmission direction of each bit on the port.

BLOCK	This parameter is the array to which data will be transferred using the Block subroutines RDBLK, WRTBLK, FRDBLK, and FWTBLK.
COUNT	Indicates how many array elements to transfer.

CONCLUSION

For complete information about the Parallel Interface, refer to the manual shipped with the PIB. That manual includes:

- Specifications
- Interface timing
- Sample programs
- Performance Testing
- Theory of Operation
- Schematic
- Interface Description
- Software Library Description

1752A USER INFORMATION

Option 17XXA-004/005

Magnetic Bubble Memory

INTRODUCTION

The Option 17XXA-004/-005 Bubble Memory Modules provide additional memory for the 1752A.

Like a floppy disk, the Bubble Memory is treated by FDOS as a file-structured device. Information is retained in the device when the power is turned off.

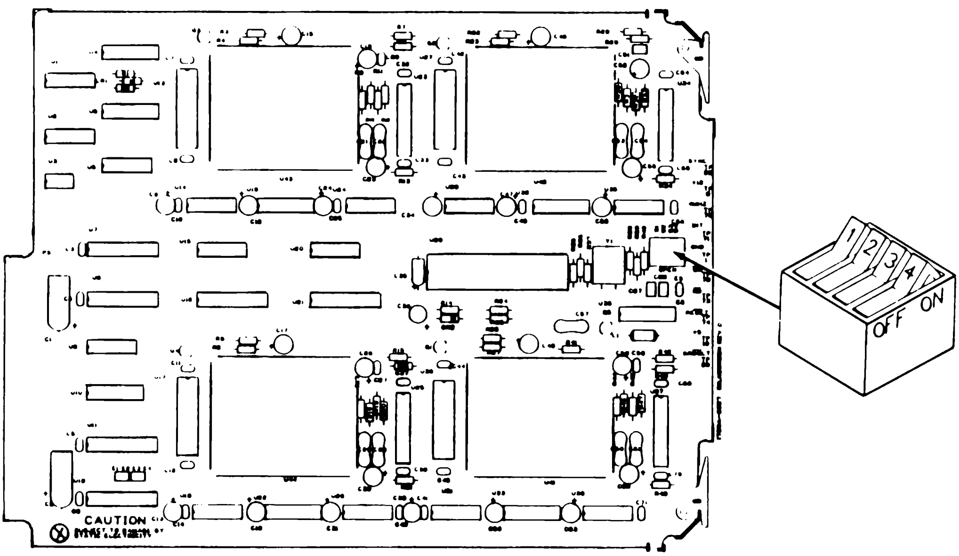
The 17XXA-004 Bubble Memory Module contains 256K bytes of memory; the 17XXA-005 module contains 512K bytes. The maximum amount that can be installed in a system is 1.5M bytes (any combination of three modules).

PRE-INSTALLATION CHECKOUT

Inspect the shipping carton for damage. Notify the shipper immediately if the carton appears to have been damaged in shipping. Unwrap the module and inspect it for damage. If everything seems to be in order, go on to the next step, setting the board's address switch.

Board Addressing

In order for the Operating System (FDOS) to operate properly using bubble memory, there must be a unique address associated with each Bubble Memory Module installed. Use the drawing below to locate the board address switch (SW1).



The 1752A has five slots available for options. Bubble Memory Modules can be installed in slots 1, 3, 4, 5 or 6. The SW1 address switch settings determine the device names, and the switch settings are identical for Option -004 (256K bytes) or Option -005 (512K bytes).

DEVICE NAME	ADDRESS CODE	SWITCH POSITIONS 1	2	3	4
MB0:	111X	on	on	on	X
MB1:	110X	on	on	off	X
MB2:	101X	on	off	on	X
MB3:	100X	on	off	off	X

NOTES: 1. "1" = on, "0" = off, "X" = don't care.

2. Four device names are available. However, only three are used at a time because only three modules can be installed at one time.

INSTALLATION AND CHECKOUT

1. Once address switch SW1 has been set, follow the directions in the Options Section "Installing Hardware Options" to install the module into the 1752A's card cage. Be sure to turn the power off before beginning.

2. Power up the 1752A and insert the 1752A System Software disk. If required, use the System Generation Utility program (SYSGEN) to make a new System Software disk that includes the Bubble Memory driver. Refer to Section 3 of the System Guide for instructions.

3. When a new System Disk has been generated, press the RESTART button and let the disk load the new FDOS.

4. Use the File Utility Program (FUP) to format the Bubble Memory Module. Type MBx:/F, where x is the device number given the module by its address switch setting. See Section 4 of the System Guide for complete details about using FUP.

5. Test the new memory board by running the Bubble Memory diagnostic, as described in Appendix G, "System Diagnostic Software", or by transferring files to and from the bubble memory device using FUP.

6. If the diagnostic test executes successfully, your Bubble Memory Module is now available for use.

7. If any trouble develops, first make sure that the address selection switch is set properly, and that the module is seated firmly in the card cage. If everything seems to be in order, but the failure continues, refer to Appendix G, "System Diagnostic Software", or call your Fluke Technical Service Center for assistance in tracking down the problem. The Bubble Memory Module is included in Fluke's Module Exchange Program.

1752A USER INFORMATION

Option 17XX-006/007 Memory Expansion Module

INTRODUCTION

The Option 17XX-006 and 007 Memory Expansion Modules provide additional memory for the 1752A. This added memory can also be configured as Electronic disk.

The Operating System treats memory configured as E-Disk as an electronic version of a floppy disk. This means that files are stored and retrieved from E-Disk in a formatted fashion like a floppy disk. See the File Utility Program in Section 4 of the System Guide for instructions on how to configure E-Disk space.

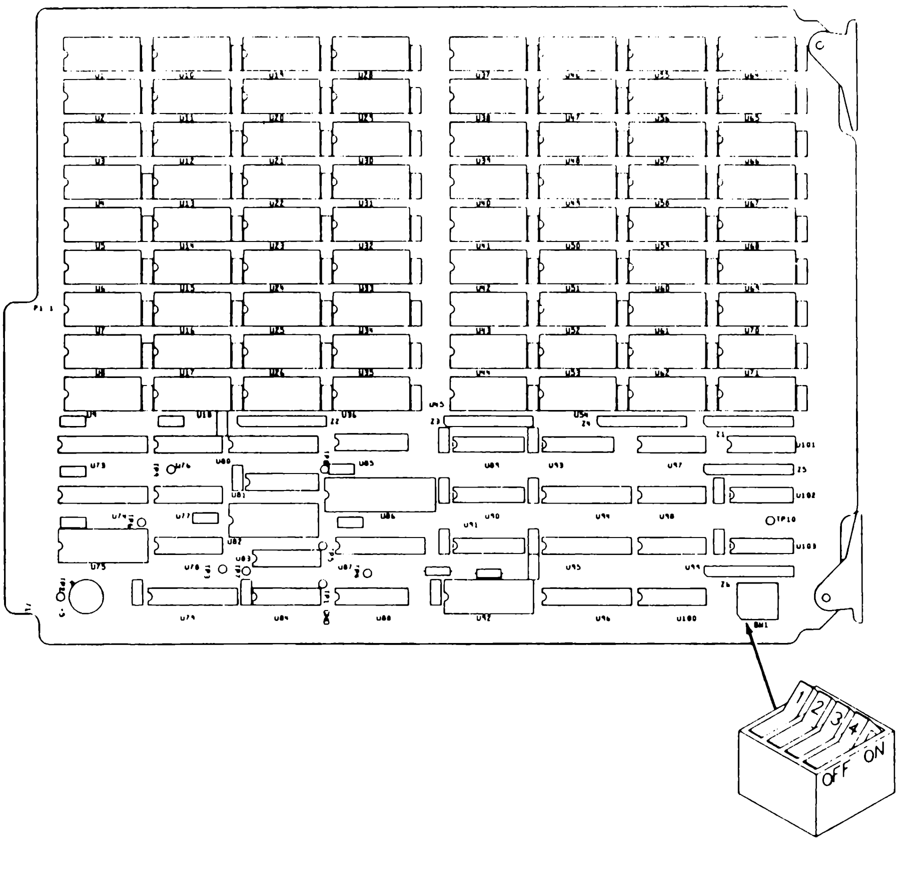
The 17XX-006 Memory Expansion Module contains 256K bytes of dynamic RAM; the 17XX-007 module contains 512K bytes. Any combination of up to four modules can be installed in a 1752A at a time. If all four slots are filled with 17XX-007 boards, then the total expansion memory added to the 1752A would be over 2.1M bytes.

PRE-INSTALLATION CHECKOUT

Inspect the shipping carton for damage. Notify the shipper immediately if the carton appears to have been damaged in shipping. Unwrap the module and inspect it for damage. If everything seems to be in order, go on to the next step, setting the board's address switch.

Board Addressing

In order for the Operating System to operate properly using expanded memory, each memory board installed must have unique addresses set. Each module has a memory switch, located in the drawing below.



The 1752A has four slots available for options. Expansion memory modules can be installed into any or all of them. To assure proper operation of diagnostics software, the first module added should be given the address for Unit One as shown in the table below. Subsequently added modules are given addresses in ascending unit number order. The SW1 address switch settings shown below are identical for Option -006 (256K bytes) or Option -007 (512K bytes).

UNIT NUMBER	ADDRESS CODE	SWITCH POSITIONS			
		1	2	3	4
1	1110	off	off	off	on
2	1100	off	off	on	on
3	1010	off	on	off	on
4	1000	off	on	on	on

NOTES

1) "0" = on and "1" = off on the option's address switch label.

2) Although the 1752A may operate properly with addresses set out of order, setting them in the recommended order ensures that diagnostic software can correctly identify faulty components, and will prevent possible contention problems when mixing options -006 and -007.

Example:

Two -007 Options, and two -006 Options are to be installed. In this case, we will set the addresses for the larger memory sizes first by setting their switches to 1110 and 1100. Next, the first -006 option's switches are set to 1010, and the second one to 1000.

These settings leave a 256K byte gap between the memory addresses occupied by the -006 modules. This gap is transparent when the module is in use.

INSTALLATION AND CHECKOUT

1. Follow the directions in the Options Section titled "Installing Hardware Options" to install the module into the 1752A's card cage. Be sure to turn the power off.

2. To check the new memory module, power up the system and load the Operating System software. Observe the amount of memory message that appears when FDOS loads. The memory message should indicate the additional memory that is now available, both in bytes and blocks. (1 block = 512 bytes.)

3. To exercise the new memory, use the File Utility Program to configure all available free blocks as E-Disk, then transfer a large amount of files to the E-Disk, and see that they can be read to the screen. If everything is in order, this is an adequate check that the Memory Expansion Module is operational. Section 4 of the 1752A System Guide, Devices and Files, explains all the operations of the File Utility Program.

4. If any trouble develops, first recheck your work. Make sure that the address selection switches are set properly, and that the module is properly seated into the connector on the motherboard. If everything seems to be in order, refer to Appendix G, System Diagnostics, or call your Fluke Technical Service Center for assistance in tracking down the trouble. The Memory Expansion module is included in Fluke's Module Exchange Program.

1752A USER INFORMATION

Option 17XXA-008
IEEE-488/RS-232C Interface

INTRODUCTION

This section of the 1752A System Guide covers the Option 17XXA-008 IEEE-488/RS-232-C Interface Module. The module provides the 1752A with one additional IEEE-488 port and one additional RS-232-C port.

As shipped, the standard 1752A has a single IEEE-488 port and one RS-232-C port. The IEEE-488 port has the device name GP0: when used as a serial device (output only), and the device name Port 0 when used by a program as an instrument port. The standard configuration RS-232-C port has the device name KB1:.

The IEEE-488 port on the -008 option has the device name GP1: or Port 1, and the RS-232 port has the device name KB2:. See Section 4 of the System Guide for more information on devices.

PRE-INSTALLATION CHECKOUT

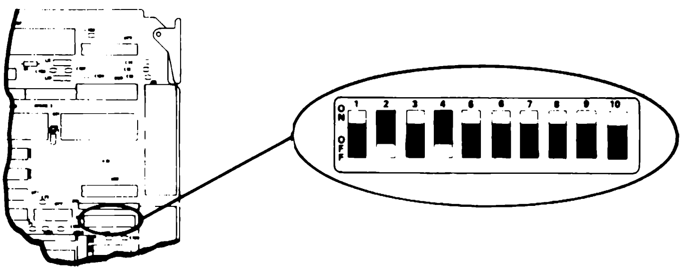
Inspect the shipping carton for damage. Notify the shipper immediately if the carton appears to have been damaged in shipping. Unwrap the module and inspect it for damage. If everything seems to be in order, go on to the next step, Installation.

INSTALLATION

1. Refer to the drawing below to locate and set the configuration switches. The initial setup establishes the module's IEEE-488 address as 0, and its function as "system controller". This switch setting also sets the RS-232 port to 4800 baud for power up, but the baud rate can easily be changed later using the Set Utility Program. See Section 5 of the System Guide for details.

NOTE

Both the standard IEEE-488 port and the one added by the -008 option can be set up as "system controller", because the two ports are effectively two separate systems. However, if both of them will be connected to the same bus, then one of the ports must be set up as "idle controller".



SWITCH 1

1	2	3	4	5	6	7	8	9	10	
—	S2	S1	S0	SC		A4	A3	A2	A1	
x										unused
										IEEE-488 ADDRESS
						0	0	0	0	0
						0	0	0	1	1
						0	0	1	0	2
						0	0	1	1	3
						0	1	0	0	4
						0	1	0	1	5
						0	1	1	0	6
						0	1	1	1	7
						1	0	0	0	8
						1	0	0	1	9
						1	0	1	0	10
						1	0	1	1	11
						1	1	0	0	12
						1	1	0	1	13
						1	1	1	0	14
						1	1	1	1	15
										IEEE-488 CONTROLLER
				0	0					System Controller
				1	1					Idle Controller
										BAUD RATE
	0	0	0							110
	0	0	1							300
	0	1	0							600
	0	1	1							1200
	1	0	0							2400
	1	0	1							4800
	1	1	0							9600
	1	1	1							19200

NOTE: 1 = on; 0 = off

2. Once the switch has been set, use the directions in the Options section "Installing Hardware Options" to install the -008 option into the 1752A.

3. Power up the 1752A and test the new interface by using the System Diagnostic software. Appendix G of the System Guide explains how to use the System Diagnostic software to test the -008 option.

4. In case of problems with the new module, recheck your work to ensure that the board is fully seated in the card cage, and that port connectors are attached securely. If everything is in order but the failure continues, refer to Appendix G for troubleshooting information, or call your local Fluke Service Center. The IEEE-488/RS-232 Interface module is included in Fluke's Module Exchange Program.

1752A USER INFORMATION
OPTION 17XXA-009
DUAL SERIAL INTERFACE

INTRODUCTION

The 17XXA-009 Dual Serial Interface (DSI) provides the 1752A with two additional serial communications ports. The ports are addressed as SP0: through SP9: via the Set Utility program and high-level languages. The ports are treated similarly to KB0: and KB1: on the Single Board Computer (SBC), and KB2: on the IEEE/RS-232-C Option (-008).

Up to three DSI modules can be installed in the 1752A. Each port may be configured for these electrical interfaces:

- RS-232-C
- RS-422
- 20 mA Current Loop

Each port buffers incoming and outgoing data and signals the external device when the buffers are nearly full to prevent loss of data. The signaling method or protocol may be selected as discussed below. The ports are controlled by a microprocessor which reduces the overhead on the Single Board Computer (SBC). System throughput is a function of the data being transferred at the floppy disk and the IEEE-488 and KBx: ports. If the load from these devices is heavy, external devices will be held off more frequently regardless of the data rate selected.

The Operating System (FDOS) which is supplied on the 1752A System Disk includes a device driver for the -009 option. This operating system must be used in order for your software to be able to access the Dual Serial Interface. If necessary, you can reconfigure the operating system by using this disk and following the instructions in Section 3 of the 1752A System Guide.

The disk also includes the Serial Port Software Driver (SPIO.OBJ). This driver allows a BASIC program to directly monitor and control the various lines of the serial interface, including RS-422 ENABLE, which some users may require. Information later in this section describes the operation of the driver.

PRE-INSTALLATION CHECKOUT

Inspect the shipping carton for damage. Notify the shipper immediately if the carton appears to have been damaged in shipping. Unwrap the module and inspect it for damage.

INSTALLATION

The Option has been configured at the factory as follows:

Electrical Interface	RS-232-C
Data rate	4800 Baud
Data Bits	7
Parity	none
Stop Bits	2
Flow Control	off
Board Address	0 (SP0: and SP1:)

The jumpers and switches are set at the factory as shown in the figure on the next page. Jumpers JPR1, 2, 3, 4, 5, 6, 29, and 35 are not user-configurable and must remain in the positions shown. Check all jumpers and switches to be sure they match the factory configuration.

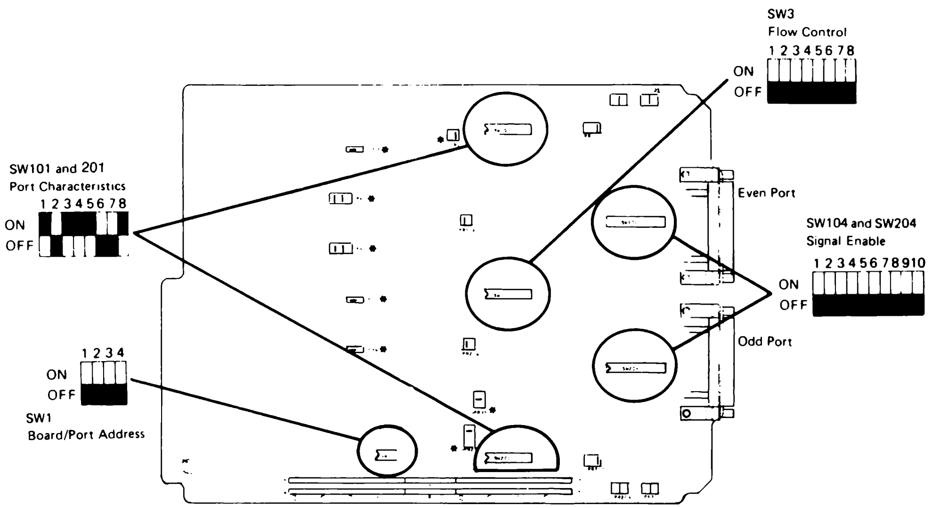
Use the directions in the Options Section of the System Guide “Installing Hardware Options” to install the -009 option into the 1752A.

If you have reconfigured the Operating System with the SYSGEN Program, make sure that the Dual Serial Interface Driver has been included. You can check this by rebooting the 1752A with your version of the Operating System and then running the CONFIG Program supplied on the standard System Disk.

The signals for all three interface types are available on each of the port connectors. Since this is a modification of the RS-232-C standard, the redefined pins can be switched off using SW104 and SW204 when RS-232-C is desired. This allows using cables with connections for signals defined by the standard to be used. Refer to the section "Electrical Interfaces" for connections to external devices.

In case of problems with the new module, recheck your work to ensure that the board is fully seated in the card cage, and that the port connectors, jumper positions, and switch settings are correct. If the failure continues, refer to Appendix G for troubleshooting information, or call your local Fluke Service Center. The Dual Serial Interface is included in Fluke's Module Exchange Program.

FACTORY CONFIGURATION



Note: * indicates jumpers that are not user-configurable.
Do not move these jumpers.

POWER ON CONFIGURATION

When the Dual Serial Interface is powered on, these things take place:

- The configuration switches are read.
- The RS-422 drivers are enabled.
- The DSI waits for input or output.

The initial states of the control lines are shown later. See "Electrical Interfaces - RS-232-C".

PROTOCOLS

XON/XOFF

The ASCII Standard defines two codes that may be used to control data transfer between devices. If the input buffer of the device receiving data is full or nearly full, XOFF is sent to the transmitting device to request the transmission be stopped. When the receiver can accept more data, the XON code is sent to resume the transmission. The Set Utility program can be used to enable or disable this protocol, and refers to the protocol as "stall input" and "stall output".

Secondary Request to Send (RS-232)

SRTS is a handshake line that is used for flow control with external devices that cannot respond to XON/XOFF codes. The polarity of SRTS is set by the Flow Control configuration switch (SW3).

RECONFIGURATION

The electrical interfaces and power-up configuration can be changed using the switches and jumpers described in the following tables. After the tables, the next sections discuss each interface and typical setups. The Set Utility program can also be used to change port characteristics. See Section 5 of the System Guide for details.

NOTE

The Port Characteristics and Flow Control switches are read only at power-up.

Board/Port Addresses (SW1)

BOARD	PORT	1 2 3 4
1	1, 0	0 0 0 0
2	3, 2	0 0 0 1
3	5, 4	0 0 1 0
4	7, 6	0 0 1 1
5	9, 8	0 1 0 0

ON = 1 = closed = enabled
OFF = 0 = open = disabled

J1 = EVEN Port
J2 = ODD Port

Port Characteristics (SW101 AND SW201)

1 2 3 4	
1 1 1 1	19200 Baud
1 1 1 0	19200
1 1 0 1	9600
1 1 0 0	7200
1 0 1 1	4800
1 0 1 0	3600
1 0 0 1	2400
1 0 0 0	2000
0 1 1 1	1800
0 1 1 0	1200
0 1 0 1	600
0 1 0 0	300
0 0 1 1	150
0 0 1 0	134
0 0 0 1	110
0 0 0 0	75

5 6 7 8		
1	Data Bits:	8
0		7
1 1	Parity:	even
1 0		odd
0 1		none
0 0		none
1	Stop Bits:	2
0		1

Flow Control (SW3)

1	2	3	4	5	6	7	8	
								Even Port
1								enable
0								disable
	1							active high
	0							active low
		x	x					not used
								Odd Port
				1				enable
				0				disable
					1			active high
					0			active low
						x	x	not used

Port Connector Signal Enable Switches (SW104 and SW204)

1	RS-422	Rx+
2		Rx-
3	20mA	Rx+
4		Rx-
5		I1-/-12v
6		I2-/-12v
7	RS-422	Tx+
8		Tx-
9	20mA	Tx+
10		Tx-

Clear To Send (CTS) Input to UART (JPR116 and JPR216)

Left	CTS
Right	CTS AND SRLSD

UART Receive Input (JPR117 and JPR217)

Left	20mA loop
Middle	RS-422
Right	RS-232-C

Voltage/Current Sources (JPR102, 103, 202, 203)

Left	20mA
Middle	OFF
Right	+12v

ELECTRICAL INTERFACES

RS-232-C

Maximums

Distance	50 ft
Data Rate	19200 Baud

Typical Applications

Data Communications Equipment (DCE)
(Use RS-232-C Cable)

Modems

Data Terminal Equipment (DTE)
(Use Null Modem Cable)

1780A
VT100
Printers (Use Null Modem Cable)

Port Connections

To DCE Pin		Circuit Function	From DCE Pin		Circuit Function
1	AA	Shield			
7	AB	Signal Common			
2	BB	Transmitted Data	3	BA	Received Data
4	CA	Request to Send	5	CB	Clear to Send
20	CD	Data Terminal Ready	6	CC	Data Set Ready
19	SCA	Secondary Request to Send			
11	UND	RS-232-B			
			22	CE	Ring Indicator
			12	SCF	Secondary Received Line Signal Detector
			8	CF	Received Line Signal Detector

The Clear To Send jumpers (JPR116 and 216) allow the CTS input to the UART to be either the CB circuit or the logical AND of CB and SCF. This feature is useful for external devices that use the SCF circuit as a "busy" indicator.

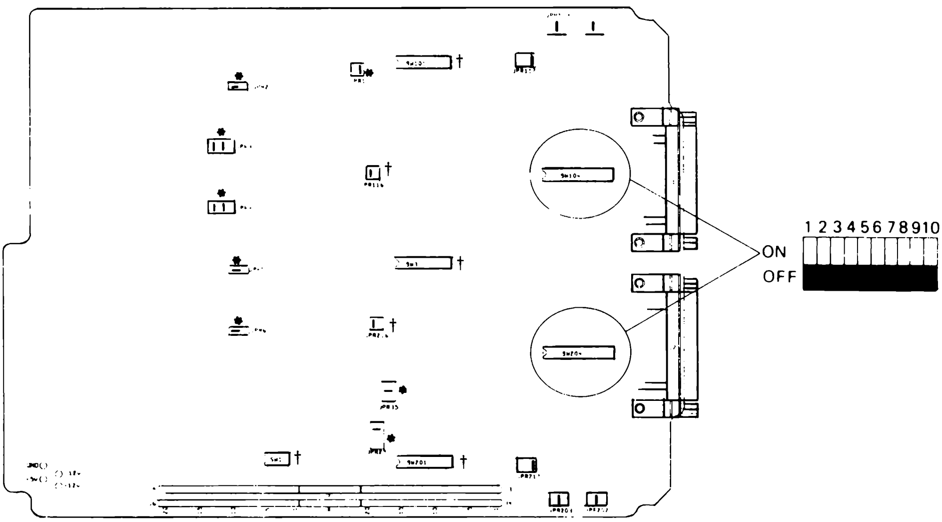
Power-on States of the Control Lines

SIGNAL	PIN	STATE
BB	2	MARK
CA	4	ON
SCA	11, 19	OFF
CD	20	ON

NOTE

Clear To Send (Pin 5) controls the transmission from the port. When it is ON, the corresponding UART is permitted to transmit. When it is OFF, the UART stops transmitting, beginning at the next character boundary. This behavior is a function of the UART hardware, and always applies. It cannot be disabled. Leave Pin 5 unterminated if not used by the receiving device.

RS-232-C CONFIGURATION



Notes: * *indicates jumpers that are not user-configurable. Do not move these jumpers.*

† *indicates switches or jumpers that can be set as desired. Refer to tables in the text for settings.*

RS-422

Maximums

Distance	4000 feet
Data Rate	19200 Baud

Typical Applications

2400B
1780A/AU

Protection Networks

The circuitry incorporates protection networks on the drivers and receivers to reduce susceptibility to high voltage transients and faults.

Port Connections

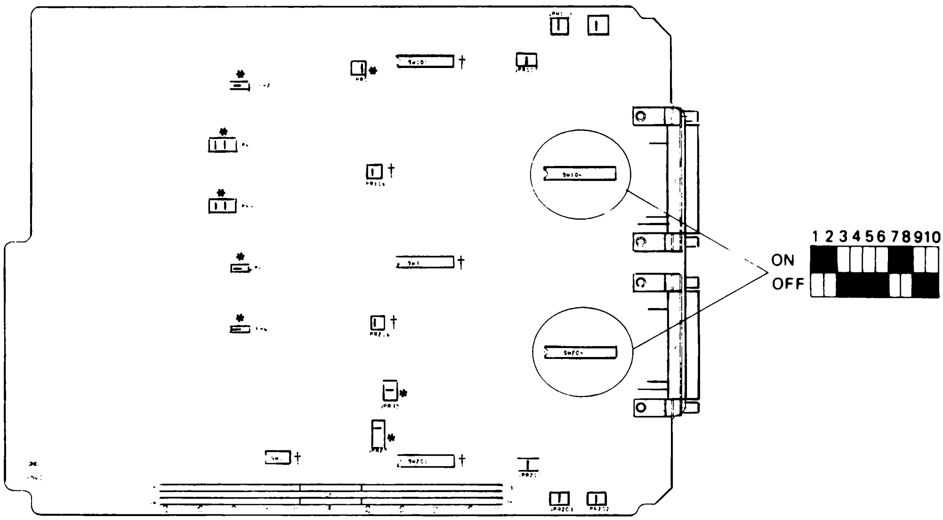
To External Device

PIN	SWITCH	SIGNAL
9	7	Tx+
10	8	Tx-
7		Signal Ground

From External Device

PIN	SWITCH	SIGNAL
14	1	Rx+
15	2	Rx-

RS-422 CONFIGURATION



Notes: * *indicates jumpers that are not user-configurable. Do not move these jumpers.*

† *indicates switches or jumpers that can be set as desired. Refer to tables in the text for settings.*

20 MA LOOP

Maximums

Distance	1000 feet
Data Rate	4800 Baud
Voltage	30 Vdc

Typical Application

Teletype (TTY)

Port Connections

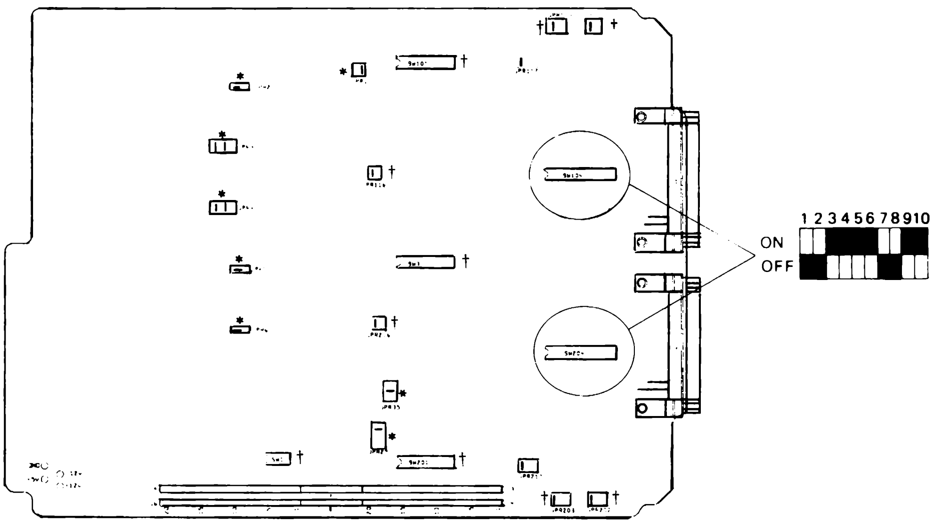
To External Device

PIN	SWITCH	SIGNAL
12	9	Tx+
23	10	Tx-
18	5	I1-/-12v
16	6	I2-/-12v

From External Device

PIN	SWITCH	SIGNAL
24	3	Rx+
25	4	Rx-
17	JPRx02	I1+/+12v
13	JPRx03	I2+/+12v

20 MA CURRENT LOOP CONFIGURATION



Notes: * *indicates jumpers that are not user-configurable. Do not move these jumpers.*

† *indicates switches or jumpers that can be set as desired. Refer to tables in the text for settings.*

USING THE DUAL SERIAL INTERFACE

The option -009 device driver in FDOS allows the programmer to use the DSI from all languages and utility programs just as any other serial device. The ports are specified as SP0: through SP9: depending on the board address selected.

Two additional routines are provided to gain more direct control of the interface lines. The user can read the current state of all the input and output lines and can set the output lines to any state desired. The user can also enable and disable the RS-422 drivers. These routines are supplied in the following files for each programming language:

LANGUAGE	FILENAME	SUPPLIED WITH
Interpreted BASIC	SPIO.OBJ	1752A System Disk
Compiled BASIC	BASIC.LIB	17XXA-203 Compiled BASIC
Extended BASIC	BASIC.LIB	17XXA-205 Extended BASIC
FORTRAN	FLUK22.LIB	17XXA-202 FORTRAN

PROGRAMMING IN BASIC

The routines can be used from Interpreted BASIC, by using the LINK statement as follows:

LINK "SPIO" ⟨RETURN⟩

To use the routines in Compiled BASIC, use the FIND command in the Linking Loader, as follows:

F BASIC ⟨RETURN⟩

Description

The module consists of two routines: SPGETS and SPSETS. SPGETS is called to get a "snapshot" of the inputs and outputs. A port number between 0 and 9 is specified, along with an integer word into which the status is to be stored by SPGETS. When it is called, SPGETS retrieves the current input and output status and returns the value to the caller in the specified integer variable. The state of each line is represented by a corresponding bit in the status word. The bit assignments are illustrated below.

SPSETS sets the state of the output lines. A port number between 0 and 9 is specified, along with an integer word which contains the control data to be output to the control lines. The bit assignments in the control word are identical to the assignments in the status word. The input portion of the status/control word is ignored by this function.

Usage

```
CALL SPGETS(port%, status%)
      INTEGER port%              ! 0 for SP0:
                                 ! 1 for SP1:
                                 ! ........
                                 ! 9 for SP9:
                                 !
              status%            ! RS-232 input/output line status
                                 ! bit
                                 ! value  pin       description
                                 ! -----  ---  -------------------------
                                 !    1    20  Data Terminal Ready
                                 !    2    19  Secondary Request To Send
                                 !    4    11  Undefined
                                 !    8     4  Request To Send
                                 !  128     -  RS-422 Output Enable
                                 ! ----    --  --------------------------
                                 !  256    12  Sec Rcv Line Sig Detector
                                 !  512    22  Ring Indicator
                                 ! 1024     8  Rcv Line Signal Detector
                                 ! 2048     6  Data Set Ready
                                 ! 4096     5  Clear To Send
-----------------------------------------------------------------

CALL SPSETS(port%, status%)
      INTEGER port%              ! 0 for SP0:
                                 ! 1 for SP1:
                                 ! ........
                                 ! 9 for SP9:
                                 !
              status%            ! RS-232 output line control
                                 ! bit
                                 ! value  pin       description
                                 ! -----  ---  -------------------------
                                 !    1    20  Data Terminal Ready
                                 !    2    19  Secondary Request To Send
                                 !    4    11  Undefined
                                 !    8     4  Request To Send
                                 !  128     -  RS-422 Output Enable
```

Errors

All errors are recoverable:

5100 FDOS function call failed (usually means option missing or faulty)

5101 invalid port number

External Effects

The RS-232 output lines may be changed. Turning off Request To Send inhibits transmission on the corresponding port.

FORTRAN PROGRAMMING

From FORTRAN, the routines are called in the same manner as they are in BASIC, except that an error parameter is passed as follows:

```
CALL SPGETS (port, status, error)

CALL SPSETS (port, status, error)
```

If the error value returned is non-zero, then one of the following has occurred:

No device driver in FDOS
Option missing or faulty
Illegal port number specified

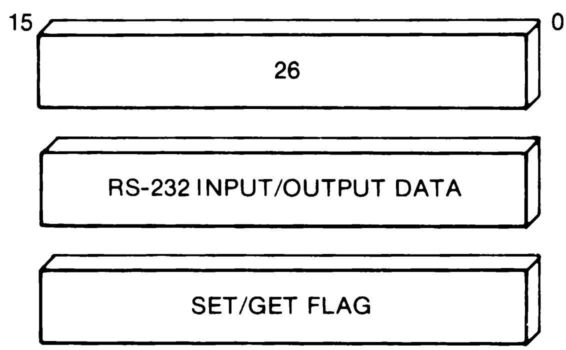
ASSEMBLY LANGUAGE PROGRAMMING

The FDOS driver for the option supports all the applicable functions, which are listed below. For a description of the call conventions, refer to the FDOS direct I/O functions for KBx: ports in the Assembly Language Manual. The functions are:

0	Read a Record
2	Write a Record
4	(unused)
6	Initialize Driver
8	Get Port Configuration
10	Set Port Configuration
12	Return Number of characters/lines in Input Buffer
14	Get a Character
16	Put a Character
18	Send a Break
20	(unused)
22	(unused)
24	Return Number of Characters in Output Buffer

Full control of the UARTs on the Dual Serial Interface option requires one additional FDOS call, described below.

26 READ/WRITE RS-232 LINES



Description

This function either sets or gets the the current state of the RS-232 signal lines. When the Set/Get flag is non-zero, the lower byte of the RS-232 data word is loaded into the latches that drive the RS-232 lines. When the set/get flag is zero, the entire RS-232 data word is returned with the current state of the RS-232 interface, including both input and output lines.

In addition, this word also controls the RS-422 data driver. When the 422EN bit is set to one (1), the RS-422 driver is active. When 422EN is set to zero (0), the RS-422 output drivers are tri-stated.

The set function has no effect on the input signal lines. They are always sampled directly during a get function.

Format of RS-232 Data Word

	(READ ONLY)								(READ/WRITE)							
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				CTS	DSR	RLSD	RI	SRLSD	422EN				RTS	SRTS	SRTS	DTR
CIRCUIT NAME				CB	CC	CF	CE	SCF					CA	—	SCA	CD
PIN NUMBER				5	6	8	22	12					4	11	19	20
				CLEAR TO SEND	DATA SET READY	RECEIVED LINE SIGNAL DETECT	RING INDICATOR	SECONDARY REC'D LINE SIG. DET.	RS-422 DRIVER ENABLE				REQUEST TO SEND	UNDEFINED	SECONDARY REQUEST TO SEND	DATA TERMINAL READY

1752A USER INFORMATION

OPTION 1752A-010

ANALOG MEASUREMENT PROCESSOR

INTRODUCTION

The 1752A-010 Analog Measurement Processor is a single-board, 14-bit analog-to-digital converter that can be software configured for a total of 32 single-ended channels, 16 differential channels, or a combination of both. Range and filtering are programmable for each channel. Conversion is accomplished with a microprocessor and a 16-bit digital-to-analog converter, using a successive approximation technique. The microprocessor controls input FET multiplexing, sample and hold, data conversion, auto-zero, auto-calibration, averaging, and range selection. The Analog Measurement Processor interfaces directly with the 1752A bus without isolation, and inputs are protected with fusible resistors.

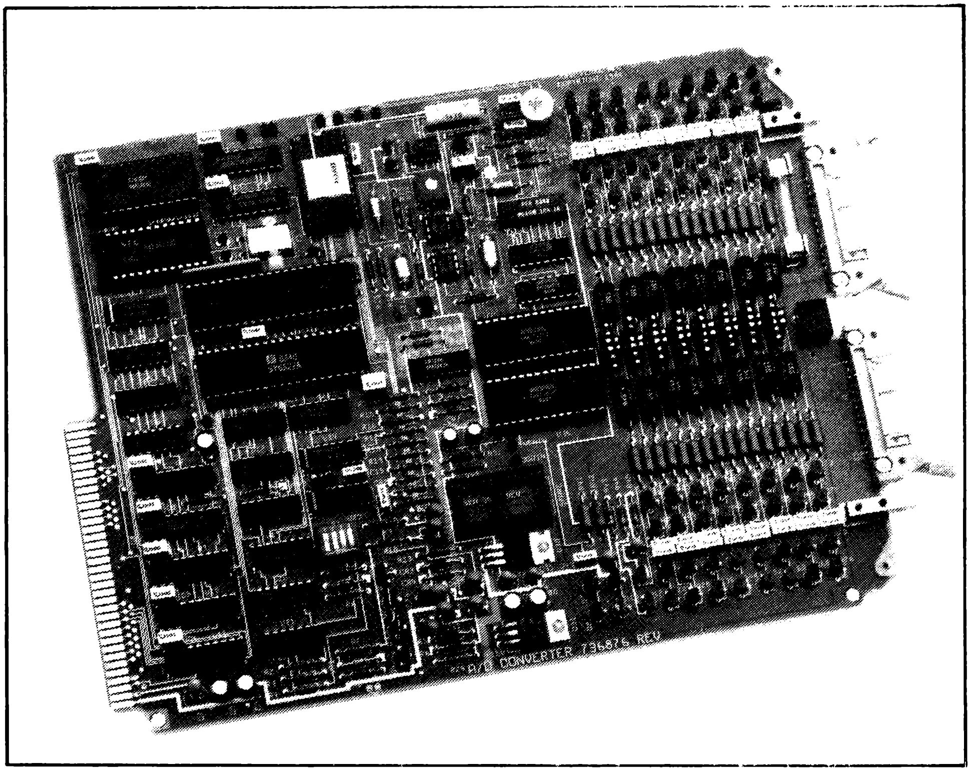
The Analog Measurement Processor can be mounted in slots 1, 3, 4, or 5 of the 1752A chassis. Analog inputs are accessible on two 26-pin connectors at the rear of the chassis.

Features of the Analog Measurement Processor include:

- 32 addressable input channels
- Two voltage ranges
- Two current ranges
- Single-ended and differential input modes
- Selectable filtering
- Synchronous and asynchronous scanning modes
- External phase locking to line frequency to increase noise rejection

Refer to the 1752A Data Acquisition and Control manual for further information about these features.

1752A-010 Analog Measurement Processor



SPECIFICATIONS

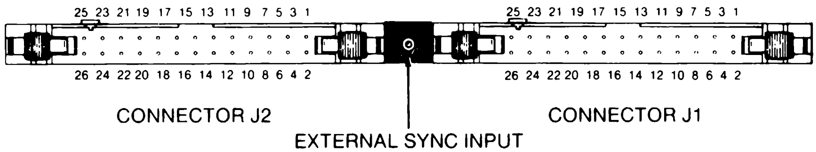
Number of Channels	32 single-ended or 16 differential (single-ended and differential channels may be mixed.) Maximum of four Analog Measurement Processors per 1752A. (128 single-ended channels).
Synchronization Modes	Internally Synchronized: 50, 60, or 400 Hz Externally Synchronized: 45 to 65 Hz, or 360 to 520 Hz Asynchronous
Ranges (full scale, each channel)	± 10.158V ± 1.0158V ± 67.718 mA 4 to 20 mA, displayed as 0 to 100% of scale
Reading Rate	Synchronized Modes: 400 readings/sec @ 50 Hz 480 readings/sec @ 60 Hz 400 readings/sec @ 400 Hz Asynchronous Mode: 1000 readings/sec
Accuracy (90 days, 10°C to 40°C)	10V Range: ±(0.02% of reading + 1.24 mV) 1V Range: ±(0.02% of reading + 248 μV) 65 mA Range: ±(0.05% of reading + 16.533 μA)
Resolution (14 bits)	10V Range: 310 μV 1V Range: 31 μV 65 mA Range: 2.0667 μA (15Ω, 0.04%, 20 ppm TC sense resistor)

Filtering	Software filter, 1 to 128 readings, software selectable in powers of two.
Common Mode Voltage (including measurement voltage)	10V Range: ± 10.5V 1V Range: ± 6.5V
Common Mode Rejection	dc: 77 dB @ 10°C-40°C 50/60 Hz: 60dB @ 10°C-40°C
Normal Mode Rejection	Internally Synchronized mode: 20 dB (50/60 Hz ±0.3 Hz) Externally Synchronized mode: 50 dB (45 Hz to 65 Hz) 45 dB (360 Hz to 520 Hz) Asynchronous Mode: 0 dB
Input Protection	50V rms without side effects Fuse-resistor protected to 240V rms (400V peak)
Automatic Self-Calibration	Performed approximately every 17 seconds. Requires 10 msec. May be disabled by user software.

CONNECTOR DESCRIPTION

The Analog Measurement Processor has three rear-panel connectors: two 26-pin analog input connectors and one external sync input connector. Pinouts for the analog input connectors are shown below.

INPUT CONNECTORS



INPUT CONNECTOR PIN ASSIGNMENTS

PIN NUMBER (CONNECTOR J1)	CHANNEL NO.	PIN NUMBER (CONNECTOR J2)	CHANNEL NO.
1	0	1	16
2	1	2	17
3	(Signal Ground)	3	(Signal Ground)
4	2	4	18
5	3	5	19
6	(Signal Ground)	6	(Signal Ground)
7	4	7	20
8	5	8	21
9	(Signal Ground)	9	(Signal Ground)
10	6	10	22
11	7	11	23
12	(Signal Ground)	12	(Signal Ground)
13	8	13	24
14	9	14	25
15	(Signal Ground)	15	(Signal Ground)
16	10	16	26
17	11	17	27
18	(Signal Ground)	18	(Signal Ground)
19	12	19	28
20	13	20	29
21	(Signal Ground)	21	(Signal Ground)
22	14	22	30
23	15	23	31
24	(Signal Ground)	24	(Signal Ground)
25	(Not Used)	25	(Not Used)
26	(Power Ground)	26	(Power Ground)

Notes: Signal grounds are connected together internally.
Channel numbers are shown for board address 0.

INSTALLATION

Complete installation and configuration instructions for the Analog Measurement Processor are provided in Section 3 of the 1752A Data Acquisition and Control manual.

CONTROLLING THE ANALOG MEASUREMENT PROCESSOR

The Analog Measurement Processor is controlled by BASIC subroutines that are supplied on the 1752A System Disk. You can call the subroutines from programs that you write in BASIC, or you can call the subroutines directly by using Interpreted BASIC in the Immediate mode.

The subroutines are supplied on the 1752A System Disk in two forms: in the AIOLIB library, for use with Interpreted BASIC, and in BASIC.LIB, for use with Compiled BASIC and Extended Basic.

It is a simple matter to use the subroutines in a program. Here are a few things to keep in mind:

- You must link to the library before using the subroutines. In Interpreted BASIC, the line reads, LINK "AIOLIB". For Compiled BASIC or Extended BASIC, consult the consult the programming manual for those languages.
- Most of the subroutines can be used simply by giving the program name followed by any parameters or arguments required.
- Parameters in which a value is returned from a subroutine must be initialized (assigned a value) before they can be used with the subroutines.

SUBROUTINE LIBRARY

The subroutines available on the 1752A System Disk for use with the Analog Measurement Processor are presented below, organized by function.

Board Functions

ADCAL(board%,calibration%)	Enable/disable automatic self-calibration.
ADSTAT(board%,sync%,calibration%)	Get calibration and sync mode status.
ADSYNC(board%,sync%)	Set sync mode.

Channel Functions

AIMODE(channel%,mode%)	Set an input channel to single-ended or differential mode.
AIENBL(channel%,enable%)	Set an input channel to be read or skipped.
AIRNGE(channel%,range%)	Select the range for an input channel.
AIFLTR(channel%,filter%)	Select the filter for an input channel.
AISET(channel%,mode%,enable%, range%,filter%)	Set all parameters for an input channel.
AISTAT(channel%,mode%,enable%, range%,filter%)	Get the status of an input channel.

Read Functions

AIREAD(channel%,reading)	Read an input channel in floating-point format.
AISCAN(first%,last%,set%,array())	Take sets of readings in floating-point format.
AISCNI(first%,last%,set%,array%())	Take sets of readings in integer format.

CONCLUSION

The previous pages have provided you with an overview of the capabilities, operating parameters, and specifications for the Analog Measurement Processor. For complete installation, configuration, and programming instructions refer to the 1752A Data Acquisition and Control manual.

1752A USER INFORMATION

OPTION 1752A-011
ANALOG OUTPUT

INTRODUCTION

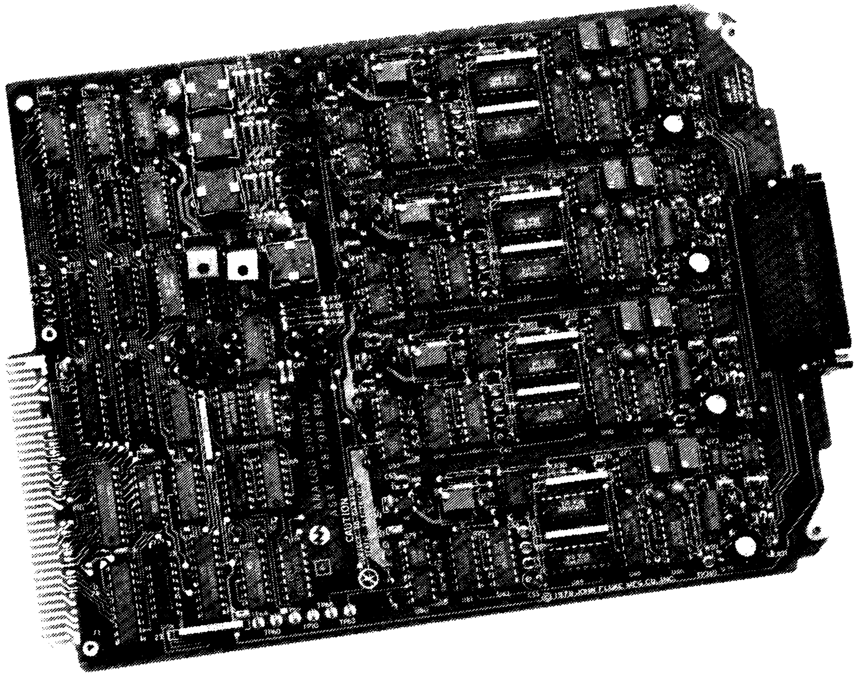
The Analog Output module allows the 1752A Data Acquisition System to send programmable voltages or currents to external control points, under the control of a BASIC program or BASIC commands. The Analog Output module gives the 1752A the ability to control many kinds of devices and processes.

The module has four fully-isolated, addressable output channels. By setting jumpers on the module, you can configure each output channel to output either voltage or current. When the jumpers are changed, the channel must be recalibrated. (The module can also be calibrated at a Fluke Technical Service Center.)

Output voltage from the analog output module is programmable over a −10.2375 to +10.2375V range with a resolution of 2.5 mV. Current from the module can be programmed over a 0 to 20.475 mA range, with a resolution of 5 μA. Each channel is isolated from digital common and the other channels on the board (maximum 30V common mode).

The Analog Output module mounts in slots 1, 4, or 5 of the 1752A chassis. Each Analog Output module is assigned a board address by the user, which determines the channel numbers of the four outputs.

1752A 011 Analog Output



SPECIFICATIONS

Output Channels:	Four per module
System Capacity:	Two modules (eight channels per system). 32 modules (128 channels) when using 1702A Extender Chassis
Configuration:	Each channel user-configurable for voltage or current. (Note: When changing configuration, the channel must be recalibrated.)
Update Rate:	1000 updates/sec
Slew Rate:	1.0V/μsec
Voltage Configuration	
Range:	± 10.2375V
Resolution:	2.5mV
Maximum Source Current:	±5 mA
Capacitive Load:	Will drive up to 10,000 pF without instability.
Output Protection:	All outputs are short-circuit protected.
Current Configuration	
Range:	20.475 mA
Resolution:	5μA
Maximum Load:	750 ohms
Maximum External Voltage:	±24V
Compliance Voltage:	15V
Accuracy:	0.1% of full scale (90 days, 18°C to 28°C)
Isolation:	Outputs are isolated from each other and from ground up to 30V dc.
Compatible Connectors:	Fluke Option 2400A-110 Screw Terminal Connector or 2400A-111 Solder Pin Connector.
Mounting:	Plugs into the 1752A chassis.
Weight:	1.4 kg (3 lbs)

CONNECTOR DESCRIPTION

The four channels on the Analog Output module are available at the male 50-pin D-connector supplied with the option. This connector is permanently connected to the module. Pin assignments for the output connector are shown in the following table.

PIN	SIGNAL	PIN	SIGNAL
50	+ Voltage Output A	22	- Current Output C
49	- Voltage Output A	40	Test Point 26C
32	+ Voltage Sense A	41	Test Point 26C
31	- Voltage Sense A	6	Test Point 27C
17	+ Current Output A	9	Not Used
16	- Current Output A	1	+ Voltage Output D
47	Test Point 26A	2	- Voltage Output D
48	Test Point 26A	19	+Voltage Sense D
33	Test Point 27A	18	- Voltage Sense D
3	Not Used	35	+ Current Output D
12	+ Voltage Output B	20	- Current Output D
13	- Voltage Output B	36	Test Point 26D
28	+ Voltage Sense B	37	Test Point 26D
29	- Voltage Sense B	34	Test Point 27D
10	+ Current Output B	14	Not Used
27	- Current Output B	15	Not Used
44	Test Point 26B	21	Not Used
45	Test Point 26B	25	Not Used
11	Test Point 27B	26	Not Used
4	Not Used	30	Not Used
7	+ Voltage Output C	38	Not Used
8	- Voltage Output C	39	Not Used
23	+ Voltage Sense C	42	Not Used
24	- Voltage Sense C	43	Not Used
5	+ Current Output C	46	Not Used

INSTALLATION

Complete installation and configuration instructions for the Analog Measurement Processor are provided in Section 3 of the 1752A Data Acquisition and Control manual.

CONTROLLING THE ANALOG OUTPUT MODULE

The Analog Output module is controlled by BASIC subroutines that are supplied on the 1752A System Disk. You can call the subroutines from programs that you write in BASIC, or you can call the subroutines directly by using Interpreted BASIC in the Immediate mode.

The subroutines are supplied on the 1752A System Disk in two forms: in the AIOLIB library, for use with Interpreted BASIC, and in BASIC.LIB, for use with Compiled BASIC and Extended BASIC.

It is a simple matter to use the subroutines in a program. Here are a few things to keep in mind:

- You must link to the library before using the subroutines. In Interpreted BASIC, the line reads, LINK "AIOLIB". For Compiled BASIC or Extended BASIC, consult the consult the programming manual for those languages.
- Most of the subroutines can be used simply by giving the program name followed by any parameters or arguments required.
- Parameters in which a value is returned from a subroutine must be initialized (assigned a value) before they can be used with the subroutines.

SUBROUTINE LIBRARY

The following subroutines are available on the 1752A System Disk for use with the Analog Output:

AOCRNT(channel%,amps)	Output current on a channel.
AOVLTG(channel%,volts)	Output voltage on a channel.

CONCLUSION

The previous pages have provided you with an overview of the capabilities and use of the Analog Output module. For complete installation, configuration, and programming information, refer to Section 4 of the 1752A Data Acquisition and Control manual.

1752A USER INFORMATION
OPTION 1752A-012
COUNTER/TOTALIZER

INTRODUCTION

The 1752A-012 Counter/Totalizer is a plug-in module that measures frequency and time, and totalizes with a gateable bidirectional totalizer. The different modes of operation are selected under software control.

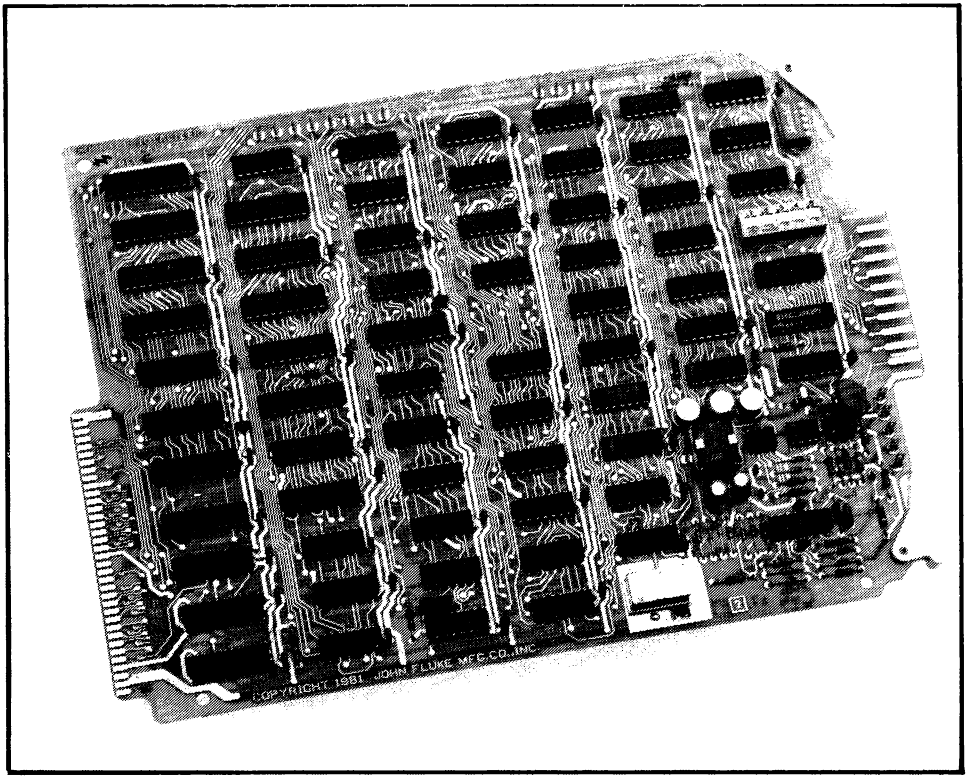
The Counter/Totalizer provides several inputs, including two frequency and time measurement inputs, two gating inputs, one trigger input, and four totalizer inputs. All of the measurement inputs except the analog input may include switch closures or logic levels. These inputs are not isolated from the 1752A or ground. The analog input for frequency and time measurement may be a periodic analog signal; this input is isolated from the 1752A and ground.

The Counter/Totalizer also provides an output, which indicates an overflow condition. This output is a logic level and is not isolated from the 1752A or ground.

Several features of the Counter/Totalizer provide for system synchronization. Such features include external triggering, single or continuous measurement cycling, and presettable bidirectional totalizing with logic-level gating.

The Counter/Totalizer mounts in option slots 4 or 5 in the 1752A chassis. The 12-contact, screw-terminal connector attached to the module provides connection for all nine inputs, one output, isolated common, and non-isolated common.

1752A-012 COUNTER/TOTALIZER



Frequency and Time Measurement

Either of two inputs may be selected for frequency and time measurement. The first is an isolated analog input that works with signals having frequencies up to 200 kHz. The second is a non-isolated, TTL-level input that works with frequencies up to 900 kHz. Readings can be triggered either by an external input or can be under software control.

Frequency measurements may be made with four selectable gate times. A time measurement may be made between any two events from 1.2 μs to 3.8 hours apart on separate Gate 1 and Gate 2 inputs. Time measurements may be made on non-isolated TTL level input from 1.2 μs to 6.7 seconds and on the isolated analog input from 2.5 μs to 100 ms. Control software allows the user full flexibility to program time measurements between different edge transitions.

Frequency and time measurements may be triggered by the user's software or by an external input.

Totalizer

The totalizer allows the 1752A to count events. Two different types of input signal pairs may be used: Count Up and Count Down, or Count and Up/Down. Totalizing can be enabled or disabled by two external gating lines or by software under program control. The total can be accessed at any time and can be preset to any value within its capacity. Totalizer inputs can be individually configured for contact bounce elimination, allowing connection to relay contacts.

SPECIFICATIONS

NUMBER OF PCB ASSEMBLIES	1 Counter/Totalizer Module (1 channel) per 1752A. Maximum 32 Counter/Totalizer Modules (32 channels) with 1702A.
NONISOLATED INPUTS	TTL Input, Gate 1, Gate 2, Trigger, Count Up, Count Down, Up / Down, Count, Common (Nonisolated).
ISOLATED INPUTS	Analog Input and Common (Isolated). These inputs are isolated in common from the 1752A and/or ground, up to 30V and 1.0 volt/microsecond maximum slew rate.
INPUT LOGIC	
Nonisolated Inputs	Pull-up resistors are installed to allow contact closures and logic levels as inputs.
Isolated Inputs (Analog Input)	Any periodic analog signal that satisfies signal level, frequency, duty cycle, and isolation requirements is allowed.
LOGIC LEVELS	All nonisolated inputs are low power Schottky inputs with static protection and a minimum of 400 mV of hysteresis. 0V to 0.5V for input low, 2.0V to 7V for input high.
OVERFLOW OUTPUT	
Output Isolation	The Overflow output is not isolated from the 1752A or ground.
Output Circuitry	Open - collector TTL output.
Maximum Sink Current	40 mA.
Output Voltage	
AT 40 mA SINK CURRENT	0.7V max
AT 16 mA SINK CURRENT	0.4V max
Maximum OFF-State Voltage	30V
TTL Compatibility	The output can drive up to six standard TTL loads or up to twelve low-power Schottky loads. An external pull-up resistor is not required.
FREQUENCY MEASUREMENTS	
Inputs	TTL Input, Analog Input, Trigger
Range	
NONISOLATED INPUT	0 to 900 kHz
ISOLATED INPUT	10 Hz to 200 kHz

Gate Times

1	3.277 ms
2	26.21 ms
3	209.7 ms
4	3.355 sec

Resolution

1	305.2 Hz
2	38.15 Hz
3	4.768 Hz
4	0.2980 Hz

Accuracy .... ± 1 count ± .005% of resolution (timebase accuracy)

TIME MEASUREMENTS

Inputs .... TTL Input, Analog Input, Gate 1, Gate 2 , Trigger

Range

NONISOLATED INPUT

Fast Rate .... 1.2 μs to 6.7s (all)

Slow Rate .... 819 μs to 3.82 hr (G1, G2 only)

ISOLATED INPUT .... 2.5μs to 50ms pulse width or 5μs to 100ms period

Frequency Counted

FAST RATE .... 2.5 MHz (all)

SLOW RATE .... 1.221 kHz (G1, G2 only)

Resolution

FAST RATE .... 400 ns (all)

SLOW RATE .... 819 μs (G1, G2 only)

Accuracy .... ± 1 count ±.005% of reading (timebase accuracy) ± trigger error of signal on isolated input

TOTALIZE

Count Inputs .... $\overline{\text{Count Up}}$, $\overline{\text{Count Down}}$, $\overline{\text{Count}}$, Up / $\overline{\text{Down}}$

Gate Inputs .... Gate 1, Gate 2

Range .... DC to 900 kHz

Capacity .... -8,388,608 to +8,388,607

ANALOG INPUT SENSITIVITY

Sinewave

10 Hz to 100 kHz .... 50 mV rms

100 kHz to 200 kHz .... 100 mV rms

Pulse .... 150 mV with minimum pulse width of 2.5 μs. 5% < duty cycle < 95%.

ANALOG INPUT IMPEDANCE

Nonlimiting .... 10 megohm, ≤50 pF

Limiting .... 1.2 megohm in parallel with 47 pF

TIMEBASE

Frequency .... 10.000 MHz

Accuracy .... ±50 ppm

NONISOLATED INPUT TIMING .... See timing diagram

Trigger Input

TRIGGERABLE MODES .... Frequency, Time

PULSE WIDTH (Tpw) .... 100 ns min.

Input Debouncing

INDIVIDUALLY DEBOUNCEABLE INPUTS (SWITCH SELECTABLE) ...... $\overline{\text{Count}}$, $\overline{\text{Count Up}}$, $\overline{\text{Count Down}}$, Gate 1, Gate 2

INPUT PULSE WIDTH TO CHANGE STATE (Tip) .... 15 ms min.

BOUNCE PULSE WIDTH (Tbp) . 10 ms max.

Measurement Input Timing (All Debouncers Off)

APPLICABLE INPUTS .... TTL Input, Gate 1, Gate 2, $\overline{\text{Count}}$, $\overline{\text{Count Up}}$, $\overline{\text{Count Down}}$, Trigger

INPUT PULSE WIDTH EXCEPT TRIGGER (Tpw) ...... 500 ns min.

TRIGGER SET-UP TIME IN TIME MODES (TTRsu) .... 1.0 μs minimum to guarantee that the period or time interval immediately following the trigger is the one measured.

GATE 1- TO -GATE 2 OR GATE 2- TO -GATE 1 SEPARATION (TGGsep) .... 500 ns minimum to guarantee that the positive edge of one input is recognized as occurring before the other.

GATE TO COUNT OR COUNT TO GATE SEPARATION (TGCsep) .... 1.0 μs minimum to guarantee that count pulses are totalized after the gate enables the totalizer and not before, and that count pulses are totalized before the gate disables the totalizer and not after.

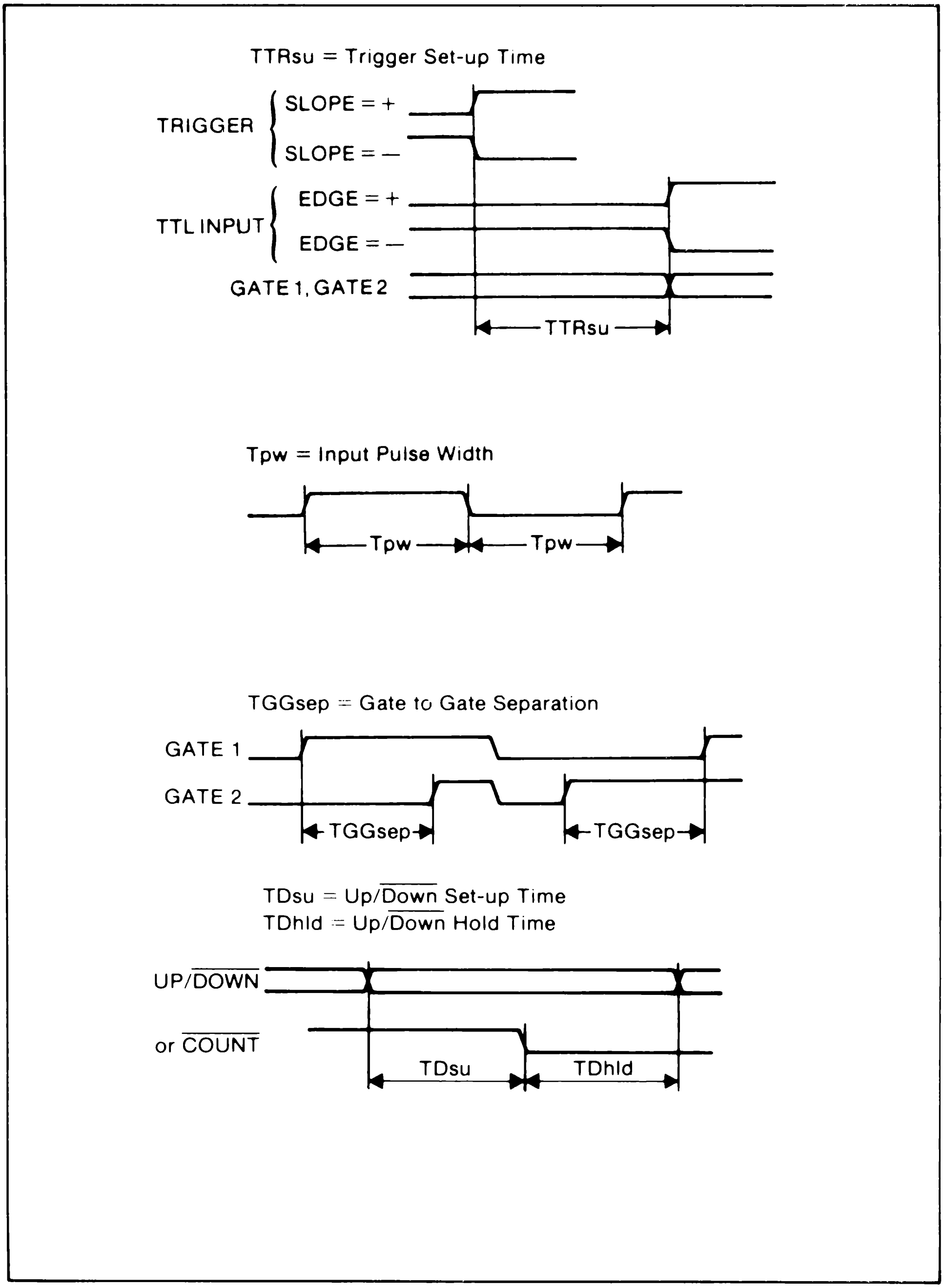
UP/DOWN INPUT STABLE SET-UP (TDsu) .... 0 ns minimum prior to Count.

UP/DOWN INPUT STABLE HOLD (TDhld) .... 1.0 μs minimum after Count.

POWER CONSUMPTION .... 12 watts typical per PCB assembly added to the system.

WEIGHT .... 0.4 kg (1.0 lb)

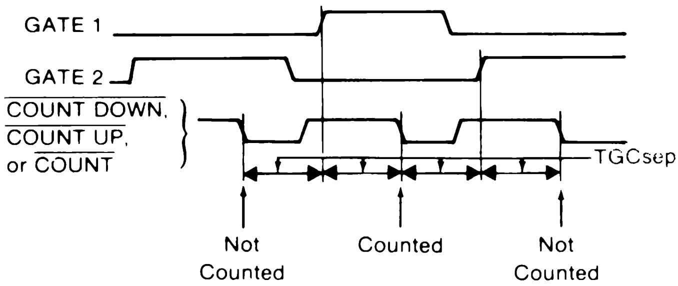
Non-Isolated Input Timing



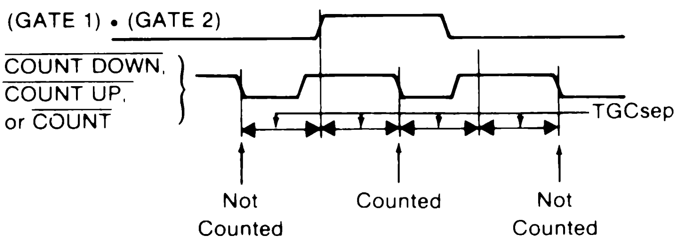
Non-Isolated Input Timing (cont)

TGCsep = Gate to Count/Count to Gate Separation

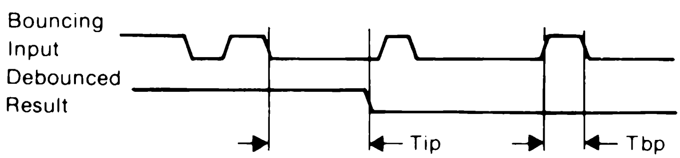
Case 1 : GATE = G1 TO G2



Case 2 : GATE = G1 AND G2



Tbp = Bounce Pulse Width
Tip = Change State Pulse Width
(shown for a high-to-low input transition, invert signals for corresponding low-to-high timing).



CONNECTOR DESCRIPTION

The Counter/Totalizer has a 12-terminal, screw-type connector for connecting to remote input sources. Connector pin assignments are shown below. For signal level and timing specifications, refer to the specifications presented earlier. All inputs except the analog input are TTL-level signals.

Connector Pin Assigtnments

TERMINAL	SIGNAL	COMMON TERMINAL
1	Common (Isolated)	N.A.
2	Analog Input	1
3	$\overline{\text{Overflow}}$	12
4	Gate 1	12
5	Gate 2	12
6	Trigger	12
7	$\overline{\text{Count Up}}$	12
8	$\overline{\text{Count Down}}$	12
9	Up / $\overline{\text{Down}}$	12
10	$\overline{\text{Count}}$	12
11	TTL Input	12
12	Common (Non-isolated)	N.A.

INSTALLATION

Complete installation instructions for the Counter/Totalizer are in Section 5 of the 1752A Data Acquisition and Control manual.

CONTROLLING THE COUNTER/TOTALIZER

The Counter/Totalizer is controlled by BASIC subroutines that are supplied on the 1752A System Disk. You can call the subroutines from programs that you write in BASIC, or you can call the subroutines directly by using Interpreted BASIC in the Immediate mode.

The subroutines are supplied on the 1752A System Disk in two forms: in the DIOLIB library, for use with Interpreted BASIC, and in BASIC.LIB, for use with Compiled BASIC and Extended Basic.

It is a simple matter to use the subroutines in a program. Here are a few things to keep in mind:

- You must link to the library before using the subroutines. In Interpreted BASIC, the line reads, LINK "DIOLIB". For Compiled BASIC or Extended BASIC, consult the programming manual for those languages.
- Most of the subroutines can be used simply by giving the program name followed by any parameters or arguments required.

SUBROUTINE LIBRARY

The subroutines available on the 1752A System Disk for use with the Counter/Totalizer are presented below, organized by function.

Setup Functions

CTFREQ(channel%,source%,gatetime%)	Set up a channel for frequency measurement.
CTTIME(channel%,source%,slope%)	Set up a channel for time measurement.
CTTOTL(channel%,gate%,initial%)	Set up a channel for totalizing or counting events.
CTTRCF(channel%,type%)	Select the trigger type for a channel.
CTMODE(channel%,mode%)	Put a channel in continuous or single measurement mode.

Measurement Functions

CTRDY(channel%,ready%)	See if a reading is available at a channel.
CTTRGR(channel%,trigger%)	Trigger a measurement or enable totalizing.
CTREAD(channel%,reading)	Read a single channel.
CTSCAN(first,last%,array())	Read a group of channels.

CONCLUSION

The previous pages have provided you with an overview of the capabilities, operating parameters, and specifications for the Counter/Totalizer module. For complete installation and programming instructions, refer to the 1752A Data Acquisition and Control manual.

1752A USER INFORMATION

1760A and 1761A DISK DRIVE SYSTEMS

WARNING

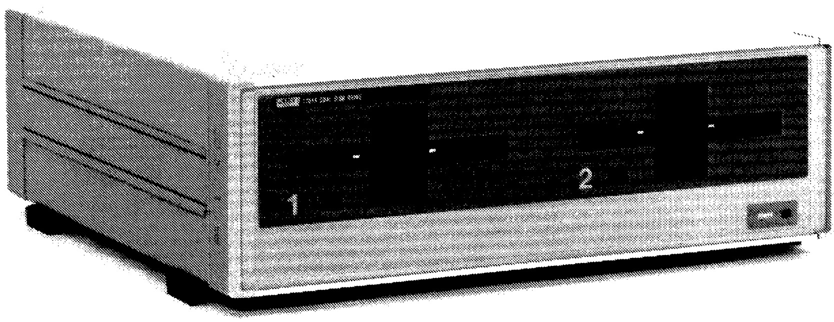
This equipment generates and uses radio frequency energy and if not installed and used in accordance with the instruction manual, may cause interference to radio and television reception. It has been tested and found to comply with the limits for a Class B computing device pursuant to Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference in a residential installation. If this equipment does cause interference, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by relocating the equipment with respect to the receiver or plugging the computer into a different outlet so that the computer and receiver are on separate branch circuits.

9-81

INTRODUCTION

The Fluke 1760A and 1761A Disk Drive Systems provide additional floppy disk mass storage for the 1752A. They connect to the 1752A's IEEE-488 connector and are recognized and addressed as additional logical devices MF1: - MF4:.

In addition to the internally mounted floppy disk drive (MF0:), up to four other drives can be connected to the 1752A. The Fluke model 1760A is a single 5-1/4 inch unit, and the 1761A has two drives. When the drivers are installed, they act similar to the internal drive; operation over the IEEE-488 bus is transparent to the user.



PRE-INSTALLATION CHECKOUT

The Disk Drive System is carefully inspected at the factory before shipment. Remove it from the shipping carton and inspect it for any signs of physical damage that might have occurred during shipment. If you find anything wrong, notify the nearest Fluke Service Center immediately and file a claim with the carrier.

INSTALLATION

These procedures describe how to install the Disk Drive System. There are two main steps:

- Setting the proper line voltage.
- Setting the IEEE-488 address.

Setting the Line Voltage

The Disk Drive System is shipped configured to the line voltage specified on the order form. If no voltage is specified, the factory ships the drive to operate on 120V ac, 47 to 63 Hz. If that is the voltage you will be using, skip this procedure, and go on to the next step, setting the IEEE-488 address. This procedure is only required if the voltage setting is different than that required, but it just takes a moment to check, so we recommend that you check the voltage as part of the incoming inspection.

1. Place the drive on a suitable work surface with the rear facing you, and with the line cord disconnected.

2. Slide the clear plastic cover (A) to the side, so that it covers the line plug connector.

3. Look at the wafer (B) at the top of the AC line connector. It shows the voltage for which the drive is configured. If the number shown differs from the voltage at the installation site, remove the wafer from its slot, and re-install it so that the correct number shows. This may be either 100, 120, 200, or 240.

4. Slide the cover back over the wafer, and double check that the correct voltage is still visible.

NOTE
Changing the wafer's position changes the voltage configuration. No frequency conversion is needed.

5. Check the fuse (C) against this table, and replace it for the correct value if needed.

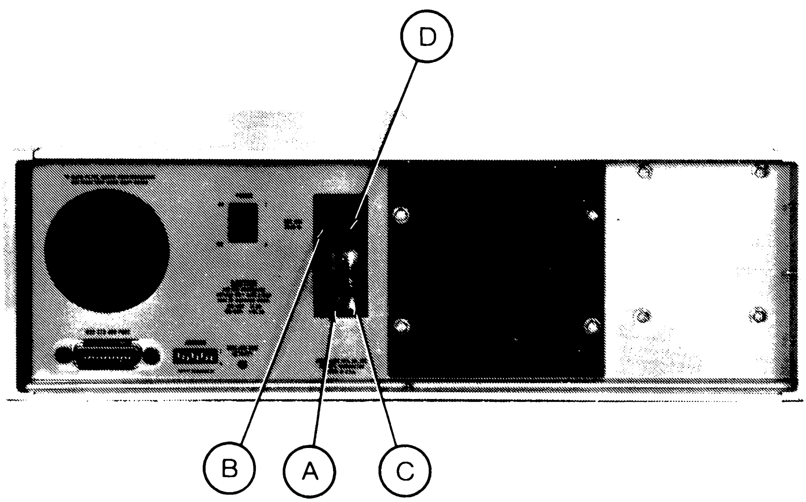
Voltage Setting	Fuse Rating
100 to 120	1 ampere, 250 volt, slow blow
200 to 240	1/2 ampere, 250 volt, slow blow

6. Check the power cord supplied with the Disk Drive System to ensure that it matches the receptacle. If it does not match, contact your sales representative to obtain the correct cord.

WARNING

If it is necessary to replace the power cord, the replacement must have the same polarity as the original. If it does not, a shock hazard exists, and the Disk Drive System may be damaged.

7. Plug the AC line plug into the connector (D) at the rear of the Disk Drive System. Do not apply power until the complete installation procedures have been done.



Selecting the IEEE-488 Address

After the power requirements have been taken care of, the next step is to connect the Disk Drive System to the Controller's IEEE-488 port.

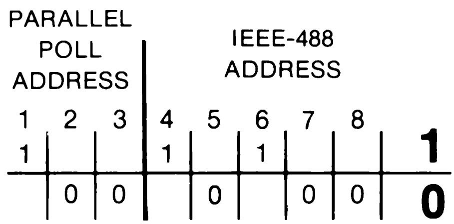
After the power requirements have been taken care of, the next step is to connect the Disk Drive System to the 1752A's IEEE-488 port.

1. Disconnect power from the 1752A and the Disk Drive System.

2. Remove the Single Board Computer module from the 1752A (slot 7). Check configuration switch S1 to be sure that position 5 and 6 are set to the OFF position. Doing so makes the 1752A the System Controller rather than an Idle Controller. Only a System Controller can manipulate the bus control lines to initialize the disk drive.

3. Replace the Single Board Computer Module into slot 7, and reattach the 1752A's rear cover.

4. Locate the switch labeled "Device Address" on the back of the Disk Drive System. Set the switch to IEEE-488 Address 20, Parallel Poll Address 4.



NOTE

If the 1752A is installed in a system, make sure that no other connected equipment is set to Address 20, because bus contention problems might occur, in which case neither the drive nor the other device will operate properly. Also make sure that the 1752A itself is not set to address 20.

If a second Disk Drive System is connected to the 1752A, it must be set to Device Address 21, Parallel Poll Address 5. (Switches ON: 1, 3, 4, 6, and 8.)

5. Attach the Disk Drive System IEEE-488 connector to the IEEE-488 connector on the 1752A's Single Board Computer module. If you need to order the cable, consult the table below:

Fluke P/N	**Length**
Y8021	1 meter
Y8022	2 meters
Y8023	4 meters

TESTING THE DISK DRIVE SYSTEM

At this point, all the steps have been completed to prepare for using the new Disk Drive System. To begin using it, remember that it provides new logical devices: MF1:, and so on. Also be sure to format floppy disks prior to use, as they are not formatted at the factory prior to shipment.

WARNING

Whenever the 1752A is accessing the floppy disk drive, do not press the RESTART or ABORT buttons or enter 〈CTRL〉/P. Doing so may damage the directory or data files on the disk. The disk is being accessed any time the red LED indicator is lit.

1. Connect the Disk Drive System to the 1752A's IEEE-488 Port 0 connector, and power up both units.

2. Follow the instructions in Appendix G of the System Guide to run the diagnostic MFXTST. This test checks the disk drive speed and media detection logic, then seeks, formats, writes, and reads data back from a disk installed in each drive.

USING THE DISK DRIVE SYSTEM

There is a software device driver for the 1760A and 1761A that can be linked into the 1752A operating system (FDOS). The device driver allows the system software and applications programs to treat the Disk Drive System as another logical device in the system. The device names are MF1: and MF2: for a Disk Drive System at IEEE-488 Address 20, and MF3: and MF4: for a Disk Drive System at Address 21.

The operating system (FDOS2.SYS) supplied on the 1752A System Disk does not contain the device driver for the 1760A and 1761A. You must run the SYSGEN program to create a new operating system that includes this driver before you can use the Disk Drive System. Follow the instructions in Section 3 of the 1752A System Guide to reconfigure the operating system.

1752A USER INFORMATION

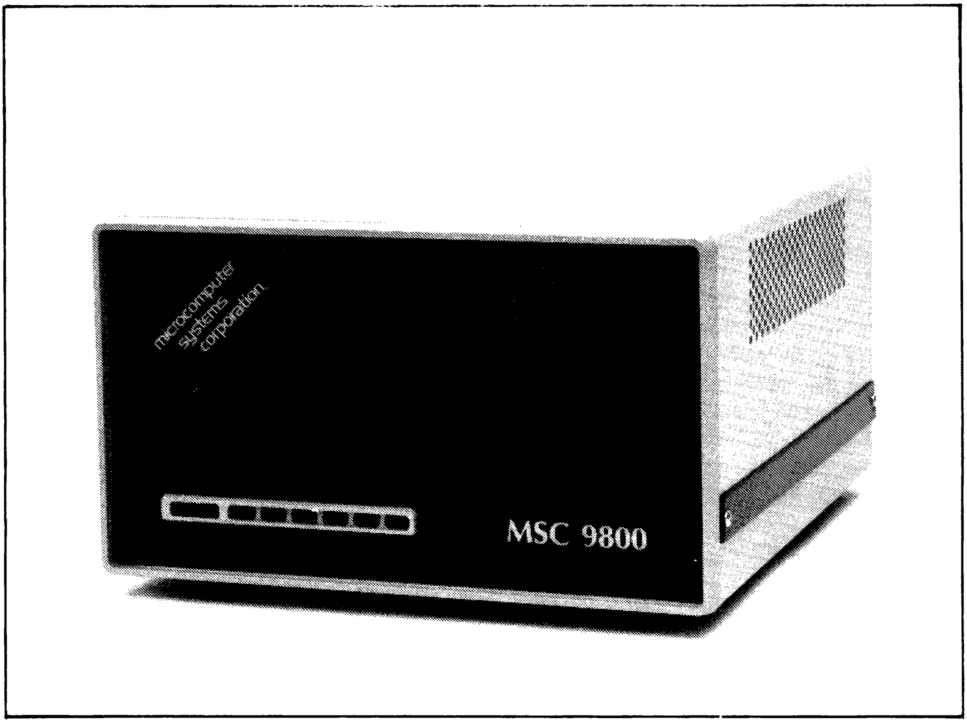
1765A/AB WINCHESTER DISK DRIVE

WARNING

This equipment generates, uses, and can radiate radio frequency energy and if not installed and used in accordance with the instruction manual, may cause interference to radio communications. It has been tested and found to comply with the limits for a Class A computing device pursuant to Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference when operated in a commercial environment. 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 necessary to correct the interference.

INTRODUCTION

The Fluke 1765A/AB Winchester Disk Storage System provides 10 megabytes of mass storage for the 1752A. It connects to the 1752A's IEEE-488 connector, and is recognized and addressed as four logical devices: WD0:, WD1:, WD2:, and WD3:.

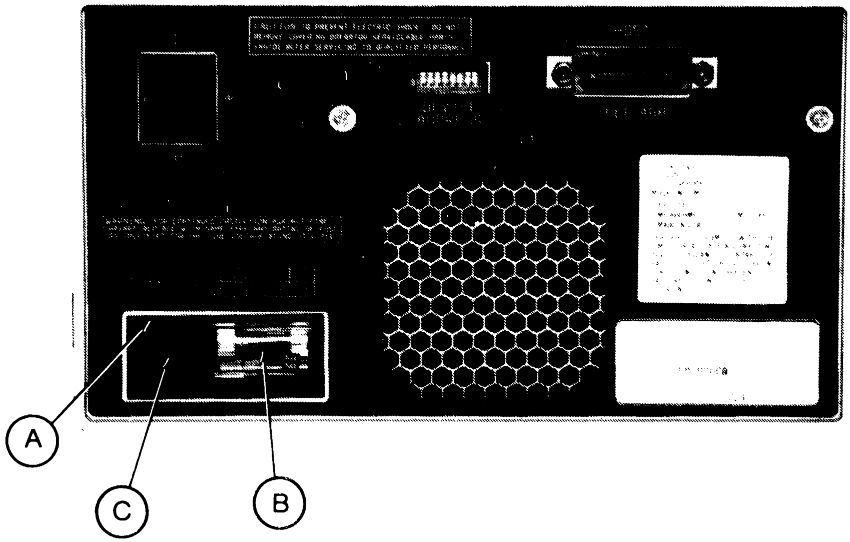


PRE-INSTALLATION CHECKOUT

The disk drive is carefully inspected at the factory before shipment. Remove it from the shipping carton and carefully inspect it for any signs of physical damage that might have occurred during shipment. If you find any damage, notify the nearest Fluke Service Center immediately and file a claim with the carrier.

These procedures describe how to install the disk drive. There are two main steps:

- Setting the proper line voltage.
- Setting the IEEE-488 address.


A
C
B

Setting the Line Voltage

The 1765A/AB is shipped configured to the line voltage specified on the order form. If no voltage is specified, the factory ships the drive to operate on 120V ac, 47 to 63 Hz. If that is the voltage you will be using, skip this procedure, and go on to the next step, setting the IEEE-488 address. This procedure is only required if the voltage setting is different than that required, but it just takes a moment to check, so we recommend that you check the voltage as part of the incoming inspection.

1. Place the drive on a suitable work surface with the rear facing you, and with the line cord disconnected.

2. Look at the wafer (A) at the top of the AC line connector. It shows the voltage for which the drive is configured. If the number shown differs from the voltage at the installation site, remove the wafer from its slot, and re-install it so that the correct number shows. This may be either 100, 120, 200, or 240.

   NOTE

   Changing the wafer's position changes the voltage configuration. No frequency conversion is needed.

3. Check the fuse (B) against this table, and replace it for the correct value if needed.

Voltage Setting	Fuse Rating
100 to 120	2 ampere, 250 volt, slow blow
200 to 240	1 ampere, 250 volt, slow blow

4. Check the power cord supplied with the disk drive to ensure that it matches the receptacle. If it does not match, contact your sales representative to obtain the correct cord.

   WARNING

   If it is necessary to replace the power cord, the replacement must have the same polarity as the original. If it does not, a shock hazard exists, and the disk drive may be damaged.

5. Plug the AC line plug into the connector (C) at the rear of the disk drive. Do not apply power until the complete installation procedures have been done.

Selecting the IEEE-488 Address

After the power requirements have been taken care of, the next step is to connect the 1765A/AB to the 1752A's IEEE-488 port.

1. Disconnect power from the 1752A and the disk drive.

2. Remove the Single Board Computer module from the 1752A (slot 7). Check configuration switch S1 to be sure that positions 5 and 6 are set to the OFF position. Doing so makes the 1752A the System Controller rather than a Controller in Charge. Only a System Controller can manipulate the bus control lines to initialize the disk drive.

3. Replace the Single Board Computer Module into slot 7, and reattach the 1752A's rear cover.

4. Locate the switch labeled "Device Address" on the back of the 1765A/AB. Set it to IEEE-488 Address 12. The orientation of the switch differs between revisions of the Disk Drive System, but the back panel makes clear the required direction to place a switch ON switch settings. When a switch is set to 1, it is ON (closed). For Address 12, switches 5 and 6 should be ON, and all others OFF as shown:

1	2	3	4	5	6	7	8	
				1	1			**1**
0	0	0	0			0	0	**0**

NOTE

If the 1752A is installed in an IEEE-488 system, make sure that no other connected equipment is set to Address 12, because bus contention problems may occur, in which case neither the drive nor the other device will operate properly. Also make sure that the 1752A itself is not set to address 12.

5. Attach the disk drive IEEE-488 connector to the IEEE-488 connector on the 1752A's Single Board Computer module. If you need to order a cable, consult the table below:

Fluke P/N	Length
Y8021	1 meter
Y8022	2 meters
Y8023	4 meters

Testing the Hard Disk Drive

At this point, all the steps have been completed to prepare for using the new hard disk. To begin using it, remember that it provides four new logical devices: WD0: - WD3:. There is no need to format the disk prior to use, as it has been formatted at the factory prior to shipment.

WARNING

Whenever the 1752A is accessing the Winchester disk drive, do not press the RESTART or ABORT buttons or enter ⟨CTRL⟩ /P. Doing so may damage the directory or data files on the disk. The disk is being accessed any time the red LED indicator is lit.

1. Connect the drive to the 1752A's IEEE-488 Port 0 connector, and power up both units.

2. Follow the instructions in Appendix G of the System Guide to run the diagnostic WDXTST. This test program performs two functions:

   Self Test — This program sends a self-test command to the disk controller inside the 1765A/AB. When it receives this command, the 1765A/AB performs an internal disk controller test and communications test.

   Verify — This program checks each sector on the disk for errors, and reports any errors to the display.

3. When the program has been selected, four options will be presented on the screen:

   Option 1 - Self Test

   Option 2 - Verify

   Option 3 - Self Test and Verify

   Option 4 - Exit (Exits to FDOS.)

4. Select the option desired. For new installations, Option 3 is probably the most useful. If the message "correctable ECC error" is displayed during the Verify operation, press ⟨RETURN⟩ to continue the test. This indicates that a software correctable error occured. Any other errors indicate a faulty unit, and should be reported to the local Fluke Service Center.

Using the Winchester Disk Drive

There is a software device driver for the 1765A/AB that can be linked into the 1752A operating system (FDOS). The device driver allows the system software and applications programs to treat the Winchester Disk Drive as four additional logical devices in the system. The device names are WD1:, WD2:, WD3: and WD4:.

The operating system (FDOS2.SYS) supplied on the 1752A System Disk does not contain the device driver for the 1765A/AB. You must run the SYSGEN program to create a new operating system that includes this driver before you can use the Winchester Disk Drive. Follow the instructions in Section 3 of the 1752A System Guide to reconfigure the operating system.

1752A USER INFORMATION

Option 1752A-013

1702A Extender Chassis and Mainframe Interface

THE 1702A I/O EXTENDER CHASSIS

The Fluke Model 1702A I/O Extender Chassis provides additional installation space for control and measurement options. Up to eight Extender Chassis may be used in a system. Each 1702A contains a switch selectable 120V ac/ 240V ac power supply and 11 slots for I/O option installation. The Parallel Interface (-002 option), Analog Measurement Processor (-010 option), Analog Output (-011 option), and Counter Totalizer (-012 option) may reside in the 1702A chassis. However, the total current required for all installed option assemblies must not exceed 11 Amps. Refer to the following table for the current draw for each option and the maximum number of boards allowed per system. For more information on the Extender Chassis, see the manual supplied with the 1702A.

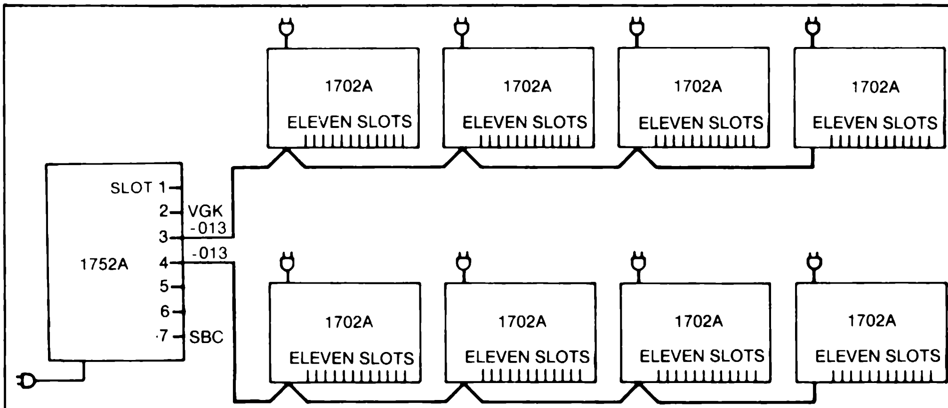
Option	Current Requirements	Maximum No. of Boards	Maximum No. of Channels
-002 Parallel Interface	.73A	8	16
-010 Analog Measurement Processor	.6A	4	128
-011 Analog Output			
Volts	1.3A	32	128
Current	1.6A		
-012 Counter/Totalizer	1A	32	32

THE 1752A-013 MAINFRAME INTERFACE ASSEMBLY

The 1752A-013 Mainframe Interface Assembly forms a digital communication link between the 1752A Instrument Controller and the I/O options installed in the 1702A Extender Chassis. The Mainframe Interface Assembly is installed in slots 1, 3, 4 or 5 of the 1752A chassis and is connected by a shielded ribbon cable to the Extender Interface Assembly located in the 1702A.

The Mainframe Interface Assembly is a plug-in printed circuit board (pcb) assembly that connects to the motherboard in the 1752A. Two 1752A-013 modules may be installed per 1752A system. The modules are connected to the extenders by a standard 2-meter cable (shipped with the 1702A) or the Option 2402A-502 9-meter cable.

The daisy-chain in the following figure shows the maximum extension possibilities for the 1752A, utilizing the 1752A-013 and 1702A options.



MAXIMUM 2 -013 OPTIONS PER SYSTEM.

MAXIMUM 4 1702A's PER DAISY CHAIN.

MAXIMUM 8 1702A's PER SYSTEM.

MAXIMUM CABLE LENGTH FROM MOST DISTANT 1702A TO 1752A 15 METERS (50 FT.).

See the 1702A I/O Extender Instruction Manual for more information on the 1752A-013 Mainframe Interface option.

THE 2402A-502 I/O EXTENDER CABLE

The 2402A-502 Extender Cable is an extended length, optional 9-meter cable that provides a connection between the 1752A and 1702A. It carries data, addresses, status and commands between the 1752A Single Board Computer and the I/O options installed in the 1702A chassis. A maximum of 15 meters of cable is allowed from the 1752A to the last extender in a leg.

INSTALLING THE MAINFRAME INTERFACE OPTION

Inspect the shipping carton and contents for damage. Notify the shipper immediately if there appears to have been damage in shipping. If everything seems to be in order, proceed to the next step, setting the board address switch.

Setting the Board Address Switch

To install one 1752A-013 option in the 1752A, it is not necessary to set the board address switch. If two -013 options will be installed, slide switch JPR1 must be set on the pcb.

On one pcb, set JPR1 to board 0 by moving the small slide switch forward. On the other pcb, set JPR1 to board 1 by moving the slide switch backward.

Installation

Once the 1752A-013 option's board address has been set, install the 1752A-013 option by following the directions in the Options section under the heading, Installing Hardware Options. The option may be installed in slot 1, 3, 4 or 5 of the 1752A.

CONNECTING THE 1752A TO THE 1702A

To connect the 1752A to the 1702A, perform the following steps:

1. Locate the Extender Interface Assembly in the back of the 1702A chassis. The Extender Interface Assembly is located on the rear panel to the left of the slots for the I/O options.

2. Connect the 2-meter cable (supplied with the 1702A) or the 9-meter cable (Option 2402A-502) to the top connector of the Extender Interface Assembly. Connect the other end of the cable to the Mainframe Interface located in the 1752A chassis. Install the Termination Assembly supplied with the Mainframe Interface into the lower connector.

3. To daisy-chain up to four 1702A chassis, do not install the Termination Assembly. Connect the cable from the next chassis in the chain to the lower connector on the Extender Interface Assembly in the first chassis.

4. Connect the other end of the cable to the top connector of the second chassis in the chain.

5. Repeat steps 3 and 4 for each additional chassis in the chain. If more than one daisy chain is to be installed, use the connector on a second Mainframe Interface option to start the other chain. No more than 15 meters of cable are allowed from the most distant 1702A to the 1752A.

6. Install the Termination Assembly in the lower connector of the last Extender Interface.

7. For option installation information, see the 1702A I/O Interface manual.

Appendices

CONTENTS

A	Specifications	A-1
B	RS-232 Reference	B-1
C	IEEE-488 Reference	C-1
D	Glossary	D-1
E	Custom Character Sets	E-1
F	Primary Character Set	F-1
G	System Diagnostic Software	G-1
H	Fluke Sales and Service Centers	H-1

Appendix A
Specifications

CRT DISPLAY

Scanning Method	Non-interlaced raster scan.
Refresh Rate	50 or 60 Hz, selectable.
Character Memory	1280 bytes of dedicated display memory. 16 lines of 80 characters.
CRT Screen	High-contrast green phosphor, low profile, rectangular. 8.6 cm x 20.3 cm (3.4 in x 8.0 in).
Character Capacity	16 x 80 cells, or 8 x 40 cells.
Standard Character Set	96 Standard ASCII characters, graphics characters, match, and other useful symbols.
Character Cell	7 x 9 dots in an 8 x 14 dot matrix.
Character Enhancements	Reverse video, blinking, underlining, and highlighting.
Cursor	Blinking, underline, block, or suppressed.
Raster Size	7.6 cm x 19 cm (3.0 in x 7.5 in).
Graphics Screen Capacity	640 x 224 pixels
Graphics Memory Capacity	64K bytes (2048 x 256 pixels). Independent of main memory.
Display Alignment	Character with respect to touch-sensitive grid at 25°C after 30 minute warmup: ±1.5 character horizontal ±0.5 character vertical Change in character positions over the operating temperature range: 1 character horizontal 0.5 character vertical

DISK DRIVE

Disk Size	Industry standard 5¼-inch
R/W Heads	2 per disk drive
Tracks	80 (40 per side)
Track Density	48 tracks per inch
Rotational Speed	300 RPM
Disk Error Rates	
RECOVERABLE	1 per 10^9 bits read
NON-RECOVERABLE	1 per 10^{12} bits read

Recording Format	
BYTES PER SECTOR	512
SECTORS PER TRACK	10
ENCODING METHOD	MFM (double density)
Total Formatted Storage	409,600 bytes per disk

ENVIRONMENTAL

Operating	
WITH DISK MEDIA	10°C to 40°C (50°F to 104°F) 20% to 80% RH (non-condensing)
WITHOUT DISK MEDIA	10°C to 40°C (50°F to 104°F) 5% to 90% RH (non-condensing)
Storage	
WITH DISK MEDIA	10°C to 52°C (50°F to 126°F) 8% to 80% humidity (non-condensing)
WITHOUT DISK MEDIA	-40°C to 60°C (-40°F to 140°F 5% to 90% humidity (non-condensing)

GENERAL

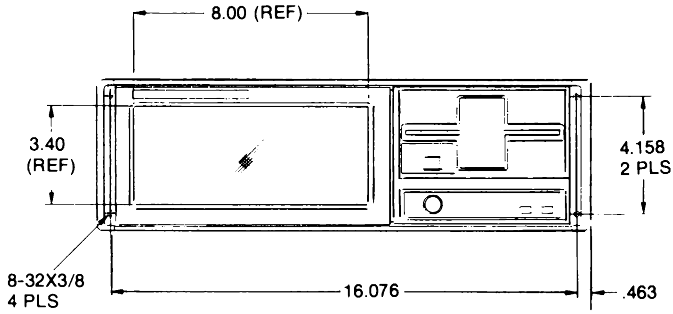
Dimensions	13 cm H x 43 cm W x 55 cm L (plus feet) (5.25 in H x 17.0 in W x 21.5 in L). See Outline Dimensions.
Weight	
SYSTEM UNIT	15.5 kg (34 lbs).
KEYBOARD	1.4 kg (3 lbs).
Power	175W max.
VOLTAGE	90-132 ac, 47-63 Hz (use 3A AGC fuse) 180-264 ac, 47-63 Hz (use 2A AGC fuse)
POWER DISSIPATION	175W max.

NOTE

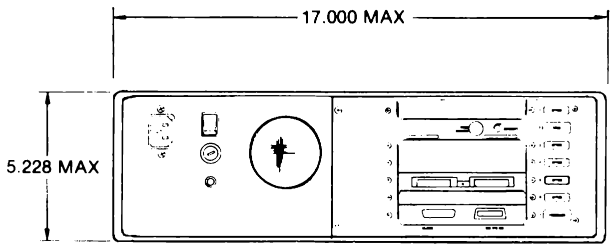
Specifications for the 17XXA-002 Parallel Interface, the 1752A-010 Analog Measurement Processor, the 1752A-010 Analog Output, and the 1752A-012 Counter/Totalizer are located in this manual in Section 9.

1 ⌖ THESE HOLES USED FOR MOUNTING OF FEET. CAN BE USED FOR NON STRUCTUAL INSTRUMENT MOUNTING.

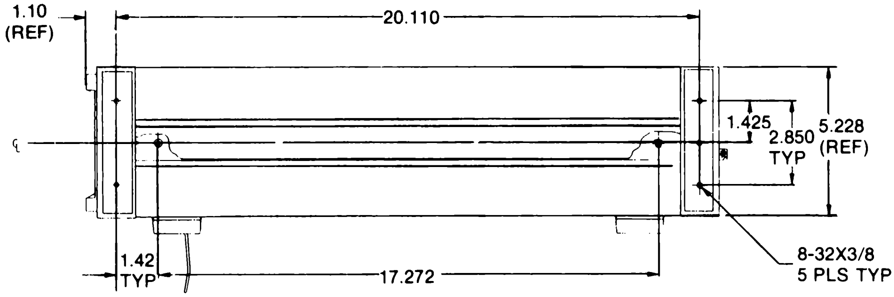
2 ALLOW 4" MINIMUM AT REAR AND CRT SIDE OF INSTRUMENT FOR AIRFLOW.



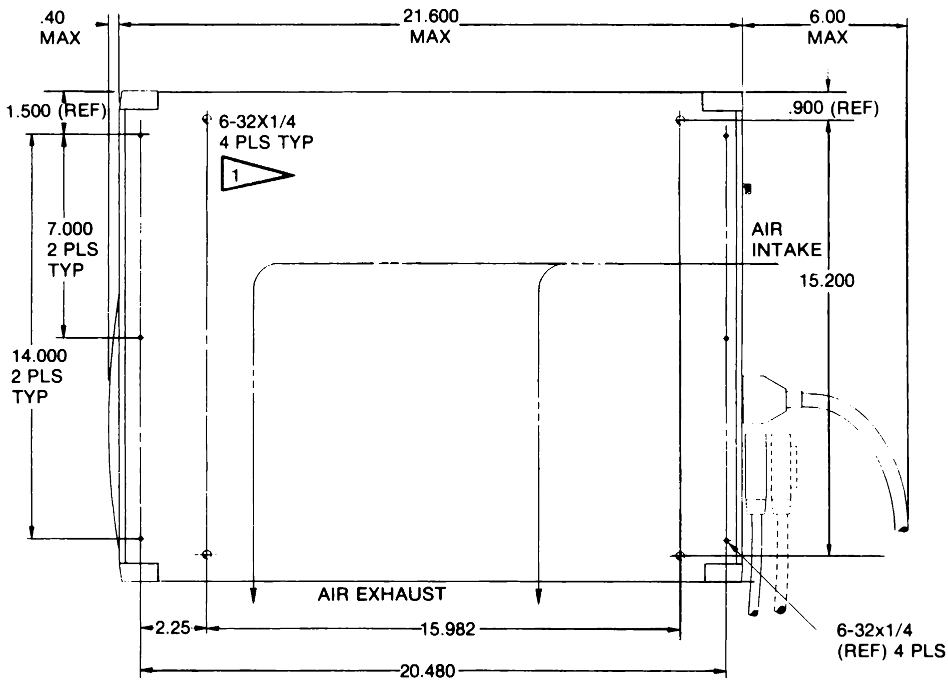
FRONT



BACK

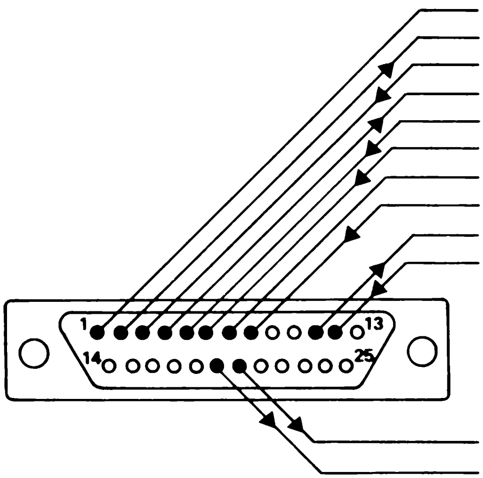

1.10
(REF)
20.110
1.425
2.850
TYP
5.228
(REF)
1.42
TYP
17.272
8-32X3/8
5 PLS TYP

SIDE


.40
MAX
21.600
MAX
6.00
MAX
1.500 (REF)
.900 (REF)
6-32X1/4
4 PLS TYP
1
7.000
2 PLS
TYP
AIR
INTAKE
15.200
14.000
2 PLS
TYP
AIR EXHAUST
2.25
15.982
20.480
6-32x1/4
(REF) 4 PLS

BOTTOM

Appendix B
RS-232 Reference



PIN	NAME	FUNCTION
1	GND	Chassis Ground
2	XMT	Transmitted Data
3	REC	Received Data
4	RTS	Request to Send
5	CTS	Clear to Send
6	DSR	Data Set Ready
7	RET	Signal (Logic) Ground
8	CD	Carrier Detect
11	UND	Undefined
12	SRLSD	Secondary Receive Line Signal Detect
20	DTR	Data Terminal Ready
19	SRTS	Secondary Request to Send

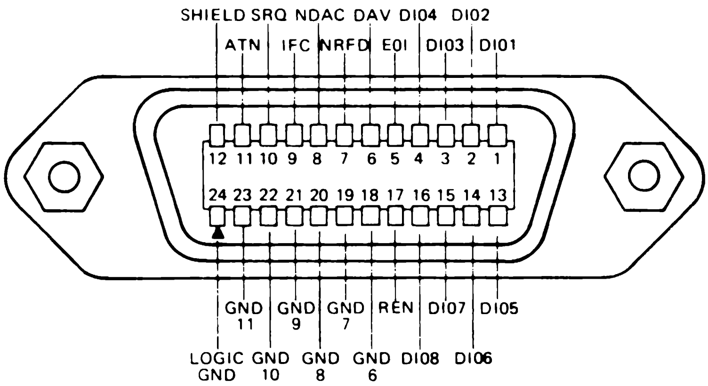
——▶—— Indicates signal direction

○ Indicates unused pins

<u>RS-232 CONNECTOR REAR PANEL VIEW</u>

Appendix C
IEEE-488 Reference

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 1752A when peripheral devices are being assigned as listeners and talkers. The 1752A 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 1752A Controller by setting SRQ active low.
	9 21	IFC IFC Return	Interface clear. Set by the 1752A to place all instruments on the bus in a predetermined reset state. The 1752A 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.

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.

ASCII and IEEE-488 Mnemonic Abbreviations

ACK	Acknowledge	MSA	My Secondary Address
ASCII	American Standard Code for Information Interchange	MTA	My Talk Address
		NAK	Negative Acknowledge
ATN	Attention	NDAC	Not Data Accepted
BEL	Bell	NRFD	Not Ready For Data
BS	Backspace	NUL	Null
CAN	Cancel	OSA	Other Secondary Address
CR	Carriage Return	OTA	Other Talk Address
DCL	Device Clear	PCG	Primary Command Group
DCn	Device Control 1, 2, 3, or 4	PPC	Parallel Poll Configure
DEL	Delete	PPD	Parallel Poll Disable
DIOn	Data Input/Output 1 through 8	PPE	Parallel Poll Enable
DLE	Data Link Escape	PPRn	Parallel Poll Response 1 through 8
ENQ	Enquiry	PPU	Parallel Poll Unconfigure
EOF	End of File	REN	Remote Enable
EOI	End or Identify	RS	Record Separator
EOT	End of Transmission	SDC	Selected Device Clear
ESC	Escape	SI	Shift In
ETB	End of Transmission Block	SO	Shift Out
ETX	End of Text	SOH	Start of Heading
FF	Form Feed	SP	Space
GET	Group Execute Trigger	SPD	Serial Poll Disable
GND	Ground	SPE	Serial Poll Enable
GTL	Go To Local	SRQ	Service Request
HT	Horizontal Tab	STB	Status Byte
IEEE	Institute of Electrical and Electronic Engineers	STX	Start of Text
		TCT	Take Control
IFC	Interface Clear	UNL	Unlisten
LF	Line Feed	UNT	Untalk
LLO	Local Lockout	US	Unit Separator
MLA	My Listen Address		

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	**0111 1000**		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			

Appendix D
Glossary

ABORT

Front panel push switch that causes the 1752A to terminate the current program and return to the shell without clearing memory or performing the power-up self-test. When pressed simultaneously with RESTART, causes the system to perform a cold start.

address

A coded number representing the location of an item. Examples include bus address and program address.

Address command

A bus command from a controller commanding an instrument at a designated address to talk or listen.

Addressed command

A bus command from a controller intended for all instruments that have been addressed as talker or listener.

alias

A shortened or more familiar form of a command. In the 1752A, all aliases must be recorded on a file named ALIAS.SYS.

application program

A user written program designed to perform specified functions in a working environment.

array

A collection of data items, organized as a row x column matrix.

array element identifier

The subscript of an array variable that identifies the row and column of the desired array element. In the expression: A$(3,5), (3,5) is the array element identifier, referring to row 3, column 5.

ASCII

Acronymn for American Standard Code for Information Interchange. ASCII is a standardized code set of 128 characters, including full alphabetic (upper and lowercase), numerics, and a set of control characters.

asynchronous data

Information transmitted at random times, normally one character at a time, and at predefined, self-clocking baud rates. See synchronous data.

BASIC

Beginner's All-purpose Symbolic Instruction Code, a general purpose, high-level language that has been widely accepted because of its versatility and the ease with which it can be learned. Fluke BASIC has added commands for instrument control.

baud rate

The serial transfer rate in bits per second including all framing bits used to identify the start and end of characters or messages.

binary

A number system based on zero (0) and one (1) representations. It is often used to represent data or instruction codes. There are only two numbers, so digital computers can use binary for their operations because each number can be represented as the state (on or off) of a transistor.

bit

A contraction of binary digit. A bit is either a one or a zero and respresents the smallest single unit of computer information. Bits are often used in groups of eight to represent ASCII characters.

block

Memory size equal to 512 bytes.

bootstrap

A short program permanently recorded in ROM whose only function is to read an operating system program from bulk storage into system memory and transfer control to it.

buffer

A temporary storage area in main memory used to store data.

bulk storage

A device attached to a computer that can store much more program or data information than the computer's main memory can hold. The 1752A incorporates two types of bulk storage: floppy disk and hard disk. Also called mass storage.

bus

The IEEE-488-1978 standardized interconnection system used for connecting instruments. Also, bus can refer to any set of parallel connections that have the same meaning for each unit connected to them.

Bus address

A 7-bit code placed on the IEEE-488 bus in command mode to designate an instrument as a talker or listener.

byte

A grouping of eight bits of information into a coded representation of all or part of a number or instruction. Often a 7-bit ASCII character is referred to as a byte, with the eighth bit available for parity if needed. Bytes are commonly considered as 8-bit storage areas to represent ASCII characters.

chaining

A method of operating a program that is larger than available main memory. The technique is to break the program into smaller elements, and call in the next element from bulk storage as each succeeding element is completed. Requires highly modular programming to be effective. See structured programming.

channel

A communication path opened between an application program and a file or a system device.

character plane

The portion of the display memory that is used for displaying the normal- and double-sized characters. The character set includes the upper- and lower- case alphabet, the ten numerals, and punctuation. See graphics plane.

character string

A grouping of ASCII characters.

cold start

The power-up activities of the 1752A. These include clearing all memory including E-Disk, performing a self-test, and loading the operating system. A cold start occurs when the system is powered up or when RESTART and ABORT are pressed simultaneously.

Command file

A file that, when designated active by FDOS, is used as a substitute for keyboard inputs. For the 1752A, a command file with the name STRTUP.CMD is processed each time the 1752A is initialized by a cold start or power-on.

Command mode

An IEEE-488 term indicating that a controller has set the ATN (attention) line. In this mode, instruments on the bus are addressed or unaddressed as talkers and listeners.

constants

Fixed values which may be floating-point, integer, or string data types.

control character

Used to produce specific actions such as terminating program execution, exiting from the Editor, halting and restarting scrolling.

controller

A device connected to a bus capable of designating instruments as talkers or listeners by using bus message sequences. A device does not need to be programmable to act as a controller. However, only a controller can examine the data or status of instruments to determine appropriate conditions for designation changes. There can be only one active controller on a bus at one time.

CPU

Central Processing Unit, the controlling instruction and data processor in any computer system. In the Fluke 1752A, the CPU is the microprocessor and its supporting components located on the Single Board Computer module.

CRT

Cathode Ray Tube, the display screen on the 1752A front panel.

current position

The pixel location defined by X,Y coordinates in the graphics plane that is the starting position for turning the beam on or off to paint a line, or to move to another location. On power up, the current location is X0, Y0.

cursor

The visible pointer on the CRT display that allows the user to recognize the position being pointed to by the system software.

data

Numerical information that has been collected for interpretation by a program.

data base

A stored and defined collection of data that is made available for report generation or further calculations by a program.

data file

A file holding either random or sequential access information. Contrasted to a program file.

Data lines

Eight of the sixteen bus lines which carry either data or multiline bus messages (Universal, Addressed and Address commands).

Data mode

The default mode of the bus when the controller has left the ATN (attention) line false. All transfers of data or instructions are between instruments.

data processing

The ability to perform calculations on collected data and formatting it into readable reports.

debugging

Any method of detecting and correcting syntax and structure errors in a program.

default

That option which system software selects when the user does not specify an option.

device

A hardware resource that can act as a source or destination of data. In this manual, device is used in two different ways: 1. To represent the internal devices recognized by the Operating System. In this usage, the 1752A's devices are identified by two letters, a number, and a colon. For example, MF0: identifies the mini-floppy drive. 2. The symbol "@" followed by a number from 0 to 30 represents external devices, such as instruments connected to the IEEE-488 bus. The BASIC language statement "PRINT @ 2" followed by program data would address instrument 2 as a listener device and send it program data.

Device Address

A number used by a program to designate an external device for data transfer.

Display control

An ANSI-standard character sequence of ASCII characters which produces a desired display effect such as cursor position or reverse image.

E-Disk

Fluke Trade Mark for the Electronic disk, a memory configuration that makes use of memory as if it were a file-structured device. See Electronic Disk.

editor

A system software program that enables a user to generate and update an application program.

EIA

Electronic Industries Association, publishers of standard RS-232-C for serial data ports.

Electronic Disk

A portion of the memory designated as a file-structured device. Part of the system's dynamic RAM memory is configured so that it functionally emulates a floppy disk. The electronic disk is about 100 times faster than the floppy disk and has no moving parts to wear or cause noise.

EPROM

Erasable Programmable Read Only Memory. A ROM that can be erased and reprogrammed by an equipment manufacturer using specialized equipment.

Escape sequence

A string of characters including an escape (ESC) character, a numeric parameter and a function code which is recognized as a Display control.

expression

A combination of data-names, numeric literals, and named constants, joined by one or more arithmetic operators in such a way that the expression as a whole can be reduced to a single numeric value.

Extended Listener

A listener instrument that requires a two-byte address. See secondary commands.

Extended Talker

A talker instrument that requires a two-byte address. See secondary commands.

FDOS

Floppy Disk Operating System program. FDOS is the executive monitor program of the 1752A and is supplied as a file on the System Disk with the filename FDOS2.SYS. Usually called "the Operating System", FDOS is the 1752A's central program. When any other program is exited, FDOS takes control (unless the BASIC statement SET SHELL has been used to change the environment). The purpose of FDOS is to load other programs.

file

A collection of related information designated by name as a unit.

file-structured device

Any bulk memory device where programs and data may be stored and retrieved via a system directory.

File Utility Program (FUP)

The file management program provided with the standard 1752A software package. Provided on the System Disk as a file with the name FUP.FD2, this utility program permits directory listing, transferring, deleting, and renaming files, and formatting, packing, and zeroing the 1752A's devices.

firmware

Computer programs and data that are recorded in permanent memory. See ROM.

flag

A symbol that indicates a status condition. System flags can be used to indicate the presence of command files or to indicate a state of system readiness.

floating-point variable

A representation of a general-purpose number. They are characterized by wide range (up to 308 places from decimal) and high resolution (up to 15 places). When displayed without modification, up to seven of the digits are displayed, with the last one rounded if necessary. If the decimal is out of range of the display, an exponent of ten is included to bring it to just left of the first number. For example, .00123456789 is displayed as 0.1234567e-02, and 1234567.89 is displayed as 1234568. Note that the inexactness of floating-point representation occasionally must be considered. For example, IF 7*(1.7)=1 will evaluate false. See integer.

floppy disk

A bulk storage recording device that uses a flexible mylar disk similar to recording tape to record programs and data. The location of information on the disk is identified by track (distance from center) and sector (pie-shaped radial subdivision).

flowchart

A pictorial, symbolic representation of a program. Various shapes represent commands, computations, or decisions. A flowchart is the recommended step between an algorithm specification and program writing. It facilitates understanding and debugging because it breaks the program down into logical, sequential modules.

FUP

See File Utility Program

graphics plane

The portion of the display memory where the lines and patterns displayed by the graphics routines are stored. The area is measured in pixels, rather than bytes. One pixel is the smallest amount of graphics information that can be stored or displayed. The graphics plane is 2048 pixels long and 256 high. The display provides a moveable window looking into the graphics plane. The window is 640 by 224 pixels. See character plane.

handshaking

Refers to the 3-wire hardware protocol used to exchange data on the bus. The three bus lines (DAV, NRFD, and NDAC) indicate a remote instrument's readiness to send or receive data.

hexadecimal

A number system based on 16 digits. Sometimes called hex, the system uses A, B, C, D, E, and F, to represent the numbers above 9.

high-level language

Any programming language that requires conversion through a compiler or interpreter into machine code instructions. Examples of high-level languages are BASIC and FORTRAN.

IEEE

Institute of Electrical and Electronic Engineers, Inc., 345 East 47th Street, New York, NY, 10017. The IEEE is the publisher of Standard 488-1978 used for interconnecting instruments to the Fluke Instrument Controller through the bus.

IEEE-488-1978

A bus standard agreed upon by participating instrument manufacturers for the interconnections of instruments into a functional system. Also known as the GPIB (General Purpose Instrumentation Bus). The standard is published and maintained by the IEEE.

Immediate mode

A method to use BASIC directly as each line is typed in rather than storing a sequence of lines as a program for later execution. In Immediate Mode, line numbers are not used and each line is executed as soon as the RETURN key is pressed.

Instrument Controller

In an IEEE-488 system, designates the piece of equipment that asserts control over the bus, and which establishes the roles of other connected equipment as listeners or talkers.

Integer variable

A representation of an exact number. They are characterized by limited range (–32768 to 32767) and numeric resolution. Integers are normally used for event counting, and for comparisions where exactness is required. See also floating-point.

interface

A hardware and software connection of a device to a system. For example, in the Fluke 1752A, the DMA/Floppy Interface is needed for the system to gain access to the floppy disk.

interpreter

A system software program that interprets the statements of a high-level language program (such as BASIC), producing and executing machine code.

lexical file

An intermediate form of an application program that occupies less space and eliminates some processing steps for the Fluke BASIC Interpreter. Line numbers are represented in binary format and all commands and operators are reduced to binary form. Lexical files always have ".BAL" extensions.

listener

A bus device designated by a controller to receive data or instructions from a designated talker or controller. There can be more than one listener on a bus at the same time.

loader

A program which places another program into main memory for execution.

logical expression

An expression containing variables, constants, function references, etc., separated by logical operators and parentheses.

logic operator

A functions that performs comparions, selections, matching, etc. In BASIC, the logical operators are AND, OR, NOT, and XOR. These are used for either Boolean operation or for bit-manipulation.

machine code

The coded bit-patterns of directly executable machine-dependent computer instructions, represented by numbers or binary patterns.

machine-dependent program

A program that operates on a particular model of computer.

machine-independent program

A program that operates on any computer system that has the necessary hardware and supporting software.

main memory

The RAM memory from which the microcomputer directly executes all instructions and which is used for fast, intermediate storage of data or programs.

main memory array

An array that is stored in main memory.

management lines

Five of the sixteen lines on an IEEE-488 bus. The lines are ATN (Attention), IFC (Interface Clear), REN (Remote Enable), EOI (End Or Identify), and SRQ (Service Request), and call for an immediate and specific action, or flag a condition existing on the bus.

Operating System

A computer program that manages the resources of computer through task scheduling, I/O handling, and file management. See FDOS.

operator

A term for symbols within an application program (such as + or <) that identify operations to be performed.

Operator's Keyboard

The Touch-Sensitive Display.

parallel poll

A method of simultaneously checking the status of up to eight instruments on a bus by assigning each instrument a data line to transmit a service request.

parity

A method of error detection that uses one extra bit for each unit of information (such as a byte). The parity bit is set to one or zero so that the total number of one-bits in the byte is even or odd.

pathname

The full designation of a file. The three parts are the device name, file name and extension. The first two are separated by a colon, and the last two by a period.

pixel

Acronym for picture element; the smallest amount of visual information that the display is able to resolve; one dot.

port

A connection point used for data transfer. See interface.

Primary command

An ASCII character typically used as a bus command.

program

Any meaningful sequence of computer instructions that cause a system to accomplish a desired task.

PROM

Programmable Read Only Memory, a memory IC that can be recorded by an equipment manufacturer using specialized equipment.

protection state

Files prepared on the 1752A are assigned a value, either + or – to indicate the intent of the author either to prevent or allow alteration. A file with the + state is protected and will not be written over. A – state indicates that the file may be altered if desired. All newly created files are assigned the – state. All files supplied on the System Disk with the 1752A are protected. The File Utility Program includes commands for changing the protection state of files.

protocol

A set of rules for exchange of information between a system and a device or between two systems.

RAM

Random Access Memory. Through common usage, the term has come to mean the high-speed volatile semiconductor memory that is normally used for system and user memory.

random access

A method of obtaining information out of memory; each word of a file can be accessed via its own discrete address. See also sequential access.

raster

The scanning pattern of an electron beam on a CRT display. A raster display uses the same scan pattern all the time, forming images by turning the beam on and off at appropriate times.

RESTART

Front panel switch that resets the system. When pressed alone, RESTART causes a warm start. When pressed at the same time as ABORT, causes the system to perform a cold start.

ROM

Read Only Memory, used for permanently recorded computer programs and data.

RS-232-C

A digital communications standard agreed upon by participating manufacturers of data communication equipment for the transfer of serial digital data between data communication equipment (DCE) and data terminal equipment (DTE). The 1752A is a DTE device. The standard is published and maintained by the Electronic Industries Association.

scientific notation

A system for describing real or integer numbers via a shorthand form of floating-point notation.

Secondary command

IEEE-488 bus commands used to increase the address length of extended talkers and listeners to two bytes.

serial data

Information transmitted one bit at a time over a single wire at a predefined baud rate.

sequential access

A method of accessing data in a file by looking at each piece of data, in order, until a match is found. See also random access.

serial poll

A method of sequentially determining which instrument on a bus has requested service. One instrument at a time is checked via the eight data lines.

serial port

An external connector that conforms to the industry standard RS-232-C. Normally, asynchronous ASCII codes are used unless otherwise desired.

SET Utility Program

The program that changes the parameters of the 1752A's serial communications ports. Supplied on the System Disk with the filename SET.FD2, this program permits configuring the 1752A so it is able to communicate with other devices that implement the RS-232-C Serial Data Communications standard. Parameters that can be changed include baud rate, parity bit, number of bits per character, stall input and output characters, and time out value.

shell

The 1752A's environment, either defaulted to FDOS or changed by the BASIC language SET SHELL statement. When RESTART is pressed, the 1752A returns to the program named by the SET SHELL statement, rather than to the bootstrap loader PROM.

simple variable

Fluke BASIC program variable that is either an integer or floating-point value (not a character string) and contains only one value (not dimensional).

soft-sectored

In floppy disks, the beginning of every sector on a disk is determined by checking certain data patterns. Hard-sectored disks have predetermined sector beginnings designated by a physical marker, such as a hole.

software

Computer programs and data recorded and used on a medium that can be erased and rewritten by program command.

source

This term has two meanings: 1. The pathname where information presently resides when using a File Utility Program command that moves a file from one place to another; the input side of the channel. 2. An instrument connected to the bus and transmitting either command mode or data mode information.

string variable

An expression that represents collections of characters that may or may not be numeric.

structured programming

A method of programming which require an initial design process to lay out the program structure in a modular form. Structured programming minimizes 'spaghetti code' programs by keeping GOTO statements to a minimum and by using subroutines to structure the program into discrete, easily readable modules.

subroutine

A section of a program that performs a specific function on request of the main program or another subroutine. Subroutines are used in BASIC via the GOSUB statement.

synchronous data

Digital information transmitted in predetermined message block sizes with a clock signal to synchronize the receiver. See asynchronous data.

syntax

The proper grammar required for an interpreter to recognize and execute a program statement.

syntax diagram

A pictorial representation of the grammar required for the execution of a program statement.

system

Any interconnection of instruments or other devices that cooperate to accomplish a task. A controller is an essential part of a system whenever the designations of talkers and listeners needs to be changed during the task. A controller is a necessary part of any system that requires data processing or a centralized control point.

System Device

The designated file-structured device on the 1752A that acts as the primary file storage module. The floppy disk or electronic disk may be designated as the system device by the File Utility program's Assign option. The floppy disk drive (MF0:) is the default system device.

system directory

The listing of program and data files on a bulk storage, file structured device.

system memory

Those portions of the Random Access Memory allocated for use by the operating system and utilities or BASIC Interpreter.

system software

The collection of programs used to handle file management procedures on a system.

Talker

An IEEE-488 connected instrument that has been designated by the controller on the bus to send data to listeners.

Time and Date Utility Program

The program that sets the time and date of the 1752A's real-time clock. Supplied on the System Disk with the filename TIME.FD2, this program accepts the time and date by keyboard inputs, and transmits the information to the real-time clock. With battery backup, the clock maintains the correct time and takes into account leap years. The clock can be used to time and date stamp programs or data collected by programs or to perform an operation at a specified time.

Touch-Sensitive Display

The combination of the display screen and the touch-sensitive panel which acts as the operator's keyboard.

warm start

The activities the 1752A performs when the RESTART switch is pressed: ceases the current operation and returns to the shell. See cold start.

Universal command

A message sent across the data lines of a bus that affects all connected instruments whether or not they are designated as listeners.

user memory

Area reserved in main memory for storage and execution of user-written application programs and data.

variable

A representation of a quantity, or the quantity itself, which can assume any of a given set of values. A variable may be integer, string, or floating point value designators.

virtual array

A matrix stored on a file-structured storage medium as a random access file. Virtual arrays can be integer, string, or floating-point arrays with one or two dimensions. Once a virtual array file has been opened and the virtual array has been dimensioned, the array elements are handled by the programmer exactly as they are in main memory array.

yank buffer

A temporary memory location where the System Editor program can store data "yanked" from a file.

Appendix E
Custom Character Sets

This appendix describes the relationships among data in the Character EPROM (U32), ASCII codes received as input, and the images displayed on the 1752A screen. This information and an EPROM programmer allow you to create custom character sets for your 1752A.

Character cell dot patterns are stored in a 2732A type EPROM. The standard character set capacity is 128 characters. The alternate character set capability provides an additional 128 characters for a total of 256 characters. Each character uses 16 of the PROM's locations (0 though F). In both the standard and the alternate character set modes, 115 characters can be displayed directly. Eleven of the remaining characters are displayed through the Character Graphics Mode.

Depending on the revision level of the Video-Graphics-Keyboard module, the character sets may be identical, or the alternate font may contain a selection of symbols and non-English characters. Each character is contained in a cell 8 dots wide by 14 dots high. Since every dot in each character cell can be displayed, all character codes are available for graphics.

CAUTION

Leave character position 32 (decimal) blank. ASCII character 32 is the space character. Erase operations write this character on the screen, so position 32 must be left blank.

The rules by which standard ASCII display characters are defined follow. This information is provided for reference as you design any characters you wish.

- Standard ASCII characters are 9 dots high and up to 7 dots wide.
- The topmost row is left blank to provide the spacing between the lines.
- The leftmost column is left blank for spacing between characters.
- Row D is reserved for underlines.

  Upper-case characters, numerals, and symbols like the percent and dollar signs, brackets and braces, conform to these rules:

  1. They start on row 1.
  2. They extend to row 9.
- Lower-case letters with ascenders (like the letter 'h') start at row 2.
- Lower-case letters with descenders (like 'g') extend down to row D.
- Other characters (like 'a' or the symbol '@') start on row 4.

Sixteen bytes are reserved in the EPROM for each character cell. The first byte corresponds to the top row of dots in the cell. The fourteenth byte corresponds to the bottom row. The last two bytes are not used, because the hardware does not address these locations.

The hexadecimal EPROM address of each byte is its ASCII code in hexadecimal followed by its byte number within the cell. For example, 412 is the EPROM address of the third row of ASCII character number 41 (A). A 1-bit corresponds to a displayed dot.

EXAMPLE

This example shows how the capital letter H (ASCII 48) is encoded.

ADDRESS IN ROM	1752A DISPLAY								CODE BYTE
	8	4	2	1	8	4	2	1	
480	•	•	•	•	•	•	•	•	00
481	•	1	•	•	•	•	•	1	41
482	•	1	•	•	•	•	•	1	41
483	•	1	•	•	•	•	•	1	41
484	•	1	•	•	•	•	•	1	41
485	•	1	1	1	1	1	1	1	7F
486	•	1	•	•	•	•	•	1	41
487	•	1	•	•	•	•	•	1	41
488	•	1	•	•	•	•	•	1	41
489	•	1	•	•	•	•	•	1	41
48A	•	•	•	•	•	•	•	•	00
48B	•	•	•	•	•	•	•	•	00
48C	•	•	•	•	•	•	•	•	00
48D	•	•	•	•	•	•	•	•	00
48E	•	•	•	•	•	•	•	•	00
48F	•	•	•	•	•	•	•	•	00

THINGS TO KEEP IN MIND

Before starting, program the first 128 locations in the new EPROM with the standard Character set. Set aside the EPROM that is presently in the 1752A. Now program the new character set into the last 128 locations of the copy. Taking these measures will ensure that if an error is made, you will still be to use the original EPROM in the 1752A. Not only is the display needed in order to perform diagnostics, but the original EPROM must be in place if you ever need to send the module in for exchange.

Thirteen character codes in each character set are interpreted as control codes. Eleven of the characters in these locations can be displayed in the Character Graphics Mode. To select this mode, send the sequence ESC [2p.

In the Character Graphics Mode, character patterns are selected for display from the EPROM start addresses listed in the table below.

CHARACTER GRAPHICS MODE EPROM START ADDRESSES

CHARACTER RECEIVED	EPROM PATTERN START ADDRESS (HEXADECIMAL)	
	STANDARD	ALTERNATE
0	000	800
1	070	870
2	080	880
3	090	890
4	0A0	8A0
5	0B0	8B0
6	0C0	8C0
7	0D0	8D0
8	0E0	8E0
9	0F0	8F0
:	110	910

NOTE

The alternate character set contains the double-size graphics characters, so it is necessary to send the escape sequence for double size when addressing these locations.

EPROM INSTALLATION PROCEDURE

CAUTION

You may violate your warranty if you damage the 1752A during the following procedure. To be sure that your warranty stays intact, any Fluke Service Center will be pleased to install the new EPROM.

Once the PROM has been programmed, use this procedure to install it:

1. Set the 1752A Power switch to OFF and disconnect line power.

2. Remove the card cage cover (Phillips head screws) and slide out the Display Module from slot number 2.

3. Use a proper IC removal tool to remove the standard character set EPROM (U32).

NOTE

Save the standard EPROM. If the 1752A should need repair, it must have the standard character set EPROM installed. The Fluke Service Center will need it for proper diagnostic displays. If you return a 1752A for repair without a standard character set EPROM, you'll probably be charged for a new one.

4. Use an IC installation tool to install the custom character set EPROM.

5. Reinstall the Display Module and the rear cover.

6. Connect line power and turn on the power.

7. Test the new PROM by using the sample BASIC language program on the next page to display both character sets. The program works by displaying the primary character set when you command "RUN", and the second when you touch the screen. Both fonts are displayed in double-size to let you see the characters more clearly, and the program toggles between the two character sets for comparison.

```
10 ON CTRL/C GOTO 200
20 E$ = CHR$(27) + "[" \ BL$=" " \ FL%=0
30 PRINT E$ + "1p";
40 PRINT CHR$(15); \ IF FL% = 1 THEN 70
50 PRINT CPOS (1,0); "Standard - Touch Screen for Alternate"
60 GOTO 80
70 PRINT CPOS (1,0); "Alternate-Touch Screen for Standard" \PRINT CHR$(14);
80 PRINT CPOS (2,4);
90  FOR I% = 1 TO 6 \ PRINT CHR$(I%); \ NEXT I%
100 FOR I% = 7 TO 15 \ PRINT BL$; \ NEXT I%
110 FOR I% = 16 TO 26 \ PRINT CHR$(I%); \ NEXT I% \ PRINT BL$;
120 FOR I% = 28 TO 31 \ PRINT CHR$(I%); \ NEXT I%
130 PRINT CPOS (4,4); \ FOR I% = 32 TO 63 \ PRINT CHR$(I%); \ NEXT I%
140 PRINT CPOS (6,4); \ FOR I% = 64 TO 95 \ PRINT CHR$(I%); \ NEXT I%
150 PRINT CPOS (8,4); \ FOR I% = 96 TO 127 \ PRINT CHR$(I%); \ NEXT I%
160 WAIT FOR KEY \ ON KEY GOTO 170
170 K% = KEY \ FL% = -FL% + 1
180 GOTO 40
200 PRINT CHR$(27) + "[p" \ PRINT CHR$(15) \ END
```

EPROM Programming Worksheet

This page can be photocopied for aiding in the design of custom character sets.

Address	8	4	2	1	8	4	2	1	Data
_ _ 0									_ _
_ _ 1									_ _
_ _ 2									_ _
_ _ 3									_ _
_ _ 4									_ _
_ _ 5									_ _
_ _ 6									_ _
_ _ 7									_ _
_ _ 8									_ _
_ _ 9									_ _
_ _ A									_ _
_ _ B									_ _
_ _ C									_ _
_ _ D									_ _
E									
F									

Address	8	4	2	1	8	4	2	1	Data
_ _ 0									_ _
_ _ 1									_ _
_ _ 2									_ _
_ _ 3									_ _
_ _ 4									_ _
_ _ 5									_ _
_ _ 6									_ _
_ _ 7									_ _
_ _ 8									_ _
_ _ 9									_ _
_ _ A									_ _
_ _ B									_ _
_ _ C									_ _
_ _ D									_ _
E									
F									

_ _ 0									_ _
_ _ 1									_ _
_ _ 2									_ _
_ _ 3									_ _
_ _ 4									_ _
_ _ 5									_ _
_ _ 6									_ _
_ _ 7									_ _
_ _ 8									_ _
_ _ 9									_ _
_ _ A									_ _
_ _ B									_ _
_ _ C									_ _
_ _ D									_ _
E									
F									

_ _ 0									_ _
_ _ 1									_ _
_ _ 2									_ _
_ _ 3									_ _
_ _ 4									_ _
_ _ 5									_ _
_ _ 6									_ _
_ _ 7									_ _
_ _ 8									_ _
_ _ 9									_ _
_ _ A									_ _
_ _ B									_ _
_ _ C									_ _
_ _ D									_ _
E									
F									

EPROM Programming Worksheet

Address	8	4	2	1	8	4	2	1	Data
_ _ 0									_ _
_ _ 1									_ _
_ _ 2									_ _
_ _ 3									_ _
_ _ 4									_ _
_ _ 5									_ _
_ _ 6									_ _
_ _ 7									_ _
_ _ 8									_ _
_ _ 9									_ _
_ _ A									_ _
_ _ B									_ _
_ _ C									_ _
_ _ D									_ _
E									
F									

Address	8	4	2	1	8	4	2	1	Data
_ _ 0									_ _
_ _ 1									_ _
_ _ 2									_ _
_ _ 3									_ _
_ _ 4									_ _
_ _ 5									_ _
_ _ 6									_ _
_ _ 7									_ _
_ _ 8									_ _
_ _ 9									_ _
_ _ A									_ _
_ _ B									_ _
_ _ C									_ _
_ _ D									_ _
E									
F									

_ _ 0									_ _
_ _ 1									_ _
_ _ 2									_ _
_ _ 3									_ _
_ _ 4									_ _
_ _ 5									_ _
_ _ 6									_ _
_ _ 7									_ _
_ _ 8									_ _
_ _ 9									_ _
_ _ A									_ _
_ _ B									_ _
_ _ C									_ _
_ _ D									_ _
E									
F									

_ _ 0									_ _
_ _ 1									_ _
_ _ 2									_ _
_ _ 3									_ _
_ _ 4									_ _
_ _ 5									_ _
_ _ 6									_ _
_ _ 7									_ _
_ _ 8									_ _
_ _ 9									_ _
_ _ A									_ _
_ _ B									_ _
_ _ C									_ _
_ _ D									_ _
E									
F									

Appendix F
Primary Character Set

FLUKE® ASCII & IEEE-488 BUS CODES

B^7 B^6 B^5 B^4 BITS	0 0 0 0	0 0 0 1	0 0 1 0	0 0 1 1	0 1 0 0	0 1 0 1	0 1 1 0	0 1 1 1 (2^7 2^6 2^5 2^4)
B^3 B^2 B^1 B^0	CONTROL		NUMBERS SYMBOLS		UPPER CASE		LOWER CASE	
0 0 0 0	0 0 **NUL** NUL	16 10 **ρ** DLE	32 20 **SP** SPACE MLA0	48 30 **0** 0 MLA16	64 40 **@** @ MTA0	80 50 **P** P MTA16	96 60 **`** MSA0	112 70 **p** p MSA16
0 0 0 1	1 1 **β** SOH GTL	17 11 **σ** DC1	33 21 **!** ! MLA1	49 31 **1** 1 MLA17	65 41 **A** A MTA1	81 51 **Q** Q MTA17	97 61 **a** a MSA1	113 71 **q** q MSA17
0 0 1 0	2 2 **γ** STX	18 12 **τ** DC2	34 22 **"** " MLA2	50 32 **2** 2 MLA18	66 42 **B** B MTA2	82 52 **R** R MTA18	98 62 **b** b MSA2	114 72 **r** r MSA18
0 0 1 1	3 3 **δ** ETX	19 13 **υ** DC3	35 23 **#** # MLA3	51 33 **3** 3 MLA19	67 43 **C** C MTA3	83 53 **S** S MTA19	99 63 **c** c MSA3	115 73 **s** s MSA19
0 1 0 0	4 4 **ε** EOT SDC	20 14 **φ** DC4 DCL	36 24 **$** $ MLA4	52 34 **4** 4 MLA20	68 44 **D** D MTA4	84 54 **T** T MTA20	100 64 **d** d MSA4	116 74 **t** t MSA20
0 1 0 1	5 5 **ζ** ENQ PPC	21 15 **χ** NAK PPU	37 25 **%** % MLA5	53 35 **5** 5 MLA21	69 45 **E** E MTA5	85 55 **U** U MTA21	101 65 **e** e MSA5	117 75 **u** u MSA21
0 1 1 0	6 6 **η** ACK	22 16 **ψ** SYN	38 26 **&** & MLA6	54 36 **6** 6 MLA22	70 46 **F** F MTA6	86 56 **V** V MTA22	102 66 **f** f MSA6	118 76 **v** v MSA22
0 1 1 1	7 7 **BEL** BEL	23 17 **ω** ETB	39 27 **'** ' MLA7	55 37 **7** 7 MLA23	71 47 **G** G MTA7	87 57 **W** W MTA23	103 67 **g** g MSA7	119 77 **w** w MSA23
1 0 0 0	8 8 **BS** BS GET	24 18 **Ω** CAN SPE	40 28 **(** (MLA8	56 38 **8** 8 MLA24	72 48 **H** H MTA8	88 58 **X** X MTA24	104 68 **h** h MSA8	120 78 **x** x MSA24
1 0 0 1	9 9 **HT** HT TCT	25 19 **√‾** EM SPD	41 29 **)**) MLA9	57 39 **9** 9 MLA25	73 49 **I** I MTA9	89 59 **Y** Y MTA25	105 69 **i** i MSA9	121 79 **y** y MSA25
1 0 1 0	10 A **LF** LF	26 1A **→** SUB	42 2A ***** * MLA10	58 3A **:** : MLA26	74 4A **J** J MTA10	90 5A **Z** Z MTA26	106 6A **j** j MSA10	122 7A **z** z MSA26
1 0 1 1	11 B **VT** VT	27 1B **ESC** ESC	43 2B **+** + MLA11	59 3B **;** ; MLA27	75 4B **K** K MTA11	91 5B **[** [MTA27	107 6B **k** k MSA11	123 7B **{** { MSA27
1 1 0 0	12 C **FF** FF	28 1C **↑** FS	44 2C **,** , MLA12	60 3C **<** < MLA28	76 4C **L** L MTA12	92 5C **** \\ MTA28	108 6C **l** l MSA12	124 7C **¦** ¦ MSA28
1 1 0 1	13 D **CR** CR	29 1D **÷** GS	45 2D **-** - MLA13	61 3D **=** = MLA29	77 4D **M** M MTA13	93 5D **]**] MTA29	109 6D **m** m MSA13	125 7D **}** } MSA29
1 1 1 0	14 E **SO** SO	30 1E **Σ** RS	46 2E **.** . MLA14	62 3E **>** > MLA30	78 4E **N** N MTA14	94 5E **∧** ∧ MTA30	110 6E **n** n MSA14	126 7E **~** ~ MSA30
1 1 1 1	15 F **SI** SI	31 1F **≈** US	47 2F **/** / MLA15	63 3F **?** ? UNL	79 4F **O** O MTA15	95 5F **—** — UNT	111 6F **o** o MSA15	127 7F **■** RUB-OUT MSA31
2^3 2^2 2^1 2^0	ADDRESSED COMMANDS	UNIVERSAL COMMANDS	LISTEN ADDRESSES		TALK ADDRESSES		SECONDARY ADDRESSES OR COMMANDS	

KEY

decimal		hex
	38 26 **&** & MLA6	1722A DISPLAY
ASCII		IEEE-488

Appendix G
System Diagnostic Software

INTRODUCTION

This appendix covers the operation of the System Diagnostic software and assists the user in diagnosing problems with the 1752A Data Acquisition System.

The System Diagnostic software is provided on a floppy disk and is designed to be a customer, manufacturing, and field service tool. Successful completion of testing the 1752A using the System Diagnostic software gives the user confidence that the 1752A is operating properly.

MAINTENANCE PHILOSOPHY

The maintenance philosophy for field-service repair is at the module level, including the SBC, VGK, video electronics, power supply, floppy disk drive, and the options. Faulty modules are identified by using the System Diagnostic software.

Replacement modules are available through your local Fluke Service Center. Contact your local Fluke Service Center for details on in-warranty repair, and contact the Module Exchange Center for out-of-warranty repair.

DIAGNOSTIC DESIGN

The System Diagnostic software uses a menu system and presents the 1752A as a set of modules. A module is either a circuit board, an externally connected peripheral, or a major subsystem. There is a set of subtests for each module. Each subtest covers one specific function of a module. The test selections are presented on the 1752A display and use touch-sensitive command blocks for making choices.

The menu system allows individual modules to be selected and tested. A particular combination of tests is called a test configuration. The Standard Test Configuration can be used to test a standard 1752A with no options installed. Other test configurations can be created, edited, stored, and recalled as Configurations A, B, and C.

Some module tests can cover up to five units (modules of the same type). For example, the test for the 512K byte Memory Expansion module (Option -007) can test from one to five modules.

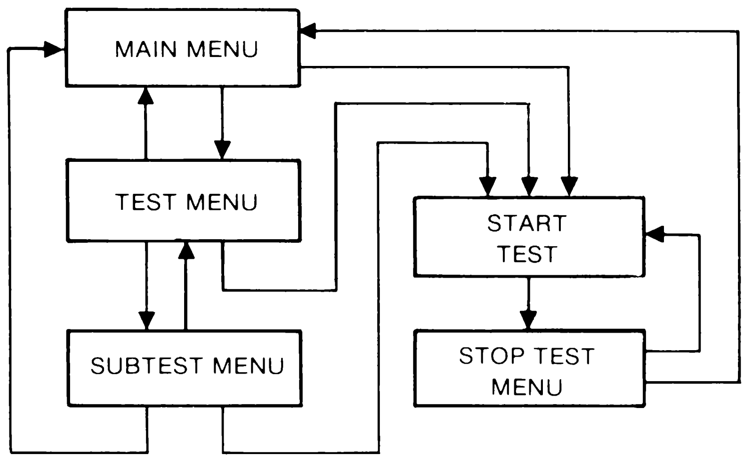
MENU LEVELS

The diagnostic software has three menu levels:

- The Main Menu presents the test configurations, test modes, and a destination for the test results.
- The Test Menu presents the modules and options that can be selected for testing.
- The Subtest Menu presents the subtests that can be selected for each module and option.

Here is the menu level diagram:

1752A SYSTEM DIAGNOSTIC



OPERATION

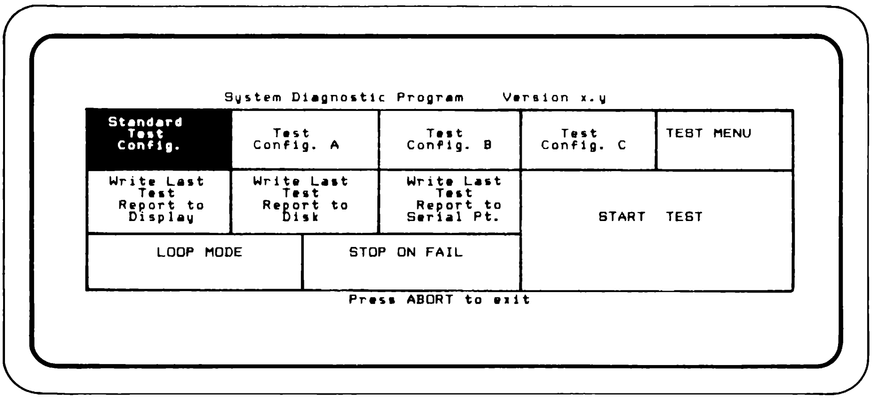
NOTE

Copy any programs stored in the 1752A E-Disk onto a floppy disk before loading the System Diagnostic software. When the System Diagnostic software is loaded into the 1752A memory, portions of the E-Disk and program memory in the 1752A are erased.

Use the following procedure to run the System Diagnostic software.

1. Insert the System Diagnostic disk into the disk drive. Power up the 1752A, or press the RESTART and ABORT switches on the front panel simultaneously if the 1752A is already running.

2. After the System Diagnostic software is loaded from the floppy disk, the 1752A display reads:



3. Touch the START TEST command block on the display to execute the Standard Test Configuration, or select individual tests as described in the following paragraphs.

4. The test will stop and display a prompt at any point where a user response is required. At the end of the test, the results will be displayed on the screen.

NOTE

If any problems are encountered while loading the System Diagnostic software or during the execution of a test, refer to "Troubleshooting" at the end of this appendix.

Main Menu

The Main Menu has 11 command blocks, described below:

1. The Standard Test Configuration is a set of tests that has been preselected to test the proper functioning of standard 1752A system modules. The tests allow untrained personnel to perform diagnostic testing. Additional hardware or test cables are not used with this test configuration. If the 1752A passes the Standard Test Configuration, the 1752A is fully functional up to the I/O drivers. The I/O drivers are tested with additional subtests and external hardware.

   NOTE

   The standard test configuration for the 1752A system diagnostic disk includes subtests for the 17XXA-010 Analog Measurement processor.

2. Test Config. A, or B, or C

   The purpose of Test Configurations A, or B, or C is to allow the creation of special test configurations. To create a configuration, select any combination of module tests and subtests, and store the special configuration if it will be needed again. Instructions on storing special configurations are given in the descriptions of the Test Menu selections.

3. Test Menu

   By pressing the Test Menu command block, the Test Menu is displayed. From this menu, alternate test configurations can be defined and stored by the user.

4. Write Last Test Report to Display, or Disk, or Serial Pt. (Port).

   After a test has stopped running, these command blocks allow the user to redisplay the test results or send them to a floppy disk or printer for future use.

If Write Last Report to Disk is selected, the test results are stored in the file MF0:REPORT.DAT. If Write Last Report to Serial Pt. is selected, the test results are sent to serial port KB1: and can be output to a printer for a hard copy. The default baud rate set by the start-up command file when the System Diagnostic software is loaded is 1200 baud. Set your printer to this baud rate. See Section 5 of the 1752A System Guide for more information about RS-232-C communications and using the SET Utility Program.

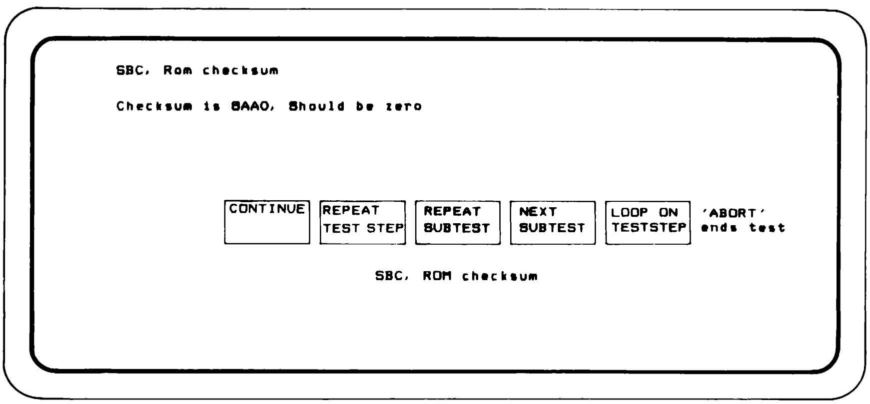
5. Loop Mode

   The LOOP MODE command block performs like a toggle switch. When it is off (the power-up default), the test configuration executes one time and then stops. When LOOP MODE is on, the test configuration continues until the ABORT switch on the 1752A front panel is pressed.

6. Stop On Fail

   The STOP ON FAIL command block performs like a toggle switch. When the System Diagnostic software is first loaded, STOP ON FAIL is off. When STOP ON FAIL is off, the System Diagnostic software tests do not stop if a failure occurs.

   When STOP ON FAIL is selected, the System Diagnostic software will halt the test and give a Stop Test Menu whenever a failure is encountered. Here is an example of a Stop Test Menu:



7. Start Test

The START TEST command block begins execution of the currently selected test configuration. The tests are executed and any failures are logged.

To begin any of the test configurations, press START TEST on the 1752A display. As the test runs, the 1752A display is updated at the completion of each subtest. To return to the Main Menu, or to halt any test, press the ABORT switch on the 1752A front panel. If all the tests in the chosen configuration are successful, the display reads 'No failures'. If any of the tests in the selected configuration are unsuccessful, three things happen:

a. The 1752A display indicates the name of the test in progress at the time of the failure.

b. The errors are stored on the E-Disk.

c. A Stop Test Menu is displayed with an error log and more operator choices.

CAUTION

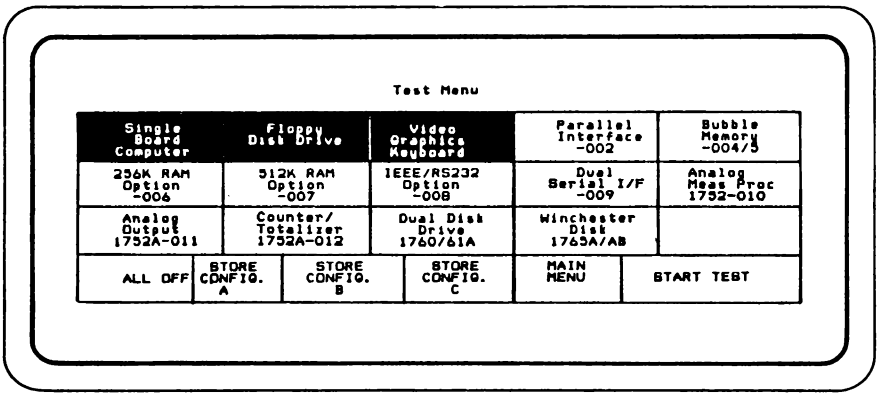
Portions of the System Diagnostic software can write over areas of RAM used by FDOS. Always reload the operating system by executing a cold start (press the RESTART and ABORT buttons on the front panel simultaneously) if ⟨CTRL⟩/P is used to stop the System Diagnostic software.

TEST MENU

The Test Menu lets the user specify which modules are to be included in a particular test configuration. An example of a test configuration is the Standard Test Configuration which includes tests for the SBC, VGK, and floppy disk drive. If optional modules are installed in the 1752A, they can also be selected for testing from this menu.

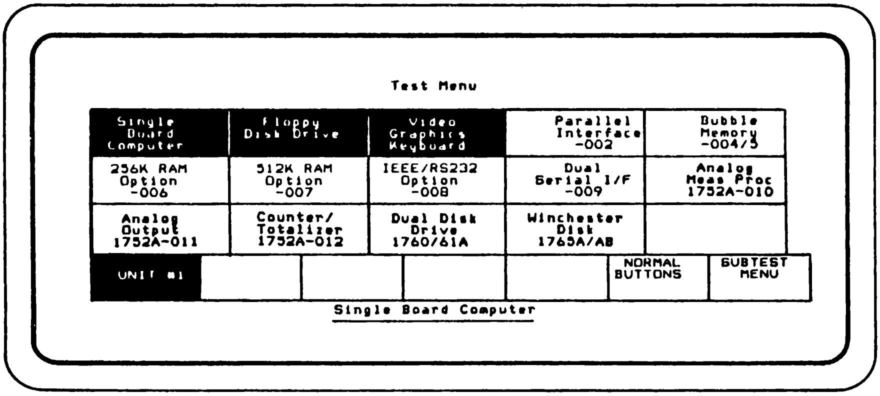
The Test Menu is displayed on two screens. The first screen has 15 possible module selection blocks and six additional command blocks. The module selection blocks act as toggle switches to select and deselect the module.

The Test Menu includes module selection blocks for the SBC, Floppy Disk Drive, and VGK. It also includes module selection blocks for options and peripherals. Here is an example of the first screen of the Test Menu:



When a module is selected from the Test Menu the second screen is displayed. The lower six command blocks in the Test Menu are replaced with another set of command blocks. If the desired module is already selected, it must be toggled off and on again to display the second screen.

From the second screen of the Test Menu, the user can choose how many of the selected modules are to be tested, return to the first screen of the Test Menu, or go to the Subtest Menu for the selected module. Here is an example of the second screen of the Test Menu:



The command blocks for both screens of the Test Menu are described in the following paragraphs.

1. All Off

   The All Off command block deselects all the modules in the Test Menu that were previously selected for testing.

2. Store Configuration A, or B, or C

   After a test configuration has been selected by lighting the appropriate display blocks in the Test Menu and subtest Menus, the test configuration may be stored as Test Configuration A, B, or C by touching one of these command blocks. The special configuration is stored on the floppy disk and can be used again by selecting it from the Main Menu.

3. Main Menu

   When the Main Menu command block is selected, the System Diagnostic software returns to the Main Menu.

4. Start Test

   When the Start Test command block is touched, the currently selected test configuration begins to execute.

5. Unit # 1, 2, 3, 4, 5

   From the second screen of the Test Menu the user can select the number of units (modules of the same type) to be included in the testing.

6. Normal Buttons

   When the Normal Buttons command block is selected, the System Diagnostic software returns to the first screen of the Test Menu.

7. Subtest Menu

   When the Subtest Menu command block is selected, the Subtest Menu for the chosen module is displayed.

SUBTEST MENUS

There is a Subtest Menu for each module listed in the Test Menu. From the Subtest Menu the user can select specific tests to be executed for the chosen module. This is useful for obtaining more specific information about a module that has failed. For example, a failure at an RS-232 port may be due to a problem on either side of the I/O buffers. To isolate the problem, there is an Internal Loopback Test, an External Loopback Test, and a Port-to-Port Loopback Test, any of which can be selected to exercise a particular portion of the module circuitry.

For each module subtest, there are 15 toggle type subtest selection blocks and five command blocks. An example of the SBC Subtest Menu is shown below.

Single Board Computer

Clock	Rom Checksum	Non Destructive Macrostore	Non Destructive RAM	Address Bus
Destructive Macrostore (exit SD)	Destructive RAM (exit SD)	RS232 Internal Loop	RS232 External Loop	RS232 Port to Port Loop
IEEE Internal Loop	IEEE Port to Port Loop			
ALL OFF	SELECT STANDARD SUBTESTS	MAIN MENU	TEST MENU	START TEST

The command blocks in the Subtest Menu operate identically to the command blocks described above for the Test Menu. The subtest selection blocks for each of the modules are described in the following paragraphs.

NOTE

The subtest selection blocks labelled "exit SD" are stand-alone programs that are loaded into memory, deleting the System Diagnostic software in the process. They cannot be run in LOOP MODE, because they do not return to SD after the test is completed. To reload the System Diagnostic software, press the RESTART and ABORT buttons simultaneously.

Subtest Descriptions

SBC (Single Board Computer)

The SBC Subtest Menu includes the following tests:

1. Clock

   Checks time rollover and storage of all stages of the real-time clock registers.

2. ROM Checksum

   Generates a checksum for the contents of the BOOT ROM. The test fails if the checksum is not equal to zero.

3. Non-Destructive Macrostore

   Performs a read/write test of Macrostore memory. The contents of Macrostore are left intact at the end of the test.

4. Non-Destructive RAM

   This tests portions of SBC memory not currently used by the System Diagnostic software. The test checks for stuck data bits, addressing errors, refresh errors, parity errors, and timeout errors.

5. Address Bus Test

   This test uses the same algorithm as the Non-Destructive RAM test, but covers all of physical memory, on the SBC and on any RAM options that are installed in the 1752A. Timeout errors are not reported, so if a RAM option fails to respond (missing or faulty) it will not be tested. When known good RAM options are installed in the 1752A, this test will check out the address and data buffers on the SBC.

6. Destructive Macrostore

   This is a comprehensive memory test including pattern sensitivity. The System Diagnostic software loads another program from the disk to do this test, deleting the Operating System and the System Diagnostic software from the 1752A memory in the process. This test takes several minutes to complete. At the conclusion of the test, the Operating System and the System Diagnostic software must be reloaded.

7. Destructive RAM

   This is an intensive memory test requiring approximately 30 minutes to execute. When the test is over, the System Diagnostic software must be reloaded because the test deletes the software from memory.

8. RS-232-C Internal Loop

   This test sets an internal loopback mode on the UART (Universal Asynchronous Receiver Transmitter) to allow the 1752A to send and receive data internally. The I/O buffers to off-board devices are not tested.

9. RS-232 External Loop

   This test uses a special test connector (Fluke P/N 732107) to allow any port to talk to itself. This tests the buffers and registers of the RS-232 port.

10. RS-232 Port-to-Port Loop

    A Null Modem cable (Fluke Model Number Y1705) is required to permit two RS-232 ports to communicate with each other and test the entire communications interface including the buffers. The Option 17XXA-008 IEEE/RS-232 Interface module must be installed to run this test. The Null Modem cable connects the RS-232 ports on the SBC and the Option -008 module.

11. IEEE Internal Loop

    The IEEE-488 Internal Loop tests the operation of the IEEE-488 interface. The buffers to the IEEE-488 bus are not tested.

12. IEEE Port to Port Loop

    The IEEE-488 Port-to-Port Loop Test verifies the port's ability to transmit or receive data and drive the IEEE-488 bus. Use any standard Fluke IEEE-488 cable, Y8021, Y8022, or Y8023. The Option 17XXA-008 IEEE/RS-232 module must be installed to run this test. The IEEE cable connects the IEEE-488 ports on the SBC and the Option -008 module. Switch positions 5 and 6 on the SBC must be in the "OFF" position and switch positions 5 and 6 on the Option -008 module must be in the "ON" position for the test to complete successfully.

    NOTE

    This test requires the user to remove the SBC or Option -008 module from the 1752A chassis in order to verify proper switch settings. It is not necessary to run this test unless there are IEEE-488 bus problems that cannot be identified using the Standard Test Configuration supplied with the System Diagnostic software. If it becomes necessary to run this test, refer to the discussion on "Installing Hardware Options" in Section 9 in the 1752A System Guide. This discussion explains how to remove the rear panel to gain access to the SBC and Option -008 module. Also, be sure to return the SBC and -008 option to their initial switch configuration after running the test.

13. Dip Switch SW1 Status Test

    The Dip Switch SW1 Status Test reads the Single Board Computer switch settings and displays their status. Switch 5, which sets the System Controller function, is not tested by the SW1 Status Test. Correct switch settings for the SBC may be found in Section 2 of this manual.

Floppy Disk Drive

The Floppy Disk Drive menu includes the following tests:

1. Write Protect Switch

   This test displays the status of the Write Protect Switch. Removing or inserting a disk causes the switch to toggle. The System Diagnostic software tests to insure that the switch does toggle. When testing is complete, touch the screen, and the System Diagnostic software reports its findings.

2. Track 0 Sw & Stepper Motor

   A disk is not needed for this test. The read/write head is moved from track 00 to track 39 and back again to test the operation of the track 00 indicator and the stepper motor.

3. Disk RPM Check/Adjust

   A disk must be loaded to perform this subtest. Any disk will do, because nothing is written or read from the disk.

   In this subtest the disk speed is measured and tested to insure it is within tolerance. The disk speed is displayed continuously if the System Diagnostic software is not in the LOOP MODE. When the System Diagnostic software is in the LOOP MODE, the disk speed is sampled and displayed for about 15 seconds, then the test continues to the next subtest. If the test results indicate that the disk speed needs to be adjusted, please refer to the 1752A Service Manual or return the unit to your local Fluke Service Center for servicing.

4. Format Disk

   Formats a floppy disk in standard 1752A (double-sided) format. The operator will be asked for confirmation before proceeding.

5. Bad Block Scan

   Each block on the disk is read and checked for errors. A total of 1600 blocks is read (two passes over the disk). The disk is never written on. Use this subtest to check a disk for bad blocks. If this subtest is used to check the disk drive itself, a disk with flawless format is required.

6. Soft Error Rate

   This subtest does an extensive test of the disk drive's ability to read a worst case data pattern over many passes of the disk. A scratch disk with error free format is required. After writing the worst case data pattern (hex 6DB6DB...) over the entire disk surface, the disk is read for 306 passes (over 1 billion bits). This test takes approximately 3 hours to complete.

   The display indicates the progress of the subtest while the disk is being read, showing the disk pass number, blocks read, bits read, and the current number of soft and hard errors. Refer to "Troubleshooting" at the end of this section for a description of soft and hard errors.

   If the System Diagnostic software is in LOOP MODE, the total number of disk passes, blocks read, etc., for all the times the subtest was executed, are also shown in the display. The display is updated at the end of each disk pass.

7. Random block I/O

   This subtest tests the floppy disk drive's worst case ability to read, write, and seek. It is identical to the Soft Error Rate test except that the block number is chosen at random. A seek to the chosen block is followed by writing the worst case data pattern and reading it back. The total number of blocks tested is 800. The soft and hard errors are handled the same as in the Soft Error Rate Test.

VGK (Video/Graphics/Keyboard Interface)

The VGK menu includes the following subtests:

1. Alignment pattern

   Displays the alignment pattern used for the initial factory setup of the CRT and permits later checking for shift of the display.

2. Keyboard

   A picture of the keyboard is displayed on the screen. Each time a key is pressed on the 1752A keyboard, the corresponding key on the display toggles either on or off. Follow the instructions to light all the keys on the display and then touch the screen to continue. The diagnostic software records an error if a keystroke was not detected.

3. Touch-Sensitive Overlay

   The Touch-Sensitive Overlay (TSO) grid is displayed. Each square covers exactly one TSO touch pad. Light each square by touching it. Press each one again to turn it off. The subtest passes if each touch pad responds at least twice. If a square does not work, exit the subtest by pressing the ABORT switch on the 1752A front panel.

NOTE

Graphics RAM, Vector Generator, and character Attribute RAM tests will not work with versions of the VGK Firmware prior to version 1.4.

4. Character RAM

   Performs a read/write test of character and attribute memory and checks for stuck data bits and addressing errors. The display will be filled with a test pattern during this test.

5. Graphics RAM

   Performs a read/write test of graphics memory and checks for stuck data bits and addressing errors. The display will be filled with a test pattern during this test.

6. Vector Generator

   Performs a comprehensive test of the vector generator by drawing a series of vectors on the screen and reading back the contents of graphics memory.

7. Display Tests

   Provide visual confirmation that the character generator, character graphics, and character attributes sections of the VGK are functional. Touching the screen cycles the test through several displays including double-size and single-size mode with attributes, standard and alternate character sets, and single-size and double-size character graphics. No error reporting is done, but information on the screen indicates what to look for.

8. Remote Control

   Performs a functional test of the Handheld Remote Control Unit (RCU) supplied with the 1722A/AP Instrument Controller. Each LED on the RCU is turned on in sequence, and then the three push buttons are tested for continuity. Prompts indicate which RCU buttons to use to proceed through the test steps. A failure is indicated any time an LED does not light at the appropriate time, or when the test fails to respond to a button press. Exit the subtest by pressing the ABORT button on the front panel.

Parallel Interface Option -002

PIBTST is a program that tests up to three Parallel Interface Modules (Option 17XXA-002). The program performs three separate tests including writing to a port and reading back from the same port (Readback), writing to one port and reading back on the other port (Loopback), and an Interrupt test. When the program runs, the numbers of the modules under test are displayed across the top of the screen and the tests that are executing are displayed down the left side of the screen. In order to pass the Loopback test, a special test cable (JF/PN 632968) must be connected between the two ports on the module. If the user does not have a test connector, the Readback and Interrupt tests may be run individually by touching the PASS/FAIL block on the screen at the bottom of the column corresponding to the module under test. At this point a second menu is displayed. The user may run the tests individually on either port of the selected module by touching the screen at the appropriate point.

Bubble Memory Options -004 and -005

MBXTST is a program for testing up to three Bubble Memory Modules (Options 17XXA-004 and 17XXA-005). The program writes test patterns to the Bubble Memory to check for bubble collapse errors (a "1" bit turning into a "0") and pattern sensitivity problems with the Formatter/Sense Amplifier (FSA). Before executing the test, the program checks to see if there are any files on the bubble devices MB0: through MB3: and the user is asked to confirm whether he wants the files deleted. The bubble devices must be formatted in order for the test to execute properly. The bubble memories were formatted at the factory and should not need to be reformatted unless there is a problem with the module. Any errors found during the test will be displayed on the screen and summarized at the end of the test. If the test will not execute properly try formatting the bubble memory using the File Utility Program (refer to Section 4 of the System Guide).

256K RAM Option -006

Up to five units can be tested at once by selecting the unit numbers from the Test Menu. Each unit number corresponds to a specific switch setting as described in the information supplied with the option or in Section 9 of the 1752A System Guide.

The menu for the 256K RAM option includes the following tests:

1. Non-Destructive RAM Test

   This subtest operates the same as the SBC Non-destructive RAM test.

2. Destructive RAM Test

   This subtest operates the same as the SBC Destructive RAM test. The test may take several hours to complete depending on how many Option -006 modules are installed.

512K RAM Option -007

Up to five units can be tested at once by selecting the unit numbers from the Test Menu. Each unit number corresponds to a specific switch setting as described in the information supplied with the option or in Section 9 of the 1752A System Guide.

The menu for the 512K RAM includes the following tests:

1. Non-Destructive RAM Test

   This subtest operates the same as the SBC Non-destructive RAM test.

2. Destructive RAM Test

   This subtest operates the same as the SBC Destructive RAM test. The test may take several hours to compete depending on how many Option -007 modules are installed.

IEEE-488/RS-232C Option -008

The subtest for the IEEE-488/RS-232C option operates the same as SBC Loop and Switch Status Tests. The cables described in the section on SBC subtests are also used for the IEEE and RS-232 port tests for the -008 option. For the IEEE Port-to-Port test, however, switch position 5 and 6 on the SBC are set to the "ON" position and switch positions 5 and 6 on the Option -008 module are set to the "OFF" position. Here is a list of the subtests for Option -008.

1. RS-232 Internal Loop
2. RS-232 External Loop
3. RS-232 Port-to-Port Loop
4. IEEE-488 Internal Loop
5. IEEE-488 Port-to-Port Loop
6. Dip Switch SW1 Status

NOTE

The IEEE-488 port-to-port Loop test requires the user to remove the SBC or Option -008 module from the 1752A chassis in order to verify proper switch settings. It is not necessary to run this test unless there are IEEE-488 bus problems that cannot be identified using the Standard Test Configuration supplied with the System Diagnostic software. If it becomes necessary to run this test, refer to the discussion on "Installing Hardware Options" in the section entitled "Options" in the 1752A System Guide. This discussion explains how to remove the rear panel to gain access to the SBC and Option -008 module. Also, be sure to return the SBC and -008 option to their initial switch configuration after running the test.

Dual Serial Interface Option -009

SPTEST performs a port-to-port loopback test between the two ports on up to three -009 Options installed in the 1752A. 256 characters are transmitted and received on each port. An RS-232 Null Modem cable (Fluke Model Y1705) must be connected between the two ports on each -009 Option to test the RS-232 interface. A DSI Test Cable (P/N

754648) is required to test the RS-422 or 20 mA current loop interfaces. The port addresses must be set according to the information for the -009 Option in Section 9 of the System Guide.

Analog Measurement Processor Option -010

NOTE

Some of the Analog Measurement Processor subtests require the Solar Cell Board (P/N 765206) and Line Sync Transformer Accessory Y1752 (or equivalent). The Solar Cell Board is supplied with the 1752A System. If another line sync transformer is used, its output should not exceed 5 V ac, peak.

Up to four units can be tested at once by selecting the unit numbers from the Test Menu. Each unit number corresponds to a board address as shown below. (To change a board's address, refer to section 3 of the 1752A Data Acquisition and Control manual.)

Unit 1 = board address 0
Unit 2 = board address 1
Unit 3 = board address 2
Unit 4 = board address 3

In the Analog Measurement Processor subtests, the lowest 16 channels of each board are referred to as channels 0-15, and the highest 16 channels are referred to as channels 16-31.

1. EEPROM Checksum Test

   This test generates a checksum for the contents of the Analog Measurement Processor's firmware. The test fails if the checksum is non-zero.

2. RAM Test

   This test performs a read/write check on the Analog Measurement Processor's RAM. The test verifies proper operation of the board's bus interface logic and identifies stuck data bits or board addressing errors.

3. Calibrate Routine

   This routine allows you to calibrate the Analog Measurement Processor. The routine requires that the board's firmware be write-enabled. For calibration instructions, refer to the Calibration Procedure in Section 3 of the 1752A Data Acquisition and Control manual.

   Note that the Calibrate Routine halts and waits for user input. Consequently, the System Diagnostic Software cannot be run unattended while in the loop mode if this routine is chosen.

4. Phase-Locked Loop Test

   This test verifies that the Analog Measurement Processor's phase-locked loop circuitry is operating properly. The test requires a reference signal from a line sync transformer (Fluke Accessory Y1752 or equivalent). Connect the output of the line sync transformer to the external sync input (connector J3) on the Analog Measurement Processor.

   The Phase-Locked Loop Test displays two sync frequencies: a 50/60 Hz Sync Frequency and a 400 Hz Sync Frequency. If the board is jumpered for 50/60 Hz operation, the 50/60 Hz Sync Frequency should be approximately 50/60 Hz. If the board is jumpered for 400 Hz operation, the 400 Hz Sync Frequency should be approximately 400 Hz.

   Because you must connect and disconnect the external sync input during the test, the System Diagnostic Software cannot be run unattended while in the loop mode if the Phase-Locked Loop test is chosen.

5. Open Test (ch0-15)

   This test scans channels 0-15 while their inputs are open-circuited. The test checks for channel inaccuracies caused by gross faults in the channel protection circuitry.

6. Grounded Test (ch0-15)

   This test scans channels 0-15 while the respective inputs are shorted to the Analog Measurement Processor's reference input. The test checks for the accuracy of the board's a/d circuitry, for unusual channel leakage currents, and for voltage offsets.

The Solar Cell Board is required for this test. Connect the cable from the Solar Cell Board to connector J1 (the right-hand connector) on the Analog Measurement Processor and set the switch on the Solar Cell Board to L.

7. Isolation Test (ch0-15)

   This test scans the 15 legs of an external voltage divider to verify channel isolation between channels 0-15. The test verifies proper operation of the board's multiplexor circuitry.

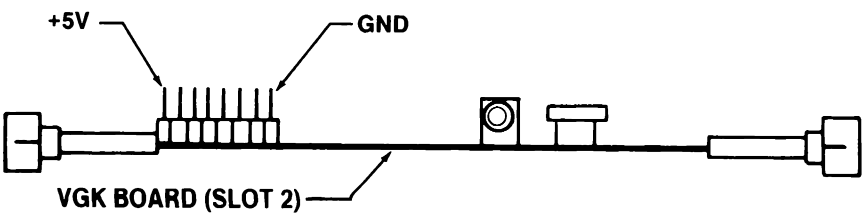
   This test requires the Solar Cell Board, which contains the 15-leg voltage divider, and a voltage input of 1-5 volts. Connect the cable from the Solar Cell Board to connector J1 on the Analog Measurement Processor (the right-hand connector), and set the switch on the Solar Cell Board to L.

CAUTION

To avoid potential damage to the Analog Measurement Processor or the 1752A, the external voltage input must not exceed 30V.

Connect a voltage input of 1-5 volts between the INPUT and GND terminals on the Solar Cell Board. The voltage input can be provided by an external voltage source, or, for convenience, it can be provided by jumpering from the +5 and GND test points on the VGK. These test points are shown in the following illustration.

VGK TEST POINTS



8. Open Test (ch16-31)

   This test performs the same checks on channels 16-31 as Open Test ch0-15 performs on channels 0-15.

9. Grounded Test (ch16-31)

   This test performs the same checks on channels 16-31 as the Ground Test ch0-15 performs on channels 0-15. Before running this test, connect the cable from the Solar Cell Board to connector J2 (the left-hand connector) on the Analog Measurement Processor and set the switch on the Solar Cell Board to L.

10. Isolation Test (ch16-31)

    This test performs the same checks on channels 16-31 as Isolation Test ch0-15 performs on channels 0-15. Follow the instructions for that test, except connect the Solar Cell Board to connector J2 (the right-hand connector) instead of J1.

11. ADTST

    ADTST allows the user to easily program and scan the input channels to the Analog Measurement Processor while applying an external input. (Refer to the 1752A Data Acquisition and Control Manual, Section 3, for programming information). This test is a menu-driven program that relies on the touch sensitive screen for input.

    While ADTST is running, the program prints 4 menu buttons along the bottom of the screen. Each button represents a different function and pressing it will bring up a different screen. The functions associated with these buttons are listed below:

    bd — allows the user to program board parameters (e.g. synchronization mode).

    chan — allows the user to program input channel parameters for the selected board. The screen also has an input readout for the selected channel.

    scan — prints the readings for all 32 input channels of the selected board. The reading are continuously updated.

    exit — exits ADTST and reloads the System Diagnostic.

Analog Output Option -011

Up to four units can be tested at once by selecting the unit numbers from the Test Menu. Each unit number corresponds to a board address as shown below. (To change a board's address, refer to Section 4 of the 1752A Data Acquisition and Control manual.)

Unit 1 = board address 0
Unit 2 = board address 1
Unit 3 = board address 2
Unit 4 = board address 3

Several of the Analog Output subtests require that you loop back (jumper) an output from the Analog Output board to an input on the Analog Measurement Processor. When making these connections, refer to Sections 3 and 4 of the 1752A Data Acquisition and Control manual for connector pinouts. To facilitate these connections, the following equipment is recommended:

1. Screw Terminal Connector (Option 2400A-110)

2. Terminal Block with Cable (Accessory Y1750)

The loopback tests require that the Analog Measurement Processor used be configured as board address 0. To change the board address of the Analog Measurement Processor, refer to Section 3 of the 1752A Data Acquisition and Control manual.

The Analog Output subtest menu includes the following tests:

1. Register R/W Test

   This test performs a read/write check on the board's registers. The test checks for stuck data bits and for proper operation of the board's bus interface logic.

   Please note that this test will output several voltages (or currents, depending on the channel's set-up) on both boards' channels. Consequently, you may want to disconnect the boards' outputs while running this test.

2. Channel 0 Voltage Loopback Test

   This test verifies the accuracy of channel 0 when it is configured as a voltage output. The test requires that you loop back the voltage output from channel 0 to an input channel on the Analog Measurement Processor with board address 0. The test program tells you which channel to use.

3. Channel 1,2,3 Voltage Loopback Tests

   These tests perform the same checks on channels 1 through 3 as Channel 0 Voltage Loopback Test performs on channel 0.

4. Channel 0 Current Loopback Test

   This test verifies the accuracy of channel 0 when it is configured as a current output. The test requires that you loop back the + current and - current outputs from channel 0 to two input channels on the Analog Measurement Processor with board address 0. The test program tells you which channel to use.

5. Channel 1,2,3 Current Loopback Tests

   These tests perform the same checks on channels 1 through 3 as Channel 0 Current Loopback Test performs on channel 0.

6. DATST

   This test output a voltage or current on any channel selected by the user. DATST is a menu-driven program that relies on the Touch-Sensitive screen for input.

Counter/Totalizer Option -012

Up to five units can be tested at once by selecting unit numbers from the Test Menu. Each unit number corresponds directly to a board address. (Unit 1 = board address 0, Unit 2 = board address 1, etc.) To change a board's address, refer to Section 3 of the 1752A Data Acquisiton and Control manual.

The Counter/Totalizer subtest menu includes the following tests:

1. Counter Chain R/W Test

   This test performs a read/write check on the board's counter chain while the board is configured as a Totalizer. The test checks for stuck data bits in the counter chain and measurement storage registers, as well as proper operation of the board's bus interface logic (e.g. bus buffers, interface timing logic, etc.)

2. Timebase Selfcheck Test

   The Timebase Selfcheck Test verifies that all types of time measurements can be made by the -012 option. This test internally loops back the signals comprising gate times 0-3. It then configures the board to take the following time measurements on these signals:

   -rising edge to rising edge
   -falling edge to rising edge using the fast rate clock
   -falling edge to rising edge using the slow rate clock

3. CTTST

   CTTST is a program that allows the user to easily set-up the Counter/Totalizer to take any kind of measurement and display the reading. It is a menu driven program relying on the Touch-Sensitive screen for input.

1760A and 1761A Disk Drive Systems

MFXTST is a program to test up to two 1760A or 1761A Disk Drive Systems. Refer to Section 9 in the 1752A System Guide or to the information supplied with the 1760A or 1761A for instructions on switch settings and connecting the unit to the 1752A. The program will check the disk drive speed and disk detection logic, then seek, format, write and read back data from a disk installed in each drive. The program will erase any files on the disk. Refer to the 1760A/1761A Manual for more information.

1765A/AB Winchester Disk Drives

WDXTST is a progam for testing the 1765A/AB Winchester Disk Drive. The test selections are displayed on the screen and are selected by touching the desired menu item. The tests include a self test of the 1765A/AB Winchester controller board and a verification of all of the blocks on the disk. If a bad block is found on the disk, that block is no longer used for storing data and an alternate block is automatically assigned.

TROUBLESHOOTING

When the System Diagnostic software encounters an error during the execution of a test, an entry is made in an error log. If STOP ON FAIL has been selected from the Main Menu, an error message is displayed on the screen. This message includes the name of the module under test and the particular subtest being executed when the error occurred.

Generally, the module under test is at fault whenever an error occurs. If an error is reported, check that the switch settings are correct on the module in question (if applicable), then run the test again. If the error condition persists, contact your local Fluke Service Center for information about replacing the faulty module. A list of replacement module part numbers is included at the end of this section.

If you are experiencing difficulty in getting the System Diagnostic software to load and execute properly, the following paragraphs provide some general guidelines for diagnosing problems with the 1752A.

WARNING

These service instructions are for use by qualified personnel only. To avoid electrical shock, do not perform any servicing other than that contained in the operating instructions unless you are qualified to do so.

Power-up Problems

When the 1752A is first powered up, the fan on the rear panel starts up, and two beeps can be heard; one from the 1752A chassis and one from the keyboard if it is plugged in. If these things do not happen, check to see that the line cord is properly installed and that the fuse is intact. Also verify that the line voltage indicated on the rear panel matches your line power source. If everything is in order, the power supply is faulty.

If the fan is running but there is no beep from the chassis, either the VGK or the PUP (power-up) assembly are faulty.

Display Problems

After the 1752A has been on for at least one minute, messages should be visible on the display. If the display is completely blank, except for a blinking cursor, try a cold start. Press the RESTART and ABORT switches on the 1752A front panel simultaneously. If the problem persists, the VGK, CRT, and video electronics are functional and the SBC is probably faulty.

If there is no display at all after a power-up or cold start, but there is a beep from the chassis, the VGK is functional, and either the CRT, video electronics, or associated cables are faulty.

Floppy Disk Drive Problems

After the 1752A has executed the power-up self-test sequence, it attempts to load the operating system (FDOS2.SYS) from the floppy disk. The message "LOADING" should be displayed on the screen, and the disk drive activity light on the front panel should come on. If the disk drive activity light does not come on, either the disk drive or associated cables are faulty.

If the disk drive activity light comes on, but the operating system does not load, an error message appears on the screen. Refer to Sections 3 and 4 of the 1752A System Guide for a description of the self-test and system error messages.

A soft error is any single failure to read a block correctly. It can be caused by improper seating of the disk or by a dust particle momentarily passing under the read/write head. An occasional soft error is of no concern. Twenty successive soft errors are considered a hard error and indicate a problem either with the disk or the disk drive.

In general, whenever a disk error occurs, the disk should be reseated by opening and reclosing the disk drive door. If the problem persists, try another disk. If this solves the problem, the original disk was faulty.

If the problem cannot be solved by using another disk, then the disk drive is either faulty or out of calibration. One other possibility is that the disk controller on the SBC is out of calibration. In either case, calibration should only be performed by authorized service personnel. Contact your local Fluke Service Center.

Keyboard Problems

During a power-up or cold start, the keyboard should emit an audible tone, and both the CAPS LOCK and PAGE MODE indicators should light momentarily. If this does not occur, the keyboard is faulty.

After the operating system is loaded, it may be necessary to use the keyboard to access some of the diagnostic programs. If the keyboard does not respond when a key is pressed, press ‹CTRL›/T to reset the keyboard and display. If this doesn't help, unplug the keyboard from the front panel and then plug it in again to perform a power-up reset of the keyboard. If the problem still persists, the keyboard is faulty.

Other Problems

Once the System Diagnostic software is loaded, if there are problems with the appearance of the menus, the VGK may be faulty. The software makes extensive use of block graphics and video attributes. If some of the reverse video blocks or highlighted characters do not appear to be correct, replace the VGK.

If the display does not respond when a menu block is touched, try other menu blocks. Keep in mind the possibility of parallax error as discussed in Section 2 of the 1752A System Guide. If the problem persists, the Touch-Sensitive Display is faulty.

If you experience other problems with the 1752A that are not discussed in this section, call your local Fluke Service Center for further assistance.

REPLACEMENT PARTS

This list gives part numbers for all replacement modules in the 1752A, as well as part numbers for the various loopback cables and connectors required for the IEEE-488 and RS-232-C tests.

DESCRIPTION	PART NUMBER
Standard Modules	
Power-Up Assembly (PUP)	704353
Power Supply	718064
Floppy Disk Drive	661629
Single-Board Computer (SBC)	767541
Video/Graphics/Keyboard Interface (VGK)	661587
CRT and Video Electronics	718056
Touch-Sensitive Overlay	705301
Keyboard	718106
Analog Measurement Processor (Option 1752A-010)	736876
Optional Modules	
Parallel Interface (Option -002)	717397
256K byte Bubble Memory (Option -004)	657346 or 777235*
512K byte Bubble Memory (Option -005)	657353 or 777391*
256K byte RAM Expansion (Option -006)	718684
512K byte RAM Expansion (Option -007)	718692
IEEE-488/RS-232C Interface (Option -008)	718221
Dual Serial Interface (Option -009)	718734
Analog Output (Option 1752A-011)	610329
Counter/Totalizer (Option 1752A-012)	630186
Mainframe Interface Assembly (Option 1752A-013)	750612

*NOTE: Order replacement from the part number listed on the module.

Miscellaneous

PIB Loopback Cable	632968
RS-232 Loopback Connector	732107
RS-232 Null Modem Cable	518696
IEEE-488 Cable (2 meter)	682401
I/O Extender Cable (9 meter)	532887
Extender Chassis (1702A)	750711
Power Cord	284174
Solar Cell Board	765206
3A Fuse (115V operation)	109199
2A Fuse (230V operation)	109173
Disk Drive Shipping Insert	707984
System Disk	759167
System Diagnostic Disk	759175
Service Manual	732156

Field Service Kit

A Field Service Kit (Fluke P/N 767533) is available which contains all the above modules. One Field Service Kit should be sufficient to support approximately ten 1752As.

The Field Kit is intended for use by qualified service personnel. Use Module Exchange as backup support.

Appendix H
Fluke Sales and Service Centers

U.S. Sales for all Fluke Products

AL, Huntsville
John Fluke Mfg. Co., Inc.
3322 S. Memorial Parkway
Suite 96
Huntsville, AL 35801
(205) 881-6220

AZ, Tempe
John Fluke Mfg. Co., Inc.
2211 S. 48th Street
Suite B
Tempe, AZ 85282
(602) 438-8314

Tucson
(602) 790-9881

CA, Irvine
P.O. Box 19676
Irvine, CA 92713
16969 Von Karman
Suite 100
Irvine, CA 92714
(714) 863-9031

Los Angeles
John Fluke Mfg. Co., Inc.
20902 South Bonita St.
Carson, CA 90746
(213) 538-3900

San Diego
John Fluke Mfg. Co., Inc.
4540 Kearny Villa Rd., Suite 115
San Diego, CA 92123
(619) 292-7656

Santa Clara
John Fluke Mfg. Co., Inc.
2300 Walsh Ave., Bldg. K
Santa Clara, CA 95051
(408) 727-0513

CO, Denver
John Fluke Mfg. Co., Inc.
1980 South Quebec St. #4
Denver, CO 80231
(303) 750-1222

CT, Hartford
John Fluke Mfg. Co., Inc.
Glen Lochen East
41-C New London Turnpike
Glastonbury, CT 06033
(203) 659-3541

FL, Orlando
John Fluke Mfg. Co.,Inc.
940 N. Fern Creek Ave.
Orlando, FL 32803
(305) 896-4881

GA, Atlanta
John Fluke Mfg. Co., Inc.
2600 Delk Rd., Suite 150
Marietta, GA 30067
(404) 953-4747

IA, Iowa City
(319) 354-2811

IL, Chicago
John Fluke Mfg. Co., Inc.
3740 Industrial Ave.
Rolling Meadows, IL 60008
(312) 398-0850

IN, Indianapolis
John Fluke Mfg. Co., Inc.
8777 Purdue Road
Suite 101
Indianapolis, IN 46268
(317) 875-7870

KS, Kansas City
(913) 381-9800

LA, New Orleans
(504) 455-0814

MA, Burlington
John Fluke Mfg. Co., Inc.
25 "B" Street
Burlington MA 01803
(617) 273-4674

MD, Baltimore
(301) 792-7060

Rockville
John Fluke Mfg. Co., Inc.
5640 Fishers Lane
Rockville, MD 20852
(301) 770-1570

MI, Detroit
John Fluke Mfg. Co., Inc.
33031 Schoolcraft
Livonia, MI 48150
(313) 522-9140

MN, Bloomington
John Fluke Mfg. Co., Inc.
1801 E. 79th St., Suite 9
Bloomington, MN 55420
(612) 854-5526

MO, St. Louis
John Fluke Mfg. Co., Inc.
2029 Woodland Parkway
Suite 105
St. Louis, MO 63141
(314) 993-3805

NC, Greensboro
John Fluke Mfg. Co., Inc.
1310 Beaman Place
Greensboro, NC 27408
(919) 273-1918

NJ, Paramus
John Fluke Mfg. Co., Inc.
P.O. Box 930
West 75 Century Road
Paramus, NJ 07652
(201) 262-9550

NM, Albuquerque
John Fluke Mfg. Co., Inc.
1108 Alvarado Drive N.E.
Albuquerque, NM 87110
(505) 881-3550

NY, Rochester
John Fluke Mfg. Co., Inc.
4515 Culver Road
Rochester, NY 14622
(716) 323-1400

OH, Cleveland
John Fluke Mfg. Co., Inc.
7830 Freeway Circle
Middleburg Heights, OH 44130
(216) 234-4540

Columbus
(614) 889-5715

Dayton
John Fluke Mfg. Co., Inc.
4756 Fishburg Rd.
Dayton, OH 45424
(513) 233-2238

OK, Tulsa
(918) 749-0190

OR, Portland
John Fluke Mfg. Co., Inc.
2700 NW 185th
Suite 2080
Portland, OR 97229
(503) 629-5928

PA, Philadelphia
John Fluke Mfg. Co., Inc.
1010 West 8th Ave., Suite H
King of Prussia, PA 19406
(215) 265-4040

Pittsburgh
(412) 261-5171

TX, Austin
(512) 459-3344

Dallas
John Fluke Mfg. Co., Inc
14400 Midway Road
Dallas, TX 75234
(214) 233-9990

Houston
John Fluke Mfg. Co., Inc
4800 Sugar Grove Blvd
Suite 150
Stafford, TX 77477

San Antonio
John Fluke Mfg. Co., Inc
10417 Gulfdale
San Antonio, TX 78216
(512) 340-2621

UT, Salt Lake City
6914 So. 3000 East
Suite 206
Salt Lake City, UT 82021
(801) 268-9331

WA, Seattle
John Fluke Mfg. Co., Inc
5020 148th Ave. N.E
Suite 110
Redmond, WA 98052
(206) 881-6966

Service Center Areas

CA, Burbank (213) 849-4641
CA, Santa Clara (408) 727-8121
CO, Denver (303) 750-1228
FL, Orlando (305) 896-2296
IL, Chicago (312) 398-5880
MA, Burlington (617) 273-4678
MD, Rockville (301) 770-1576
NJ, Paramus (201) 262-9550
TX, Dallas (214) 233-9945
WA, Everett (206) 356-5560

Argentina ●
Coasin S A
Virrey del Pino 4071 DEP E-1
1430 CAP FED
Buenos Aires, Argentina
Tel (1) 552-5248. TLX (390) 22284

Australia ●
Elmeasco Instruments Pty. Ltd
P.O Box 30
Concord, N S W 2137
Australia
Tel (2) 736-2888. TLX (790) AA25887

Elmeasco Instruments Pty. Ltd
P.O Box 623
12 Maroondah Highway
Ringwood, Victoria 3134
Australia
Tel (3) 879-2322. TLX (790) AA36206

Elmeasco Instruments Pty. Ltd
Professional Suites Bldg
G.P O Box 2360
Brisbane 4001, Australia
Tel (7) 369-8688. TLX (790) AA44062

Elmeasco Instruments Pty Ltd
G.P O Box 1240
Adelaide, South Australia 5001
Tel (8) 271-1839

Elmeasco Instruments Pty. Ltd
P O Box 95
Gosnells, West Australia 6110
Australia
Tel (9) 398-3362

Austria ■
Walter Rekirsch Elektronische
Gerate GmbH & Co
Vertrieb KG
Obachgasse 28
1220 Vienna, Austria
Tel (222) 235555. TLX (847) 134759

Bahrain ■
Basma
P O Box 5701
Manama, Bahrain
Tel 973-251364. TLX (955) 9003

Bangladesh ●
Motherland Corporation
24 Hatkhola Road, Tikatuli
Dacca-3, Bangladesh
Tel 257249 or 255776
TLX (950) 642022

Belgium ■
Fluke (Belgium) S.A /N V
6 Rue de Geneve
1140 Brussels
Belgium
Tel (2) 2164090. TLX (846) 26312

Bolivia ●
Coasin Bolivia S R L
Casilla 7295
La Paz, Bolivia
Tel (2) 40962. TLX (336) 3233

Brazil ●
Fluke Brasil-Industria e
Comercio Ltda
Al Amazonas 422,
Alphaville, Barueri,
CEP 06400, Sao Paulo, Brazil
Tel (11) 421-5477. TLX (391) 01135589

Fluke Brasil-Industria e
Comercio Ltda
Av Henrique Valadares, No 23/401
Rio de Janeiro, Brazil
Tel (21) 252-1297

Brunei ●
Rank O'Connor's Berhad
No. 8 Blk D
Sufri Shophouse Complex
Mile 1 Jalan Tutong
Bandar Seri Begawan
Negara Brunei Darussalam
Tel (2) 23109 or 23557
TLX (799) 2265

Canada ●
Allan Crawford Assoc. Ltd
5835 Coopers Avenue
Mississauga, Ontario
L4Z 1Y2 Canada
Tel (416) 890-2010. TLX 06961235

Allan Crawford Assoc. Ltd
7018 Cote de Liesse
St Laurent, Quebec
H4T 1E7 Canada
Tel (514) 731-8564
TLX 05824944

Allan Crawford Assoc. Ltd
881 Lady Ellen Place
Ottawa, Ontario
K1Z 5L3 Canada
Tel (613) 722-7682. TLX 0533600

Allan Crawford Assoc. Ltd
Suite No 106
4180 Lougheed Hwy
Burnaby, British Columbia
V5C 6A7 Canada
Tel (604) 294-1326. TLX 0454247

Allan Crawford Assoc., Ltd
1935 30th Avenue, N E
Calgary, Alberta
T2E 6Z5 Canada
Tel (403) 230-1341. TLX 03821186

Allan Crawford Assoc. Ltd
800 Windmill Road
Suite 205
Dartmouth, N S
B3B 1L1 Canada
Tel (902) 463-8640

Chile ●
Intronica Chile, Ltda
Manuel Montt 024-Of D
Casilla 16228
Santiago 9, Chile
Tel (2) 44940. TLX (332) 240301

China ●
Contact Roland Chua
Regional Sales Manager
Fluke Park x5511

Colombia ●
Sistemas E Instrumentacion, Ltda
Carrera 13, No 37-43, Of 401
Ap Aereo 29583
Bogota DE, Colombia
Tel 232-4532. TLX (396) 45787

Cyprus ■
Chris Radiovision, Ltd
P.O Box 1989
Nicosia, Cyprus
Tel (21) 66121. TLX (826) 2395

Cyprus, Northern
Ucok Buroteknik
2C & 2D Muftu Ziyai Street
Lefkosa, Northern Cyprus
Mersin 10, Turkey
Tel (741) 357-20-71777
TLX (821) 57267

Denmark ■
Tage Olsen A/S
Ballerup Byvej 222
2750 Ballerup
Denmark
Tel (2) 658111. TLX (855) 3593

Ecuador ●
Proteco Coasin Cia. Ltda
P.O Box 228-A
Ave 12 de Octubre 2285
y Orellana
Quito, Ecuador
Tel (2) 529684. TLX (393) 2865

Proteco Coasin Cia Ltda
P.O Box 9733
Ave Principal No 204
y Calle Segunda
Urbanizacion Miraflores
Guayaquil, Ecuador
Tel (4) 387519

Egypt and Sudan ■
Electronic Engineering Liaison Office
P.O Box 2891 Horreya
Heliopolis, Cairo
Egypt
Tel (2) 695705. TLX (927) 22782

England ■
Fluke GB, Ltd
Colonial Way
Watford, Herts.
WD2 4TT England
Tel (923) 40511. TLX (851) 934583
Rapifax 44-923-25067

Fluke Northern Service Centre
Middle Floor
Mersey House
220 Stockport Road
Stockport, Chesire
England
Tel (606) 491-2391

Eastern European Countries ■
Amtest Associates, Ltd
Clarence House
31 Clarence Street
Staines, Middlesex TW18 4SY
England
Tel (784) 63555. TLX (851) 928855

Fiji ●
Awa Fiji
47 Forster Road
Walu Bay
Suva, Fiji
Tel 312079. TLX (792) FJ2347

Finland ■
Instrumentarium Elektroniikki oy
P.O Box 64
02631 Espoo 63
Finland
Tel (0) 5281. TLX (857) 124426

International Sales Offices

France ■
M.B. Electronique S.A
606, Rue Fourney
P.O. Box 31
78530 BUC, France
Tel. (3) 956-8131, TLX: (842) 695414

German Democratic Republic ■
Amtest Associates Ltd.
Clarence House, 31 Clarence St
Staines, Middlesex TW18 4SY
United Kingdom
Tel: (784) 63555, TLX: (851) 928855

Germany, West ■
Fluke (Deutschland) GmbH
Postfach 11 53
8045 Ismaning/Munich
West Germany
Tel: (89) 96050, TLX (841) 0522472

Fluke (Deutschland) GmbH
Viertriebsburo - Dusseldorf
Meineckestrasse 53, 4000 Dusseldorf-30
West Germany
Tel: (211) 450831, TLX (841) 17-2114233

Fluke (Deutschland) GmbH
Vertriebsburo - Hamburg
Habichthorst 42
2000 Hamburg 42
West Germany
Tel: (40) 5519031, TLX (841) 02174556

Greece ■
Hellenic Scientific Representations Ltd
11, Vrassida Street
Athens 612, Greece
Tel: (1) 711140, TLX (863) 219330

Hong Kong ●
Schmidt & Co (H.K.), Ltd
18th Floor, Great Eagle Centre
23 Harbour Road
Wanchai, Hong Kong
Tel: (5) 8330-222
TLX: (780) 74766 or (780) 76762

Hungary
Amtest Associates Ltd
Clarence House, 31 Clarence St
Staines, Middlesex TW18 4SY
United Kingdom
Tel: (784) 63555, TLX (851) 928855

India ●
Hinditron Services Pvt., Ltd
69/A L. Jagmohandas Marg
Bombay 400 006, India
Tel: (22) 8121316, 8125344, TLX (953) 112326

Hinditron Services Pvt., Ltd
8th Main Road
33/44A Raj Mahal Vilas Extension
Bangalore 560 080, India
Tel: (910) 33139, 367289, TLX (953) 845741

Hinditron Services Pvt. Ltd
Shantiniketan, Office No. 6
6th Floor, 8 Camac Street
Calcutta 700 017, India
Tel: (910) 033-434032, 447541; TLX 214153

Hinditron Services Pvt. Ltd
204-5-6 Hemkunt Tower
98 Nehru Place
New Delhi, 110019, India
Tel: (910) 640380, TLX. (953) 3161458

Hinditron Services Pvt., Ltd
15, Community Centre
Panchshila Park
New Delhi 110 017, India
Tel: 011-661869

Hinditron Services Pvt. Ltd
Srinath Complex, 5th Floor
1-1-58/1 to 1-1-58/11
Sarojini Devi Road
Secunderabad 500 003, India
Tel: (842) 821117, 823751, TLX: (953) 155-333

Indonesia ●
P.T. Dwi Tunggal Jaya Sakti
P.O. Box 4435
Jalan Panglima Polim Raya #29
Kebayoran Baru
Jakarta Selatan, Indonesia
Tel: (21) 734512, TLX (796) 47308

Iran ■
Arma Baynelmeleli Co., Ltd
P.O. Box 19575/473
Tehran
Tel: (21) 248717, TLX (951) 213648

Ireland ■
Euro Instruments & Electronics Ltd
Euro House
Swords Road, Santry
Dublin 9, Ireland
Tel: (1) 425666, TLX (851) 31821

Israel ■
R.D.T. Electronics Engineering Ltd
P.O. Box 43137
Tel Aviv 61430
Israel
Tel: (3) 483211, TLX (922) 371452

Italy ■
Sistrel S.p.A
Via Pelizza da Volpedo 59
20092 Cinisello Balsamo
Milan, Italy
Tel: (2) 6181893, TLX (843) 334643

Sistrel S.p.A.
Via Giuseppe Armellini No. 39
00143 Rome, Italy
Tel: (6) 591-5551, TLX (843) 680356

Sistrel S.p.A
Via Cinta
Parco S. Paolo 35
80126 Naples, Italy
Tel: (81) 7679700

Japan ●
John Fluke Mfg. Co., Inc
Japan Branch
Sumitomo Higashi Shinbashi Bldg
1-1-11 Hamamatsucho
Minato-ku, Tokyo 105, Japan
Tel: (3) 434-0181, TLX (781) 2424331
FAX: 81-3-434-0170

John Fluke Mfg. Co., Inc
Japan Branch
Katsushige Building
2-45 Kohraibashi
Higashi-ku, Osaka 541
Japan
Tel: (6) 229-0871

Korea ●
Myoung Corporation
Rm #1501, Sinsong Bldg
25-4 Yeouido-Dong, Young Deung Po-Ku
Seoul 150, Korea
Tel: (2) 784-9942, TLX K24283

Kuwait ■
Al Bahar International Group
P.O. Box 26672 Safat
Kuwait, Arabian Gulf
Tel: 848601, 847598, TLX (959) 44822

Lebanon and Jordan ■
Mabek (Electronics Division)
P.O. Box 13-5657
Beirut, Lebanon
Tel: 812523, 812538, TLX (923) 22889

Malaysia ●
Mecomb Malaysia SND BHD
Lot 20, Jalan 225
P.O. Box 24
Petaling Jaya, Malaysia
Tel: (3) 743422, TLX (784) MA37764

Malta ■
Fabian Enterprises
20, Msida Road
Gzira, Malta
Tel: 513283/40216, TLX (838) 1837

Mexico ●
Mexicana de Electronica
Industrial, S.A. (Mexel)
Diagonal No. 27
Entre Calle de Eugenia Y Ave
Colonia del Valle
C.P. 03100, Mexico
Tel: (905) 687-2099, TLX (383) 1771038

Morocco ■
Oussama S.A
Angle Boulevard Emile Zola et
Rue Rethel
P.O. Box 2007 Casa
Casablanca
Morocco
Tel: 212-24-13-38, TLX 22730 M

Nepal ●
Associated Enterprises
GPO Box 790, Pyaphal Tole
Kathmandu, Nepal
Tel: 13868

Netherlands ■
Fluke (Holland) B.V
P.O. Box 2269
5600 CG Eindhoven
Netherlands
Tel: (40) 458045, TLX (844) 51846
FAX: 31-40-457515

Fluke (Nederland) B.V
Gasthuisring 14
P.O. Box 115
5000 AC Tilburg
The Netherlands
Tel: (13) 352455, TLX (844) 52683

New Zealand ●
McLean Information Technology, Ltd
459 Khyber Pass Road, Newmarket
P.O. Box 9464, Newmarket
Auckland 1, New Zealand
Tel: (9) 501-801, TLX: (791) NZ 21570

McLean Information Technology, Ltd
P.O. Box 496
Wellington, New Zealand
Tel: (4) 851-450, 844-424 TLX: (791) NZ3303

Norway ■
Morgenstierne & Co A/S
Konghellegate 3
P.O. Box 6688, Rodelokka
Oslo 5, Norway
Tel: (2) 356110, TLX: (856) 71719

Oman ■
OHI Telecommunications
P.O. Box 889
Muscat
Sultanate of Oman
Tel: (968) 703862, TLX: (926) 3168

Pakistan ●
International Operations (PAK), Ltd
505 Muhammadi House
I.I. Chundrigar Road
P.O. Box 5323, Karachi, Pakistan
Tel: (21) 221127, 239052, 221133
TLX: (952) 24494

Peru ●
Importaciones y Representaciones
Electronicas S.A.
Avda. Franklin D. Roosevelt 105
Lima 1, Peru
Tel: (14) 28-8650, TLX: (394) 25663

Philippines, Republic of ●
Spark Radio & Electronics, Inc
Greenhills P.O. Box 610
San Juan, Metro Maila, Zip 3113
Philippines
Tel: (2) 775192, 704096, 704097
TLX: (722 or 732) 27901

Poland
Amtest Associates Ltd
Clarence House, 31 Clarence St
Staines, Middlesex TW18 4SY
United Kingdom
Tel: (784) 63555, TLX: (851) 928855

Portugal ■
Decada Espectral
Equipamentos de Electronica
Av. Bombeiros Voluntarios
Lote 102B, Miraflores/Alges
1495 Lisbon, Portugal
Tel: (1) 2103420, TLX: (832) 15515

Romania
Amtest Associates Ltd
Clarence House, 31 Clarence St
Staines, Middlesex TW18 4SY
United Kingdom
Tel: (784) 63555, TLX: (851) 928855

Singapore, Republic of ●
Rank O'Connor's (PTE) Ltd
O'Connor House
98 Pasir Panjang Road
Singapore 0511
Republic of Singapore
Tel: 4737944, TLX: (786) RS21023

South Africa ●
Fluke S.A. (Pty) Ltd
P.O. Box 39797
Bramley 2018
Republic of South Africa
Tel: (11) 786-3170, TLX: (960) 424328
FAX: 27-11-786-5304

Spain ■
ESSA
Equipos y Sistemas S.A.
C/Apolonio Morales, 13-B
Madrid 16, Spain
Tel: (1) 458-0150, TLX: (831) 42856

Sri Lanka ●
Computerlink Data Systems, Ltd
294 Union Place
Colombo 2, Sri Lanka
Tel: (1) 28641/2 or 54605/6,
TLX: (954) 21321

Sweden ■
Teleinstrument AB
Maltesholmsvagen 138
P.O. Box 4490
162 04 Vallingby 4
Sweden
Tel: (8) 380370, TLX: (854) 15770

Switzerland ■
Traco Electronic AG
Jenatschstrasse 1
8002 Zurich
Switzerland
Tel: (1) 201-0711, TLX: (845) 815570

Syria ■
Mabek (Electronics Division)
P.O. Box 4238
Damascus, Syria

Taiwan ●
Schmidt Electronics Corp
5th Fl, Cathay Min Sheng
Commercial Building,
344 Min Sheng E. Road, Taipei
Taiwan, R.O.C.
Tel: (2) 562-2132, TLX: (785) 11111

Thailand ●
Measuretronix Ltd
2102/63 Ramkamhaeng Rd
Bangkok 10240
Thailand
Tel: (2) 378-2516, 378-0632
TLX: (788) 81143

Tunisia ■
Selep S.A.R.L
6, Rue de Sparte
Tunis - 1000 RP
Tunisia
Tel: (1) 248093, TLX: (934) 13030

Turkey ■
Erkman Elektronik Aletler
Ticaret Anonim Sirketi
Necatibey Cad 92/3
Karakoy, Istanbul, Turkey
Tel: (11) 441546, 447651
TLX: (821) 24399

United Arab Emirates ■
Haris Al-Afag, Ltd
P.O. Box 545
Sharjah, U.A.E.
Tel: (6) 359120, TLX: (958) 68540

Uruguay ●
Coasin Uruguaya S.A.
Libertad 2529
Casilla de Correo 1400
Montevideo, Uruguay
Tel: (2) 789015, 789204
TLX: (398) UY23010

USSR
Amtest Associates Ltd
Clarence House, 31 Clarence St
Staines, Middlesex TW18 4SY
United Kingdom
Tel: (784) 63555, TLX: (851) 928855

Venezuela ●
Coasin C.A.
Calle 9 Con Calle 4, Edif Edinurbi
Apartado de Correos NR-70 136
Los Ruices
Caracas 1070-A, Venezuela
Tel: (2) 239-0967, TLX: (395) 21027

Yugoslavia ■
Amtest Associates Ltd
Clarence House, 31, Clarence St.
Staines, Middlesex TW18 4SY
United Kingdom
Tel: (784) 63555, TLX: (851) 928855

■ Supplied and Supported by —
Fluke (Holland) B.V.
P.O. Box 2269
5600 CG Eindhoven
Netherlands
Tel: 31-40-45805, TLX: (844) 51846
FAX: 31-40-457515

● Supplied and Supported by —
Fluke International Corporation
P.O. Box C9090
Everett, WA 98206 U.S.A.
Tel: (206) 356-5500
TLX: 152662 JOHN FLUKE EVT
FAX: 206-356-5116

The following countries are represented by
Fluke (Holland) B.V.
P.O. Box 2269
5600 CG Eindhoven
Netherlands
Tel: 31-40-45805, TLX: (844) 51846
FAX: 31-40-457515

Abu Dhabi
Afghanistan
Albania
Algeria
Angola
Benin
Bornholm
Botswana
Chad
Corsica
Czechoslavakia
Dubai
Ethiopia
Guinea
Ibiza
Iceland
Iraq
Kuwait
Liberia
Libya
Luxembourg
Maderia
Mallorca
Mauritania
Menorca
Mongolia
Niger
Nigeria
Qatar
Rodhos
Russia
Sardinia
Saudia Arabia
Scotland
Senegal
Sierra Leone
Somalia
Togo
Upper Volta
Wales
Wstn Sahara
Yemen
Zaire
Zambia
Zimbabwe

U.S.A.

CA, Burbank
John Fluke Mfg Co Inc
(213) 849-4641

CA, Santa Clara
John Fluke Mfg Co Inc
(408) 727-8121

CO, Denver
John Fluke Mfg Co Inc
(303) 750-1228

FL, Orlando
John Fluke Mfg Co Inc
(305) 896-2296

IL, Rolling Meadows
John Fluke Mfg Co., Inc
(312) 398-5800

MA, Burlington
John Fluke Mfg Co Inc
(617) 273-4678

MD, Rockville
John Fluke Mfg Co Inc
(301) 770-1576

NJ, Paramus
John Fluke Mfg Co Inc
(201) 262-9550

TX, Dallas
John Fluke Mfg Co Inc
(214) 233-9945

WA, Everett
John Fluke Mfg Co Inc
(206) 356-5560

Other Countries

Argentina, Buenos Aires
Coasin S A
Tel 552-5248/3485
TLX 22284 COASN AR

Australia, Concord
Elmeasco Instruments Pty Ltd
Tel 736-2888
TLX 25887 ELSCO

Australia, Mount Waverley
Elmeasco Instruments Pty Ltd
Tel 233-4044
TLX 36206 ELMVIC

Australia, Brisbane
Elmeasco Instruments Pty Ltd
Tel 369-8688
TLX 44062 ELMQLD

Austria, Vienna
Walter Rekirsch
Elektronische Gerate GmbH & Co
Tel 235555
TLX 134759

Belgium, Brussels
Fluke (Belgium) SA/NA
Tel 2164090
TLX 26312

Brazil, Sao Paulo
Fluke Brasil-Industria E Comercio Ltda
Tel 421-5477
TLX 1135589 FLKE BR

Canada, Calgary, AB
Allan Crawford Associates Ltd
Tel (403) 230-1341

Canada, Burnaby, BC
Allan Crawford Associates Ltd
Tel (604) 294-1326

Canada, Mississauga, ON
Allan Crawford Associates Ltd
Tel (416) 678-1500

Canada, St. Laurent, PQ
Allan Crawford Associates Ltd
Tel (514) 731-8564

Chile, Santiago
Intronica Chile Ltda
Tel 44940
TLX 240301

China, Beijing
Beijing Radio Research Institute
Tel 445612

Colombia, Bogota
Sistemas E Instrumentacion Ltda
Tel 232-4532
TLX 45787 COASN CO

Denmark, Ballerup
Tage Olson A/S
Tel 658111
TLX 35293 TOAS DK

Ecuador, Quito
Proteco Coasin Cia., Ltda
Tel 529684, 526759
TLX 2865 PROTEC ED

Egypt, Cairo
Electronic Engineering Liaison Office
Tel 695705
TLX 23082

England, Watford, Herts
Fluke (Great Britain) LTD
Tel 40511
TLX 934583

Finland, Espoo
Instrumentarium Electronics
Tel 5281
TLX 124426 HAVUL SF

France, Paris
M B Electronique S A
Tel 9568131
TLX 695414

Greece, Athens
Hellenic Scientific Representations
Tel 7211140
TLX 219330

Hong Kong, Wanchai
Schmidt & Co (H.K.) Ltd
Tel 8330-222
TLX 74766 SCHMC HX

India, Bombay
Hinditron Services Pvt Ltd
Tel 8121316
TLX 112326 HSPL IN

India, Bangalore
Hinditron Services Pvt Ltd
Tel 33139
TLX 0845741 HSPL IN

India, New Delhi
Hinditron Services Pvt Ltd
Tel 619118
TLX 0314890 SRMP IN

Indonesia, Jakarta Selatan
P T Dwi Tunggal Jaya Sakti
Tel 716374
TLX 47308 DIJS IA

Israel, Ramat Hasharon
R.D.T Electronics Engineering Ltd
Tel 483216
TLX 32143 RDT IL

Italy, Milan
Sistrel S.p A
Tel 6181893
TLX 334643

Italy, Rome
Sistrel S.p.A
Tel 5915551
TLX 680356

Japan, Tokyo
John Fluke Mfg Co., Inc
Japan Branch
Tel 434-0185
TLX 2424331 (FLUKJPJ)

Korea, Seoul
Electro-Science Korea Co
Tel 583-7703
TLX 25381 ESKOREA

Malaysia, Petaling Jaya
Mecomb Malaysia SDN BHD
Tel 743422
TLX MA37764 MECOMB

Mexico, Mexico D.F.
Electronica y Tecnologia
Avanzada S A de C V (ETA)
Tel 393 09 02 or 393 57 62
TLX 172697 BIOSME

Netherlands, Tilburg
Fluke (Nederland) B.V
Tel: 352455
TLX 52683

New Zealand, Auckland
McLean Information Technology, Ltd
Tel 501-801, 501-219
TLX NZ21570 THERMAL

Norway, Oslo
Morgenstierne & Co A/S
Tel: 356110
TLX 71719

Pakistan, Karachi
International Operations (PAK), Ltd
Tel: 221127, 239052
TLX 24494 PIO PK

Peru, Lima
Importaciones Y Representaciones
Electronicas S.A.
Tel: 288650
TLX: 25663 PE IREING

Philippines, Metro Manila
Spark Radio & Electronics Corp
Tel: 775192, 704096
TLX: 27901 RLA PH

Portugal, Lisbon
Decada Espectral Equipamentos
de Electronica, Lda
Tel 2103420
TLX 15515

Republic of Singapore, Singapore
Rank O'Connor's (PTE) Limited
Tel: 637944, 239052
TLX: RS21023 OCONSIN

Republic of South Africa, Wynberg
Fluke S.A. (Pty) Ltd.
Tel: 786-3170
TLX 424328

Spain, Madrid
Equipos y Sistemas S.A
Tel: 458-0150
TLX 42856 EYS E

Sweden, Vallingby
Teleinstrument AB
Tel: 380370
TLX 15770

Switzerland, Zurich
Traco Electronic AG
Tel: 2010711
TLX 815570 TRCO CH

Taiwan, Taipei
Schmidt Electronics Corp
Tel 5013468
TLX: 11111 SCHMIDT
10548 EVERGOFT

Thailand, Bangkok
Measuretronix Ltd
Tel 378-2516
TLX 81143 DEJOBKK TH

Turkey, Istanbul
Erkman Elektronik Aletler
Tel: 4415461
TLX 24399

Uruguay, Montevideo
Coasin Uruguaya S.A
Tel 789015
TLX: UY6571 OROCUER SA

Venezuela, Caracas
Coasin C.A
Tel: 239-0967, 239-5079
TLX 21027 EMVEN VE

West Germany, Ismaning/Munich
Fluke (Deutschland) GmbH
Tel: 96050
TLX 0522472

Index

ABORT, **7-7**

Accessories, **9-5**

Alias file
- Creating Aliases, **7-9**
- Defined, **4-9, 7-9**
- Standard Aliases, **7-11**

Analog Measurement Processor (1752A-010), **2-8, 9-55**

Analog Output (1752A-011), **2-8, 9-63**

BASIC, **2-4**
- Compiler Program, **6-3**
- Data acquisition and control, **2-5**
- Entry from FDOS, **3-8**
- Environment (SET SHELL), **7-8**
- Interpreter Program, **6-3**
- Prompt, **3-8**

Baud Rate
- Changing, **5-9, 5-21**
- Default, **5-21**

Bootstrap Loader, **3-3**

Character Graphics, **8-4, 8-6**

Character EPROM, **E-1**

Cold Start, **4-3**

Command Files
- Creating, **7-3**
- Defined, **4-9, 7-3**
- Startup Command File, **3-6, 4-9, 7-6, 7-32**

Command Line Interpreter
- Defined, **3-10**
- Editing Features of, **6-6**
- Special Characters in Command Files, **7-4**

Commands
- Command File, **7-4**
- Edit Program, **6-8 - 6-47**
- FDOS, **6-5**
- File Utility Program, **4-11, 4-18 - 4-31**
- Set Utility Program, **5-17, 5-21 - 5-25**

Communications
- IEEE-488, **5-3 (Sample Program, 5-13)**
- RS-232, **5-16 (Sample Program, 5-28)**

Configuration
- Illustrated, **2-15**
- Table, **9-7**

Connecting
- IEEE-488 Bus, **2-19, 5-8**
- RS-232 Cable, **5-31**

Connector, IEEE-488 Pinout **C-1**

Controller in Charge (switch), **2-14, 5-9**

Copying files, **4-39**

Counter/Totalizer (1752A-012), **2-8, 9-69**

Data Acquisition and Control
- Installing system, **2-26**
- Software, **2-5**
- *Also see* Measurement and Control

Definitions, **D-1**

Devices
- Defined, **4-3**
- Electronic Disk, **4-8**
- IEEE-488, **4-7**
- Mini-Floppy, **4-7**
- Optional, **4-4**
- RS-232, **4-6**
- Standard, **4-4**
- System Device, **4-3**
- Winchester Drive, **4-8**

Diagnostic Software, **G-1**

Display (Section 8)
  Characters, **8-3 - 8-17**
  Graphics, **8-18**
  Planes, **8-3, 8-18**
  Printing, **8-40**

Edit Program (System Editor)
  Command Mode, **6-17**
  Control Commands, **6-35**
  Error Messages, **6-47**
  Essential Commands, **6-12**
  Global Commands, **6-39**
  Insertion Mode, **6-13 (table, 6-16)**
  Marker Commands, **6-29**
  Search Commands, **6-27**
  Target Commands, **6-37**
  Yank Buffer, **6-17**

Editor Program (BASIC), **6-2**

Electronic Disk (E-Disk), **4-8**

Environment, Physical
  Floppy Disks, **2-14**
  Specifications, **A-1**

Environment, Programming
  SET SHELL Statement, **7-8**

Error Messages - *see* System Messages

Extensions (file types), **4-5**

FDOS, **2-4**
  Commands, **6-5**
  Prompt, **2-19**

Features, **2-6**

Filenames, **1-9, 4-14**

File Types (*also see* extensions)
  Alias File, **7-9**
  Command Files, **7-3**

File Utility Program (Section 4)
  Automating, **7-17**
  Commands
    Described, **4-18**
    Syntax, **4-32**
    Table, **4-12**
  Entering, **4-11**
  Exiting, **4-30**
  Help Command, **4-11**
  Messages, **4-36**
  Syntax Diagrams, **4-32**

Fuse, **2-11, A-2**

Glossary, **D-1**

Graphics
  Character Graphics, **8-4, 8-7**
  Link to Object File, **8-18**
  Plane, **8-18**
  Printing, **8-40**
  Routines, **8-22 - 8-39**

Hardware
  Configuration, **2-9**
  Options, **9-1**
  Physical Layout, **2-10**

IEEE-488
  Address Switch, **2-20, 5-9**
  Automating a System, **7-18**
  Connector Pinout, **C-1**
  Defined, **5-3**
  References, **C-1**
  Sample Program, **7-29**
  Setting up a System, **2-28, 7-18**

Installation (Section 2)
  Bench Mounting, **2-22**
  Data Acquisition and Control System, **2-26**
  Rack Mounting, **2-24**
  Sample System, **2-27**

Keyboard, **2-12, 2-24**

Manual Usage, Section 1

Measurement and Control Modules
- About, **2-8**
- Configuring, **2-21**
- Installing, **2-26**
- Programming, **2-5**

Multiple Controller Systems, **2-20, 5-9**

Notation
- Keyboard Inputs, **1-10**
- Syntax Diagrams, **1-8**

Operating System
- Command Line Interpreter, **3-10, 6-5**
- How to Create New System Disks, **3-12**

Options, Section 9

Parallax Error, **2-22**

Parallel Interface Board
- (17XXA-002), **2-8, 9-13**

Pathname, **4-14**

Peripherals, **9-4**

Physical Layout
- Front Panel, **2-11**
- Interior, **2-10**
- Rear Panel, **2-11**

Power
- Input, **2-11**
- Specifications, **A-2**
- Switch, **2-11**

Programming
- Command Files, **7-3**
- Communications, **5-12, 5-27**
- Sample System, **7-18**
- Selecting a Language, **6-3**

Prompt
- BASIC, **3-8**
- FDOS, **2-19**
- File Utility Program, **4-11**
- Set Utility Program, **5-17**
- Time and Date Utility Program, **3-7**

Rack Mounting, **2-24**

RESTART, **3-3, 3-9, 4-3**

RS-232, **5-16, B-1**

Sample Programs
- IEEE-488, **5-13**
- RS-232, **5-28**
- System, **7-29**

Self Test, **3-3**

Serial Port
- Setting Parameters, **5-17**

Setting
- IEEE-488 Bus Address, **5-9**
- The Time, **3-7**
- RS-232 Port Parameters, **5-21**

Software, **3-9, 6-3**
- Optional Languages, **B-2**

Startup Command File, **3-6**

Static, **2-14**

Syntax Diagrams
- Defined, **1-8**
- How to Read, **1-8**
- File Utility Program, **4-32 - 4-35**
- Set Utility Program, **5-20**
- Creating and Using Aliases, **7-9, 7-10**

System Controller (switch), **5-9**

System Generation Utility Program, **3-14**

System Messages
Alias File, **7-11**
File Utility Program, **4-36 - 4-38**
Self-test, **3-4**
Set Utility Program, **5-26**
System Editor Program, **6-47**

TCOPY, **4-39**

Time and Date Utility Program
Setting the Time, **3-7**
Use in Command Files, **7-15**
Use in Programs, **7-15**

Touch Sensitive Display, Section 8, **8-7**

24-Hour Format, **3-7**

Unpacking, **2-12**

Utility Programs
File Utility, **4-11, 7-17**
Set Utility, **5-16, 7-17**
System Generation Utility, **3-12**
Time and Date Utility, **3-7, 7-15**

Warm Start, **4-3**

Winchester Device, **4-8**

Workbench Installation, **2-24**

WELCOME TO PROCEDURE WRITING

This Introductory Training Manual, along with your 7404B Procedure Writing Manual, is your introduction to the world of procedure writing. Quality in your 7404B calibration procedures is the key to quality in your instrument calibrations. Your knowledge of procedure writing will unleash the power of your 7404B Computer-Aided Calibration Cluster, giving you the greatest flexibility in the procedure tests performed.

OBJECTIVE OF THIS MANUAL

The objectives of this Introductory Training Manual are to teach the student the basics of procedure writing, to give the student the tools necessary to write his own calibration procedures, and to show the student where he can find information on the details of procedure writing.

HOW TO USE THIS MANUAL

The 7404B Introductory Training Manual should be used in conjunction with the 7404B Procedure Writing Manual. This Training Manual gives you a general introduction to procedure writing, descriptions of your most often used Function Selection Codes, exercises dealing with these FSCs, and answers to the exercises. For detailed descriptions of specific points, you will be referred to the 7404B Procedure Writing Manual.

The following is a list of sections in this manual and what each section covers:

Section 1 Lessons

Contains an introduction to the manual, suggestions on where to look for information to get you started writing procedures, and five procedure writing lessons.

Section 2 Exercises

Contains five procedure writing exercises which should be performed in turn after reading the associated lessons.

Section 3 Answers

Contains the answers to the associated exercises. These answers should be viewed only after completing the exercises.

SYSTEM DOCUMENTATION

The 7404B System is supported primarily by three manuals, the 7404B Operator's Manual, the 7404B Procedure Writing Manual, and the 7404B Introductory Training Manual. These three manuals are bound together in the 7404B Manual Set binder. In addition, a quick reference card and the individual Instruction Manuals of the System's component instruments are provided to document the 7404B System. The following paragraphs describe the other manuals in the 7404B Manual Set.

7404B Operators Manual

The 7404B Operators Manual describes how to operate the 7404B Computer Aided Calibration Cluster (referred to in this manual as the 7404B or System). It includes examples and a step-by-step tutorial describing how to use the 7404B System to calibrate an instrument and handle data. This manual is intended for individuals who will operate the system.

7404B Procedure Writing Manual

The 7404B Procedure Writing Manual explains how to write or modify calibration procedures. The manual is intended for technicians or engineers who have a working knowledge of instrument calibration. However, writing and modifying procedures does not require programming experience.

7404B Introductory Training Manual

The 7404B Introductory Training Manual is a good place for a person with little or no background in procedure writing to start. In the 7404B Introductory Training Manual you will find FSC descriptions, exercises dealing wit FSCs, and answers to the exercises. use the 7404B Introductory Training Manual to develop or enhance your procedure-writing skills.

7404B Procedure Writing Quick Reference Card

The 7404B Procedure Writing Quick Reference Card summarizes the essential information contained in the Procedure Writing Manual.

Reference Documentation

The 7404B Operators Manual contains all the information necessary for normal operation of the 7404B System. For more detailed information about a specific instrument in the System, please refer to one of the manuals listed below. These manuals contain information required for troubleshooting and provide the sophisticated user with the detailed information necessary to meet unusual needs and requirements.

- 1722A Instrument Controller System Guide
- 5100B Calibrator Instruction Manual
- 8506A Digital Multimeter Instruction Manual
- 5205A Precision Power Amplifier Instruction Manual (Option)
- 5220A Transconductance Amplifier Instruction Manual (Option)

WHAT IS A PROCEDURE?

A procedure is a data file which defines the tests to be executed when calibrating an instrument. When you write a procedure you supply the system with this data information. The calibration tests entered into the calibration procedure are defined using a specialized "computer program," designed for the meter calibration environment. This is known as the 7404B procedure generator/editor software and is found on the 7404B System Software disk. For additional information on calibration procedures, turn to page 1-4 of the 7404B Procedure Writing Manual.

PROCEDURE FORMAT

Procedures are structured data files. All information in a procedure is placed according to a set format.

Procedures all begin with the same heading format. When you create a new procedure with the Procedure Generator Editor, this heading is automatically available for you to supply the necessary information. After the heading, each procedure has 10 columns (fields). These fields require different information for each function you want the system to perform. However, for most system functions you only need to supply up to three pieces of data.

If you turn to page 1-7 in the 7404B Procedure Writing Manual, you can see the heading and fields of a procedure.

PROCEDURE HEADING

As far as the heading goes, the only information the procedure writer needs to supply are the Instrument, Author, Revision, and Adjustment Threshold. The rest of the heading data is automatically updated.

Instrument, Author, and Revision are self explanatory. "Adjustment Threshold" is the point at which you want the operator to start making adjustments. Our procedures have a default Adjustment Threshold of 70%. This means that if the percent tolerance error at a particular test point is 70% or less, then the operator will not be allowed to make an adjustment at that point. If the percent of tolerance error at a test point is over 70%, the operator may go ahead and adjust the instrument as long as the procedure writer has placed an adjustment procedure at that point in the calibration procedure.

The adjustment threshold for a single test point may be viewed in the following manner:

	PERCENT ERROR	PASS/FAIL ADJ?
	over 100%	FAIL with ADJ
	71% to 100%	PASS with ADJ
Adjustment Threshold = 70%	70%	PASS with NO ADJ
	0% to 69%	PASS with NO ADJ

PROCEDURE FIELDS

Turn to page 1-13 in the 7404B Procedure Writing manual for a thorough discussion of the fields. The Step number is incremented automatically by the system. The procedure writer merely selects the FSC (Function Selection Code) and fills in the remaining data.

The data required by each FSC varies from FSC to FSC. If you ever have a question as to what information is required by a particular FSC, the Procedure Writing Manual contains thorough descriptions of each FSC in Section 4.

HOW DO YOU WRITE A PROCEDURE?

Turn to section 3 of your 7404B Procedure Writing Manual for information on procedure writing. Basically you must perform the following steps:

1. Boot your 7404B System Software Disk
2. Select [ ] Write of Modify a Procedure
3. a) Select "[ ] Write a New Procedure" if you are starting a new procedure.
   b) If you select "[ ] Create New from old Procedure" you will have both the original procedure and the new version when you are through.
   c) If you select "[ ] Modify a Procedure" you will be left with only the revised procedure when you are through.
4. Write your procedure by making selections on the 1722A screen and adding data as needed. Descriptions of the Procedure Edit List Options are given beginning on page 3-10 of your Procedure Writing Manual.
5. Save your procedure on your procedure disk.

PROCEDURE WRITING TUTORIAL

There is a procedure writing tutorial on page 2-1 of the 7404B Procedure Writing Manual. This tutorial provides an easy introduction to procedure writing. Step through this tutorial if you have not done so already. You may need a formatted procedure disk if you wish to save the Tutorial procedure you will be generating. For information on how to format a blank floppy, turn to the appropriate sections in the 1722A System Guide or in the 7404B Operator Manual. (You may also wish to review the 1722A Getting Stared Guide.)

LESSONS IN FUNDAMENTAL PROCEDURE WRITING

This section contains five procedure writing lessons along with five associated exercises. You should cover the material stated in this lesson plan, then do the associated exercise. Your objective should be to get through all five exercises. The exercises can be found in Section 2 of this manual. Answers to the exercises are in Section 3. Do not look at the answers until you have finished each exercise.

Make sure you understand the answers, then go on to the next lesson.

LESSON ONE - AUTOMATIC MESSAGES AND YOUR FIRST FSC

The 7404B Cluster automatically inserts certain messages and requests certain information to aid the operator in performing his calibrations. At the beginning of each calibration event, the 7404B will automatically request the operator to enter the UUT (unit under test) Serial Number and UUT Code (any special number associated with that instrument like asset number or property number). The operator will enter that information through the touch sensitive display of the 1722A. This information will be stored and printed out with the calibration results.

As the operator runs the calibration procedure, the system will notify him about which connections need to be made to the UUT. For example, if the procedure requires a voltage from the 5100B, the operator will be asked to connect the UUT to the 5100B output. This message will be automatically given to the operator before the stimulus is presented by the 5100B. This prevents connection to live ports on the 5100B.

All the instrument FSCs, i.e. 5100, 8506, 5205, 5220, will generate automatic connection messages. When the connection is no longer needed, the operator will be instructed to change the connections.

The 7404B will also present range setting messages to the operator automatically. As the ranges need to be changed on the UUT, the operator will be notified.

All the following messages are generated automatically by the system. The operator does not need to enter them into his procedure:

a) "Enter UUT Serial Number"
b) "Enter UUT Code"
c) Connection messages
d) Range setting messages
e) Disconnection messages

FUNCTION SELECTION CODES

The FSCs indicate the function of the particular procedure line. Basically the FSCs fall into three groups:

a) Evaluation FSCs (produce a pass/fail result)
b) message oriented FSCs (present messages/instructions to the operator)
c) Transparent FSCs (perform functions invisible to the operator)

See page 4-1 of the 7404B Procedure Writing Manual for a thorough description of Function Selection Codes.

5100 FSC

In order to see the information available in the Procedure Writing Manual and to learn about the most-used FSC, you should go over the 5100 FSC in detail. Turn to page 4-5 in the Procedure Writing Manual. The manual gives a general description of each FSC. Then each FSC parameter is discussed, from "Range" through "Con".

Key facts to remember are:

RANGE: <range> must be followed by R. Do not enter the units of the test. The range will default to the Nominal units.

Tolerance: The tolerance an be entered virtually the same way it is written in spec sheets. you can enter a floor value, a percent of reading value, and a percent of range value, and the system will compute the total tolerance for that test.
percent of nominal (reading): follow by % sign
percent of range: follow by / sign
in absolute units: follow by U

PROCEDURE WRITING EXERCISE 1

Turn to Exercise 1 at the start of Section 2 and follow the instructions. You will need to format a blank floppy disk to store the procedure you will be creating. Information on formatting is found in the 1722A System Guide. Basically, you must perform the following steps:

How to Format a Disk

1. Boot your 7404B System Software Disk. Remove it from the drive when instructed.
2. Enter CTRL T on the keyboard.
3. Enter CTRL P.
4. Type FUP.
5. Place a blank floppy disk in the 1722A disk drive.
6. Type mf0:/f
   (That is a zero after the first f.)

Proceed with Exercise 1 at this time. After you have saved your procedure on the blank disk, select "Calibrate an Instrument" from the main menu and run the procedure.

After running the procedure select "List Results on Screen" and view the results.

LESSON TWO - VERIFICATION PROCEDURES

Basically there are two types of tests you will be performing with your automated calibration system. These are:

1) Verification
2) Calibration

The system performs verification tests by presenting a stimulus to the UUT, allowing the operator to slew the stimulus until the UUT is reading appropriately, then calculating the amount of error in the instrument. In this manner the system "verifies" the UUT is within spec.

The procedure you wrote in Exercise 1 was a verification procedure even though it only tested one point.

HEAD FSC

Verification procedures will contain series of tests for DC Voltage, AC Voltage, and Resistance, for example. Each section should have a HEADing indicating the types of tests to be run.

Review the HEAD FSC in the 7404B Procedure Writing Manual. The HEADing is a one line title which will stay at the top of the 1722A screen throughout the calibration tests until changed.

If you want the HEADing printed in the results, you should place the HEADing in braces { }. Any information in braces will be directly printed in the results.

5100 FSC (FOR AC)

If the tests you are performing call for AC, you will use the 5100 FSC just as you did for DC tests. Merely add the frequency of the test while writing the procedure. Frequency should be followed by an "H".

RESF FSC

The 5100B contains fixed resistors form 1 ohm to 10 Mohm. These resistor values are called using the RESF FSC. you must enter a range and tolerance along with the resistance value when using RESF in verification evaluations.

PROCEDURE WRITING EXERCISE 2

Exercise 2 builds upon Exercise 1. Take the 1 line procedure from Exercise 1 and modify it into this Exercise 2 Procedure and again save it on your blank disk.

LESSON THREE - CALIBRATION PROCEDURES WITH ADJUSTMENTS

True calibration procedures contain the necessary steps to see if each test point is in cal or not and, if not, contain the necessary commands to bring the UUT into cal.

Calibration procedures are merely verification procedures with "adjustment" routines built into them.

ADJUSTMENT ROUTINES

Normally in your procedure you will perform a verification test to determine the degree of error in the UUT. If the percent of tolerance error is beyond the "adjustment threshold", the operator will be able to go into an adjustment routine. The adjustment routine will allow the operator to adjust the UUT and bring it back into spec.

Adjustment routines can easily be seen when viewing the listing of a procedure because they are indented. In the 8040A procedure listing on page 1-7 of the 7404B Procedure Writing Manual, there is an adjustment routine beginning on line 24.003.

A standard adjustment routine will have a stimulus presented (setup) by the calibrator. This stimulus will be held constant while instructions are presented telling the operator which adjustments to make to bring the UUT into spec. Then the routine will have the operator JuMP back to the initial verification check to determine whether the UUT is now within spec.

The approach in writing adjustment routines is summarized as follows:

1. Setup stimulus from the calibrator.
2. have a DISP and a DRAW which tells the operator which pot to turn.
3. JuMP back to the initial evaluation.

CALIBRATION PROCEDURE FORMAT

Each calibration procedure should begin with the following format:

a) First DRAW a picture of the meter for future reference in showing the locations of settings and adjustment controls.
b) Make a HEADing to indicate the type of tests being conducted.
c) Then have your instrument tests with adjustment procedures according to the instrument manual calibration procedure. Additional instructions can be given to the operator using the DISplay FSC.

You may refer to the calibration procedure listed in the 7404B Procedure Writing manual on page 1-7 to illustrate the above FSCs. Remember that any text inclosed in braces { } in a HEADing or DISPlay will be printed in the results.

DRAW FSC

Each procedure you write should begin with the DRAW Function Selection Code. The DRAW FSC will tell the system the shape of the meter for future reference in locating adjustment pots or setting controls.

Merely select the DRAW FSC at the start of the procedure, then shape the meter using the touch sensitive keys on the 1722A screen.

Later in the procedure when you want to show the location of an adjustment pot, for example, simply type in [DRAW,,L] if the pot is on the left side of the instrument, [DRAW,,R] if the pot is on the right side of the instrument, T for top, B for back, F for front. The system will then present to you a graphic locater with which to show the exact location of the pot on that side of the UUT.

DISP FSC

The DISP FSC shows a one-time display message to the operator with the [ADV] advance key to step past the display. While the HEAD FSC will remain until it is changed, the DISPlay will disappear when the operator presses ADV.

DISP is used to give the operator special calibration instructions.

JMP FSC

The JMP FSC allows and unconditional jump to another point in the procedure. In adjustment procedures, a JMP is used to return to the initial verification evaluation to see if the UUT is in cal after the adjustment routine.

You also have JMPT and JMPF and JMPZ FSCs which allow different portions of a procedure to be performed depending upon the outcome of tests or depending upon the intentions of the operator.

ADJUSTMENT PROCEDURE EXAMPLE

A typical adjustment procedure might look like this:

```
STEP  FSC    RANGE  NOMINAL  TOLERANCE    MOD1     MOD2  3   2   4 CON

1.001 DRAW            8 78 8 78           67   5 33 50
1.002 HEAD          {DC VOLTAGE CALIBRATION}
1.003 5100   200    190mV    .6U                                   2W
1.004        5100   190mV                                        S 2W
1.005        DISP   {ADJUST POT R6} FOR 190mV.[DRAWO,25,L]
1.006        JMP    1.003
```

PROCEDURE WRITING EXERCISE 3

Do Exercise 3. Build upon Exercise 1 and 2, modifying the procedure into the Exercise 3 procedure.

LESSON FOUR - "FAST" PROCEDURES

You can make your procedures "faster" by manipulating the automatic messages which the system normally presents. An experienced operator may not need to see every connection message. He may not need every range setting message. This lèsson teaches you how to control these messages.

ASK+, ASK- FSCs

The ASK+ and ASK- FSCs allow the procedure writer to control these automatic messages. For example, to turn off the automatic range message, the procedure writer would select "ASK- R." To turn off the automatic connection message, select "ASK- W."

Go through the description of ASK+ and ASK- in the Procedure Writing Manual.

OPBR FSC

The OPerator BRanch FSC will ask the operator a question and enable a JuMP to another part of the procedure depending on the response of the operator.

JMPF, JMPT FSCs

The OPerator BRanch FSC will place a 1 in a memory register [MEM1] if the operator answers YES to a question. OPBR will place a -1 in [MEM1] if the operator answers NO to that question.

JMPF <line number> will make procedure execution go to that line number if the operator answers NO. JMPF will not have any effect if the operator had answered YES.

JMPT <line number> will make procedure execution go to that line number if the operator answers YES. JMPT will not have any effect if the operator had answered NO.

PROCEDURE WRITING EXERCISE 4

Do Exercise 4 at this time. Again build on the procedure you have been working on.

LESSON FIVE - MEM MANIPULATION

This lesson teaches you manipulation of data using the MEMory registers.

[MEM] AND [MEM1]

Your system gives you two memory registers, MEM and MEM1 which you may use to manipulate data. Evaluation and "Nomset" tests (see discussion of MOD 4 in the 5100 FSC description) place the final value of the test in MEM. Setups like those used in adjustment routines place their values in MEM1.

MEMI FSC

Various FSCs allow you to utilize these memory registers. MEMI will present the operator with an instruction, then allow the operator to enter a value into the MEM register via the touch sensitive screen.

MEM*, MEM/, MEM+, AND MEM-

These FSCs allow manipulation of the value in [MEM]. The procedure writer can enter a value which he wants to multiply by, divide into, add to, or subtract from the value in [MEM]. For example, "MEM* 10" would multiply the value in MEM by 10.

If the operator wishes to multiply, divide, add, or subtract the value in MEM and the value in MEM1, he merely selects MEM*, /, +, or - without a number following the selection.

The final value in each case will be in MEM.

RSLT FSC

The ReSuLT FSC allows the entry of data directly into the results. Any statements or data after an = sign will be placed directly into the results.

If you want the value in MEM placed in the results, merely enclose MEM in brackets [MEM] and the value in MEM will be printed.

Go over the description of the RSLT FSC in the Procedure Writing Manual.

PROCEDURE WRITING EXERCISE 5

At this time turn to Exercise 5, again building upon your procedure.

CONGRATULATIONS

Exercise 5 was the last of the procedure exercises. Performing these five exercises will have given you a good basis for generating good procedures with a sound knowledge of the most important FSCs.

To complete your training in procedure writing, turn to Section 4 of the Procedure Writing Manual and study the descriptions of the FSCs which were not covered in these exercises. As practice, try adding steps requiring these FSCs to the procedure you have been generating.

Section 2
Introductory Training Exercises

7404B PROCEDURE WRITING EXERCISE 1

Write a procedure which performs the following:

1. allows the operator to enter the UUT serial number into the results,
2. allows the operator to enter a "UUT Code" into the results,
3. tells the operator to make the proper connections to the TI Panel,
4. tells the operator to set the UUT to the proper range, and
5. tells the operator to remove the TI Panel connections when he is done.

This procedure should also perform one verification test. That test should tell the operator to set the UUT to the 2 volt range, elicit 1.9 volts DC from the 5100B Calibrator, and test to a tolerance of .006 volt.

After you have written the procedure:

1. "Backup" the procedure and save it on your blank disk,

7404B PROCEDURE WRITING EXERCISE 2

Continuing with the procedure you began in Exercise 1, write a procedure which contains verification tests for all the test points listed below. You should have separate HEADings for DC Voltage, AC Voltage, and Resistance tests.

When you are through writing this verification procedure, save it on your blank disk.

DC VOLTAGE CALIBRATION

RANGE	INPUT VOLTAGE	TOLERANCE	CALIBRATION COMPONENT
200mV	190mV	.6mV	R6
200mV	-190mV	.6mV	None
2V	1.9V	.006V	None
20V	19V	.06V	None
200V	190V	.6V	None
1000V	1000V	3V	None

AC VOLTAGE CALIBRATION

RANGE	INPUT VOLTAGE	INPUT FREQUENCY	TOLERANCE	CALIBRATION COMPONENT
200mV	190mV	100Hz	1.6mV	R4
2V	1.9V	100Hz	.016V	None
20V	19V	100Hz	.16V	None
200V	190V	100Hz	1.6V	None
1000V	750V	400Hz	9V	None

RESISTANCE CALIBRATION

RANGE	INPUT RESISTANCE	TOLERANCE	CALIBRATION COMPONENT
200 ohms	10 ohms	.9 ohm	None
2 Kohms	1.0 Kohm	.005 Kohm	None
20 Kohms	10 Kohms	.05 Kohm	None
200 Kohms	100 Kohms	.5 Kohm	None
2 Mohms	1.0 Mohm	.005 Mohm	None
20 Mohms	10 Mohms	.2 Mohm	None

7404B PROCEDURE WRITING EXERCISE 3

Make your Verification Procedure from Exercise 2 into a Calibration Procedure.

If you look at the specifications listed for Exercise 2 you will sée there are two Calibration Components listed. These are R6 for the 190 mV DC Voltage test and R4 for the 190 mV AC Voltage test.

Add R6 and R4 adjustment routines to the verification procedure you wrote in Exercise 2. These adjustment routines should contain graphic displays showing the locations of R6 and R4 along with the appropriate instructions to the operator.

The front of the meter looks like this:

```
 --------------------------
|  -----------             |
| |           |            |
|  -----------             |
 --------------------------
```

Both pots are located on the left side of the UUT:

```
 --------------------------
|                          |
|                R4 o      |
|   o R6                   |
 --------------------------
```

When you are through modifying the Exercise 2 Verification Procedure into this Exercise 3 Calibration Procedure, save it on your blank disk. Then boot your Instrument Calibration Disk and run the procedure to see how the DRAWings are displayed.

7404B PROCEDURE WRITING EXERCISE 4

Using the skills you have learned, continue improving the procedure you have been creating. Allow the operator to decide whether he wants:

1) all automatic messages,

2) no range messages, or

3) no connection messages.

Add this feature to your procedure, then run the procedure and note the changes in the automatic messages.

Remember when you delete certain automatic messages, the operator must know the UUT well enough to calibrate the instrument without the automatic instructions.

7404B PROCEDURE WRITING EXERCISE 5

This exercise focuses on use of the MEM register to manipulate data and perform calculations.

This addition to your procedure is for a cal lab which wants the lab temperature printed in the results in degrees Celcius. The cal lab, however, has a thermometer which only reads degrees Fahrenheit.

Have the operator enter the lab temperature in degrees Fahrenheit. Then in the procedure convert degrees Fahrenheit to degrees Celcius using the calculating capability you have with the MEM register.

Remember the formula for converting degrees Fahrenheit to degrees Celcius?

TC = (TF - 32) * 5/9

Finally, have the temperature in degrees Celcius printed in the results.

Save this procedure on your disk and try it out. Run through a few tests, then terminate the procedure and look at the results.

Section 3
Introductory Training Answers

JOHN FLUKE MFG. CO., INC. 7404B PROCEDURE

===

INSTRUMENT: PROCEDURE WRITING EXERCISE 1

DATE: 31-Aug-83
AUTHOR: DWIGHT HYLAND
REVISION: 1.0

ADJUSTMENT THRESHOLD: 70%

NUMBER OF TESTS: 1
NUMBER OF LINES: 22
CONFIGURATION: 256

===

STEP	FSC	RANGE	NOMINAL	TOLERANCE	MOD1	MOD2	3	4	CON
1.001	5100	2	1.9V	.006U					2W

END

JOHN FLUKE MFG. CO., INC. 7404B PROCEDURE

==

INSTRUMENT: PROCEDURE WRITING EXERCISE 2

DATE: 31-Aug-83
AUTHOR: DWIGHT HYLAND
REVISION: 1.0

ADJUSTMENT THRESHOLD: 70%

NUMBER OF TESTS: 17
NUMBER OF LINES: 41
CONFIGURATION: 256

==

STEP	FSC	RANGE	NOMINAL	TOLERANCE	MOD1	MOD2	3	4	CON
1.001	HEAD		{DC VOLTAGE CALIBRATION}						
1.002	5100	200	190mV	.6U					2W
2.001	5100	200	-190mV	.6U					2W
3.001	5100	2	1.9V	.006U					2W
4.001	5100	20	19V	.06U					2W
5.001	5100	200	190V	.6U					2W
.001	5100	1000	1000V	3U					2W
.001	HEAD		{AC VOLTAGE CALIBRATION}						
7.002	5100	200	190mV	1.6U	100H				2W
8.001	5100	2	1.9V	.016U	100H				2W
9.001	5100	20	19V	.16U	100H				2W
10.001	5100	200	190V	1.6U	100H				2W
11.001	5100	1000	750V	9U	400H				2W
12.001	HEAD		{RESISTANCE CALIBRATION}						
12.002	RESF	200	100Z	.9U					2W
13.001	RESF	2	1KZ	.005U					2W
14.001	RESF	20	10KZ	.05U					2W
15.001	RESF	200	100KZ	.5U					2W
16.001	RESF	2	1MZ	.005U					2W
17.001	RESF	20	10MZ	.2U					2W

END

```
================================================================================

INSTRUMENT:               PROCEDURE WRITING EXERCISE 3

DATE:                     31-Aug-83
AUTHOR:                   DWIGHT HYLAND
REVISION:                 1.0

ADJUSTMENT THRESHOLD:     70%

NUMBER OF TESTS:          17
NUMBER OF LINES:          48
CONFIGURATION:            256

================================================================================

 STEP     FSC     RANGE NOMINAL        TOLERANCE     MOD1          MOD2  3   4   CON

  1.001   DRAW            8 78   8 78                  67   5 33 50
  1.002   HEAD          {DC VOLTAGE CALIBRATION}
  1.003   5100    200   190mV          .6U                                         2W
  1.004           5100  190mV                                                  S   2W
  1.005           DISP  {ADJUST POT R6} FOR A READING OF 190mV.[DRAW0,25,L]
  1.006           JMP   1.003
  2.001   5100    200   -190mV         .6U
  3.001   5100    2     1.9V           .006U
  4.001   5100    20    19V            .06U                                        2W
  5.001   5100    200   190V           .6U                                         2W
  6.001   5100    1000  1000V          3U                                          2W
  7.001   HEAD          {AC VOLTAGE CALIBRATION}
  7.002   5100    200   190mV          1.6U          100H                          2W
  7.003           5100  190mV                        100H                      S   2W
  7.004           DISP  {ADJUST POT R4} FOR A READING OF 190mV.[DRAW33,70,L]
  7.005           JMP   7.002
  8.001   5100    2     1.9V           .016U         100H                          2W
  9.001   5100    20    19V            .16U          100H                          2W
 10.001   5100    200   190V           1.6U          100H                          2W
 11.001   5100    1000  750V           9U            400H                          2W
 12.001   HEAD          {RESISTANCE CALIBRATION}
 12.002   RESF    200   100Z           .9U                                         2W
 13.001   RESF    2     1KZ            .005U                                       2W
 14.001   RESF    20    10KZ           .05U                                        2W
 15.001   RESF    200   100KZ          .5U                                         2W
 16.001   RESF    2     1MZ            .005U                                       2W
 17.001   RESF    20    10MZ           .2U                                         2W
END
```

==

INSTRUMENT: PROCEDURE WRITING EXERCISE 4

DATE: 31-Aug-83
AUTHOR: DWIGHT HYLAND
REVISION: 1.0

ADJUSTMENT THRESHOLD: 70%

NUMBER OF TESTS: 17
NUMBER OF LINES: 60
CONFIGURATION: 256

==

STEP	FSC	RANGE	NOMINAL	TOLERANCE	MOD1	MOD2	3	4	CON
1.001	DRAW		8 78 8 78		67 5 33 50				
1.002	DISP		THIS PROCEDURE IS ADAPTABLE TO VARYING OPERATOR SKILL						
1.002	DISP		LEVELS. IF THE METER YOU ARE CALIBRATING IS FAMILIAR TO						
1.002	DISP		YOU, THE AUTOMATIC RANGE AND CONNECTION MESSAGES MAY BE						
1.002	DISP		DELETED BY ANSWERING THE FOLLOWING QUESTIONS						
1.002	DISP		APPROPRIATELY.						
.003	OPBR		WOULD YOU LIKE TO TURN OFF AUTOMATIC RANGE MESSAGES?						
.004	JMPF		1.006						
1.005	ASK-	R							
1.006	OPBR		WOULD YOU LIKE TO TURN OFF AUTOMATIC CONNECTION						
1.006	OPBR		MESSAGES?						
1.007	JMPF		1.009						
1.008	ASK-								W
1.009	HEAD		{DC VOLTAGE CALIBRATION}						
1.010	5100	200	190mV	.6U					2W
1.011		5100	190mV					S	2W
1.012		DISP	{ADJUST POT R6} FOR A READING OF 190mV.[DRAW0,25,L]						
1.013		JMP	1.010						
2.001	5100	200	-190mV	.6U					2W
3.001	5100	2	1.9V	.006U					2W
4.001	5100	20	19V	.06U					2W
5.001	5100	200	190V	.6U					2W
6.001	5100	1000	1000V	3U					2W
7.001	HEAD		{AC VOLTAGE CALIBRATION}						
7.002	5100	200	190mV	1.6U	100H				2W
7.003		5100	190mV		100H			S	2W
7.004		DISP	{ADJUST POT R4} FOR A READING OF 190mV.[DRAW33,70,L]						
7.005		JMP	7.002						
8.001	5100	2	1.9V	.016U	100H				2W
9.001	5100	20	19V	.16U	100H				2W
10.001	5100	200	190V	1.6U	100H				2W
11.001	5100	1000	750V	9U	400H				2W
12.001	HEAD		{RESISTANCE CALIBRATION}						
12.002	RESF	200	100Z	.9U					2W
.001	RESF	2	1KZ	.005U					2W
.001	RESF	20	10KZ	.05U					2W
15.001	RESF	200	100KZ	.5U					2W
16.001	RESF	2	1MZ	.005U					2W
17.001	RESF	20	10MZ	.2U					2W

END

==

INSTRUMENT: PROCEDURE WRITING EXERCISE 5

DATE: 31-Aug-83
AUTHOR: DWIGHT HYLAND
REVISION: 1.0

ADJUSTMENT THRESHOLD: 70%

NUMBER OF TESTS: 17
NUMBER OF LINES: 65
CONFIGURATION: 256

==

STEP	FSC	RANGE	NOMINAL	TOLERANCE	MOD1	MOD2	3	4	CON
1.001	DRAW		8 78 8 78		67 5 33 50				
1.002	DISP		THIS PROCEDURE IS ADAPTABLE TO VARYING OPERATOR SKILL						
1.002	DISP		LEVELS. IF THE METER YOU ARE CALIBRATING IS FAMILIAR TO						
1.002	DISP		YOU, THE AUTOMATIC RANGE AND CONNECTION MESSAGES MAY BE						
1.002	DISP		DELETED BY ANSWERING THE FOLLOWING QUESTIONS						
1.002	DISP		APPROPRIATELY.						
1.003	OPBR		WOULD YOU LIKE TO TURN OFF AUTOMATIC RANGE MESSAGES?						
1.004	JMPF		1.006						
1.005	ASK-	R							
1.006	OPBR		WOULD YOU LIKE TO TURN OFF AUTOMATIC CONNECTION						
1.006	OPBR		MESSAGES?						
1.007	JMPF		1.009						
1.008	ASK-								W
1.009	MEMI		ENTER THE LAB TEMPERATURE IN DEGREES FAHRENHEIT.						
1.010	MEM-		32						
1.011	MEM*		5						
1.012	MEM/		9						
1.013	RSLT		=LAB TEMPERATURE IS [MEM] DEGREES CELCIUS.						
1.014	HEAD		{DC VOLTAGE CALIBRATION}						
1.015	5100	200	190mV	.6U					2W
1.016		5100	190mV					S	2W
1.017		DISP	{ADJUST POT R6} FOR A READING OF 190mV.[DRAW0,25,L]						
1.018		JMP	1.015						
2.001	5100	200	-190mV	.6U					2W
3.001	5100	2	1.9V	.006U					2W
4.001	5100	20	19V	.06U					2W
5.001	5100	200	190V	.6U					2W
6.001	5100	1000	1000V	3U					2W
7.001	HEAD		{AC VOLTAGE CALIBRATION}						
7.002	5100	200	190mV	1.6U	100H				2W
7.003		5100	190mV		100H			S	2W
7.004		DISP	{ADJUST POT R4} FOR A READING OF 190mV.[DRAW33,70,L]						
7.005		JMP	7.002						
8.001	5100	2	1.9V	.016U	100H				2W
9.001	5100	20	19V	.16U	100H				
10.001	5100	200	190V	1.6U	100H				
11.001	5100	1000	750V	9U	400H				2W
12.001	HEAD		{RESISTANCE CALIBRATION}						
12.002	RESF	200	100Z	.9U					2W
13.001	RESF	2	1KZ	.005U					2W

14.001	RESF	20	10KZ	.05V	2W
15.001	RESF	200	100KZ	.5V	2W
16.001	RESF	2	1MZ	.005V	2W
17.001	RESF	20	10MZ	.2V	2W

END