Training Manual

Tektronix

Tektronix Logic Analyzer Family
TLA 7QS QuickStart Training Board

070-9717-05

This document applies to System Software Version 3.2 and above.

www.tektronix.com

Printed in the U.S.A.

Tektronix, Inc., P.O. Box 1000, Wilsonville, OR 97070–1000

HARDWARE WARRANTY

Tektronix warrants that the parts, assemblies and supplies (“products”) that it manufactures and sells will be free from defects in materials and workmanship for a period of three (3) months from the date of shipment. If a product proves defective during this warranty period, Tektronix, at its option, either will repair the defective product without charge for parts and labor, or will provide a replacement in exchange for the defective product.

In order to obtain service under this warranty, Customer must notify Tektronix of the defect before the expiration of the warranty period and make suitable arrangements for the performance of service. Customer shall be responsible for packaging and shipping the defective product to the service center designated by Tektronix, with shipping charges prepaid. Tektronix shall pay for the return of the product to Customer if the shipment is to a location within the country in which the Tektronix service center is located. Customer shall be responsible for paying all shipping charges, duties, taxes, and any other charges for products returned to any other locations.

This warranty shall not apply to any defect, failure or damage caused by improper use or improper or inadequate maintenance and care. Tektronix shall not be obligated to furnish service under this warranty a) to repair damage resulting from attempts by personnel other than Tektronix representatives to install, repair or service the product; b) to repair damage resulting from improper use or connection to incompatible equipment; c) to repair any damage or malfunction caused by the use of non-Tektronix supplies; or d) to service a product that has been modified or integrated with other products when the effect of such modification or integration increases the time or difficulty of servicing the product.

THIS WARRANTY IS GIVEN BY TEKTRONIX IN LIEU OF ANY OTHER WARRANTIES, EXPRESS OR IMPLIED. TEKTRONIX AND ITS VENDORS DISCLAIM ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. TEKTRONIX’ RESPONSIBILITY TO REPAIR OR REPLACE DEFECTIVE PRODUCTS IS THE SOLE AND EXCLUSIVE REMEDY PROVIDED TO THE CUSTOMER FOR BREACH OF THIS WARRANTY. TEKTRONIX AND ITS VENDORS WILL NOT BE LIABLE FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES IRRESPECTIVE OF WHETHER TEKTRONIX OR THE VENDOR HAS ADVANCE NOTICE OF THE POSSIBILITY OF SUCH DAMAGES.

THIS WARRANTY IS GIVEN BY TEKTRONIX IN LIEU OF ANY OTHER WARRANTIES, EXPRESS OR IMPLIED. TEKTRONIX AND ITS VENDORS DISCLAIM ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. TEKTRONIX’ RESPONSIBILITY TO REPLACE DEFECTIVE MEDIA OR REFUND CUSTOMER’S PAYMENT IS THE SOLE AND EXCLUSIVE REMEDY PROVIDED TO THE CUSTOMER FOR BREACH OF THIS WARRANTY. TEKTRONIX AND ITS VENDORS WILL NOT BE LIABLE FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES IRRESPECTIVE OF WHETHER TEKTRONIX OR THE VENDOR HAS ADVANCE NOTICE OF THE POSSIBILITY OF SUCH DAMAGES.

SOFTWARE WARRANTY

Tektronix warrants that the media on which this software product is furnished and the encoding of the programs on the media will be free from defects in materials and workmanship for a period of three (3) months from the date of shipment. If a medium or encoding proves defective during the warranty period, Tektronix will provide a replacement in exchange for the defective medium. Except as to the media on which this software product is furnished, this software product is provided “as is” without warranty of any kind, either express or implied. Tektronix does not warrant that the functions contained in this software product will meet Customer’s requirements or that the operation of the programs will be uninterrupted or error-free.

In order to obtain service under this warranty, Customer must notify Tektronix of the defect before the expiration of the warranty period. If Tektronix is unable to provide a replacement that is free from defects in materials and workmanship within a reasonable time thereafter, Customer may terminate the license for this software product and return this software product and any associated materials for credit or refund.

THIS WARRANTY IS GIVEN BY TEKTRONIX IN LIEU OF ANY OTHER WARRANTIES, EXPRESS OR IMPLIED. TEKTRONIX AND ITS VENDORS DISCLAIM ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. TEKTRONIX’ RESPONSIBILITY TO REPLACE DEFECTIVE MEDIA OR REFUND CUSTOMER’S PAYMENT IS THE SOLE AND EXCLUSIVE REMEDY PROVIDED TO THE CUSTOMER FOR BREACH OF THIS WARRANTY. TEKTRONIX AND ITS VENDORS WILL NOT BE LIABLE FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES IRRESPECTIVE OF WHETHER TEKTRONIX OR THE VENDOR HAS ADVANCE NOTICE OF THE POSSIBILITY OF SUCH DAMAGES.

Table of Contents

General Safety Summary . . . . . **ix**

Preface . . . . . **xi**

How to Use This Document xi

Contacting Tektronix xii

Getting Started

Prerequisites 1–1

Accessories 1–2

Configuration 1–2

Care and Maintenance 1–3

Obtaining Parts 1–4

General Purpose Exercises

General Purpose Exercises Setups **2–1**

Hardware Setups 2–1

Set up the P6417 Probes 2–1

Connect the Probes 2–2

Load the Setups 2–4

Exercise Overview 2–6

Triggering on a Glitch (Exercise 1) **2–7**

Load the Setup 2–8

View the Resultant Data 2–9

View the Trigger Setup 2–11

View the Channel Setups 2–13

Capture a Pulse Width Violation (Exercise 2) **2–15**

Load the Setup 2–15

View the Acquired Data 2–16

View the Trigger Program 2–18

View the Channel Setups 2–19

Capture a Setup and Hold Violation (Exercise 3) **2–21**

Load the Setup 2–21

View the Resultant Data 2–21

View the Trigger Setup 2–23

Counting Setup and Hold Violations (Exercise 4) **2–25**

Load the Setup 2–25

View the Status Monitor 2–26

View the Setups 2–27

Capturing Data Bursts Using Transitional Storage (Exercise 5) **2–29**

Load the Setup 2–30

View the Channel Setups 2–31

View the Trigger Program Using Conventional Storage 2–32

View the Acquired Data Using Conventional Storage 2–33

View the Trigger Program Using Transitional Storage 2–34

View the Acquired Data Using Transitional Storage 2–35

Optional 2–37

Conclusion 2–38

Using the DSO to Trigger the Logic Analyzer on a Runt Pulse (Exercise 6) 2–39
Load the Setup 2–39
View the Resultant Data 2–40
View the Trigger Setups 2–42
Using the Logic Analyzer to Trigger the DSO (Exercise 7) 2–45
Load the Setup 2–45
View the Resultant Data 2–46
View the Setups 2–48

Microprocessor Support Debug Exercises

Microprocessor Exercises Setup 3–1
Hardware Setups 3–1
Set Up the P6418 or P6417 Probes 3–1
Connect the Probes 3–2
Load the Setups 3–5
Trigger on a Power-on Reset and Capture the Controller Startup Code (Exercise 1) 3–9
Load the Setup 3–9
View the Resultant Data 3–10
View the Trigger and Channel Setups 3–12
Use Trigger Timers to Measure Interrupt Latency (Exercise 2) 3–13
Load the Setup 3–13
Measure Timers 3–14
View the Resultant Data 3–15
View the Setups 3–16
Trigger on Faulty Data Written to the LED Display (Exercise 3) . . . 3–17
Load the Setup 3–18
View the Resultant Data 3–18
View the Setups 3–19
Trigger on Faulty Data Read by the CPU (Exercise 4) 3–21
Load the Setup 3–21
View the Resultant Data 3–22
View the Setups 3–24
Trigger on a Setup Violation of the CPU Read Cycle (Exercise 5) . . . 3–25
Load the Setup 3–25
View the Resultant Data 3–25
View the Trigger Setups 3–27

Embedded Software Debug Exercises

Embedded Software Debug Exercises Setup 4–1

Source Code Window Background (Exercise 1) 4–3
Source Window Structure 4–3

Debugging Real-Time Execution of High Level Source (Exercise 1) . 4–5
Load the Saved System 4–5
Using the Source Window 4–9
Triggering on Source Code Statements 4–18
Source Window Features 4–20
Understanding the Tektronix Logic Analyzer Symbol Support 4–22
Summary 4–26

Automating System Verification (Exercise 2) 4–27
Load the Saved System 4–27
Create Reference Saved System 4–28
Set Up Memory Comparison 4–30
Set Up Repetitive Acquisition 4–34
Run the QuickStart Program and View the Acquired Data 4–35
View the Trigger Program 4–38

Tektronix Logic Analyzer Programmatic Interface (TPI) Background (Exercise 3) 4–39
General Characteristics 4–39
TLAScript 4–40

Remotely Controlling the Tektronix Logic Analyzer with the Programmatic Interface (Exercise 3) 4–43
Application Requirements 4–43
Client Application Description 4–43
Create Test Module Setup 4–44
Run the Client Application and View the Acquired Data 4–45
Single-Step Through the Client Application 4–47

Histogram Window Background (Exercises 4 and 5) 4–51
Histogram Window Using Channel Groups 4–51
Histogram Window Using Counters and Timers 4–51

Characterizing System Performance (Exercise 4) 4–53
Load the Saved System 4–53
Create a New Histogram Window 4–54
Run the QuickStart Program and View the Acquired Data 4–56
View the Trigger Program 4–60

Optimizing Execution Time of Embedded Software (Exercise 5) 4–61
Load the Saved System 4–61
Create a New Histogram Window 4–62
Run the QuickStart Program and View the Acquired Data 4–64
View the Trigger Program 4–68

Pattern Generator Exercises

Pattern Generator Exercises Setup 5–1
Hardware Setups 5–1
Connect the Logic Analyzer Probes 5–2
Connect the Pattern Generator Probe 5–3

Pattern Generator Exercises 5–7
Load the Setups 5–8
View the Resultant Data 5–11
View the Pattern Generator Setup 5–13
View the Logic Analyzer Setup 5–15
Going Further 5–15

Appendices

Appendix A: How to Create Setups A–1
Load the Default Setup A–1
Define the Setup Window A–1
Define the Trigger Window A–3
Define the Waveform Window A–5

Appendix B: Training Board Connections B–1

List of Figures

Figure 2–1: Removing probe podlets 2–1
Figure 2–2: P6418 probe connections for the timing exercises 2–2
Figure 2–3: P6417 probe connections for the timing exercises 2–3
Figure 2–4: Load System dialog box 2–4
Figure 2–5: Restored system for Exercise 1 2–8
Figure 2–6: Glitch data for Exercise 1 2–9
Figure 2–7: Waveform display with the MagniVu feature 2–10
Figure 2–8: Trigger window for Exercise 1 2–11
Figure 2–9: Glitch Detection dialog box for Exercise 1 2–12
Figure 2–10: Channel setups for Exercise 1 2–13
Figure 2–11: Restored system for Exercise 2 2–15
Figure 2–12: Timing waveform display for Exercise 2 2–16
Figure 2–13: Timing waveform using MagniVu acquisition 2–17
Figure 2–14: LA Trigger window for Exercise 2 2–18
Figure 2–15: Channel setups for Exercise 2 2–19
Figure 2–16: Timing waveform display for Exercise 3 2–21
Figure 2–17: Define Violation dialog box for Exercise 3 2–23
Figure 2–18: Status Monitor for Exercise 4 2–26
Figure 2–19: LA Trigger window for Exercise 4 2–27
Figure 2–20: Restored system for Exercise 5 2–30
Figure 2–21: Setup window for Exercise 5 2–31
Figure 2–22: Trigger window using conventional storage for Exercise 5 2–32
Figure 2–23: Timing waveform using conventional storage for Exercise 5 2–33
Figure 2–24: Trigger window using transitional storage for Exercise 5 2–34
Figure 2–25: Timing waveform using transitional storage for Exercise 5 2–35
Figure 2–26: Waveform data at 100 ns Time/Div 2–36
Figure 2–27: Timing waveform using transitional storage with MagniVu for Exercise 5 2–37
Figure 2–28: Timing waveform display for Exercise 6 2–41
Figure 2–29: DSO channel 1 setups for Exercise 6 2–42
Figure 2–30: DSO trigger setups for Exercise 6 2–43
Figure 2–31: Waveform display for Exercise 7 2–46
Figure 2–32: Resultant waveform data for Exercise 7 2–47

Figure 2–33: DSO channel trigger setups for Exercise 7 2–48
Figure 3–1: Microprocessor exercise P6417 probe connections 3–2
Figure 3–2: Microprocessor exercise P6418 probe connections 3–3
Figure 3–3: Microprocessor exercise P6434 probe connections 3–4
Figure 3–4: Load System dialog box 3–5
Figure 3–5: Load System Options dialog box 3–6
Figure 3–6: Listing window for Exercise 1 3–10
Figure 3–7: Timing waveform display for Exercise 1 3–11
Figure 3–8: Trigger window for Exercise 1 3–12
Figure 3–9: Status Monitor 3–14
Figure 3–10: Listing window for Exercise 2 3–15
Figure 3–11: Trigger window for Exercise 2 3–16
Figure 3–12: Listing window for Exercise 3 3–18
Figure 3–13: Trigger window for Exercise 3 3–19
Figure 3–14: Listing window for Exercise 4 3–22
Figure 3–15: Waveform window for Exercise 4 3–23
Figure 3–16: Waveform window with dashed zoom box 3–23
Figure 3–17: Trigger window for Exercise 4 3–24
Figure 3–18: Waveform data window for Exercise 5 3–26
Figure 3–19: Expanded data channels showing a setup violation for Exercise 5 3–26
Figure 3–20: Trigger window for Exercise 5 3–27
Figure 3–21: Define Violation dialog box 3–28
Figure 4–1: Restored system for Exercise 1 4–5
Figure 4–2: Real-time trace of the STOP LITES program 4–6
Figure 4–3: New Data Window wizard dialog box 4–7
Figure 4–4: Lites Source window after connecting to the Lites List Listing window 4–8
Figure 4–5: Recommended window layout for the portable mainframe 4–9
Figure 4–6: Source and Listing windows after stepping forward eight times 4–10
Figure 4–7: Source and Listing windows after using “Move Cursor 1 Here” 4–12
Figure 4–8: Source and Listing windows after placing a mark (step 1) 4–13
Figure 4–9: Source and Listing windows after step 2 4–14
Figure 4–10: Source and Listing windows after step 3 4–15
Figure 4–11: Source and Listing windows after stepping forward twice 4–17

Figure 4–12: Trigger window containing the copied address 4–19
Figure 4–13: Source window 4–20
Figure 4–14: Symbols dialog box 4–22
Figure 4–15: Select Symbol File dialog box 4–23
Figure 4–16: Load Symbols Options dialog box 4–23
Figure 4–17: Symbols dialog box 4–24
Figure 4–18: Function symbols 4–25
Figure 4–19: Variable symbols 4–25
Figure 4–20: Source symbols 4–26
Figure 4–21: Restored system for Exercise 2 4–27
Figure 4–22: Good Data Listing window, used as reference data 4–29
Figure 4–23: Good Timing Waveform window, identifying valid data read 4–30
Figure 4–24: LA module Setup window 4–31
Figure 4–25: Disable the comparison (136 channel LA modules only) 4–32
Figure 4–26: LA module Define Compare dialog box 4–33
Figure 4–27: Repetitive Properties dialog box 4–34
Figure 4–28: Set up the logic analyzer to display a message 4–34
Figure 4–29: Test Run Listing window, identifying data that has changed 4–36
Figure 4–30: Test Timing Waveform window 4–37
Figure 4–31: Trigger window for Exercise 2 4–38
Figure 4–32: Saving Test Module Setup 4–44
Figure 4–33: Client application connected to the logic analyzer server (before acquiring data) 4–46
Figure 4–34: VBA Source window after setting the breakpoint 4–47
Figure 4–35: Results after clicking on the client application RUN button 4–48
Figure 4–36: Source code of the VBA client application (part one of two) 4–49
Figure 4–37: Source code of the VBA client application (part two of two) 4–50
Figure 4–38: Restored system for Exercise 4 4–53
Figure 4–39: Setting the LA parameters 4–54
Figure 4–40: Loading the symbol file 4–55
Figure 4–41: Customized Overview Histogram window 4–56
Figure 4–42: Histogram data window showing program activity during normal operation 4–57
Figure 4–43: Histogram window showing where the application is spending most of its time 4–58

Figure 4–44: Histogram window showing program activity sorted by decreasing execution time 4–59
Figure 4–45: Overview Histogram window showing split screen capabilities 4–59
Figure 4–46: Trigger window for Exercise 4 4–60
Figure 4–47: Restored system for Exercise 5 4–61
Figure 4–48: Setting the data source parameters 4–62
Figure 4–49: Setting the Range parameters 4–63
Figure 4–50: The Int 3 ACK Histogram window 4–63
Figure 4–51: Accumulated results for Timer 1 after 7 acquisitions . . 4–65
Figure 4–52: Results of Timer 2 after 7 acquisitions 4–66
Figure 4–53: Integrated view of the Listing and Histogram windows 4–67
Figure 4–54: Trigger window 4–68
Figure 5–1: P6434 Probe connections 5–2
Figure 5–2: P6418 probe connections 5–3
Figure 5–3: Connecting the pattern generator probe to the top of the training board 5–4
Figure 5–4: Connecting the pattern generator board to the edge connector 5–5
Figure 5–5: Restored system for the pattern generator exercises 5–8
Figure 5–6: Restored system for the logic analyzer 5–9
Figure 5–7: System window for the pattern generator 5–10
Figure 5–8: Waveform display showing IRQ~6_Run assertion 5–11
Figure 5–9: Listing window showing the Run assertion 5–12
Figure 5–10: Program dialog box 5–13
Figure 5–11: Sequence showing program selection and run 5–14
Figure A–1: Setup window for Exercise 1 . . . . . A–3
Figure A–2: Glitch Detection dialog box A–4
Figure A–3: Trigger window setups A–5

General Safety Summary

Review the following safety precautions to avoid injury and prevent damage to this product or any products connected to it. To avoid potential hazards, use this product only as specified.

Only qualified personnel should perform service procedures.

While using this product, you may need to access other parts of the instrument. Read the *General Safety Summary* in other manuals for warnings and cautions related to operating the instrument.

To Avoid Fire or Personal Injury

Connect and Disconnect Properly. Do not connect or disconnect probes or test leads while they are connected to a voltage source.

Observe All Terminal Ratings. To avoid fire or shock hazard, observe all ratings and marking on the product. Consult the product manual for further ratings information before making connections to the product.

Do not apply a potential to any terminal, including the common terminal, that exceeds the maximum rating of that terminal.

Use Proper AC Adapter. Use only the AC adapter specified for this product.

Use Proper Fuse. Use only the fuse type and rating specified for this product.

Avoid Exposed Circuitry. Do not touch exposed connections and components when power is present.

Do Not Operate With Suspected Failures. If you suspect there is damage to this product, have it inspected by qualified service personnel.

Do Not Operate in Wet/Damp Conditions.

Do Not Operate in an Explosive Atmosphere.

Keep Product Surfaces Clean and Dry.

Provide Proper Ventilation. Refer to the manual's installation instructions for details on installing the product so it has proper ventilation.

Symbols and Terms

Terms in this Manual. These terms may appear in this manual:

WARNING. *Warning statements identify conditions or practices that could result in injury or loss of life.*

CAUTION. *Caution statements identify conditions or practices that could result in damage to this product or other property.*

Terms on the Product. These terms may appear on the product:

DANGER indicates an injury hazard immediately accessible as you read the marking.

WARNING indicates an injury hazard not immediately accessible as you read the marking.

CAUTION indicates a hazard to property including the product.

Symbols on the Product. The following symbols may appear on the product:

WARNING
High Voltage

Protective Ground
(Earth) Terminal

CAUTION
Refer to Manual

Double
Insulated

Preface

The TLA 7QS QuickStart Training Manual is part of the TLA 7QS QuickStart package. It is intended to be used with the Tektronix Logic Analyzer Family as a training tool to learn some of the specific features of the logic analyzer.

You can use the training manual together with the online help to learn a basic overview of the Tektronix Logic Analyzer Family.

How to Use This Document

Use this training manual with the TLA 7QS QuickStart board to work through a series of exercises showing the features and benefits of the Tektronix Logic Analyzer Family. The exercises are intended to simulate typical problems that you may encounter and require a logic analysis system to remedy the problems.

The training manual is made up of the following sections:

- The *Getting Started* chapter provides information on prerequisites, product accessories, product installation, and basic product care and maintenance.
- The *General Purpose Exercises* provide examples to demonstrate the timing analysis features of the logic analyzers.
- The *Microprocessor Support Exercises* provide examples to demonstrate the microprocessor features of the logic analyzers.
- The *Embedded Software Debug Exercises* provide examples to demonstrate the software debugging features of the logic analyzers.
- The *Pattern Generator Exercises* provide examples to demonstrate the pattern generation features of the logic analyzers.
- *Appendix A: How to Create Setups Used in this Book* shows how to use the menus of the logic analyzer to create the setups for the first general purpose exercise. It provides a means to become familiar with the menus in creating a logic analyzer setup.
- *Appendix B: Training Board Connections* shows the hardware connections on the training board.

Contacting Tektronix

Phone	1-800-833-9200*****
Address	Tektronix, Inc. Department or name (if known) 14200 SW Karl Braun Drive P.O. Box 500 Beaverton, OR 97077 USA
Web site	www.tektronix.com
Sales support	1-800-833-9200, select option 1*****
Service support	1-800-833-9200, select option 2*****
Technical support	Email: support@tektronix.com 1-800-833-9200, select option 3***** 1-503-627-2400 6:00 a.m. – 5:00 p.m. Pacific time

*** This phone number is toll free in North America. After office hours, please leave a voice mail message.**
Outside North America, contact a Tektronix sales office or distributor; see the Tektronix web site for a list of offices.

Getting Started

Getting Started

This chapter provides basic information that you need to be aware of before attempting any of the product exercises. It includes the following information:

- Prerequisites. This section contains a list of requirements that you need to be aware of before using this product.
- Accessories. This section provides a brief overview of standard product accessories.
- Configuration. This section provides information on the allowable logic analyzer configurations for the exercises in this manual and software installation information.
- Care and Maintenance. This section provides basic cleaning information, static precautions, and information on obtaining replacement parts.

Prerequisites

The exercises and examples in this training manual are based on the following prerequisites and assumptions:

- To use this product, it is assumed that you are familiar with Microsoft Windows. The Tektronix Logic Analyzers operate on a Microsoft Windows platform. This training manual is not intended to teach you how to use Windows. You may want to refer to the Windows online help for information on using a Windows based product.
- It is assumed that you are familiar with the basics of logic analysis. This training manual is not intended to teach you logic analysis and basic digital circuit theory.
- It is assumed that you have some familiarity with the Tektronix Logic Analyzer Family application. Refer to the online help or to the printed documentation for details on using the Tektronix Logic Analyzer Family application.
- It is assumed that the Tektronix Logic Analyzer and any accessories are properly set up. For example, it is assumed that any oscilloscope probes that you intend to use with the DSO module are properly calibrated.

Accessories

The TLA 7QS QuickStart package comes with the following standard accessories:

- This training manual and the CD containing software setups for the exercises in training manual
- The TLA 7QS QuickStart training board with appropriate power adapter

Configuration

The following paragraphs list the configuration requirements for the Tektronix Logic Analyzer and for the training board.

Logic Analyzer

To complete all the exercises in this training manual you will need a Tektronix Logic Analyzer with a logic analyzer module (with probes), an oscilloscope (DSO) module (with probes), and a pattern generator module (with probes).

To complete the general purpose exercises, you can use any version of the logic analyzer module or any version of the TLA 600 logic analyzer. To complete the microprocessor exercises and the embedded software debug exercises, you will need a logic analyzer with 102 data acquisition channels or higher (for example, a TLA 7x3, TLA 7x4 module, TLA 6x3, or TLA 6x4).

You will need to install the TLA 7QS Exercise software on the logic analyzer. The exercise software consists of instrument setups and associated software for the individual exercises.

Complete the following steps to install the TLA 7QS Exercise software.

NOTE. *This version of QuickStart software is compatible with Tektronix Logic Analyzer software version 3.2 and above.*

Before continuing, remove any previous versions of QuickStart software by opening the Windows Explorer and deleting the Quick Start folder under C:\Program Files\TLA 700.

---

1. Insert the CD labeled *TLA 7QS QuickStart Software* in the CD ROM drive.
2. Click on the Windows Start button and select Run.
3. Enter the program name D:\QStart.exe in the dialog and click OK. (If the CD ROM drive has a different designation enter that letter instead of D).

   The logic analyzer will install the exercise software and place it in the following location on the hard disk: C:\Program Files\TLA 700\Quick Start.

4. Click the Unzip button in the dialog that appears.
5. Click OK when the dialog appears indicating that all files were installed successfully.
6. Click the Close button.
7. Click on the Windows Start button and select Run.
8. Enter the program name D:\Support\Setup.exe in the dialog and click OK.

   The logic analyzer will install the support software and place it in the following location on the hard disk: C:\Program Files\TLA 700\Supports\.
9. Click OK when the dialog appears indicating that all files were installed successfully.
10. Click OK and then remove the CD.

Training Board

The TLA 7QS QuickStart Training Board requires no special configuration procedures. Connect the probes to the training board as described in each chapter. Then connect the power adapter and apply power to the training board. The training board will execute the power-on diagnostics. When the power-on diagnostics are complete, select the program as defined in the individual exercises.

Care and Maintenance

The TLA 7QS QuickStart Training Board does not require scheduled or periodic maintenance. However to keep good electrical contact and efficient heat dissipation, keep the training board free of dirt, dust, and contaminants. When not in use, store the training board in the protective box.

Cleaning

Clean dirt and dust with a soft bristle brush. For more extensive cleaning, use only a damp cloth moistened with deionized water; do not use any other chemical cleaning agents.

Preventing Electrostatic Discharge

When handling the training board adhere to the following precautions to avoid damaging electronic components.

CAUTION. *Static discharge can damage semiconductor components on the training board.*

1. Minimize handling of the training board.
2. Transport and store the training board in the static protected container.
3. Discharge the static voltage from your body by wearing a grounded antistatic wrist strap while handling the training board.
4. Nothing capable of generating or holding a static charge should be allowed on the work station surface.
5. Handle the training board by the edges when possible.
6. Do not slide the training board over any surface.
7. Avoid handling the training board in areas that have a floor or work-surface covering capable of generating a static charge.

Obtaining Parts

Electrical and mechanical replacement parts are described in the *TLA 7QS Technical Reference Manual*. Refer to that manual for replacing and ordering parts and for any other service information.

General Purpose Exercises

General Purpose Exercises Setups

The following series of exercises use the Signal Sources section of the training board to demonstrate the timing analysis features of the Tektronix Logic Analyzer. You should read through this section before attempting any exercises in this training manual. The exercises rely on a single setup of the training board.

Hardware Setups

Connect the P6418 or P6417 probe to the C2/C3 connector on the logic analyzer module. You will use the same probe connections through the remainder of this chapter. If you have an oscilloscope module (DSO), connect an oscilloscope probe to Channel 1 of the DSO module.

Set up the P6417 Probes

Depending on the number of your logic analyzer probes, you may need to reconfigure your probes. The following sections provide basic probe setup information.

Removing Podlets

To complete the exercises in this section, you must remove and separate the P6417 probe podlets from the podlet holder. Refer to Figure 2–1 and open the podlet holder as shown and remove the podlets one at a time.

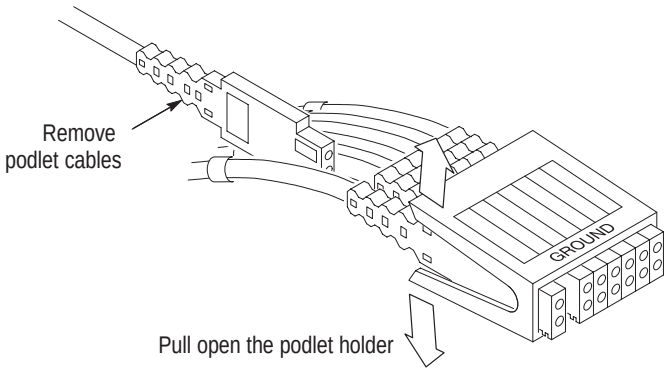


Figure 2–1: Removing probe podlets

Reinstalling Podlets

When reinstalling the podlets, make sure all the ground sides of the podlets line up with the ground side of the podlet connector. Arrange the podlets by their color-coded rings; use the key on the reverse side of the podlet holder to place the podlets in the correct channel order.

Connect the Probes

Complete the following steps to connect the logic analyzer probes to the training board. If you are using the P6418 probes, use the flying lead sets and refer to Figure 2–2 while connecting the probes; if you are using the P6417 probes, refer to Figure 2–3 while connecting the probes to the training board. Make sure that you connect the signal side of the probes to signal side of the square pins (ground signals are toward the rear of the training board as you face the LCD display).

1. Connect the clock channel (CK3) of the C2-C3 probe to J760 (CNTR CLK) on the training board.
2. Connect the C2:0, C2:1, C2:2 probe podlets of the C2-C3 probe to J860 on the training board.
3. Connect the Channel 1 oscilloscope probe to the unused J860 signal square pin labeled FF-Q.

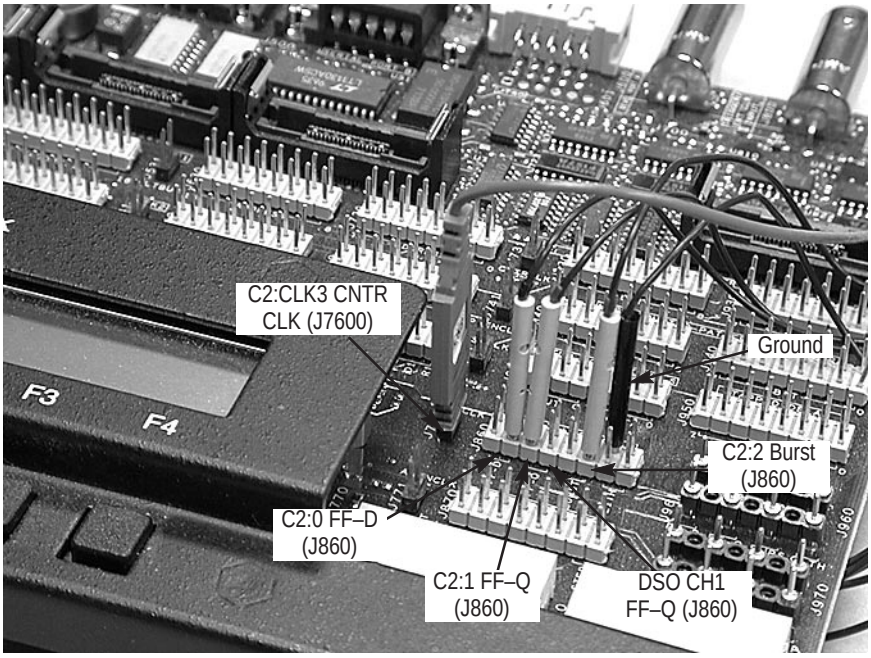


Figure 2–2: P6418 probe connections for the timing exercises

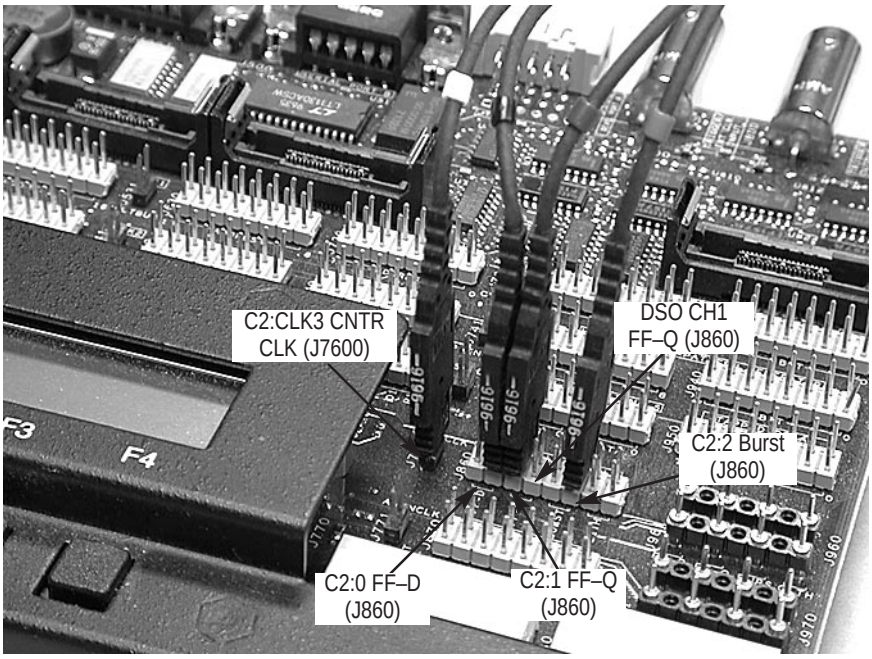


Figure 2–3: P6417 probe connections for the timing exercises

Start the Tektronix Logic Analyzer Family application and continue with the following steps. You will use the same hardware setups through the remainder of this chapter.

Load the Setups

Perform the following steps to load the timing exercises. All timing exercises are contained in the same folder (C:\Program Files\TLA 700\Quick Start\Hardware Analysis).

NOTE. *Each exercise contains two saved setup files. One file contains only setup information and requires you to capture live acquisition data to complete the exercises. The other file contains setup information and saved data. Use the saved data files to complete nearly every exercise without the need for acquiring live data.*

You can also use TLAVu to complete the exercises off-line without needing any acquisition hardware.

1. Select Load System from the File menu.
2. Open the Workbook folder and select the file as defined in the individual exercises (for example, load the 1-Capture and Trigger on a Glitch.tla file for Exercise 1).

   The logic analyzer display should look similar to Figure 2–4.

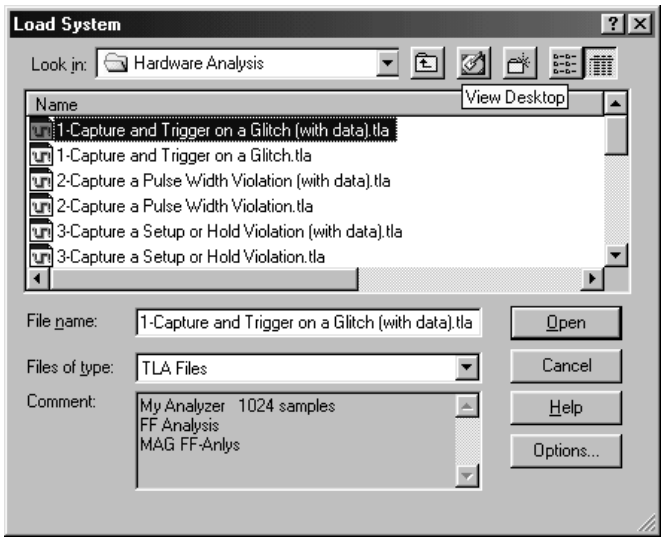


Figure 2–4: Load System dialog box

3. When prompted to load without saving the current system, click on Yes.
4. If you get a message telling you that the configuration in the file does not match your current hardware, click on OK. A new dialog box appears.
   a. Drag the "My Analyzer" icon (and the "My DSO" icon, if needed) from the top of the window to the shaded LA1 icon in the Current System window.
   b. Click on OK to continue with the exercise.

NOTE. *If the Load System Options dialog box appears and you want to view saved data, you may need to complete steps 5 through 8 to load a data window with saved acquisition data.*

To determine if you need to perform the following steps, click on one of the data window icons. If data appears in the data window, you can continue with the exercise. If no data appears, complete the following steps after you complete step 4 above.

---

5. Select Load Data Window from the Window menu; the Load Data Window dialog box displays.
6. Click the Browse button, navigate to the Hardware Analysis setup folder, and then double-click the setup for the current exercise.
7. Select the data window from the list in the Load Data Window dialog and then click OK; the Name Data Window dialog box displays.
8. Change the name of the data window (it is recommended that you only change one or two characters so the name is similar to the original data window).

Now you have a data window that displays saved acquisition data for the current exercise. If you want to continue with the exercise and capture live data, refer back to the original data window that was loaded with the setup.

Exercise Overview

These exercises focus on a metastable problem caused by driving the clock input of a flip-flop with a 50 MHz clock and the D input (FF-D) with an asynchronous 10 MHz data stream. Under these conditions the setup and hold times of the flip-flop are often violated. When the violation occurs, the output of the flip-flop goes into a potentially metastable state.

When the flip-flop is in a metastable state, the outputs are unpredictable. This can result in the flip-flop producing a glitch (the output pulse is shorter than normal), the outputs could briefly oscillate, or the outputs may not change at all. These type of failures are usually intermittent and can be difficult to find.

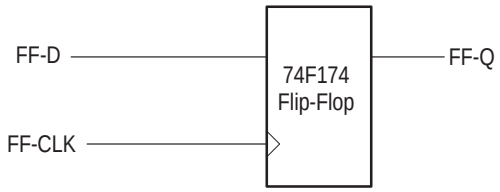
The key to isolating these kinds of timing faults is to trigger the logic analyzer when the Q output (FF-Q) of a flip-flop has a shorter pulse width than the period of the driving clock signal (FF-CLK). When an edge-triggered flip-flop works properly, the minimum time between changes on the Q output must be equal to or greater than the driving clock period.

The device used in these exercises is a 74F174 flip-flop with the specifications in Table 2–1. The training board provides connections to the flip-flop (see Figure 2–2 on page 2–2 or Figure on 2–3 page 2–3).

Table 2–1: 74F174 electrical characteristics

Parameter	Limits[1]
Setup time (t_S)	3.0 ns
Hold time (t_h)	1.0 ns
Propagation Delay (t_{PHL})	4.0 ns (min.) to 8.0 ns (max.)

1 **Some of the specifications, such as the propagation delay, may vary because of circuit design and load conditions.**



Triggering on a Glitch (Exercise 1)

In system design, glitches are annoying pulses that can be very difficult to detect and capture. The glitches can be caused by any number of conditions such as cross talk, race conditions, or timing violations.

Glitches can be extremely intermittent and hard to find. They are usually narrow pulses and require high-speed timing to see. Their effects on a system-under-test are often unpredictable.

This exercise shows how you can set up the Tektronix Logic Analyzer to capture a glitch on a specific channel or waveform. You will use a glitch detector as an event in the trigger machine. You can then use the MagniVu feature with the 2 GHz resolution to determine the cause of the glitch.

In this exercise, you will monitor the Q output of the flip-flop described in the previous section.

Load the Setup

1. Select Load System from the File menu.
2. Open the Hardware Analysis folder and load the system setup 1-Capture and Trigger on a Glitch.tla.

   The restored setup should look similar to Figure 2–5. You will not have a DSO icon if your logic analyzer does not contain a DSO module.

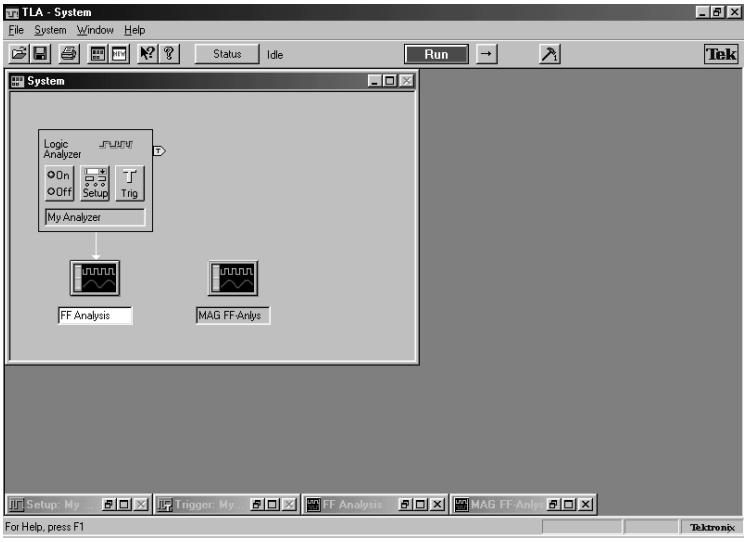


Figure 2–5: Restored system for Exercise 1

3. Click on the Run button to begin acquiring data.

   The logic analyzer will acquire and capture glitch data.

View the Resultant Data

1. Open the waveform data window labeled FF Analysis.
2. Note the glitches, displayed in red on the FF Q-OUT(0) waveform (see Figure 2–6).
3. Click on the Run button a few times to see if the glitch data changes.

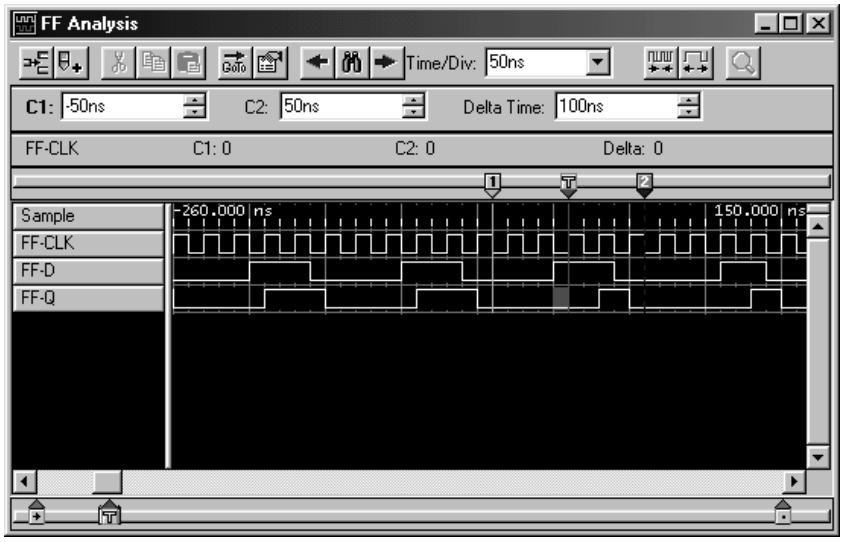


Figure 2–6: Glitch data for Exercise 1

Notice the red area to the left of the trigger indicator. These markings indicate the presence of a glitch on those signals. But what actually caused the glitches? Was it a single pulse or several pulses? To find out, you can use the MagniVu feature of the Tektronix Logic Analyzer to look at the data more closely.

4. Minimize the current data window and then open the waveform data window labeled MAG FF-Anlys.

Notice the greater resolution available with the 2 GHz sampling. The MagniVu feature provides you with the ability to make measurements with 500 ps resolution. The glitch data in Figure 2–6 now appears as a single narrow pulse (see Figure 2–7). The pulse only became visible using the MagniVu feature with 2 GHz sampling.

Look at the relationship of other channels using the MagniVu feature to determine the events that caused the glitch. The glitch is a pulse that appeared between clock edges (these pulses are often missed by basic logic analyzers).

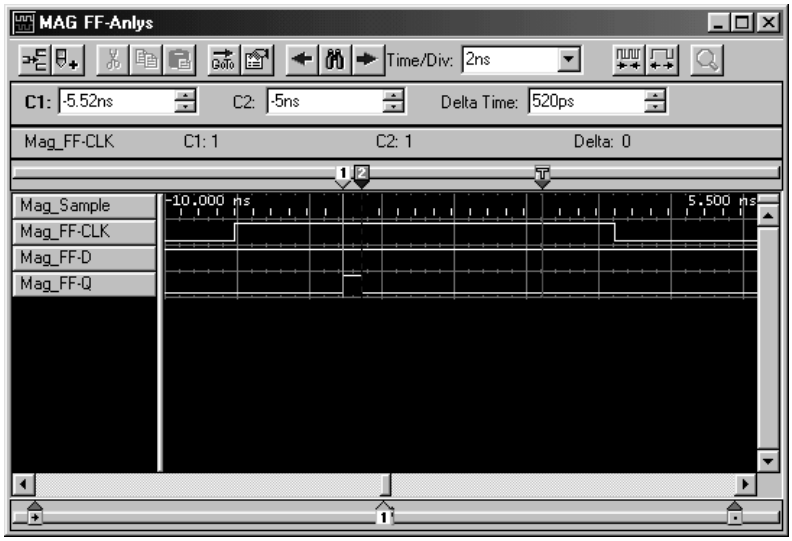


Figure 2–7: Waveform display with the MagniVu feature

The next few steps will show you how to measure the pulse width using the cursors in the waveform window:

5. Click on the Zoom In button (the right-most active button on the tool bar) or use the front panel horizontal SCALE or POSITION controls of the portable logic analyzer to zoom in on the pulse.

   a. Move the mouse pointer to the leading (rising) edge of the pulse (on the FF-Q channel) to the left of the trigger mark. Right-click the mouse and select Move Cursor 1 Here. Cursor 1 will be placed at the position of the pointer.

   b. Move the mouse pointer to the trailing (falling) edge of the pulse to the left of the trigger mark. Right-click the mouse and select Move Cursor 2 Here.

   c. Determine the actual pulse width by the readouts at the top of the display.

***NOTE**. Use the procedure in the above steps to move the cursors from anywhere outside the displayed information in the window to the mouse position. This saves time by not having to scroll through the data to find the cursors.*

The pulse in Figure 2–7 appears to be 500 ps wide (the pulse width on your logic analyzer may be different). Most logic analyzers do not have the resolution to measure such a narrow pulse. Using the MagniVu feature of the Tektronix Logic Analyzer, you can capture and identify very narrow pulses.

View the Trigger Setup

1. Minimize the data window and click on the Trig button on the logic analyzer icon to open the Trigger window. The Trigger window should be similar to Figure 2–8.

   The Trigger window shows the conditions that the logic analyzer uses to detect and capture the data of interest. The Trigger window is made up of trigger states with a series of action and event clauses.

NOTE. *For a detailed explanation of how to define the trigger setups and how the trigger setups work in this exercise, refer to* Define the Trigger Window *on page A–3.*

Figure 2–8: Trigger window for Exercise 1

2. Click on the If-Then button to view the clause definition for the trigger state.
3. Click on the Define Glitches button in the upper right corner of the dialog box.

The Glitch Detection dialog box defines which channels will acquire the glitches. In this exercise you are only interested in the Q output of the flip-flop. The dialog box should be similar to Figure 2–9.

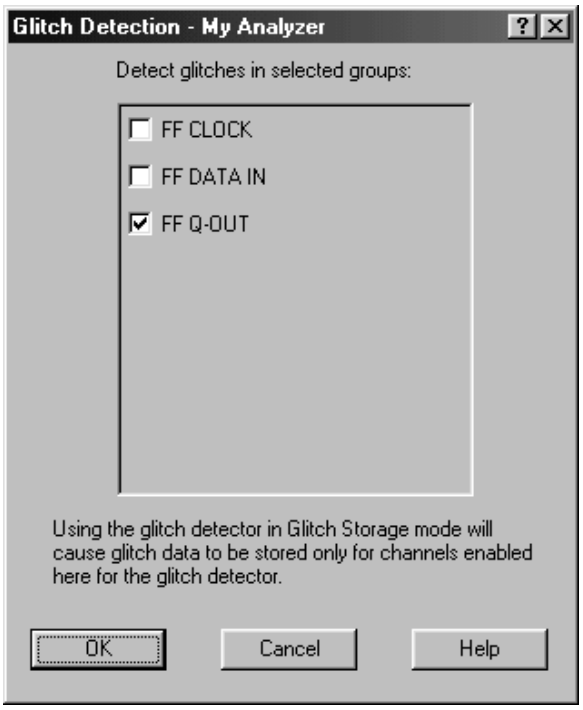


Figure 2–9: Glitch Detection dialog box for Exercise 1

The Tektronix Logic Analyzer uses high-speed glitch detectors to watch for glitches, store glitches, and trigger on glitches. These high-speed glitch detectors are available on all channels of the logic analyzer.

View the Channel Setups

1. Open the Setup window to see how the probe channels are defined and labeled for this exercise. Figure 2–10 shows an example of the Setup window; similar setups will be used for the logic analyzer module for all of the general purpose exercises.

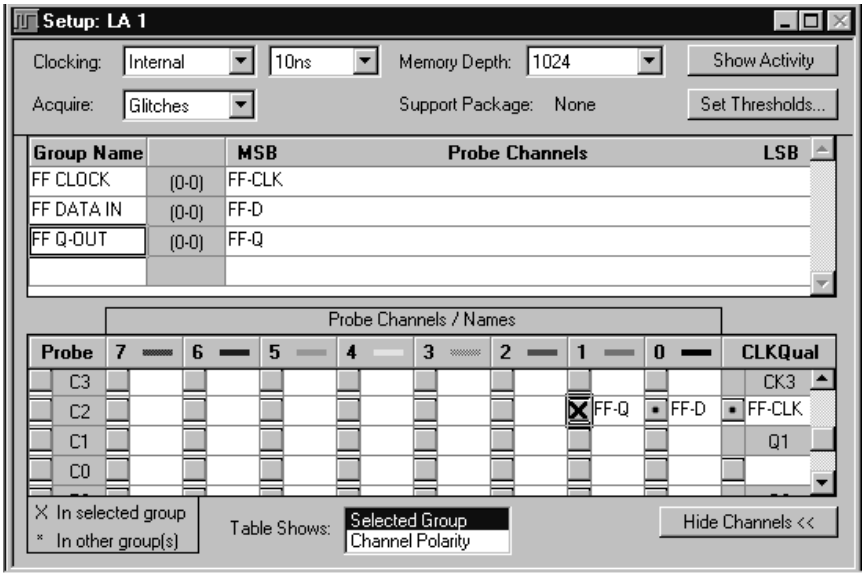


Figure 2–10: Channel setups for Exercise 1

After viewing the setups, minimize any open windows (except the System window), and continue with the next exercise.

Capture a Pulse Width Violation (Exercise 2)

The output of a flip-flop should never be less than the period of the driving clock. If the output of the flip-flop is stable for less than the period of the driving clock, a timing violation occurs. You can use the logic analyzer to monitor a single data channel and measure the actual pulse width of the flip-flop output. In the following exercise you will set up the logic analyzer to trigger if the Q-output pulse of a flip-flop is less than or equal to 16 ns.

Load the Setup

1. Select Load System from the File menu.
2. Load the following saved system:
   C:\Program Files\TLA 700\Quick Start\Hardware Analysis\
   2-Capture a Pulse Width Violation.tla.

The restored system should look similar to Figure 2–11. You may have a DSO icon if your logic analyzer contains a DSO module.

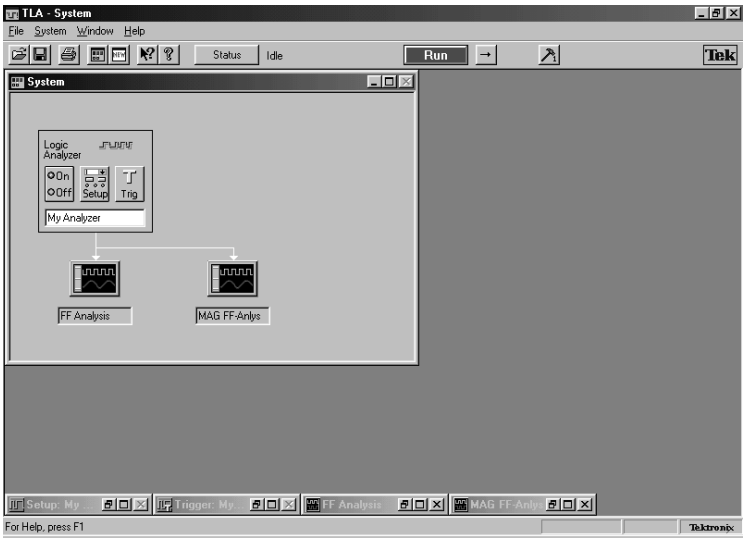


Figure 2–11: Restored system for Exercise 2

3. Click on the Run button to begin acquiring data.

View the Acquired Data

1. Open the view labeled FF-Analysis.

The logic analyzer monitored the Q-output signal. As long as the pulse width was greater than 16 ns, the logic analyzer did not trigger. However, when the pulse width was less than 16 ns, the logic analyzer triggered. Your waveform display should look similar to Figure 2–12.

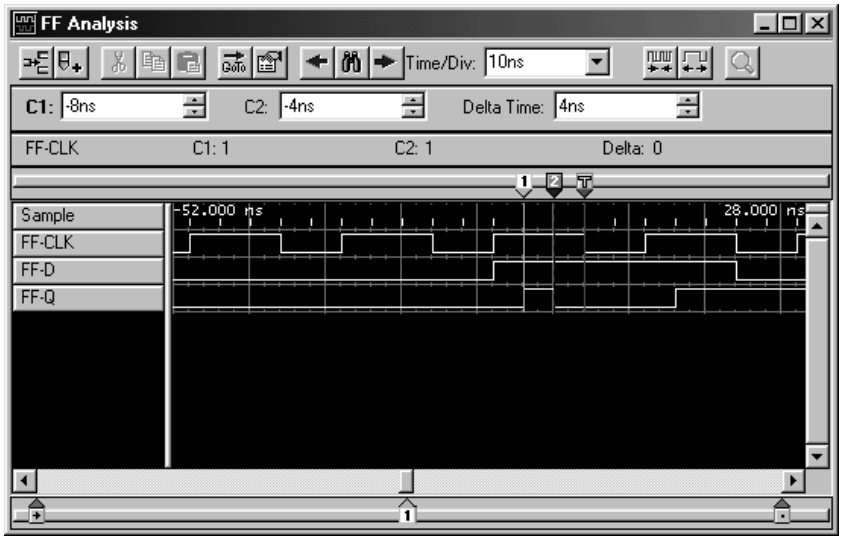


Figure 2–12: Timing waveform display for Exercise 2

2. Use the cursors to measure the pulse width:

   a. Move the mouse pointer to the leading (rising) edge of the pulse (on the FF-Q channel) to the left of the trigger mark.

   b. Right-click the mouse and select Move Cursor 1 Here. Cursor 1 will be placed at the position of the pointer.

   c. Move the mouse pointer to the trailing (falling) edge of the same pulse.

   d. Right-click the mouse and select Move Cursor 2 Here.

NOTE. *Use the procedure in step 2 to move the cursors from areas outside the window to the mouse position. This saves time by not having to scroll through the data to find the cursors.*

3. Determine the actual pulse width by the Delta Time readout at the top of the display.

The sample rate of the conventional acquisition system is 250 MHz (4 ns) and the pulse can only be resolved in 4 ns increments. Therefore, a captured pulse is always represented in 4 ns increments (4 ns, 8 ns, 12 ns ...). The sample rate of the MagniVu feature is 2 GHz, providing 500 ps resolution for your measurements.

4. Minimize the current Waveform window and then open the Waveform window labeled MAG_FF-anlys (see Figure 2–13). Notice the greater resolution available with the 2 GHz sampling rate. Using 2 GHz sampling (MagniVu acquisition), you can take measurements with 500 ps resolution.

5. Repeat the measurement performed in step 2.

The pulse in Figure 2–12 appears to be approximately 4 ns wide because of the 250 MHz resolution. However, by using the higher resolution available with the MagniVu feature, the actual pulse width is closer to 1.5 ns (±500 ps) as shown in Figure 2–13. No pulse can appear to be shorter than the clock period in a logic analyzer.

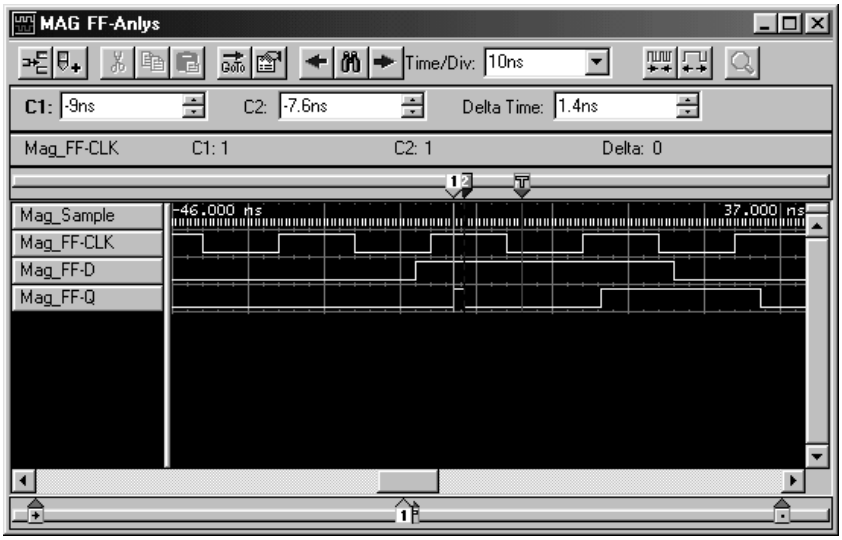


Figure 2–13: Timing waveform using MagniVu acquisition

View the Trigger Program

1. To understand how the logic analyzer triggered on the signal, view the Trigger window. Your trigger should be similar to the setup shown in Figure 2–14. (You will need to scroll or resize the Trigger window).

2. To see the details of the trigger setup, click on the If-Then button to display the details of the Clause Definition dialog box. You may want to examine other areas within the Clause Definition dialog box for specific information on the channel definitions.

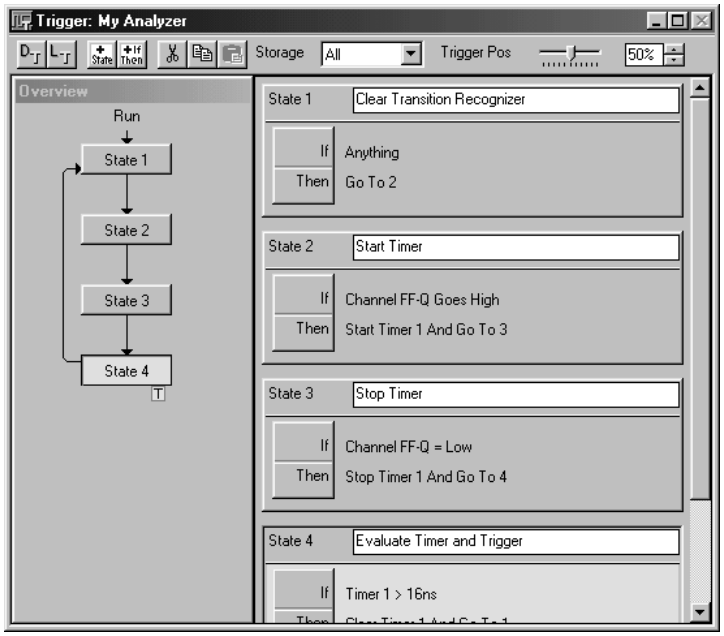


Figure 2–14: LA Trigger window for Exercise 2

The logic analyzer ignored all the pulses on Channel FF-Q that were greater than or equal to the value in the Time field (16 ns). The logic analyzer triggered only when the conditions in the Trigger window were met.

***NOTE**. For a detailed explanation of how to define the trigger and how the trigger works in this exercise, refer to* Define the Trigger Window *on page A–3.*

View the Channel Setups

1. Open the Setup window to see how the probe channels are defined and labeled for this exercise. Figure 2–15 shows an example of the Setup window.

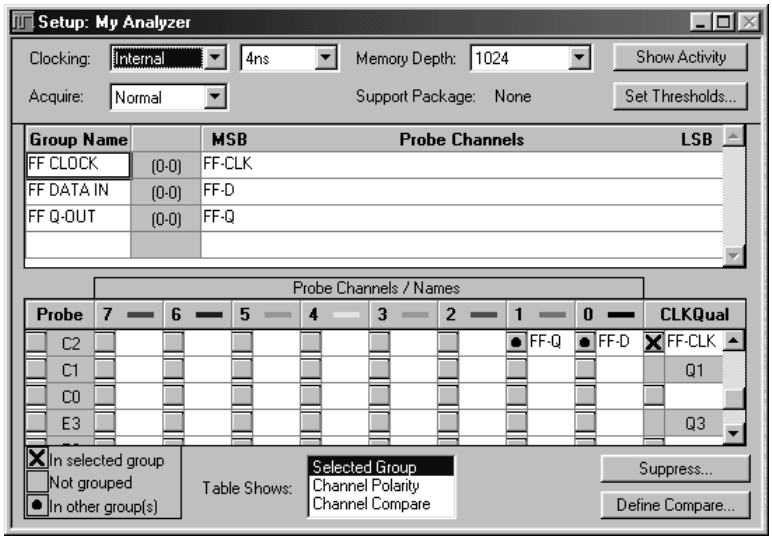


Figure 2–15: Channel setups for Exercise 2

2. After viewing the setup, proceed with the next exercise.

Capture a Setup and Hold Violation (Exercise 3)

Setup and hold violations can be difficult to detect and capture even with a high-speed timing analyzer. It may even be impossible to detect the violations with a high-channel count, general purpose logic analyzer. However, with the special setup and hold triggering of the Tektronix Logic Analyzer, capturing setup and hold violations becomes an easy task.

Load the Setup

1. Select Load System from the File menu.
2. Load the saved system 3-Capture a Setup or Hold Violation.tla.
3. Click on the Run button to begin acquiring data.

View the Resultant Data

1. Open the waveform data window labeled MAG_FF-Anlys.
2. Measure the setup and hold times nearest the trigger mark (measure from the rising edge of the clock to the data transition of the FF-D channel).

   Notice the excellent resolution available with the MagniVu feature using the 2 GHz sampling. The waveform display should look similar to Figure 2–16.

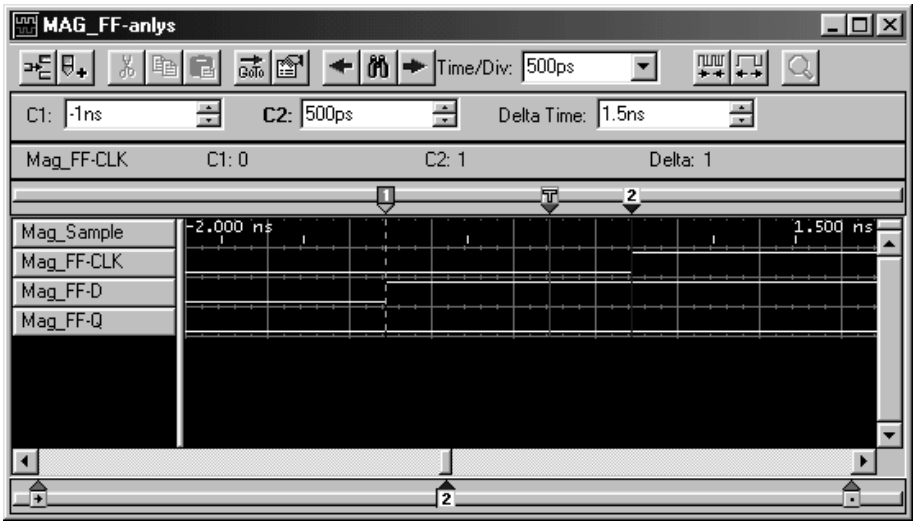


Figure 2–16: Timing waveform display for Exercise 3

3. Click on the Run button several times to see how the timing violation varies with each data acquisition.

   The logic analyzer actually evaluates every clock for a setup and hold violation. While the logic analyzer monitors millions of events, it captures only those that fail the setup and hold requirements.

4. Using the data just acquired, select an adjacent clock edge and use the cursors to measure the propagation delay of the flip-flop (measure from the rising edge of the FF-CLK channel to the FF-Q channel transition). If necessary, use the Zoom Out button on the Tool bar to display a wider range of samples to take the measurement.

   Does the delay exceed the specifications of the flip-flop? (See Table 2–1 on page 2–6.)

View the Trigger Setup

To understand how the logic analyzer captured the setup and hold violation, view the setups in the Trigger window.

1. Minimize the Waveform window and open the Trigger window.
2. Click on the If-Then button to view the clause definition.

   Note the Define Violation button in the upper right corner of the clause definition.
3. Click on the Define Violation button to open the dialog box.

   The dialog box lets you define the setup and hold parameters for the channels. The dialog box should be similar to Figure 2–17.

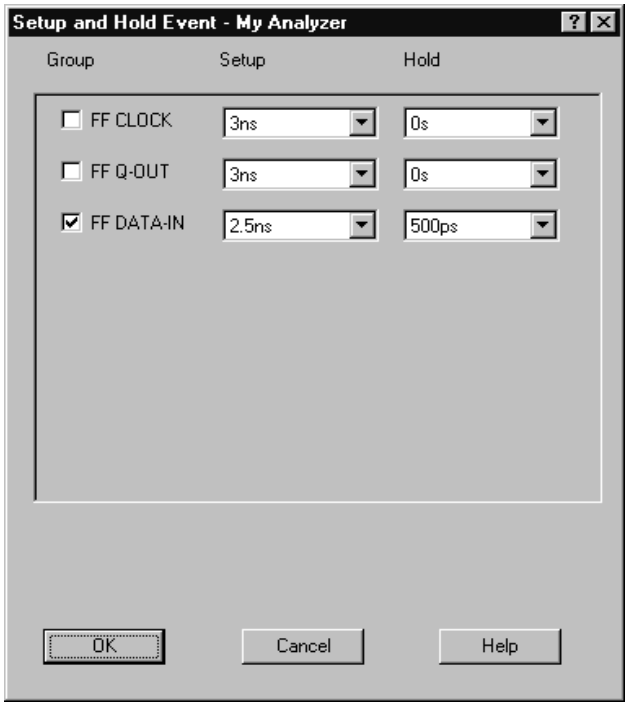


Figure 2–17: Define Violation dialog box for Exercise 3

4. Close the Trigger window and click on the Setup button in the Logic Analyzer icon.

   Note that the clocking is set up to use CLK3 as the external clock source. To trigger on a setup and hold violation, you must use either External Clocking or Custom Clocking (used with microprocessor support).
5. After viewing the setups, minimize any open windows (except the System window), and continue with the next exercise.

Counting Setup and Hold Violations (Exercise 4)

In addition to detecting setup and hold violations, you may want to count how many violations occur over a period of time. This can help you decide if your electronic circuit needs to be redesigned depending on the actual number of violations. The frequency of some violations may not cause any noticeable problems. However, some violations may have a noticeable effect on your circuitry and the problems must be eliminated.

In this example, you will use a counter in the logic analyzer to track the actual number of setup and hold violations. You will also use a timer to let you know the number of violations occurring over a period of time. When a specific number of violations occur (100 million), you can trigger the logic analyzer.

Load the Setup

1. Select Load System from the File menu.
2. Load the system setup 4-Counting Setup and Hold Violations.tla.
3. Click on the Status button to the left of the Run button in the System window to open the Status Monitor.

   You can use the Status Monitor to view the status of the logic analyzer while it is waiting for the trigger conditions to be met. The Status Monitor shows the status of any counters, timers, or other flags set up in the Trigger window.
4. Click on the Run button to begin acquiring data (if necessary, move the Status Monitor dialog box to access the Run button).

View the Status Monitor

Watch the count in the Status Monitor as it increases to the terminal count of 100 million. The Tektronix Logic Analyzer has a zero (0 ns) latency that allows it to count every failure without missing a single one. In some situations a failure could happen once a minute, once an hour, and even less than once a day. A critical application in a real time system with one failure a week could be disastrous. The Tektronix Logic Analyzer can help find these types of failures.

The Status Monitor should look similar to Figure 2–18.

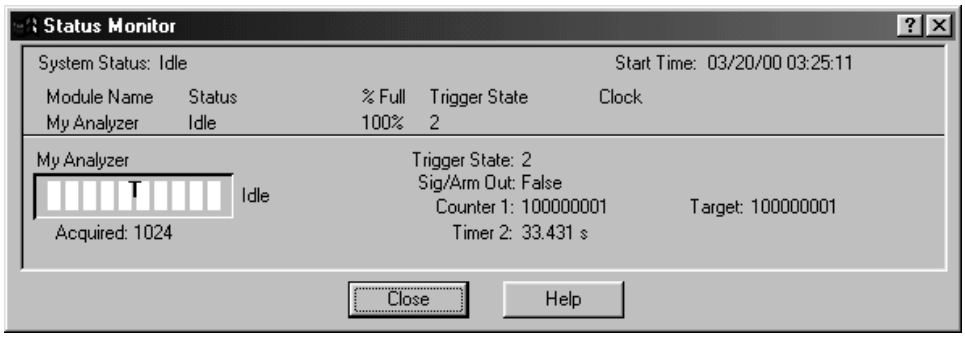


Figure 2–18: Status Monitor for Exercise 4

NOTE. *If you do not want to wait for the logic analyzer to reach the terminal count, you can click on the Stop button at any time.*

Note the use of timers in Figure 2–18. You can use a timer to keep track of how many violations occurred over a period of time. In Figure 2–18, you can see that the logic analyzer detected and counted over 100,000,000 violations in just under 34 seconds. By using a timer on intermittent violations, you can see how many violations actually occurred over certain number of hours or days.

View the Setups

Close the Status Monitor and open the Trigger window. Your trigger setups should be similar to Figure 2–19.

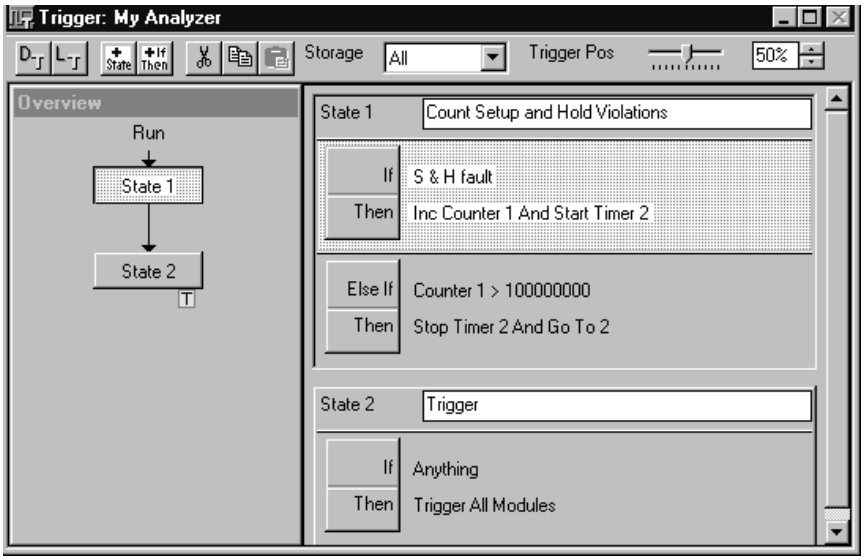


Figure 2–19: LA Trigger window for Exercise 4

The Trigger window is set up to detect setup and hold violations. When the logic analyzer detects the first violation, it starts a counter and a timer. The counter counts the number of violations while the timer keeps track of the time since the first violation.

The logic analyzer continues to count the violation until it reaches a terminal count. When the terminal count is achieved, the logic analyzer stops the timer and triggers all modules. The final counter and timer values are displayed in the Status Monitor.

After viewing the setups, minimize any open windows (except the System Window), and continue with the next exercise.

Capturing Data Bursts Using Transitional Storage (Exercise 5)

Often times you need to capture a series of events that are separated by wide time intervals. Using conventional storage, your logic analyzer quickly fills with redundant data thereby limiting your overall time window. What is needed is a method where your logic analyzer stores data only when there is a transition thereby avoiding redundant data storage. A logic analyzer with transitional storage effectively extends your acquisition memory by lengthening the overall time window covered by the acquisition. Using the Tektronix Logic Analyzer's transitional storage, you can sample at up to 4 ns on all channels so that events that are seconds, minutes, hours, or even up to 6.5 days apart are still captured with 4 ns timing resolution. You can then use the MagniVu 500 ps timing resolution for a closer look at the data.

In the following exercise, you will set up the logic analyzer to capture a repetitive signal consisting of four groups of eight 20 ns (50 MHz) pulses separated by 1.3 ms of inactivity using both conventional and transitional storage techniques. You will then contrast the difference in an effective overall time window.

***NOTE**. Make sure that the C2:2 channel of the acquisition probe is connected to the Burst signal on J860 on the training board. If necessary, refer to* Connect the Probes *on page 2–2 for probe connection information.*

Load the Setup

1. Select Load System from the File menu.
2. Load the system setup:
   5a-Capture Data Bursts Using Conventional Storage.tla.

   The restored system should look similar to Figure 2–20.

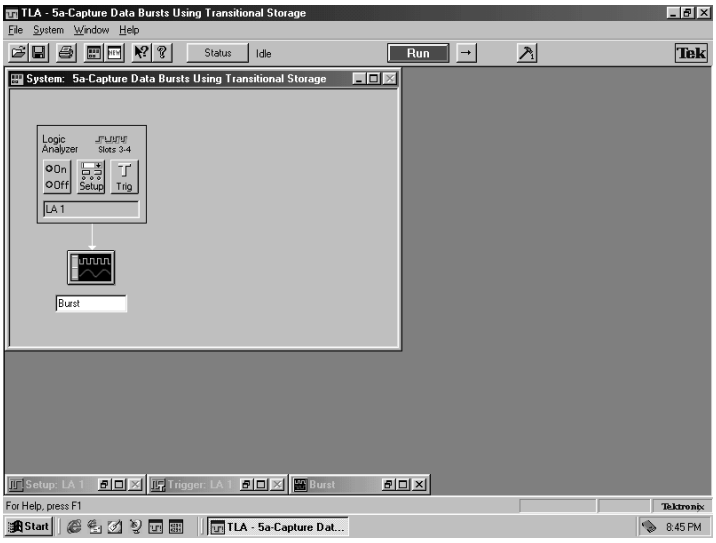


Figure 2–20: Restored system for Exercise 5

View the Channel Setups

1. Open the Setup window; it should look similar to Figure 2–21. The sampling is set to 4 ns and the memory depth is set to 1 K for the Burst signal.

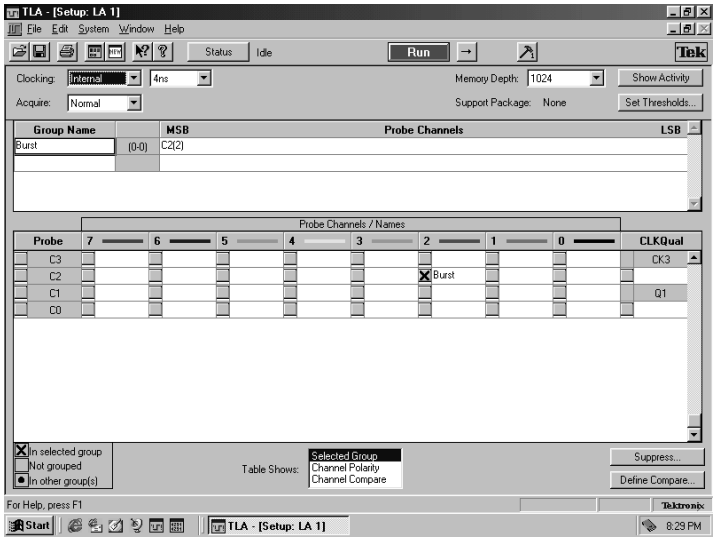


Figure 2–21: Setup window for Exercise 5

2. Minimize the Setup window.

View the Trigger Program Using Conventional Storage

1. Open the Trigger window; your trigger setups should look similar to Figure 2–22.

The logic analyzer is set to trigger on the first data burst of the Burst signal using conventional storage (transitional storage is disabled).

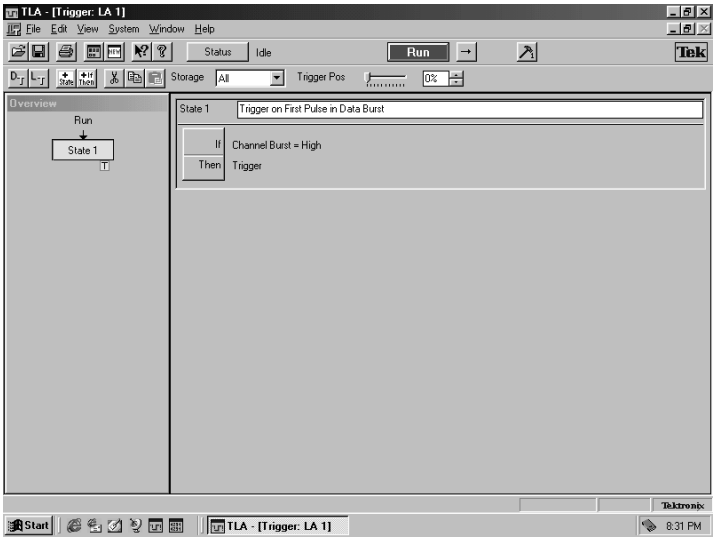


Figure 2–22: Trigger window using conventional storage for Exercise 5

2. Minimize the Trigger window.

View the Acquired Data Using Conventional Storage

1. Open the waveform window labeled Burst and ensure that it is maximized.
2. Click the Run button to begin acquiring data. The acquired data should look similar to Figure 2–23.

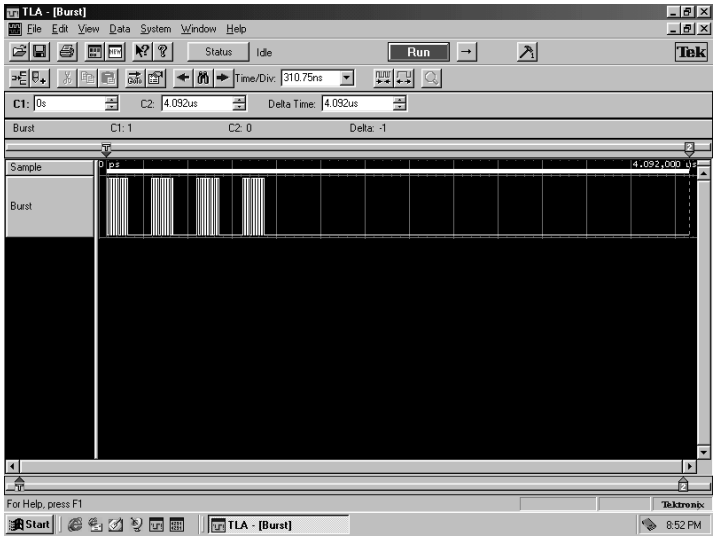


Figure 2–23: Timing waveform using conventional storage for Exercise 5

There will be four groups of eight 20 ns (50 MHz) pulses following the trigger. The next four data bursts appear approximately 1.31 ms later with no signal activity in between. However, since you have selected 1 K memory depth, you can only see the first burst of four groups of eight pulses. You don't have the depth to see the next burst which is approximately 1.31 ms away.

3. Right-click the mouse in the waveform window and Select Zoom > All (it may take a few seconds for the right-click menu to appear).

Notice that the overall time window is only approximately 4.1 µs.

Observe that there is a tremendous amount of redundant data being stored (all zeros). What if you only stored when the data transitioned thereby widening the time window to see the next data burst and yet maintain the 4 ns timing resolution?

4. Minimize the waveform window.

View the Trigger Program Using Transitional Storage

1. Select Load System from the File menu.
2. Load the system setup:
   5b-Capture Data Bursts Using Transitional Storage.tla.
3. Go to the Trigger window. It should look similar to Figure 2–24.

The logic analyzer is set to trigger on the first data burst of the Burst signal with transitional storage enabled. Note that the Storage drop-down menu has Transitional selected. Also note that the Change Detect clause under Storage is open and shows that the Burst group is selected.

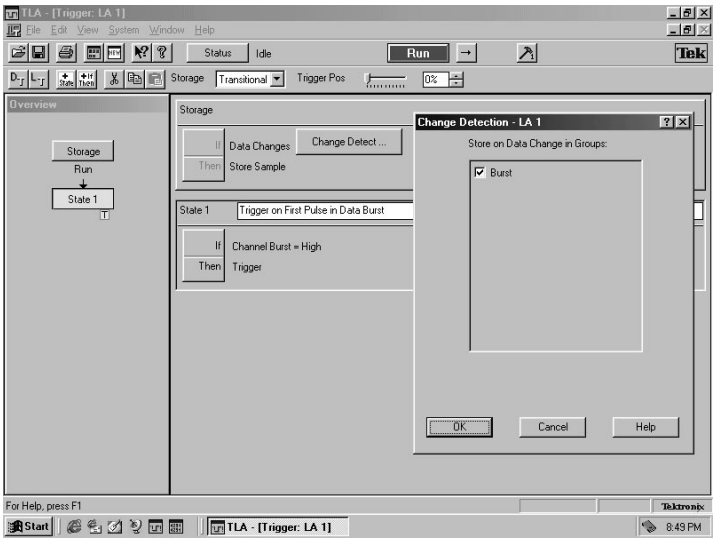


Figure 2–24: Trigger window using transitional storage for Exercise 5

4. Minimize the Trigger window.

View the Acquired Data Using Transitional Storage

1. Open the waveform window labeled Burst and ensure that it is maximized.
2. Click the Run button to begin acquiring data.
3. Right-click the mouse in the waveform window and Select Zoom All (it may take a few seconds for the right-click menu to appear).
4. The waveform window should look similar to Figure 2–25.

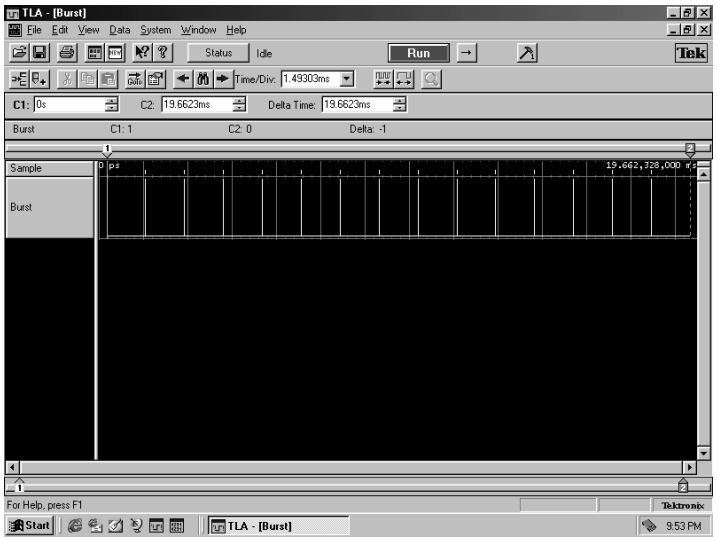


Figure 2–25: Timing waveform using transitional storage for Exercise 5

Notice that instead of one burst of four groups of eight pulses, the memory is filled with 16 bursts of the four groups of eight pulses and that the time window changed from approximately 4.1 μs to 19.7 ms.

For the Burst signal, this represents an overall time window that is almost 4800 times greater using transitional storage compared to conventional storage!

5. Change the Time/Div to 100 ns.
6. Right-click the mouse, and select Go To, followed by Go To to display the Go To dialog box.
7. Double-click System Trigger in the Go To dialog box.
8. Center the four groups of eight pulses so that the waveform window appears similar to Figure 2–26 (use the horizontal scroll bars in the waveform window or use the front-panel Horizontal Position control on the TLA600 or on the portable mainframe).

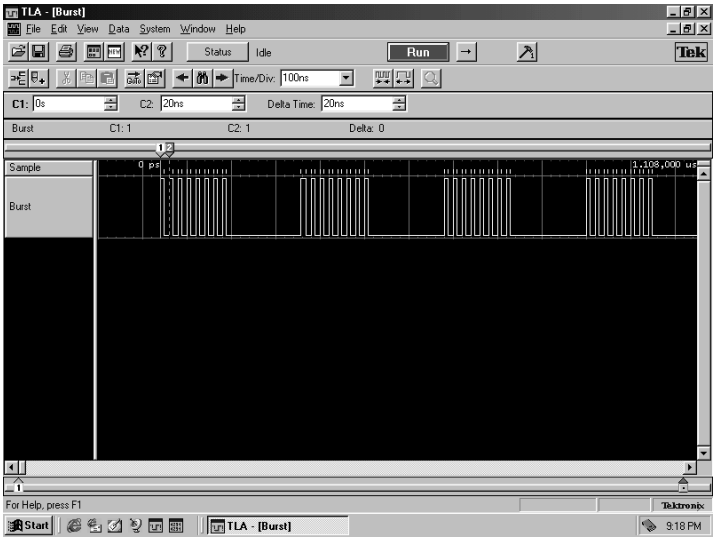


Figure 2–26: Waveform data at 100 ns Time/Div

9. Move Cursor 1 to the leading edge of the first pulse in the burst.

10. Move Cursor 2 to the leading edge of the second pulse in the burst.

Note that the periods is 20 ns (50 MHz) and the the data was acquired with 4 ns resolution.

11. To obtain 500 ps timing resolution, add the MagniVu version of the Burst signal by doing the following steps:

a. Click the Add Waveform button (the left-most button) on the Waveform window tool bar to display the Add Waveform dialog box.

b. Select LA 1 - MagniVu from the Data Source list at the top of the dialog box.

c. Select Sample and then click the Add button on the top right side of the dialog box.

d. Click on the plus sign to the left of Mag_Burst, select Mag_Burst(0), and then click on the Add button again.

e. Click the Close button.

12. Select the Mag_Burst(0) waveform and change the vertical size using the Vertical Size knob until the waveform window appears similar to Figure 2–27. (You can also select the waveform, right-click the mouse and then select Properties. Adjust the Height selection in the dialog box.)

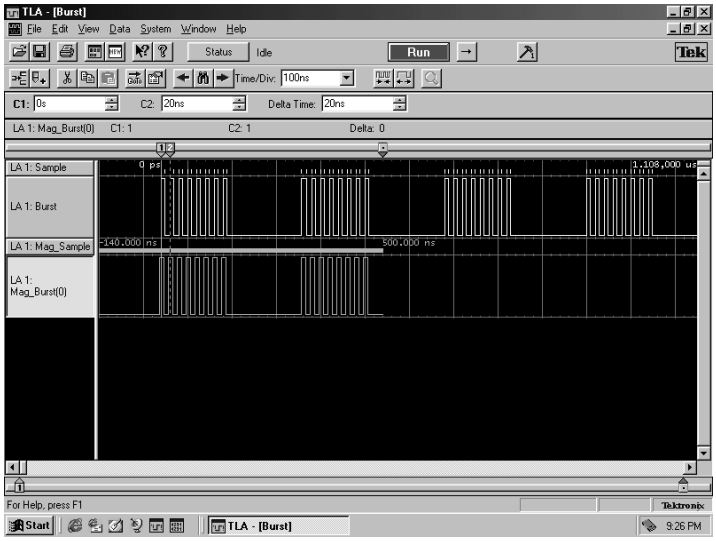


Figure 2–27: Timing waveform using transitional storage with MagniVu for Exercise 5

You can now view the Burst signal with eight times better resolution (500 ps timing resolution).

Transitional storage also works with synchronous or external clocking; that is why transitional storage is a trigger storage function. Also note that the maximum time window with transitional storage enabled is highly dependent on the data rate.

Optional

Set the memory depth of the logic analyzer to the maximum depth. Use the cursors to observe the maximum time window difference between conventional and transitional storage techniques. For example, Table 2–2 shows the number of data bursts and overall time windows at the specified memory depths.

Table 2–2: Relationship between the number of data bursts and overall time windows

Memory depth using transitional storage	Memory depth using conventional storage	Overall time window	Number of data bursts captured
256 KB	1.2 GB	5.4 seconds	44, 886
4 MB	19.2 GB	1 minute, 26 seconds	715, 684

You can find the overall time window in the upper, right corner of the waveform window on the Sample group. You should perform a Zoom-All operation so that the entire acquisition memory is displayed. You can also monitor the progress of the amount of acquisition memory by opening the Status Monitor dialog box (click the Status button to the left of the Run button in the System window).

Conclusion

With transitional storage, you can quickly capture intermittent events which dramatically extends the time window covered by the logic analyzer. You can then use the 500 ps MagniVu timing resolution to obtain greater timing detail.

Using the DSO to Trigger the Logic Analyzer on a Runt Pulse (Exercise 6)

NOTE. *This exercise can only be completed with a Tektronix Logic Analyzer with a digitizing oscilloscope module (DSO).*

Runt pulses can be the result of the same conditions that cause a glitch (described in the first exercise). However, runt pulses can also be due to other system problems such as power supply dropouts, bus contention, or high resistance connections. Runt pulses are serious problems in the design of digital systems. The digitizing oscilloscope (DSO) adds the flexibility to your Tektronix Logic Analyzer to detect events of more analog characteristics than using a logic analyzer alone.

A runt pulse is defined as a pulse with the amplitude greater than the lower threshold voltage, but less than the higher threshold voltage. A runt pulse can also be defined in terms of pulse width (that is, any pulse width equal to or greater than a specified pulse width). The logic analyzer may not be able to detect the runt pulse by itself. However you can set up the DSO to look for a runt pulse and then trigger the logic analyzer.

Load the Setup

1. Select Load System from the File menu.
2. Open the Hardware Analysis folder and load the saved system 5-DSO Captures a Runt and Triggers LA.tla.

   Make sure that you restore both the My DSO and My Analyzer setups.
3. Click the On button in the MY DSO icon.

   You can enable or disable a module from a setup by clicking the On or Off button in the module icon. The DSO module had been disabled in the previous exercises; to enable it, you must click the On button.

View the Resultant Data

1. Open the waveform data window labeled FF-Analysis.
2. Note the Repetitive Run button to the right of the Run button.

   The button should have a loop as shown in the illustration below. The Repetitive Run button lets you acquire data continuously until you click on the Stop button or until you click on the Repetitive Run button to change it to the Single Acquisition mode (default setting).

3. Click on the Run button to begin acquiring data.

   The logic analyzer will acquire data continuously.
4. Notice how the data appears with each acquisition.

   The data is different at various acquisitions because of when the logic analyzer sampled the data. You can use the Repetitive Run mode to see how often the data changes and to help you decide when you have a good data sample to continue your measurements.
5. Click on the Stop button.
6. If the waveform data window doesn't show the data that you want to measure, acquire another sample and then click on the Stop button again.

   The waveform data window should look similar to Figure 2–28.

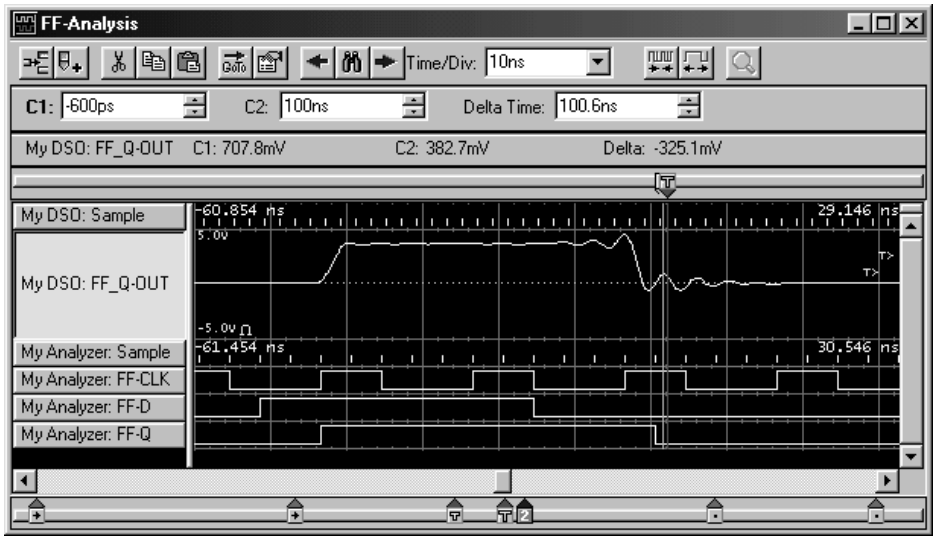


Figure 2–28: Timing waveform display for Exercise 6

7. Use the cursors to measure width and voltage magnitude of the runt pulse.

To measure the voltage levels, move cursor 1 over the peak of the runt pulse, then click on the My DSO:FF_Q-OUT waveform label. The voltage level for the captured runt pulse will appear on the measurement bar, next to C1.

8. Refer to the logic analyzer data and view the pulse.

Depending on the amplitude of the runt pulse, the logic analyzer may not have captured the runt pulse (see Figure 2–28).

You can use the high resolution mode to see the runt pulse more often. Add a new waveform using the MagniVu feature by completing the following steps:

9. Position the mouse cursor over the waveform My DSO:FF_Q-OUT and right-click on the mouse.

10. Select Add Waveform from the popup menu.

11. In the Data Source field, select My Analyzer-MagniVu.

12. Click on By Group and then click on the plus sign next to the Mag_FF Q-OUT waveform.

13. Select the line containing the signal name Mag_FF-Q.

14. Click on Add and then Close.

15. Click on Run and notice the difference in the waveforms when the logic analyzer captures the runt pulse. Also notice how the logic analyzer data correlates with the DSO data.

NOTE. *Because of minor differences in the training boards, you may have perform several acquisitions to see the runt pulse captured by the logic analyzer.*

View the Trigger Setups

1. Close the waveform data window and click on Setup in the DSO icon.

   The DSO Setup window shows the setups for each vertical input channel, horizontal channel information, and trigger information. Your DSO setups for Channel 1 should be similar to Figure 2–29.

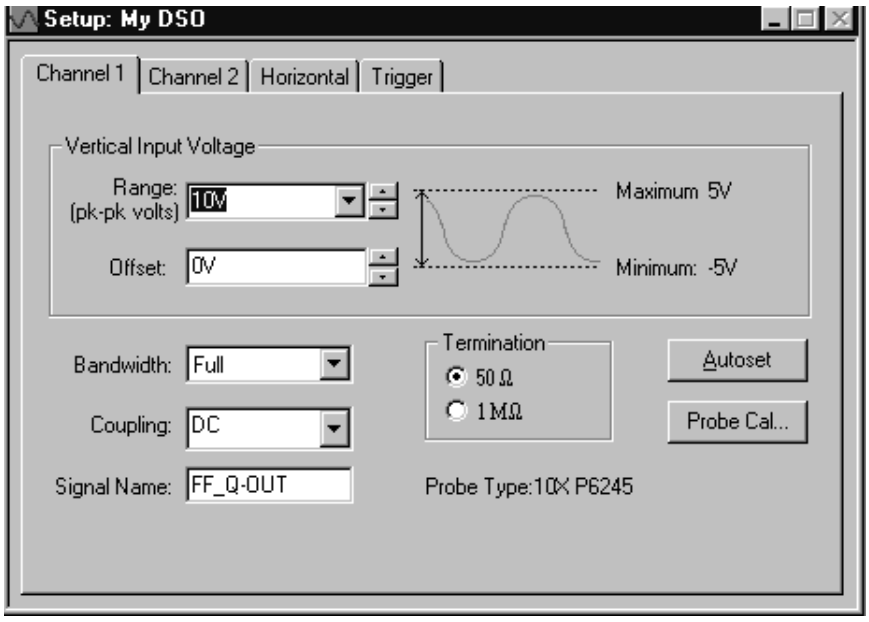


Figure 2–29: DSO channel 1 setups for Exercise 6

2. Click on the DSO Trigger tab. The DSO Trigger window should be similar to Figure 2–30.

   The DSO Trigger shows the trigger setups for the DSO. For this exercise, the DSO is looking for a runt pulse within the specified parameters. When the DSO detects a runt pulse, it triggers the logic analyzer module.

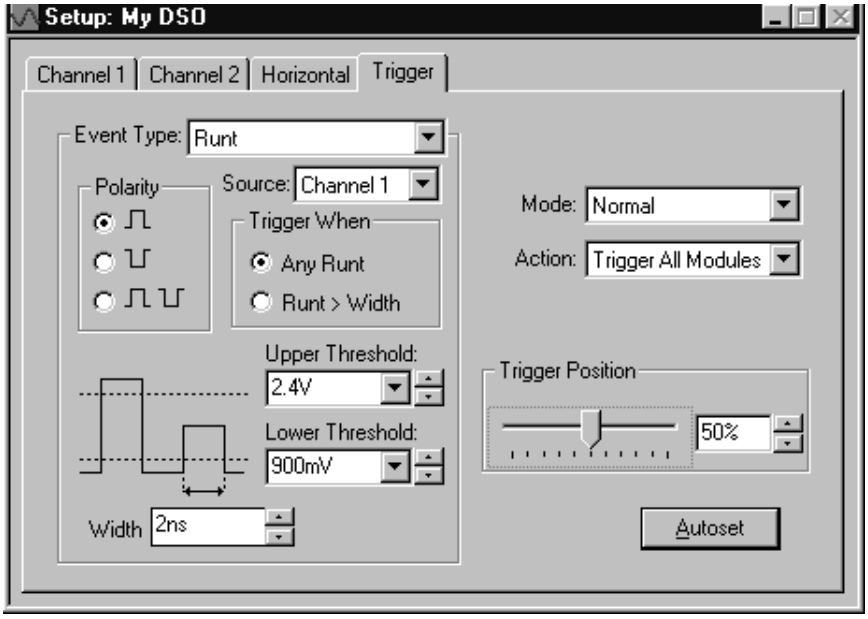


Figure 2–30: DSO trigger setups for Exercise 6

3. Now look at the logic analyzer setups.

Note that in this exercise the logic analyzer did not specify a trigger condition. In other words, you used the DSO to look for a trigger event and then display the DSO data and logic analyzer data without specifying any special logic analyzer trigger details.

4. After viewing the setups, minimize any open windows (except the System Window), and continue with the next exercise.

Using the Logic Analyzer to Trigger the DSO (Exercise 7)

NOTE. *This exercise can only be completed with a Tektronix Logic Analyzer with a digitizing oscilloscope module (DSO).*

With the high speeds of digital circuitry, design engineers are no longer able to rely on circuits working perfectly because they are logically or functionally correct. Analog characteristics of digital signals can become critical. Phenomenons such as reflections caused by unterminated signals or stubs can cause serious design problems.

Today's high-speed digital designer needs the ability to closely examine analog characteristics to determine the actual cause of failures. The ability to use a logic analyzer to trigger on a failure and then to analyze the analog characteristics of the signals involved in causing such failures can be critical in providing a correct solution for the problem.

This exercise uses setups similar to Exercise 1. The logic analyzer is set up to look for a glitch using the MagniVu feature.

Load the Setup

1. Select Load System from the File menu.
2. Load the system setup 6-LA Triggers DSO.tla.
3. Click on the Run button to begin acquiring data.

   The logic analyzer will acquire and capture glitch data.

View the Resultant Data

1. Open the waveform data window labeled FF Analysis. The window should look similar to Figure 2–31.

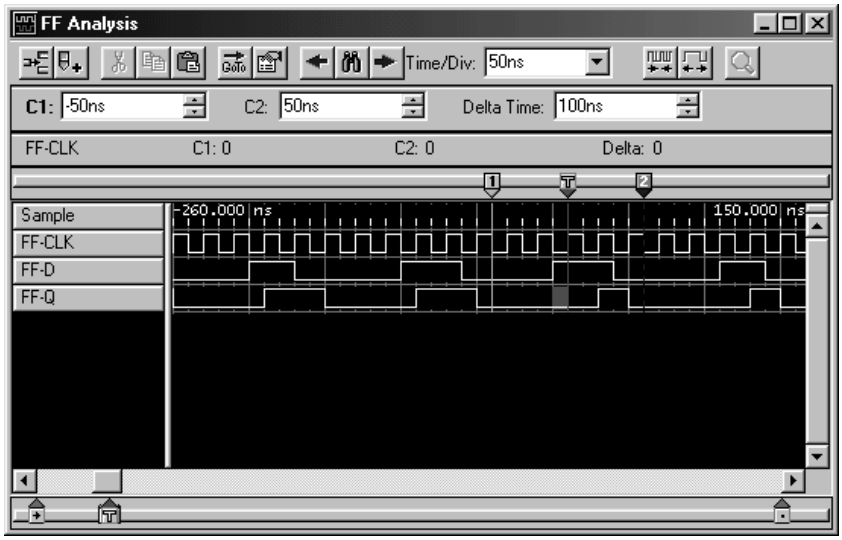


Figure 2–31: Waveform display for Exercise 7

The logic analyzer captured a glitch, but you really want to know more about the glitch. You want to know what caused the glitch and what was the actual signal amplitude. To do this you want to look at the signal with an oscilloscope.

You can easily add an oscilloscope waveform to the current waveform display and correlate the data.

2. Move the mouse cursor over the FF–Q waveform and right-click the mouse.
3. Select Add Waveform from the popup menu.
4. In the Data Source field, select DSO 1.
5. Select Channel 1.
6. Click on Add and then Close.

   The DSO waveform is added to the view.
7. Click on the Run button to acquire the data.

Figure 2–32 shows the resultant data (your data may be different). Notice how the glitch correlates to the signal acquired by the DSO. By using a logic analyzer side-by-side with a DSO, you have a powerful troubleshooting and design tool.

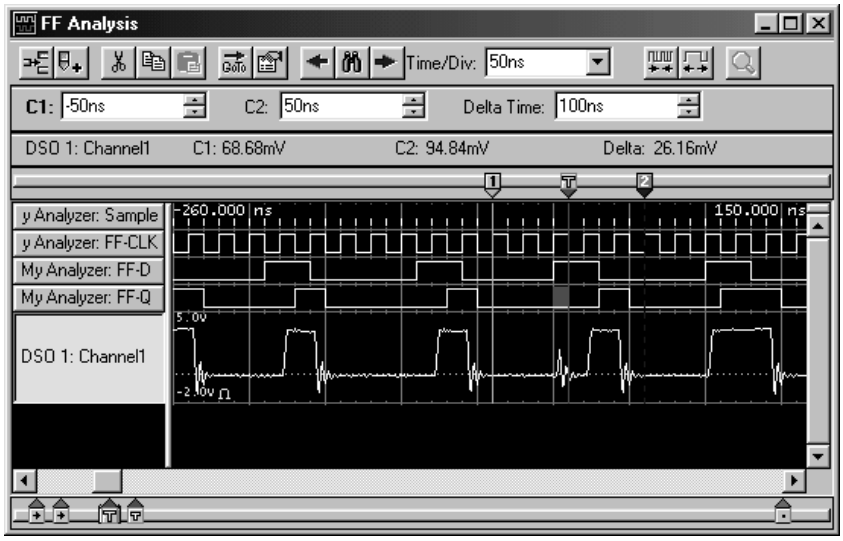


Figure 2–32: Resultant waveform data for Exercise 7

View the Setups

The logic analyzer Trigger window was set up to detect a glitch and then to trigger all modules. The DSO trigger was set up to wait for the system trigger from the logic analyzer module (see Figure 2–33). When the DSO received the system trigger, it acquired the data on the channel 1 probe.

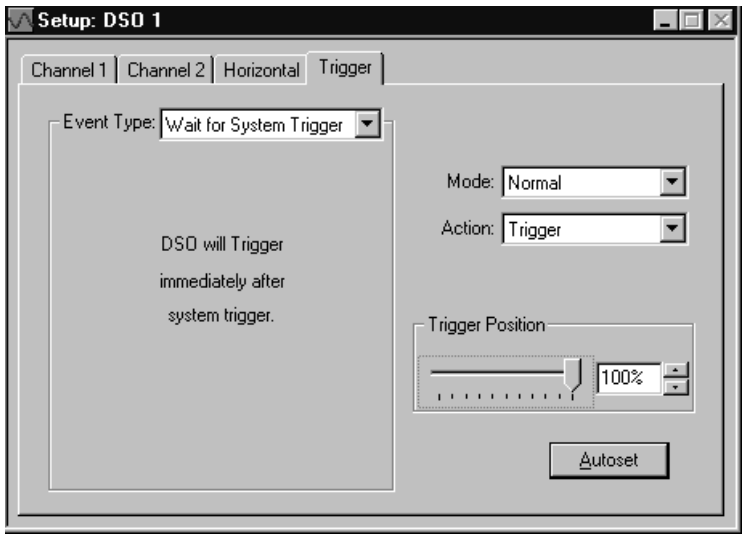


Figure 2–33: DSO channel trigger setups for Exercise 7

This completes the general purpose hardware exercises. The exercises in the next chapter show how to use a logic analyzer and an oscilloscope to identify and isolate problems in microprocessor-based systems.

Microprocessor Support Debug Exercises

Microprocessor Exercises Setup

The following exercises use the Microprocessor section of the training board to demonstrate the microprocessor hardware and software analysis features of the Tektronix Logic Analyzers.

Hardware Setups

To complete the exercises in this chapter, you will need a logic analyzer with 102 or more data channels (TLA 7N3, TLA 7N4, TLA 7P4 logic analyzer modules, or TLA 6x3 or TLA 6x4) with six P6418 or P6417 Probes or three P6434 Probes. A DSO module is optional, but not required for any of the exercises in this chapter.

Connect the probes to the appropriate A, D, and C connectors on the logic analyzer module. You will use the same probe connections through the remainder of this chapter.

Set Up the P6418 or P6417 Probes

You will need all P6418 or P6417 Probe podlets properly installed in the podlet holders. If you need to reinstall the probe podlets, refer to *Reinstalling Podlets* on page 2–2, and reinstall the podlets in the podlet holders before continuing with the exercises in this chapter.

***NOTE**. If you decide to connect the P6418 or P6417 Probe podlets individually, make sure that you connect each podlet in the correct order.*

Connect the Probes

Refer to Figure 3–1 through Figure 3–3 and connect the probes to the Microprocessor section of the TLA 7QS QuickStart training board as shown. Choose the figure according to the type of probe that you have.

If you have P6417 probes, connect the probes to the training board as shown in Figure 3–1. Connect the address, data, and control groups to the appropriate connectors on the training board. Connect the clock/qualifier leads to appropriate pins on the training board. Make sure that you connect the signal side of the probes to signal side of the square pins (ground connections are toward the rear of the training board as you face the LCD display; refer to the markings on the training board, if necessary).

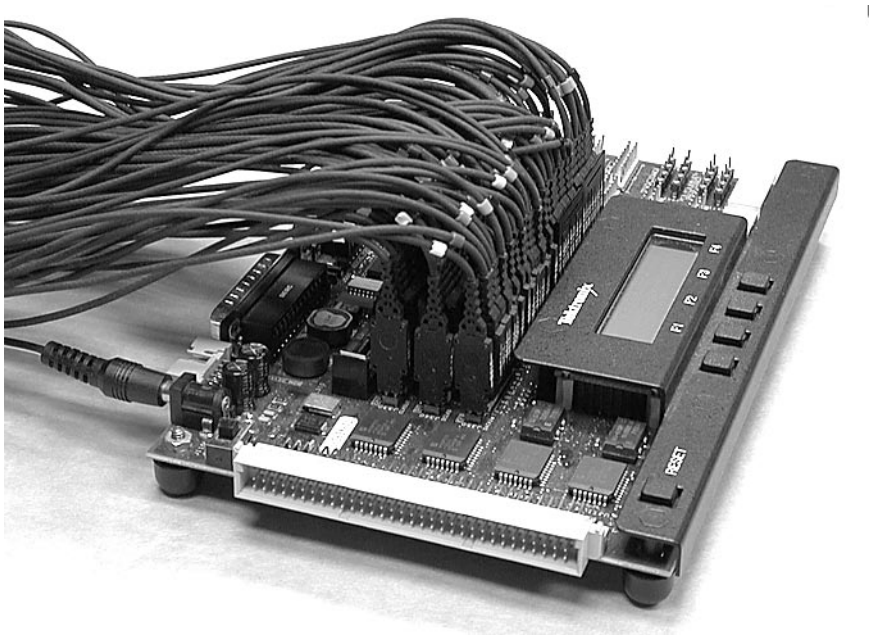


Figure 3–1: Microprocessor exercise P6417 probe connections

If you have P6418 probes, connect the probes to the training board as shown in Figure 3–2. Connect the address, data, and control groups to the appropriate connectors on the training board. Connect the clock/qualifier leads to appropriate pins on the training board. Make sure that you connect the signal side of the probes to signal side of the square pins (ground connections are toward the rear of the training board as you face the LCD display; refer to the markings on the training board, if necessary).

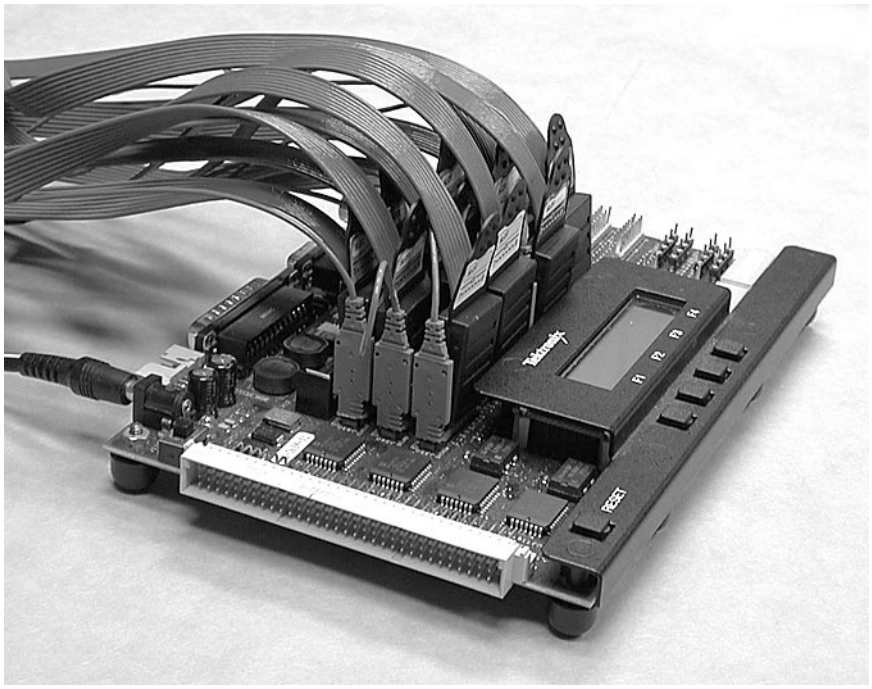


Figure 3–2: Microprocessor exercise P6418 probe connections

If you have P6434 probes, connect the probes to the training board as shown in Figure 3–3.

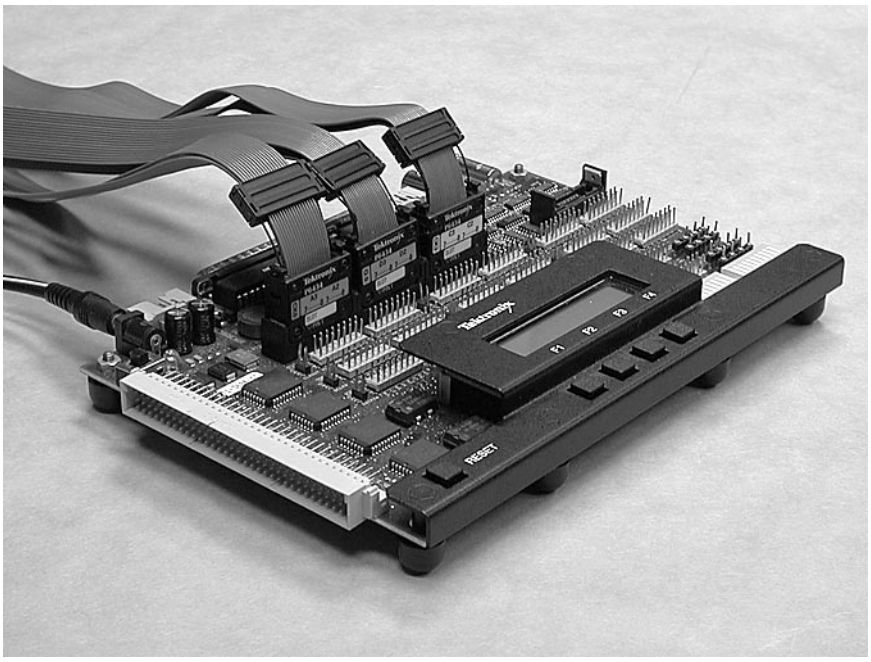


Figure 3–3: Microprocessor exercise P6434 probe connections

Start the Tektronix Logic Analyzer Family application and continue with the following steps. You will use the same hardware setup through the remainder of this chapter.

Load the Setups

Perform the following steps to load the microprocessor exercises. All microprocessor exercises are contained in the same folder (C:\Program Files\TLA 700\Quick Start\Microprocessor Analysis).

NOTE. *Each exercise contains two saved setup files. One file contains only setup information and requires you to capture live acquisition data to complete the exercises. The other file contains setup information and saved data. Use the saved data files to complete nearly every exercise without the need for acquiring live data.*

You can also use TLAVu to complete the exercises off-line without needing any acquisition hardware.

1. Select Load System from the File menu.
2. Open the Microprocessor Analysis folder and select the file as defined in the individual exercises (for example, load the 1-Capture Power Up Code.tla file for Exercise 1).

   The logic analyzer display should look similar to Figure 3–4.

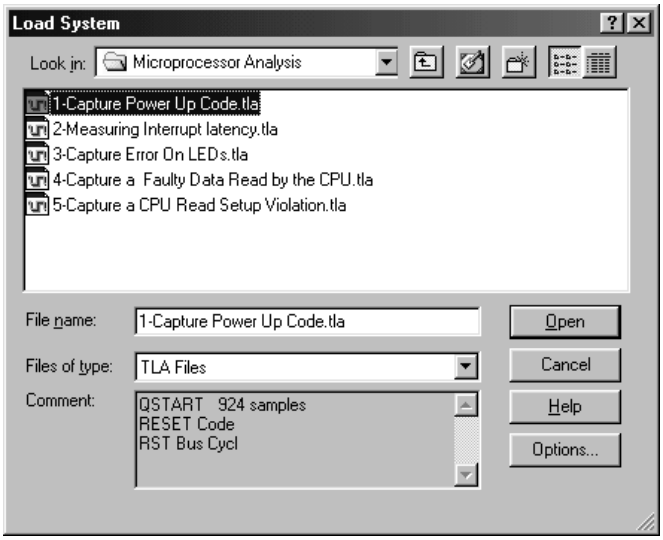


Figure 3–4: Load System dialog box

3. When prompted to load without saving the current system, click on Yes.
4. If you get a message telling you that the configuration in the file does not match your current hardware, click on OK.

5. A Load System Options dialog box appears, similar to Figure 3–5.

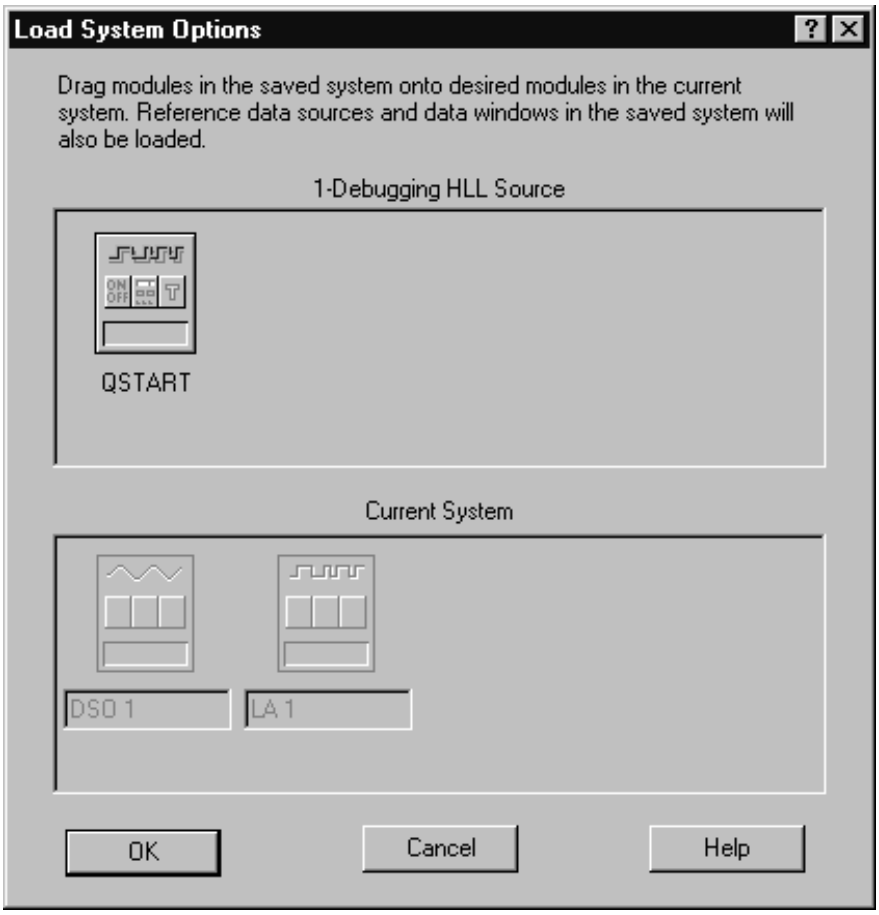


Figure 3–5: Load System Options dialog box

6. Drag the QSTART icon (and the My DSO icon, if needed) from the top of the dialog box to the shaded LA1 icon in the Current System area of the dialog box.

7. Click on OK to continue with the exercise.

***NOTE**. If the Load System Options dialog box appears and you want to view saved data, you may need to complete steps 8 through 11 to load a data window with saved acquisition data.*

To determine if you need to perform the following steps, click on one of the data window icons. If data appears in the data window, you can continue with the exercise. If no data appears, complete the following steps.

8. Select Load Data Window from the Window menu; the Load Data Window dialog box displays.

9. Click the Browse button, navigate to the Microprocessor Analysis setup folder, and then double-click the setup for the current exercise.

10. Select the data window from the list in the Load Data Window dialog and then click OK; the Name Data Window dialog box displays.

11. Change the name of the data window (it is recommended that you only change one or two characters so the name is similar to the original data window).

Now you have a data window that displays saved acquisition data for the current exercise. If you want to continue with the exercise and capture live data, refer back to the original data window that was loaded with the setup.

Trigger on a Power-on Reset and Capture the Controller Startup Code (Exercise 1)

Verifying the startup code of a microprocessor-based system can be easy using a logic analyzer. If you know the reset vectors, you can set up the logic analyzer to look for the reset address and then display the disassembled data in a Listing window.

In this exercise you will capture the startup code of the training board.

Load the Setup

1. Power off the training board.
2. Load the 1-Capture Power Up Code.tla system setup from the C:\Program Files\TLA 700\Quick Start\Microprocessor Analysis folder.
3. Click on the Run button to begin acquiring data.
4. When the Run button changes to Stop, the logic analyzer status changes to Running, and the animated TEK icon in the upper right hand corner is moving, power on the training board.

   The logic analyzer should trigger and capture the startup code of the training board.

View the Resultant Data

1. Open the Listing window labeled RESET Code. The window should contain the disassembled startup code and should look similar to Figure 3–6.

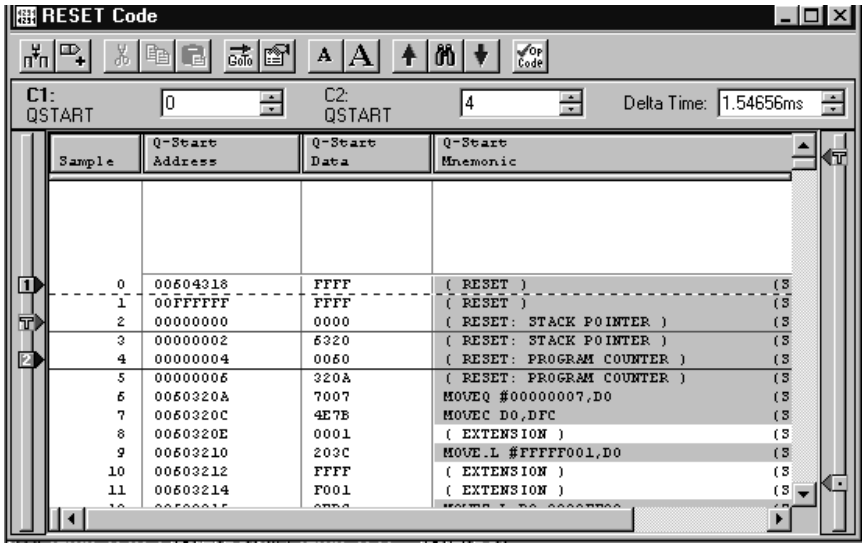


Figure 3–6: Listing window for Exercise 1

2. Notice how the RESET sequence and all software instructions are highlighted.
3. Examine the code to see how the microprocessor reads the stack pointer followed by the startup address.
4. Scroll through the data until you find a flush in the mnemonics column.

   The flush should be approximately 70 samples after RESET. A flush indicates that an instruction was fetched into the prefetch queue, but was then flushed from the queue because a branch was taken before the instruction was executed.
5. Right-click once on any column label to open a menu.
6. Select Properties from the menu; then select the Disassembly tab in the Properties dialog box.
7. Change the Highlight and Show fields in the Properties dialog box.
8. Apply the changes and see how they affect the display.

   Notice how you can focus on hardware, software, control flow, or subroutines.

9. Right-click on any column label to open a menu.

10. Select Properties from the menu; then select the Column tab.

11. Change the color of the selected column.

12. Apply the changes and see how you can use color to emphasize one or more columns in the display.

13. Minimize the RESET Code Listing window.

14. Open the RST Bus Cycl Waveform window to view the detailed timing of a bus cycle. Notice the high resolution provided by the MagniVu feature with 2 GHz sampling.

The RST Bus Cycl view should look similar to Figure 3–7.

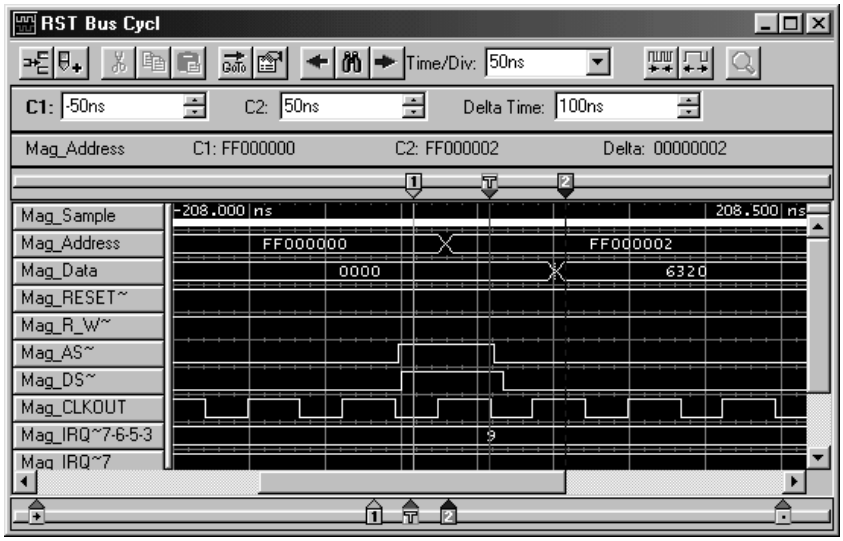


Figure 3–7: Timing waveform display for Exercise 1

View the Trigger and Channel Setups

1. Minimize the Waveform window and click on the Trig button in the logic analyzer icon.
2. The Trigger window should be similar to Figure 3–8.
3. Click on the If-Then button to open the Clause Definition dialog box.

   Notice that the Trigger setups use symbol files rather than hexadecimal or binary code. Symbol files make it easier to define and display the setups.

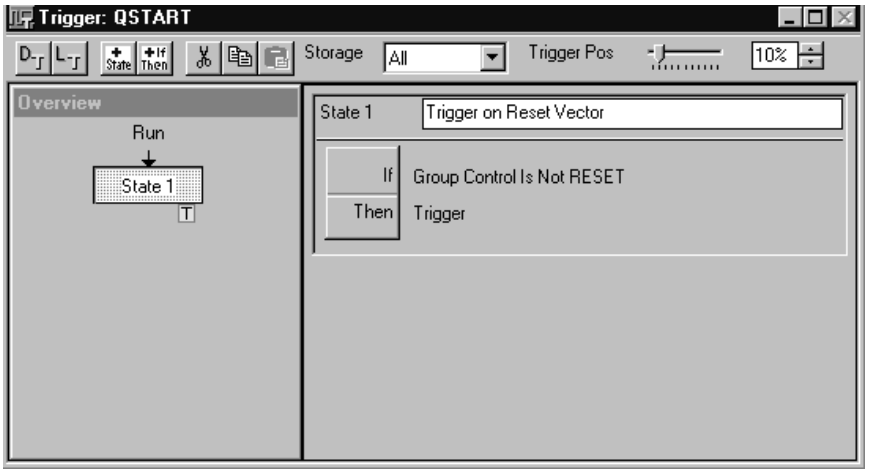


Figure 3–8: Trigger window for Exercise 1

4. Close the Trigger window and click on the Setup button in the logic analyzer icon.

   The channel setups are dependent on the microprocessor support package. Note the use of Custom clocking and the individual channel assignments.

5. When you are done looking at the setups, minimize any open windows (except the System window), and continue with the next exercise.

Use Trigger Timers to Measure Interrupt Latency (Exercise 2)

The training board is a microprocessor-based system controlled by buttons. The buttons send interrupts to the microprocessor. The microprocessor interprets the interrupt and determines the proper action to respond to the interrupt.

This exercise focuses on using multiple trigger states and timers to measure interrupt latency. To do this, the logic analyzer starts a timer on the assertion of an interrupt (when a button is pushed) and then triggers when the interrupt service routine ends.

Load the Setup

1. Select Load System from the File menu.
2. Load the system setup 2-Measuring Interrupt Latency.tla.
3. Click on the Run button to begin acquiring data.

Measure Timers

1. Click on the Status button to the left of the Run button in the System window to open the Status Monitor.
2. When the Run button changes to Stop, push the F1 button on the training board.
3. Note the timer values after the logic analyzer triggers.

NOTE. *You may not have a DSO section in the Status Monitior.*

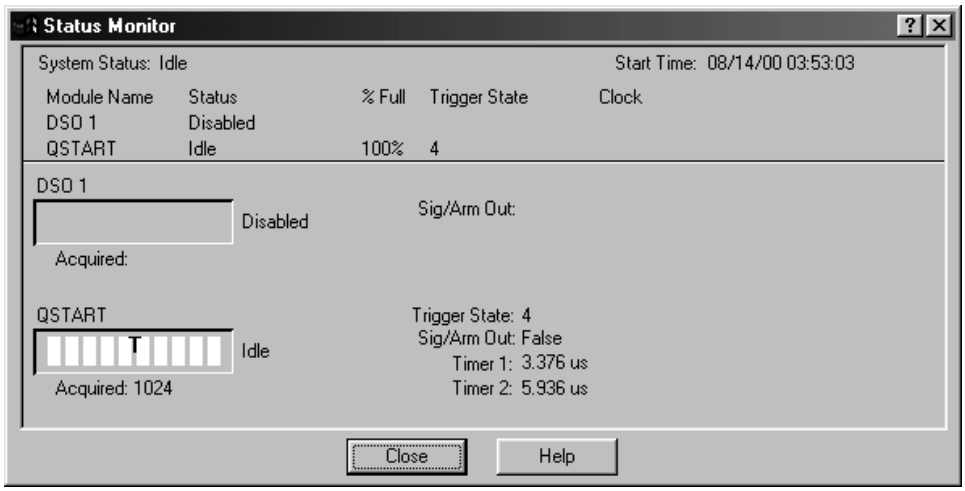


Figure 3–9: Status Monitor

Timer 1 measured the time from the assertion of the interrupt (when you pushed the F1 button) to the time that the microprocessor read the resultant interrupt vectors. Timer 2 measured the time to finish the interrupt service routine.

View the Resultant Data

1. Close the Status Monitor and open the Int Svc Code Listing window.
2. Notice the highlighted instructions in the window.

   Highlighting helps you identify the Int Ack Cycle, interrupt auto vectors, the start of the interrupt service routine, and the RTE instruction that ends the interrupt service routine (you may need to scroll the display to view all of the instructions).

The Int Svc Code Listing window should look similar to Figure 3–10.

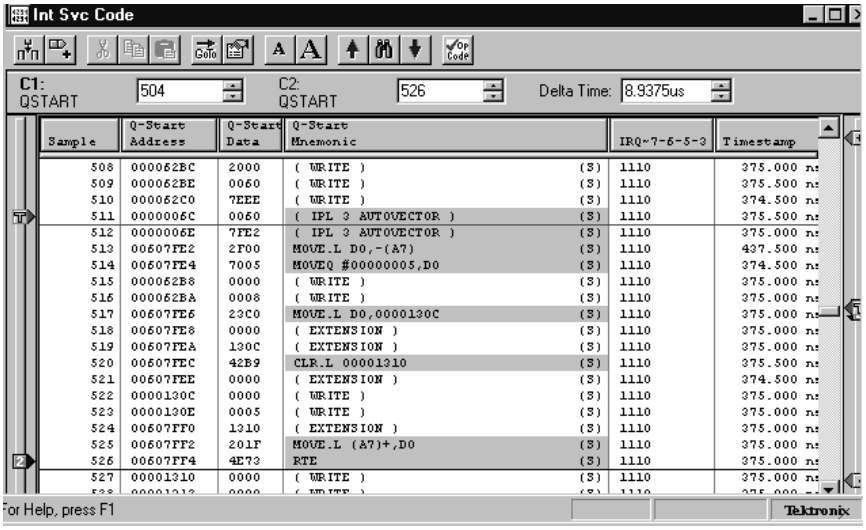


Figure 3–10: Listing window for Exercise 2

View the Setups

1. To understand how the logic analyzer uses timers to measure interrupt latency, view the setups in the Trigger window (see Figure 3–11).

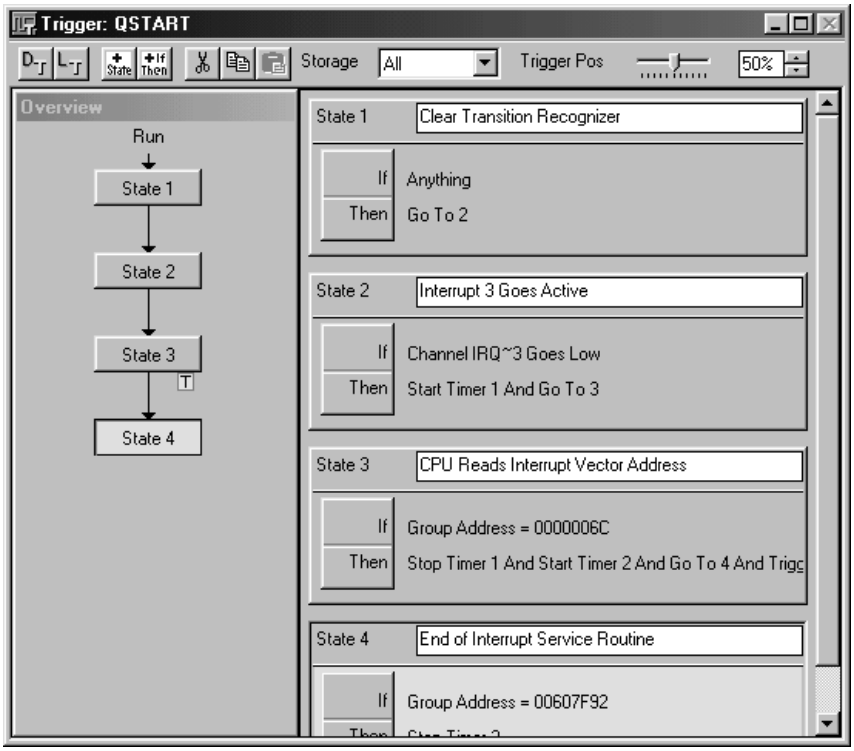


Figure 3–11: Trigger window for Exercise 2

2. State 1 clears the transition recognizer to a known state before looking for a transition on channel IRQ~3 in State 2. You must never use a transition recognizer in State 1.
3. Notice the multiple trigger states that start a timer on the insertion of an interrupt and then trigger the logic analyzer when the Interrupt Service Routine ends.
4. After studying the trigger setups, minimize any open windows (except the System window), and continue with the next exercise.

Trigger on Faulty Data Written to the LED Display (Exercise 3)

The remaining exercises in this chapter focus on using a logic analyzer to troubleshoot the system so that you can detect software or hardware problems. The exercises use one of the programs on the training board, the AUTO DELAY program. The program simulates a hardware problem in the form of a data readback error that causes an improper LED display on the training board.

A data read error causes the microprocessor to write the wrong data value to the LEDs. The LED register must contain a logic low to light the LED. For example, if the value AAAA is written to the LED port on the QuickStart training board, the LEDs display 5555.

The following exercises are set up to simulate a typical troubleshooting sequence. In exercise 3, you will use disassembly to determine where the data is being written. In Exercise 4, you will verify that the written data is actually wrong and use the high resolution MagniVu feature to determine from where the microprocessor gets the incorrect data. Exercise 5 shows how to use setup and hold triggering to verify that a timing error is the cause of the problem.

The microprocessor executes code in the following sequence:

1. The CPU reads data from address 0054 0000.

2. The CPU complements the data.

3. The CPU then writes data to address 0044 0000 (the LED port).

Load the Setup

1. Select Load System from the File menu.
2. Load the system setup 3-Capture Error on LEDs.tla.
3. Click on the Run button to begin acquiring data.
4. Find the AUTO DELAY program on the training board by using the UP and DN buttons to scroll through the program list.
5. Press the RUN button on the training board to start the AUTO DELAY program.

   The LEDs should display the data as directed by the program. At one point the LEDs will display incorrect data that will trigger the logic analyzer.

View the Resultant Data

1. After the logic analyzer triggers and captures the data, open the LED Error Listing window. This window contains the disassembled data and should look similar to Figure 3–12.

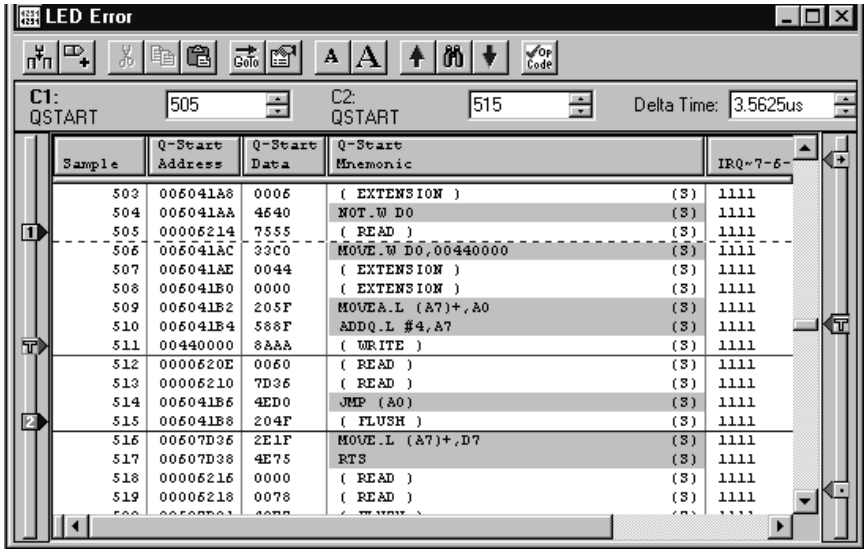


Figure 3–12: Listing window for Exercise 3

2. Locate the trigger mark in the LED Error Listing window and find the data value that caused the logic analyzer to trigger.

   The data written by the microprocessor to address 0044 0000 was not AAAA as expected.

View the Setups

1. To understand how the logic analyzer triggered on the faulty data, view the setups in the Trigger window (see Figure 3–13).
2. Notice that the word recognizer is set up to trigger when the microprocessor writes data other than AAAA to the LED.

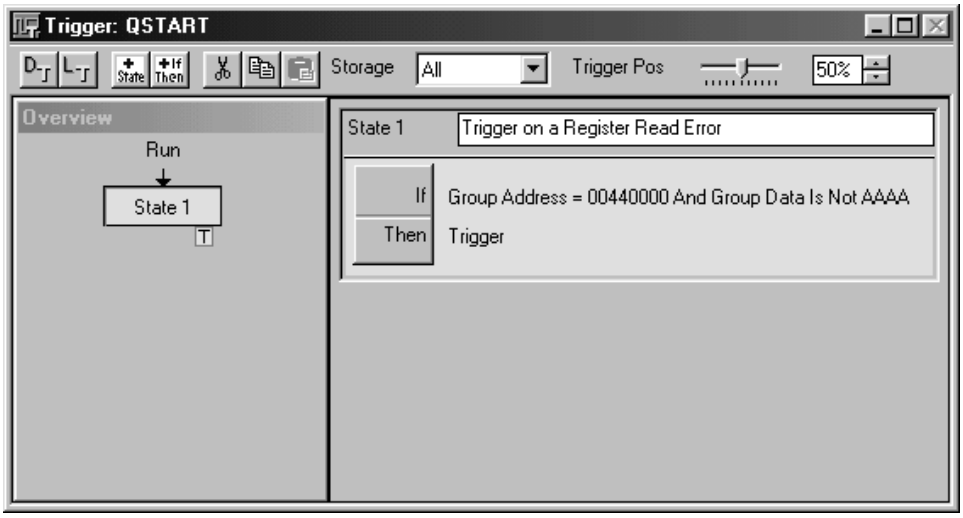


Figure 3–13: Trigger window for Exercise 3

3. Click on the If/Then button to see other trigger definitions and windows that are used in this exercise.
4. Before proceeding to the next exercise, press the STOP button on the training board, and minimize all open windows (except the System window).

Trigger on Faulty Data Read by the CPU (Exercise 4)

In the previous exercise, you used the logic analyzer to verify that the data written to the LEDs was incorrect. You already know that you do not have an LED hardware problem because the LEDs occasionally display the correct data. The next step in isolating the problem is to determine if the microprocessor reads the correct data.

In this exercise you will use the logic analyzer to trigger on faulty data being read by the microprocessor to capture a register read error. You will use the high resolution timing MagniVu feature to check the circuit timing.

***NOTE**. Make sure that the AUTO DELAY program is not running on the training board before you start this exercise.*

Load the Setup

1. Select Load System from the File menu.
2. Load the saved system 4-Capture a Faulty Data Read by the CPU.tla.
3. Click on the Run button to begin acquiring data.
4. Press the RUN button on the training board to start the AUTO DELAY program.

View the Resultant Data

1. Open the Read Error Listing window after the logic analyzer triggers.
2. Locate the trigger mark and see what data value the microprocessor read.

   The Read Error Listing window contains the disassembled data and should look similar to Figure 3–14. The data read from address 0054 0000 was not 5555 as expected. This explains why the data written to the LEDs is incorrect. The next step is to find out why the data is incorrect.

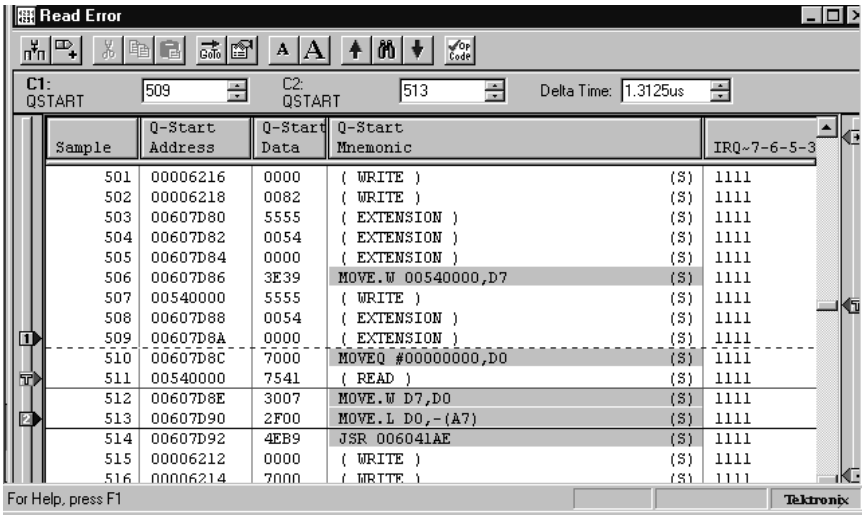


Figure 3–14: Listing window for Exercise 4

3. Open the Mag Read Err Waveform window and check the circuit timing.

   The window uses the MagniVu feature of the logic analyzer to help you analyze the timing of the failing bus cycle. According to the microprocessor specifications, the data must be stable for 5 ns before the falling edge of the clock before the data strobe (DS) signal goes high. The Mag Read Err Waveform window should look similar to Figure 3–15.

Notice that the data was not stable 5 ns before the falling edge of the clock.

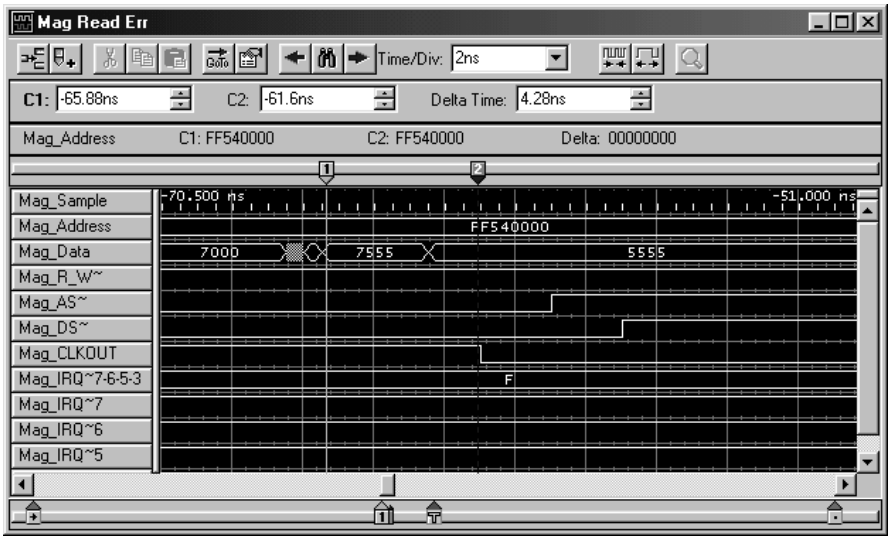


Figure 3–15: Waveform window for Exercise 4

4. Zoom in on the faulty data, and you can see the actual data read by the logic analyzer in the Listing window (Figure 3–14).

5. To zoom in on the data, move the mouse pointer to just before the data changes on the Mag_Data bus. Press and hold the left mouse button and drag the mouse to create a dashed box. Your display should look like Figure 3–16. Right-click the mouse and select Zoom from the menu.

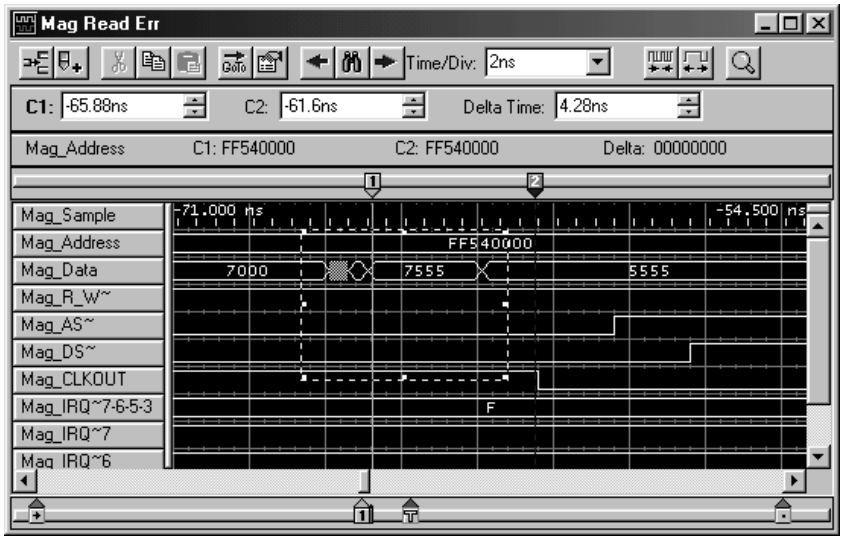


Figure 3–16: Waveform window with dashed zoom box

6. To alternately zoom on the data, click on the Zoom In button on the top right of the display. If you have a portable mainframe, you can use the front panel Horizontal SCALE and POSITION controls to zoom in on the data.

View the Setups

1. To understand how the logic analyzer triggered on the faulty data, view the setup in the Trigger window (see Figure 3–17).

   Notice that the word recognizer is set up to trigger when the CPU reads data other than 5555.

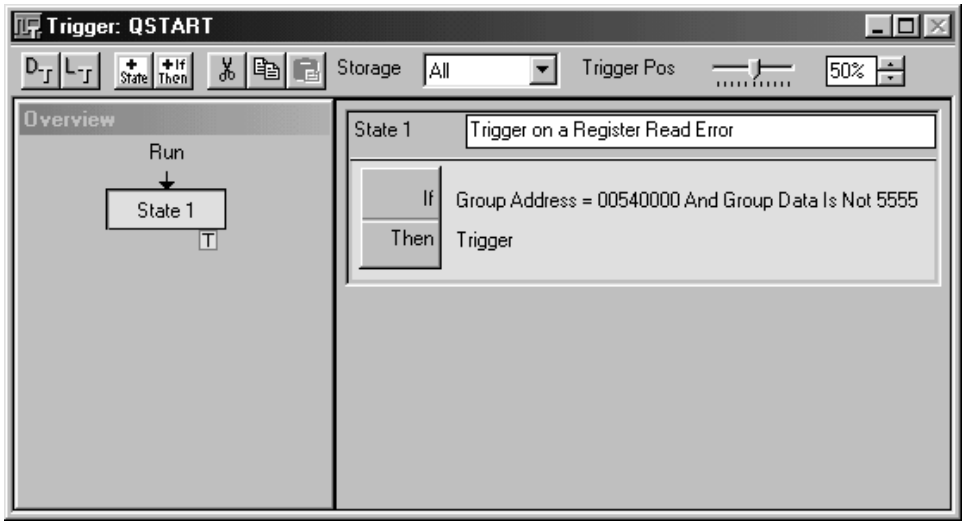


Figure 3–17: Trigger window for Exercise 4

2. Press the STOP button on the training board.

3. Before proceeding to the next exercise, minimize all open windows (except the System window).

Trigger on a Setup Violation of the CPU Read Cycle (Exercise 5)

In the previous two exercises, you discovered that a timing problem is causing the LEDs to display the wrong data. In this exercise, you will use the logic analyzer to trigger on a setup violation on the data signals to verify the problem.

***NOTE**. Make sure that the AUTO DELAY program is not running on the training board before you start this exercise.*

Load the Setup

1. Select Load System from the File menu.
2. Load the saved system 5-Capture a CPU Read Setup Violation.tla.
3. Click on the Run button to begin acquiring data.
4. Press the RUN key on the training board to start the AUTO DELAY program.

View the Resultant Data

1. Open the Mag Read Err waveform data window after the logic analyzer triggers. The view should look similar to Figure 3–18.
2. Notice how the logic analyzer displays the data and address lines in busforms rather than displaying each address or data line individually.

   The busforms help you focus on the area of interest. In this case you are interested at the point where the data transition takes place. Using the MagniVu timing mode to analyze the timing of the failed bus cycle, you can see that the data was not stable at the time the microprocessor executed the read cycle. The microprocessor specifications require that the data must be stable for 5 ns before the falling edge of the clock before the data strobe (DS) signal goes high.

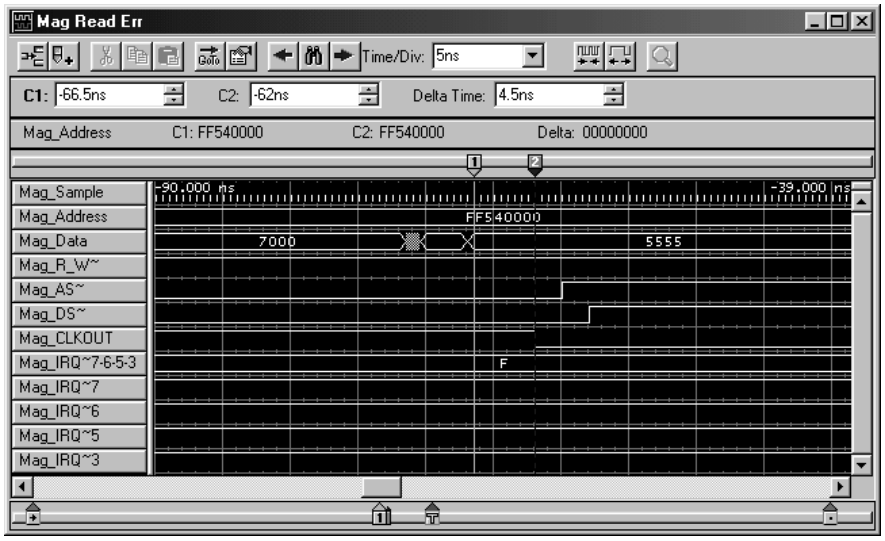


Figure 3–18: Waveform data window for Exercise 5

3. To see which data channel caused the setup failure, you can expand the data bus. Right-click on the Mag_Data signal label on the left side of the waveform window. Then select Expand Channels. Notice that the 16 data channels now show in the waveform window.

4. Scroll up until you see Mag_Data_13. The waveform data should look similar to Figure 3–19. This data channel is transitioning too late and causes the setup violation.

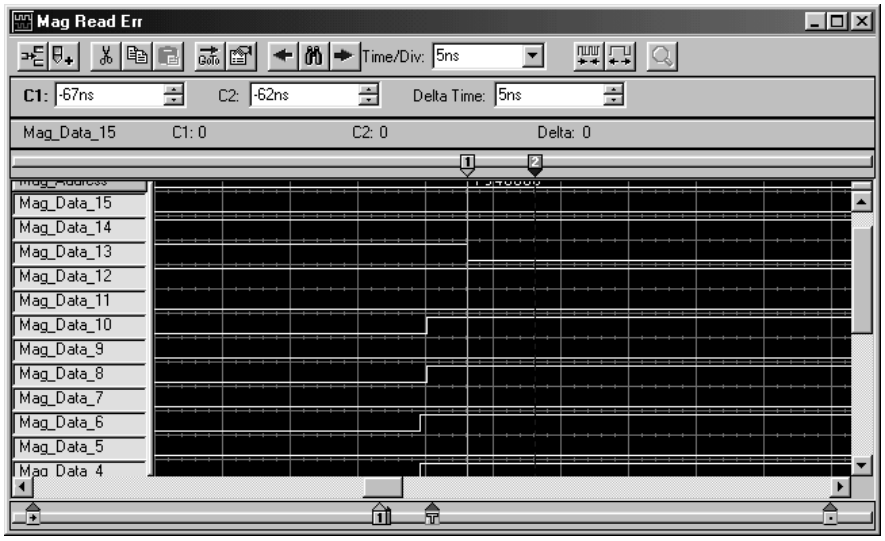


Figure 3–19: Expanded data channels showing a setup violation for Exercise 5

View the Trigger Setups

1. To understand how the logic analyzer triggered on the setup and hold fault, view the setups in the Trigger window.
2. Notice that the event recognizer is set to trigger on a setup and hold fault (see Figure 3–20).

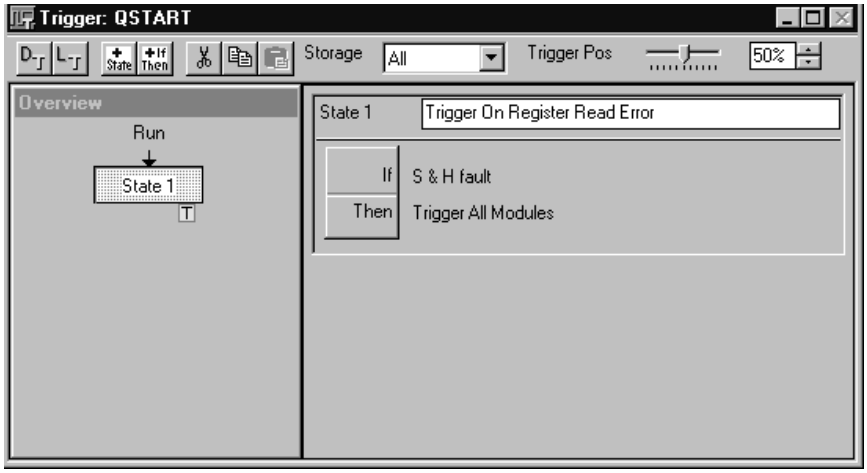


Figure 3–20: Trigger window for Exercise 5

3. Click on the If then clause in State 1.
4. Open the Define Violation dialog box to see how the setup and hold violation is specified. (See Figure 3–21 on page 3–28.)
5. Notice that the logic analyzer looked for the setup and hold violation on the data group.

   You can specify which group you are interested in by checking the box adjacent to the setup and hold values. You can also modify the individual setup and hold values.
6. Try changing some of the setup and hold values and then see how the logic analyzer captures the setup and hold violations.

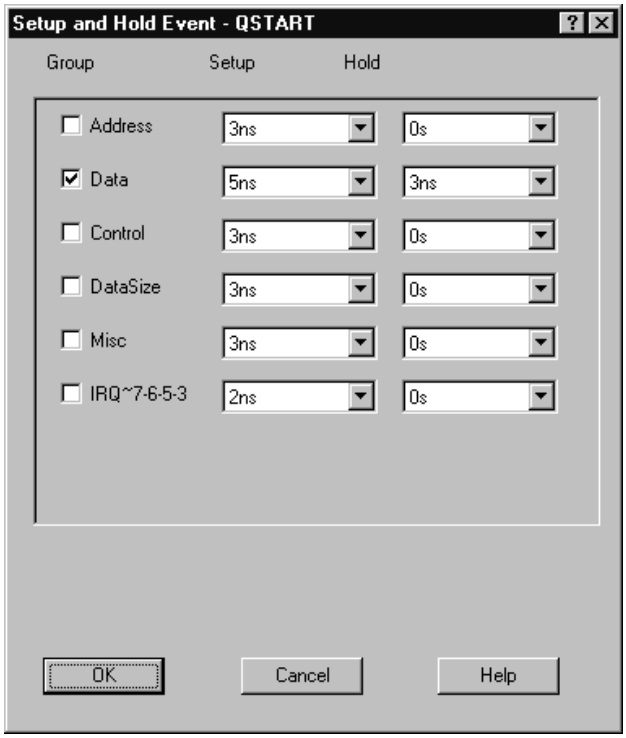


Figure 3–21: Define Violation dialog box

Embedded Software Debug Exercises

Embedded Software Debug Exercises Setup

The following series of exercises demonstrate the analysis capabilities of the Tektronix Logic Analyzer to debug real-time embedded software. These exercises have the same setup requirements as those used in the Microprocessor section. Therefore, it is recommended that you read *Microprocessor Exercise Setup*, beginning on page 3–1, before starting the exercises in this section.

***NOTE**. Each exercise contains two saved setup files. One file contains only setup information and requires you to capture live acquisition data to complete the exercises. The other file contains setup information and saved data. Use the saved data files to complete nearly every exercise without the need for acquiring live data.*

You can also use TLAVu to complete the exercises off-line without needing any acquisition hardware.

Source Code Window Background (Exercise 1)

The Source window displays the contents of source files that correspond to the acquisition data in a Listing window. The Listing window displays the assembly execution trace of the high level source. The relationship between the acquisition data and the specific source is determined by examining the debug information stored in the executable file that was created by code-creation tools such as compilers, linkers, and loaders. The specific file being displayed, the cursor positions, and user-defined marks in the file are automatically updated as you use the controls in the Source window and the associated Listing window.

The source statements in each of the source files displayed in the Source window have associated address ranges that define the memory locations occupied by the microprocessor instructions that implement the statement. The address ranges determine the mapping from samples in the Listing window to statements in the Source window.

Because Source windows interact with associated Listing windows, you will normally want to display both windows at the same time. Moving the cursors in the Source window moves the cursors in the associated Listing window, and moving the cursors in the Listing window moves the cursors in the Source window.

Source Window Structure

The Source window consists of three main components, the Source toolbar, the Source control bar, and the Source data area. See Figure 4–13 on page 4–20.

- The Source toolbar provides tools to change the appearance of the data area and to search for data. A Scan Listing field specifies the search direction in the Listing window when you move the cursors in the Source window.
- The Source control bar provides means to step through data, step between marks, and scroll through data. These controls also indirectly affect the cursors in the associated Listing window.
- The Source data area displays the contents of the source file. This area also displays diagnostic messages if the Source window is unable to display a source file. When the window displays a source file, the header displays the path of the file.

Please refer to the online help for more information.

Debugging Real-Time Execution of High Level Source (Exercise 1)

In the following exercise you will correlate a real-time execution trace with high level source code using the Tektronix Logic Analyzer integrated source code data window. Specifically, you will capture real-time execution of the STOP LITES program on your embedded system (QuickStart training board) and correlate it with the actual “C” source code.

Load the Saved System

1. Select Load System from the File menu.
2. Load the following saved system:
   C:\Program Files\TLA 700\Quick Start\Embedded Software Debug\ Source\1-Debugging HLL Source.tla.

The restored system should look similar to Figure 4–1. You will not have a DSO icon if your logic analyzer does not contain a DSO module.

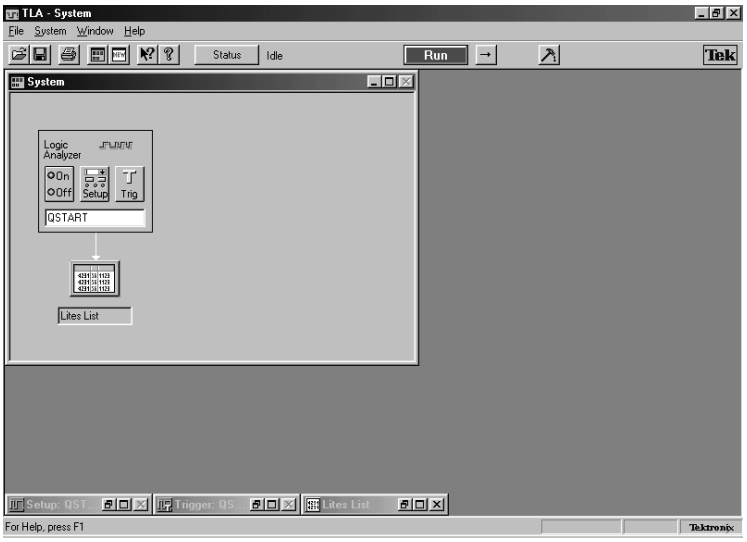


Figure 4–1: Restored system for Exercise 1

3. Open the Lites List data window.
4. Click the Run button and wait for it to change to Stop.

5. Push the DN (F2) button on the training board to scroll down the program list until you reach STOP LITES.
6. Push the RUN (F3) button to run the STOP LITES program. When the logic analyzer triggers, the Listing window contains data.
7. Stop the STOP LITES program by pushing the F4 button on the training board.

The Lites List window now contains data from the real-time trace of the STOP LITES program. Note that at sample 101, the logic analyzer triggered on the execution of the StopLite subroutine. See Figure 4–2.

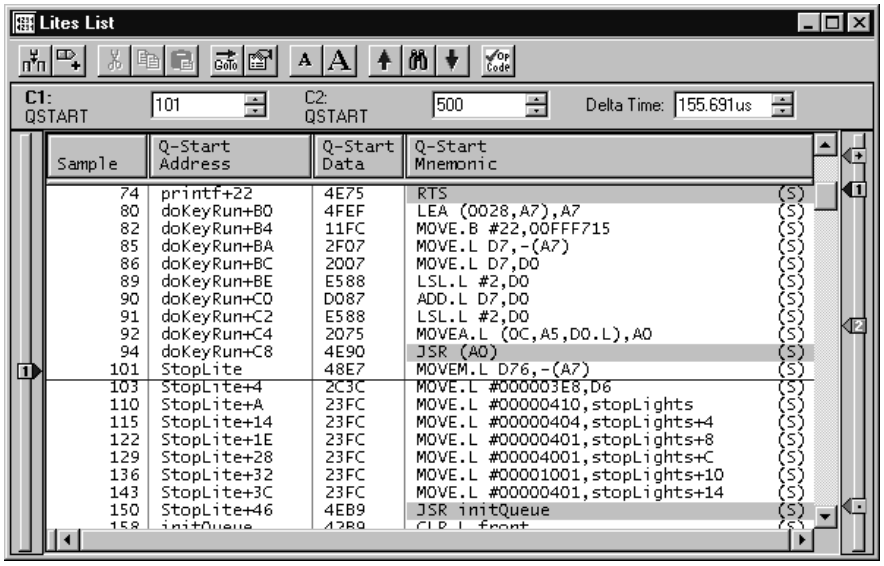


Figure 4–2: Real-time trace of the STOP LITES program

Correlate the assembly trace shown in the LitesList window with the high-level source code and you will see the StopLite subroutine displayed in its native “C” high-level language.

NOTE*. Remember to load your microprocessor support package and set the Listing window disassembly display mode to Software before you create the Source window.*

This exercise used a preconfigured setup to display the Listing window in Software mode.

---

8. Click on the System window, go to the System menu and select Options.
9. Click on the Source Files tab and make sure that the Search Path List option is selected. Click the Add button.
10. Click the Browse button and navigate to the following location: C:\Program Files\TLA 700\Quick Start\files
11. Click OK three times to close the dialog boxes.
12. Select New Data Window from the Window menu or the NEW button on the system toolbar.
13. Select Source from the dialog box. Click Next.
14. Select "An existing Listing window." Click Next. The dialog box in Figure 4–3 appears.

***NOTE**. Before clicking on the Next button in this dialog box, take note of the required information to correlate a Source window with a Listing window. The labels in Figure 4–3 identify this information. These parameters were preset for this exercise; you will need to provide your own information in future instances.*

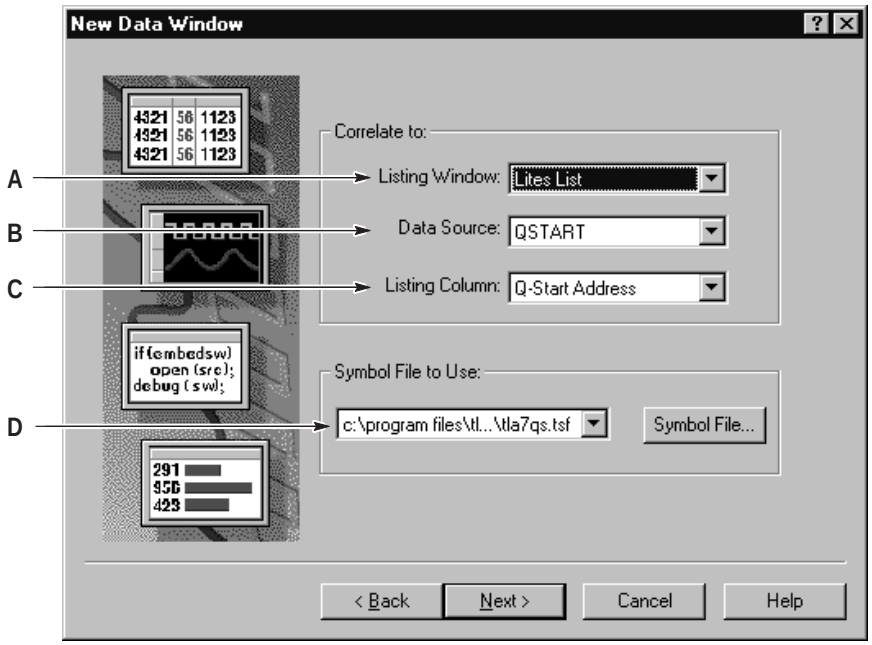


Figure 4–3: New Data Window wizard dialog box

A. Associate a Listing window from the list of all currently available Listing windows containing data.

B. Associate a data source from the selections based on the Listing window. The data source can be data from an installed module or a saved data file.

C. Associate a column (channel group) from the selected data source. In most cases, you will want to select a microprocessor address column. Time stamp columns, Sample columns, and channel groups without assigned data channels are not available as data groups for this purpose.

D. Associate a symbol file, containing the source code symbols, from a list of all loaded symbol files. Valid symbol files include those that correspond to the channel group column (when used with a symbolic radix). The symbol file can be the executable file that has been compiled and linked for debug, or it can be a TSF format file (see the online help for more information on TSF file format). The loaded symbol file must also contain source code symbols that correspond to the executed code captured by the logic analyzer.

Refer to *Understanding the Tektronix Logic Analyzer Symbol Support* on page 4–22 for additional information.

15. Click Next in the New Data Window wizard dialog box.

16. Enter a name to assign to the new Source window (for example, Lites Source). Click Finish.

Figure 4–4 shows the Source window after a successful connection to the Listing window. You can increase the size of the font by clicking on the larger of the two As on the Source window toolbar.

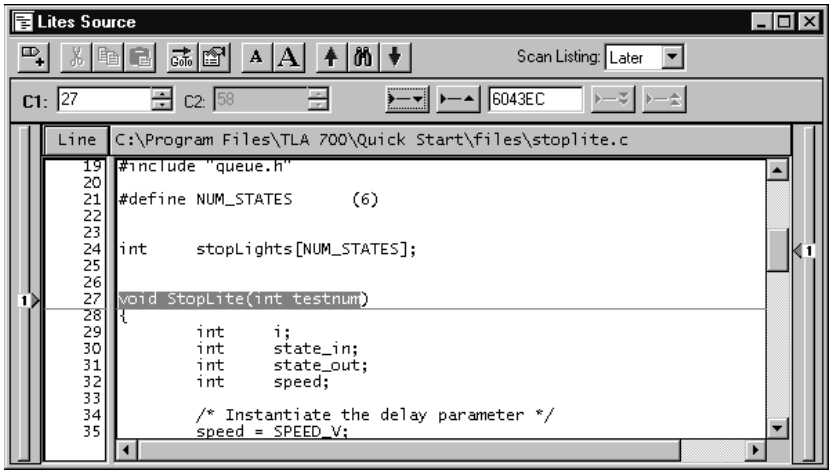


Figure 4–4: Lites Source window after connecting to the Lites List Listing window

Recommended Window Layout for a Portable Mainframe Display

Figure 4–5 shows the recommended window layout for the portable mainframe. This layout maximizes the amount of viewable data on a portable mainframe display. To reduce the font size, click the smaller of the two As on the window toolbar.

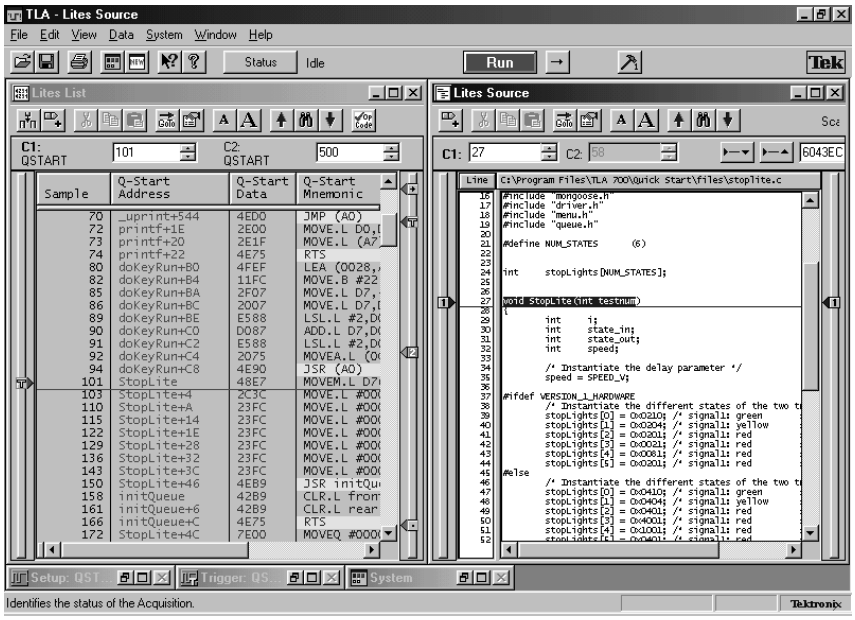


Figure 4–5: Recommended window layout for the portable mainframe

Using the Source Window

As noted previously in *Source Code Window Background (Exercise 1)* on page 4–3, cursor movements can be driven from either the Source window or Listing window. However, in this exercise, you will focus primarily on navigating through your source files from within the Source window. There are several different ways to move around in the Source window.

This exercise will walk you through four techniques. Two of the techniques use a structured approach, one uses an unstructured or random approach, and the last demonstrates a cursor movement technique in the Source window that is unique to the Tektronix Logic Analyzer.

Refer to Figure 4–13 on page 4–20 as a visual aid to the following procedures. More detailed information about the Source window is available in the online help.

Moving Through Source Files Using the "Step Forward" and "Step Backward" Buttons

The following section describes navigating through source files using the "Step Forward" and "Step Backward" buttons.

Use the Step Forward and Step Backward buttons (items B and C respectively, in Figure 4–13 on page 4–20) to trace the order of execution of source statements. The Step Forward button moves to the next executed source statement. The Step Backward button moves to the previous executed source statement. To demonstrate this concept, perform the following step:

1. Click on the Step Forward button eight times so that the active cursor (cursor 1) in the Source window is at line 56 in the file stoplite.c as shown in Figure 4–6.

After each click, note that the cursor in the Source window advances to the next executed source statement. Also note that the Listing window cursor is correlated with the Source window cursor and also moves with each click.

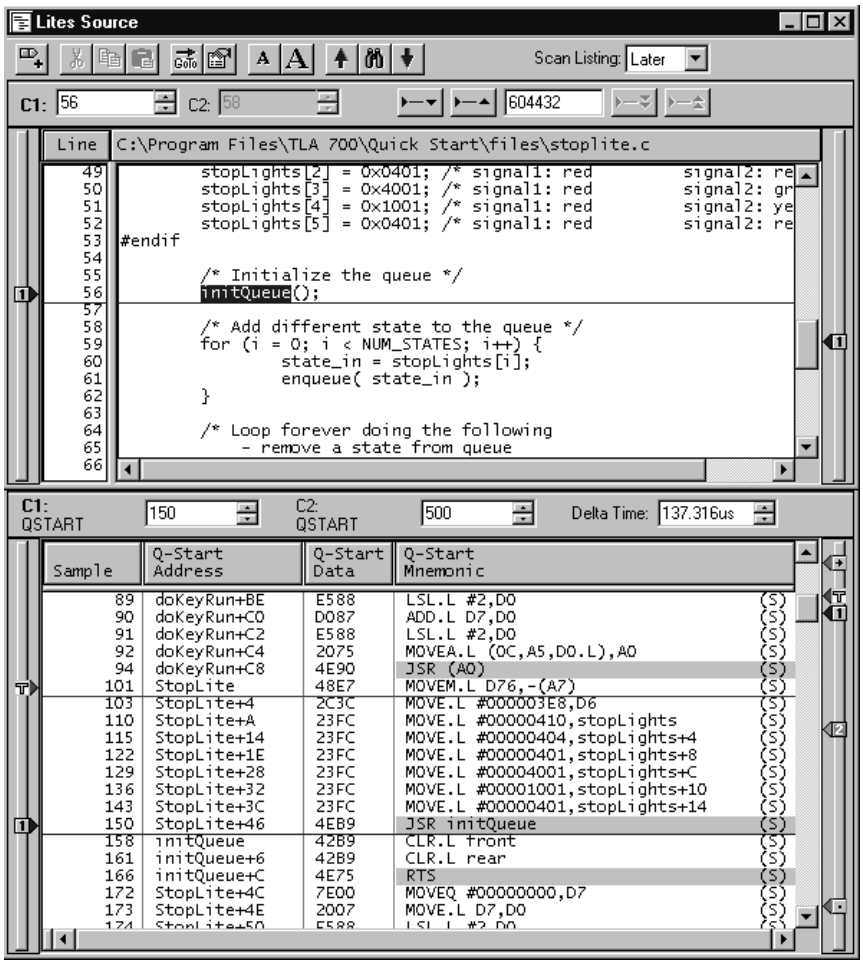


Figure 4–6: Source and Listing windows after stepping forward eight times

The data area in the Source window now displays the "C" source file (stoplite.c) that contains the main routine for the STOP LITES program. The active cursor in the Source window is positioned at the call to the initQueue() routine while the corresponding cursor (cursor 1) in the Listing window shows the assembly language equivalent of this statement.

2. Click once more on the Step Forward button and note that the Source window switches files to display the execution of the InitQueue() function.

3. Click the Step Backward button once to return to the point just before InitQueue() was called. The Source window cursor is again on line 56 of stoplite.c and the Listing window cursor is on sample 150.

Moving Though Source Files Using the "Move Cursor X Here" Mechanism (Random Movement)

The following section describes navigating through source files using the "Move Cursor X Here" mechanism.

What if you now want to skip over the initQueue() routine? Instead of single-stepping through this routine, you can move the cursor to the statement after the call to initQueue(). This feature is analogous to the "Step Over" function in a software debugger. To demonstrate this function, perform the following steps:

1. Go to line 59 in the current file, stoplite.c, and position the mouse pointer over the "for (I =0;" portion of the source statement.

NOTE. *A single line in the Source window can consist of more than one statement on a single line, for example: for (i=0; i < NUM_STATES; i++). The ability of the Source window to discriminate between multiple statements on the same line depends on the amount of information provided by the code-generation tools. If there is enough information to identify multiple statements per line, the Source window cursors will include character highlighting to identify individual statements on a line.*

---

2. Right-click the mouse and choose "Move Cursor 1 Here." Figure 4–7 shows the result after performing this step.

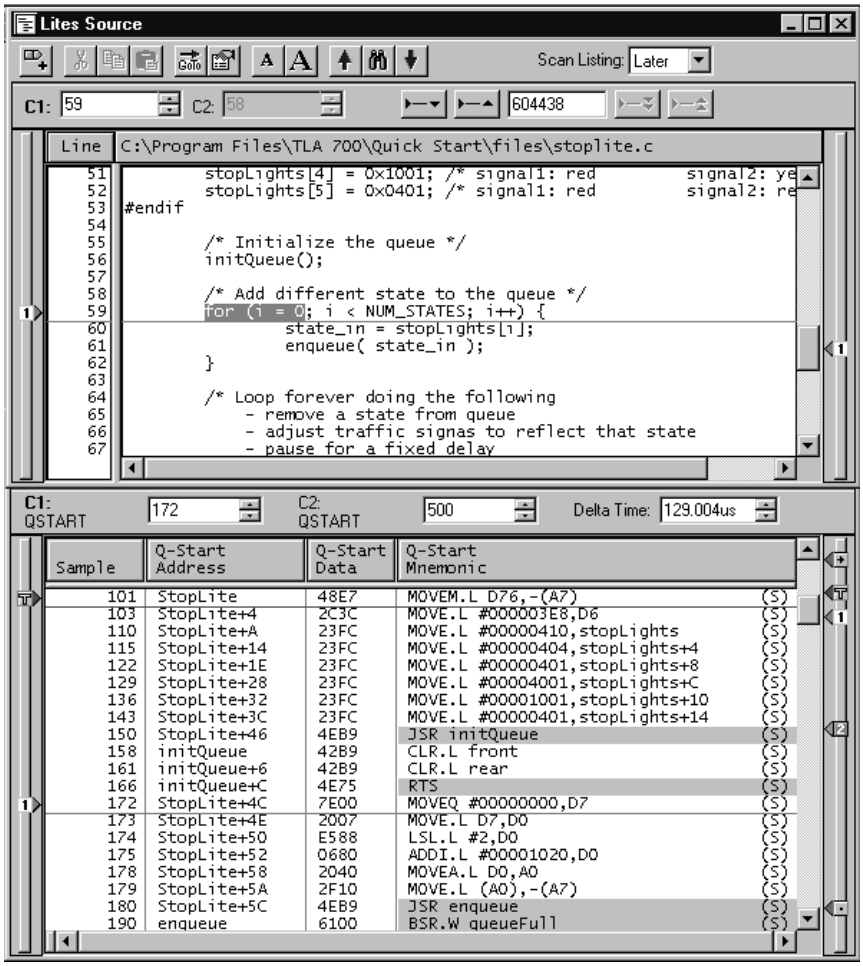


Figure 4–7: Source and Listing windows after using “Move Cursor 1 Here”

Note that the Listing window cursor advanced to sample 172, skipping over the samples associated with the call of the InitQueue() function.

This is an example of moving the cursor to what is essentially a random location, because you can move the cursor to any location in any source file. A random location in this context means a source code statement whose execution order relative to the previous cursor location cannot be determined with certainty. Therefore, the ambiguity is resolved through use of the Scan Listing control (item F in Figure 4–13 on page 4–20) in the Source window toolbar. Using this control, the user can force the Source window to proceed in a specific direction. For a more detailed description of the Scan Listing control, refer to the online help.

Moving Through Source Files Using Marks (Simulating Debugger Software Breakpoints)

Another alternative to single-stepping through each source statement is setting marks in the Source window (as breakpoints), and stepping between the marks in execution order to move through the source code. This is similar to setting a breakpoint in a software debugger. For example, you can use this feature to find the next execution of the same statement in a loop to verify that it executed the correct number of times. To demonstrate this concept, perform the following steps:

1. Position the mouse pointer anywhere over the source statement at line 61 in the current file (stoplite.c), right-click the mouse, and select "Add Mark Here." Figure 4–8 shows the results after this operation.

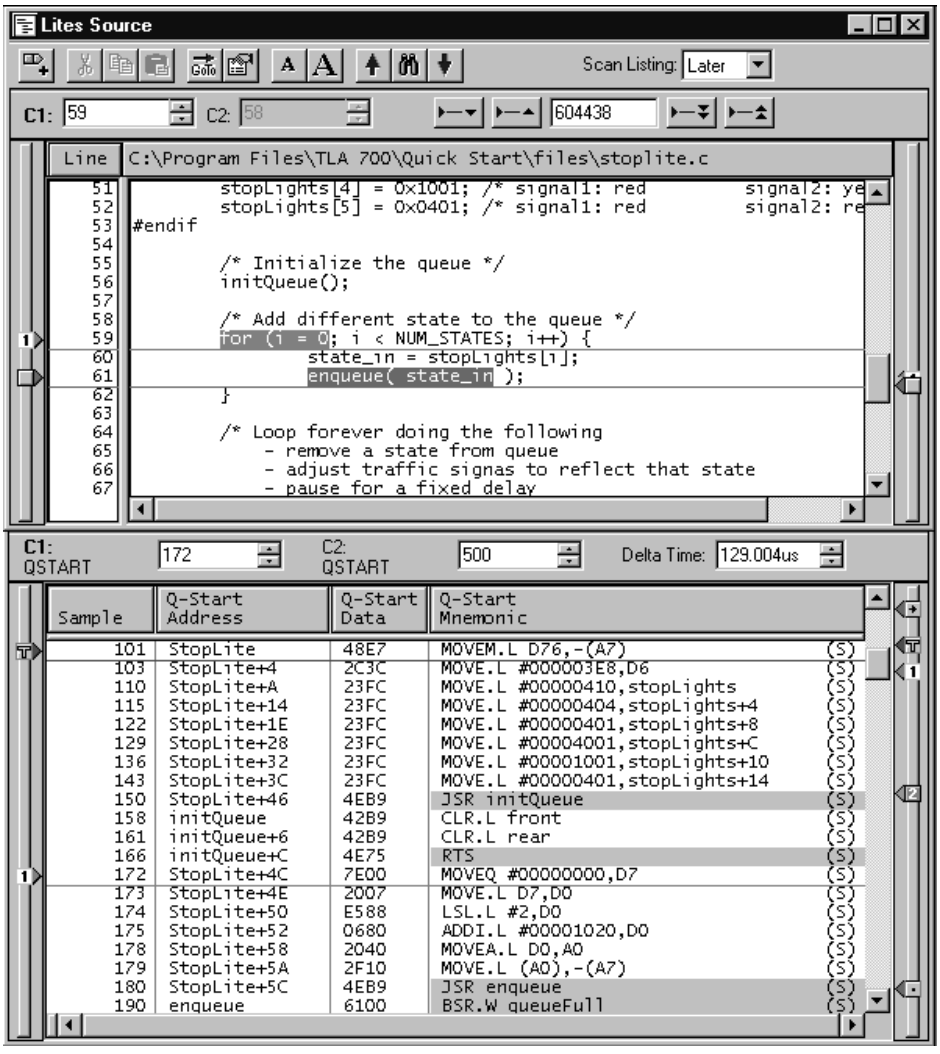


Figure 4–8: Source and Listing windows after placing a mark (step 1)

2. Click once on the “Next Mark” button (item E in Figure 4–13 on page 4–20) and the active cursor (cursor 1) moves to the marked location as shown in Figure 4–9.

   Note that the Listing window cursor is now on sample 179, which corresponds to the first execution of this source statement.

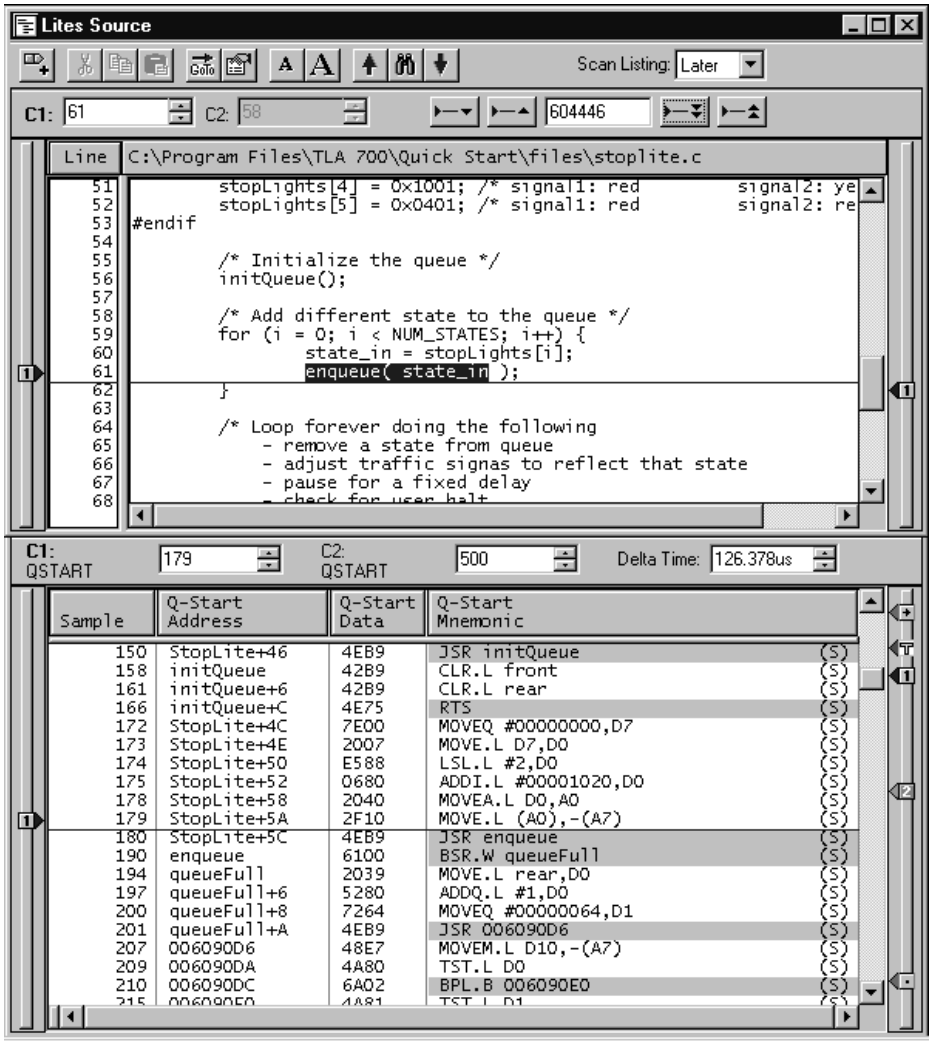


Figure 4–9: Source and Listing windows after step 2

3. Click on the Next Mark button again. The active cursor (cursor 1) in the Source window remains at line 61, while the corresponding cursor (cursor 1) in the Listing window moves to sample 346, which is the next execution of that statement in the for loop. Figure 4–10 shows the results after this operation.

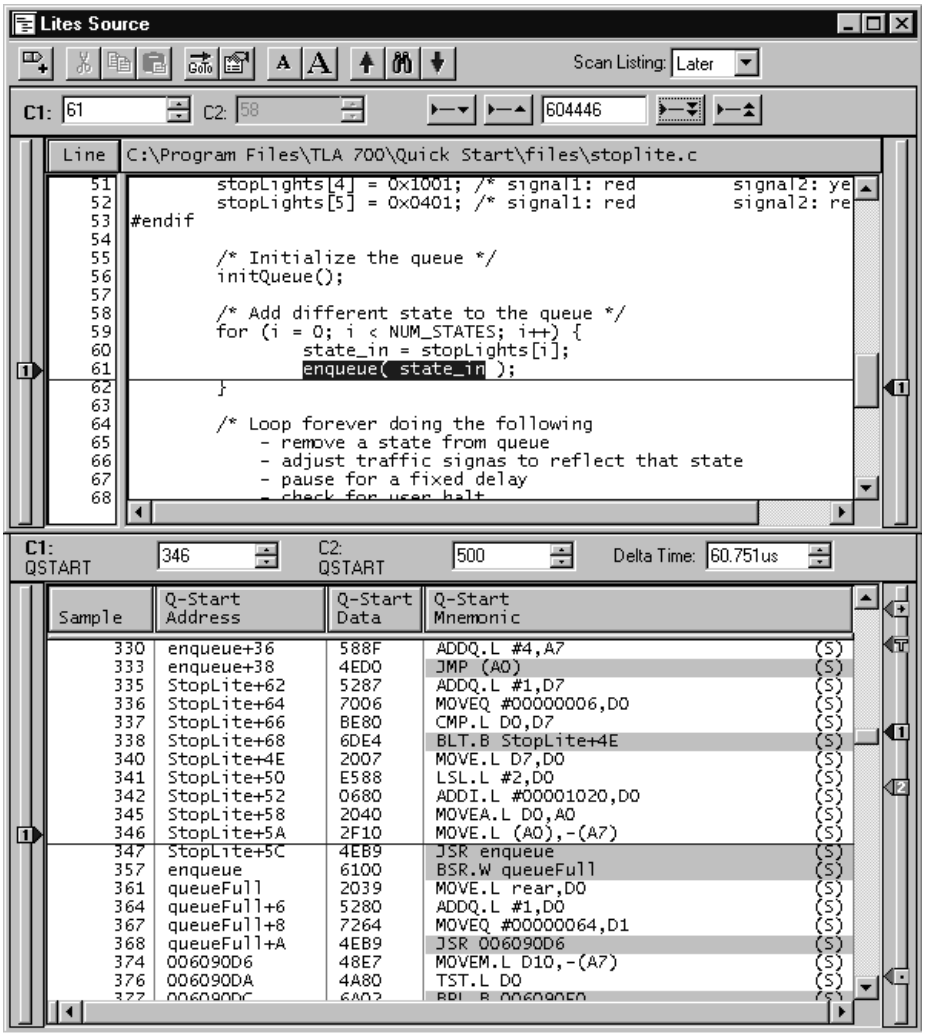


Figure 4–10: Source and Listing windows after step 3

Moving Backward in Execution Order in Source Files (Unique to the Tektronix Logic Analyzer)

The following section demonstrates a cursor movement technique in the Source window that is unique to Tektronix Logic Analyzer.

Earlier, we saw how the Step to Next Mark function of the logic analyzer was similar to the Run to Breakpoint capability of a software debugger. Now we will look at the logic analyzer's ability to Step or Run backward, which is unparalleled by any function of a software debugger.

Because the logic analyzer maintains a deep execution trace buffer, it can offer this unique ability. This feature is especially helpful when tracking down a real-time problem in your embedded system, where you are typically starting from an observed symptom and tracing back to the root cause. Therefore, you need the ability to go back through the real-time execution history of your code.

The ability to quickly step back up the execution stream is also a useful technique for unraveling a deeply nested call stack when browsing source files. For example, you can place a mark in the source statement prior to entering a routine, then you can dive as deep as you need into the call stack and pop back up to the top level with a single click of the mouse button. To demonstrate this feature, execute the following steps:

1. After you have placed a mark at line 61 in the current source (stoplite.c), step down the enqueue() routine two levels by clicking on the Step Forward button in the Source window twice. Figure 4–11 shows the results of this operation.

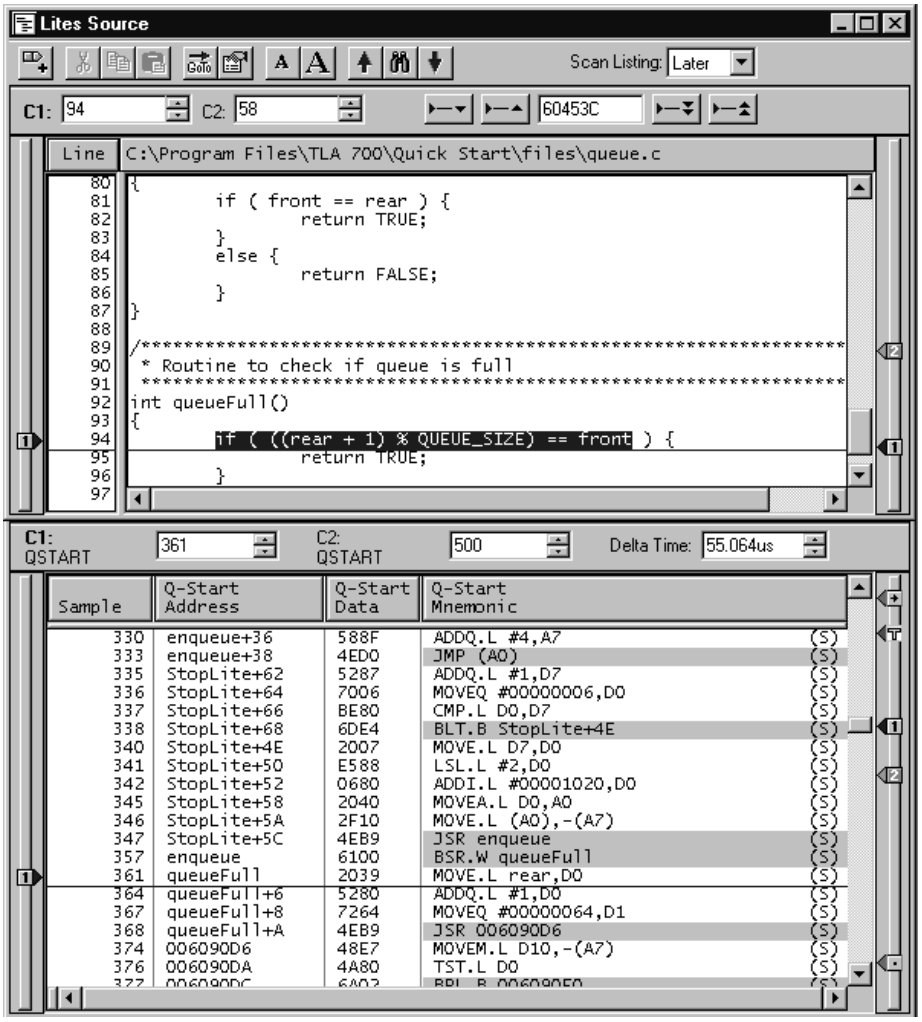


Figure 4–11: Source and Listing windows after stepping forward twice

2. Now return to where you placed the previous mark at the enqueue() routine by clicking on the “Previous Mark” button (item G in Figure 4–13 on page 4–20) once. The active cursor (cursor 1) in the Source window should now be positioned at the marked location (line 61 in the file stoplite.c).

3. Push the “Previous Mark” button once more to return to the previous execution of the source statement on line 61.

Note that the Listing window cursor is now on sample 180, whereas the first time we stopped on sample 61, the Listing window cursor was on sample 179. Samples 179 and 180 are both associated with the source statement on line 61. The Listing window cursor is placed on the first sample it encounters, depending on the direction of movement (forward or backward).

This completes the section *Using the Source Window.*

Triggering on Source Code Statements

You can set up the logic analyzer to trigger on a source code statement. There are two ways to accomplish this:

- Use the address displayed in the Active cursor readout (item D in Figure 4–13 on page 4–20) as the trigger value for the Address Group.
- Use symbolic names displayed in the Source window as a reference value for a trigger condition.

The following steps show how to trigger on a source code statement by using the address value displayed in the Active cursor readout. Specifically, you will set the logic analyzer to trigger on the call to the initqueue() routine (line 56 in the Source window).

1. Change the Scan Listing box (item F in Figure 4–13 on page 4–20) to Auto.

   The Auto setting configures the Source window to automatically set the direction that the Listing window looks for a corresponding acquisition sample. (For more information, see the online help topic *Scan Listing mode*).

2. Position the mouse pointer over the call to initqueue() at line 56 (Stoplite.c) in the Source window, then right-click the mouse and choose Move Cursor 1 Here.

3. Double-click on the address value displayed in the Active cursor readout, (item D in Figure 4–13 on page 4–20) and copy it to the clipboard (for example, Ctrl-C or right-click the mouse and select copy). This is the base address for the source statement marked by the active cursor (cursor 1) in the Source window.

4. Open the LA Trigger window (QSTART) by clicking on the Trig button on the Logic Analyzer module icon.

5. Click on the If/Then button to open the Clause Definition dialog box.

6. Change the Group Radix from Symbolic to Hex.

7. Select the top-right list box specifying the desired address and paste the copied “Active cursor readout” value into this field. Click OK

Your trigger window will now look similar to Figure 4–12 on page 4–19.

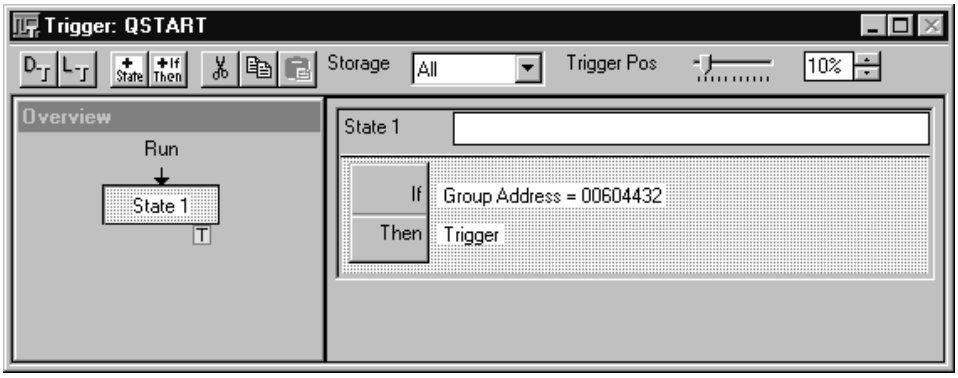


Figure 4–12: Trigger window containing the copied address

8. Click the Run button and wait for it to change to Stop.

9. Run the STOP LITES program on the training board:

a. Push the DN (F2) button on the training board to scroll down the program list until you reach STOP LITES.

b. Push the RUN (F3) button to run the STOP LITES program.

c. When the logic analyzer triggers, stop the STOP LITES program by pushing the STOP (F4) button on the training board.

10. The logic analyzer triggered on the call to the initqueue() routine. Reposition cursor 1 to the trigger location by first selecting the Listing window. Click on the Go To button in the Listing window toolbar, then select "QSTART: Trigger" from the list.

11. The sample containing the trigger location appears in the Listing window. Position the mouse pointer anywhere over the sample containing the trigger mark in the Listing window.

12. Right-click the mouse and select Move Cursor 1 Here. Your Source window should now position cursor 1 at the statement that triggered the logic analyzer. This statement is the call to the initqueue() routine (line 56).

A final topic related to *Using the Source Window* is recognizing when the Source and Listing windows are uncorrelated. There are situations when you might move the cursor in the Listing window to a location where there is no corresponding source statement. For these situations, the cursor in the Source window remains at the current location and changes the color of the Source text to signify that an uncorrelated state exists between the Source and Listing windows. When this happens, you can click the Step Forward or the Step Backward buttons to search for an executable source line in the Listing window; this correlates the active cursors in both windows.

Source Window Features

Figure 4–13 highlights some Source window features.

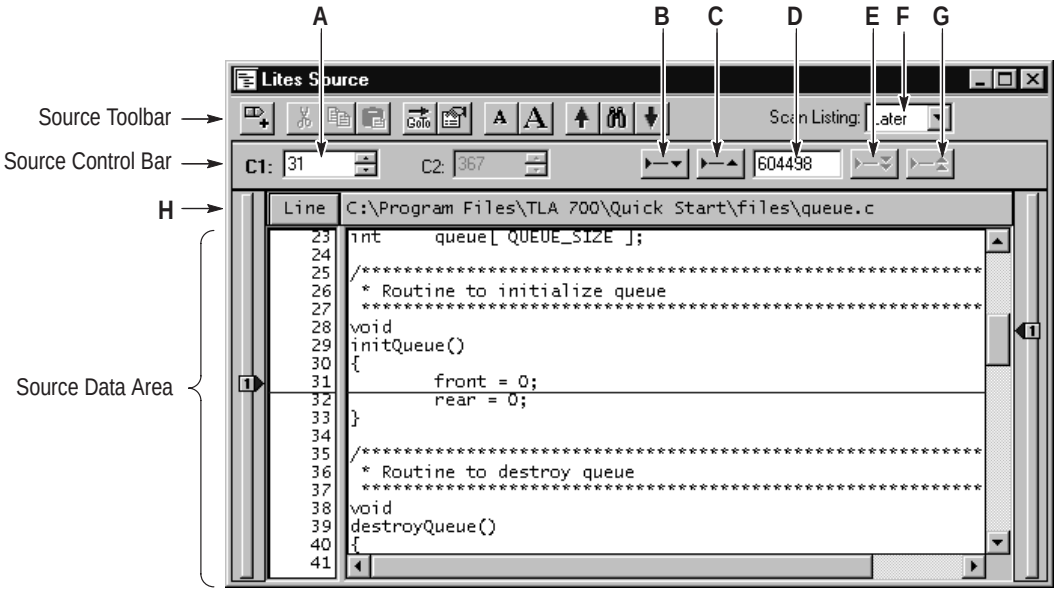


Figure 4–13: Source window

A. Use the cursor controls in either the Source or Listing window to move the cursors. You can also move the cursors in either window by clicking and dragging the cursor handles, by adjusting the cursor spin box controls, or using the Ctrl-arrow keys.

B. Use the Step Forward (item B) and Step Backward (item C) buttons to trace the order of execution of source statements. Click the Step Forward button to move to the next executed source statement.

C. Click the Step Backward button to move to the previous executed source statement.

D. "Active cursor readout" displays the address of the source statement at the active cursor. The address shown is the low address bound. The radix of the address is always hexadecimal (see the online help topic *Active cursor readout* for its usage).

E. The Next Mark and Previous Mark buttons operate similar to the Step Forward and Step Backward buttons. However, rather than stepping through every executed statement, you can define marks in the Source window as breakpoints and then step between marks in execution order to move though the source code.

F. The Scan Listing mode specifies the direction to search for samples in the associated Listing window. Moving the cursor in the Source window does not necessarily move the cursor in the same direction in the Listing window. (See the online help topic *Scan Listing mode* for more details).

G. The Previous Mark and Next Mark buttons operate similar to the Step Backward and Step Forward buttons. However, rather than stepping through every executed statement, you can define marks in the Source window as breakpoints and then step between marks in execution order to move though the source code.

H. The header of the Source Data Area shows the path of the displayed source file. Clicking on the header brings up the Source window property sheet.

Understanding the Tektronix Logic Analyzer Symbol Support

The logic analyzer has an integrated, global symbol capability that can read and load symbols from a wide variety of object file formats including IEEE695, OMF51, OMF86, OMF166, OMF286, OMF386, COFF, Elf/Dwarf1, Elf/Dwarf2, Elf/Stabs, and an ASCII format called Tektronix Logic Analyzer Symbol File, or TSF. Once loaded into the Tektronix Logic Analyzer global symbol database, all loaded symbols are available to the various tools in the Tektronix Logic Analyzer application (Setup, Source, Histogram, and Listing windows).

You can easily view which symbol files are loaded:

1. Select Symbols from the System menu. The Symbols dialog box appears (Figure 4–14).

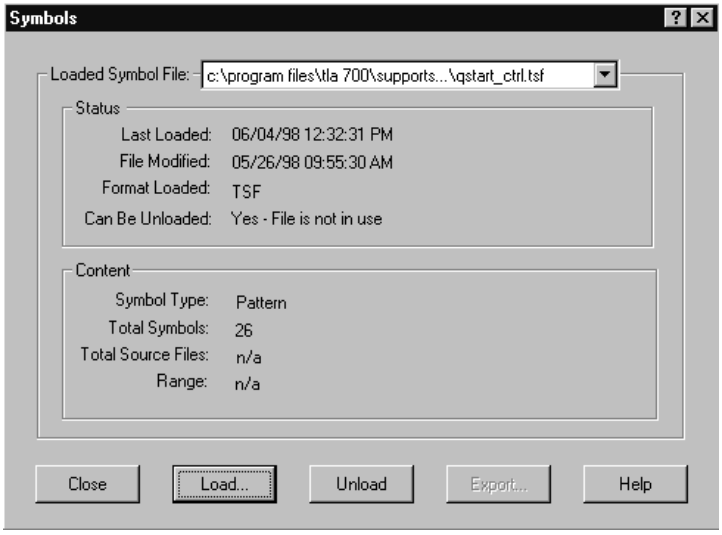


Figure 4–14: Symbols dialog box

2. Click the Load button.
3. Browse to the C:\Program Files\TLA 700\Quick Start\Files directory. (See Figure 4–15).
4. Select the Tla7qs.x file to load. This file is in the IEEE695 file format.

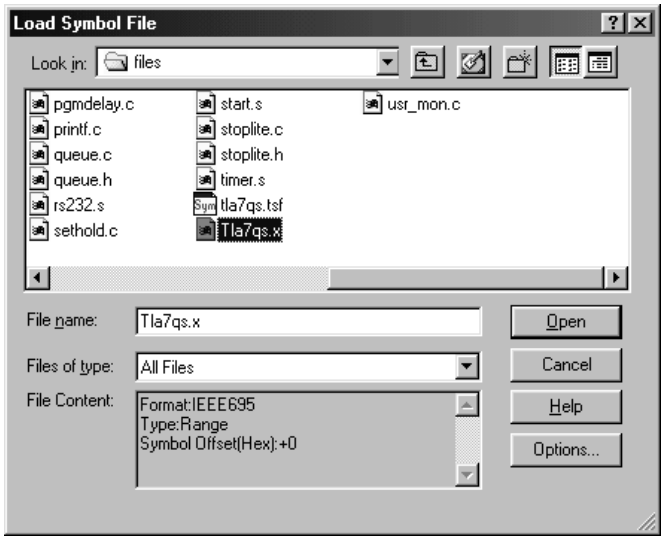


Figure 4–15: Select Symbol File dialog box

5. Click the Options button. The screen shown in Figure 4–16 appears.

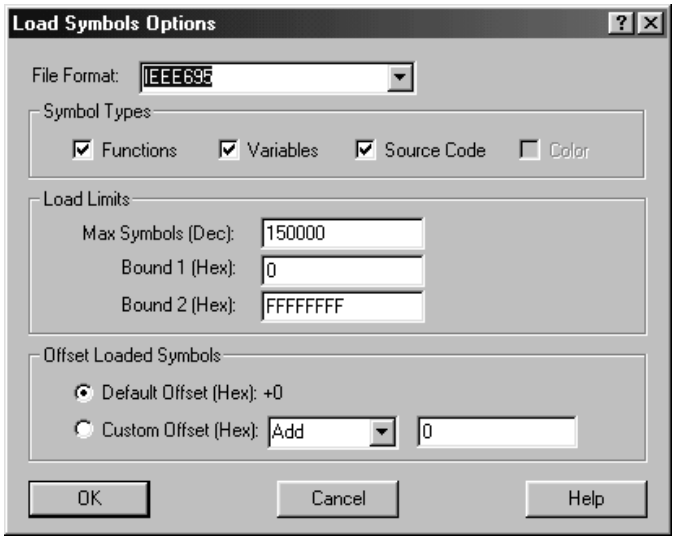


Figure 4–16: Load Symbols Options dialog box

Note that you are given the choice of four range symbol types to load: Functions, Variables, Source Code, and Color. You are also given selections for how many symbols to load and whether to apply an offset. Make sure all available range symbol types are checked.

6. Click OK.

7. Click Open to open and load the file. The Symbols dialog box reappears as shown in Figure 4–17.

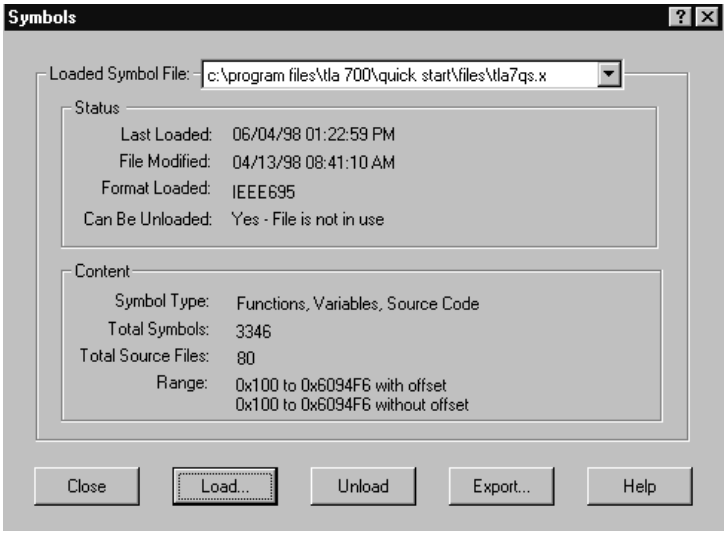


Figure 4–17: Symbols dialog box

8. Click the Export button, which exports the loaded symbol file in an ASCII format.

9. Save the symbol file to a temporary file called temp.tsf on the Windows Desktop, then close the Symbols dialog box.

10. Minimize the application window.

11. From the desktop, double-click on the temp.tsf file icon to open the file in WordPad. Note that the file contains three types of range symbols, shown in the following three figures:

    - Figure 4–18. Shows the function symbols. The function symbols describe the beginning and ending addresses of software functions.
    - Figure 4–19. Shows the variable symbols. The variable symbols describe the beginning and ending addresses of software variables.
    - Figure 4–20. Shows the source symbols. The source symbols describe the beginning and ending addresses of source statements.

The Source window uses only the source symbols to perform the linkage with the acquired data in the Listing window.

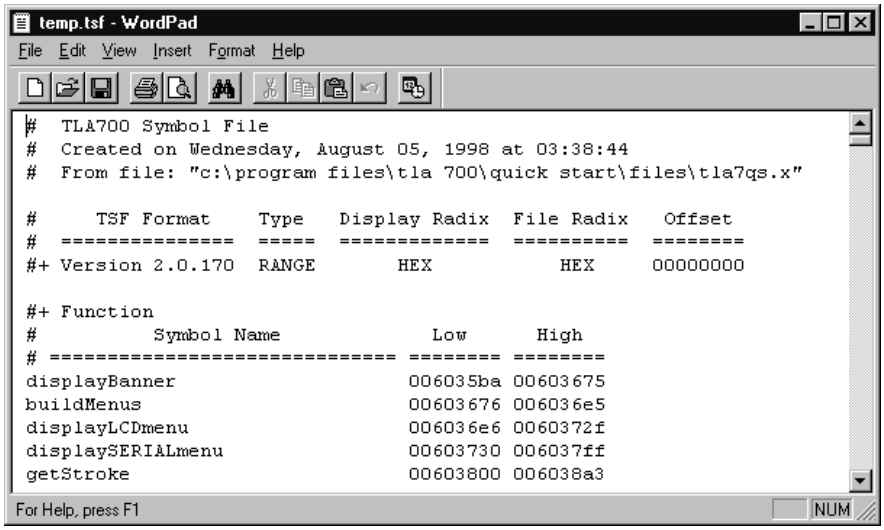


Figure 4–18: Function symbols

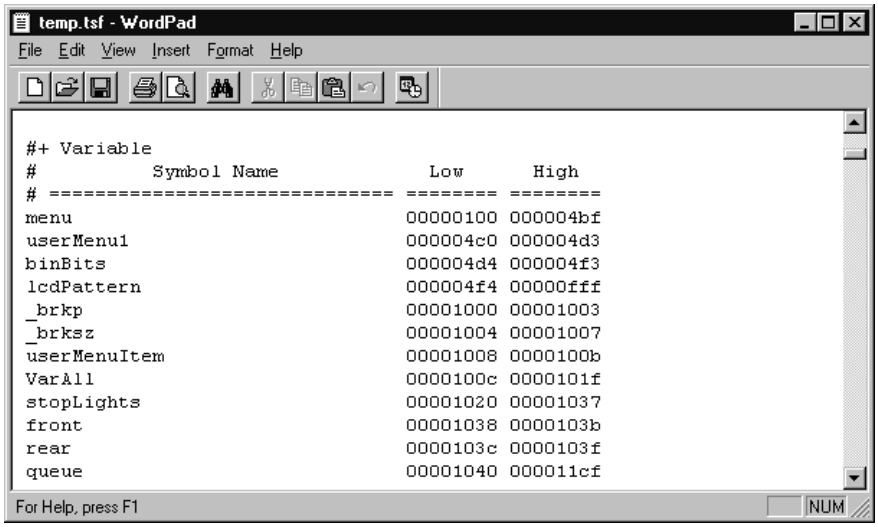


Figure 4–19: Variable symbols

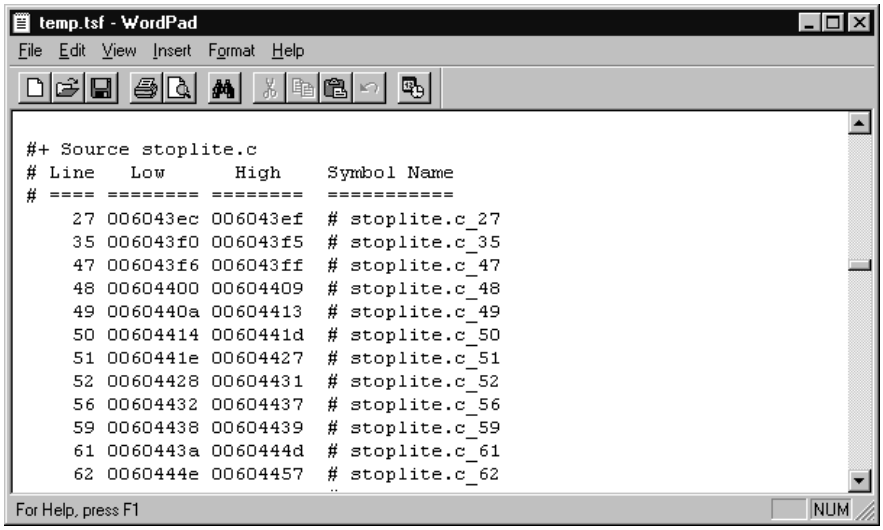


Figure 4–20: Source symbols

Summary

Understanding the contents of a loaded symbol file can be very useful in understanding how the various Tektronix Logic Analyzer tools operate. The exported file format is also a TSF file, so you can edit the exported symbol file and then reload it into the logic analyzer.

Automating System Verification (Exercise 2)

Often, a single measurement cannot fully characterize a timing failure. You might need to observe several failures or determine how frequently they occur to adequately analyze the problem. In this exercise, you will use the logic analyzer's memory comparison and repetitive acquisition capabilities to automate the verification of the embedded system on the QuickStart training board.

Specifically, you will set up the logic analyzer to repeatedly acquire data from the LA module, and compare it with data from a saved system setup file. When the compared data is not equal, the repetitive acquisition will halt and differences will be displayed.

Load the Saved System

1. Select Load System from the File menu.
2. Load the following saved system:
   C:\Program Files\TLA 700\Quick Start\Embedded Software Debug\
   Compare & Repetitive\2–Compare & Repetitive.tla.
3. The display should look similar to Figure 4–21.

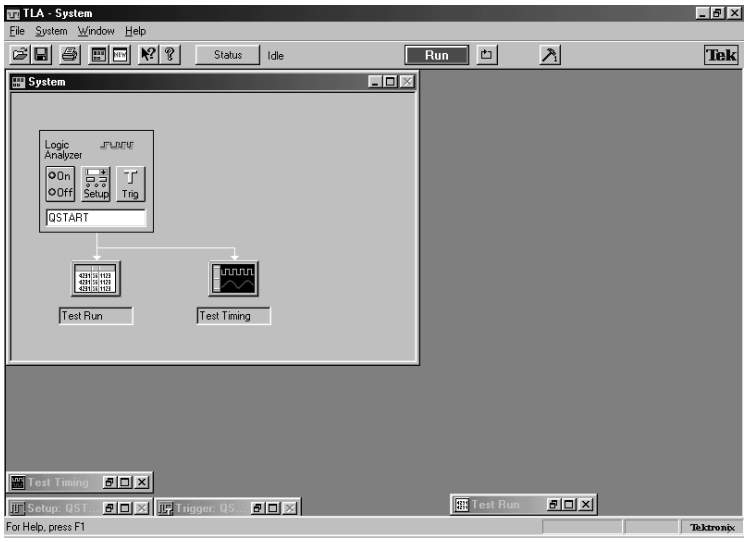


Figure 4–21: Restored system for Exercise 2

Create Reference Saved System

The following steps show you how to create a saved system setup that is used in this exercise as the reference source for performing memory comparison against repetitive test acquisitions.

1. Click on the repetitive button next to the toolbar Run button to set it to Single-run mode. (→)
2. Change the name of the "Test Run" Listing window to "Good Data" by selecting the title of the icon for this Listing window.
3. Change the name of the "Test Timing" Waveform window to "Good Timing."
4. Click the Run button, wait for it to change to Stop, then run the AUTO DELAY program on the training board. (The topic, *Run QuickStart Program and View the Acquired Data*, on page 4–35, shows how to do this.)
5. When the logic analyzer triggers, stop the AUTO DELAY program by pushing the F4 (STOP) button on the training board.
6. Save this system setup; it will be used as the reference saved system setup for this memory comparison and repetitive exercise:
   a. Select Save System As from the File menu.
   b. Ensure that you select Save all Acquired Data under the Save Options.
   c. Save the setup as "golden.tla", and place it under the directory: C:\Program Files\TLA 700\Quick Start\Embedded Software Debug\ Compare & Repetitive. (Overwrite the existing file if prompted.)
7. Open the Good Data Listing window (Figure 4–22).

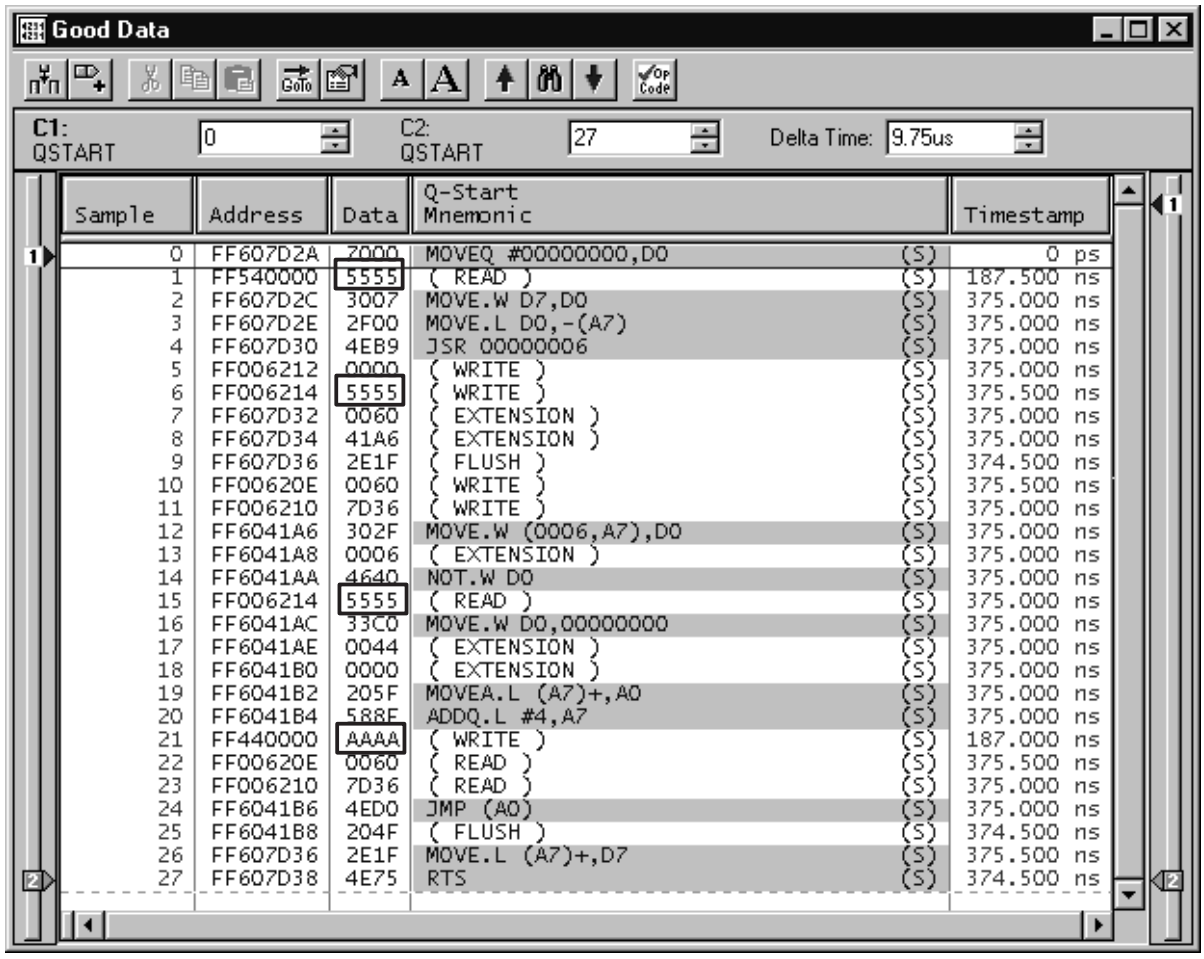


Figure 4–22: Good Data Listing window, used as reference data

The Listing window in Figure 4–22 shows data that will be used as the reference source to compare against future acquisitions. This listing is from a diagnostic routine, and the boxes at samples 1, 6, 15, and 21 identify the data that will intermittently change. Note that the values shown are the values determined to be valid.

8. Open the Good Timing Waveform window (Figure 4–23).

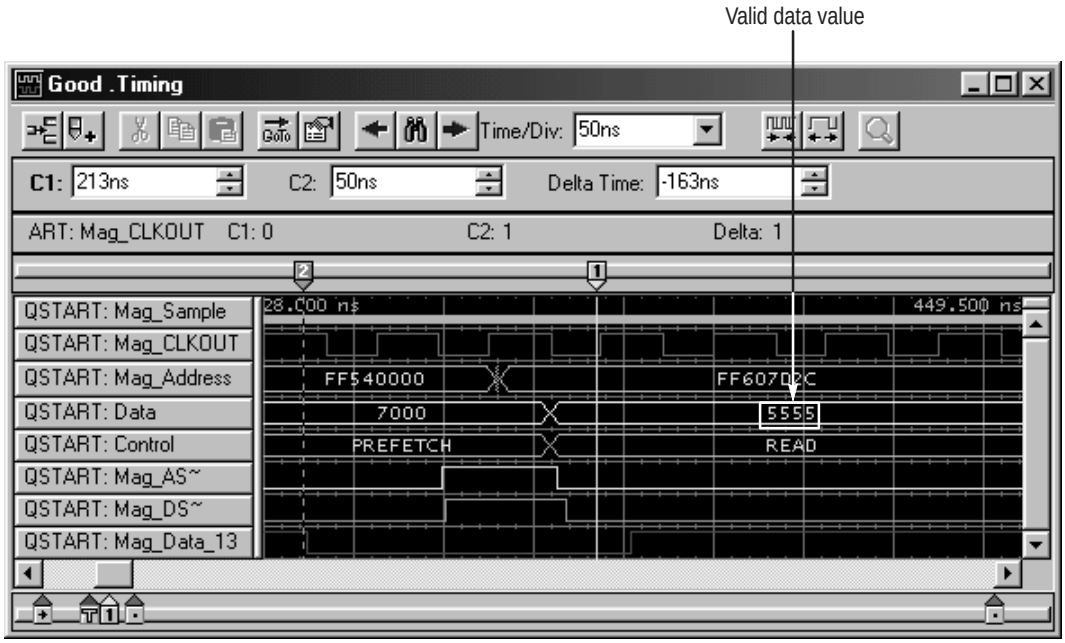


Figure 4–23: Good Timing Waveform window, identifying valid data read

The Waveform window in Figure 4–23 shows the detailed timing of the memory read cycle at sample 1, captured from the good acquisition run. Note that the valid data value read by the processor should be 5555.

Set Up Memory Comparison

Now that you have created the reference saved system setup, you are ready to set up the logic analyzer for repetitive acquisitions and memory comparisons.

1. Select Load System from the File menu. Reload the following saved system: C:\Program Files\TLA 700\Quick Start\Embedded Software Debug\ Compare & Repetitive\2–Compare & Repetitive.tla.
2. Click on the LA module Setup icon. The LA module Setup window appears (see Figure 4–24 on page 4–31).
3. Scroll down the Table Shows list and verify that Channel Compare is selected.

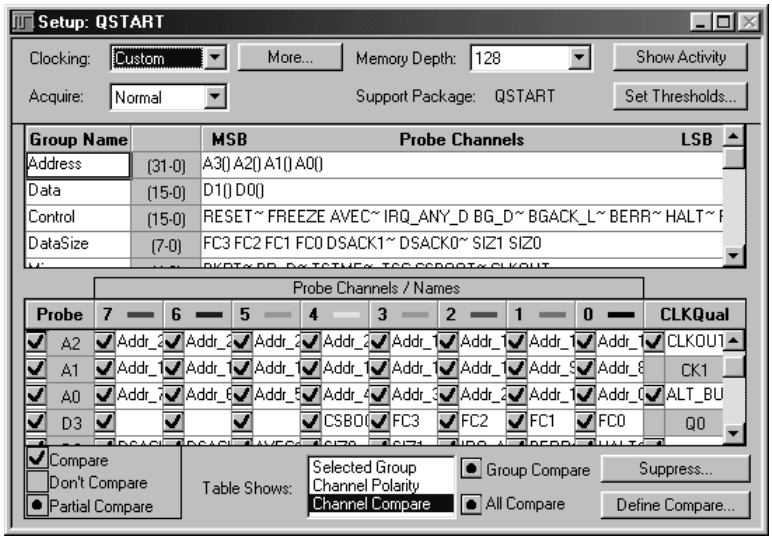


Figure 4–24: LA module Setup window

Figure 4–24 shows the Setup window where you define which groups or physical channels to compare. In this exercise, you will choose to ignore the upper 8 address channels (assigned to probe A3) in your comparison. You accomplish this by leaving the A3 probe unselected.

NOTE. *For 136 channel LA modules only: If you are using a 136 channel LA module, disable (unselect) the comparison on channels E3, E2, E1, E0, Q3, and Q2. Do this by scrolling down to the E channel probes in the Probe Channels/Names section of the Setup window and uncheck the boxes at the left of the screen (see Figure 4–25 on page 4–32).*

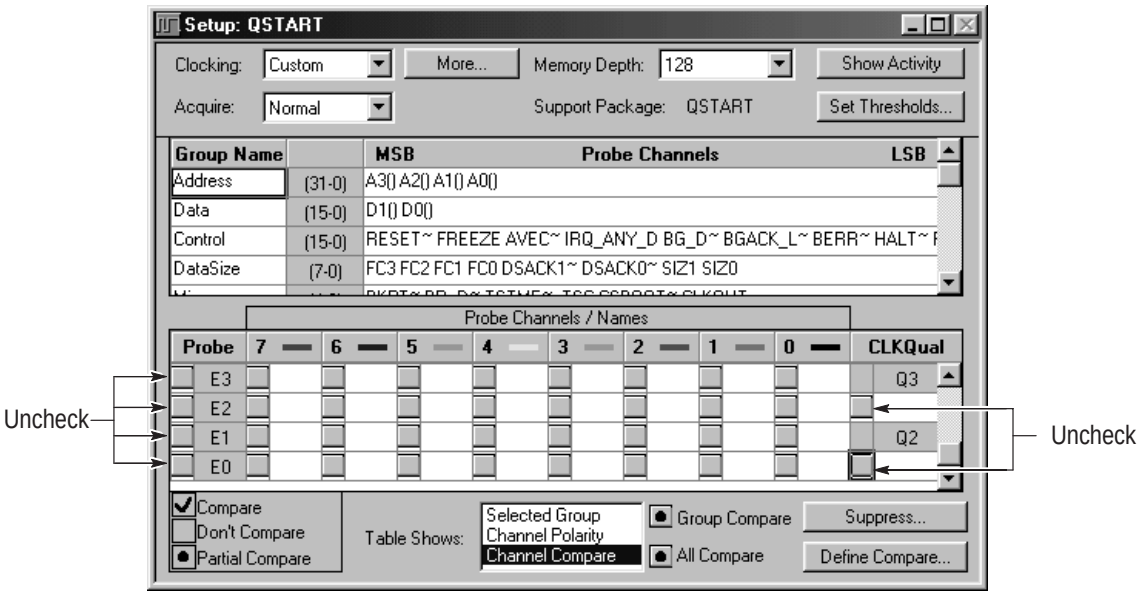


Figure 4–25: Disable the comparison (136 channel LA modules only)

4. Click the Define Compare button in the Setup window.
5. Click the Add Data Source button and browse to where you saved the reference data source that you created in the previous steps. This is the location used in step 6 of *Create Reference Saved System* on page 4–28.
6. Select Golden.tla as the reference data source.
7. Select the LA module (QSTART). Click Add. The LA module Define Compare dialog box appears, as shown in Figure 4–26. When you are finished viewing the dialog box, click OK.

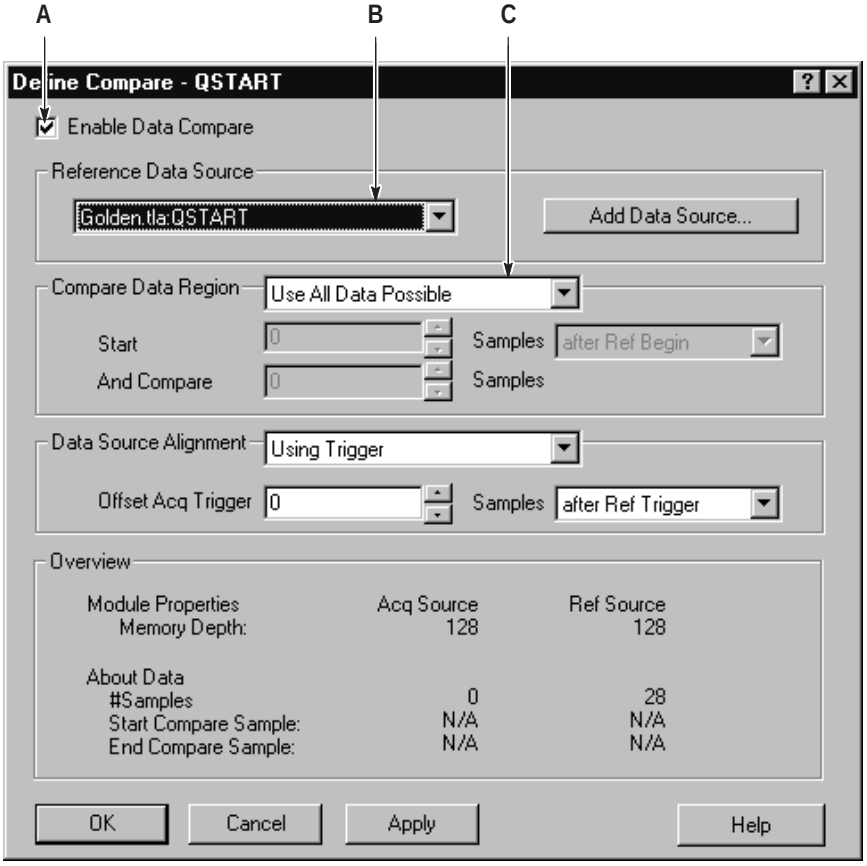


Figure 4–26: LA module Define Compare dialog box

The settings in the Define Comparison dialog box define the comparison criteria, as described below.

A. The Enable Data Compare check box enables memory comparison. Since this box is checked, memory differences will now be highlighted in the Listing and Waveform windows. The color assignment for differences is defined in the individual window property sheet.

B. A module in a saved system was chosen as the reference source for the comparison. The reference source could also be the acquisition memory of the same module.

C. This exercise is set up to compare the entire acquisition memory. However, it is possible to constrain the memory comparison to relevant samples.

Set Up Repetitive Acquisition

1. Select Repetitive Properties from the System menu. The Repetitive Properties dialog box appears (see Figure 4–27).

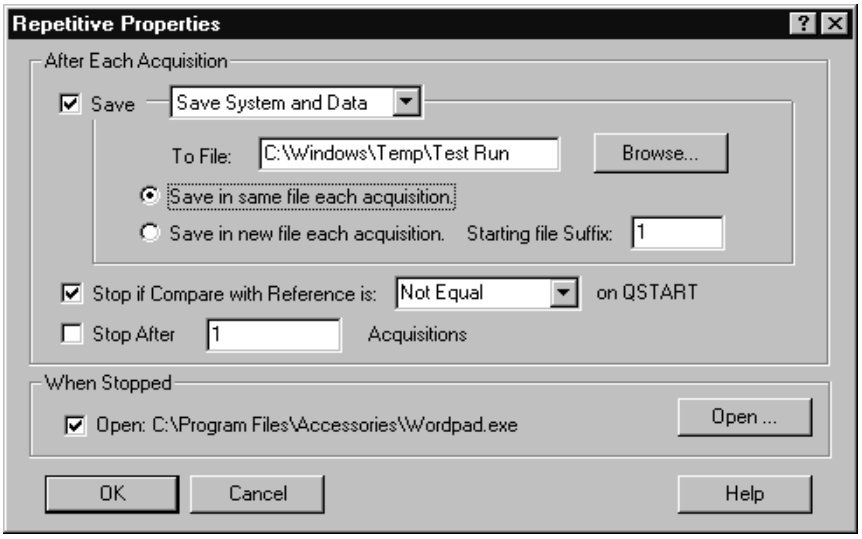


Figure 4–27: Repetitive Properties dialog box

Figure 4–27 shows the logic analyzer set up to do a comparison at each iteration of a repetitive acquisition. After each iterative acquisition, the logic analyzer saves the system setup and data to the same file. For this exercise, the repetitive acquisition is set up to stop if the reference source is *not equal* to the current repetitive acquisition. When the repetitive acquisition stops, the logic analyzer is set up to display a message defined in Figure 4–28. The logic analyzer can even notify you by email, paging, or other means, with the necessary software when your repetitive test finds a problem.

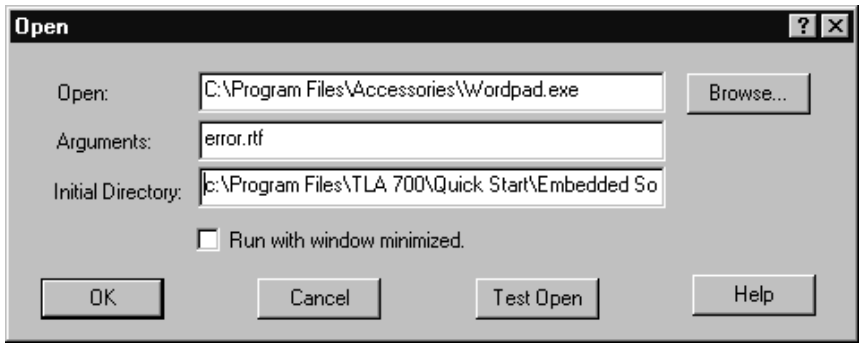


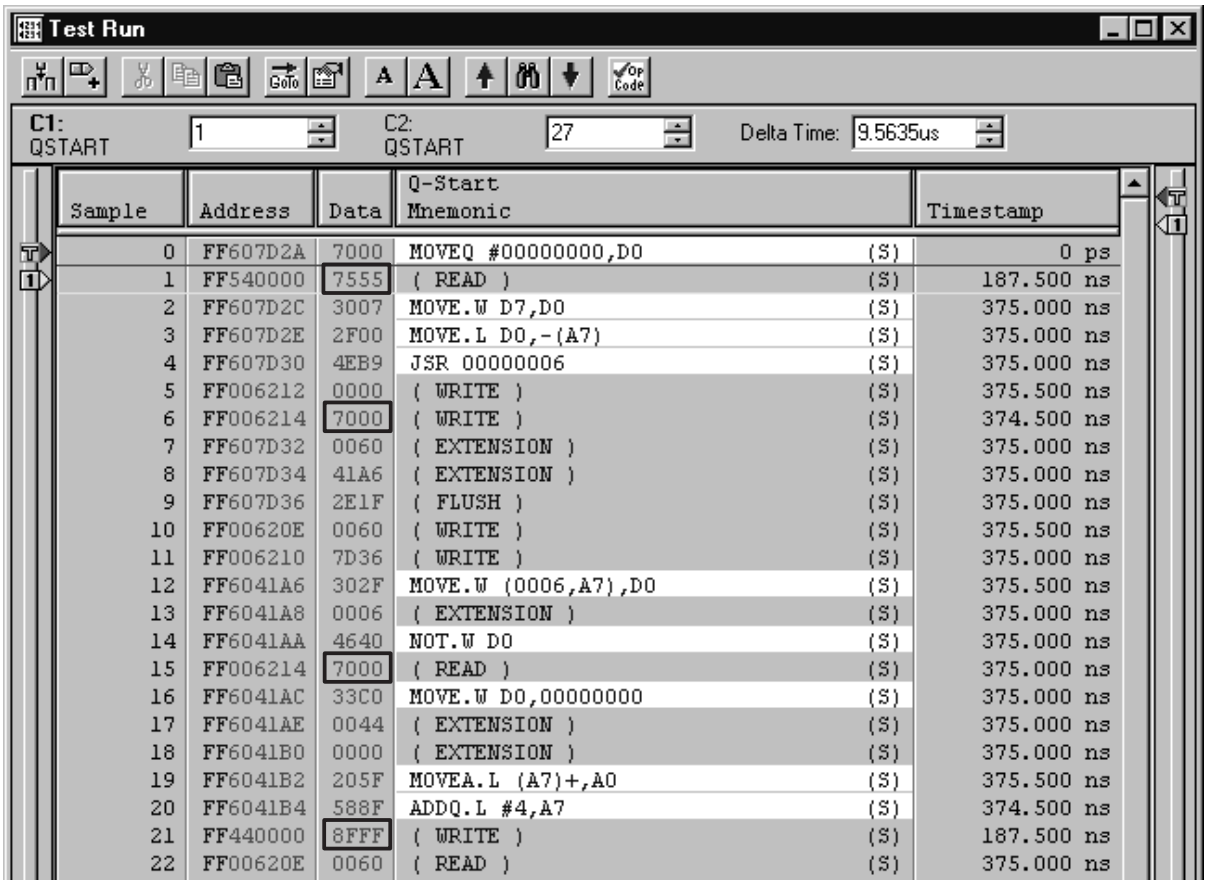
Figure 4–28: Set up the logic analyzer to display a message

Run the QuickStart Program and View the Acquired Data

1. Turn off the power to the QuickStart training board, then turn it back on. The power switch is located at the top-left corner of the board, near the power jack.
2. Open the Test Run Listing window.
3. Click the Tektronix Logic Analyzer Run button and wait for it to change from Run to Stop.
4. Push the DN (F2) button on the training board to scroll down the program list until you reach AUTO DELAY.
5. Push the RUN (F3) button to run the AUTO DELAY program.

The logic analyzer will make several acquisitions before the AUTO DELAY program generates a fault that causes the repetitive acquisition to stop. It could take several iterative runs before the logic analyzer finds a problem. Note that the logic analyzer indicates that it is saving each iterative test to a file by displaying status information next to the Status button.

When the repetitive acquisition stops, the logic analyzer will display the error message as specified in the repetitive dialog box. Differences are highlighted in red in the listing window. Channels that show no differences are displayed in green. Note that the upper 8 bits of the address bus are displayed in blue which indicates they were not compared, as specified in the Memory Comparison dialog box. (See Figure 4–29 on page 4–36).



Sample	Address	Data	Q-Start Mnemonic		Timestamp
0	FF607D2A	7000	MOVEQ #00000000,D0	(S)	0 ps
1	FF540000	7555	(READ)	(S)	187.500 ns
2	FF607D2C	3007	MOVE.W D7,D0	(S)	375.000 ns
3	FF607D2E	2F00	MOVE.L D0,-(A7)	(S)	375.000 ns
4	FF607D30	4EB9	JSR 00000006	(S)	375.000 ns
5	FF006212	0000	(WRITE)	(S)	375.500 ns
6	FF006214	7000	(WRITE)	(S)	374.500 ns
7	FF607D32	0060	(EXTENSION)	(S)	375.000 ns
8	FF607D34	41A6	(EXTENSION)	(S)	375.000 ns
9	FF607D36	2E1F	(FLUSH)	(S)	375.000 ns
10	FF00620E	0060	(WRITE)	(S)	375.500 ns
11	FF006210	7D36	(WRITE)	(S)	375.000 ns
12	FF6041A6	302F	MOVE.W (0006,A7),D0	(S)	375.500 ns
13	FF6041A8	0006	(EXTENSION)	(S)	375.000 ns
14	FF6041AA	4640	NOT.W D0	(S)	375.000 ns
15	FF006214	7000	(READ)	(S)	375.000 ns
16	FF6041AC	33C0	MOVE.W D0,00000000	(S)	375.000 ns
17	FF6041AE	0044	(EXTENSION)	(S)	375.000 ns
18	FF6041B0	0000	(EXTENSION)	(S)	375.000 ns
19	FF6041B2	205F	MOVEA.L (A7)+,A0	(S)	375.500 ns
20	FF6041B4	588F	ADDQ.L #4,A7	(S)	374.500 ns
21	FF440000	8FFF	(WRITE)	(S)	187.500 ns
22	FF00620E	0060	(READ)	(S)	375.000 ns

Figure 4–29: Test Run Listing window, identifying data that has changed

NOTE. *Your data at samples 1, 6, 15 and 21 in the Test Run Listing window may vary slightly from the data shown in Figure 4–29. However, any differences will be highlighted in both the Listing and Waveform windows.*

6. Open the Test Timing Waveform window (Figure 4–30).

Differences are also shown in the Test Timing Waveform window. The highlighted (red) portion of the data bus reveals that the microprocessor read a different value (7555), than the value it read in the reference data source (5555) in the Listing window in Figure 4–22 on page 4–29. Again, note that your test run data may not be 7555, but because it is highlighted in red, it differs from your reference data.

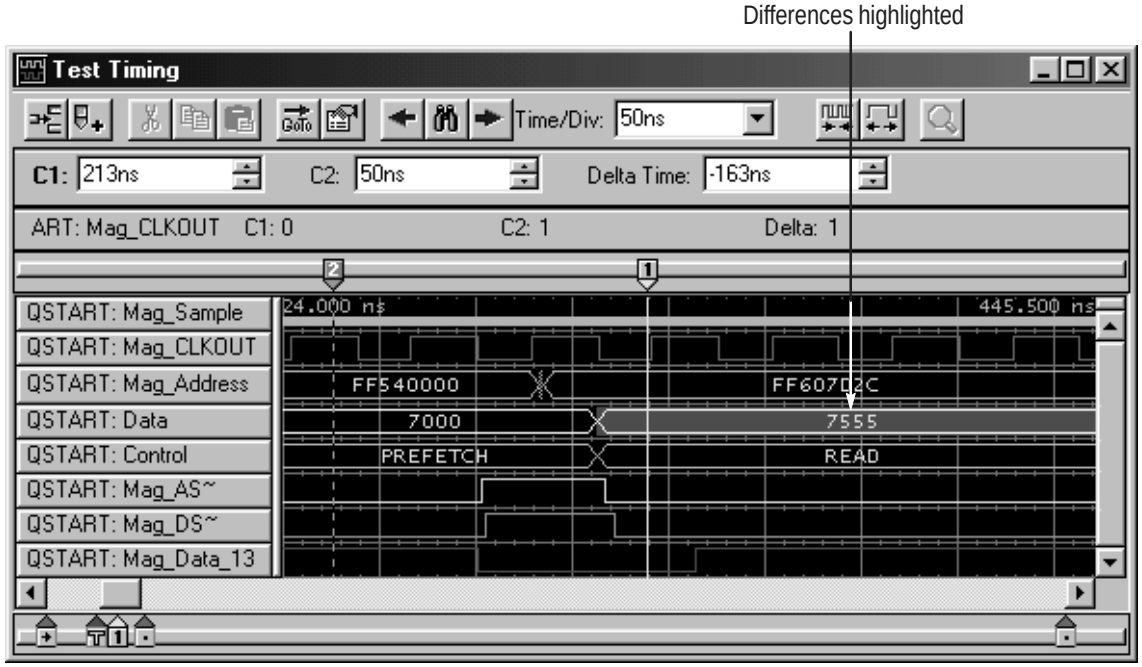


Figure 4–30: Test Timing Waveform window

View the Trigger Program

To understand how the logic analyzer was set up to capture the data in the above exercise, open the LA Trigger window (see Figure 4–31).

The logic analyzer was set up to capture and store only the routine of interest and to ensure that the acquired data was consistent for each repetitive acquisition run.

Figure 4–31: Trigger window for Exercise 2

NOTE. *Make sure to stop the AUTO DELAY program on the training board (F4 button), before moving on to the next exercise.*

Tektronix Logic Analyzer Programmatic Interface (TPI) Background (Exercise 3)

The Tektronix Logic Analyzer Programmatic Interface (TPI) is based on Microsoft's Component Object Model (COM). Using TPI, you can control the Tektronix Logic Analyzer from a separate user program running on the logic analyzer or on a remote PC.

In the context of TPI, the Tektronix Logic Analyzer application is called the server and the program that is written by the user and controls the logic analyzer through the TPI is called the client. The user program can be written in any language or programming environment that supports the Microsoft Component Object Model (COM), such as Microsoft Visual C++ or Microsoft Visual Basic.

General Characteristics

The following is a list of the general characteristics of the Tektronix Logic Analyzer Programmatic Interface (TPI):

- TPI is based on the Microsoft Component Object Model (COM). It is consistent with programmatic interfaces exported by other Microsoft Windows applications.
- All of the interfaces exported by the server are dual interfaces, supporting static and dynamic binding.
- The application must be fully initialized before a client attempts to connect to it. This includes dismissing any diagnostic errors that occur at startup time. At any given time, only a single client will be allowed to connect to the server. If a client attempts to connect when another client is already connected or before the application is fully initialized, it will receive an error indicating access is denied.
- A client running locally on the Tektronix Logic Analyzer will connect to an existing instance of the server, if there is one. If the server is not running, it will be launched automatically.
- Because of restrictions imposed by Microsoft Windows, a client running on a remote host cannot launch the server automatically. In this situation, you must launch the server before the remote client can connect.
- When a client connects to the server, the main window of the server is automatically hidden.

- Clients may show the server's window using the programmatic interface. If the window is shown, you can directly interact with the server. There will be an indicator in the status bar of the main window to show that a client is connected.
- The server will continue to run after a client is disconnected. The server window is always made visible when a client disconnects (in case the client hid it).
- TPI operates within the main thread of the application.

Please refer to the online help and TPI online help for more information on TPI.

TLAScript

TLAScript is a macro language for the Tektronix logic analyzer family. It is easy to learn and simple to use. You can use TLAScript to automate some of the tasks that you do on a regular basis.

TLAScript also provides a front-end to TPI. Therefore, TLAScript is a great way to get acquainted with TPI. It also provides easy ways for serious TPI users to quickly see how a particular TPI method behaves.

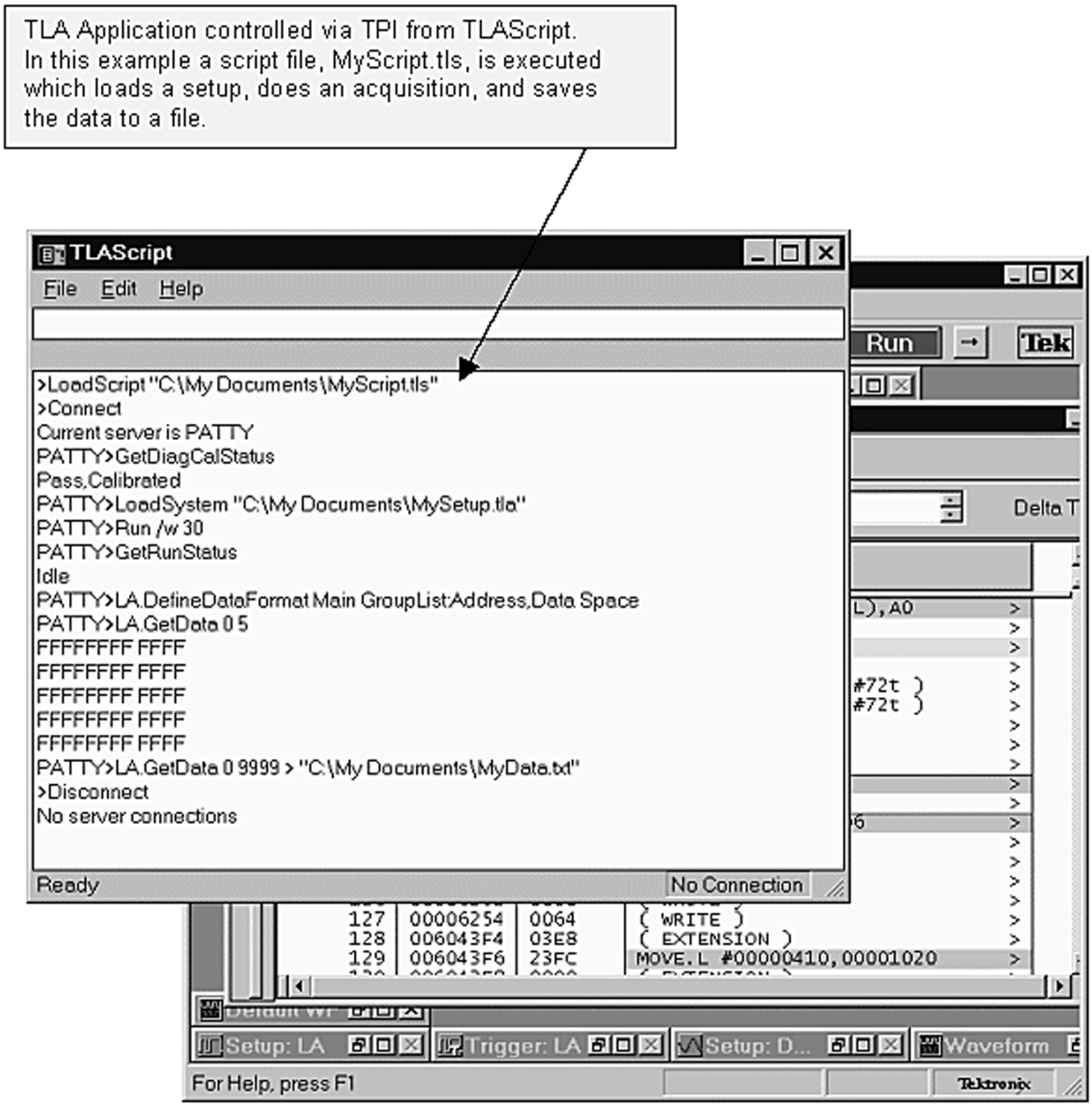
You can enter the TLAScript commands interactively or you can execute the commands from a script file. TLAScript can control the local logic analyzer application or a remote logic analyzer. You can run TLAScript from a regular PC; TLAScript is included in the TPI client software package.

Combined with a third party RSH daemon running on the Tektronix logic analyzer, you can use TLAScript to programmatically control the logic analyzer from a remote Unix work station.

Each TLA600 and TLA700 logic analyzer contains a sample TLAScript macro. To access and run the sample file, Select Start > Program Files > Tektronix Logic Analyzer > TLAScript. This starts the TLAScript program.

You can then open a sample file by selecting Open from the File menu and navigating to the following path:

```
C:\Program Files\TLA700\Samples|TLAScript Samples\Sample1.tls
```


TLA Application controlled via TPI from TLAScript.
In this example a script file, MyScript.tls, is executed
which loads a setup, does an acquisition, and saves
the data to a file.
TLAScript
File Edit Help
>LoadScript "C:\My Documents\MyScript.tls"
>Connect
Current server is PATTY
PATTY>GetDiagCalStatus
Pass,Calibrated
PATTY>LoadSystem "C:\My Documents\MySetup.tla"
PATTY>Run /w 30
PATTY>GetRunStatus
Idle
PATTY>LA.DefineDataFormat Main GroupList:Address,Data Space
PATTY>LA.GetData 0 5
FFFFFFFF FFFF
FFFFFFFF FFFF
FFFFFFFF FFFF
FFFFFFFF FFFF
FFFFFFFF FFFF
PATTY>LA.GetData 0 9999 > "C:\My Documents\MyData.txt"
>Disconnect
No server connections
Ready
No Connection
Run
Tek
Delta T
127 00006254 0064 (WRITE)
128 006043F4 03E8 (EXTENSION)
129 006043F6 23FC MOVE.L #00000410,00001020
Setup: LA
Trigger: LA
Setup: D...
Waveform
For Help, press F1
Tektronix

Remotely Controlling the Tektronix Logic Analyzer with the Programmatic Interface (Exercise 3)

The following exercise uses TPI to control the logic analyzer by remotely connecting, loading a module setup, acquiring data, and then transferring data using an external client application. In this case, the external client application is a program written in the Microsoft Excel Visual Basic for Applications (VBA) programming language.

Application Requirements

This exercise uses the Microsoft Office version of Excel. However, you can apply this concept to other COM/DCOM enabled applications. You should be familiar with Visual Basic or Visual Basic for Applications, Microsoft Excel, and the Microsoft COM/DCOM interface.

Client Application Description

The following summarizes the actions performed by the client application:

- Create and obtain a reference to an Application object. This automatically establishes a connection between the client and Tektronix Logic Analyzer server.
- From the Application object, get a reference to the Tektronix Logic Analyzer System object.
- Use the GetModuleBySlot method of the System object to obtain a reference to the desired LA module.
- Use the LoadModule method of the module object to load a module setup.
- Use the Run method of the System object to acquire data and wait for completion.
- Use the GetData method of the module object to extract the data from the LA module.
- Display the extracted data in a list box in an Excel spreadsheet.

Create Test Module Setup

The following steps will show you how to create a test module setup to be loaded onto your LA module using TPI.

1. Select Default System from the File menu.
2. Select the LA Module by clicking on the Logic Analyzer icon in the System window.
3. Select Load Support Package from the File menu.
4. Select QStart and click the Load button.
5. Select Save Module As from the File menu.
6. When the Save As dialog box appears:
   a. Enter “LAMod1” in the Filename field (by default, it will have the name LAMod1)
   b. Select the Don’t save the acquired data button in the lower left corner of the dialog box.
   c. In the Save In field, navigate to the following directory: C:\Program Files\TLA 700\Quick Start\Embedded Software\Debug\Tpi
   d. Click the Save button.

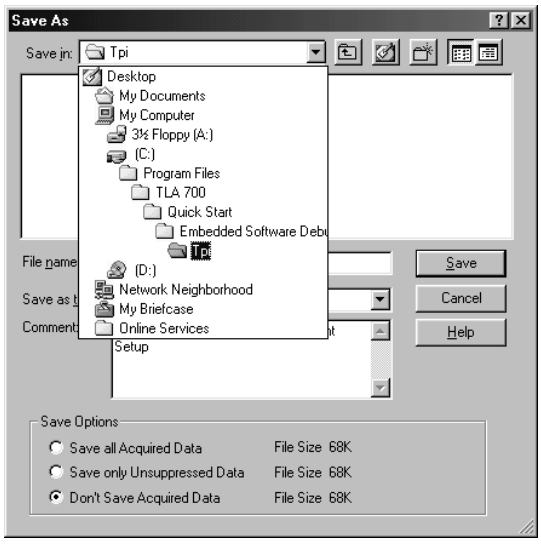


Figure 4–32: Saving Test Module Setup

Run the Client Application and View the Acquired Data

1. Make sure the logic analyzer status bar is visible:
   a. Select System from the File menu.
   b. Select Options, then Preferences.
   c. Select Status Bar, then select Show.
2. Verify that the training board is powered on.
3. Right-click on the Windows Start button. Select Explore and navigate to the following directory:
   C:\Program Files\TLA 700\Quick Start\Embedded Software Debug\Tpi
4. Run the TPI Client by double-clicking on the file TPI Client.xls. Select Enable Macros if asked by Excel. If necessary, resize the TPI Client window by clicking the restore button in the upper-right corner.
5. In the TPI Client window, enter the slot number for the LA module that you would like to load the test module setup onto.

NOTE. *The slot number is one of the two slots occupied by the LA module. Slot numbers are printed to the side of the modules on a portable mainframe, and above the modules on a benchtop mainframe. By default, the TPI Client assumes that your LA module is in the factory configuration – slot 3 or 4 for a TLA 714 mainframe or slot 3 for a TLA6xx mainframe. The TPI Client generates an error if you specify the wrong slot number for your LA module.*

---

6. Click the RUN button on the TPI Client.

NOTE. *If you cannot get the TPI Client to respond properly, or if TPI Client.xls comes up with a black background, try closing, then re-running TPI Client.xls as described in step 4.*

---

7. Clear the list box containing the output data by clicking the CLEAR button on the TPI Client.

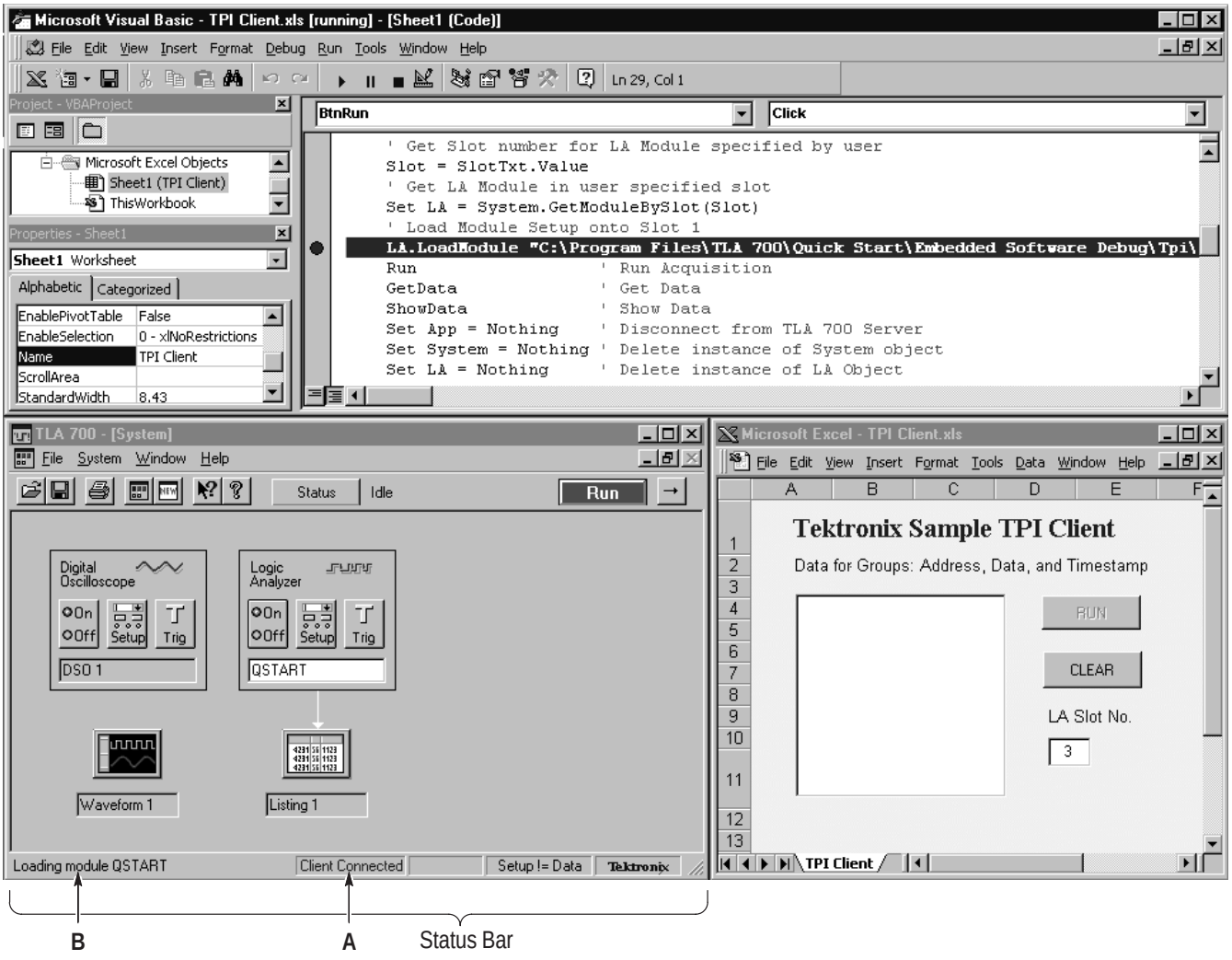


Figure 4–33: Client application connected to the logic analyzer server (before acquiring data)

Figure 4–33 shows the logic analyzer screen with the server, client, and VBA windows simultaneously displayed. Note the following information in the status bar in the server window:

A. Status shows that a client application is connected to the TLA 700 application.

B. The client application is loading a setup file to the LA module remotely using TPI.

Single-Step Through the Client Application

Now that you have seen what the client application does, we will take a look at how it accomplishes its task by single-stepping through the VBA code. The following steps show you how to bring up the source code to the client application and set a breakpoint to begin single-stepping. However, we will leave it as an exercise for you to step through the entire application. The source code is well commented and will provide you with a fundamental understanding of the actions taken by the application.

1. Select View → Toolbars → Visual Basic from the client's Excel menu.
2. On the floating Visual Basic toolbar (Figure 4–35 on page 4–48), do the following:
   a. Click the button that has the tooltip Design Mode.
   b. Click the button that has the tooltip Visual Basic Editor.
3. To set a breakpoint, scroll through the source window in the VBA editor until you reach the statement "On Error GoTo ErrorHandler" as shown in Figure 4–34.
4. Click in the location identified as Breakpoint in Figure 4–34. Your VBA Source display should look similar to Figure 4–34.
5. Select the TPI_Client.xls spreadsheet and click the "Exit Design Mode" button on the floating Visual Basic toolbar (Figure 4–35 on page 4–48).

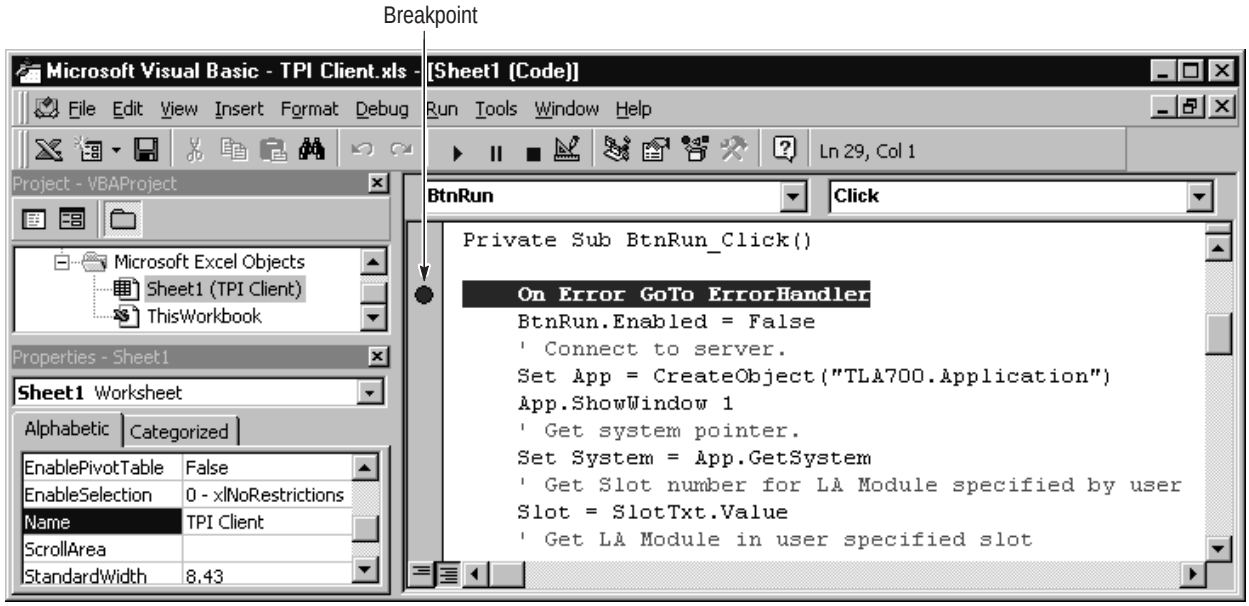


Figure 4–34: VBA Source window after setting the breakpoint

6. Click the RUN button on the Excel spreadsheet. Execution should stop where the breakpoint was set.

7. Single-step through the source code by pushing the F8 key or selecting Debug → Step Into from the menu in the Visual Basic Source window.

8. Continue to single-step (F8) until you reach the breakpoint. The output of the client application should look like Figure 4–35.

***NOTE**. If the TPI Client is not responding, or you need to reset it, close the Excel spreadsheet by selecting Exit from the File menu in the "TPI Client.xls" window. Run the spreadsheet again as described in step 5 on page 4–47.*

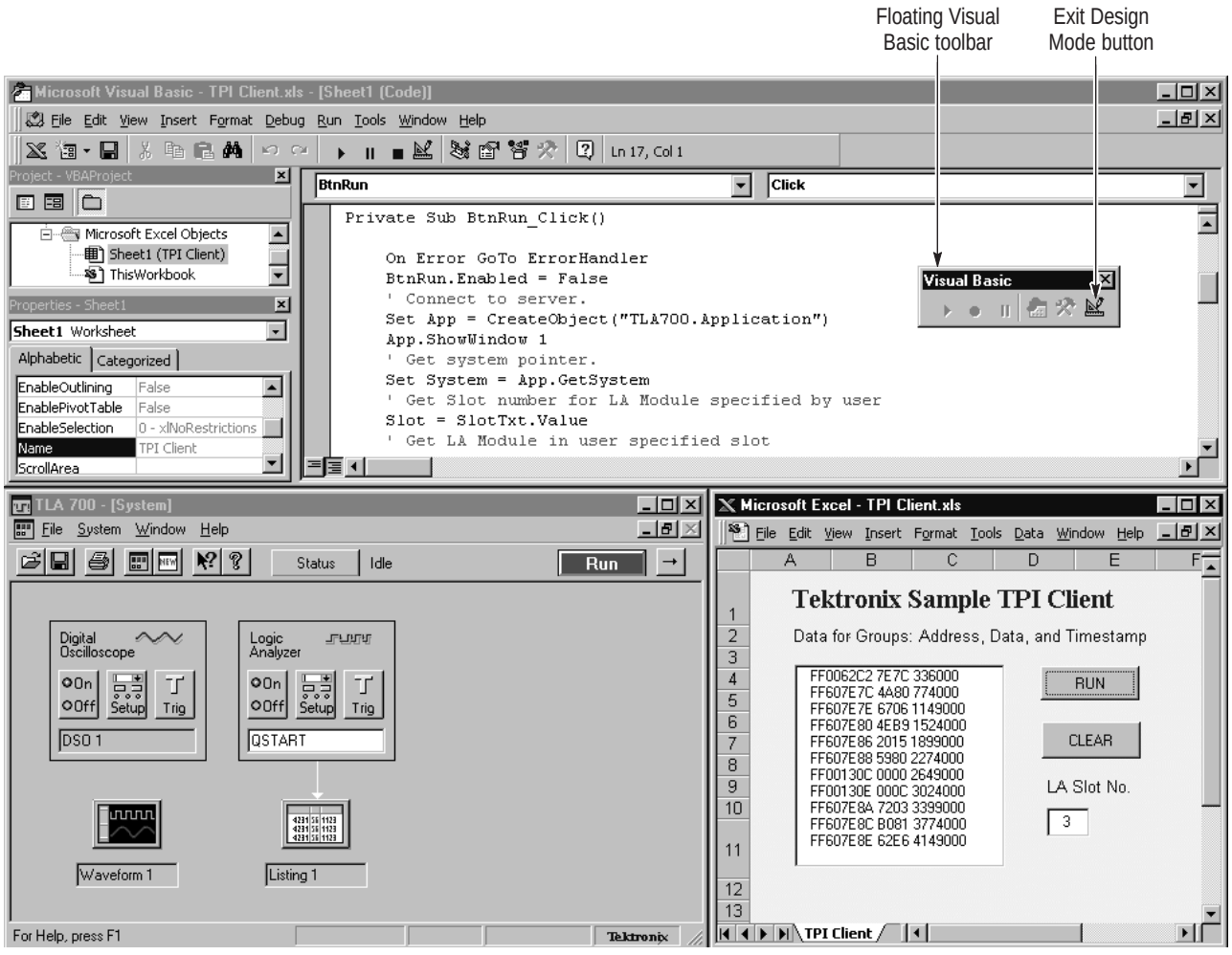


Figure 4–35: Results after clicking on the client application RUN button

9. Figures 4–36 and 4–37 show the contents of all the subroutines in the VBA client application.

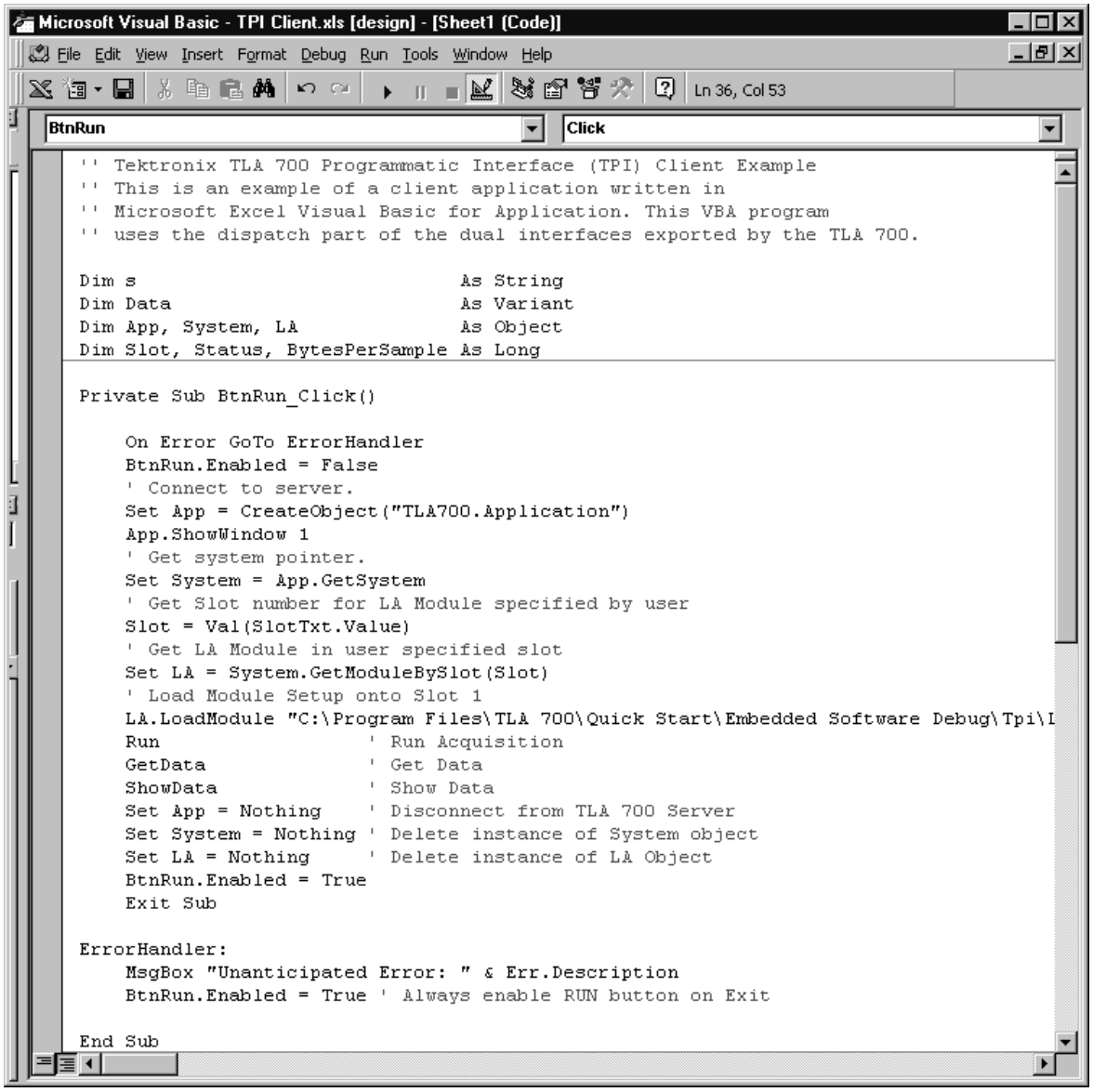


Figure 4–36: Source code of the VBA client application (part one of two)

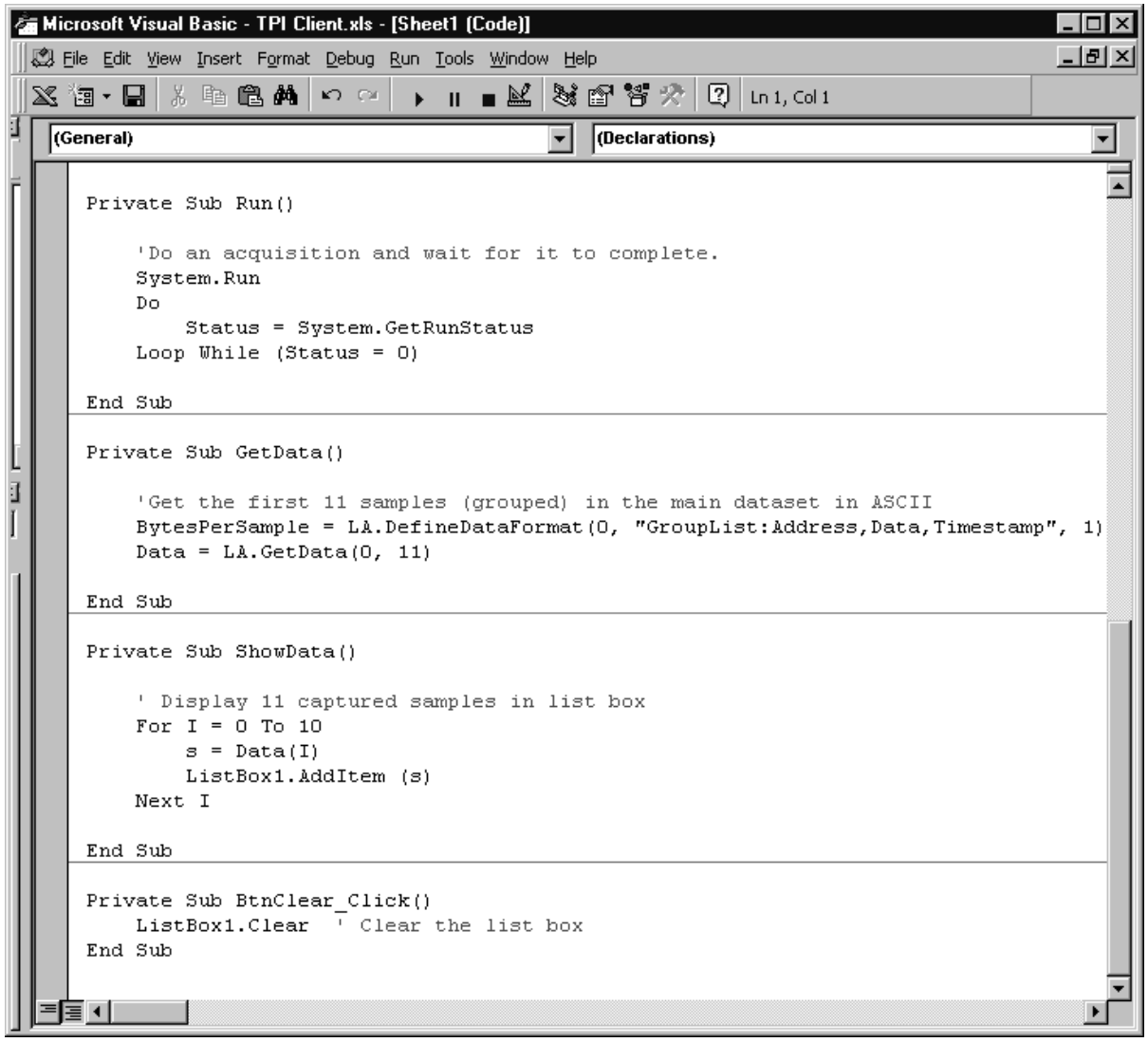


Figure 4–37: Source code of the VBA client application (part two of two)

Histogram Window Background (Exercises 4 and 5)

Histogram windows are used to display performance analysis data. Performance analysis is an automated data collection, reduction, and processing technique. It is used primarily for analyzing system performance once software is stable and is running on prototype hardware. Using performance analysis, you can trace software execution in real time.

The performance analysis capabilities of the Tektronix Logic Analyzer feature the ability to post-process logic analyzer acquisition results. Then, displayed in a Histogram window, are the number of samples of a specific channel group that fall into a set of user-specified ranges of values.

Histograms give a graphic profile of where time is spent during execution. Use the histogram to easily identify areas in your software which, when optimized, will yield the greatest improvements in system performance.

There are two basic modes of performance analysis implemented on the logic analyzer; one uses channel groups and the other uses counters and timers.

Histogram Window Using Channel Groups

The Histogram window provides an overall picture of system activity. In the channel group mode, you define ranges (numeric, symbolic, or logarithmic) to bin address hits. Each range consists of a low and high bound within a specific group. The histogram that is generated shows the samples acquired and processed, the number of occurrences falling within each range, and the percentage of overall microprocessor activity that these occurrences represent.

The logic analyzer begins to take repetitive acquisitions based on the conditions specified in the Trigger Specification Menu. The logic analyzer compares the data from these acquisitions to the defined ranges and keeps an accumulated count of the state values of the specified group falling within each range. This is a statistical form of performance analysis. The performance analysis is real-time when the desired data is captured within one acquisition.

Histogram Window Using Counters and Timers

The counter and timer mode provides a closer look at specific system events. In this mode, you define event measurements that specify a starting point, a target event, a measurement type, and a stopping point in the trigger menu.

The logic analyzer begins to take repetitive acquisitions based on the event measurement's starting and stopping points. The logic analyzer waits for the

defined events, and then triggers and processes the measurement. The logic analyzer then displays the results in a histogram format.

Three statistical calculations are automatically performed in event measurement mode: minimum, maximum, and average. You can make repeated measurements of an event duration, or count occurrences of events and compare the statistical distribution of the results. Using event measurement mode, you can validate system performance under real-time stress.

Characterizing System Performance (Exercise 4)

In this exercise, you will characterize the performance of your embedded system with the Histogram window, using channel groups. The channel group mode will provide an overview of your system execution so that you can quickly identify the areas where your application is spending the majority of its time. By optimizing this portion of your code, you will yield the greatest improvement in overall system performance.

Load the Saved System

1. Select Load System from the File menu.
2. Load the saved system:
   C:\Program Files\TLA 700\Quick Start\Embedded Software Debug\
   Overview PA\4–Overview.tla

The restored system should look similar to the one shown in Figure 4–38. You will not have a DSO icon if your logic analyzer does not contain a DSO module.

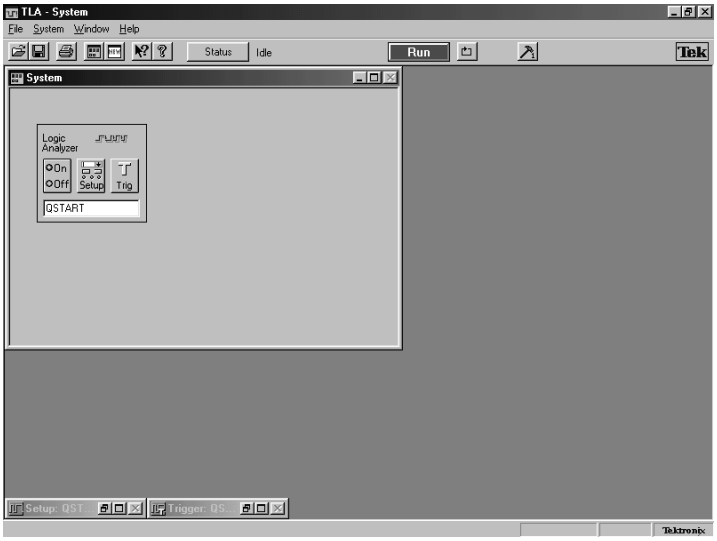


Figure 4–38: Restored system for Exercise 4

Create a New Histogram Window

1. Select New Data Window from the Window menu or the NEW button on the system toolbar.
2. Select Histogram from the dialog box. Click Next.
3. Select “Data from an LA in the system”. Click Next. The dialog box shown in Figure 4–39 appears.
4. Set your parameters to those shown in Figure 4–39. After setting your parameters, click Next.
5. Enter a name for the new data window (for example, Overview). Click Finish.

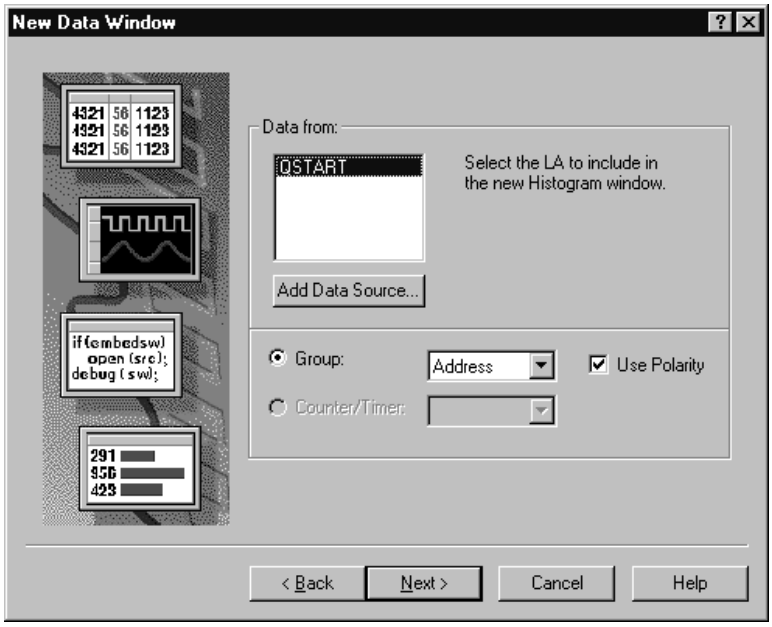


Figure 4–39: Setting the LA parameters

6. Click the Properties icon (top-left button of the toolbar, in the new data window you just named). The property sheet opens.
7. In the Histogram Window tab, set the Data Font Size to 10.
8. Click the Ranges tab.
9. Change the selection for “Define Ranges Using” to Symbolic.
10. Click the Symbol File button to load the symbol file under C:\Program Files\TLA 700\Quick Start\Embedded Software Debug\Overview PA\Overview.tsf. See Figure 4–40.
11. Click OK.

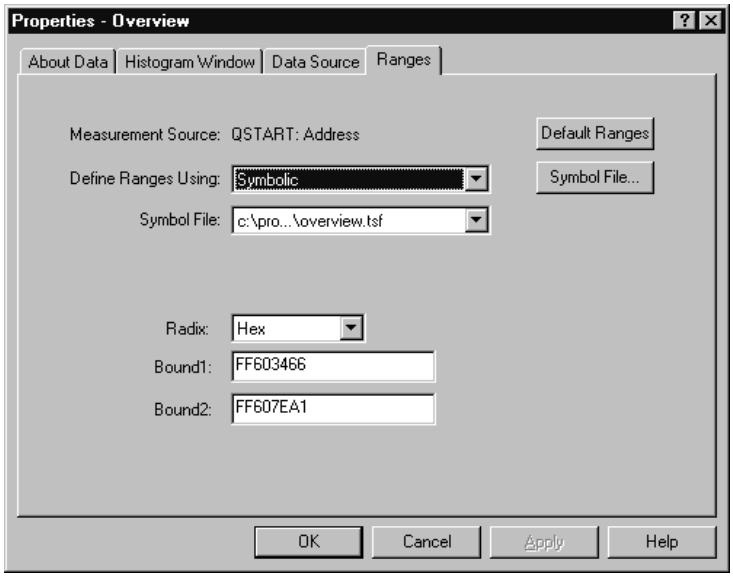


Figure 4–40: Loading the symbol file

Run the QuickStart Program and View the Acquired Data

1. Turn off the power to the training board, then turn it back on.
   The power switch is located at the top-left corner of the board, near the power jack.
2. Open the Overview Histogram window.
3. Adjust the column size in the Overview Histogram window to match Figure 4–41.

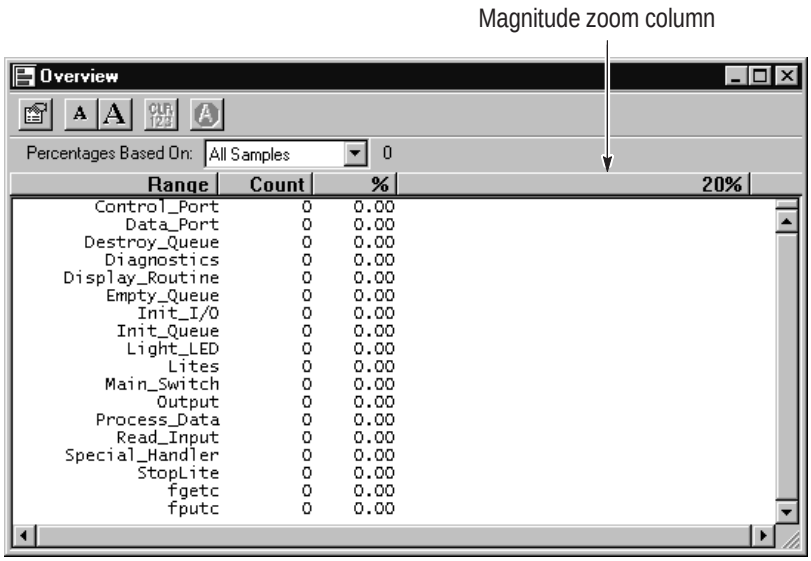


Figure 4–41: Customized Overview Histogram window

4. Click on the magnitude zoom column and select 20% zoom (see Figure 4–41).
5. Click the Run button on the logic analyzer to begin acquiring data. Let the logic analyzer capture data for about 1–2 seconds to get an overview of the system activity.

The activity that the logic analyzer is capturing is the monitor application running on the training board. The monitor is in a loop waiting for external events to occur. That is why the bars in the Histogram window stabilize after a few seconds. There should not be any programs running on the training board.

The captured data should look similar to Figure 4–42, which shows a graphic profile of where time is spent on the system (training board).

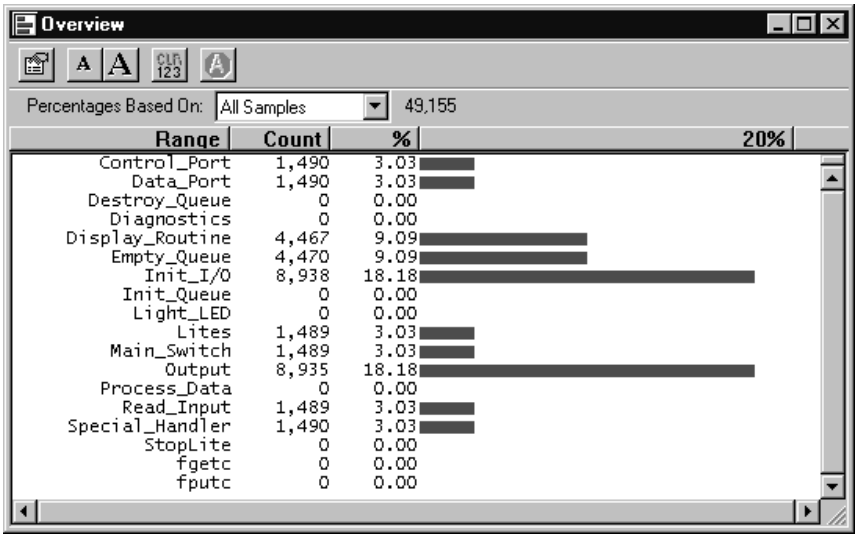


Figure 4–42: Histogram data window showing program activity during normal operation

Now, inject an external stimulus into the system and observe how the system responds. To simulate an external event that consumes a large portion of the processor time, you will run the LITES program on the training board.

6. Use the DN (F2) button on the training board to scroll down the program list until you reach LITES.

7. Push the RUN (F3) button to execute LITES.

Observe that system performance degrades as the microprocessor responds to this external interrupt by executing the Process_Data routine. You can see this happening in real time as the red bar next to the Process_Data routine grows and the bars from other routines shrink; the Process_Data routine slowly monopolizes CPU cycles. The data presented in the Overview window should look similar to the data presented in Figure 4–43.

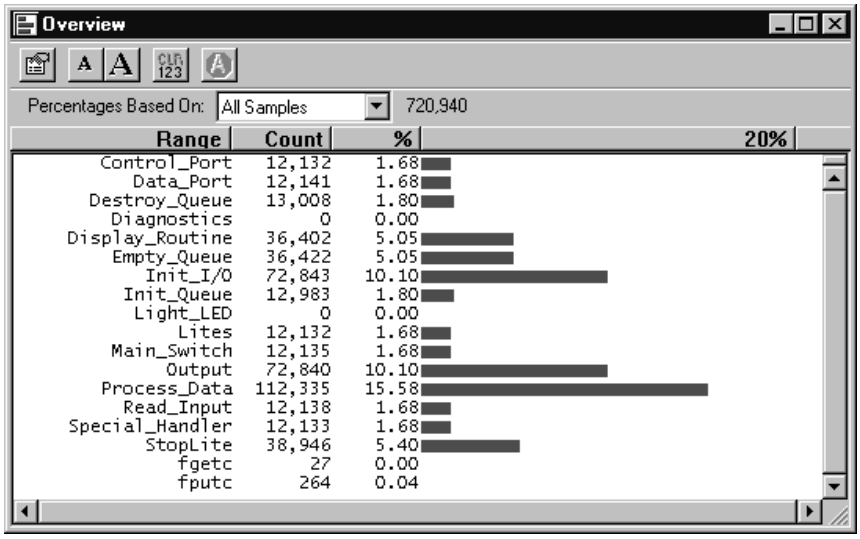


Figure 4–43: Histogram window showing where the application is spending most of its time

8. Click the Stop button after the count reaches about 400,000 samples (near the top of the **%** column).

9. Quickly identify high usage functions by clicking on the **%** column twice to sort by decreasing execution time.

The Overview window should now look similar to Figure 4–44. The function Process_Data has the most activity. That is where the application is spending the majority of its time. With this knowledge, you can optimize this routine, thereby achieving the greatest benefit in overall system performance.

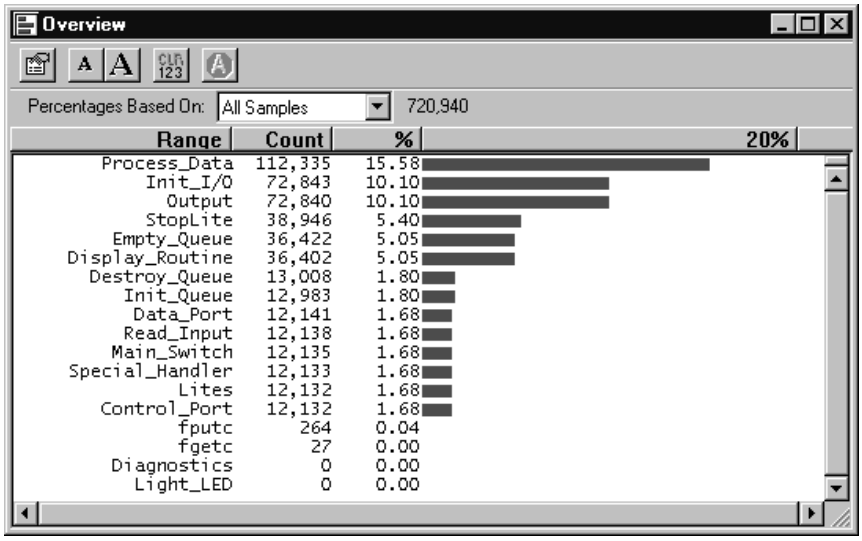


Figure 4–44: Histogram window showing program activity sorted by decreasing execution time

You can even split the Histogram window to leave specific areas of interest on the screen while you scan the rest of your code. See Figure 4–45.

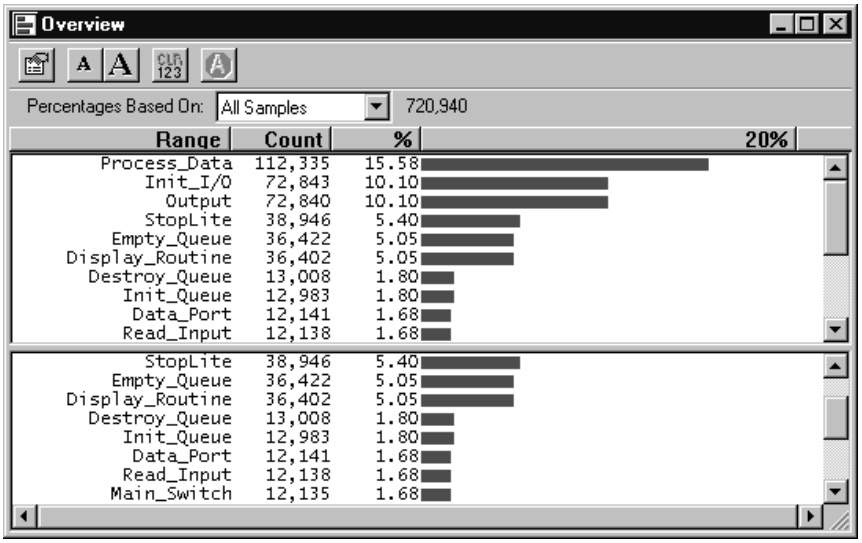


Figure 4–45: Overview Histogram window showing split screen capabilities

View the Trigger Program

To understand how the logic analyzer was set up to capture the data in the previous exercise, open the Trigger window (see Figure 4–46). The LA module was set up to trigger on anything. By triggering on anything for the Address Overview mode of performance analysis, the logic analyzer was able to capture program activity on the entire address space of the embedded application running on the training board.

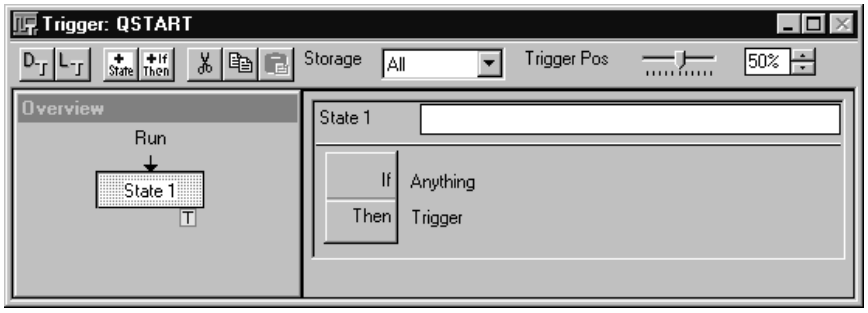


Figure 4–46: Trigger window for Exercise 4

Optimizing Execution Time of Embedded Software (Exercise 5)

In Exercise *2* on page 3–13, you used multiple trigger states and timers to measure interrupt latency of the buttons on the training board. In this exercise, you will use the same setup and trigger resources of the Histogram window. You will obtain a statistical distribution of interrupt latency in order to optimize the execution time of your embedded software.

Load the Saved System

1. Select Load System from the File menu.
2. Load the saved system:
   C:\Program Files\TLA 700\Quick Start\Embedded Software Debug\
   Event PA\5–Event Measurement.tla.

The restored system should look similar to the one shown in Figure 4–47. You will not have a DSO icon if your logic analyzer does not contain a DSO module.

Figure 4–47: Restored system for Exercise 5

Create a New Histogram Window

1. Select New Data Window from the Window menu or the NEW button on the system toolbar.
2. Select Histogram from the dialog box. Click Next.
3. Select “Data from an LA in the system”. Click Next.
4. Set your data source parameters to those shown in Figure 4–48. After setting the parameters, click Next.

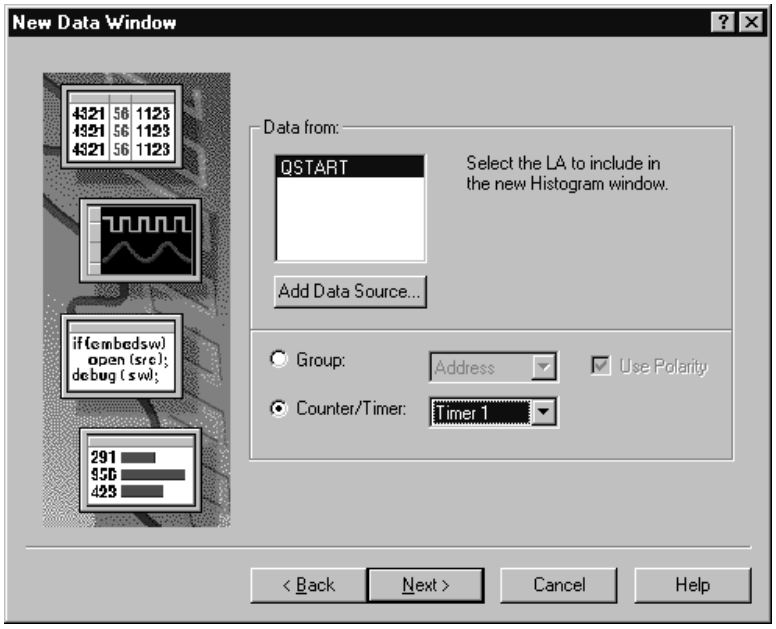


Figure 4–48: Setting the data source parameters

5. Enter a name for the new data window in the dialog box (“Int 3 ACK” is suggested). Click Finish.
6. Open the property sheet in the new Int 3 ACK Histogram window (top-left button on the toolbar).
7. Click on the Histogram Window tab and change the Data Font Size to 10.
8. Click on the Ranges tab and set each field to match Figure 4–49. Click OK.

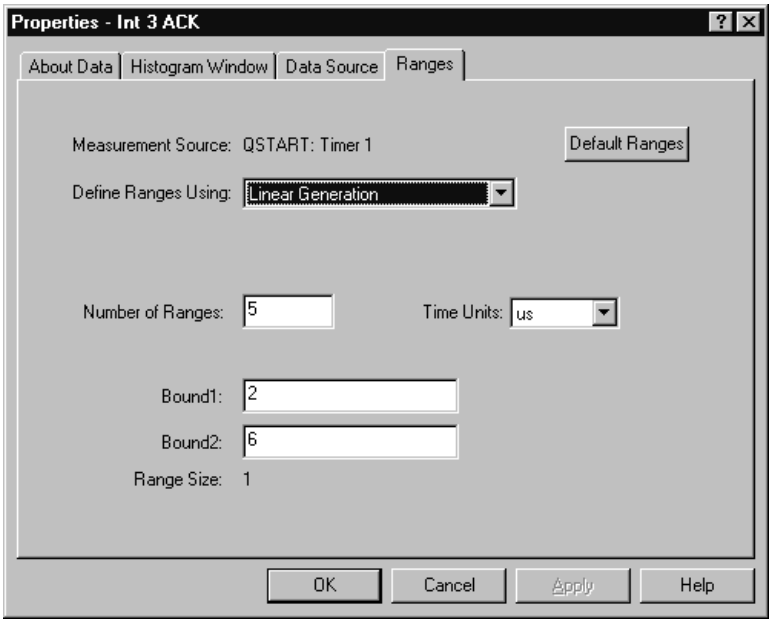


Figure 4–49: Setting the Range parameters

9. Adjust the column size in the new Int 3 ACK Histogram window to match Figure 4–50 and adjust the magnitude zoom of the window by clicking on the 100% column and select 60%.

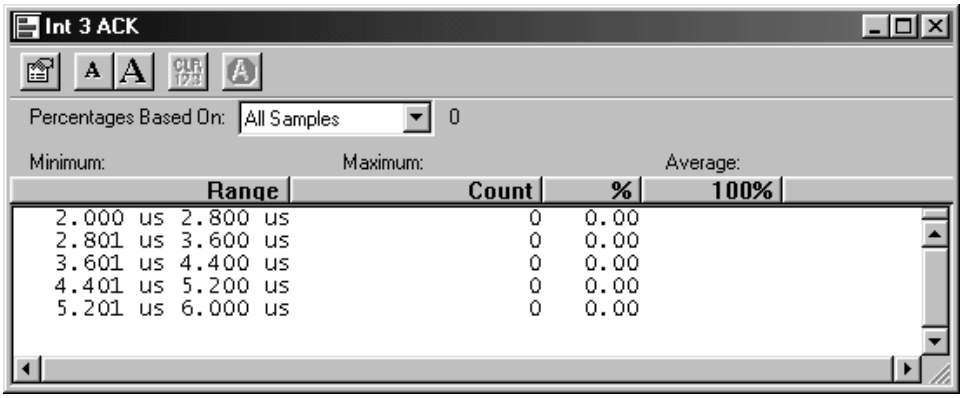


Figure 4–50: The Int 3 ACK Histogram window

Run the QuickStart Program and View the Acquired Data

1. Turn off the power to the training board, then turn it back on.
   The power switch is located at the top-left corner of the board, near the power jack.
2. Open the INT 3 ACK and ISR Routine Histogram windows.
3. Click the Run button on the logic analyzer to begin acquiring data.
4. When the Run button changes to Stop, push the F1 button on the training board. Note the updated bars in the Histogram windows.
5. Continue pushing the F1 button several more times to acquire a statistical sampling of the interrupt latency time for the button.

Figure 4–51 shows the accumulated results for Timer 1 after 7 acquisitions (pressing the F1 button 7 times). Timer 1 measured the time from the assertion of the interrupt (when you pressed the F1 button) to the time that the microprocessor read the resultant interrupt vector. The INT 3 ACK Histogram window reveals that most of interrupt acknowledge response time occurs within an acceptable time limit (in the case of the training board, 2.8 – 3.6 μs). But, occasionally, it takes almost twice as long (5.2 – 6.9 μs) to acknowledge the interrupt.

NOTE. *Due to variations in training boards, your acquisition data may be different than the data shown in this example.*

Figure 4–51 highlights some Histogram window features.

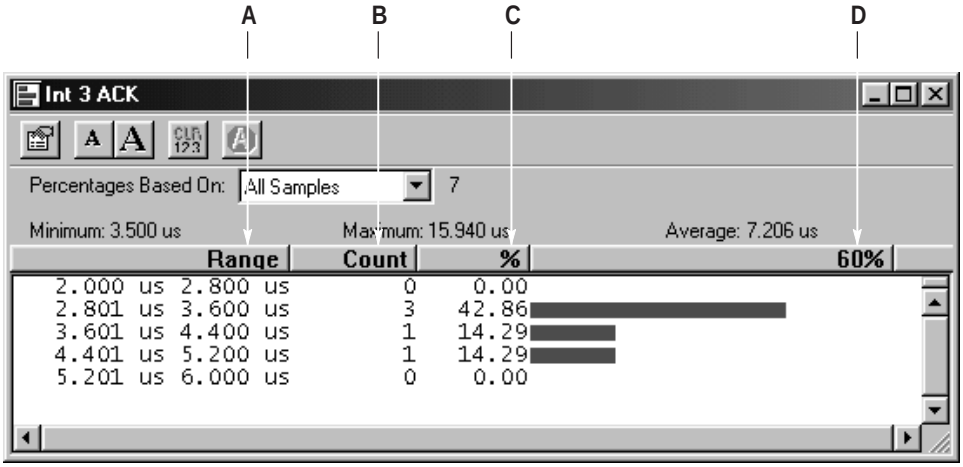


Figure 4–51: Accumulated results for Timer 1 after 7 acquisitions

A. The Range column shows the range bounds as defined in the Range page of the Histogram window.

B. The count column shows the number of hits that fall within the particular time range.

C. The % column shows the count as defined above expressed as a percentage of total number of acquisitions.

D. The magnitude zoom feature magnifies the display. This example shows the Histogram bars at 60% zoom.

Figure 4–52 shows the results of Timer 2 after 7 acquisitions (pushing the F1 button 7 times). Timer 2 measured the time duration of the Interrupt Service Routine (ISR) for the F1 button. Therefore, the sum of the interrupt latency times and execution time of the ISR for responding to the F1 button is the sum of Timer 1 and 2.

Note that the training board delivered consistent response by processing the ISR within an acceptable time limit (between 5.33 and 6.9 μs). However, to verify that your software meets all time constraints of your embedded system, use the Event Measurement capability of the logic analyzer to statistically characterize and optimize system response under real-time stress.

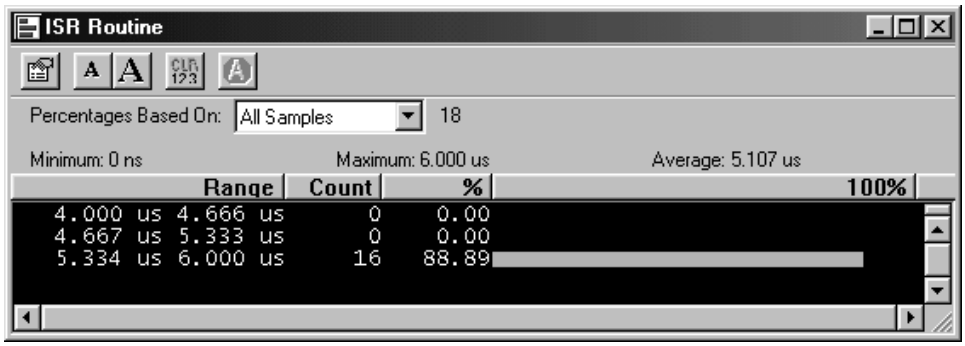


Figure 4–52: Results of Timer 2 after 7 acquisitions

Figure 4–53 is an integrated view of the Listing and Histogram windows showing the interrupt acknowledge cycles and ISR execution in conjunction with accumulated acquisitions.

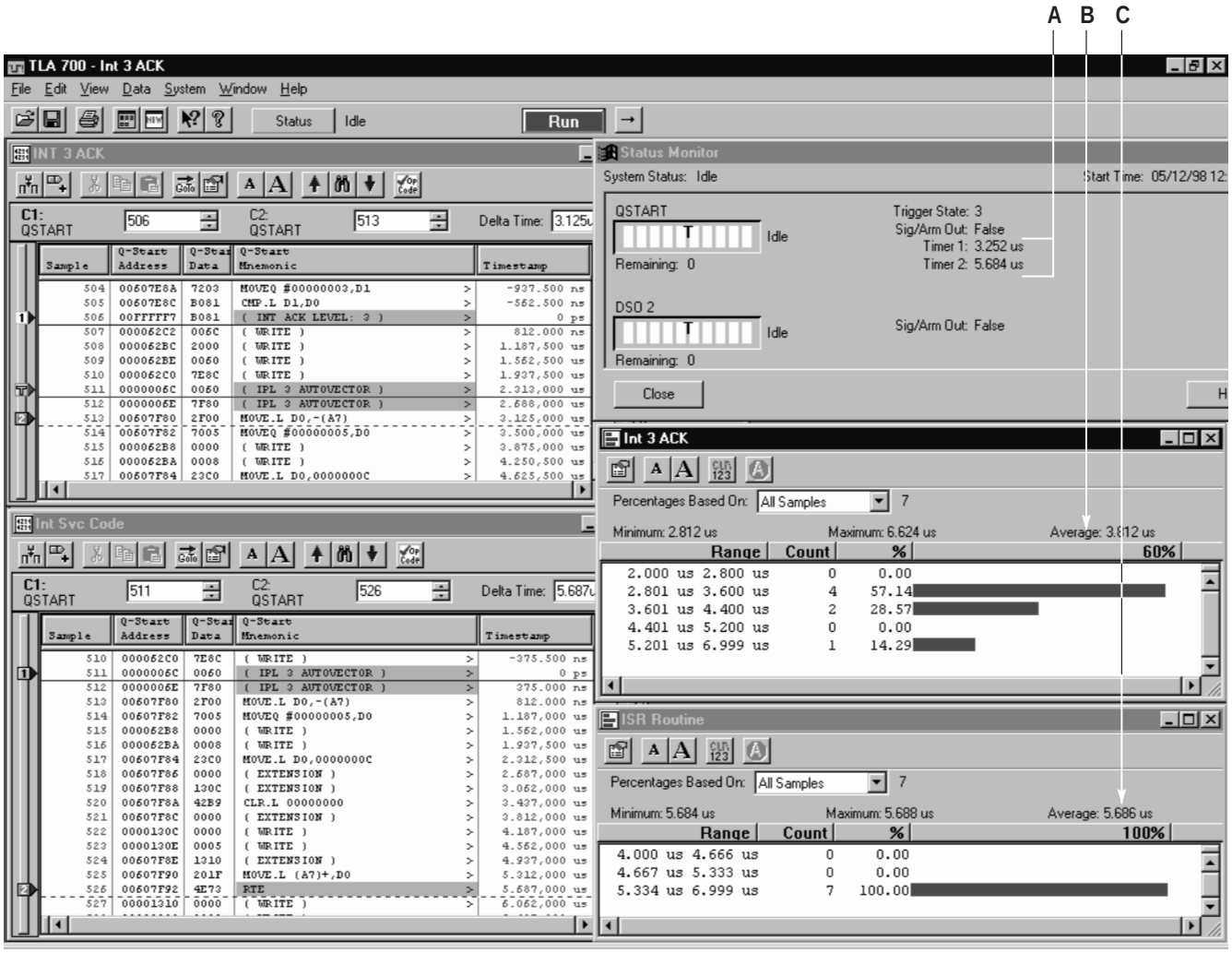


Figure 4–53: Integrated view of the Listing and Histogram windows

A. Timer 1 and 2 values for the last acquisition (the 7th time the F1 button was pressed). Note that this is consistent with the time stamp displayed in the two Listing windows.

B. The average time it took to acknowledge the interrupt (F1 button pressed) and beginning of the interrupt service routine. This is the statistical average over 7 acquisitions.

C. The time it took to execute the ISR after 7 acquisitions was 5.686 µs.

View the Trigger Program

To understand how the logic analyzer was set up to capture the data in the previous exercise, open the Trigger window (see Figure 4–54). This is exactly the same trigger setup used in Exercise 2 from the Microprocessor section, on page 3–16. Please refer to that exercise for further explanation.

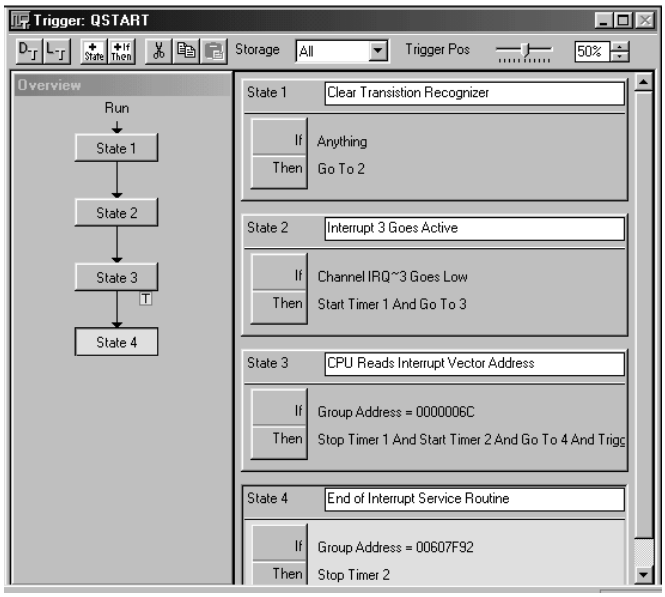


Figure 4–54: Trigger window

Pattern Generator Exercises

Pattern Generator Exercises Setup

The following exercises demonstrate the pattern generation capabilities of the Tektronix Logic Analyzer. These exercises have similar setup requirements as those used in the Microprocessor section. Therefore, it is recommended that you read *Microprocessor Exercise Setup*, beginning on page 3–1, before starting the exercises in this section.

NOTE. *These exercises can only be completed with a Tektronix Logic Analyzer with a pattern generator module.*

NOTE. *Each exercise contains two saved setup files. One file contains only setup information and requires you to capture live acquisition data to complete the exercises. The other file contains setup information and saved data. Use the saved data files to complete nearly every exercise without the need for acquiring live data.*

You can also use TLAVu to complete the exercises off-line without needing any acquisition hardware.

Hardware Setups

To complete the exercises in this chapter, you will need a logic analyzer with 102 or more data channels (TLA7N3, TLA7N4, TLA7P4) with three P6434 Probes (recommended) or six P6418 or P6417 Probes. You will also need a TLA7PG2 Pattern Generator module with one P6470 Pattern Generator Probe and one flying lead set.

It is recommended that pattern generator module is located in slots 1–2 and the logic analyzer module in slots 3–4.

1. Connect the logic analyzer probes to the appropriate A, D, and C connectors on the logic analyzer module.
2. Connect the P6470 Pattern Generator Probe to the A connector on the pattern generator module.
3. If you already have the logic analyzer probes connected from the previous exercise, skip the next section and continue with *Connect the Pattern Generator Probe* on page 5–3.

Connect the Logic Analyzer Probes

Refer to Figure 5–2 and connect the probes to the Microprocessor section of the TLA 7QS QuickStart training board as shown.

***NOTE**. If you use the P6434 probes you can connect the pattern generator probe to the pins on top of the training board. However, if you use the P6417 or P6418 probes, you need to connect the pattern generator probe to edge connector. For more information refer to* Connect the Pattern Generator Probe.

If you use the P6434 Probes, connect the probes to the appropriate group connectors on the training board (see Figure 5–1).

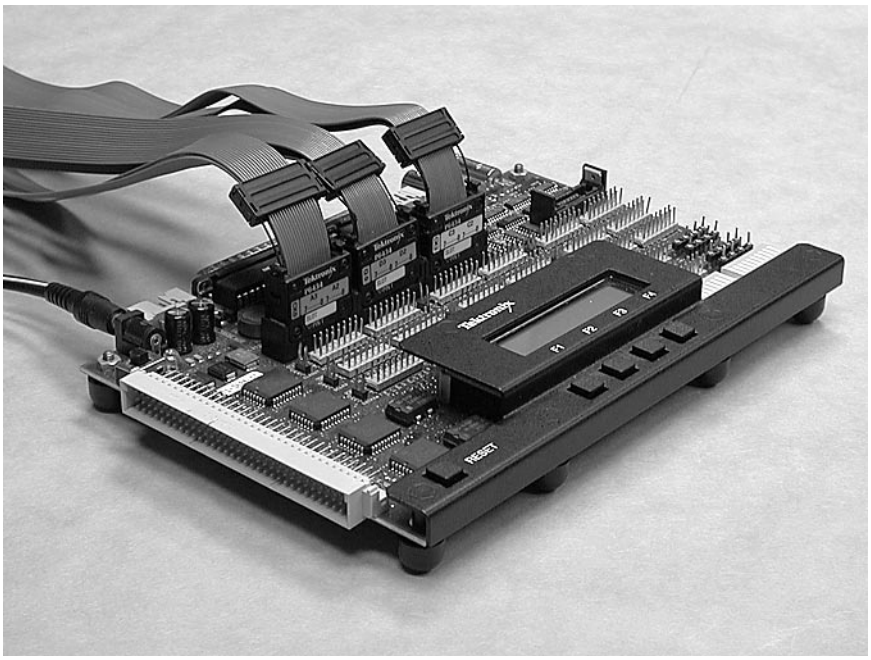


Figure 5–1: P6434 Probe connections

If you have P6417 or P6418 Probes, connect the probes to the appropriate connector on the training board (see Figure 5–2). Make sure that you connect the signal side of the probes to signal side of the square pins (ground connections are toward the rear of the training board as you face the LCD display; refer to the markings on the training board, if necessary).

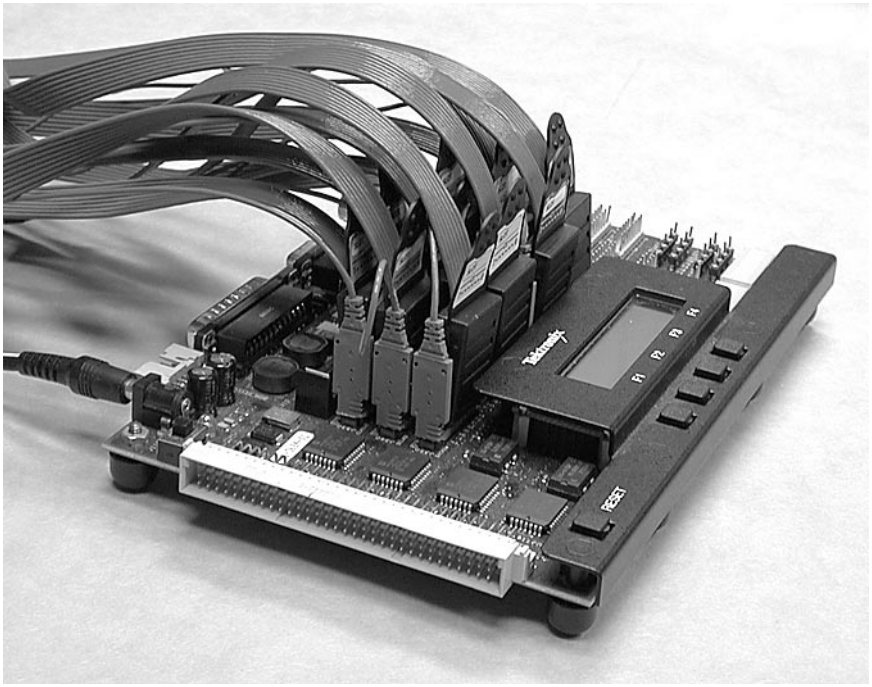


Figure 5–2: P6418 probe connections

Connect the Pattern Generator Probe

If you have the P6434 logic analyzer probes, you can connect the P6470 Pattern Generator Probe to the top of the training board or to the edge connector on the side of the training board. However, if you use either the P6417 or P6418 probes, you must connect the pattern generator probe to the edge connector. Refer to the following steps to connect the pattern generator probe.

Connecting to the Top of the Training Board

To connect the pattern generator probe to the top of the training board, refer to Figure 5–3 and do the following steps:

1. Connect the lower channel, A0(0), of the pattern generator probe to the pin labeled I3 (Interrupt Level 3, IP3~) of the C1 Group (J650) on the training board. Connect the associated gray ground lead to the adjacent ground pin.
2. Connect the next channel, A0(1) to the pin labeled I6 (Interrupt Level 6, IP6~) of the C1 Group (J650) on the training board. Connect the associated gray ground lead to the adjacent ground pin.

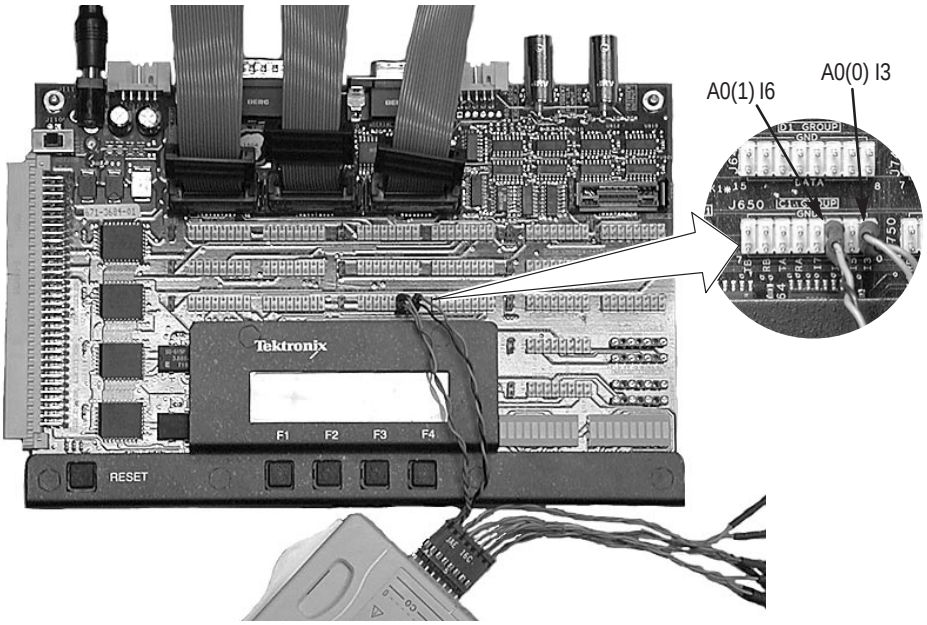


Figure 5–3: Connecting the pattern generator probe to the top of the training board

Connecting to the Edge Connector

Refer to Figure 5–4 while connecting the pattern generator probe to the training board in the following steps.

1. Connect the lower channel, A0(0), of the pattern generator probe to pin B27 on the edge connector of the training board. Connect the associated gray ground lead to pin B23 on the edge connector.
2. Connect the next channel, A0(1) to pin–B25 on the edge connector of the QuickStart training board. Connect the associated gray ground lead to pin B20 on the edge connector.

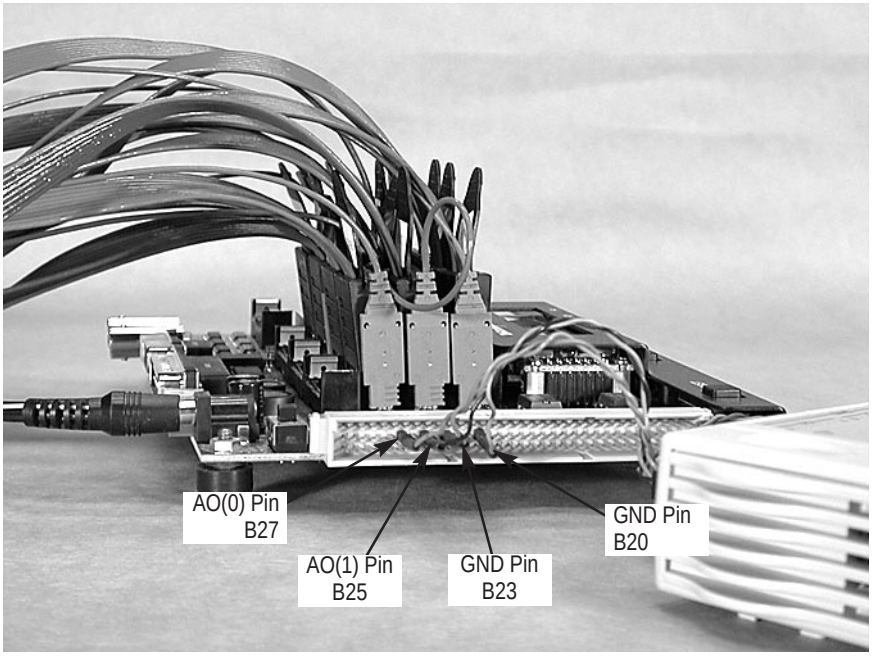


Figure 5–4: Connecting the pattern generator board to the edge connector

Pattern Generator Exercises

NOTE. *This exercise can only be completed with a Tektronix Logic Analyzer with a pattern generator module.*

Pattern Generation can be used to simulate a digital component in a system. Often times digital engineers are designing systems made up of subsystems or components. These component's design schedules don't always line up with each other. To test a system, sometimes it is necessary to simulate a component in the system. A Pattern Generator is a very useful tool for such an application.

In this exercise, you will use a pattern generator to control the program selection on the training board and run the selected program. To do this you will connect pattern generator channels to interrupt signals that control the F1–F4 buttons on the training board.

In the following exercises, the pattern generator module and the logic analyzer module interact to synchronize the stimulus patterns with the execution of the microprocessor on the training board.

Load the Setups

1. Power off the training board if it is currently turned on.
2. Load the 1–Program Selection and Run_Pattern Generator.tpg system setup from the C:\Program Files\TLA 700\Quick Start\Pattern Generator folder. Your display should look similar to Figure 5–5.

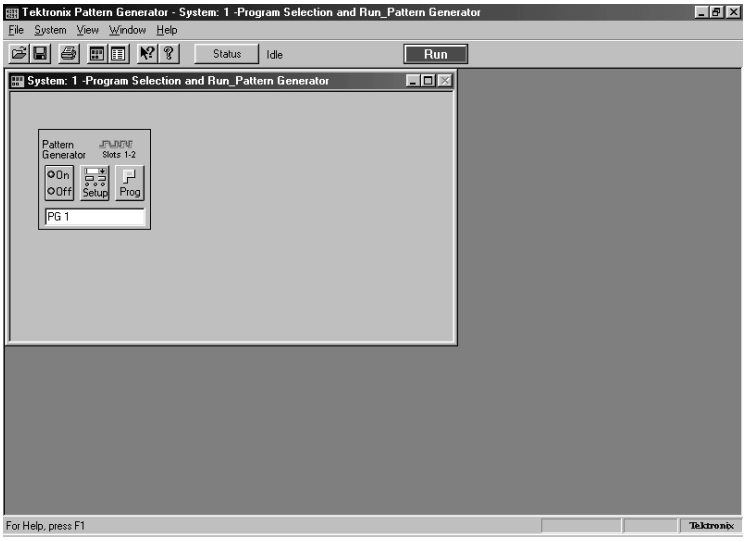


Figure 5–5: Restored system for the pattern generator exercises

3. Load the 1–Program Selection and Run_Logic Analyzer.tla system setup from the C:\Program Files\TLA 700\Quick Start\Pattern Generator folder. Your display should look similar to Figure 5–6 on page 5–9.

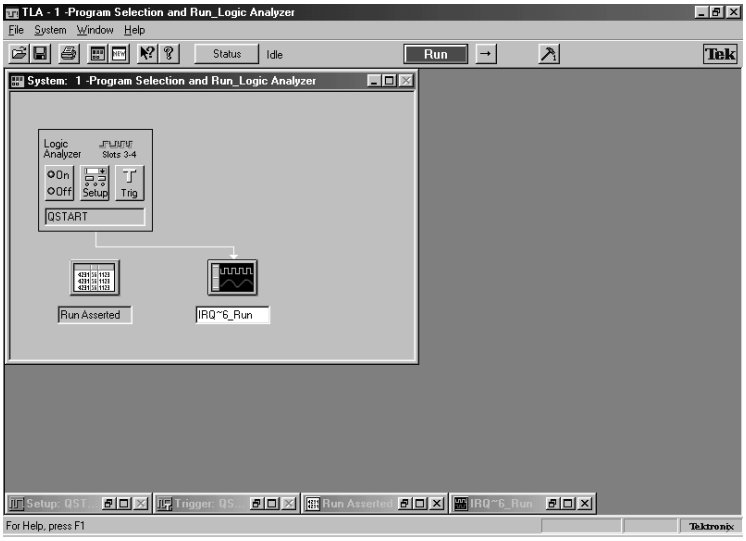


Figure 5–6: Restored system for the logic analyzer

4. Power on the training board.
5. In the Tektronix Logic Analyzer application, press the Run button.
6. Switch to the pattern generator application window. The TLA Application software will start the pattern generator software automatically.
7. When the Run button changes to Stop and turns Red, press the reset button on the training board. Your display should look similar to Figure 5–7.

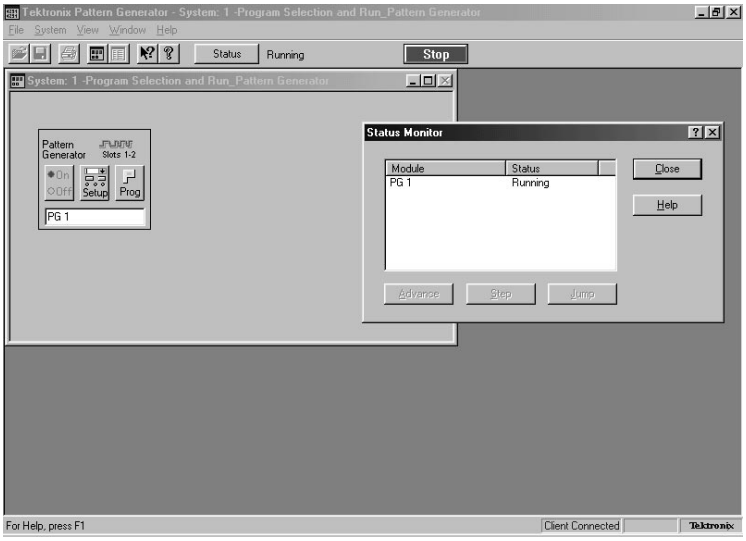


Figure 5–7: System window for the pattern generator

8. Observe that the training board performs thorough self-check software at RESET and then displays test choices on the liquid crystal display on the training board.

9. Observe that the pattern generator steps through the test selections on the training board until it reaches the Lites program. The Lites program then begins.

10. When the pattern generator Stop button changes to Run and turns Green, the program has ended and you can switch to the logic analyzer application window to view the captured data.

View the Resultant Data

1. Open the Waveform window labeled IRQ~6_Run.

   The window should contain the waveform information for the interrupt lines that the pattern generator is connected to. Your display should look similar to Figure 5–8.

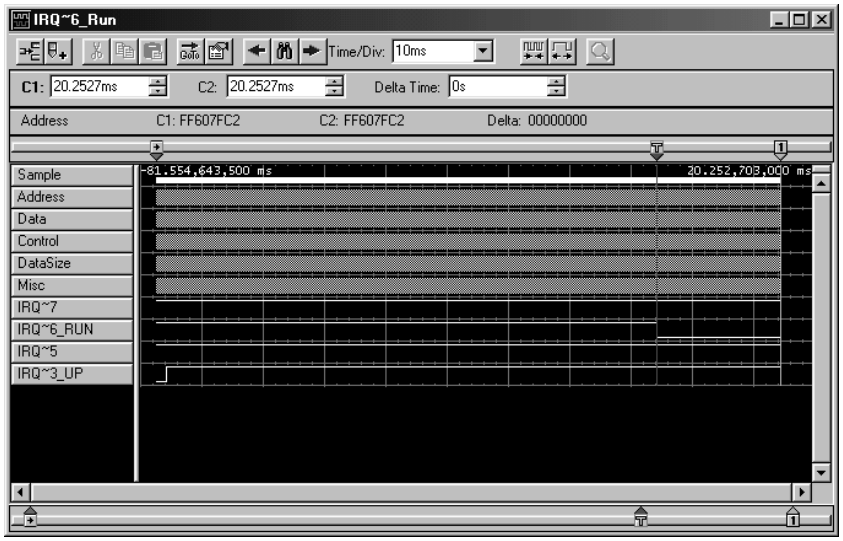


Figure 5–8: Waveform display showing IRQ~6_Run assertion

   The run command on the training board is asserted when the signal IRQ~6_RUN goes low.

2. Open the listing window labeled Run Asserted.

   The Listing window shows that the logic analyzer triggered on a condition when the IRQ~6_RUN signal went low. The code that was running on the training board before the IRQ~_Run signal went low may be different than the code shown in Figure 5–9. However, starting at the trigger point, the display should look similar to Figure 5–9 Look in the column IRQ~7–6–5–3 for the result.

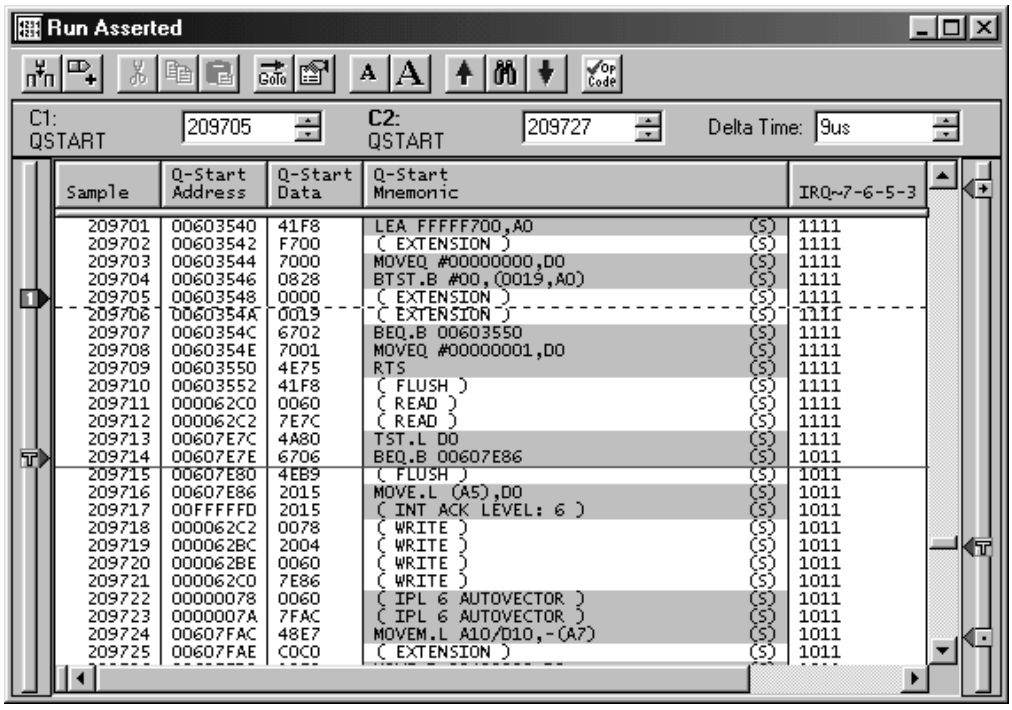


Figure 5–9: Listing window showing the Run assertion

3. Switch to the pattern generator application window.
4. Close the Status Monitor window.

View the Pattern Generator Setup

1. From the System window of the Pattern Generator Application select the Program icon labeled Prog. Your display should look similar to Figure 5–10.

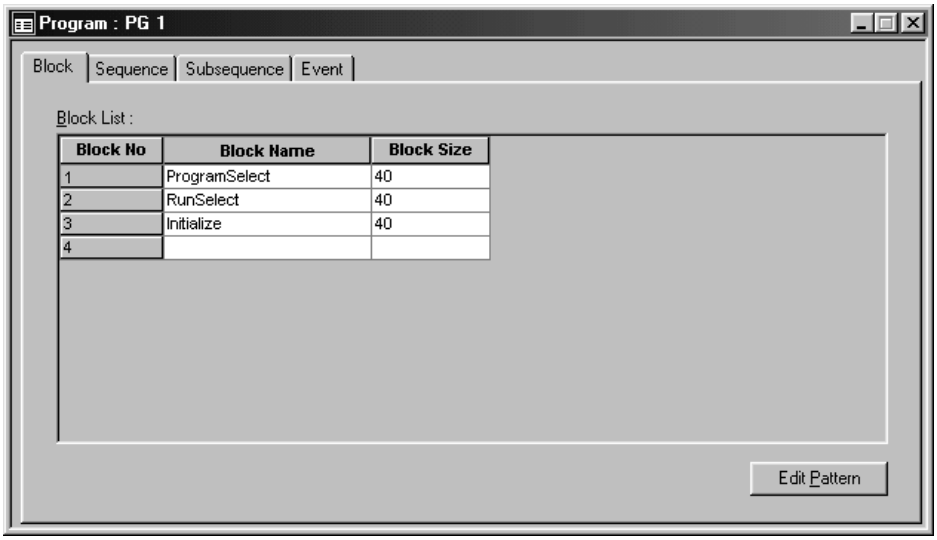


Figure 5–10: Program dialog box

The program box indicates that there are three different blocks of data each 40 bits long.

2. Select the Sequence tab in the Program window. Your display should look similar to Figure 5–11 on page 5–14.

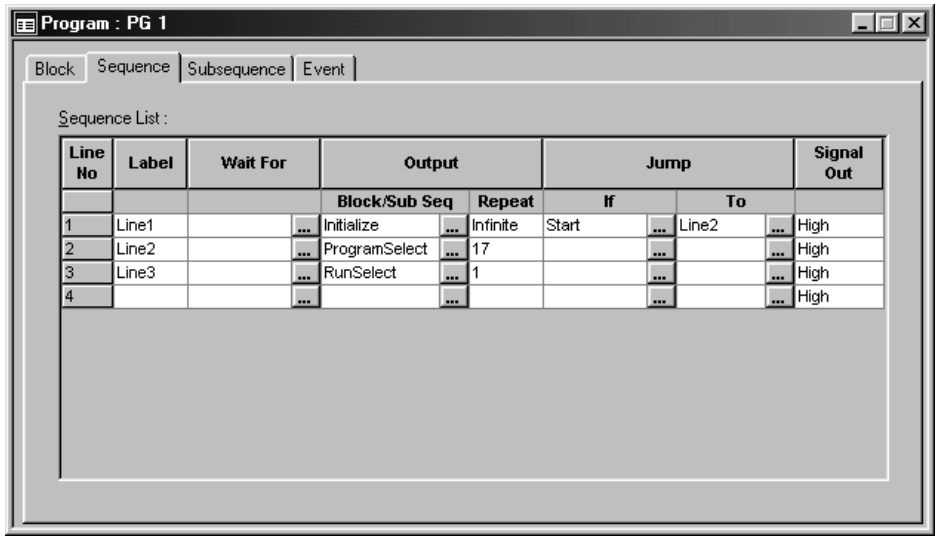


Figure 5–11: Sequence showing program selection and run

The first line in the sequence, Initialize, initializes the pattern generator and tells it to wait to run the program.

The next line shows that the ProgramSelect block runs 17 times. The Program-Select block selects the correct program on the training board; it runs when the pattern generator receives a signal from the logic analyzer indicating that the training board is out of RESET.

The last line shows that the RunSelect block runs one time. This line selects Run on the training board.

You can use sequences to generate complex programs to simulate a variety of different digital components.

View the Logic Analyzer Setup

From the System window of the Logic Analyzer Application select the Trigger icon. Follow the trigger program through the following logical steps:

1. The logic analyzer waits for the training board to reset.
2. The logic analyzer looks for the execution of a specific part of the software that indicates the power-up testing has completed and that the software is ready for inputs. When the software execution is found, system signal three is asserted to tell the pattern generator to continue to the next block.
3. The specific order of the IRQ3 and IRQ6 signals is detected and the logic analyzer triggers and fills memory.

This exercise shows how the pattern generator and logic analyzer modules can work together to accomplish functional tests. The pattern generator provides the appropriate stimuli and the logic analyzer synchronizes the pattern generator data with the software execution.

Going Further

Now that you have completed the basic exercises in this book, you are encouraged to experiment with different setups. For specific information on the capabilities of the Tektronix Logic Analyzer, you are encouraged to use the online help.

Appendices

Appendix A: How to Create Setups

This appendix shows how to create the setup for *General Purpose Exercise 1: Triggering on a Glitch*. Use the information in this appendix to create similar setups for your own applications. The Embedded Software Debug Exercises contain details for creating setups for use with debugging software.

The information in this appendix assumes that you have properly connected the probes to the training board.

If you are unsure how to complete the steps in this appendix, use the online help for the Tektronix Logic Analyzer application.

Load the Default Setup

Before creating a new setup it is a good idea to start from a known reference point. The easiest way to start from a known reference point is to start from the default setup. You can start from either the power-on setups or from the default system setup (under the File menu).

Select Default System from the File menu.

Define the Setup Window

After you have set the logic analyzer to the default setup, you need to define which channel groups and probe channels you will be using. Unless you intend to use default channel groups, you should delete any unused channel groups and define the ones you will need. This example uses the channel 1, channel 0, and CLK 3 of the C2 logic analyzer probe.

Click on the Setup button on the logic analyzer Icon to open the Setup window.

Delete Unused Groups

Perform the following steps to delete unused channel groups:

1. Click on the channel group that you want to delete under the Group Name column. For this example, click on the Address group.
2. Select Delete Group from the Edit menu.
3. Confirm your actions by clicking OK in the dialog box.
4. Delete any remaining groups.

Define New Channel Groups

Perform the following steps to define new channel groups:

1. In the lower half of the Setup window, select the empty probe C2, channel 1 field and enter the name FF-Q.

   You will assign channel 1 to the Q output of the flip-flop. You can rename any of the probe or clock channels to help display information. This makes it easier to relate individual channel names to the signals on the system under test.

2. Select the empty field for channel 0 and enter the name FF-D.

   This assigns channel 0 to the D input of the flip-flop.

3. Select the empty clock field and enter FF-CLK.

   This assigns clock channel CK3 to the clock signal of the flip-flop.

   Now that you have identified the individual probe channels, you need to assign the channels to a channel group.

4. Click on the top Group Name field and enter the name FF CLOCK. Now click on the box to the left of the FF-CLK field. An X appears indicating that the FF-CLK channel is assigned to the FF CLOCK group.

5. Click on the next Group Name field and enter the name FF DATA IN. Now assign the FF-D channel to the FF DATA IN group.

6. Click on the next Group Name field and enter the name FF Q-OUT. Now assign the FF-Q channel to the FF Q-OUT group.

Complete the Setup WIndow

To complete the changes to the Setup window, set the following fields to the values shown below:

Table A–1: Setup window field values

Field	Value
Clocking	Internal, 10 ns
Memory Depth	1024
Acquire	Glitches

The completed Setup window should look similar to Figure A–1.

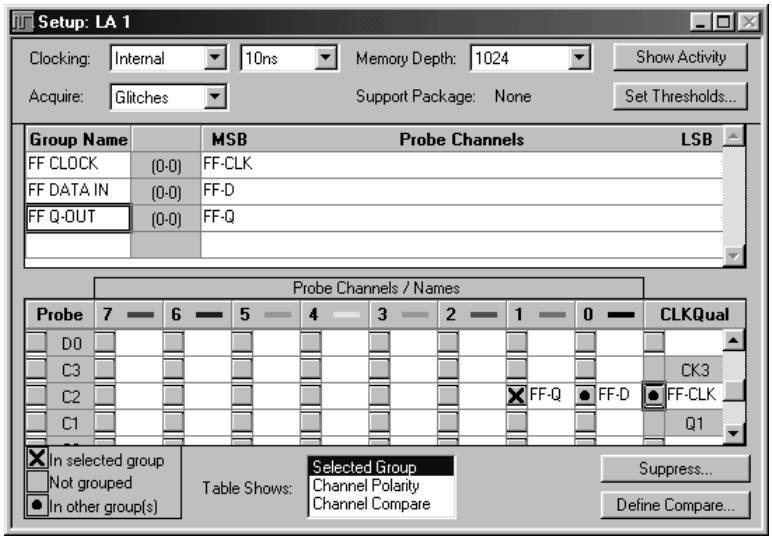


Figure A–1: Setup window for Exercise 1

Close the Setup window.

Define the Trigger Window

After you have defined the channel groups, you can define the trigger parameters. Click on the Trig button on the logic analyzer module icon to open the Trigger window.

1. Set the Storage field to All; the logic analyzer will store all data samples.
2. Set the Trigger Position to 10%.

The next few steps show how to set up the logic analyzer to trigger on a glitch on the FF-Q channel group.

Define Trigger State 1

Trigger State 1 looks for a glitch in the specified channel group. To set up Trigger State 1, complete the following steps:

1. Enter the name Trigger on Glitches in the Trigger State 1 field.

   You can assign meaningful names or comments to each trigger state to help you identify at a glance what each trigger state is doing.

2. Click on the If-Then button to open the State 1 Clause Definition dialog box.

3. Change the top left field from Anything to Glitch.

   This tells the logic analyzer to look for a glitch event in the specified channel group. A new button appears at the top right of the window labeled Define Glitches.

4. Click on the Define Glitches button to open the Glitch Detection dialog box.

   The Glitch Detection dialog box contains a list of all defined channel groups. You can define the channel groups where you want to capture glitches. By default, all channels are set to look for glitches.

5. Click on the FF CLOCK and the FF DATA IN groups to disable the glitch detection for those groups; the check mark disappears when the channels have been disabled.

6. Leave the check mark on the FF Q-OUT channel group. The Glitch Detection dialog box should look similar to Figure A–2.

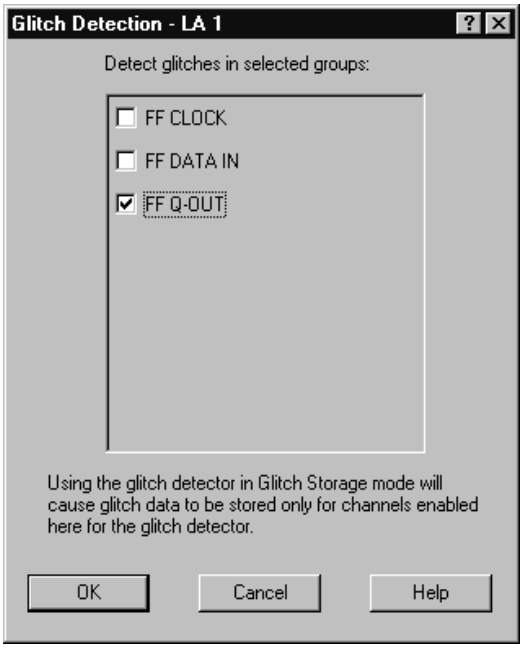


Figure A–2: Glitch Detection dialog box

7. Click OK to save your changes and to exit the dialog box.

8. Change the Action field in the lower half of the window from Trigger to Trigger All Modules.

   The logic analyzer module will trigger all remaining modules when a glitch is detected.

9. Click OK to close the State 1 Definition dialog box. The Trigger window should look similar to Figure A–3.

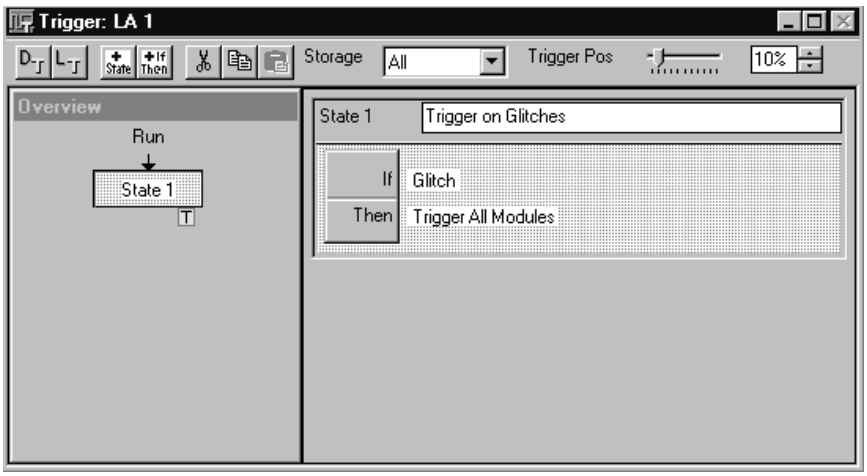


Figure A–3: Trigger window setups

10. Click the minimize button to close the Trigger window.

You have now completed the setups for the Trigger window. If necessary, you could create a more complex Trigger window by adding several states with specific test conditions. However, for the purposes of these setups, you need only one trigger state.

The next steps show how to set up the data window.

Define the Waveform Window

After you define the Setup and Trigger windows, you need to determine how you will display the acquired data. You can display acquired data in a list (Listing window), or you can display the data as a waveform (Waveform window). Because you are interested in a single data channel, you will set up a new Waveform data window.

Delete the Unused Data Window (Optional)

If your logic analyzer has any unused windows, you can delete those you no longer need. You can delete an unwanted window by opening the window and clicking on the close button in that window's titlebar. You will be asked to confirm your choice.

Create a New Waveform Window

Complete the following steps to create a new Waveform window:

1. Click on New Data Window in the Window menu.

   A New Data Window dialog box will display.

2. Change the Window Type field to Waveform.

3. Set up the dialog box so that the logic analyzer module is the source module for the data window by selecting the logic analyzer module in the second field from the top of the menu.

4. If desired, enter a name for the new window in the Window Name field.

5. Click OK to close the dialog box.

   A new Waveform window displays. The next step is to define how you want to view the waveforms.

Define the Waveform Window

Complete the following steps to define how you want to display the data in the Waveform window:

1. Open the Waveform window.

2. Double-click on the FF-Q waveform to display the Waveform property sheet.

   A waveform property page exists for each waveform defined in the Setup window.

3. If desired, select a color for the waveform.

4. Click OK to close the Waveform property sheet.

You are now ready to acquire and display the data in Exercise 1.

Appendix B: Training Board Connections