TEXAS INSTRUMENTS

XDS522A Emulation System

User's Guide

1996

Digital Signal Processing Solutions

Printed in U.S.A., August 1996
D412009-9761 revision B

SPRU169B

User's Guide

XDS522A Emulation System

Release 0.96

1996

XDS522A Emulation System User's Guide

August 1996
SPRU169

Printed on Recycled Paper

IMPORTANT NOTICE

Texas Instruments (TI) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

TI warrants performance of its semiconductor products and related software to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Certain applications using semiconductor products may involve potential risks of death, personal injury, or severe property or environmental damage ("Critical Applications").

TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS.

Inclusion of TI products in such applications is understood to be fully at the risk of the customer. Use of TI products in such applications requires the written approval of an appropriate TI officer. Questions concerning potential risk applications should be directed to TI through a local SC sales office.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards should be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used.

Preface

Read This First

About This Book

This book describes the XDS522A™ emulation system, which is an analysis and debugging tool. The system adds breakpoint, tracing, and timing (BTT) functionality to the Texas Instruments (TI™) C source debugger. These functions allow you to monitor the CPU, to benchmark performance, to gather precise timing information, to generate more compact and efficient code, and to improve debugging efficiency.

Before you start using this book, you should install the following tools:

Tool	Refer to…
XDS510™ or XDS510PP™ and XDS511™	*XDS51x Emulator Installation Guide*
XDS522A (including the BTT software)	*XDS522/XDS522A Emulation System Installation Guide*
TMS320C2xx C source debugger	*TMS320C2xx PC Emulator Installation Guide*

How to Use This Book

This book is divided into four parts:

- **Part I: Overview** gives you a broad look at the XDS522A emulation system and describes the basic steps you need to follow when you are testing or debugging your code.
  - Chapter 1 summarizes of the XDS522A emulation system's features and describes its components and architecture.
  - Chapter 2 describes the XDS522A interface and how you can customize it to meet your needs.
  - Chapter 3 provides a quick glance of all of the steps that you need to follow when you are using the system to test and debug your code.

- **Part II: Tutorial** provides you with a step-by-step, hands-on tutorial of the system. The tutorial is divided into several chapters. Each chapter builds on the previous chapter(s). To learn how to use the XDS522A emulation system correctly, you should start at the beginning of Part II and continue working through the tutorial until you reach the end of Part II, rather than skipping lessons or doing them out of order.
- **Part III: User Reference** tells you how to configure the XDS522A emulation system.
  - Chapter 12 describes how to define events, which control most of the activity in the XDS522A.
  - Chapters 13 and 14 describe how to collect samples and how you can use some of the system's features to isolate specific samples from the samples you have collected.
  - Chapters 15, 16, and 17 explain how to set up the XDS522A to perform an action when an event occurs. Chapter 15 describes how you can use the counters, Chapter 16 describes how you can use the sequencer, and Chapter 17 describes how you generate a trigger pulse or a hardware breakpoint.
- **Part IV: Appendixes** provides supplementary information and includes a summary of commands, a troubleshooting section, a glossary, and an index.

The way you should use this book depends on your experience with similar products. As with any book, it would be best for you to begin on page 1 and read to the end. Because most people don't read technical manuals from cover to cover, here are some suggestions for choosing what to read.

- If you have used the XDS522A emulation system before, you may want to:
  - Read the introductory material in Part I.
  - Use the tutorial in Part II to learn about any unfamiliar features. Make sure your system is set up as described in Chapter 4.
  - Read the chapters in Part III that are of most interest to you.
  - Use the appendixes in Part IV as necessary.
- If this is the first time that you have used the XDS522A emulation system, you may want to:
  - Read the introductory material in Part I.
  - Complete the tutorial in Part II.
  - Refer to the chapters in Part III for additional information.
  - Use the appendixes in Part IV as necessary.

Notational Conventions

This document uses the following conventions:

- The TMS320C209 and TMS320C209SE devices are referred to as the ’C2xx.
- Debugger and BTT commands are not case sensitive; you can enter them in lowercase, uppercase, or a combination of both.
- Program listings, program examples, and interactive displays are shown in a `special typeface` similar to a typewriter’s. Interactive displays use a **`bold version`** of the special typeface to distinguish commands that you enter from items that the system displays (such as prompts, command output, error messages, etc.).

  Here is a sample message that you might see in the display area of the COMMAND window:

  ```
  Cannot append in binary mode
  ```

  Here is an example of a command that you might enter:

  `cfgtrace paddress,hex` ↵
- In syntax descriptions, the instruction or command is in **bold face,** and parameters are in *italics*. Portions of a syntax that are in **bold** should be entered as shown; portions of a syntax that are in *italics* describe the kind of information that you enter. Here is an example of a command syntax:

  bload *filename*

  bload is the command. This command has one parameter, indicated by *filename*.
- Square brackets ([and]) identify an optional parameter. If you use an optional parameter, you specify the information within the brackets; you don’t enter the brackets themselves. Here’s an example of an instruction that has an optional parameter:

  alias [*alias name* [**, ”***command string***”**]]

  The ALIAS command has two optional parameters.
- Braces ({ and }) indicate a list. The symbol **|** (read as *or*) separates items within the list. Here’s an example of a list:

  TraceFilter { **on** | **off** }

  This provides two choices: **TraceFilter on** or **TraceFilter off**.

  Unless the list is enclosed in square brackets, you must choose one item from the list.

- Part II contains symbols that indicate where to perform an action, when a task is important, and when a task is optional:

When you see this symbol…	Perform the next set of instructions…
BTT	In the BTT application window
Debug	In the debugger application window
EVENTS Box	In the EVENTS box of the BTT setup window
SEQUENCER Box	In the SEQUENCER box of the BTT setup window
COUNTER 1 Box	In the COUNTER 1 box of the BTT setup window
COUNTER 2 Box	In the COUNTER 2 box of the BTT setup window
TRACE Box	In the TRACE box of the BTT setup window
ACTION Box	In the ACTION box of the BTT setup window
Important!	To ensure that your system works correctly
Try This:	If you are interested in doing optional tasks that can help you learn more about the XDS522A emulation system

Information About Cautions

This book contains cautions. The information in a caution is provided for your protection. Please read each caution carefully.

This is an example of a caution statement.

A caution statement describes a situation that could potentially damage your software or equipment.

Related Documentation From Texas Instruments

The following documentation is packaged with the XDS522A emulation system:

XDS522/XDS522A Emulation System Installation Guide (literature number SPRU171) describes the installation of the emulation system. Instructions include how to install the hardware and software for the XDS522™ and XDS522A™.

XDS522A Emulation System Online Help (literature number SPRC002) is an online help file that provides descriptions of the BTT software user interface, menus, and dialog boxes.

The following documents provide additional information about the XDS522A emulation system. To obtain a copy of any of these documents, call the Texas Instruments Literature Response Center at (800) 477–8924. When ordering, please identify the book by its title and literature number.

TMS320C2xx C Source Debugger User's Guide (literature number SPRU151) tells you how to invoke the 'C2xx emulator and simulator versions of the C source debugger interface. This book discusses various aspects of the debugger interface, including window management, command entry, code execution, data management, and breakpoints. It also includes a tutorial that introduces basic debugger functionality.

TMS320C2xx PC Emulator Installation Guide (literature number SPRU152) tells you how to install the PC™ version of the 'C2xx emulator and C source debugger interface.

TMS320C2xx User's Guide (literature number SPRU127) discusses the hardware aspects of the 'C2xx fixed-point digital signal processors. It describes pin assignments, architecture, instruction set, and software and hardware applications. It also includes electrical specifications and package mechanical data for all 'C2xx devices. The book features a section comparing instructions from 'C2x to 'C2xx.

XDS51x Emulator Installation Guide (literature number SPNU070) describes the installation of the XDS510™, XDS510PP™, and XDS510WS™ emulator controllers. The installation of the XDS511™ emulator is also described.

If You Need Assistance. . .

If you want to . . .	Contact Texas Instruments at . . .	
Visit TI online	World Wide Web:	http://www.ti.com
Receive general information or assistance	World Wide Web:	http://www.ti.com/sc/docs/pic/home.htm
	North America, South America:	(214) 644–5580
	Europe, Middle East, Africa	
	Dutch:	33–1–3070–1166
	English:	33–1–3070–1165
	French:	33–1–3070–1164
	Italian:	33–1–3070–1167
	German:	33–1–3070–1168
	Japan (Japanese or English)	
	Domestic toll-free:	0120–81–0026
	International:	81–3–3457–0972 or 81–3–3457–0976
	Korea (Korean or English):	82–2–551–2804
	Taiwan (Chinese or English):	886–2–3771450
Ask questions about Digital Signal Processor (DSP) product operation or report suspected problems	Hotline:	(713) 274–2320
	Fax:	(713) 274–2324
	Fax Europe:	+33–1–3070–1032
	Email:	4389750@mcimail.com
	World Wide Web:	http://www.ti.com/dsps
	BBS North America:	(713) 274–2323 8–N–1
	BBS Europe:	+44–2–3422–3248
	320 BBS Online:	ftp.ti.com:/mirrors/tms320bbs (192.94.94.33)
Request tool updates	Software:	(214) 638–0333
	Software fax:	(214) 638–7742
	Hardware:	(713) 274–2285
Order Texas Instruments documentation (see Note 1)		(800) 477–8924
Make suggestions about or report errors in documentation (see Note 2)	Email:	comments@books.sc.ti.com
	Mail:	Texas Instruments Incorporated Technical Publications Manager, MS 702 P.O. Box 1443 Houston, Texas 77251–1443

Notes: 1) The literature number for the book is required; see the lower-right corner on the back cover.

2) Please mention the full title of the book, the literature number from the lower-right corner of the back cover, and the publication date from the spine or front cover.

FCC Warning

This equipment is intended for use in a laboratory test environment only. It generates, uses, and can radiate radio frequency energy and has not been tested for compliance with the limits of computing devices pursuant to subpart J of part 15 of FCC rules, which are designed to provide reasonable protection against radio frequency interference. Operation of this equipment in other environments may cause interference with radio communications, in which case the user at his own expense will be required to take whatever measures may be required to correct this interference.

Trademarks

Contents

Part I: Overview

Part III: User Reference

Part IV: Appendixes

Figures

Tables

Part I
Overview

Part II
Tutorial

Part III
User Reference

Part IV
Appendixes

Part I

Chapter 1

Overview of the XDS522A Emulation System

Part I

This chapter describes the features and components of the XDS522A emulation system and how the system operates in a debugging environment.

Topic **Page**

1.1 What is the XDS522A Emulation System?

The XDS522A emulation system is an analysis and debugging tool that adds breakpoint, tracing, and timing (BTT) functionality to the development system that includes the XDS510 and XDS511. These functions allow you to:

- ❏ Monitor the CPU
- ❏ Benchmark performance
- ❏ Gather precise timing information
- ❏ Generate smaller, more efficient code
- ❏ Improve debugging efficiency

User-defined events control most activity in the XDS522A. Events are defined by address or data patterns for any combination of buses, execution and memory cycle types, and the 16 color-coded external channel probes. When these events occur, the XDS522A can:

- ❏ Start, stop, or disable tracing
- ❏ Start, stop, reload, or clock the counters
- ❏ Advance or reset the sequencer
- ❏ Generate hardware breakpoints

Features

The XDS522A includes these features:

- ❏ 32K-byte real-time trace buffer
- ❏ Event recognition and sequencing
- ❏ Dual 16-bit counters
- ❏ Trace acquisition and control
- ❏ Multiple hardware breakpoints
- ❏ Hardware timestamping
- ❏ External triggers for test equipment synchronization
- ❏ Ability to load/save screen configurations and trace samples

The XDS522A monitors and records bus and cycle type activity on the target device at the full operating speed of the target device.

The ’C209 SE device adds visibility through additional pins that monitor internal buses and cycle-type signals that are not available on production devices. These additional signals reveal internal state and bus information that improves debugging efficiency.

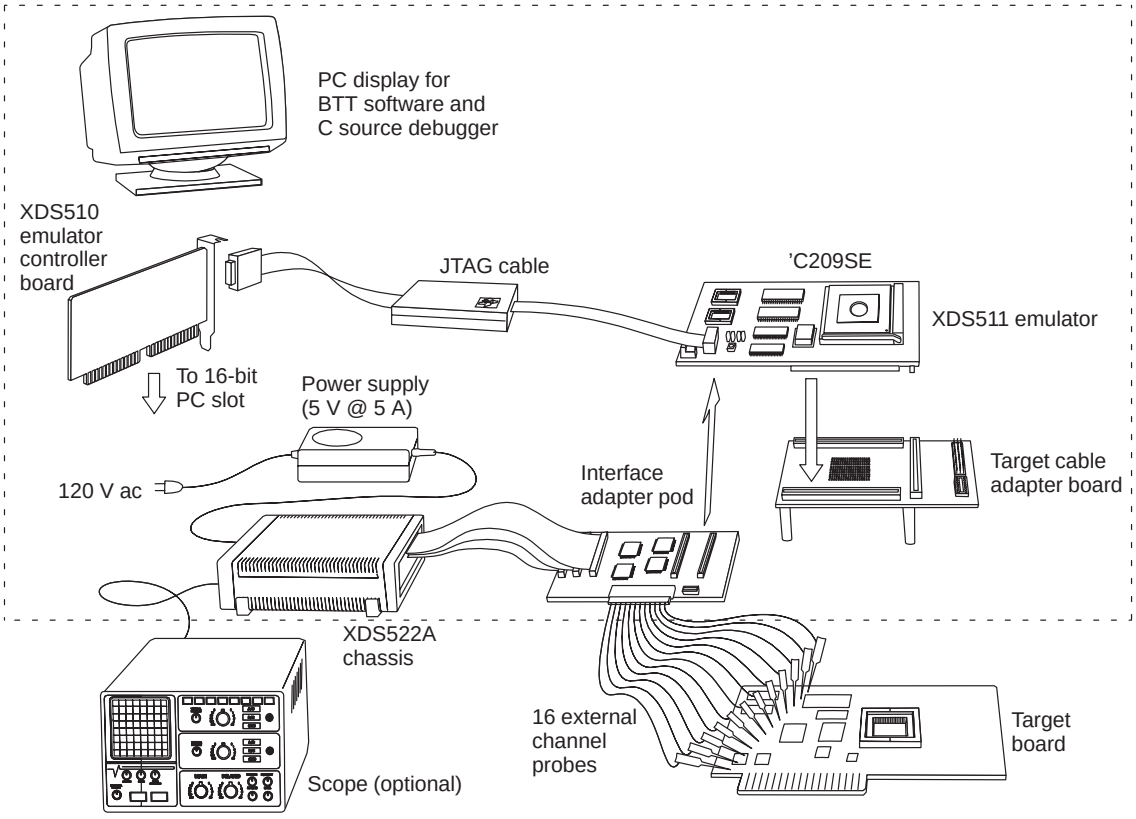
Components

The XDS522A emulation system consists of the following components:

- The PC™ display for the BTT software and C source debugger
- An XDS510 emulator controller board and a JTAG cable that connects the XDS510 to the XDS511 emulator
- The XDS511 emulator board with a ’C209 SE device and a target cable adapter board
- The XDS522A chassis
- An interface adapter pod with pipeline flattener and woven cables
- A target board of your own design
- 16 external channel probes
- A 5-V @ 5-A external power supply

Figure 1–1 shows the components of the XDS522A emulation system. The components within the dotted lines show the configuration for stand-alone mode. Stand-alone mode is the mode used in the tutorial. For more information about stand-alone mode, see Section 4.2 on page 4-4.

Figure 1–1. XDS522A Emulation System Components



You can connect your target board to the XDS511 emulator without using the target cable adapter board. To do so, attach the XDS511 to your target through the three 100 pin connectors on the bottom of the XDS511 (JP6, JP7, JP8).

1.2 Understanding the Debugging Environment

The XDS522A supports code-development and debugging activity for the ’C2xx DSP devices at speeds up to 30 MHz. The XDS522A and the target application (XDS511) operate independently of each other but share the same JTAG scan chain.

When the C source debugger is set up to use the BTT functionality, debugger actions cause the XDS522A to respond to the activity on the special emulation (SE) device. For example, if a software breakpoint occurs in the target application, the XDS522A updates the TRACE window with new information.

The typical debugging environment for a ’C2xx application includes a PC with the C source debugger and an XDS510 emulator controller that connects to a ’C2xx SE device through an XDS511 emulator.

If you are using a real-time monitor...

If you are using a real-time monitor, you must ensure that you correctly set the REALTIME switch on the interface adapter pod of the XDS522A. The REALTIME (S2) switch on the interface adapter pod affects the CPU activity that you see with the XDS522A. When the ’C2xx performs emulation actions in stop mode, the XDS522A does not collect this information.

If you use a real-time monitor, the XDS522A collects all ’C2xx information, including actions performed by the real-time monitor. To operate the ’C2xx debugger in real-time mode, you must ensure that the XDS522A S2 switch is set to On so that code between ETRAP and ERET is visible.

To operate the ’C2xx debugger in stop mode, you probably want to ensure that the XDS522A S2 switch is set to Off. If you are operating in stop mode and the S2 switch is set to On, you will collect samples when the debugger is performing emulation actions as well as normal operation information.

For more information on the REALTIME switch, see the *Step 5: Setting Dip Switches on the Interface Adapter Pod* section in the *XDS522 and XDS522A Emulation System Installation Guide*.

1.3 Understanding the TMS320C2xx Pipeline and the XDS522A Pipeline

When working with the XDS522A, you need to keep in mind that there are two distinct and independent pipelines:

- The *TMS320C2xx instruction pipeline* is the sequence of bus operations that occur during the execution of an instruction. The instruction pipeline has four independent stages: instruction fetch, instruction decode, operand fetch, and instruction execute. Figure 1–2 shows the sequence for an arbitrary instruction N in the 'C2xx instruction pipeline. For more information about the 'C2xx instruction pipeline, see the *TMS320C2xx User's Guide*.

- The *XDS522A pipeline* is the sequence that the XDS522A uses to process data and compare data with events that you define to control breakpoints, tracing, and timing. The XDS522A pipeline has three independent stages: compare events, determine action, and take action. The pipeline for the XDS522A is unrelated to the instruction pipeline of the 'C2xx device. Figure 1–3 illustrates the three clock cycles in the XDS522A pipeline.

Figure 1–2. 4-Level Instruction Pipeline of the TMS320C2xx

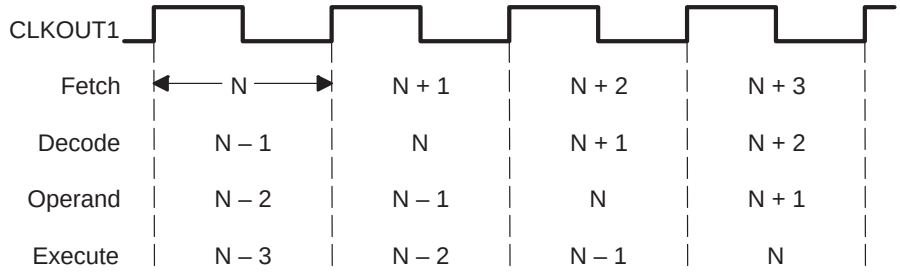
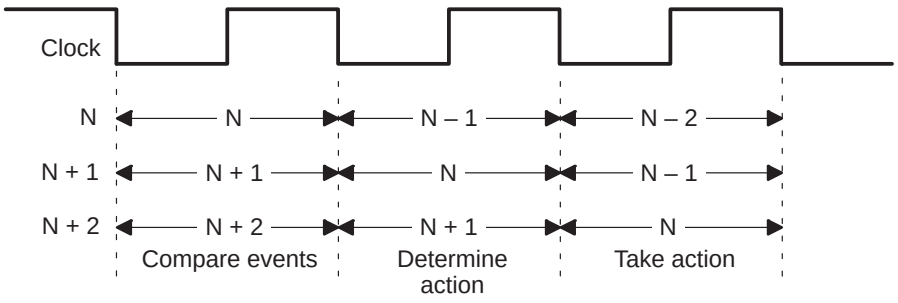


Figure 1–3. 3-Cycle Pipeline of the XDS522A



About the XDS522A pipeline

The XDS522A evaluates all processor activity simultaneously and clocks the data through the pipeline. Information remains in the pipeline for three cycles before it is discarded or written to trace memory.

The order of activity is as follows:

1) In the first clock cycle of the XDS522A pipeline, the software compares events with the incoming raw data.

2) If the comparison evaluates to true, the XDS522A latches the output in the second clock cycle.

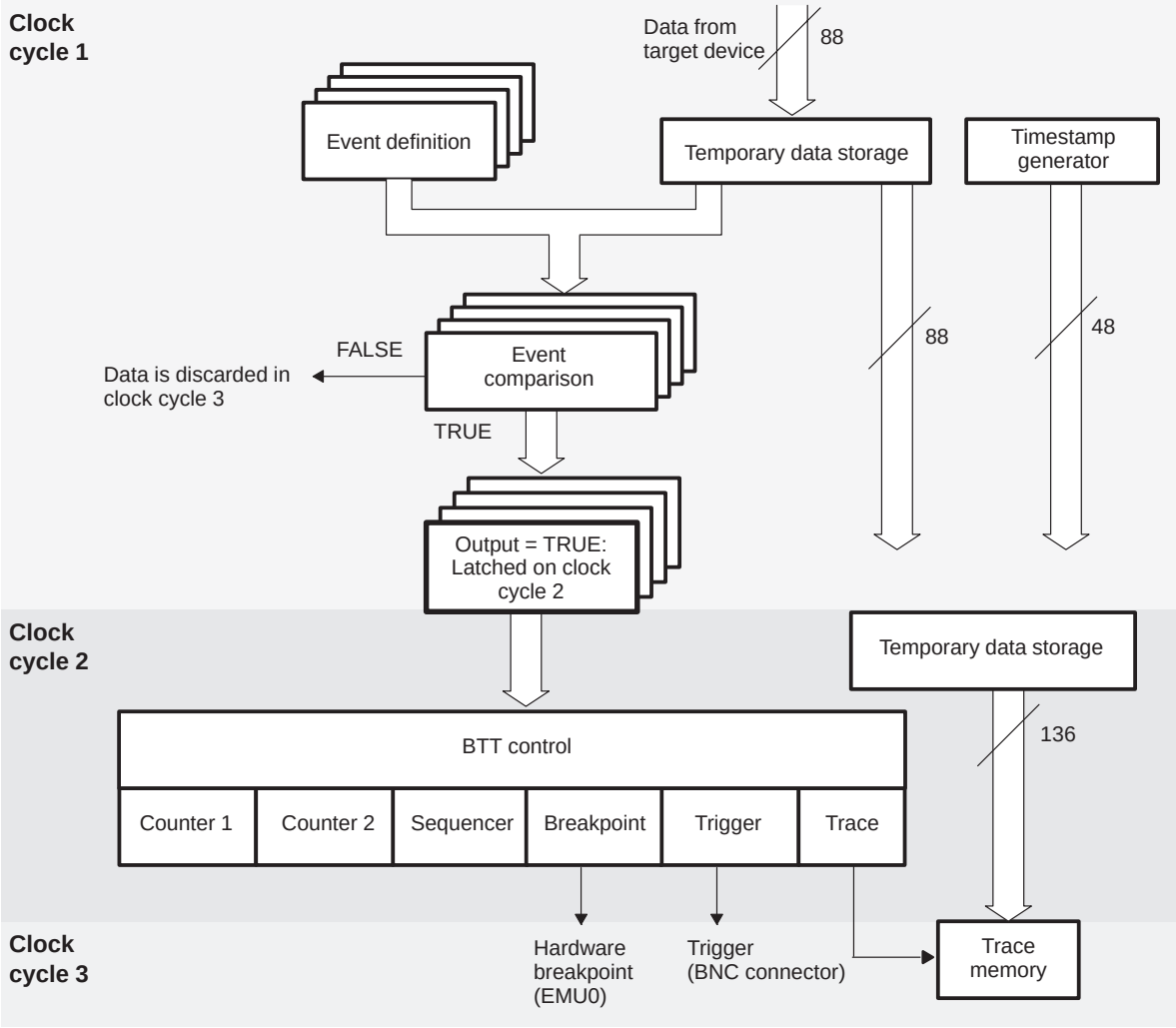
3) In the third clock cycle, your defined events are put to use in the manner you have specified:

   - ❑ To control a counter: decrement, start, stop, or reload
   - ❑ To advance the sequencer to the next level
   - ❑ To generate a hardware breakpoint
   - ❑ To generate a trigger pulse
   - ❑ To start, stop, or disable tracing

The clock that drives the pipeline is not free running; this sample clock stops when there is no valid bus activity or when the processor is in low-power mode. The XDS522A clocks out all data still in the pipeline when the sample clock stops.

Figure 1–4 shows the parallel activity with data in the pipeline.

Figure 1–4. XDS522A Pipeline Architecture



About the TMS320C2xx pipeline flattener

Figure 1–5 illustrates the stages a single instruction N passes through in the ’C2xx instruction pipeline, as well as other activity occurring at the same time for other instructions. During the cycle in which the N instruction executes, the ’C2xx fetches operands for instruction N + 1, decodes instruction N + 2, and fetches instruction N + 3.

The XDS522A contains a flattener for the ’C2xx instruction pipeline. For each execution cycle, the flattener aligns the ’C2xx pipeline stages for a single instruction. As Figure 1–6 shows, the flattener aligns the operand fetch for instruction N, the decode for instruction N, the fetch for instruction N, and the execution for instruction N.

The flattener also affects the contents of the trace buffer by filtering out unexecuted instructions from the buffer so that only executed cycles are collected. The flattener also aligns the data with the corresponding execution cycle.

Figure 1–5. Stages in the ’C2xx Instruction Pipeline

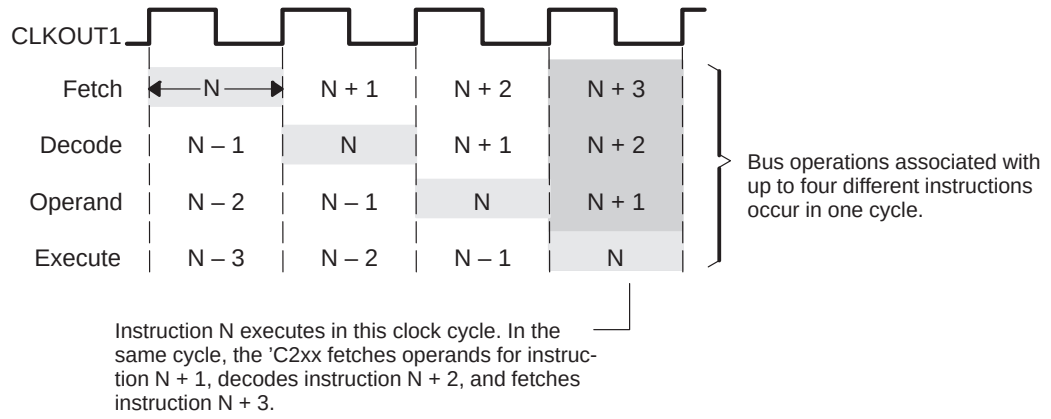
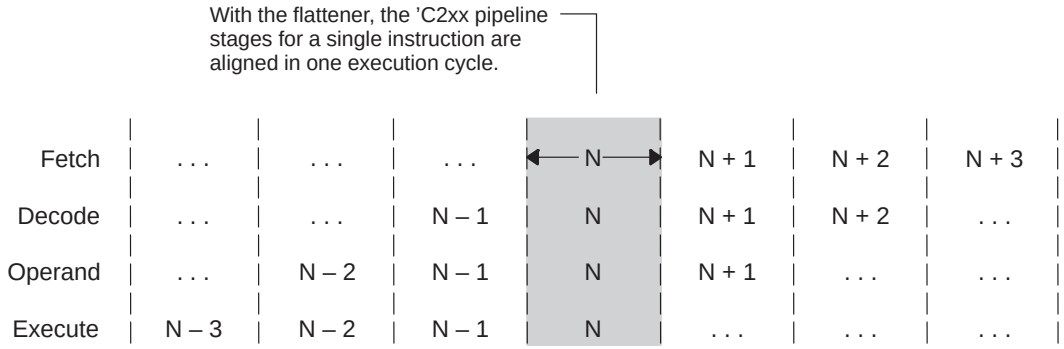


Figure 1–6. How the Flattener Affects the Stages in the ’C2xx Instruction Pipeline



When you use the flattener, you can select one of the following modes:

- *Flat multi word* shows the addresses of the first and second words in a two-word instruction. This produces more complete disassembly information, which is beneficial when the COFF file for your target code is not available. However, this mode uses more trace memory because it requires another sample to store the second word.
- *Flat first word* shows only the address of the first word of a two-word instruction. In this mode, you do not see the address of the second word; for data bus activity associated with the second word, the sample shows the address of the first word of that instruction. This mode uses less trace memory than flat multi word mode uses.

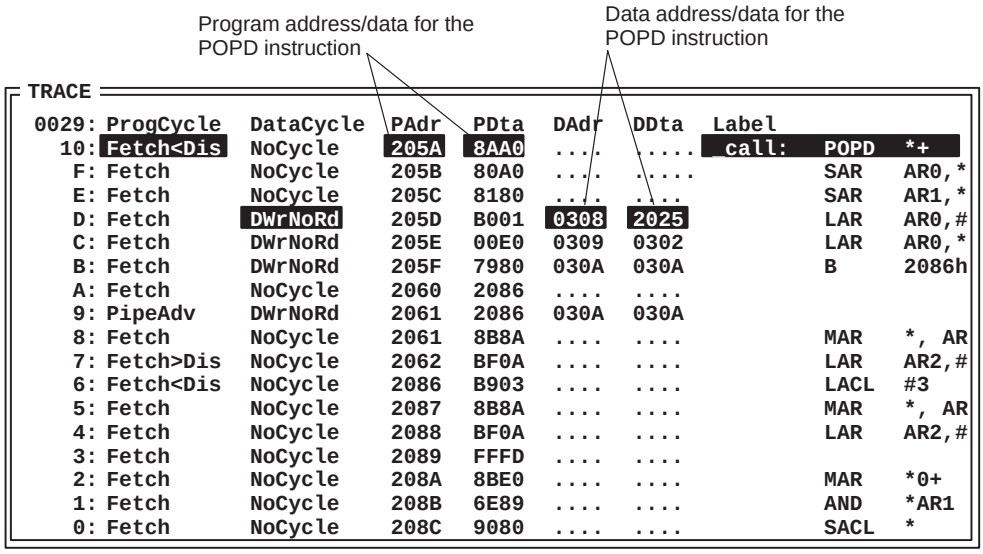
If you need to see instructions as they occur without adjustments for the ’C2xx pipeline, you can disable the flattener (unflattened mode).

See Section 3.1, *Choosing a Flattener Mode*, on page 3-2 for information about choosing a flattener mode.

Figure 1–7 (a) shows samples collected for a block of code using unflattened mode. Figure 1–7 (b) shows the samples collected for the same block of code using flat multi word mode. Notice that the flattener aligns the data accesses for a particular instruction with the corresponding execution cycle.

Figure 1–7. Unflattened Mode Versus Flat Multi Word Mode

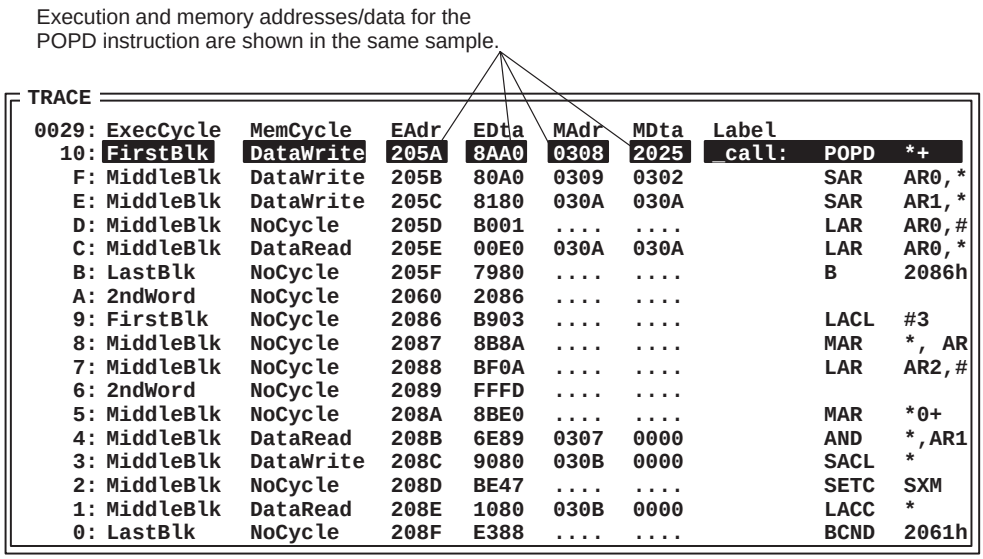
(a) Samples collected in unflattened mode



TRACE

	ProgCycle	DataCycle	PAdr	PDta	DAdr	DDta	Label		
0029:									
10:	Fetch<Dis	NoCycle	205A	8AA0	....	.....	_call:	POPD	*+
F:	Fetch	NoCycle	205B	80A0	....	.....		SAR	AR0,*
E:	Fetch	NoCycle	205C	8180	....	....		SAR	AR1,*
D:	Fetch	DWrNoRd	205D	B001	0308	2025		LAR	AR0,#
C:	Fetch	DWrNoRd	205E	00E0	0309	0302		LAR	AR0,*
B:	Fetch	DWrNoRd	205F	7980	030A	030A		B	2086h
A:	Fetch	NoCycle	2060	2086	....	....			
9:	PipeAdv	DWrNoRd	2061	2086	030A	030A			
8:	Fetch	NoCycle	2061	8B8A	....	....		MAR	*, AR
7:	Fetch>Dis	NoCycle	2062	BF0A	....	....		LAR	AR2,#
6:	Fetch<Dis	NoCycle	2086	B903	....	....		LACL	#3
5:	Fetch	NoCycle	2087	8B8A	....	....		MAR	*, AR
4:	Fetch	NoCycle	2088	BF0A	....	....		LAR	AR2,#
3:	Fetch	NoCycle	2089	FFFD	....	....			
2:	Fetch	NoCycle	208A	8BE0	....	....		MAR	*0+
1:	Fetch	NoCycle	208B	6E89	....	....		AND	*AR1
0:	Fetch	NoCycle	208C	9080	....	....		SACL	*

(b) Samples collected in flat multi word mode



TRACE

	ExecCycle	MemCycle	EAdr	EDta	MAdr	MDta	Label		
0029:									
10:	FirstBlk	DataWrite	205A	8AA0	0308	2025	_call:	POPD	*+
F:	MiddleBlk	DataWrite	205B	80A0	0309	0302		SAR	AR0,*
E:	MiddleBlk	DataWrite	205C	8180	030A	030A		SAR	AR1,*
D:	MiddleBlk	NoCycle	205D	B001	....	....		LAR	AR0,#
C:	MiddleBlk	DataRead	205E	00E0	030A	030A		LAR	AR0,*
B:	LastBlk	NoCycle	205F	7980	....	....		B	2086h
A:	2ndWord	NoCycle	2060	2086	....	....			
9:	FirstBlk	NoCycle	2086	B903	....	....		LACL	#3
8:	MiddleBlk	NoCycle	2087	8B8A	....	....		MAR	*, AR
7:	MiddleBlk	NoCycle	2088	BF0A	....	....		LAR	AR2,#
6:	2ndWord	NoCycle	2089	FFFD	....	....			
5:	MiddleBlk	NoCycle	208A	8BE0	....	....		MAR	*0+
4:	MiddleBlk	DataRead	208B	6E89	0307	0000		AND	*,AR1
3:	MiddleBlk	DataWrite	208C	9080	030B	0000		SACL	*
2:	MiddleBlk	NoCycle	208D	BE47	....	....		SETC	SXM
1:	MiddleBlk	DataRead	208E	1080	030B	0000		LACC	*
0:	LastBlk	NoCycle	208F	E388	....	....		BCND	2061h

Chapter 2

About the BTT Software Interface

The XDS522A provides you with an interface that you can customize with keyboard commands or the mouse. The BTT software interface consists of three windows within the main application window.

This chapter provides a general overview of the parts of the interface and explains various ways to manipulate the interface.

Topic **Page**

2.1 Overview of the BTT Software Interface

The BTT software interface is the main BTT application window, consisting of a menu bar, the TRACE window, the BTT setup window, and the COMMAND window. Figure 2–1 calls out the major sections of the interface. Table 2–1 lists each section with brief descriptions.

Figure 2–1. Features of the BTT Software Interface

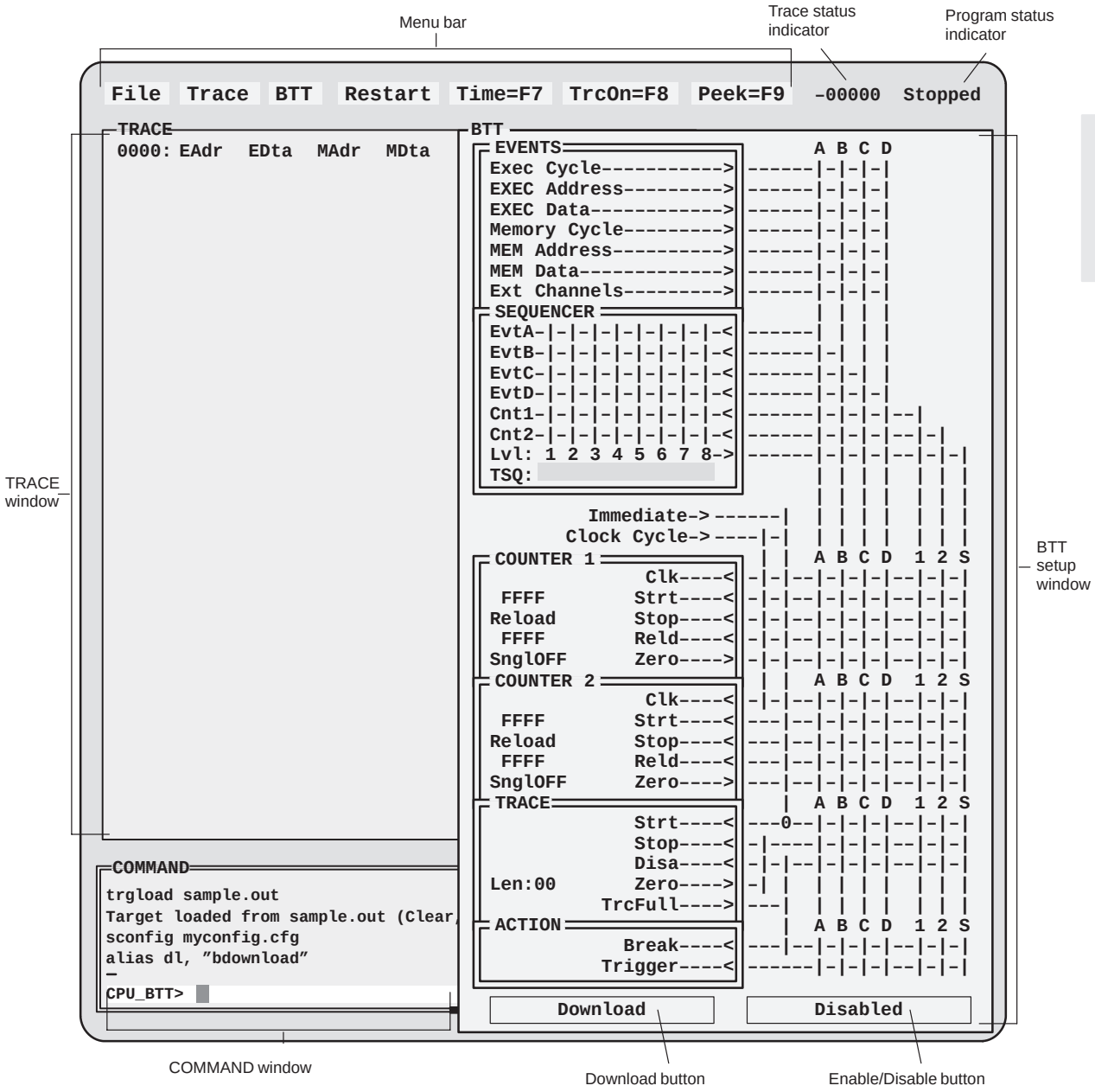


Table 2–1. Interface Feature Descriptions

This part of the interface...	Is used to...	See...
Menu bar	Control the functions of the XDS522A	
File menu	❑ Load the target symbol table	page 3-3
	❑ Save the screen configuration	
	❑ Exit the application	page 3-14
Trace menu	❑ Control the display of information in the TRACE window	page 13-16
	❑ Enable or disable tracing	page 13-2
	❑ Filter displayed samples	page 14-5
	❑ Go to a particular sample in the TRACE window	page 13-10
	❑ Define a search pattern to search for samples in the TRACE window	page 14-12
	❑ Update the information in the TRACE window	page 13-9
	❑ Select a tracing mode	page 13-8
	❑ Save the contents of the trace buffer	page 13-18
	❑ Load the saved contents of the trace buffer	page 13-20
	❑ Close the TRACE window	
BTT menu	❑ Download the configuration of the BTT setup window	page 3-10
	❑ Enable or disable the BTT setup window	page 3-11
	❑ Clears the BTT setup window of all connection symbols (0), except at the intersection of Immediate and trace Strt	page 3-7
	❑ Save the configuration of the BTT setup window	
	❑ Load a saved configuration of the BTT setup window	
	❑ Select the flattener mode	page 4-6
Restart menu button	Restore the BTT setup window to its last downloaded configuration and start tracing following a trace disable	page 13-7
Time toggle	Toggle through three timestamp display modes in the TRACE window	page 14-16
TraceON/TraceOFF menu button	Turn tracing on and off	page 13-7
Peek menu button	Display in the TRACE window one screen of the samples you have collected	page 13-9

Table 2–1. Interface Feature Descriptions (Continued)

This part of the interface...	Is used to...	See...
Trace status indicator	Show the status of the circular trace buffer: ❑ When the number in the trace status indicator has a negative sign, the trace buffer has not overflowed. ❑ When the number in the trace status indicator has a plus sign, the trace buffer is overflowing, and old trace samples are being discarded to make room for new trace samples	
Program status indicator	Show when your program is running or stopped	
TRACE window	Display the trace samples collected by the XDS522A	page 13-12
BTT setup window	Control the functions of the XDS522A	
EVENTS box	Configure up to four unique events (A through D)	Chapter 12
SEQUENCER box	Set up a sequence of events for the XDS522A to use when tracing	Chapter 16
COUNTER boxes	❑ Count clock cycles ❑ Count events ❑ Count sequence completions ❑ Count the number of times the other counter counts to zero	Chapter 15
TRACE box	Control when the XDS522A traces samples	Chapter 13
ACTION box	Specify if the XDS522A should generate a hardware breakpoint or trigger pulse	Chapter 17
Download button	Send the current configuration of the BTT setup window to the XDS522A	page 3-10
Enable/Disable button	Enable or ignore (disable) the information in the BTT setup window	page 3-11
COMMAND window	❑ Enter commands ❑ View error messages and other output of the XDS522A	page 2-6 Appendix B

2.2 Entering Commands

Like the debugger, the BTT software interface has flexible command entry. Using a mouse or key combinations, you can enter commands to the BTT through the menus, dialog boxes, and the interactive grid of the BTT setup window. You can also enter commands on the command line in the COMMAND window.

Entering commands with the mouse

Most of the commands you need to operate the BTT software are available in the pulldown menus, dialog boxes, or the interactive grid of the BTT setup window. Access these menus, boxes, and the grid with the select button of your mouse.

Entering commands with key combinations

If you prefer, you can use the keyboard to enter data and manipulate the interface. The keyboard methods for the BTT software are similar to those for the debugger interface. For example:

- You can manipulate the pulldown menus by using key combinations with the ALT key.
- You can enter information in dialog boxes by using key combinations with the ALT, TAB, SPACE, ↑, or ↓ key.
- You can move the cursor in windows or dialog boxes by using the PAGE UP, PAGE DOWN, ↑, or ↓ key.
- You can use the command history feature by using key combinations such as F2, SHIFT TAB, or TAB.

For more information about using the debugger interface, see the *TMS320C2xx C Source Debugger User's Guide*.

Entering commands from the command line

The COMMAND window does not have to be active for you to enter commands on the command line. However, you cannot enter commands on the command line when the BTT setup window is active.

2.3 Making a Window Active

The flexible command entry of the BTT software allows you to make windows active by entering commands from the COMMAND window, by using the mouse, or by using the F6 function key. Making one of the windows active brings that window to the front of the other windows, allowing you to move and size that window, as well as enter commands. The COMMAND window is the active window by default when you bring up the BTT software.

Making a window active using the mouse

You can make any of the BTT software windows active using the mouse. To make a window active using the mouse, click once anywhere on the border of the window.

When a window is active, the gray, single border around the window changes to a double-lined, yellow border. If the window is behind any other windows, it moves to the front when it becomes active.

Making a window active from the command line

You can make the all windows active from the command line. To make the TRACE and COMMAND windows active from the COMMAND line:

1) Make sure that the BTT setup window is not active.

2) From the COMMAND window, enter:

   `win TRACE` ↵

   or

   `win COMMAND` ↵

   or

   `win BTT` ↵

When a window is active, the gray, single border around the window changes to a double-lined, yellow border. If the window is behind any other windows, it moves to the front when it becomes active.

Making a window active using a function key

You can also use the F6 function key to make a window active. Pressing F6 cycles through the windows in the BTT application window, making each window active in turn.

2.4 Sizing a Window

The BTT software allows you to customize the interface by sizing its windows. You can size windows by entering commands from the command line or by using the mouse.

Sizing the main BTT application window

The main BTT application window consists of the TRACE, BTT setup, and COMMAND windows, and the menu bar. This main window cannot be sized using the mouse or from the COMMAND window; you must use the –bl and –bw command-line options from the icon properties that you can use to size the BTT application window. The *XDS522/XDS522A Emulation System Installation Guide* describes these options.

Sizing a window using the mouse

You can resize, zoom, or unzoom any of the three BTT software windows using the mouse.

To resize a window:

1) Make that window active.

2) Point to the white area in the lower-right corner of the window border.

3) Click and drag the window to the desired size and release the mouse button.

You can zoom any window by clicking the white area in the upper-left corner of the window border. A zoomed window fills the screen, covering the other windows.

Unzoom a window by clicking on the same area that you use to zoom the window. Unzooming a window returns the window to its previously configured size.

Sizing a window from the command line

You can resize the TRACE and COMMAND windows from the command line, but you cannot resize the BTT setup window from the command line. When the BTT setup window is active (when it has the double-lined borders), the COMMAND window does not recognize commands. You must resize the BTT setup window by using the mouse method.

To resize the TRACE and COMMAND windows from the command line:

1) Make the window active that you want to size.

2) From the COMMAND window, enter:

   size [*width*, *length*]

   where *width* is measured in characters and *length* is measured in lines. Do not exceed the width and length of the main BTT application window.

   You might need to experiment before you find your optimal window sizes. The optimal size for each window depends on your monitor size and resolution and the size of your BTT application window.

2.5 Moving a Window

You can customize the BTT software interface by rearranging the BTT setup, TRACE, and COMMAND windows. If you have the windows sized so that they overlap, it can also be helpful to move windows to the front and back of the display.

Moving a window using the mouse

You can move any of the three BTT windows within the main BTT application window by using the mouse.

To move a window:

1) Make that window active.

2) Click and hold the mouse pointer on the yellow, double-lined border on the top or left side of the window.

3) Drag the window to the desired position.

Moving a window using the command line

You can move the COMMAND and TRACE windows using the command line, but you cannot move the BTT setup window from the command line. The COMMAND window does not accept commands when the BTT setup window is active. To move the BTT setup window, use the mouse method.

To move the TRACE or COMMAND windows from the command line:

1) Make that window active.

2) From the COMMAND window, enter:

   move [*X position*, *Y position*]

   where *X position* is the horizontal position measured in characters and *Y position* is the vertical position measured in lines. Do not exceed the length and width of the main BTT application window.

2.6 Saving the Configuration of the BTT Software Interface

Once you have sized and moved the three windows within the main BTT application window, you can save the configuration of the interface. This allows you to bring up this configuration from the command line, rather than resizing and moving the windows every time you use the BTT software.

To save the configuration of the interface:

1) Size and move the three windows within the main BTT application window to your desired configuration.

2) From the COMMAND window, enter:

   ssave *filename*

You can include a pathname in addition to the filename.

To load an interface configuration, from the COMMAND window, enter:

sconfig *filename*

Include the pathname if you stored the saved configuration in a directory other than the default directory. The interface changes to the configuration that you saved under that filename.

Chapter 3

Overview of a Typical Testing and Debugging Session

This chapter describes the basic steps that you need to follow when using the XDS522A to test and debug your code.

Topic **Page**

3.1 Choosing a Flattener Mode

As discussed in the *About the TMS320C2xx pipeline flattener* subsection on page 1-10, the pipeline flattener filters out unexecuted instructions from the trace buffer so that only executed cycles are collected. The flattener also aligns the data with the corresponding execution cycle in the trace buffer.

You can choose between two flattened modes—flat multi word or flat first word—or you can choose the unflattened mode. To select a flattener mode, you must do the following:

Step 1: Set the dip switches on the interface adapter pod:

1) Turn off the power to the XDS522A and unplug the power supply.
2) Disconnect the XDS511 from the interface adapter pod.
3) Set switches S1 through S4, S7, and S8 on the interface adapter pod to the following default settings or to the settings you need:

S1	Down	**S3**	Down	**S7**	Down
S2	Down	**S4**	Up	**S8**	Down

For more information about setting these switches, see the *XDS522/XDS522A Emulation System Installation Guide.*

4) Set switches S5 and S6 for the flattener mode you want, as listed in the following table:

Mode	Switch	Setting
Flat multi word (default)	S5 S6	Up Down
Flat first word	S5 S6	Up Up
Unflattened	S5 S6	Down Down

5) Reconnect the interface adapter pod to the XDS511.
6) Plug in the power supply to the XDS522A and turn on the power.

Step 2: Specify the mode to the BTT software by using the –r*n* option on the command line when you invoke the BTT software:

–r3 selects the unflattened mode

–r4 selects the flat first word mode

–r5 selects the flat multi word mode

You *must* set up the hardware *and* the software for a particular flattener mode for the XDS522A to work properly.

3.2 Invoking the BTT Software

The interface for the XDS522A emulation system is the BTT software. To invoke and set up the BTT software, follow these steps:

1) Invoke the BTT software by double-clicking on one of the Windows icons for the BTT software.

   When you apply power to the XDS522A emulation system, the TMS320C2xx starts running, even if no code is loaded. When you invoke the BTT software, the XDS522A starts tracing the activity of the processor automatically.

2) Select Target code from the File menu to load the symbol-table portion of the object file that you are using. Alternatively, you can use the TRGLOAD command from the command line.

You must load the symbol-table portion of the sample.out object file so that you can use symbolic information when configuring the XDS522A. For example, if you want to be able to use the labels associated with C source functions, the XDS522A must be able to access the symbol-table information.

Note: Generating Symbol-Table Information

To generate the symbol-table information, you must compile and link your code with symbolic debugging information (using the compiler's –g option).

There are other load-type commands available:

- The BTTLOAD command (such as **bttload btt522a.cmd**) reloads the XDS522A firmware, after checking to see if the firmware is already loaded.
- The LOAD command (such as **load btt522a.cmd**) forces a reload of the XDS522A firmware.

The BTTLOAD or LOAD commands reload the firmware and *do not* provide the XDS522A with target code or symbol-table information.

For more information about TRGLOAD, BTTLOAD, and LOAD, or any other BTT software commands, see Appendix A, *XDS522A Emulation System Commands*.

3.3 Invoking the Debugger

To debug and test your code with the XDS522A emulation system, you can use the BTT software in conjunction with the C source debugger. You use the debugger to tell the processor to run your application software. In turn, the BTT software provides an environment in which you can trace and test the execution of your code.

You can invoke the debugger either from the BTT software or from Windows™.

If you plan to generate a hardware breakpoint, you must set up the debugger's on-chip analysis module to recognize and generate hardware breakpoints. This section describes how to set up the analysis module.

Invoking the debugger from the BTT software

To invoke the debugger directly from the BTT software, use the TRGEXE command. The basic syntax for the TRGEXE command is:

trgexe *debugger executable* [*filename*] *options* **–n** *processor name* **–@**

- *debugger executable* is the executable name of the debugger (for example, emu2xxwm.exe). If the debugger executable is not in the current directory, you must specify the full pathname.
- *filename* is an optional parameter that names an object file that the debugger loads into memory during invocation. This eliminates the need to use the debugger's LOAD command once the debugger is invoked. The debugger looks for the file in the current directory; if the file isn't in the current directory, you must supply the entire pathname.
- *options* supply the debugger with additional information. For more information about debugger options, see the *TMS320C2xx C Source Debugger User's Guide*.
- **–n** *processor name* is a debugger option that names the processor that you plan to debug. The processor name must match one of the names defined in your board.cfg file.
- **–@** is a debugger option that allows the debugger to recognize commands that you send from the BTT software. If you do not invoke the debugger with the –@ option, the debugger will not respond to commands from the BTT software.

Once you invoke the debugger, you must do the following to use the BTT software and debugger together:

1) Provide a valid memory map to the debugger that describes your system. You can define this in a batch file that you execute with the TAKE command, in the initialization batch file, or interactively.

2) Load the object code of the application that you want to debug.

3) Ensure that you modify the program counter (PC) to point to the entry point of your application (for example, use the debugger's ? command to modify the PC).

4) Ensure that the debugger's on-chip analysis module is set up to recognize the XDS522A.

If you invoke the debugger with the –@ option, you can send commands to the debugger from the BTT software. For more information, see the *Sending a debugger command to the debugger* subsection on page 3-12.

Invoking the debugger from Windows

To invoke the debugger from Windows, follow these steps:

1) Invoke the debugger by double-clicking on the Windows icon for the debugger.

2) Provide a valid memory map to the debugger that describes your system. You can define this in a batch file that you execute with the TAKE command, in the initialization batch file, or interactively.

3) Load the object code of the application that you want to debug.

4) Ensure that you modify the program counter (PC) to point to the entry point of your application (for example, use the debugger's ? command to modify the PC).

5) Ensure that the debugger's on-chip analysis module is set up to recognize the XDS522A.

If you invoke the debugger with the –@ option, you can send commands to the debugger from the BTT software. For more information, see the *Sending a debugger command to the debugger* subsection on page 3-12.

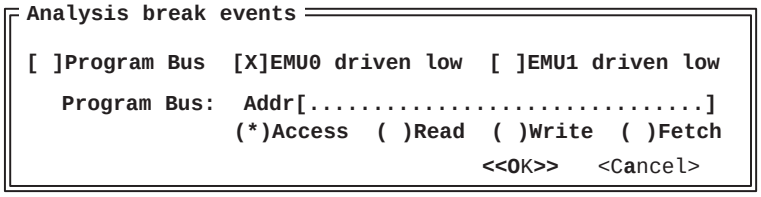
Setting up the analysis module to generate breakpoints

If you plan to generate a hardware breakpoint, you must also set up the debugger's on-chip analysis module to recognize and generate hardware breakpoints. The following steps guide you through this process. For additional information about the analysis module, see the *TMS320C2xx C Source Debugger User's Guide*.

1) From the debugger's Analysis menu, select Enable. This toggles the menu option to *Disable* and enables the debugger's analysis module.

2) From the Analysis menu, select Break. This displays the Analysis break events dialog box.

3) Click on the box next to EMU0 driven low to enable that option. Be sure that the Program Bus and EMU1 driven low options are not enabled (there is no X next to these options).



4) Click on OK.

3.4 Configuring the BTT Setup Window

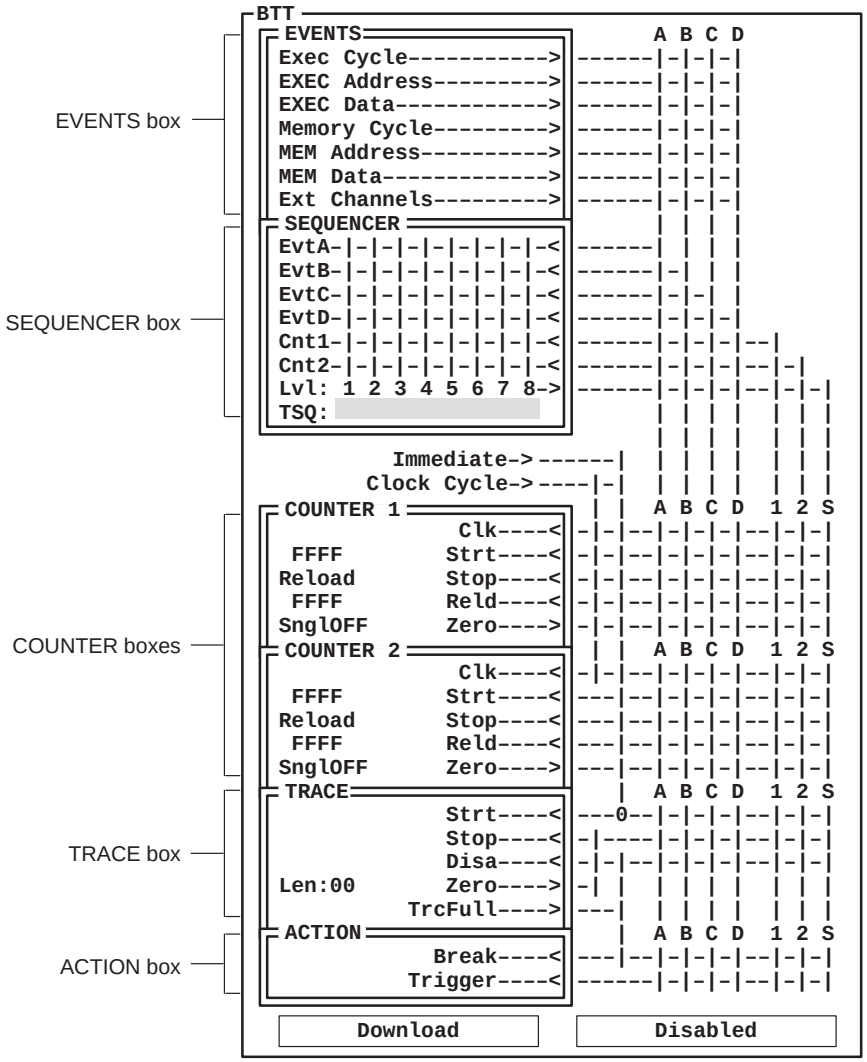
The BTT setup window, shown in Figure 3–1, controls how you want the XDS522A to monitor the activity of the processor and cause actions to occur. You can find information about setting up the BTT setup window in the following chapters:

To set up the . . .	See Chapter . . .
EVENTS box to configure up to four unique conditions	12, *Defining Events*
SEQUENCER box to configure an 8-level state machine to detect a particular sequence of events or counter outputs	16, *Sequencer*
COUNTER boxes to configure one or two 16-bit counters	15, *Counters*
TRACE box to define when to start, stop, and/or disable tracing	13, *Collecting Trace Samples and Using the TRACE Window*
ACTION box to define when to generate hardware breakpoints and trigger pulses	17, *Breakpoints and Triggers*

Before you configure the BTT setup window, clear out any previous configurations. You can clear the configuration in one of these ways:

- ❏ Select Clear from the BTT menu.
- ❏ Enter BCLEAR from the COMMAND window.

Figure 3–1. BTT Setup Window

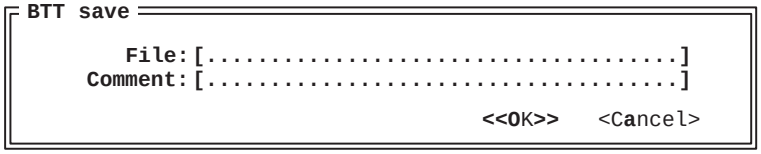


Saving and loading the configuration

You can save any XDS522A configuration that you define in the BTT setup window. This allows you to load and reuse configurations.

To save the current XDS522A configuration:

1) Select Save from the BTT menu. This displays the BTT save dialog box:



2) Enter a filename in the *File* field.

3) If you want to add a 33-character comment to the saved file, use the *Comment* field. You can use this comment to describe the configuration, which will help you distinguish among several saved configurations. You can view the file with your usual text editor; the comment is shown at the beginning of the file.

4) Click on OK.

You can also save the configuration by using a command. The syntax for the BSAVE command is:

bsave *filename*

To load a saved configuration, select Load from the BTT menu. This displays the BTT load dialog box in which you enter the file name.

You can also use the BLOAD command to load the configuration:

bload *filename*

3.5 Downloading the Configuration to the XDS522A

Once you have configured the BTT setup window, you must *download* the configuration to the XDS522A. You can download the configuration in one of these ways:

- Click on the *Download* button at the bottom of the BTT setup window.
- Select Download from the BTT menu.
- Enter BDOWNLOAD from the COMMAND window.

The Download button at the bottom of the BTT setup window changes color depending on what the XDS522A is doing or what you need to do:

- When you make changes in the BTT setup window, the Download button turns yellow to remind you to download the configuration to the XDS522A. The button also turns yellow the first time you click in the BTT setup window following a configuration download.
- During a download, the Download button turns gray.
- When a new configuration is downloaded, the Download button is blue.

3.6 Enabling the XDS522A

Once you have downloaded the configuration, you must be sure that the XDS522A is enabled. To see if it is enabled, look at the lower right corner of the BTT setup window.

- If you see the word *Enabled*, the XDS522A is enabled.
- If you see the word *Disabled*, the XDS522A is disabled.

You can enable the XDS522A in one of these ways:

- Click on the *Disabled* button at the bottom of the BTT setup window. Clicking on the word *Disabled* toggles the button to *Enabled* and enables the XDS522A. Likewise, clicking on the word *Enabled* toggles the button to *Disabled* and disables the XDS522A. The Enabled/Disabled button always shows the current state of the XDS522A (enabled or disabled).
- Select Enable from the BTT menu.
- Enter BENABLE from the COMMAND window.

3.7 Running Your Application

Once you have configured and enabled the XDS522A, you can run your program. You use the debugger to cause the processor to run your program. Follow these basic steps:

1) Ensure that you modify the program counter (PC) to point to the entry point of your application (for example, use the debugger's ? command to modify the PC or use the debugger's RESTART command).

2) Run the application by using a debugger run command (such as RUN or GO).

If you invoke the debugger with the –@ option (either from Windows or with the TRGEXE command), you can use the BTT software to send commands to the debugger. This allows you to perform the following from the BTT software:

- ❏ Send any debugger command to the debugger
- ❏ Start the target application and wait until it enters debug mode
- ❏ Single-step through the target application and wait until it enters debug mode
- ❏ Stop a target application

The remainder of this section describes how to send commands to the debugger from the BTT software.

Sending a debugger command to the debugger

You can use the BTT software to send any debugger command to the debugger. This is useful when you want to enter debugger commands periodically and avoid switching between the BTT software and debugger applications. To send commands to the debugger, use the TRGSEND command. The syntax for the TRGSEND command is:

trgsend *debugger command*

The *debugger command* that you specify is sent directly to the debugger's command interpreter. Any progress or error messages generated by the debugger are not shown in the BTT software's COMMAND window. If you want to see these error messages, you must look in the debugger's COMMAND window.

When you use the TRGSEND command, control is immediately returned to the command line of the BTT software—the BTT software does not wait for the debugger to finish executing the command.

The TRGSEND command is useful when you want to enter debugger commands periodically and avoid switching between the BTT software and debugger applications. Here are some typical command sequences that you might want to send to the debugger:

- Set up the analysis feature to recognize and generate hardware breakpoints:

```
trgsend take analysis.cmd ↵
trgsend asys_on ↵
trgsend stop_emu0 ↵
```

- Reset the program entry point and run the program:

```
trgsend restart ↵
trgsend run ↵
```

If you frequently enter these commands, you can create an alias for the command sequence. For more information, see Section A.1, *Defining Your Own Command Strings*, on page A-2.

Starting a target application

You can send the debugger's RUN command to the debugger by using the TRGRUNWAIT command, which causes the target application to start running and to wait until the processor finishes running due to a software breakpoint or an analysis break event. This command freezes the BTT software interface until the target stops running or until you press ESC in either of the application windows.

Single-stepping through code

You can single-step through your target application by using the TRGSTEPWAIT command, which causes the processor to single-step through the target application and wait until the processor finishes stepping due to a software breakpoint or an analysis break event. This command freezes the BTT software interface. To cancel the wait mode, press ESC in the BTT application window.

Stopping a target application

While the processor is running, you can send an escape sequence to the debugger by using the TRGSTOP command, which causes the processor to stop executing the target application.

3.8 Examining Your Results

In most cases, you will use the TRACE window to examine the results of your testing session. The TRACE window shows you the current contents of the trace buffer. Chapter 13, *Collecting Trace Samples and Using the TRACE Window*, describes how to update the TRACE window, display information about trace samples, and move through the TRACE window. Chapter 14, *Viewing Trace Samples*, tells you how to filter and search the TRACE window to view different trace samples.

3.9 Closing the Software

There are several methods that you can use to close the BTT software or the C source debugger:

To close the . . .	Use one of these methods . . .
BTT software	❑ From the File menu, select Exit. ❑ From the BTT COMMAND window, enter: `quit` ❑ From the Control box (in the upper left corner of the application window), select Close. ❑ Double-click on the Control box.
C source debugger	❑ From the debugger's COMMAND window, enter: `quit` ❑ From the Control box, select Close. ❑ Double-click on the Control box.

Part I
Overview

Part II
Tutorial

Part III
User Reference

Part IV
Appendixes

Part II

Chapter 4

Getting Started

This chapter describes how to use the tutorial in this book and how to find help when you get off track. It also shows you how to choose a flattener mode and how to start a testing and debugging session.

Important! This tutorial assumes that you have correctly and completely installed your XDS522A emulation system. For more information, see the *XDS522/XDS522A Emulation System Installation Guide*.

In addition, it is assumed that you have correctly installed the emulator version of the TMS320C2xx C source debugger and that you are already familiar with the debugger's basic features. If you have never used the C source debugger, complete the tutorial in the *TMS320C2xx C Source Debugger User's Guide* before you start this tutorial.

Topic **Page**

4.1 Before You Begin

Before you try the lessons in this tutorial, you need to understand how to identify the key procedures, alternate ways to perform tasks, what to do if you see unexpected results, and where to go for more help.

Important! Please familiarize yourself with the *Notational Conventions* section of the Preface, particularly the boxed symbols shown on page vi.

The lessons about related topics are grouped into chapters. These groups of lessons are not long, and because the lessons in each chapter are closely related, it's best to take breaks between chapters rather than between lessons.

Identifying key procedures

Once you have completed all parts of the tutorial, you might want to refer to many of the lessons presented and recall the procedures that you followed. To find procedures easily, look for the shaded boxes. This is an example of a procedure:

> Load a previously saved XDS522A configuration.
>
> 1) Clear out the current configuration by selecting Clear from the BTT menu.
>
> 2) From the BTT menu, select Load. This displays the BTT load dialog box.
>
> 3) Enter **myconfig.btt** as the filename and click on OK.

Using the command interface of the BTT software

Like the debugger, the BTT software interface has flexible command entry. You can type commands or use a mouse, function keys, pulldown menus, or dialog boxes.

The tutorial assumes that you are using a mouse to perform most of the functions. However, if you prefer, you can use the keyboard to enter data and manipulate the interface. The keyboard methods for the BTT software are similar to those for the debugger interface. For more information, see the *Entering commands with key combinations* subsection on page 2-6.

What to do when the software is not doing what you expect

Because the XDS522A emulation system is powerful and complex, you might experience some difficulty when trying to configure it. In the BTT software, there are many places where you can set up information, and it's easy to tell the XDS522A to do something that you did not intend.

If you try the first few lessons and the software does not seem to be doing what you expect, try the following:

1) Go through the tutorial steps for that section again.

2) If you're still experiencing problems, see Appendix B, *Troubleshooting*, for additional information.

Need additional help or hints?

Online help is available to provide information about the menu options, dialog boxes, and the contents of the BTT application window. If you have not installed the online help, find the XDS522A Online Help diskette that came with your XDS522A emulation system. The diskette includes a readme file with installation instructions.

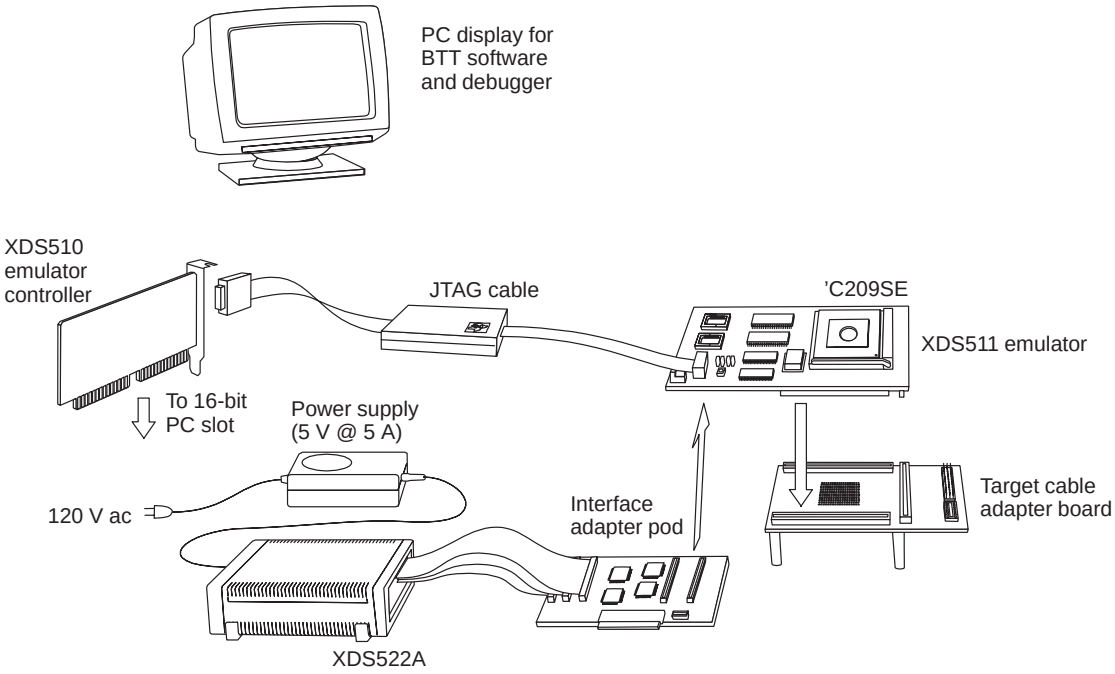
After you have installed the online help, go to the Windows program group that contains the icon for the online help:

To open the online help, double-click on the icon.

4.2 Verifying Your Hardware Setup

This tutorial assumes that you have configured your hardware for standalone mode, as shown in Figure 4–1. This setup includes the the XDS510 and XDS511 hardware, the SE device, the target cable adapter board, the interface adapter pod, and the XDS522A.

Figure 4–1. Standalone Setup for Use With the Tutorial



For this tutorial, be sure that you use the hardware shown in Figure 4–1 and that you configure your jumper settings as described in Table 4–1. For more information about installing the components of the XDS522A and about the jumper settings, see the *XDS522/XDS522A Emulation System Installation Guide* and the *XDS51x Emulator Installation Guide*.

Table 4–1. Jumper Settings for Standalone Mode

(a) Jumper settings for the XDS511 emulator

Jumper	Jumper Settings
JP1	Jumper only the following:
	1–2
	7–8
	9–10
JP5	Jumper only the following:
	1–2
	7–8
	11–12

(b) Jumper settings for the target cable adapter board

Jumper	Jumper Settings
JP1	Jumper only the following:
	1B–1C
	2B–2C
	3B–3C
	4B–4C
	5B–5C
	6B–6C
	7B–7C
	8B–8C
JP2	Jumper only 2B–3B

(c) Jumper settings for the interface adapter pod

Jumper	Jumper Settings
JP6	Jumper pins together
JP7	Jumper pins together
JP8	Jumper pins labeled XTDO and ETDI

4.3 Choosing a Flattener Mode

As discussed in the *About the TMS320C2xx pipeline flattener* subsection on page 1-10, the XDS522A provides a flattener that assists you in debugging applications for the 'C2xx. The flattener filters out unexecuted instructions from the trace buffer so that only executed cycles are collected. The flattener also aligns the data with the corresponding execution cycle in the trace buffer.

There are two flattened modes that you can choose between:

- *Flat multi word* shows the addresses of the first and second words in a two-word instruction.
- *Flat first word* shows only the address of the first word of a two-word instruction.

If you need to, you can disable the flattener features. This mode is called *unflattened* mode.

To select one of the flattened modes or to select the unflattened mode, you must do the following:

1) Set the dip switches on the interface adapter pod.
2) Specify the mode to the BTT software by using the –r*n* option.

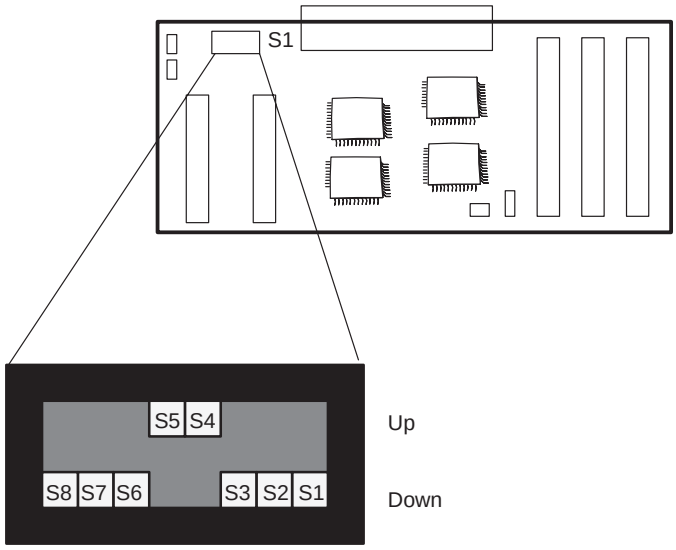
You *must* set up the hardware *and* the software for a particular flattener mode in order for the XDS522A to work properly.

In all parts of this tutorial, you'll be using the *flat multi word* mode. Before you turn on your XDS522A and invoke the software, you must ensure that the dip switches on the interface adapter pod are set correctly and that you specify the –r5 option on the command line.

Set the dip switches for flat multi word mode:

1) Turn off the power to the XDS522A and unplug the power supply.
2) Disconnect the XDS511 from the interface adapter pod.
3) Examine the dip switches on the interface adapter pod. Be sure that they are set to the flat multi word settings shown in Figure 4–2.
4) Reconnect the interface adapter pod to the XDS511.
5) Plug in the power supply to the XDS522A and turn on the power.

Figure 4–2. Dip Switch Settings for Flat Multi Word Mode



Specify flat multi word mode to the BTT software by using the –r5 option:

1) In Windows, select the icon for the flat multi word version of the BTT software.
2) From the File menu, select Properties. This displays the Properties dialog box.
3) Move your cursor to the Command Line box.
4) Be sure that the –r*n* option appears as the following:

   –r5
5) Click on OK.

4.4 Starting a Debugging/Testing Session

To debug and test your code with the XDS522A emulation system, you use the BTT software in conjunction with the C source debugger. You use the debugger to tell the processor to run your application software. In turn, the BTT software provides an environment for you to trace and test the execution of your code.

To start a debugging and testing session, you must invoke the debugger and BTT software and load the sample program. After you have invoked these tools, you can resize your windows to maximize the information shown on your PC screen.

Setting up the debugger and the BTT software

Debug

Set up the C source debugger:

1) Invoke the C source debugger by double-clicking on the Windows icon for the debugger.
2) From the COMMAND window, enter:

   `take se_init` ↵

   This command sets up the correct memory map by directing the debugger to execute the memory map commands contained in the se_init.cmd file.
3) From the COMMAND window, load the sample.out program by entering:

   `load sample` ↵

When you are using the debugger with the BTT software, you must ensure that you modify the program counter (PC) to point to the entry point of your application (for example, use the debugger's ? command to modify the PC). In this lesson, when you load the sample.out file, the PC is already pointing to the entry point of the sample program (c_int0).

BTT

Set up the BTT software:

1) Invoke the BTT software by double-clicking on the Windows icon for the flat multi word version of the BTT software.

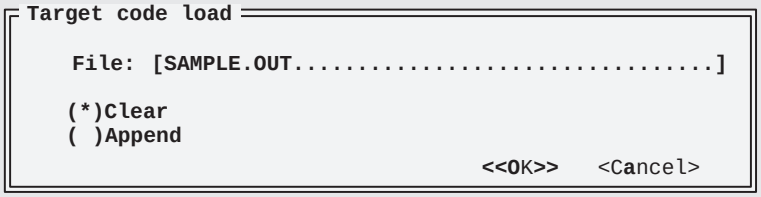
   This brings up the BTT application window. It might take a few seconds for the BTT application window to display and for the software to finish executing its initialization routine.

When you apply power to the XDS522A emulation system, the TMS320C2xx starts running, even if no code is loaded. When you invoke the BTT software, the XDS522A starts tracing the activity of the processor automatically.

2) Load the symbol-table portion of the the sample.out object file:

   a) From the File menu, select Target Code. This displays the Target code load dialog box.

   b) Enter **sample.out** for the File parameter. Be sure that the asterisk next to Clear is set, and click on OK.



You must load the symbol-table portion of the sample.out object file so that you can use symbolic information when configuring the XDS522A. For example, if you want to be able to use the labels associated with C source functions, the XDS522A must be able to access the symbol-table information.

Note: Understanding the Different Load Commands

In this lesson, you used the File→Target Code command to load the symbol table of the sample.out file. This is equivalent to entering the TRGLOAD command from the COMMAND window.

There are other load-type commands available:

- The BTTLOAD command (such as **bttload btt522a.cmd**) reloads the XDS522A firmware, after checking to see if the firmware is already loaded.
- The LOAD command (such as **load btt522a.cmd**) forces a reload of the XDS522A firmware.

For this tutorial, *do not* use the BTTLOAD or LOAD commands to load the symbol table of the sample.out file. Attempting to do so could confuse the debugger, and you will need to close the BTT software and reinvoke it.

For more information about TRGLOAD, BTTLOAD, and LOAD, or any other BTT software commands, see Appendix A, *XDS522A Emulation System Commands*.

Show me everything I need to see

When you invoke the debugger and the BTT software, two application windows—one for the BTT software and one for the debugger—cover most of your screen and overlap one another. You will need to view portions of both of these windows at the same time. This lesson discusses some tips for resizing the application windows and the windows inside the BTT application window.

Note: Manipulating the BTT Software Windows and Menus

In most cases, the interface for the BTT software works the same way that the C source debugger interface works. You can size, move, and zoom windows using the same techniques, and you can access the menu options in the same way.

For more information about using the debugger interface, see the *TMS320C2xx C Source Debugger User's Guide*.

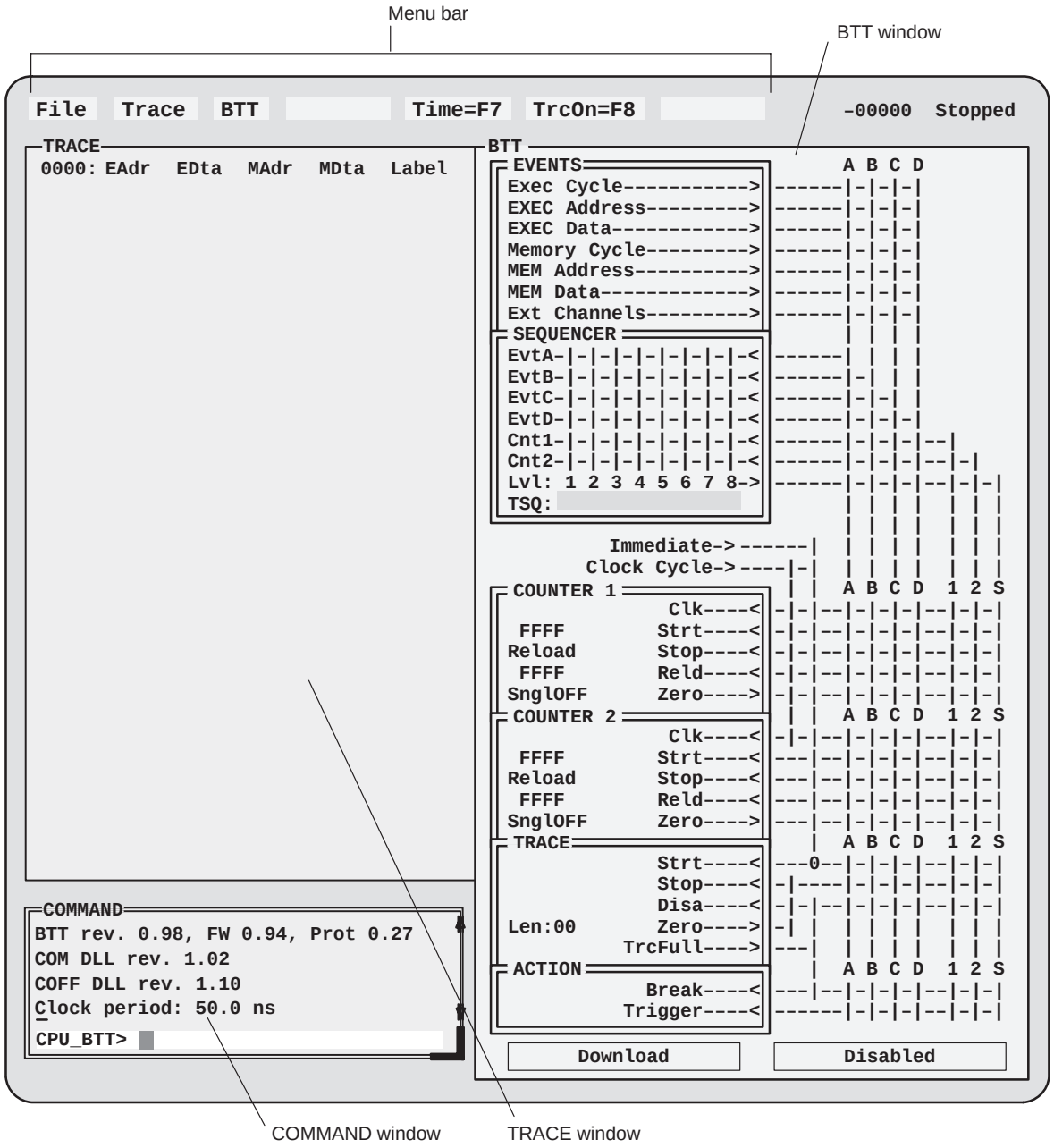
BTT

The *XDS522/XDS522A Emulation System Installation Guide* describes the –bl and –bw command-line options that you can use to size the BTT application window. Try to start out with a window that is at least 80 characters wide and 45 lines long.

Figure 4–3 shows a typical BTT application window. There are three windows inside the BTT application window:

- The TRACE window displays the samples that are in the trace buffer and allows you to search for and filter the information that you collect.
- The BTT setup window allows you to define events that specify when to collect samples and when to generate breakpoints and triggers.
- The COMMAND window provides an area for entering commands and for displaying command output, errors, and messages.

Figure 4–3. The BTT Application Window



lesson continues on the next page →

Resize the TRACE and COMMAND windows:

1) Make the TRACE window active: from the COMMAND window, enter:

   win TRACE ↵

2) With the TRACE window active, resize it to show as many trace samples as possible:

   size 80,41 ↵

   Depending on your monitor size and resolution and the size of your application window, this can cause the TRACE window to cover most of the BTT application window.

3) Make the COMMAND window active, move it to the the bottom of the screen, and resize it:

   win COMMAND ↵
   move 0,36 ↵
   size 70,8 ↵

You cannot size the BTT setup window by entering the SIZE command from the COMMAND window. When the BTT setup window is active (when it has the double-lined borders), the COMMAND window does not recognize commands. You need to resize the BTT setup window by using the mouse method.

Resize the BTT setup window:

1) Press F6 enough times to bring the BTT setup window to the top of the display. Like in the debugger, pressing F6 cycles through the windows in the BTT application window, making each window active in turn.

2) Point to the lower right corner of the BTT setup window.

3) Press and hold the left mouse button; move the mouse to make the window longer. Make the window large enough to see the word *Download* at the bottom of the window (Figure 4–3 shows the entire BTT setup window).

4) When the window is the proper size, click the left mouse button to stop resizing the window.

Try This: To avoid entering the WIN, SIZE, and MOVE commands to size and move the windows each time you invoke the BTT software, you can create a batch file that includes these commands. From the COMMAND window, you can enter the TAKE command to execute the commands listed in this batch file (for more information about creating batch files and using the TAKE command, see Section A.2, *Creating a Batch File*, on page A-4). You can also put the following commands in your batch file to automatically resize the BTT setup window:

```
win BTT
size 26,45
```

These commands work in a batch file because a batch file does not require the COMMAND window to be active for a command to be recognized.

Try This: If you don't want to create a batch file, you can save your current screen configuration to a file. From the COMMAND window, enter the SSAVE command:

```
ssave myconfig.cfg ↵
```

This saves the current screen configuration to a file (in this case, *myconfig.cfg*). When you exit the BTT software and reinvoke it, you can call up your saved screen configuration by using the SCONFIG command:

```
sconfig myconfig.cfg ↵
```

This restores the screen configuration that you saved in *myconfig.cfg.*

Debug

Resize and move the debugger application window:

1) Resize the debugger application window and move it so that it is in the lower right corner of the screen.

2) Position the debugger application window so that you can see it and the BTT application window at the same time. You'll be switching between these applications frequently.

4.5 Summary

Congratulations! In this chapter, you learned how to do the following:

- ❏ Choose the flat multi word mode
- ❏ Invoke the BTT software and debugger
- ❏ Load sample code and symbolic information
- ❏ Resize the application windows and the BTT software windows

If you feel uncomfortable in performing any of these tasks, take some time to review the material in this chapter.

If you are comfortable with these tasks, you're ready to move on to Chapter 5, *Detecting an Event and Halting the Processor*.

Need a break?

If you need a break and want to close the BTT software and debugger, follow these directions:

- ❏ To close the BTT software, select Exit from the File menu.
- ❏ To close the debugger, from the COMMAND window, enter:

  quit ↵

To start the software again, follow the instructions in Section 4.4, *Starting a Debugging/Testing Session*, on page 4-8.

Chapter 5

Detecting an Event and Halting the Processor

The XDS522A emulation system is driven by conditions that you specify. These conditions are referred to as *events*. The functional control blocks of the XDS522A (such as the sequencer, counters, and trace control) respond to events that you define. You can define up to four events.

In this chapter, you will define a condition (event) for detecting a particular execution address. When that address is detected, a hardware breakpoint will occur.

Topic **Page**

5.1 Understanding the BTT Setup Window

You can configure the XDS522A by using the BTT setup window. Figure 5–1 shows the BTT setup window with its default settings.

The BTT setup window uses grid lines that create intersections at which you can place connection symbols (connection symbols look like zeros in the intersections). When setting up the breakpoint, tracing, and timing functionality, remember these conventions:

- ❑ Gray lines indicate valid intersections for defining the BTT setup.
- ❑ Yellow lines indicate invalid intersections where no connection symbols are allowed.
- ❑ The <— symbol indicates an input condition.
- ❑ The —> symbol indicates an output condition.

BTT

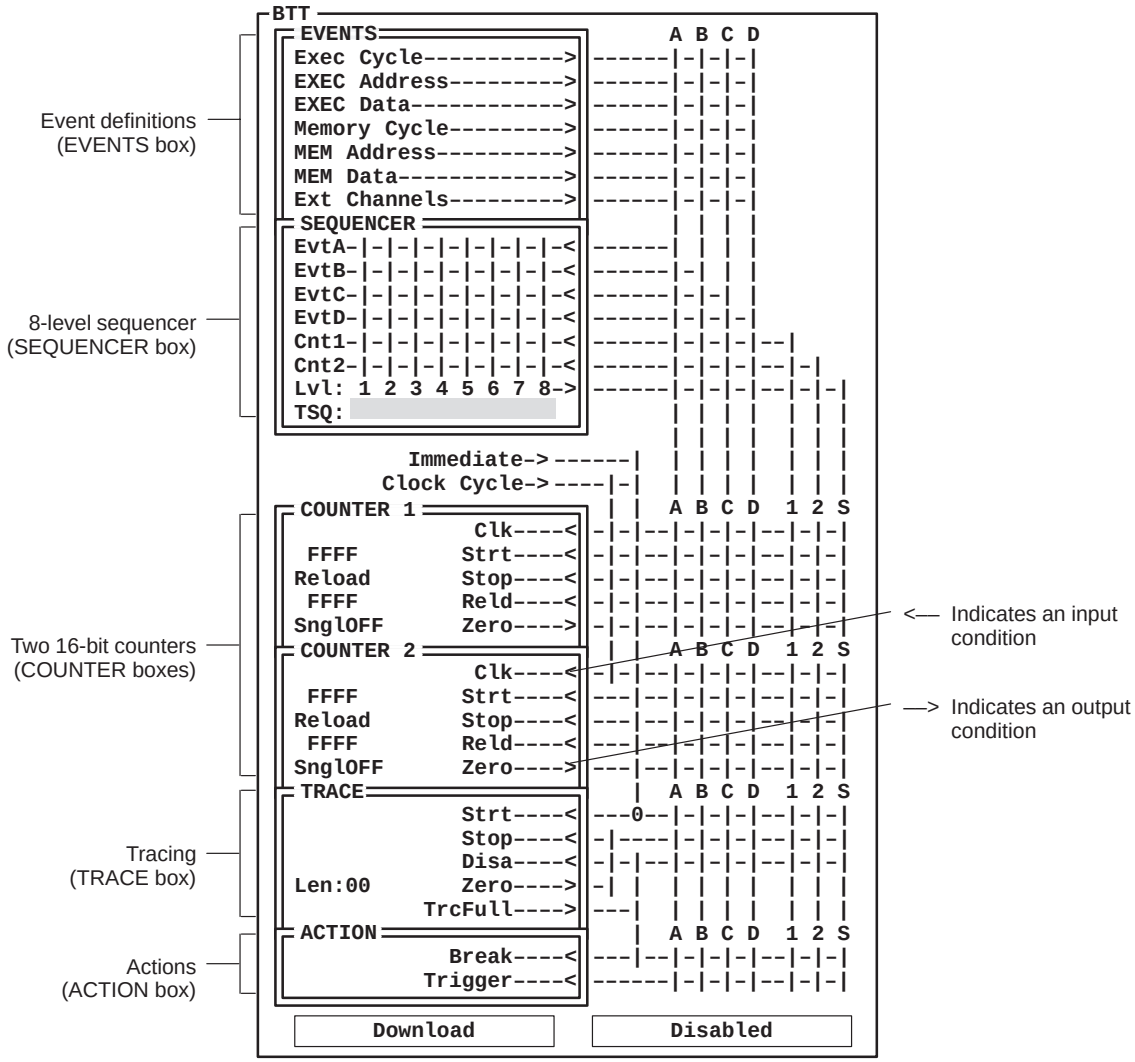
Important! Before you configure the BTT setup window, you want to clear out any previous configuration that you might have made.

Clear out previous configurations by selecting Clear from the BTT menu.

This clears the BTT setup window to its default settings, as shown in Figure 5–1. Notice that all connection symbols (0) are cleared, except at the intersection of *Immediate* and trace *Strt.*

As you go through the steps in this tutorial, if you believe you've made a mistake or want to start a new configuration, you can reset the BTT setup window by selecting the Clear option from the BTT menu.

Figure 5–1. The BTT Setup Window (in a Flattened Mode)



5.2 Defining an Event

At the top of the BTT setup window is the EVENTS box. You can define up to four events (A–D). When these predefined events occur, the XDS522A can count, sequence, start or stop tracing, and/or generate triggers and hardware breakpoints.

You define events by using one or more *qualifiers*. In either of the flattened modes, you can specify these qualifiers:

- ❑ Execution cycle types
- ❑ Execution addresses
- ❑ Data on the execution bus
- ❑ Memory cycle types
- ❑ Memory addresses
- ❑ Data on the memory bus
- ❑ External-channel activity

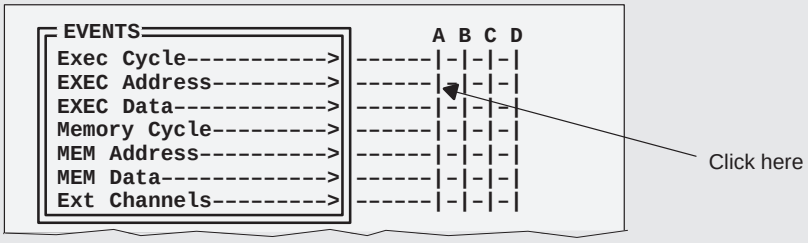
In this lesson, you'll define Event A with one qualifier. The qualifier defines Event A as any access to the address associated with the xcall( ) function. This is the function that you want use to generate a hardware breakpoint.

BTT

Define Event A as any access to the address associated with xcall():

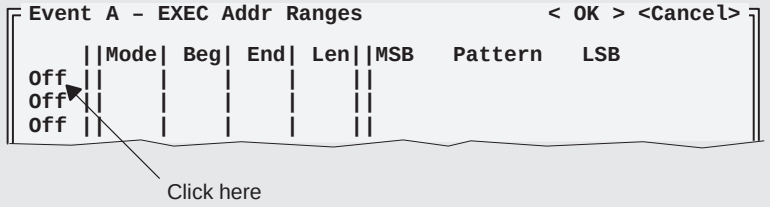
EVENTS Box

1) Set up the qualifier for Event A by clicking on the intersection of *EXEC Address* and Event A:

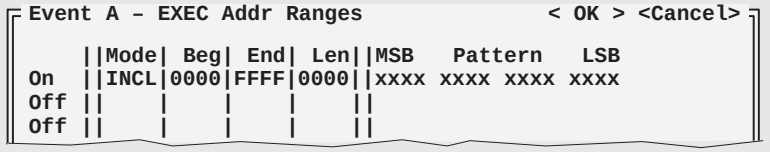


This causes the EXEC Addr Ranges dialog box to appear. Since you are defining Event A as any access to the address associated with xcall(), you want the XDS522A to examine the information on the program (execution) bus of the 'C2xx.

2) In the EXEC Addr Ranges dialog box, click on the first Off listing:

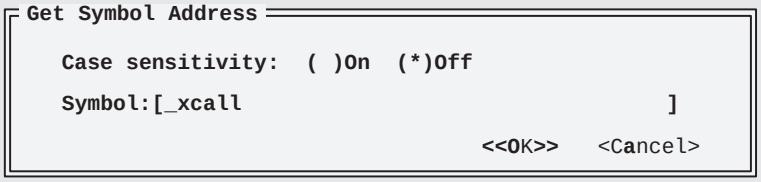


This switches Off to On:

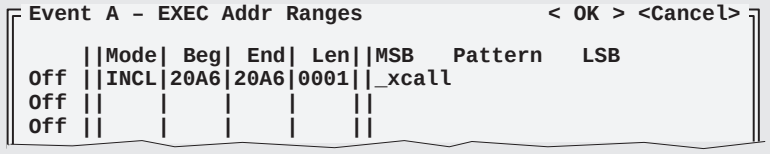


3) Under the Beg (beginning address) heading, click on 0000 and press the SPACE key. This displays the Get Symbol Address dialog box.

4) In the Get Symbol Address dialog box, enter **_xcall** as the Symbol:



5) Click on OK. This dismisses the Get Symbol Address dialog box. The EXEC Addr Ranges dialog box should now look like this:



6) Click on OK.

You should see a connection symbol (0) at the intersection of *EXEC Address* and Event A in the EVENTS box of the BTT setup window.

5.3 Collecting Trace Samples Immediately

You use the TRACE box in the BTT setup window to control when the XDS522A traces the execution of code. This box is different from the TRACE window, which displays the trace samples that the XDS522A collects.

With the TRACE box, you can direct the XDS522A to do the following:

- *Start* collecting trace samples:
  - As soon as your code begins running
  - When a particular event occurs
  - When a counter reaches zero
  - When a sequence of events completes
- *Stop* collecting trace samples:
  - When a particular event occurs
  - When a counter reaches zero
  - When a sequence of events completes
  - When the trace buffer is full
  - When the XDS522A collects a specific number of trace samples

Note: Start and Stop Tracing on the Same Event

If you start and stop tracing on the same event, the XDS522A collects only one sample.

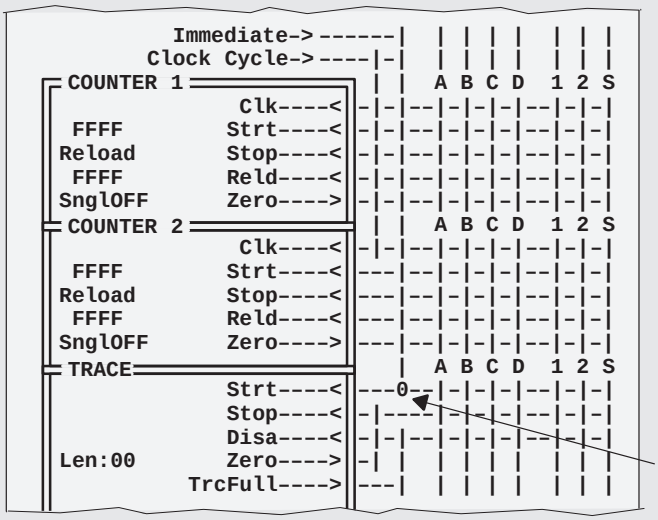
In this lesson, you'll set up the system to start collecting trace samples as soon as sample.out begins running. This type of tracing is called *trace immediate.*

BTT

Set up the XDS522A to start tracing immediately:

TRACE Box

Look at the intersection of *Immediate* and *Strt*.



- ❑ If there is no connection symbol (0), click on the intersection. This creates a connection at that intersection.
- ❑ If there is already a connection symbol, the TRACE box is ready.

5.4 Executing a Hardware Breakpoint

The ACTION box of the BTT setup window allows you to specify what the XDS522A should do upon detection of a certain set of conditions. With the ACTION box, you can direct the XDS522A to do the following:

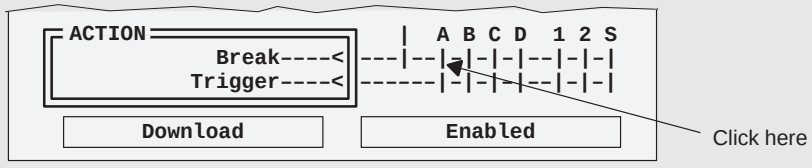
- Generate a hardware breakpoint. You can generate a hardware breakpoint:
  - When the trace buffer is full
  - When a particular event occurs
  - When a counter reaches zero
  - When a sequence of events completes
- Generate a low-to-high trigger pulse on the BNC connector. You can generate a trigger pulse:
  - When a particular event occurs
  - When a counter reaches zero
  - When a sequence of events completes

BTT

Set up the system to execute a hardware breakpoint when Event A occurs:

ACTION Box

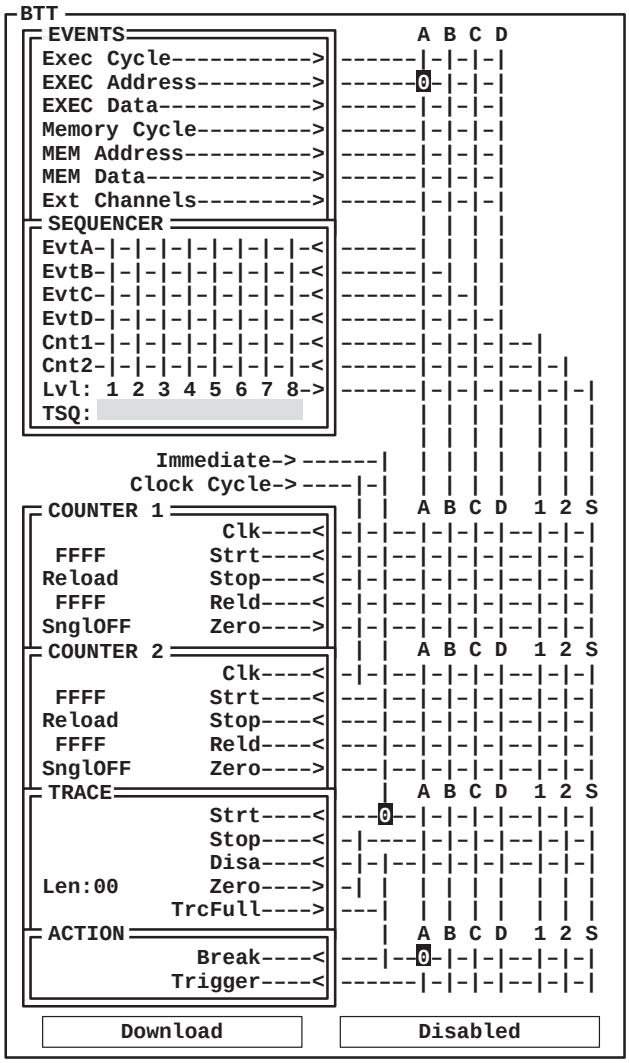
Click on the intersection of *Break* and Event A:



This creates a connection symbol at the intersection of Break and Event A.

5.5 Downloading the Configuration and Enabling the XDS522A

Now that you have configured the EVENTS box, the TRACE box, and the ACTION box, your BTT setup window should look like the following:



The only connection symbols you should see are the ones shown here.

lesson continues on the next page →

BTT

Important! Each time you configure the BTT setup window, you must *download* the configuration to the XDS522A.

Download the configuration to the XDS522A by clicking on the *Download* button at the bottom of the BTT setup window.

It might take a few seconds for the configuration to be downloaded.

The Download button changes color depending on what the XDS522A is doing or what you need to do:

- When you make changes in the BTT setup window, the Download button turns yellow to remind you to download the configuration to the XDS522A. The button also turns yellow the first time you click in the BTT setup window following a configuration download.
- During a download, the Download button turns gray.
- When a new configuration is downloaded, the Download button is blue.

Important! For the XDS522A to work properly, you must enable the XDS522A by selecting Enable from the BTT menu.

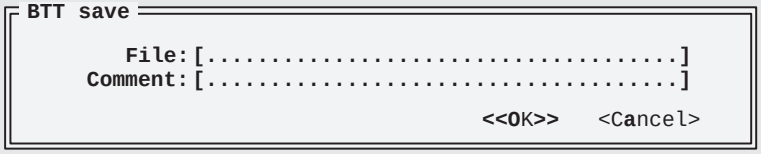
When the XDS522A is enabled, the BTT setup window shows the word *Enabled* in the lower right corner. Clicking on the word *Enabled* toggles the button to *Disabled* and disables the XDS522A. Likewise, clicking on the word *Disabled* toggles the button to *Enabled* and enables the XDS522A.

5.6 Saving the XDS522A Configuration

You can save any XDS522A configuration that you define in the BTT setup window. This allows you to load and reuse configurations.

Save the current XDS522A configuration:

1) Select Save from the BTT menu. This displays the BTT save dialog box:



2) Enter **myconfig.btt** as the File and click on OK. This dismisses the dialog box and saves the current configuration in a file called MYCONFIG.BTT in the current (working) directory.

To load a saved configuration, select Load from the BTT menu. This displays the BTT load dialog box in which you enter the file name.

5.7 Running the Application and Looking at the TRACE Window

The BTT software and the C source debugger work together. The debugger causes the processor to run your application. The BTT software provides an environment for you to trace and test the execution of your code.

Debug

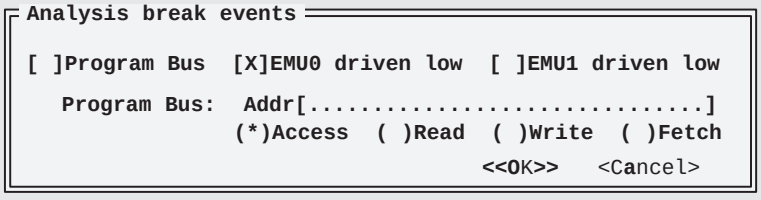
Because you want to generate a hardware breakpoint when a particular address is detected, you must set up the debugger's on-chip analysis module to recognize and generate hardware breakpoints. The following steps take you through this process. For additional information about the analysis module, see the *TMS320C2xx C Source Debugger User's Guide*.

Set up the debugger's on-chip analysis module to recognize and generate hardware breakpoints:

1) From the Analysis menu, select Enable. This toggles the menu option to *Disable* and enables the debugger's analysis module.

2) From the Analysis menu, select Break. This displays the Analysis break events dialog box.

3) Click on the box next to EMU0 driven low to enable that option. Be sure that the Program Bus and EMU1 driven low options are not enabled (there is no X next to these options). Your dialog box should look similar to this:



4) Click on OK.

5) From the Analysis menu, select View. This displays the ANALYSIS window.

Now that you've set up the BTT software and the debugger's analysis module, you are ready to run the program:

1) Reset the program entry point: from the debugger's COMMAND window, enter:

   restart ↵

2) Run the program:

   run ↵

The processor should run until the XDS522A stops execution at the function xcall().

BTT

Use the F6 key to cycle through the windows to bring the TRACE window to the top and make it active.

Alternatively, you can click on the border of the TRACE window to make the window active.

The TRACE window displays the newest samples in the bottom of the window. Notice that the processor executed several instructions past the _xcall label; this is due to the latency between the XDS522A detecting this condition, the XDS522A asserting the EMU0 signal, and the 'C2xx processor responding to the signal.

```
TRACE
000B: EAdr  EDta  MAdr  MDta  Label
   A: 2080  90A0  030B  0002          SACL   *+
   9: 2081  7A80  ....  ....          CALL   _xcall,*
   8: 2082  20A6  ....  ....
   7: 20A6  8AA0  030C  2083  _xcall  POPD   *+
   6: 20A7  80A0  030D  030A          SAR    AR0,*+
   5: 20A8  8180  030E  030E          SAR    AR1,*
   4: 20A9  B001  ....  ....          LAR    AR0,#1
   3: 20AA  00E8  030E  030E          LAR    AR0,*0+,AR0
   2: 20AB  BC04  ....  ....          LDP    #4
   1: 20AC  1A6E  ....  ....          LACC   006eh,10
   0: 20AC  ....  206E  00A9
```

Event A (_xcall)

Samples collected during the latency

5.8 Summary

Congratulations! In this chapter, you learned how to do the following:

- ❏ Clear the XDS522A configuration
- ❏ Define an event with one qualifier
- ❏ Collect trace samples as soon as the application begins running (trace immediate)
- ❏ Execute a hardware breakpoint
- ❏ Download a configuration to the XDS522A
- ❏ Enable and disable the XDS522A
- ❏ Save the XDS522A configuration
- ❏ Configure the debugger to recognize and generate hardware breakpoints
- ❏ Run an application and looking at the TRACE window

If you feel uncomfortable in performing any of these tasks, take some time to review the material in this chapter.

If you are comfortable with these tasks, you're ready to move on to Chapter 6, *Understanding the Status Indicators and the TRACE Window*.

Need a break?

If you need a break and want to close the BTT software and debugger, follow these directions:

- ❏ To close the BTT software, select Exit from the File menu.
- ❏ To close the debugger, from the COMMAND window, enter:

  quit ↵

To start the software again, follow the instructions in Section 4.4, *Starting a Debugging/Testing Session*, on page 4-8.

Chapter 6

Understanding the Status Indicators and the TRACE Window

In Chapter 5, you learned how to define an event and how to generate a hardware breakpoint when that event occurs. When you ran the sample program, the TRACE window showed the contents of the trace buffer (the listing of samples collected by the XDS522A). In this chapter, you'll look more closely at the parts of the TRACE window and the contents of the trace buffer.

Topic **Page**

6.1 Collecting Trace Samples Between Two Events

In this lesson, you will define two events. You will then set up the XDS522A to collect trace samples between those events.

BTT

From the BTT menu, select Clear to reset the XDS522A configuration.

In the BTT setup window, set up the XDS522A to start tracing when Event A is detected and stop tracing when Event B is detected:

EVENTS Box

1) Define Event A as any access to the address associated with the call() function:
   a) Click on the intersection of *EXEC Address* and Event A.
   b) In the EXEC Addr Ranges dialog box, click on the first Off listing to switch it to On.
   c) Under the Beg (beginning address) heading, click on 0000 and press the SPACE key.
   d) In the Get Symbol Address dialog box, enter **_call** as the Symbol, then click on OK.
   e) In the EXEC Addr Ranges dialog box, click on OK.
2) Define Event B as any access to the address 0x2026
   a) Click on the intersection of *EXEC Address* and Event B.
   b) In the EXEC Addr Ranges dialog box, click on the first Off listing to turn it to On.
   c) Under the Beg (beginning address) heading, click on 0000 and enter the address **2026**. Notice that the XDS522A automatically updates the End field with the same address.
   d) Click on OK.

TRACE Box

Set up tracing to start at the detection of Event A and to disable at the detection of Event B:

1) Be sure that immediate tracing is not enabled by looking at the intersection of *Immediate* and *Start*.
   - ❑ If there is a connection symbol, click on the intersection to disable the connection.
   - ❑ If there is no connection symbol (0), no action is required.
2) Click on the intersection of *Strt* and Event A.
3) Click on the intersection of *Disa* and Event B.

Important! Don't forget to download the XDS522A configuration and, if necessary, enable the XDS522A.

Debug

Reset the program entry point and run the program:

```
restart ↵
run ↵
```

Because you did not tell the XDS522A to generate a hardware breakpoint and because the code causes an infinite loop, the processor runs indefinitely. Go ahead and let the processor run. This allows you to explore the XDS522A features described in the following lessons.

6.2 Understanding the Status Indicators

Look at the upper right corner of the BTT application window. You should see the word *Running*. This is the *program status indicator*. While the processor is running your code, the status is Running. When the program is not running, the program status indicator says *Stopped.*

To the left of the program status indicator is a number that indicates the status of the circular trace buffer: the *trace status indicator*.

- When the trace status indicator has a negative sign, the XDS522A is currently not collecting trace samples. A negative sign also indicates that the trace buffer has not overflowed.
- When the trace status indicator has a plus sign, the trace buffer is overflowing, and old trace samples are being discarded to make room for new trace samples. While the trace buffer is overflowing, you see a constantly changing number preceded by a plus sign.

Some common values for the trace status indicator are:

When the trace status indicator is…	**The trace buffer is**…
–00000	Empty.
+00000	Full, and no samples have been overwritten. The trace buffer contains 32 768 (0x8000) samples.
+00001	Full, and the oldest trace sample has been discarded.

Keep in mind that when the XDS522A stops collecting trace samples, the processor does not necessarily stop running code. The processor might or might not continue running code, depending on whether or not the execution of the code completes, you halt the execution of code, etc.

Currently, the trace status indicator should be –00029 (from the previous lesson). This means that the XDS522A collected 0x0029 samples in the trace buffer before the XDS522A detected Event B and disabled tracing.

To see the samples that the XDS522A collected, select Update from the Trace menu.

This updates the TRACE window with the contents of the trace buffer.

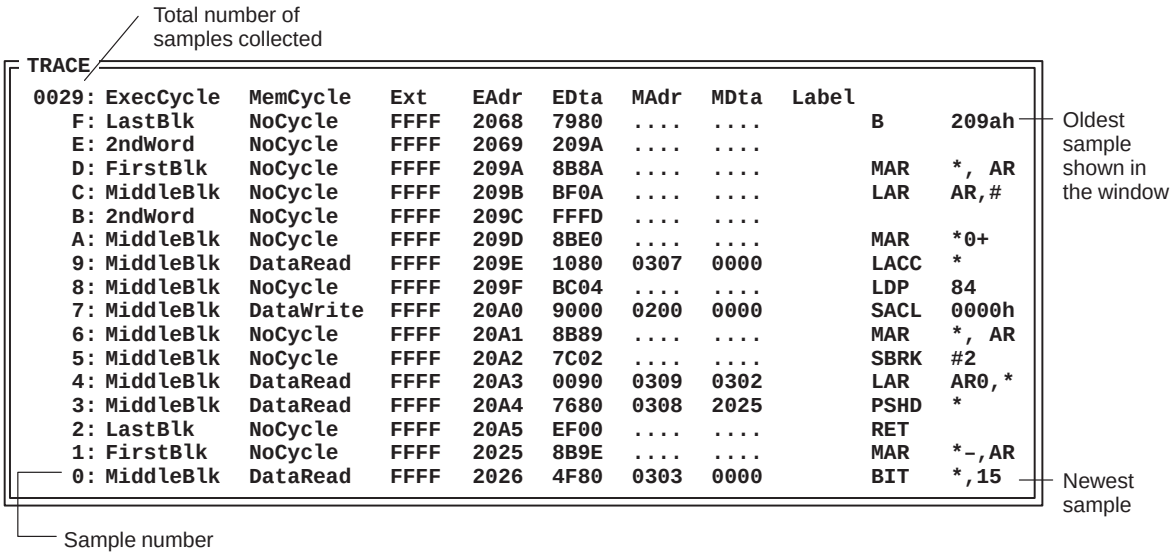
Notice that the first column of the top line in the TRACE window (the *header line*) has a sample number of 00029. This tells you that the trace buffer contains 0x0029 samples (labeled 0 to 28). This matches the number in the trace status indicator.

6.3 Moving Through the TRACE Window

Now that you know where the status indicators are and what they mean, it's time to look at the TRACE window.

> Use F6 to bring the TRACE window to the top and make it active.

The TRACE window should look similar to this:



The TRACE window displays trace samples from the oldest at the top to the newest at the bottom. The newest sample always has a trace sample number of 0. When the TRACE window is active, you can use the keys listed in Table 6–1 to move through the samples.

Table 6–1. Keys Used in the TRACE Window

To…	**Press**…
Scroll through the newest samples, one window length at a time	PAGE DOWN
Scroll through the oldest samples, one window length at a time	PAGE UP
Scroll through the most recent samples using the value that you set in the Goto trace sample dialog box (see page 6-7)	CONTROL PAGE DOWN
Scroll through the least recent samples using the value that you set in the Goto trace sample dialog box (see page 6-7)	CONTROL PAGE UP

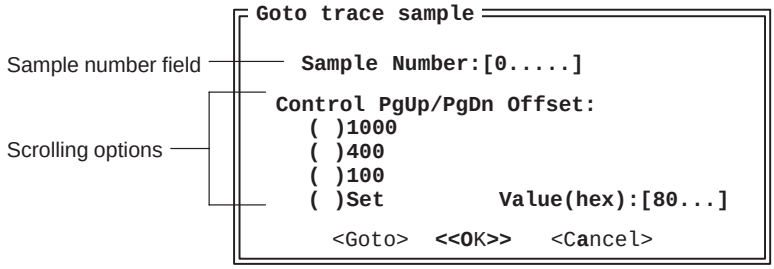
Table 6–1. Keys in the TRACE Window (Continued)

To...	Press...
Adjust the window's contents so that the newest sample is shown in the window	END
Adjust the window's contents so that the oldest sample is shown in the window	HOME
Move the cursor up, one line at a time	↑
Move the cursor down, one line at a time	↓

The BTT software allows you to move to a specific trace sample and to scroll through a specific number of samples.

To move to a specific trace sample or to scroll through a specific number of samples, select Goto from the Trace menu.

This displays the Goto trace sample dialog box:



To display a specific trace sample, enter the number of that sample in the Sample Number field and click on Goto.

To select a specific number of samples to scroll in the TRACE window, configure the scrolling options:

- ❑ Click next to 1000 to scroll 0x1000 samples at a time.
- ❑ Click next to 400 to scroll 0x0400 samples at a time.
- ❑ Click next to 100 to scroll 0x0100 samples at a time.
- ❑ Click next to Set to specify a hexadecimal value other than 1000, 400, or 100; specify the value in the Value field.

Once you configure the scrolling options, you can use the CONTROL PAGE DOWN and CONTROL PAGE UP sequences to move through the TRACE window.

6.4 Displaying Information About Trace Samples

The TRACE window allows you to display the following for each trace sample:

- ❑ The cycle type
- ❑ The data on the 16-bit program/execution address bus
- ❑ The data on the 16-bit program/execution data bus
- ❑ The data on the 16-bit data/memory address bus
- ❑ The data on the 16-bit data/memory data bus
- ❑ The data from the 16 external channels
- ❑ The program disassembly code
- ❑ A 48-bit hardware timestamp

The large amount of information stored for each trace sample can make your display difficult to read. You can hide some of the fields in the TRACE window.

To display or hide the fields in the TRACE window, follow these steps:

1) From the Trace menu, select Config.

   This displays the Trace configuration dialog box.

2) Select or deselect the options that you want to display or hide by clicking next to the option name.

3) When you are finished, click on OK.

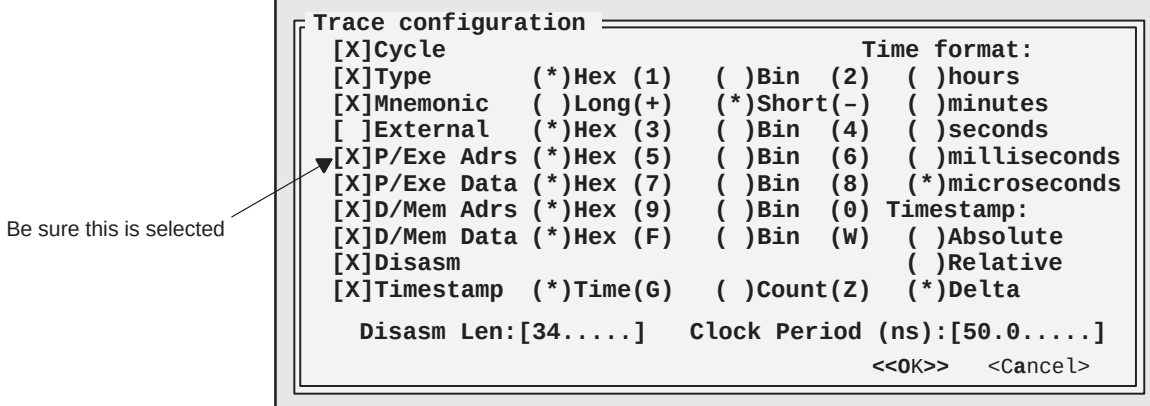
When you change the TRACE window configuration, the XDS522A saves the changes automatically and recalls the new configuration when you reinvoke the BTT software.

For more information about how to use the Trace configuration dialog box, see Section 13.6, *Displaying or Hiding Information About Trace Samples*, on page 13-16 or the *XDS522A Emulation System Online Help*.

Important! For this lesson, be sure that the execution address bus is showing; you should see the heading *EAdr* in the TRACE window. If you do not see the *EAdr* heading, modify your TRACE window configuration:

1) If you don't already have the Trace configuration dialog box displayed, select Config from the Trace menu.

2) Be sure that there is an X next to the *P/Exe Adrs* option.



3) Click on OK.

Look at the *EAdr* column in the TRACE window. With the TRACE window active, press HOME to see the oldest sample. Notice that the oldest sample has 0x205A as the execution address. This sample corresponds to the _call label that you defined as Event A.

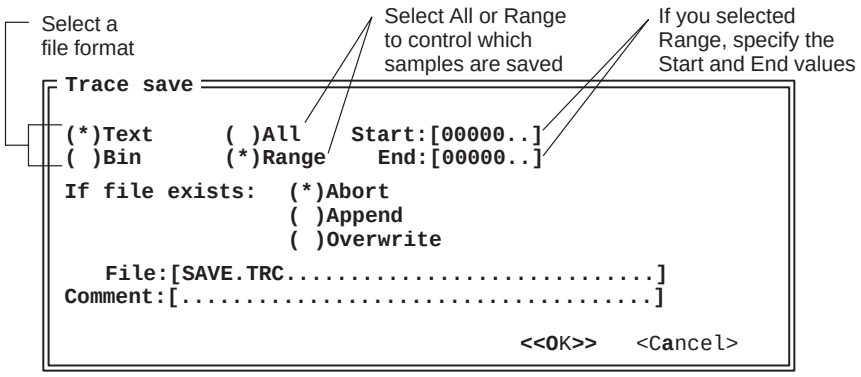
Press END to see the newest sample (shown at the bottom of the window). The *EAdr* column for the newest sample shows 0x2026, which is the execution address that you used to define Event B.

6.5 Saving the Contents of the Trace Buffer

You can save the contents of the trace buffer to a file. This is useful if you want to reload the trace buffer with the contents of previous trace or if you want to use the trace information in text format. Note that if you plan to reload the trace buffer contents, be sure to save the trace buffer in binary (Bin) format.

Save the current contents of the trace buffer by selecting Save from the Trace menu.

This displays the Trace save dialog box:



Save the trace buffer contents in text format, starting at trace number 1F and ending at trace number 0:

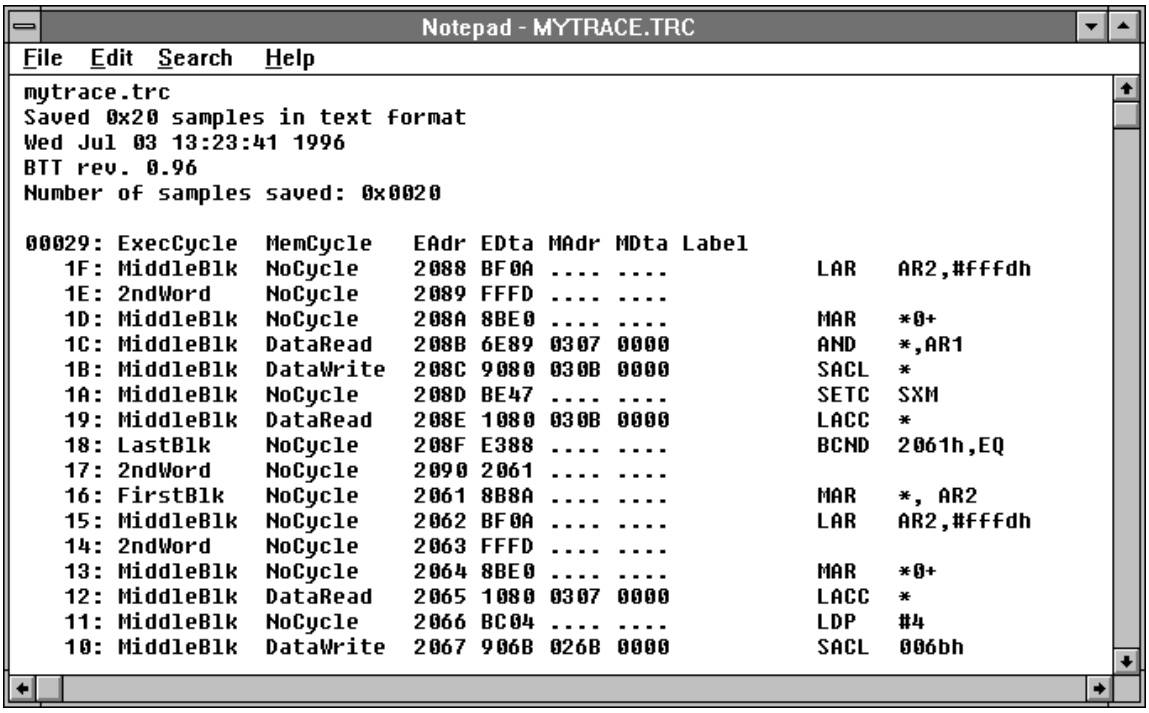
1) Select Text to save the file in text format.
2) Select Range to specify a range of trace samples.
3) In the Start field, enter 0001F.
4) In the End field, enter 00000.
5) Select Overwrite to write over the previous contents of the saved trace file.
6) Enter **mytrace.trc** as the File.
7) Enter the following in the 33-character Comment field:

   ```
   Saved 0x20 samples in text format
   ```

8) Click on OK.

This dismisses the dialog box and saves the trace buffer contents in a file called MYTRACE.TRC.

Because you saved the contents of the trace buffer as text, you can view the MYTRACE.TRC file in your Windows Notepad. As you can see, the XDS522A records the time and date when you saved the file, the BTT software revision, and the number of samples that were saved:



Notepad - MYTRACE.TRC

File Edit Search Help

```
mytrace.trc
Saved 0x20 samples in text format
Wed Jul 03 13:23:41 1996
BTT rev. 0.96
Number of samples saved: 0x0020

00029: ExecCycle  MemCycle   EAdr EDta MAdr MDta Label
   1F: MiddleBlk  NoCycle    2088 BF0A .... ....          LAR   AR2,#fffdh
   1E: 2ndWord    NoCycle    2089 FFFD .... ....
   1D: MiddleBlk  NoCycle    208A 8BE0 .... ....          MAR   *0+
   1C: MiddleBlk  DataRead   208B 6E89 0307 0000          AND   *,AR1
   1B: MiddleBlk  DataWrite  208C 9080 030B 0000          SACL  *
   1A: MiddleBlk  NoCycle    208D BE47 .... ....          SETC  SXM
   19: MiddleBlk  DataRead   208E 1080 030B 0000          LACC  *
   18: LastBlk    NoCycle    208F E388 .... ....          BCND  2061h,EQ
   17: 2ndWord    NoCycle    2090 2061 .... ....
   16: FirstBlk   NoCycle    2061 8B8A .... ....          MAR   *, AR2
   15: MiddleBlk  NoCycle    2062 BF0A .... ....          LAR   AR2,#fffdh
   14: 2ndWord    NoCycle    2063 FFFD .... ....
   13: MiddleBlk  NoCycle    2064 8BE0 .... ....          MAR   *0+
   12: MiddleBlk  DataRead   2065 1080 0307 0000          LACC  *
   11: MiddleBlk  NoCycle    2066 BC04 .... ....          LDP   #4
   10: MiddleBlk  DataWrite  2067 906B 026B 0000          SACL  006bh
```

Remember, when you save the contents of the trace buffer as text file, you cannot reload the trace buffer with that file. If you plan to reload the trace buffer with the contents of a saved file, save the the file in binary (Bin) format.

6.6 Summary

Congratulations! In this chapter, you learned how to do the following:

- ❏ Collect trace samples between two events
- ❏ Locate and read the program status indicator and the trace status indicator
- ❏ Move through the samples in the TRACE window
- ❏ Customize the display of information about each trace sample
- ❏ Save the contents of the trace buffer

If you feel uncomfortable in performing any of these tasks, take some time to review the material in this chapter.

If you are comfortable with these tasks, you're ready to move on to Chapter 7, *Using Advanced Tracing Features*.

Need a break?

If you need a break and want to close the BTT software and debugger, follow these directions:

- ❏ To close the BTT software, select Exit from the File menu.
- ❏ To close the debugger, from the COMMAND window, enter:

  quit ↵

To start the software again, follow the instructions in Section 4.4, *Starting a Debugging/Testing Session*, on page 4-8.

Chapter 7

Using Advanced Tracing Features

In Chapter 6, you learned how to trace on a particular event. The lessons in this chapter show you how to use the advanced tracing features of the XDS522A such as clearing the trace buffer and adding samples to the trace buffer.

Topic **Page**

7.1 Flushing or Accumulating Trace Samples

In all of the previous tracing lessons, the trace buffer was cleared each time you performed a new trace. This means that anything that had been displayed in the TRACE window when you started a new trace was overwritten. This tracing mode is called *trace flush*. This is the default tracing mode when you invoke the BTT software.

You can change the tracing mode to prevent previous trace samples from being overwritten. This tracing mode is called *trace accumulate*.

This lesson demonstrates the difference between the trace accumulate mode and the trace flush mode.

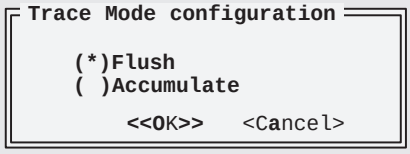
BTT

From the BTT menu, select Clear to reset the XDS522A configuration.

Select the trace flush mode:

1) From the Trace menu, select Mode.

   This displays the Trace Mode configuration dialog box:

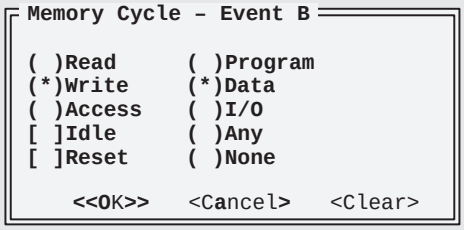


2) Click next to Flush, and click on OK.

In the BTT setup window, set up the XDS522A to trace on Events A and B:

EVENTS Box

1) Define Event A as any access to the address associated with the call() function:
   a) Click on the intersection of *EXEC Address* and Event A.
   b) In the EXEC Addr Ranges dialog box, click on the first Off listing to turn it to On.
   c) Under the Beg (beginning address) heading, click on 0000 and press the SPACE key.
   d) In the Get Symbol Address dialog box, enter **_call** as the Symbol, then click on OK.
   e) In the EXEC Addr Ranges dialog box, click on OK.
2) Using two qualifiers, define Event B as any data write on the memory address 0x030C:
   a) Click on the intersection of *Memory Cycle* and Event B. This displays the Memory Cycle dialog box.
   b) Click next to Write and next to Data. The dialog box should look like the following:



   c) Click on OK.
   d) Click on the intersection of *MEM Address* and Event B.
   e) In the Memory Addr Ranges dialog box, click on the first Off listing to turn it to On.
   f) Under the Beg (beginning address) heading, click on 0000 and enter the address **030C**.
   g) Be sure that the End (ending address) field has the value *030C*.
   h) Click on OK.

lesson continues on the next page →

TRACE Box

Set up tracing to start on the detection of Event A:

1) Be sure that immediate tracing is not enabled.
2) Click on the intersection of *Strt* and Event A.

ACTION Box

Set up the system to execute a hardware breakpoint at the occurrence of Event B by clicking on the intersection of *Break* and Event B.

Important! Don't forget to download the XDS522A configuration and, if necessary, enable the XDS522A.

Debug

Set up the debugger's on-chip analysis module to recognize and generate hardware breakpoints:

1) Be sure that the analysis module is enabled.
2) From the Analysis menu, select Break. This displays the Analysis break events dialog box.
3) Select EMU0 driven low to enable that option. Be sure that the Program Bus and EMU1 driven low options are not enabled.
4) Click on OK.

Now that the debugger is set up, you can run the program.

Reset the program entry point and run the program:

```
restart ↵
run ↵
```

BTT

When the XDS522A detects a data write on the memory address 0x030C, it causes the processor to stop running.

Use F6 to bring the TRACE window to the top and make it active.

- ❏ Press HOME to see the oldest trace sample. That sample contains the statement associated with the call() function.
- ❏ Press END to see the newest trace samples. Notice that there are 0x00D6 samples in the trace buffer.

Now that you see what is in the trace buffer, it's time to add to it.

Change to the trace accumulate mode:

1) From the Trace menu, select Mode.

2) In the Trace Mode configuration dialog box, select Accumulate and click on OK.

Debug

Reset the program entry point and run the program:

```
restart ↵
run ↵
```

BTT

Again, when the XDS522A detects a data write on memory address 0x030C, it causes the processor to stop running. Look at the TRACE window. There are twice as many samples in the buffer as there were in the previous lesson (0x01AC samples, to be exact). Because you selected the trace accumulate mode, the BTT software did not overwrite the previous contents of the trace buffer but added new samples to it.

7.2 Detecting When the Trace Buffer Is Full

The trace buffer can hold 32 768 (0x8000) samples. In this lesson, you'll use the *trace buffer full* condition to generate an event (in this case, a hardware breakpoint).

BTT

From the BTT menu, select Clear to reset the XDS522A configuration.

Important! Be sure that the tracing mode is set to Flush:

1) From the Trace menu, select Mode.
2) In the Trace Mode configuration dialog box, select Flush and click on OK.

In the BTT setup window, set up the XDS522A to detect when the trace buffer is full:

EVENTS Box

Define Event A as any access to the address associated with the xcall() function:

1) Click on the intersection of *EXEC Address* and Event A.
2) In the EXEC Addr Ranges dialog box, click on the first Off listing to turn it to On.
3) Under the Beg (beginning address) heading, click on 0000 and press the SPACE key.
4) In the Get Symbol Address dialog box, enter **_xcall** as the Symbol, then click on OK.
5) In the EXEC Addr Ranges dialog box, click on OK.

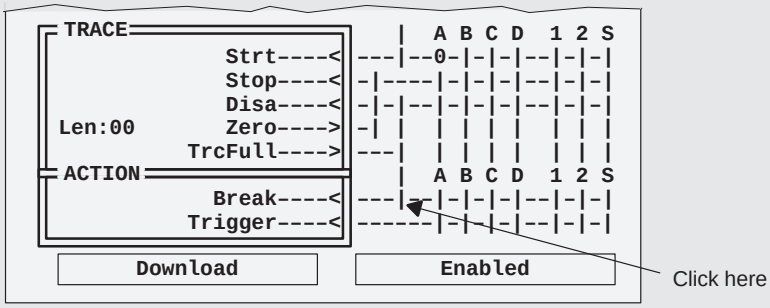
TRACE Box

Set up tracing to start on the detection of Event A:

1) Be sure that immediate tracing is not enabled.
2) Click on the intersection of *Strt* and Event A.

ACTION Box

Set up the system to execute a hardware breakpoint when the trace buffer is full by clicking on the intersection of *Break* and *TrcFull*:



Important! Don't forget to download the XDS522A configuration and, if necessary, enable the XDS522A.

Debug

Unless you have closed the debugger, you should still have the analysis interface enabled and the analysis break event set to EMU0 driven low. If you do not have the analysis interface set up, follow the debugger steps on page 7-4.

Reset the entry point and run the program:

```
restart ↵
run ↵
```

BTT

This time the processor stops running when the trace buffer is full.

Use F6 to bring the TRACE window to the top and make it active.

- ❏ Press HOME to see the oldest trace sample. That trace sample is numbered 7FFF.
- ❏ Press END to see the newest trace sample. That trace sample is numbered 0.

In this lesson, there are 0x8000 (or 32 768) samples stored in the trace buffer.

Notice that the trace status indicator is +0000A. Why isn't it +00000 (meaning that the trace buffer is full and contains 0x8000 samples)? Remember that there is a latency between the time the XDS522A detects a breakpoint condition, the time the XDS522A asserts the EMU0 signal, and the time the 'C2xx processor responds to the signal. Even though you told the XDS522A to assert a hardware breakpoint when the trace buffer is full, a few trace samples were overwritten because of the latency.

7.3 Collecting a Specific Number of Trace Samples

The XDS522A tracing feature allows you to collect a specific number of trace samples. The *Len* field in the TRACE box allows you to specify the number of additional samples that you want to collect after a certain condition occurs. You enter the length value as a hexadecimal number.

Once the XDS522A detects the start condition that you specify, it begins with the length value and internally decrements that value by 1 for each sample it collects. If the XDS522A detects the start condition again before the count reaches zero, it reloads the length value that you specified and starts decrementing again. Each time the start condition is detected, the XDS522A reloads the length value that you specified and starts decrementing.

Once the count reaches zero, the XDS522A stops or disables tracing.

- ❑ If the XDS522A detects only one start condition and does not reload the length value, the trace buffer contains Length + 1 trace samples.
- ❑ If the XDS522A detects the start condition two or more times before the count reaches zero, the trace buffer contains Length + 1 trace samples *plus* the samples collected before the length value was last reloaded.

The following lessons demonstrate what happens when a start condition occurs more than once and when a start condition occurs only once.

Collecting a large number of samples and reloading the length value

This lesson demonstrates how the XDS522A detects more than one occurrence of a start condition and reloads the length value that you specified.

BTT

From the BTT menu, select Clear to reset the XDS522A configuration.

In the BTT setup window, set up the XDS522A to collect a specific number of samples:

EVENTS Box

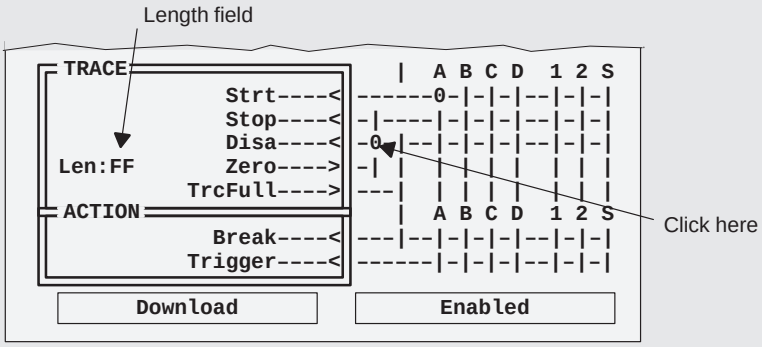
Define Event A as any access to the address associated with the xcall() function. If you need to review the specific steps, see the EVENTS box setup shown on page 7-6.

TRACE Box

Set up tracing to start on the detection of Event A and to start collecting 0xFF trace samples:

1) Set up tracing to start on the detection of Event A:
   a) Be sure that immediate tracing is not enabled.
   b) Click on the intersection of *Strt* and Event A.
2) Click on 00 in the Len field. Type **FF** as the length.
3) Click on the intersection of *Zero* and *Disa*.

   The Zero field allows you to specify what the XDS522A should do when it finishes collecting *Len* samples.



Important! Don't forget to download the XDS522A configuration and, if necessary, enable the XDS522A.

Debug

Reset the entry point and run the program:

```
restart ↵
run ↵
```

BTT

The processor will run indefinitely with this configuration. Recall that the program status indicator in the upper right corner of the BTT application window tells you the status of the processor.

lesson continues on the next page →

Look at the trace status indicator in the upper right corner of the BTT application window. Currently, the trace status indicator shows –0017A, indicating that the XDS522A collected 0x017A samples in the trace buffer. Because you set up the XDS522A to collect 0x00FF samples after Event A occurs, you might expect the trace status indicator to show –000FF or –00100. To understand the difference between the actual number of samples and the expected number samples, you need to look at the contents of the TRACE buffer.

To see the contents of the trace buffer, select Update from the Trace menu.

To bring the TRACE window to the top and make it active, press F6.

Notice that the first column of the header line in the TRACE window has a sample number of 0017A.

To see the first sample collected (the oldest sample), press HOME.

As you can see, the XDS522A detected the start condition (_xcall), started collecting trace samples, and used the length value to begin decrementing:

```
TRACE
0017A: EAdr  EDta  MAdr  MDta  Label
  179: 20A6  8AA0  030C  2083  _xcall   POPD   *+
  178: 20A7  80A0  030D  030A           SAR    AR0,*+
  177: 20A8  8180  030E  030E           SAR    AR1,*
  176: 20A9  B001  ....  ....           LAR    AR0,#1
  175: 20AA  00E8  030E  030E           LAR    AR0,*0+,AR0
  174: 20AB  BC04  ....  ....           LDP    #4
  173: 20AC  1A6E  026E  00AB           LACC   006eh,10
  172: 20AD  9080  030E  AC00           SACL   *
  171: 20AE  BE47  ....  ....           SETC   SXM
  170: 20AF  138A  030E  AC00           LACC   *,3,AR2
  16F: 20B0  BF0A  ....  ....           LAR    AR2,#fffdh
```

In the example, however, the XDS522A detected another occurrence of the _xcall.

To display the sample that contains the second occurrence of the _xcall, follow these steps:

1) From the Trace menu, select Goto.

   This displays the Goto trace sample dialog box:

```
Goto trace sample

   Sample Number:[0.....]

 Control PgUp/PgDn Offset:
   ( )1000
   ( )400
   ( )100
   ( )Set          Value(hex):[80...]

     <Goto>  <<OK>>   <Cancel>
```

2) In the Sample Number field, enter **FF**.

3) Click on Goto.

When the XDS522A detected this occurrence of the _xcall, it reloaded the length value that you specified and started decrementing again:

```
TRACE
0017A: EAdr  EDta  MAdr  MDta  Label
   FF: 20A6  8AA0  030C  2083  _xcall   POPD   *+
   FE: 20A7  80A0  030D  030A            SAR    AR0,*+
   FD: 20A8  8180  030E  030E            SAR    AR1,*
   FC: 20A9  B001  ....  ....            LAR    AR0,#1
   FB: 20AA  00E8  030E  030E            LAR    AR0,*0+,AR0
   FA: 20AB  BC04  ....  ....            LDP    #4
   F9: 20AC  1A6E  026E  00AB            LACC   006eh,10
   F8: 20AD  9080  030E  AC00            SACL   *
   F7: 20AE  BE47  ....  ....            SETC   SXM
   F6: 20AF  138A  030E  AC00            LACC   *,3,AR2
   F5: 20B0  BF0A  ....  ....            LAR    AR2,#fffdh
```

In this example, the XDS522A did not detect a third occurrence of _xcall before the count reached zero. Therefore, tracing was disabled when the XDS522A collected 0x00FF samples after the last occurrence of _xcall. You have 0x017A samples in the trace buffer because the XDS522A collected 0x007B samples before it detected the second occurrence of _xcall and reloaded the length value.

Collecting a small number of samples

This lesson demonstrates how to collect a small number of samples after an event occurs. After the specified number of samples are collected, the XDS522A disables tracing.

BTT

From the BTT menu, select Clear to reset the XDS522A configuration.

In the BTT setup window, set up the XDS522A to collect a specific number of samples:

EVENTS Box

Define Event A as any access to the address associated with the xcall() function. If you need to review the specific steps, see the EVENTS box set-up shown on page 7-6.

TRACE Box

Set up tracing to start on the detection of Event A and to start collecting 10 (0xA) trace samples:

1) Set up tracing to start on the detection of Event A:
   a) Be sure that immediate tracing is not enabled.
   b) Click on the intersection of *Strt* and Event A.
2) Click on 00 in the Len field. Type **0A** as the length.
3) Click on the intersection of *Zero* and *Disa*.

Important! Don't forget to download the XDS522A configuration and, if necessary, enable the XDS522A.

Debug

Reset the entry point and run the program:

```
restart ↵
run ↵
```

BTT

The processor will run indefinitely with this configuration.

To see the contents of the trace buffer, select Update from the Trace menu.

Tracing began at the _xcall label and continued for the next ten instructions:

TRACE

```
000B: EAdr  EDta  MAdr  MDta  Label
   A: 20A6  8AA0  030C  2083  _xcall   POPD   *+
   9: 20A7  80A0  030D  030A           SAR    AR0,*+
   8: 20A8  8180  030E  030E           SAR    AR1,*
   7: 20A9  B001  ....  ....           LAR    AR0,#1
   6: 20AA  00E8  030E  030E           LAR    AR0,*0+,AR0
   5: 20AB  BC04  ....  ....           LDP    #4
   4: 20AC  1A6E  026E  0000           LACC   006eh,10
   3: 20AD  9080  030E  0000           SACL   *
   2: 20AE  BE47  ....  ....           SETC   SXM
   1: 20AF  138A  030E  0000           LACC   *,3,AR2
   0: 20B0  BF0A  ....  ....           LAR    AR2,#fffdh
```

The XDS522A did not reload the length value, because it did not detect another occurrence of the start condition (_xcall) before the count reached zero.

7.4 Understanding the Difference Between Trace Stop and Trace Disable

In previous lessons, you used *Disa* in the TRACE box to "stop tracing". *Disa* (or disable) actually disables tracing. Once tracing is disabled, tracing does not reoccur, even if the XDS522A detects another occurrence of the start-tracing event. This is useful if you have a condition that is difficult to capture and you want to make sure that the trace samples for that condition are preserved after you obtain them.

The *Stop* tracing option tells the XDS522A to stop writing to the trace buffer until the start tracing event occurs again. If the XDS522A encounters another start-tracing event, it resumes tracing.

The following lesson demonstrates the difference between stopping the tracing activity and disabling the tracing activity. The *Collecting a small number of samples* lesson on page 7-12 showed you how to collect a small number of samples and *disable* tracing. In this lesson, you'll repeat that setup except that you tell the XDS522A to *stop* tracing instead of disable tracing.

BTT

From the BTT menu, select Clear to reset the XDS522A configuration.

In the BTT setup window, set up the XDS522A to collect a specific number of samples and stop tracing:

EVENTS Box

Define Event A as any access to the address associated with the xcall() function. If you need to review the specific steps, see the EVENTS box setup shown on page 7-6.

TRACE Box

Set up tracing to start on the detection of Event A and collect 10 (0xA) trace samples after Event A occurs:

1) Set up tracing to start on the detection of Event A.

2) Click on 00 in the Len field. Type **0A** as the length.

3) Click on the intersection of *Zero* and *Stop*.

Important! Don't forget to download the XDS522A configuration and, if necessary, enable the XDS522A.

Debug

Reset the entry point and run the program:

```
restart ↵
run ↵
```

BTT

Again, the processor runs indefinitely, and the XDS522A continues collecting trace samples.

To see the contents of the trace buffer, select Peek from the Trace menu.

Peek updates the Trace window with one screen of trace samples.

As you can see, each time the XDS522A detected the _xcall label, it started collecting trace samples for the next ten instructions. Because you didn't disable the trace buffer, each time the _xcall label was encountered, the XDS522A started collecting samples again.

TRACE

```
8000: EAdr  EDta  MAdr  MDta  Label
  18: 20AE  BE47  ....  ....          SETC  SXM
  17: 20AF  138A  030E  A800          LACC  *,3,AR2
  16: 20B0  BF0A  ....  ....          LAR   AR2,#fffdh
  15: 20A6  8AA0  030C  2083  _xcall  POPD  *+
  14: 20A7  80A0  030D  030A          SAR   AR0,*+
  13: 20A8  8180  030E  030E          SAR   AR1,*
  12: 20A9  B001  ....  ....          LAR   AR0,#1
  11: 20AA  00E8  030E  030E          LAR   AR0,*0+,AR0
  10: 20AB  BC04  ....  ....          LDP   #4
   F: 20AC  1A6E  026E  00A9          LACC  006eh,10
   E: 20AD  9080  030E  A400          SACL  *
   D: 20AE  BE47  ....  ....          SETC  SXM
   C: 20AF  138A  030E  A400          LACC  *,3,AR2
   B: 20B0  BF0A  ....  ....          LAR   AR2,#fffdh
   A: 20A6  8AA0  030C  2083  _xcall  POPD  *+
   9: 20A7  80A0  030D  030A          SAR   AR0,*
   8: 20A8  8180  030E  030E          SAR   AR1,*
   7: 20A9  B001  ....  ....          LAR   AR0,#1
   6: 20AA  00E8  030E  030E          LAR   AR0,*0+,AR0
   5: 20AB  BC04  ....  ....          LDP   #4
   4: 20AC  1A6E  026E  00A9          LACC  006eh,10
   3: 20AD  9080  030E  A400          SACL  *
   2: 20AE  BE47  ....  ....          SETC  SXM
   1: 20AF  138A  030E  A400          LACC  *,3,AR2
   0: 20B0  BF0A  ....  ....          LAR   AR2,#fffdh+
```

7.5 Summary

Congratulations! In this chapter, you learned how to do the following:

- ❏ Define an event using more than one qualifier
- ❏ Flush and accumulate trace samples
- ❏ Detect when the trace buffer is full
- ❏ Collect a specific number of trace samples
- ❏ Use trace stop and trace disable

If you feel uncomfortable in performing any of these tasks, take some time to review the material in this chapter.

If you are comfortable with these tasks, you're ready to move on to Chapter 8, *Searching and Filtering*.

Need a break?

If you need a break and want to close the BTT software and debugger, follow these directions:

- ❏ To close the BTT software, select Exit from the File menu.
- ❏ To close the debugger, from the COMMAND window, enter:

  quit ↵

To start the software again, follow the instructions in Section 4.4, *Starting a Debugging/Testing Session*, on page 4-8.

Chapter 8

Searching and Filtering

In Chapter 7, you learned how to use the advanced tracing features of the XDS522A. For example, you learned how to collect up to 32 768 (0x8000) samples in your trace buffer. When you have many samples in your trace buffer, you might want to locate a specific sample or view only specified samples in the TRACE window.

In this chapter, you will learn how to search for specific samples in the TRACE window and filter unwanted samples out of the TRACE window.

Topic **Page**

8.1 Searching for a Sample in the TRACE Window

The XDS522A emulation system has a search feature that you can use to search for specific samples in the TRACE window.

Starting the search from the top of the TRACE window

In this lesson, you will collect samples in the TRACE window. Then, you will search for a specific event by starting your search from the top of the TRACE window.

BTT

From the BTT menu, select Clear to reset the XDS522A configuration.

In the BTT setup window, set up the XDS522A to collect a specific number of samples:

TRACE Box

1) If it is not already set up to do so, set up the system to start tracing as soon as the sample program begins running (trace immediate).

2) Set up tracing to be disabled after the trace buffer collects 0x00FF samples. For information on how to set up tracing to be disabled after a specific number of samples are collected, see steps 2 and 3 on page 7-9.

Important! Don't forget to download the XDS522A configuration and enable the XDS522A.

Debug

Reset the entry point and run the program:

```
restart ↵
run ↵
```

BTT

To see the contents of the trace buffer, select Update from the Trace menu.

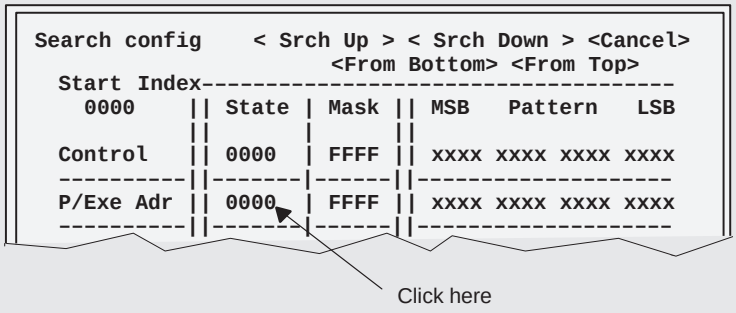
Set up the XDS522A to search for an event:

1) From the Trace menu, select Search.

   This displays the Search config dialog box, in which you set up specific search criteria.

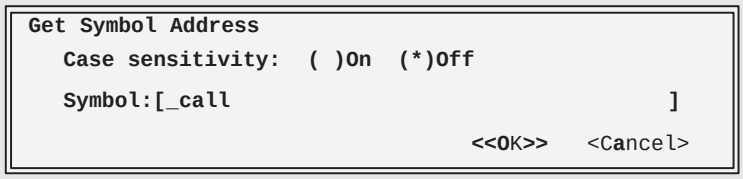
2) Specify the criteria to search for in the trace buffer:

   a) Click on 0000 in the *State* column of the *P/Exe Adr* row. This location is where you enter the execution address of the sample that you want to search for.

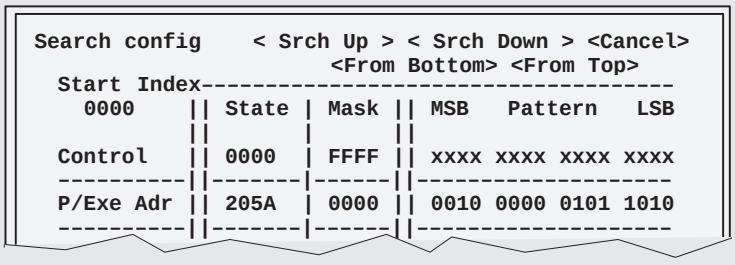


   b) Press the SPACE key. This displays the Get Symbol Address dialog box.

   c) In the Get Symbol Address dialog box, enter _**call** as the symbol name:



   d) Click on OK. This dismisses the Get Symbol Address dialog box. The Search config dialog box should look like this:



lesson continues on the next page →

Notice that the Mask column of the P/Exe Adr row is filled with zeros. You can use the Mask column to mask specific bits. By masking bits, you tell the system to ignore those bits when it searches for a bit pattern.

Notice that a bit pattern is automatically entered in the P/Exe Adr row. Instead of entering an address in the State column, you can enter the bits of that address in the Pattern column. For example, you could enter 0010 0000 0101 1010 in the Pattern column, and the State column would automatically be updated with 205A.

3) When you set up more than one search condition, the rows in the Search config dialog box are logically ANDed. You are using only one search condition for this lesson, so you must make sure that there are no other search conditions set up:

   - ❑ With the exception of the P/Exe Adr row, all of the rows in the State column should be filled with zeros.
   - ❑ With the exception of the P/Exe Adr row, all of the rows in the Mask column should be filled with the letter F.
   - ❑ With the exception of the P/Exe Adr row, all of the rows in the Pattern column should be filled with the letter x.

4) Click on *From Top.*

   This dismisses the dialog box and starts the search from the top of the TRACE window.

Notice that this message displays in the COMMAND window:

```
Trace entry found -- 000C5
```

Now, look at the TRACE window. It should look something like this:

TRACE

```
0100: EAdr  EDta  MAdr  MDta  Label
  C5: 205A  8AA0  0308  2025  _call:   POPD *+
  C4: 205B  80A0  0309  0302            SAR ARO, *+
  C3: 205C  8180  030A  030A            SAR AR1, *
```

The sample at the top of your window is the first occurrence of _call. It should be sample number C5.

Searching for other occurrences of the sample

Once you have your search criteria set up, you can quickly search for other occurrences of the same event.

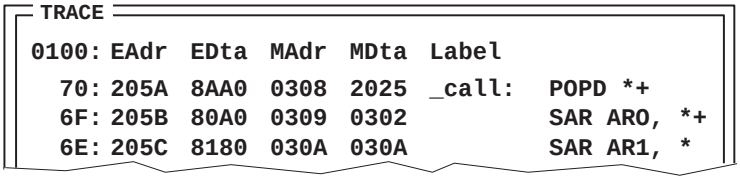
BTT

Continue searching for other occurrences of _call:

1) Select Search from the Trace menu again.

2) Click on *Srch Down*. This dismisses the Search dialog box and continues the search.

The next occurrence of _call displays at the top of your TRACE window:



Try This: You can also use the Next option in the Trace menu to see the next occurrence of _call.

You can continue searching for other occurrences of _call until the following message displays in the COMMAND window:

```
Trace entry not found
```

This message indicates that the search condition is either not in the TRACE window or not in the direction in which you are searching for it (in this case, down).

8.2 Filtering Samples Out of the TRACE Window

You can set up the TRACE window to show only the samples that you are interested in seeing. In other words, you can filter out the samples that you don't want to see. Use the filter feature when you want to see all of the occurrences of a specific sample (or set of samples) at a glance.

Filtering out more than one event

In this lesson, you will set up the filter to display only one execution data value in your TRACE window. Because several functions in the sample program (the main(), call(), and xcall() functions) begin with the same instruction, you can use the same execution data value to see where each one of those functions begins.

BTT

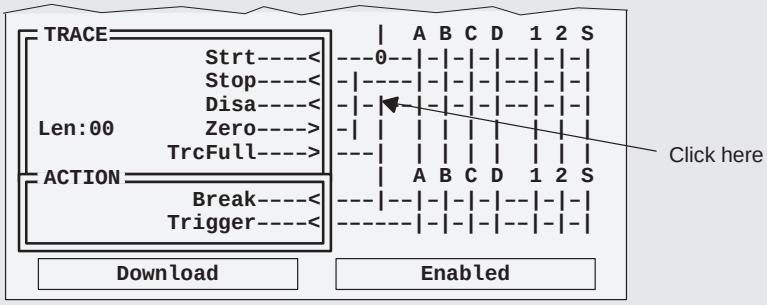
From the BTT menu, select Clear to reset the XDS522A configuration.

In your BTT setup window, do the following:

TRACE Box

1) Set up trace immediate.

2) Set up tracing to be disabled when the trace buffer is full by clicking on the intersection of *Disa* and *TrcFull.*



This creates a connection symbol at the intersection of *Disa* and *TrcFull*.

Important! Don't forget to download the XDS522A configuration and enable the XDS522A.

Debug

Reset the entry point and run the program:

```
restart ↵
run ↵
```

BTT

To see the contents of the trace buffer, select Update from the Trace menu.

Filter out the samples that you don't want to see:

1) Make sure that you start filtering samples from the top of the TRACE window. With the TRACE window active, press HOME so that the first sample that was collected is displayed. It should be labeled _c_init0.

2) From the Trace menu, select Filter.

   This displays the Filter config dialog box.

3) Set up the filter to show only the control bits in the TRACE window that you specify:

   a) Click on the zeros in the *State* column of the *P/Exe Dta* row.

      This location is where you will enter the execution control bits for the samples that you want to keep in the TRACE window.

```
Filter config           < Disable > < Enable > <Cancel>

          || State | Mask || MSB   Pattern   LSB
Control   || 0000  | FFFF || xxxx xxxx xxxx xxxx
P/Exe Adr || 0000  | FFFF || xxxx xxxx xxxx xxxx
P/Exe Dta || 0000  | FFFF || xxxx xxxx xxxx xxxx
```

Click here

   b) Enter **8AA0** as the execution data value.

      Notice that the Mask column of the P/Exe Dta row is filled with zeros and the bit pattern is automatically entered. The Filter config dialog box should now look like this:

```
Filter config           < Disable > < Enable > <Cancel>

          || State | Mask || MSB   Pattern   LSB
Control   || 0000  | FFFF || xxxx xxxx xxxx xxxx
P/Exe Adr || 0000  | FFFF || xxxx xxxx xxxx xxxx
P/Exe Dta || 8AA0  | 0000 || 1000 1010 1010 0000
```

lesson continues on the next page →

4) When you set up more than one filtering condition, the rows in the Filter config dialog box are logically ANDed. You are using only one filtering condition for this lesson, so you must make sure that there are no other filtering conditions set up:

   - ❑ With the exception of the P/Exe Dta row, all of the rows in the State column should be filled with zeros.
   - ❑ With the exception of the P/Exe Dta row, all of the rows in the Mask column should be filled with the letter F.
   - ❑ With the exception of the P/Exe Dta row, all of the rows in the Pattern column should be filled with the letter x.

5) Click on Enable.

   This dismisses the dialog box and executes the filter.

Look at your TRACE window. It should look something like this:

```
TRACE
8000: EAdr  EDta  MAdr  MDta  Label
7FF1: 2000  8AA0  0300  20E0  _main:   POPD *+
7FC5: 205A  8AA0  0308  2025  _call:   POPD *+
7F70: 205A  8AA0  0308  2025  _call:   POPD *+
7F1E: 205A  8AA0  0308  2025  _call:   POPD *+
7F70: 20A6  8AA0  030C  2083  _xcall:  POPD *+
```

Notice that the execution data value for all of the samples in the TRACE window is 0x8AA0. Because the main( ), call( ), and xcall( ) functions all begin with the POPD *+ instruction, you were able to use the same execution data value (0x8AA0) to see where each one of those functions begins.

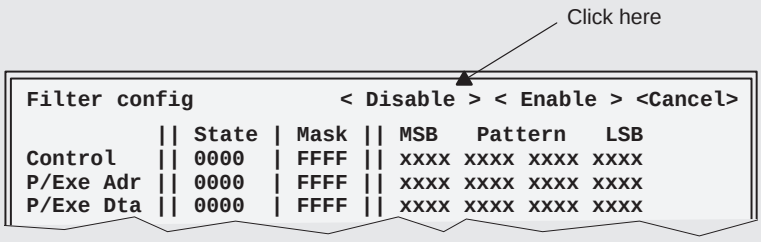
Where are the other samples?

The trace buffer didn't lose the other samples.

To see all of the samples again, disable the filter:

1) From the Trace menu, select Filter.

2) Click on Disable.



This dismisses the Filter config dialog box; the TRACE window displays all of the samples again.

8.3 Summary

Congratulations! In this chapter, you learned how to do the following:

- ❏ Search for an event
- ❏ Filter on more than one event
- ❏ Disable the filter

If you feel uncomfortable in performing any of these tasks, take some time to review the material in this chapter.

If you are comfortable with these tasks, you're ready to move on to Chapter 9, *Using the Timestamp*.

Need a break?

If you need a break and want to close the BTT software and debugger, follow these directions:

- ❏ To close the BTT software, select Exit from the File menu.
- ❏ To close the debugger, from the COMMAND window, enter:

  quit ↵

To start the software again, follow the instructions in Section 4.4, *Starting a Debugging/Testing Session,* on page 4-8.

Chapter 9

Using the Timestamp

When you want to know exactly when a trace sample was collected, when one sample was collected in relation to another sample, how long it took a loop to execute, or anything else related to the timing of events, use the timestamp option of the XDS522A emulation system.

This chapter describes how to set up the XDS522A to display the timestamp in the TRACE window and how to use the timestamp to show the relationship between two tasks.

Topic **Page**

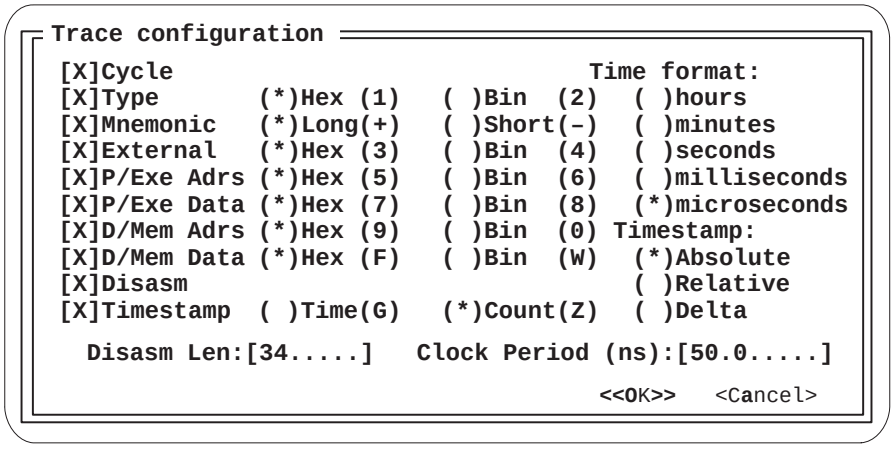
9.1 Displaying the Timestamp in the TRACE Window

You have many alternatives in setting up the timestamp display in the TRACE window. For example, you can display the timestamp in hours, minutes, seconds, milliseconds, or microseconds. The following lesson describes the timestamp options and tells you how to set up the timestamp for the next lesson, beginning on page 9-5.

BTT

To set up the timestamp for display, select Config from the Trace menu.

This displays the Trace configuration dialog box:



Note: Timestamp Values

The timestamp is a multiple of the system clock and is dependent on the clock module for your device. Because the timestamp clock intervals may be different from the system clock intervals, if you select *Count* as your timestamp option, your timestamp may look different from what you expect.

Follow these steps to display the timestamp in the TRACE window:

1) Deselect the Mnemonic and External options. (There should **not** be an X next to the *Mnemonic* or *External* option.)

   The TRACE window cannot display all of the information that you can set up at one time; you must limit the display when you use the timestamp option. You will not use the Mnemonic or External fields in the next lesson, so deselecting those options is appropriate here.

2) Select the *Timestamp* option. (There should be an X next to the *Timestamp* option.)

3) Select *Time* from the following options:

Time	Displays units of time
Count	Displays the number of clock cycles

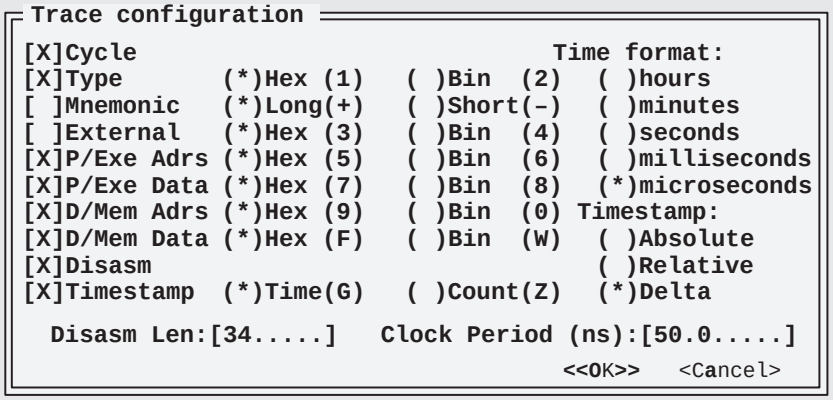
4) Because you chose Time in step 3, you need to choose the format in which you want the time displayed—hours, minutes, seconds, milliseconds, or microseconds.

   Select *microseconds*.

5) Select *Delta* from the following display modes:

Absolute	Shows the time relative to the first sample that you collected
Relative	Shows the time relative to a sample that you select in the TRACE window
Delta	Shows the time relative to the previous sample in the TRACE window

Your Trace configuration dialog box should look like this:



6) Click on OK. This dismisses the Trace configuration dialog box.

lesson continues on the next page →

Notice that the column headings in your TRACE window have automatically been updated with your new configuration:

Use this configuration for the next lesson.

9.2 Showing the Timing Relationship Between Two Tasks

This lesson illustrates how you can use the timestamp feature of the XDS522A to show the timing relationship between two function calls.

Important! Make sure that you have configured your TRACE window as described in the previous lesson.

BTT

From the BTT menu, select Clear to reset the XDS522A configuration.

In the BTT setup window, define the events that will be used in this lesson.

EVENTS Box

1) Define Event A as any access to the address associated with the call() function. For information on how to use the name of a function as an event qualifier, see steps 1 through 6 starting on page 5-4.

2) Define Event B as any access to the address associated with the xcall() function.

lesson continues on the next page →

In the BTT setup window, set up the XDS522A to start and stop tracing on the events that you defined:

TRACE Box

1) Set up tracing to start at the detection of Event A or Event B by clicking on the intersection of A and *Strt* and the intersection of B and *Strt*.

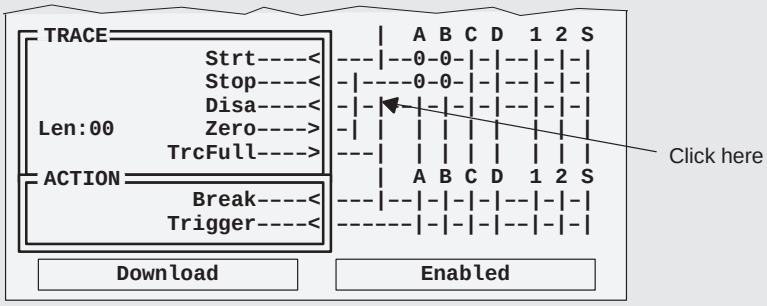
   You can set up tracing to start at the detection of either of two separate events, because all of the start events are ORed.

2) Set up tracing to stop at the detection of Event A or Event B by clicking on the intersection of A and *Stop* and the intersection of B and *Stop*.

   You can set up tracing to stop at the detection of either of two separate events because all of the stop events are ORed.

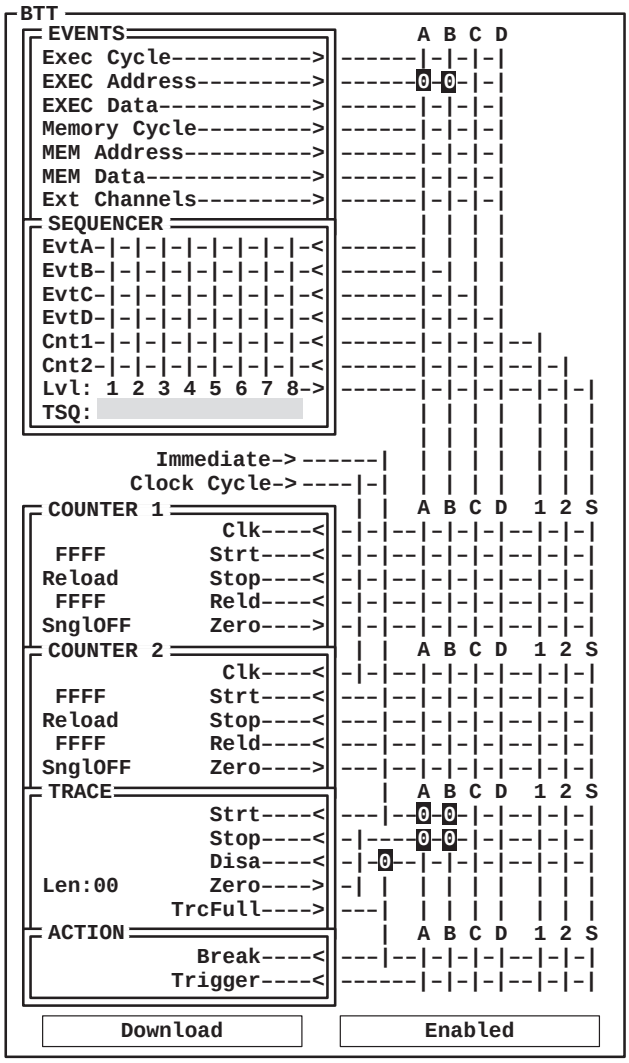
3) Make sure trace immediate is deselected.

4) Set up the system to disable tracing when the trace buffer is full by clicking on the intersection of *Disa* and *TrcFull*.



   This creates a connection symbol at the intersection of *Disa* and *TrcFull*.

Your BTT setup window should look like the following:



Important! Don't forget to download the XDS522A configuration and enable the XDS522A.

Debug

Reset the entry point and run the program:

```
restart ↵
run ↵
```

lesson continues on the next page →

BTT

To see the contents of the trace buffer, select Update from the Trace menu.

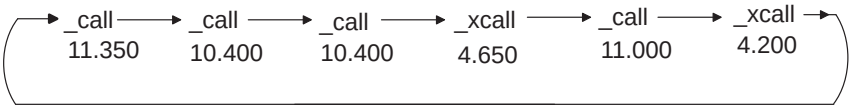
The bottom of your TRACE window should look like this:

TRACE

```
8000:     EAdr  EDta  MAdr  MDta   Label            Timestamp
   D: 51  205A  8AA0  0308  2025   _call   POPD *+  11.350
   C: 51  205A  8AA0  0308  2025   _call   POPD *+  10.400
   B: 51  205A  8AA0  0308  2025   _call   POPD *+  10.400
   A: 51  20A6  8AA0  030C  2083   _xcall  POPD *+   4.650
   9: 51  205A  8AA0  0308  2025   _call   POPD *+  11.000
   8: 51  20A6  8AA0  030C  2083   _xcall  POPD *+   4.200
   7: 51  205A  8AA0  0308  2025   _call   POPD *+  11.350
   6: 51  205A  8AA0  0308  2025   _call   POPD *+  10.400
   5: 51  205A  8AA0  0308  2025   _call   POPD *+  10.400
   4: 51  20A6  8AA0  030C  2083   _xcall  POPD *+   4.650
   3: 51  205A  8AA0  0308  2025   _call   POPD *+  11.000
   2: 51  20A6  8AA0  030C  2083   _xcall  POPD *+   4.200
   1: 51  205A  8AA0  0308  2025   _call   POPD *+  11.350
   0: 51  205A  8AA0  0308  2025   _call   POPD *+  10.400
```

Because of the configuration that you set up in the previous lesson, the timestamp displays the units of time in microseconds and in Delta mode. Delta mode is the difference in time from one sample to the next. For example, look at the timestamp for sample number A. The 4.650 means that the occurrence of _xcall listed as sample number A happened 4.650 microseconds after the occurrence of _call listed as sample number B.

By looking at the values listed in the timestamp column, you can determine the timing relationship between the call() and xcall() functions. Because the call() and xcall() functions are called within a for loop, the pattern of numbers that you see in the Timestamp column of your TRACE window directly correlates to that loop:

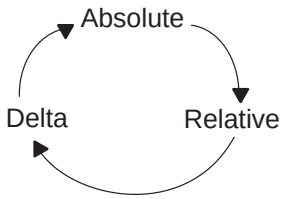


Once you know the pattern, you can look through the contents of your TRACE window to verify that the call() and xcall() functions are always called at the same intervals.

9.3 Toggling Through the Timestamp Display Modes

You can quickly change your timestamp display mode by toggling through the three display modes (Absolute, Relative, and Delta).

The order in which the timestamp toggles through the display modes is always the same. The mode that you see first depends on the configuration of the timestamp in the Trace configuration dialog box. If the timestamp configuration is set to Absolute, toggling the display mode changes the timestamp to Relative, then Delta, then back to Absolute again. If the configuration is set to Relative, toggling the button changes the timestamp to Delta, then Absolute, then back to Relative again:



Unless you have changed your configuration from the previous lesson, the timestamp is displayed in Delta mode. This lesson will show you how to change your display mode quickly without having to rerun your program.

BTT

To toggle to the next mode, click on the Time menu button or press F7 .

The timestamp changes from Delta mode to Absolute mode. Because of the configuration that you set up in the previous lesson, the timestamp still displays in microseconds. Your TRACE window should look something like this, although the numbers in your Timestamp column might be different:

```
TRACE
 8000:    EAdr  EDta  MAdr  MDta   Label             Timestamp
    D: 51 205A  8AA0  0308 2025   _call   POPD *+   283875.750
    C: 51 205A  8AA0  0308 2025   _call   POPD *+   283886.150
    B: 51 205A  8AA0  0308 2025   _call   POPD *+   283896.550
    A: 51 20A6  8AA0  030C 2083   _xcall  POPD *+   283901.200
    9: 51 205A  8AA0  0308 2025   _call   POPD *+   283912.200
    8: 51 20A6  8AA0  030C 2083   _xcall  POPD *+   283916.400
    7: 51 205A  8AA0  0308 2025   _call   POPD *+   283927.750
    6: 51 205A  8AA0  0308 2025   _call   POPD *+   283938.150
    5: 51 205A  8AA0  0308 2025   _call   POPD *+   283948.550
    4: 51 20A6  8AA0  030C 2083   _xcall  POPD *+   283953.200
    3: 51 205A  8AA0  0308 2025   _call   POPD *+   283964.200
    2: 51 20A6  8AA0  030C 2083   _xcall  POPD *+   283968.400
    1: 51 205A  8AA0  0308 2025   _call   POPD *+   283979.750
    0: 51 205A  8AA0  0308 2025   _call   POPD *+   283990.150
```

lesson continues on the next page →

Absolute mode displays the time relative to the first sample that you collected. Use absolute mode when you want to know how long execution from the trace-start event to another event.

Now, toggle to the next display mode.

To toggle to the next mode, click on the Time menu button or press F7.

The timestamp changes from Absolute mode to Relative mode. Your TRACE window should look something like this, although the numbers in your Timestamp column might be different:

TRACE

```
8000:    EAdr  EDta  MAdr  MDta   Label              Timestamp
   C: 51 205A  8AA0  0308  2025   _call   POPD *+     -114.400
   C: 51 205A  8AA0  0308  2025   _call   POPD *+     -104.000
   B: 51 205A  8AA0  0308  2025   _call   POPD *+      -93.600
   A: 51 20A6  8AA0  030C  2083   _xcall  POPD *+      -88.950
   9: 51 205A  8AA0  0308  2025   _call   POPD *+      -77.950
   8: 51 20A6  8AA0  030C  2083   _xcall  POPD *+      -73.750
   7: 51 205A  8AA0  0308  2025   _call   POPD *+      -62.400
   6: 51 205A  8AA0  0308  2025   _call   POPD *+      -52.000
   5: 51 205A  8AA0  0308  2025   _call   POPD *+      -41.600
   4: 51 20A6  8AA0  030C  2083   _xcall  POPD *+      -36.950
   3: 51 205A  8AA0  0308  2025   _call   POPD *+      -25.950
   2: 51 20A6  8AA0  030C  2083   _xcall  POPD *+      -21.750
   1: 51 205A  8AA0  0308  2025   _call   POPD *+      -10.400
   D: 51 205A  8AA0  0308  2025   _call   POPD *+        0.000
```

Relative mode shows the time relative to a sample that you select in the TRACE window. If you haven't selected a sample (by clicking anywhere on a sample), the sample at the bottom of your TRACE window acts as the selected sample.

You can use Relative mode to determine how long a section of code takes to execute.

To find out how long execution takes from the beginning of the first _call and the beginning of the third _call in the for loop, click on sample number 7.

This chooses sample number 7 as the first occurrence of _call to use as a reference point. Your TRACE window should look something like this:

TRACE

```
8000:    EAdr  EDta  MAdr  MDta   Label            Timestamp
   D: 51 205A  8AA0  0308 2025   _call   POPD *+   -52.000
   C: 51 205A  8AA0  0308 2025   _call   POPD *+   -41.600
   B: 51 205A  8AA0  0308 2025   _call   POPD *+   -31.200
   A: 51 20A6  8AA0  030C 2083   _xcall  POPD *+   -26.550
   9: 51 205A  8AA0  0308 2025   _call   POPD *+   -15.550
   8: 51 20A6  8AA0  030C 2083   _xcall  POPD *+   -11.350
   7: 51 205A  8AA0  0308 2025   _call   POPD *+     0.000
   6: 51 205A  8AA0  0308 2025   _call   POPD *+   +10.400
   5: 51 205A  8AA0  0308 2025   _call   POPD *+   +20.800
   4: 51 20A6  8AA0  030C 2083   _xcall  POPD *+   +25.450
   3: 51 205A  8AA0  0308 2025   _call   POPD *+   +36.450
   2: 51 20A6  8AA0  030C 2083   _xcall  POPD *+   +40.650
   1: 51 205A  8AA0  0308 2025   _call   POPD *+   +52.000
   0: 51 205A  8AA0  0308 2025   _call   POPD *+   +62.400
```

Your timestamp column is now a timeline with sample number 7 as the reference point:

- The timestamp for sample 7 is 0
- The timestamp for each of the samples above sample 7 is negative
- The timestamp for each of the samples below sample 7 is positive

The second occurrence of _call after sample number 7 is sample number 6. The third occurrence of _call is sample number 5. The timestamp for sample 5 shows that it takes 20.8 microseconds from the beginning of the first _call in the for loop to the beginning of the third _call in the for loop.

9.4 Changing Your Timestamp Configuration

You can change any of the timestamp options and view the updated results in the TRACE window without having to rerun the program. In this lesson, you will change the XDS522A from displaying the timestamp as units of time to displaying the timestamp as clock cycles.

BTT

Set up the XDS522A to display clock cycles:

1) From the Trace menu, select Config.

   This displays the Trace configuration dialog box.

2) Select *Count*.

3) Click on OK.

This dismisses the dialog box and updates the TRACE window. Your TRACE window should look something like this:

```
TRACE
8000:    EAdr  EDta  MAdr  MDta   Label           Timestamp
   D: 51 205A  8AA0  0308 2025   _call   POPD *+  -000000410
   C: 51 205A  8AA0  0308 2025   _call   POPD *+  -000000340
   B: 51 205A  8AA0  0308 2025   _call   POPD *+  -000000270
   A: 51 20A6  8AA0  030C 2083   _xcall  POPD *+  -000000213
   9: 51 205A  8AA0  0308 2025   _call   POPD *+  -000000137
   8: 51 20A6  8AA0  030C 2083   _xcall  POPD *+  -0000000E3
   7: 51 205A  8AA0  0308 2025   _call   POPD *+   000000000
   6: 51 205A  8AA0  0308 2025   _call   POPD *+  +0000000D0
   5: 51 205A  8AA0  0308 2025   _call   POPD *+  +0000001A0
   4: 51 20A6  8AA0  030C 2083   _xcall  POPD *+  +0000001FD
   3: 51 205A  8AA0  0308 2025   _call   POPD *+  +0000002D9
   2: 51 20A6  8AA0  030C 2083   _xcall  POPD *+  +00000032D
   1: 51 205A  8AA0  0308 2025   _call   POPD *+  +000000410
   0: 51 205A  8AA0  0308 2025   _call   POPD *+  +0000004E0
```

The TRACE window displays the number of clock cycles in the timestamp column, but the display is still in Relative mode.

9.5 Summary

Congratulations! In this chapter, you learned how to do the following:

- ❏ Display the timestamp in the TRACE window
- ❏ Use the timestamp to show the relationship between two tasks
- ❏ Toggle through the timestamp display modes
- ❏ Change your timestamp configuration

If you feel uncomfortable in performing any of these tasks, take some time to review the material in this chapter.

If you are comfortable with these tasks, you're ready to move on to Chapter 10, *Using the Counters*.

Need a break?

If you need a break and want to close the BTT software and debugger, follow these directions:

- ❏ To close the BTT software, select Exit from the File menu.
- ❏ To close the debugger, from the COMMAND window, enter:

  quit ↵

To start the software again, follow the instructions in Section 4.4, *Starting a Debugging/Testing Session*, on page 4-8.

Chapter 10

Using the Counters

The XDS522A emulation system provides you with two 16-bit counters, Counter 1 and Counter 2. Having two counters offers you the convenience of setting up the system several ways: you can use one counter at a time, both counters at once, or one counter to start the other counter.

This chapter describes how the counters work and includes lessons that illustrate how you can use the counters.

Topic **Page**

10.1 How the Counters Work

You can control every aspect of the counters: what they count, when they start, when they stop, what they use as their initial values, when they reload, and what happens after the counters finish. Table 10–1 describes how the counters work.

Table 10–1. How You Can Control the Counters

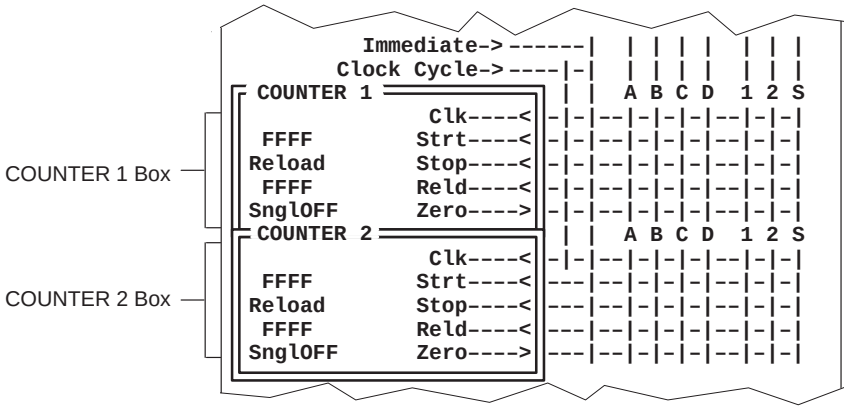
You can control…	**By specifying**…
What they count	❑ Clock cycles ❑ Events ❑ How many times a sequence completes ❑ How many times the other counter reaches 0
When they start	❑ Immediate ❑ Event A, B, C, or D ❑ When the other counter reaches 0 ❑ When the specified sequence completes
Their initial values	A value in the Initial Value field. The default initial value is 0xFFFF.
When they stop	❑ Event A, B, C, or D ❑ When the other counter reaches 0 ❑ When the specified sequence completes
When they reload	❑ Event A, B, C, or D ❑ When the other counter reaches 0 ❑ When the specified sequence completes
What happens next	❑ Advance to the next sequence level ❑ Decrement, start, stop, or reload the other counter ❑ Start, stop, or disable tracing ❑ Generate a hardware breakpoint ❑ Generate a trigger pulse

Note: The Precedence of Counter Inputs

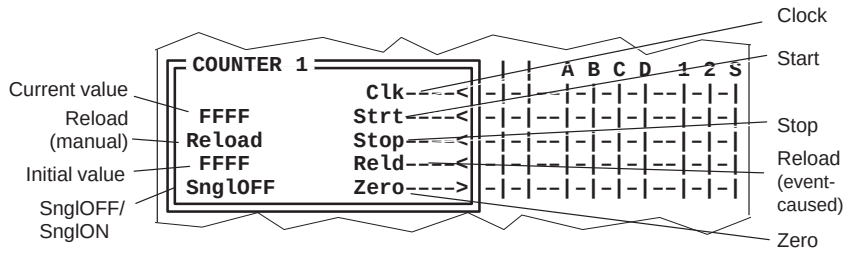
If counter inputs occur in the same cycle, this is the order of precedence:

1) Start
2) Reload
3) Clock
4) Stop

You set up the counters by putting all of their specifications into the COUNTER 1 and COUNTER 2 boxes of the BTT setup window:



The counter fields are shown in the COUNTER 1 box here:



The COUNTER 2 box has the same fields.

What's the difference between Reld and Reload?

Each of the counter boxes contains fields labeled *Reld* and *Reload.* These fields have different functions:

- You select the *Reld* field by clicking on the intersection of *Reld* and *A, B, C, D, 1, 2,* or *S.* If the *Reld* field is selected, the initial value of the counter is reloaded when one of the following occurs:
  - Specified event (A, B, C, or D)
  - Counter 1 or Counter 2 reaches 0
  - Sequence is met

  Then, the counter starts counting again.

- If you click on the *Reload* field, the current value of the counter is reset with the initial value. Typically, you do this *after* you have run your program and you want to reload the initial value into the counter. If you run your program again without clicking on *Reload*, the counter uses the number in the Current Value field as its initial value.

10.2 Counting a Specific Number of Events

In some situations, you may want to look at the contents of your trace buffer after an event has occurred a certain number of times. In those situations, you will want one of the counters to start counting at a particular value. By default, the counters start counting at 0xFFFF. However, you can specify an initial value from which the counter starts.

In this lesson, you will set up your own initial value for Counter 1 to use.

Debug

Important! Make sure that the sample program isn't running.

BTT

From the BTT menu, select Clear to reset the XDS522A configuration.

In your BTT setup window, define an event:

EVENTS Box

Define Event A as execution address 0x2086.

Set up the counter to start immediately, to be clocked by Event A, and to count 0x0004 occurrences of Event A:

COUNTER 1 Box

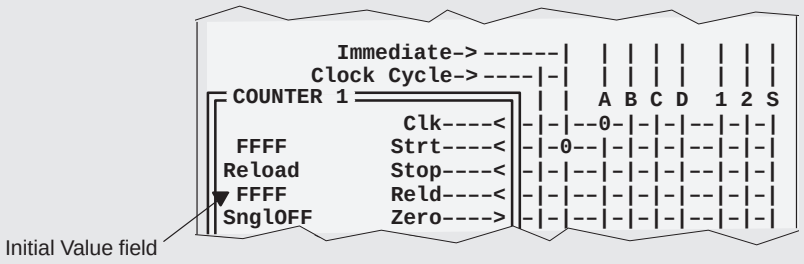
1) Set up the counter to be clocked by Event A by clicking on the intersection of *Clk* and A.
2) Set up your start event.

   Set up the counter to start counting as soon as the sample program starts running by clicking on the intersection of *Immediate* and *Strt*.

lesson continues on the next page →

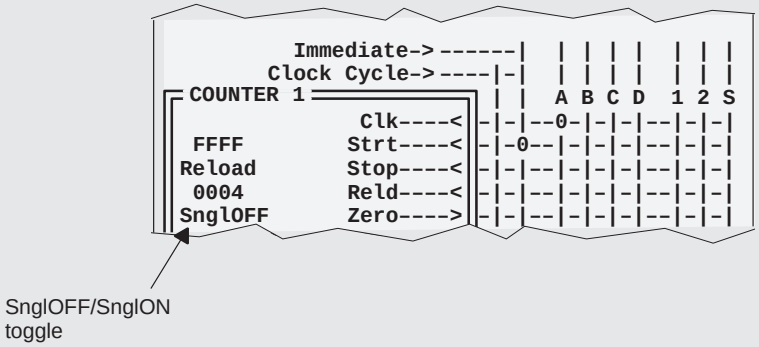
3) Set up your initial value. The initial value is the number that the counter counts down from.

   a) Click on the Initial Value field.



   b) Set up 0x0004 as your initial value by typing **0004** in the Initial Value field.

4) Set up Counter 1 as a single-shot counter by clicking once on the SnglOFF/SnglON toggle:



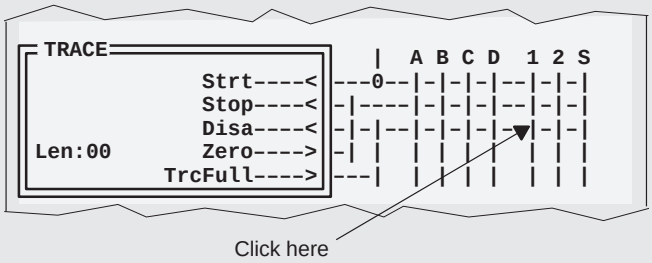
   The SnglOFF/SnglON toggle displays SnglON, meaning that Counter 1 is set up as a single-shot counter. A single-shot counter decrements from the initial value to 0 and then stops counting. You will learn more about single-shot counters when you go through the lesson on page 10-10.

Set up tracing to start immediately and to disable when the counter reaches 0:

TRACE Box

1) Set up trace immediate.

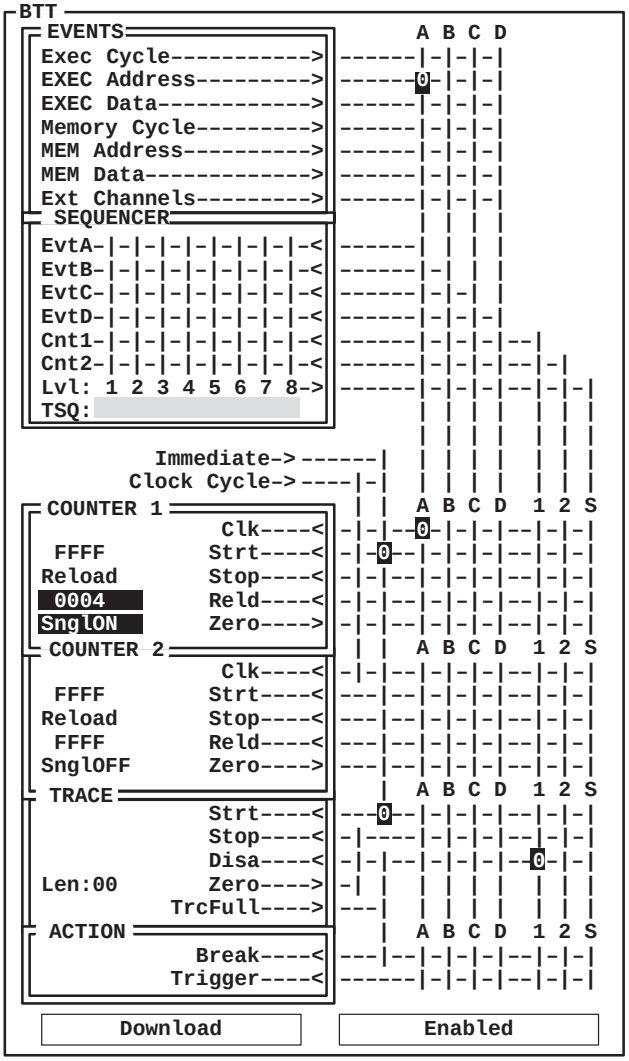
2) Set up Counter 1 to disable tracing when it reaches 0 by clicking on the intersection of 1 and *Disa*.



A connection symbol (0) appears at the intersection.

lesson continues on the next page →

Your BTT setup window should look like the following:



Important! Don't forget to download the XDS522A configuration and enable the XDS522A.

Debug

Reset the entry point and run the program:

```
restart ↵
run ↵
```

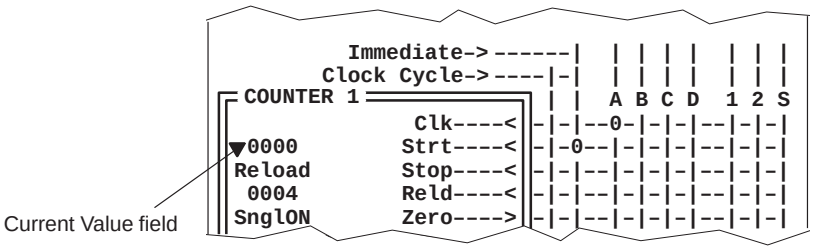
BTT

To see the contents of the trace buffer, select Update from the Trace menu.

The best way to verify your results is to set up the trace filter to display only the samples at execution address 0x2086. See Section 8.2, *Filtering Samples Out of the TRACE Window,* on page 8-6 for information on how to filter everything but that address out of the TRACE window. Once you set up your filter, your trace window should look something like this:

```
TRACE
0168: EAdr  EDta  MAdr  MDta  Label
 126: 2086  B903  ....  ....          LACL #3
  D1: 2086  B903  ....  ....          LACL #3
  7F: 2086  B903  ....  ....          LACL #3
   1: 2086  B903  ....  ....          LACL #3
```

There are four occurrences of Event A in the TRACE window. After those four samples were collected, tracing was disabled. Look at the Current Value field of the COUNTER 1 box:



The current value of Counter 1 is 0, which means that Counter 1 stopped after counting four occurrences of Event A. Because Counter 1 stopped, tracing was disabled.

10.3 Using a Single-Shot Counter

In the lesson in Section 10.2 on page 10-5, the counter functioned as a single-shot counter. Single-shot counters decrement to 0 and stop.

Sometimes you may want one of the counters to reload itself with the initial value and continue counting after it decrements to 0. If you turn off the single-shot feature, the counter decrements to 0, outputs 0, reloads the initial value, and continues decrementing.

The following lesson sets up both counters to start at the same time, to be clocked by the same event, and to count the same number of events. So that you can see the difference between using the single-shot feature and not using the feature, you will turn single-shot on for Counter 1 and leave single-shot off for Counter 2 in this lesson.

Debug

Important! Make sure that the sample program isn't running.

BTT

From the BTT menu, select Clear to reset the XDS522A configuration.

In your BTT setup window, define an event:

EVENTS Box

Set up Event A as execution address 0x20D2.

Set up Counter 1 and Counter 2 to function in exactly the same way except for the single-shot feature:

COUNTER 1 Box

1) Set up the counter to count clock cycles by clicking on the intersection of *Clock Cycle* and *Clk*.

2) Set up your start event.

   Set up the counter to start counting on Event A by clicking on the intersection of A and *Strt*.

3) Set up 0x0100 as your initial value.

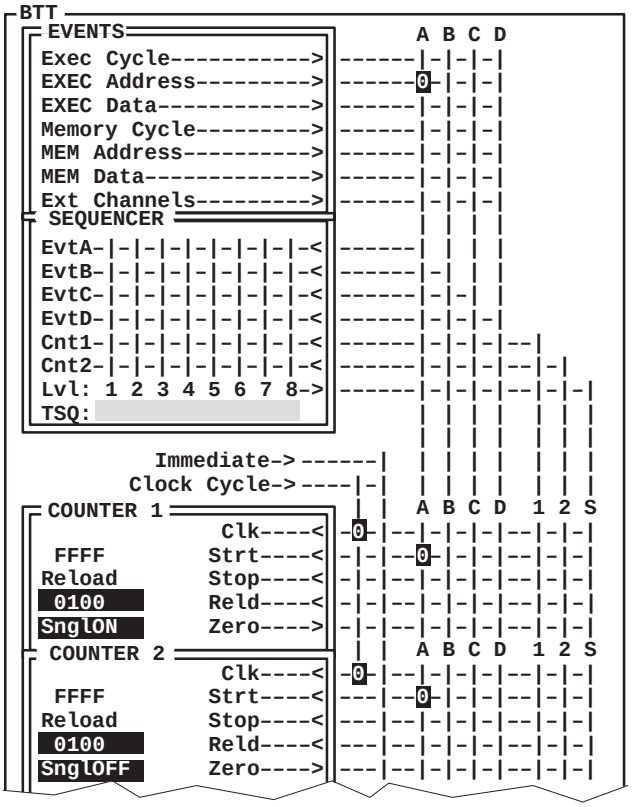
4) Set up the counter to use the single-shot feature by clicking once on the SnglOFF/SnglON toggle to turn the single-shot feature on.

COUNTER 2 Box

Set up Counter 2 the same way that you set up Counter 1, but skip step 4.

lesson continues on the next page →

Your BTT setup window should look like the following:



Important! Don't forget to download the XDS522A configuration and enable the XDS522A.

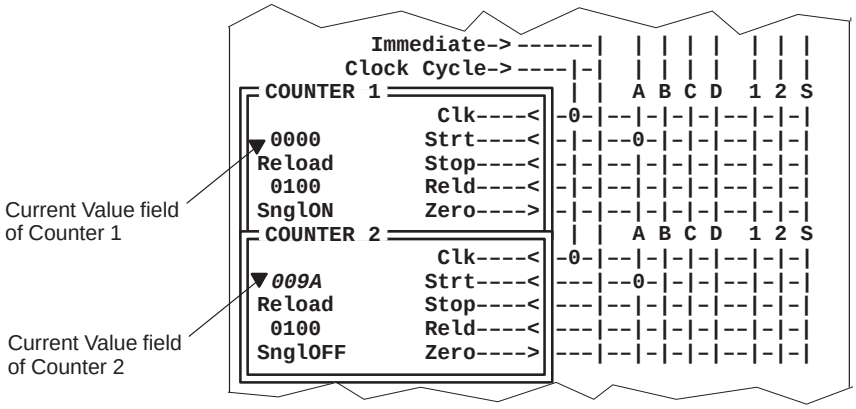
Debug

Reset the entry point and run the program:

```
restart ↵
run ↵
```

BTT

Look at the Current Value fields in the COUNTER boxes. They should look something like this:



You'll notice that the current value of Counter 1 is 0. Because you used the single-shot feature in Counter 1, the counter decremented from 0x0100 to 0 and stopped.

You'll also notice that the Current Value field of Counter 2 keeps changing and does not stop. As long as the sample program is running, the current value of Counter 2 will continue to change. On closer inspection, you will see that Counter 2's Current Value field repeatedly displays a series of numbers between 0x0 and 0x0100. Because the single-shot feature is turned off in Counter 2, the Current Value field keeps decrementing and reloading itself with the initial value; every time that Event A (the start event) occurs, Counter 2 counts down from 0x0100 to 0x0.

10.4 Counting More Than One Event

Because the XDS522A has two counters, you can use one of the counters to decrement, start, reload, or stop the other counter. One way this is useful is if you want to count more than one event. In this lesson, you will set up Counter 1 to count one event and then start Counter 2, which will count another event.

Debug

Important! Make sure that the sample program isn't running.

BTT

From the BTT menu, select Clear to reset the XDS522A configuration.

In your BTT setup window, define Event A and Event B:

EVENTS Box

1) Define Event A as any access to the address associated with the call() function.

2) Define Event B as any access to the address associated with the xcall() function.

Set up Counter 1 to function as a single-shot counter that starts immediately, is clocked by Event A, and counts 0x0004 occurrences of Event A.

COUNTER 1 Box

1) Set up Counter 1 to be clocked by Event A by clicking on the intersection of A and *Clk*.

2) Set up your start event.

   Set up the counter to start counting as soon as the sample program begins running by clicking on the intersection of *Immediate* and *Strt*.

3) Set up your initial value by clicking on the Initial Value field and enter **0004** as your initial value.

4) Set up the counter to use the single-shot feature by clicking once on the SnglOFF/SnglON toggle to turn the single-shot feature on.

Set up Counter 2 to function as a single-shot counter that starts when Counter 1 reaches 0, is clocked by Event B, and counts 0x0002 occurrences of Event B.

COUNTER 2 Box

1) Set up Counter 2 to be clocked by Event B by clicking on the intersection of B and *Clk*.

2) Set up your start event.

   Set up the counter to start counting as soon as Counter 1 reaches 0 by clicking on the intersection of 1 and *Strt*.

3) Set up your initial value by clicking on the Initial Value field and enter **0002** as your initial value.

4) Set up the counter to use the single-shot feature by clicking once on the SnglOFF/SnglON toggle to turn the single-shot feature on.

Set up tracing to start immediately and to disable when Counter 2 reaches 0:

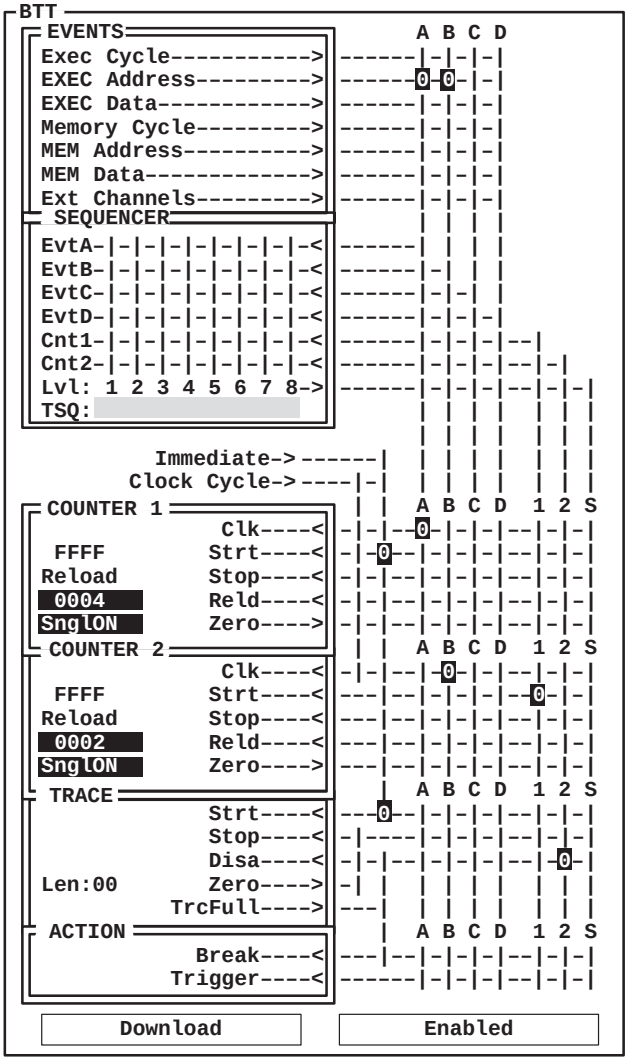
TRACE Box

1) Set up trace immediate.

2) Set up Counter 2 to disable tracing by clicking on the intersection of 2 and *Disa*.

lesson continues on the next page →

Your BTT setup window should look like the following:



Important! Don't forget to download the XDS522A configuration and enable the XDS522A.

Debug

Reset the entry point and run the program:

```
restart ↵
run ↵
```

BTT

To see the contents of the trace buffer, select Update from the Trace menu.

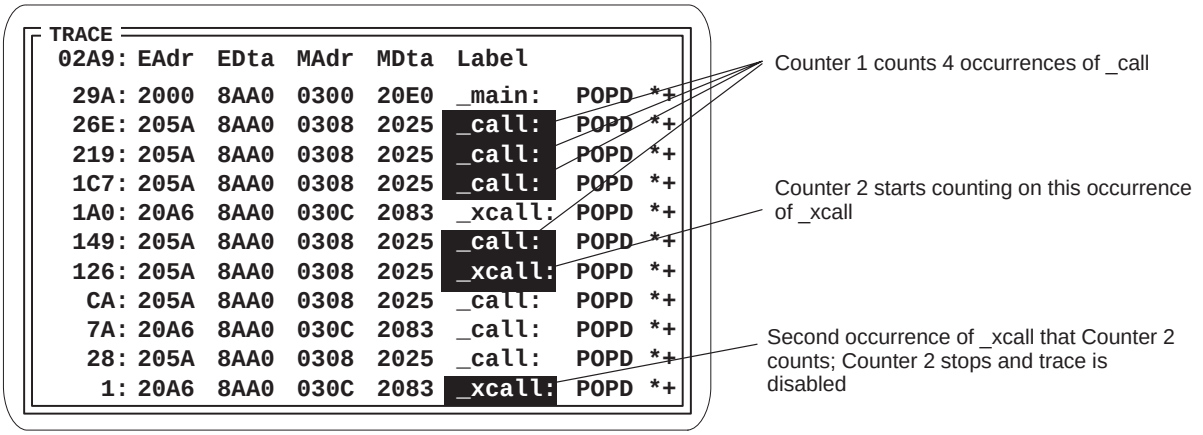
Here is what happened:

1) Counter 1 immediately started counting occurrences of _call and stopped counting when it encountered four occurrences.

2) When Counter 1 stopped, it started Counter 2. Counter 2 started counting the occurrences of _xcall. It counted two occurrences of _xcall and stopped.

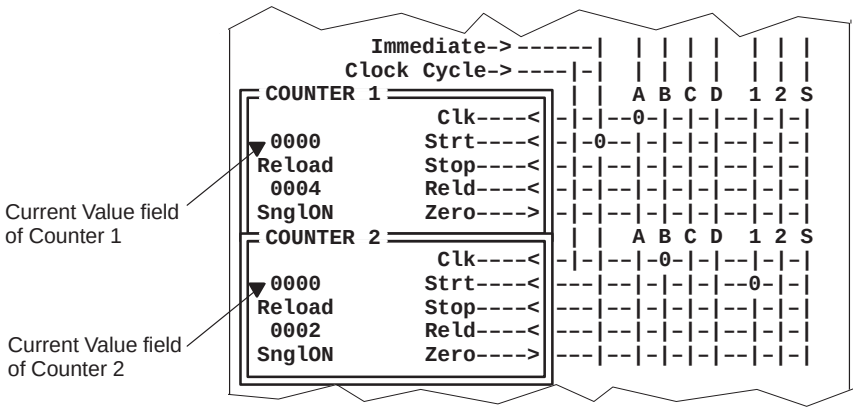
3) When Counter 2 stopped, tracing was disabled.

The best way to verify this is to set up the trace filter to display only the samples that are labeled _call and _xcall and to look at the counters' current values. See Section 8.2, *Filtering Samples Out of the TRACE Window,* on page 8-6 for more information. Once you set up your filter, your TRACE window should look something like this:



lesson continues on the next page →

Look at the counters' Current Value fields. Counter 1's Current Value field is at 0, because it counted four occurrences of _call and stopped. Counter 2's Current Value field is 0, because it counted two occurrences of _xcall and stopped.



10.5 Summary

Congratulations! In this chapter, you learned how to do the following:

- ❏ Count a specific number of events
- ❏ Use a single-shot counter
- ❏ Reload a counter
- ❏ Use Counter 1 as a catalyst for Counter 2

If you feel uncomfortable in performing any of these tasks, take some time to review the material in this chapter.

If you are comfortable with these tasks, you're ready to move on to Chapter 11, *Using the Sequencer.*

Need a break?

If you need a break and want to close the BTT software and debugger, follow these directions:

- ❏ To close the BTT software, select Exit from the File menu.
- ❏ To close the debugger, from the COMMAND window, enter:

  `quit` ↵

To start the software again, follow the instructions in Section 4.4, *Starting a Debugging/Testing Session,* on page 4-8.

Chapter 11

Using the Sequencer

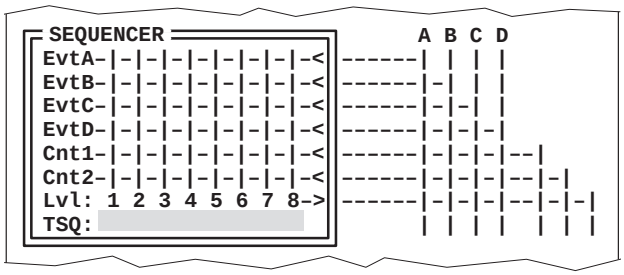
Sometimes you may want to detect when a specified series of events occurs. When the order in which events occur matters, use the sequencer. This chapter describes the sequencer and how you can use it to detect a sequence of events.

Part II

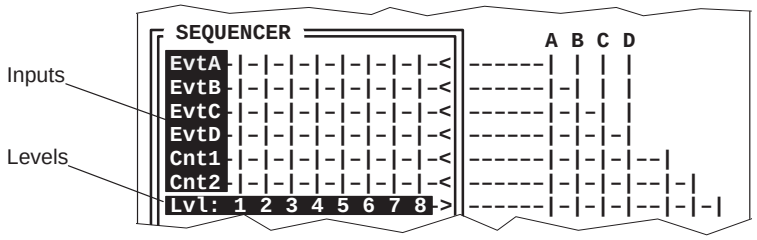
Topic **Page**

11.1 How the Sequencer Works

To set up a sequence of events, use the SEQUENCER box in the BTT setup window:



You can use six different inputs to set up your sequence: Events A, B, C, and D and Counters 1 and 2. You specify the sequence by assigning a level to each input. The first input in the sequence is Level 1, the second is Level 2, and so on. The system starts by searching for the qualifier that you have set up at Level 1. This is called your Level-1 event. Then, it searches for your Level-2 event. The system continues searching until all of the levels that you have defined have been detected. Then, the system begins looking for the Level-1 event again. You can have up to eight levels in your sequence.



When the sequence is detected, you can do any of the following:

- ❑ Start, stop, decrement, or reload a counter
- ❑ Start, stop, or disable tracing
- ❑ Generate a hardware breakpoint
- ❑ Generate a trigger pulse

11.2 Detecting a Sequence of Events

In this lesson, you'll set up the system to look for a sequence of events and to disable tracing after the sequence occurs. The sequence that you will be searching for is the execution of the function call(), the execution of the function xcall(), and another execution of the function call().

BTT

From the BTT menu, select Clear to reset the XDS522A configuration.

In the BTT setup window, define the events that will be used in the sequence:

EVENTS Box

1) Define Event A as any access to the address associated with the call() function. For information on how to use the name of a function as an event qualifier, see steps 1 through 6 starting on page 5-4.

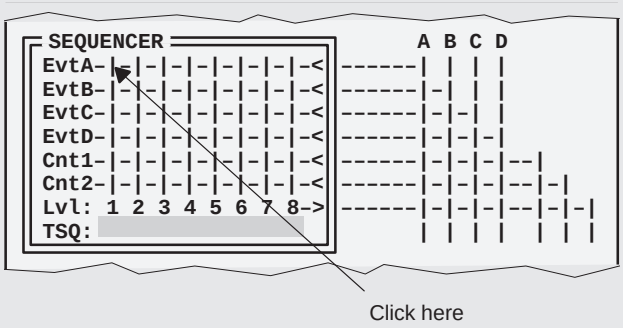
2) Define Event B as any access to the address associated with the xcall() function.

lesson continues on the next page →

Set up the system to look for a sequence of events:

SEQUENCER Box

1) Set up Event A as the first event in the sequence, the Level-1 event, by clicking on the intersection of *Evt A* and 1.



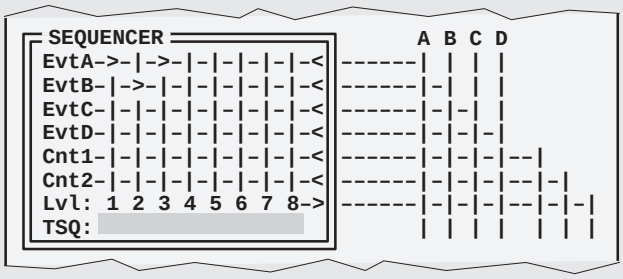
A sequence connection symbol (>) appears at the intersection.

Important! If you accidentally click on the intersection of *Evt A* and 1 twice, an R appears. If this happens, click on the intersection two more times to replace the R with the sequence connection symbol. (The R symbol is explained and used in Section 11.3.)

2) Set up Event B as the second event in the sequence by clicking on the intersection of *Evt B* and 2.

3) Even though you already have it set up as the first event in the sequence, you can also set up Event A as the third event in the sequence by clicking on the intersection of *Evt A* and 3.

The sequencer is now set up to search for Event A, followed by Event B, followed by Event A:

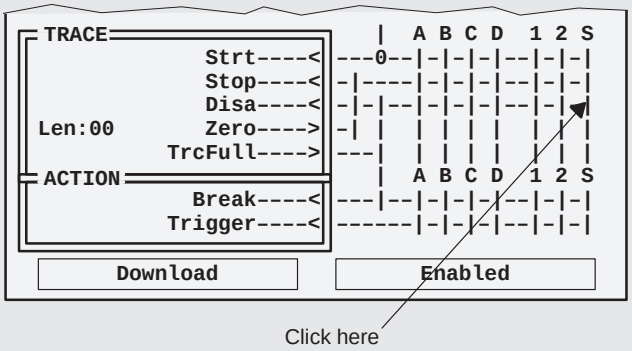


Set up tracing to start immediately and to disable when the sequence is detected:

TRACE Box

1) Set up trace immediate.

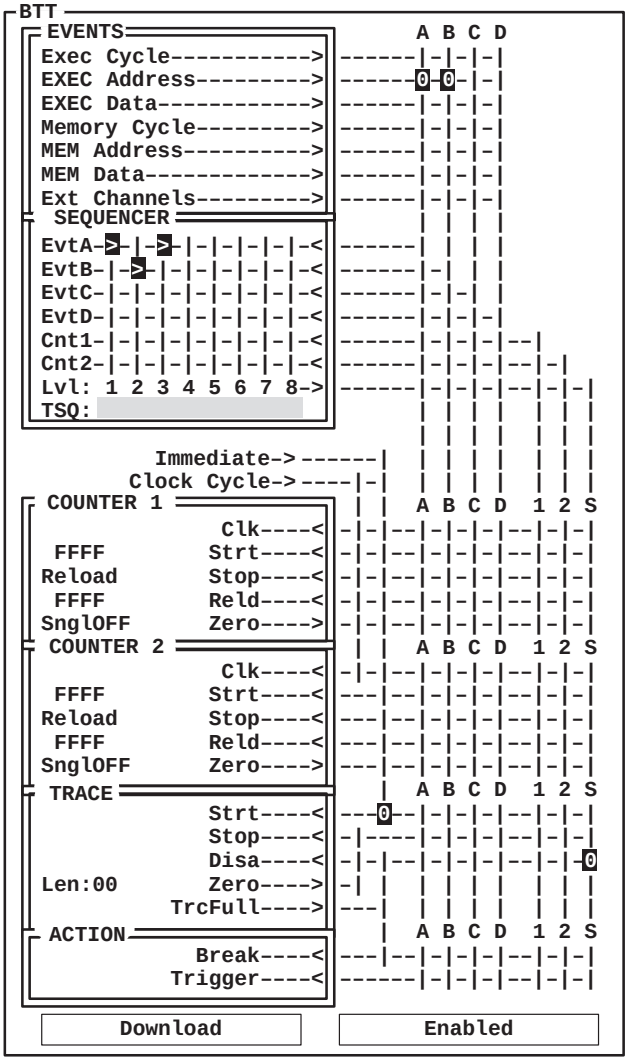
2) Disable tracing when the sequencer has detected the defined sequence by clicking on the intersection of *Disa* and S. (The S is for sequencer.)



A connection symbol (0) appears at the intersection.

lesson continues on the next page →

Your BTT setup window should look like the following:



Important! Don't forget to download the XDS522A configuration and enable the XDS522A.

Debug

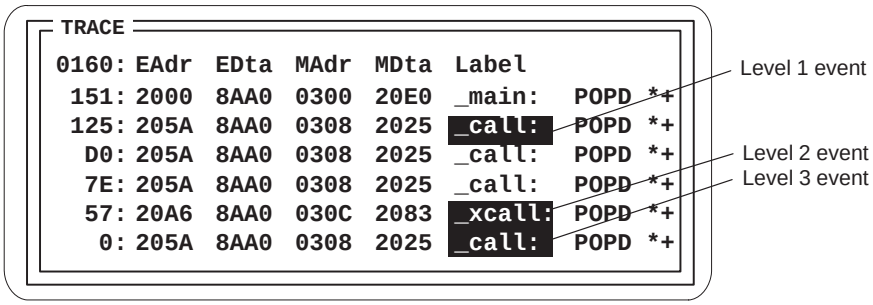
Reset the entry point and run the program:

```
restart ↵
run ↵
```

BTT

To see the contents of the trace buffer, select Update from the Trace menu.

The best way to verify your results is to set up the trace filter to display only the samples that are labeled _call and _xcall in the TRACE window. See Section 8.2, *Filtering Samples Out of the TRACE Window,* on page 8-6 for more information. (If your filter is still set up from the lesson starting on page 8-6, select Filter from the Trace menu. Then, click on Enable in the Filter config dialog box to reenable the filter.) Once you set up your filter, your TRACE window should look something like this:



The first _call after _main was the Level-1 event. Because the sequencer searches for a particular sequence of events without regard to what happens between the defined events, the second and third occurrences of _call were ignored. The next label that you see is _xcall. It was the Level-2 event. The _call at sample number 0 was the Level-3 event—the last event in the sequence. Once the sequence was detected, tracing was disabled.

11.3 Resetting the Sequencer on an Event

You may want to set up the sequencer to reset if a specific event occurs. You can use any inputs (Events A–D and the counters) to reset the sequencer.

In this lesson, the system will be looking for two occurrences of the call() function followed by the xcall() function. It sounds simple. However, if a third occurrence of call() occurs between the second call() and xcall(), the sequencer will reset.

BTT

From the BTT menu, select Clear to reset the XDS522A configuration.

In your BTT setup window, define the events that will be used in the sequence:

EVENTS Box

1) Define Event A as any access to the address associated with the call() function.

2) Define Event B as any access to the address associated with the xcall() function.

Set up the system to look for a sequence of events:

SEQUENCER Box

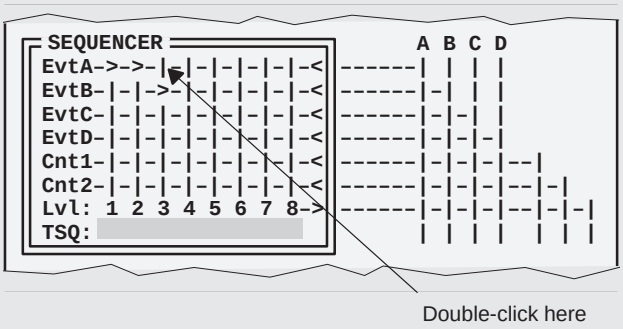
1) Set up Event A as the first and second events in the sequence.
2) Set up Event B as the third event in the sequence.

These events advance the sequencer to the next level. So far, those are the only types of sequencer events that you have set up. Now, you will set up a different type of sequencer event—a reset event.

Set up the sequencer to reset if a specified event occurs before the sequence is detected:

SEQUENCER Box

Even though you already have set up Event A as the first and second events in the sequence, you can also set it up as the reset event by double-clicking on the intersection of *Evt A* and 3.



A reset connection symbol (R) appears at the intersection.

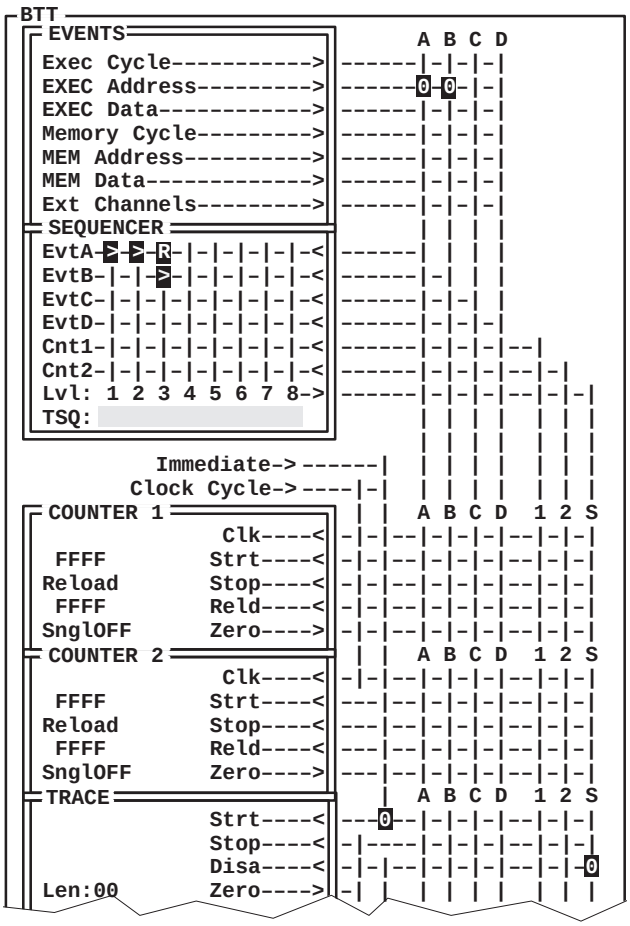
Set up tracing to start immediately and to disable when the sequence is detected.

TRACE Box

1) Set up trace immediate.
2) Set up the sequencer to disable tracing.

lesson continues on the next page →

Your BTT setup window should look like the following:



Important! Don't forget to download the XDS522A configuration and enable the XDS522A.

Debug

Reset the entry point and run the program:

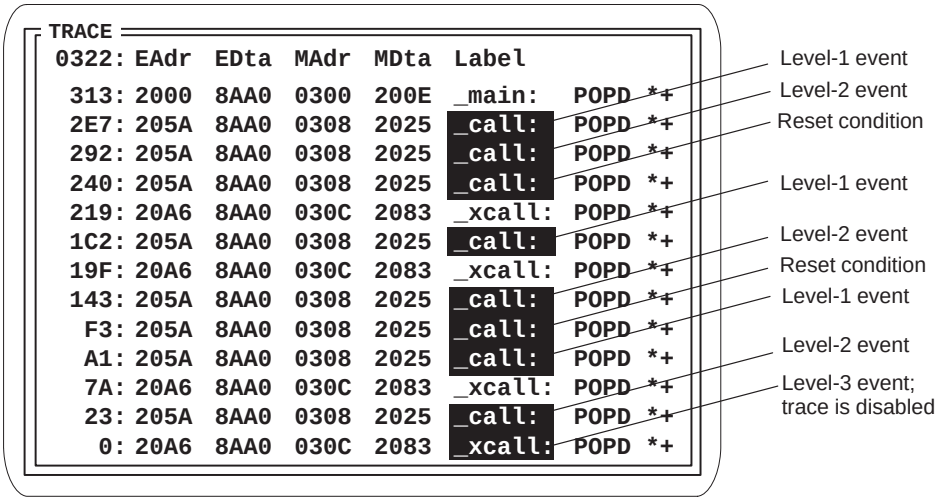
```
restart ↵
run ↵
```

BTT

To see the contents of the trace buffer, select Update from the Trace menu.

Because you are interested only in the occurrences of _call and _xcall in your TRACE window, you can filter out the other samples.

Once you set up your filter, your trace window should look similar to this:



Sample number 2E7 is labeled _call. This was the Level-1 event in the sequence. The next sample is also labeled _call. This was the Level-2 event. The next sample is another _call. You set up the sequencer to reset if it encountered an occurrence of _call between the Level-2 and the Level-3 events. Because the third occurrence of _call qualified as the reset condition, the sequencer reset itself.

Because it reset, the sequencer started looking for the Level-1 event again. (Note that the reset condition, the third occurrence of _call, did not qualify as the Level-1 event.) For that reason, the next sample, which is labeled _xcall, was ignored. Sample number 1C2, _call, became the new Level-1 event. The next sample, _xcall, was ignored. Sample number 143, _call, became the new Level-2 event. The next sample, _call, reset the sequencer again.

Sample number A1 became the next Level-1 event. Sample number 7A, labeled _xcall, was ignored. The next sample became the next Level 2 event. Finally, sample number 0 became the Level-3 event in the sequence. Because all three of the sequencer advance events were detected, tracing was disabled.

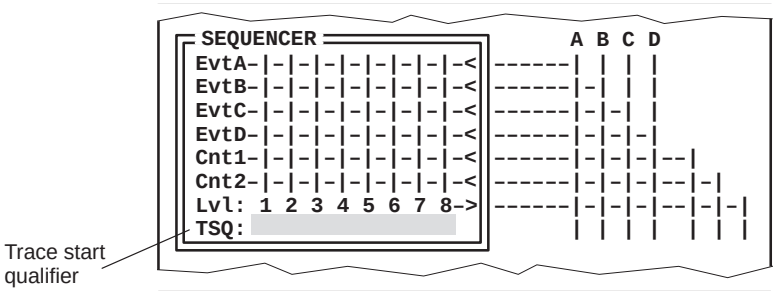
Note: Advance Has Precedence Over Reset in the Sequencer

In this lesson, if Event A and Event B had occurred during the same instruction cycle and the Level-1 and Level-2 events had already been detected, the sequencer would not have reset. Instead, it would have considered that occurrence of Event B to be the last qualifier needed in the sequence.

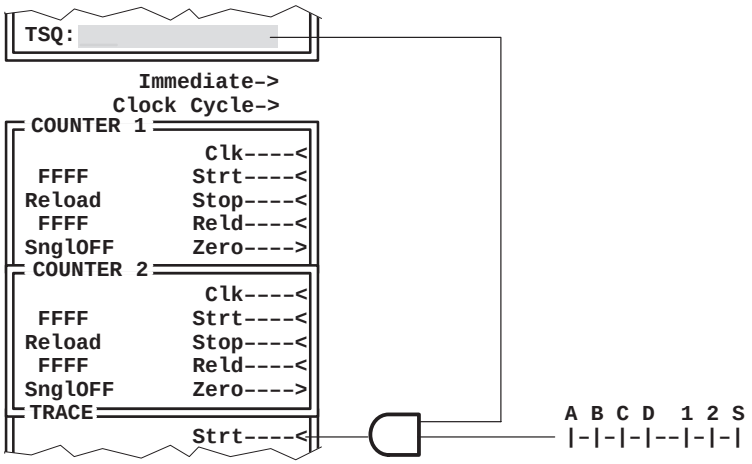
11.4 Starting Trace During a Specified Sequencer Level

Sometimes you might want to start tracing on a specified event, but only if that event occurs between two other specified events. For example, you might have a situation where you have three events (Events A, B, and C). You want to start tracing on Event C, but only when it occurs *between* Event A and Event B.

The sequencer's trace start qualifier (TSQ) lets you perform this sort of conditional tracing. The TSQ works in conjunction with the sequencer to add this conditional trace control.



The TSQ is logically ANDed with the selected trace-start event to determine whether or not tracing can start.



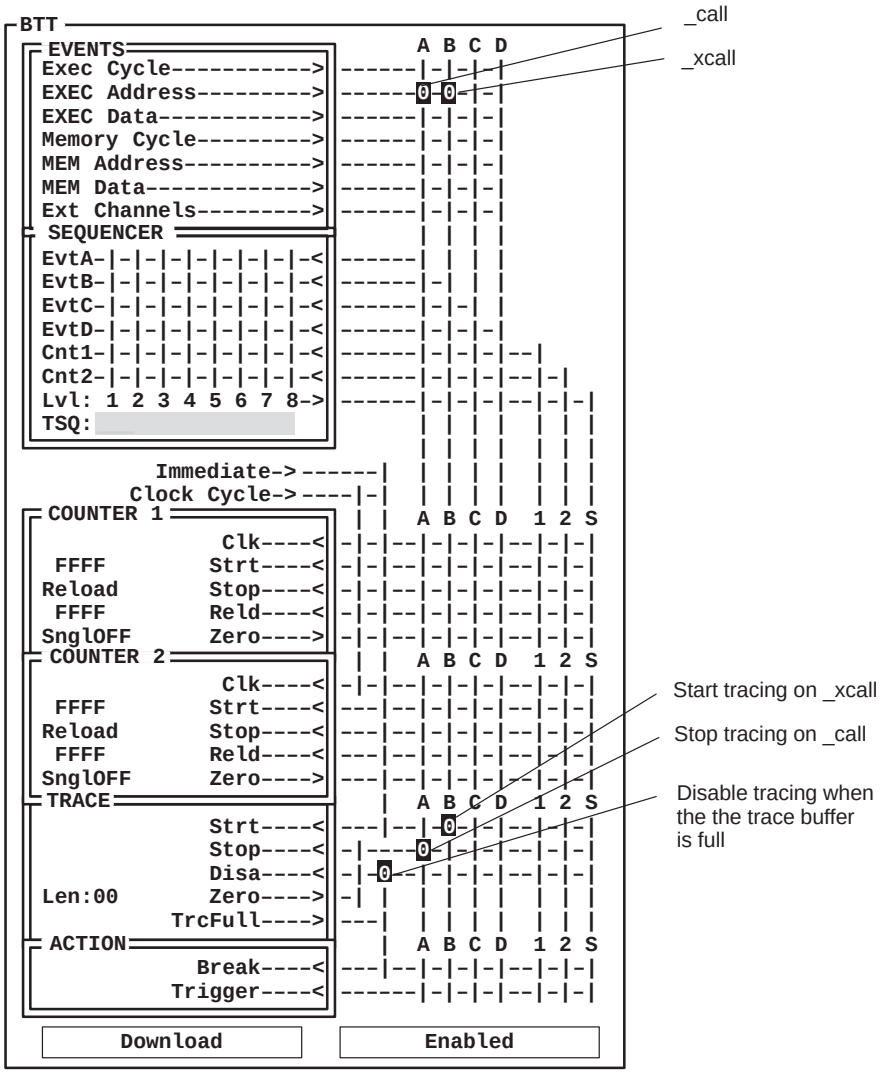
Use the TSQ to set up a condition that allows tracing to start only during specified sequencer levels. By default, all of the levels are highlighted. When all of the levels are highlighted, tracing can start at any level. You set up a TSQ event by selecting a sequencer level in which tracing *cannot* start regardless of how trace-start is specified. When a TSQ event is set up, tracing will not start until the sequencer advances to the level in which TSQ is highlighted.

The TSQ works with many other XDS522A features. To set up the TSQ properly, you must also set up tracing events and sequencer events. Often, you will want to use TSQ with at least one of the counters. When you are trying to collect specific samples, you may not always know exactly how to configure your BTT setup window, especially when you need to set up several XDS522A features at the same time. You may need to start by collecting some extraneous samples and then further qualify your events until you collect only the samples you want to see.

In the following lessons, you will start with a simple scenario that becomes more and more complex. With each new complication, you will add an XDS522A feature to qualify your events. The TSQ will be the last feature that you add to fully qualify your events.

A simple scenario

In this lesson, you will set up the XDS522A to start tracing on _xcall and stop tracing on _call. When the trace buffer is full, you will set up the XDS522A to disable tracing. As you know from previous lessons, if you set up _call as Event A and _xcall as Event B, your BTT window would look like this:

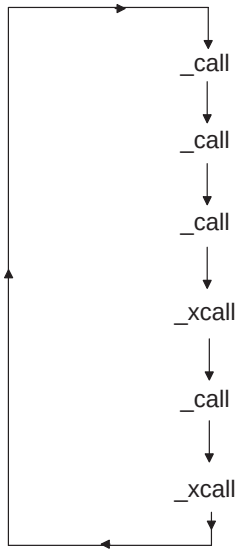


After running the sample program and filtering everything except _call and _xcall out of your TRACE window, the contents of your TRACE window would look like this:

```
TRACE
8000: EAdr  EDta  MAdr  MDta  Label
7FFF: 20A6  8AA0  030C  2083  _xcall:  POPD *+
7FA8: 205A  8AA0  0308  2025  _call:   POPD *+
7FA7: 20A6  8AA0  030C  2083  _xcall:  POPD *+
7F4B: 205A  8AA0  0308  2025  _call:   POPD *+
7F4A: 20A6  8AA0  030C  2083  _xcall:  POPD *+
7EF3: 205A  8AA0  0308  2025  _call:   POPD *+
7EF2: 20A6  8AA0  030C  2083  _xcall:  POPD *+
7E96: 205A  8AA0  0308  2025  _call:   POPD *+
```

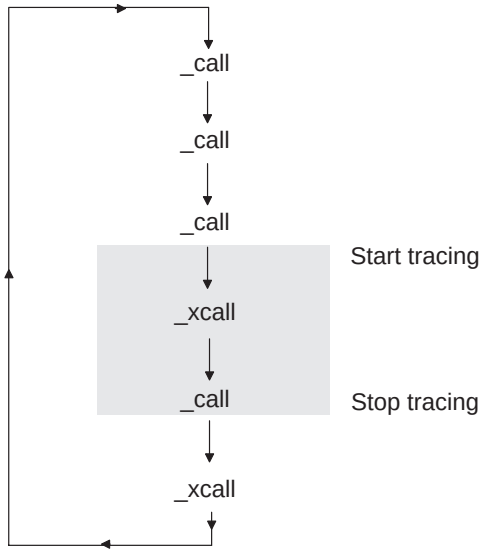

Further qualifying the events in the simple scenario

Recall how the for loop executes in the sample program:

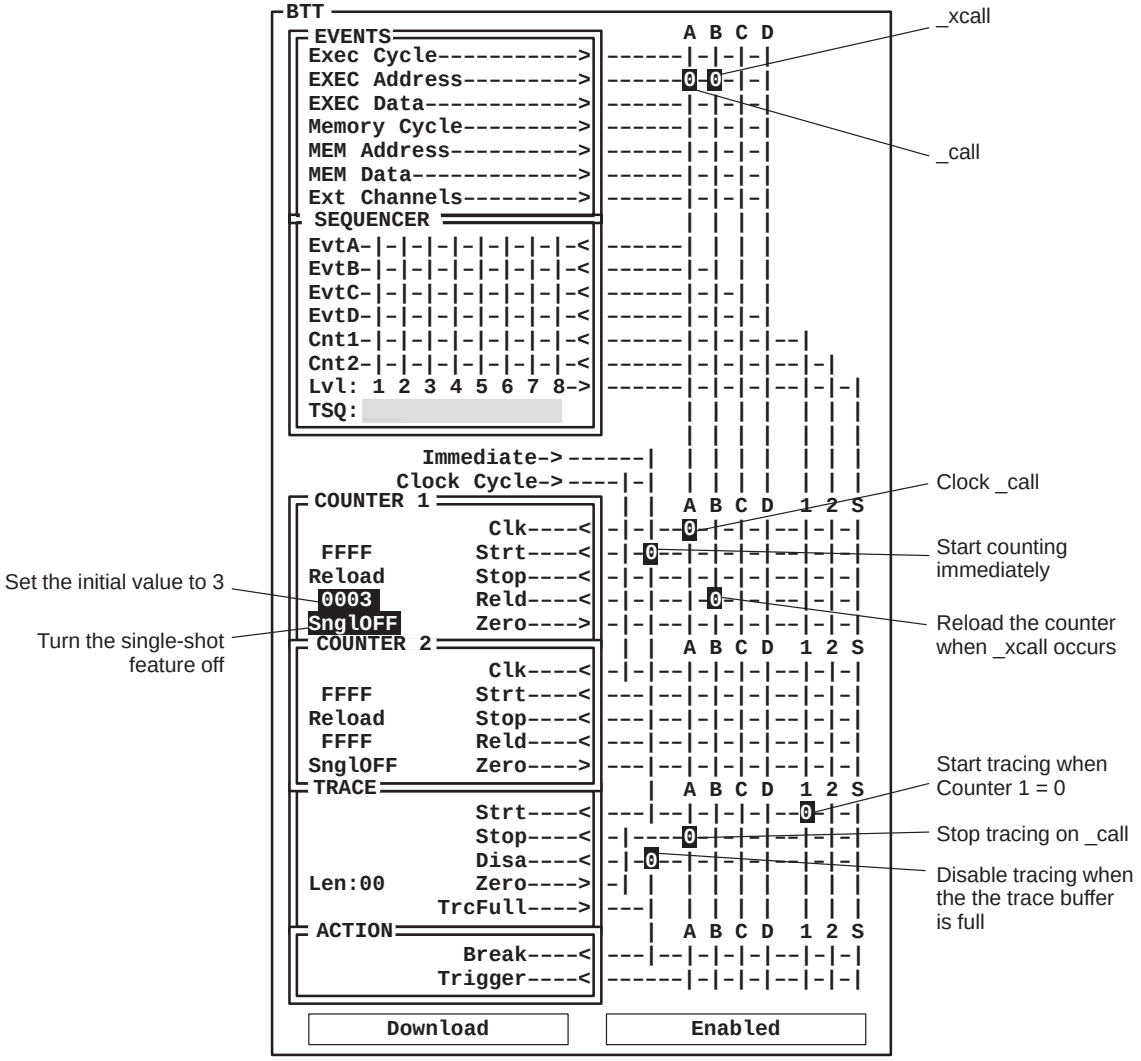


lesson continues on the next page →

Suppose you want tracing to start only after the third occurrence of _call and you want tracing to stop on the next (the fourth) occurrence of _call. If _xcall occurs before all three _calls have occurred, you want the system to start looking for three occurrences of _call again:



You can do this by using Counter 1 to count the number of _calls and by reloading Counter 1 whenever _xcall occurs. As you know from Chapter 10, *Using the Counters,* your BTT setup window would look like this:



lesson continues on the next page →

After running the sample program and filtering everything except _call and _xcall out of your TRACE window, the contents of your TRACE window would look like this:

```
TRACE
8000: EAdr  EDta  MAdr  MDta  Label

7FD9: 20A6  8AA0  030C  2083  _xcall:  POPD *+
7F82: 205A  8AA0  0308  2025  _call:   POPD *+
7F5B: 20A6  8AA0  030C  2083  _xcall:  POPD *+
7F04: 205A  8AA0  0308  2025  _call:   POPD *+
7EDD: 20A6  8AA0  030C  2083  _xcall:  POPD *+
7E86: 205A  8AA0  0308  2025  _call:   POPD *+
7E5F: 20A6  8AA0  030C  2083  _xcall:  POPD *+
7E08: 205A  8AA0  0308  2025  _call:   POPD *+
```

Notice that the sample at the top of the TRACE window is sample number 7FD9, not sample number 7FFF. Because you set up tracing to start as soon as Counter 1 reached 0 and you filtered everything out of the TRACE window except for _call and_xcall, the samples that were collected before the first _xcall are not displayed in your *filtered* TRACE window.

Here is what happened:

1) The system encountered three occurrences of Event A (_call). With each occurrence of Event A, Counter 1 decremented by 1. On the third occurrence, Counter 1 reached 0. After Counter 1 reached 0, tracing started on the next sample.

   Because Counter 1 was *not* set up as a single-shot counter, it was reloaded with the initial value of 3.

2) Event B (_xcall) was collected because it occurred between the trace-start and trace-stop events. Event B appears in your TRACE window as sample number 7FD9.

3) The system encountered another occurrence of Event A. This occurrence qualified as the trace-stop event. Event A appears in the TRACE window as sample number 7F82.

   This occurrence of Event A decremented Counter 1 from 3 to 2.

4) The system encountered another occurrence of Event B. This occurrence of Event B reloaded Counter 1 with the initial value of 3.

   Because tracing had already stopped, this occurrence of _xcall is not included in your TRACE window.

5) The for loop repeated and the system continued collecting samples in this manner until the trace buffer was full.

Table 11–1 shows the status of the trace buffer and Counter 1 *after* each occurrence of Events A and B. The highlighted lines indicate where tracing occurred.

Table 11–1. Status After Events A and B Occur

	After this event occurs	Trace status	Counter status
Before execution		Trace has not started	Current value is 3
For loop starts	Event A (_call)	Trace has not started	Current value is 2
	Event A (_call)	Trace has not started	Current value is 1
	Event A (_call)	Tracing started	Current value is 0
	Event B (_xcall)	Tracing continued	Current value is 3
	Event A (_call)	Tracing stopped	Current value is 2
	Event B (_xcall)	Tracing has not restarted	Current value is 3
For loop restarts	Event A (_call)	Tracing has not restarted	Current value is 2
	Event A (_call)	Tracing has not restarted	Current value is 1
	Event A (_call)	Tracing restarted	Current value is 0
	Event B (_xcall)	Tracing continued	Current value is 3
	Event A (_call)	Tracing stopped	Current value is 2
	Event B (_xcall)	Tracing has not restarted	Current value is 3

Notice that tracing started immediately *after* the third _call. What if you did not want tracing to start until it reached the _xcall that followed the third _call? To do this, you would need to set up a TSQ event.

Adding a conditional trace to the scenario

In this lesson, you will use TSQ to set up a conditional trace. You will set up the system to look for *only* the occurrences of _xcall that follow three occurrences of _call. If the system encounters an_xcall before all three _calls have occurred, the system will start looking for three occurrences of _call again.

This lesson is very similar to the one starting on page 11-15. However, in this lesson, you will set up the system to start tracing on the first occurrence of _xcall. That way you won't have any extraneous samples in your trace buffer.

BTT

From the BTT menu, select Clear to reset the XDS522A configuration.

In the BTT setup window, define the events that you want to use:

EVENTS Box

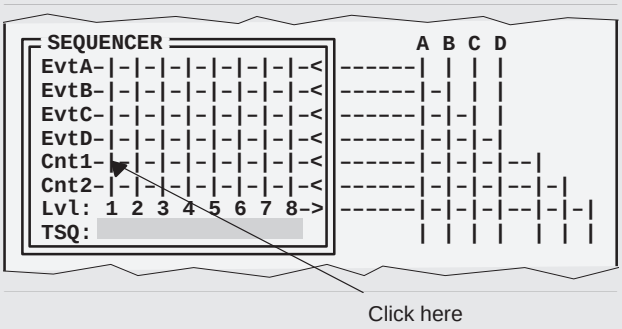
1) Define Event A as any access to the address associated with the call() function.

2) Define Event B as any access to the address associated with the xcall() function.

Set up the system to look for a sequence of events and set up the TSQ event:

SEQUENCER Box

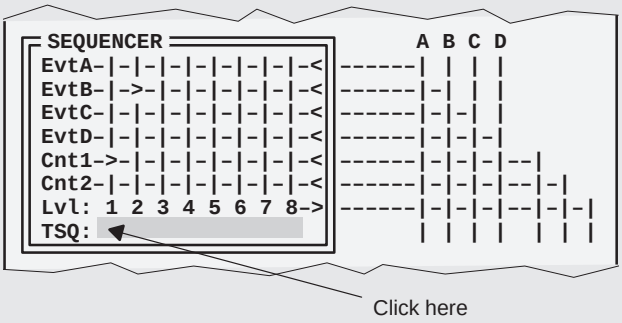
1) Set up the Level-1 event to be when Counter 1 reaches 0 by clicking on the intersection of *Cnt1* and 1.



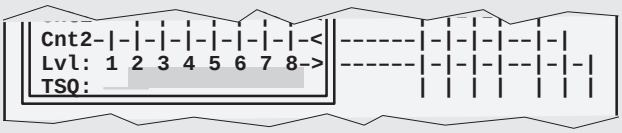
A sequence connection symbol (>) appears at the intersection.

2) Set up Event B as the Level-2 event.

3) Set up a TSQ that tells the XDS522A that it can allow trace to start *between* the Level-1 and Level-2 events only by clicking on the intersection of 1 and TSQ.



Level 1 should be the only level that is *not* highlighted in the TSQ row:



lesson continues on the next page →

Set up Counter 1 to start counting immediately, to be clocked by Event A, to count 3 occurrences of Event A, and to reload if Event B is encountered before all 3 occurrences of Event A.

COUNTER 1 Box

1) Set up Counter 1 to be clocked by Event A by clicking on the intersection of A and *Clk*.
2) Set up your start event.

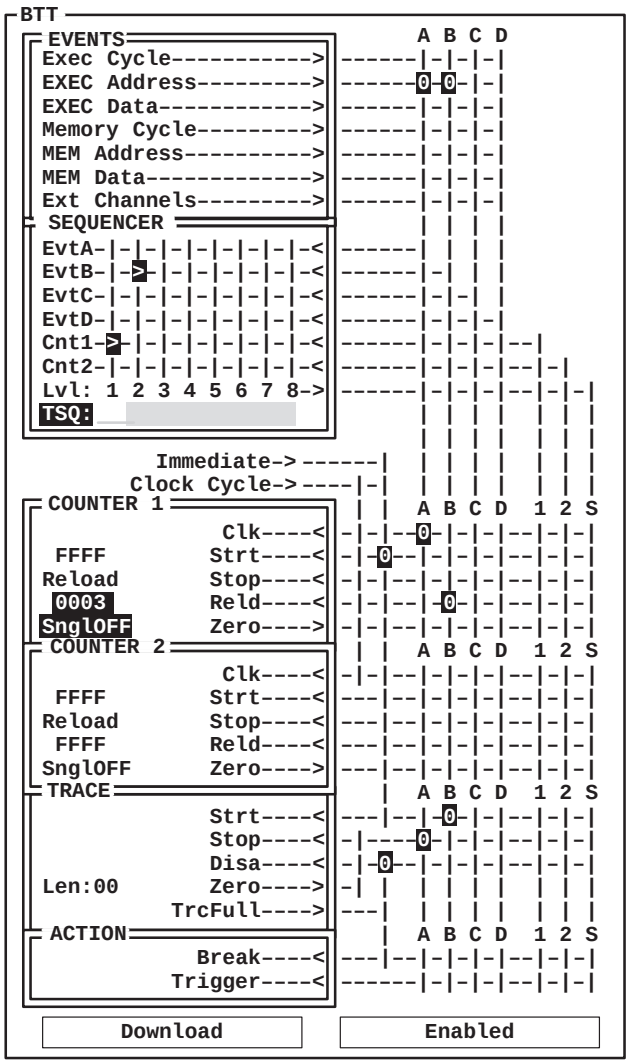
   Set up the counter to start counting as soon as the sample program begins running by clicking on the intersection of *Immediate* and *Strt*.
3) Set up 0x0003 as your initial value.
4) Set up Event B to reload the counter by clicking on the intersection of *Reld* and B.
5) Make sure the SnglOFF/SnglON toggle is turned to SnglOFF.

Set up tracing to start on Event B when it occurs between the Level-1 and Level-2 events, to stop on Event A, and to disable when the trace buffer is full.

TRACE Box

1) Set up tracing to start on Event B.
2) Set up tracing to stop on Event A.
3) Set up tracing to be disabled when the trace buffer is full.

Your BTT setup window should look like the following:



Important! Don't forget to download the XDS522A configuration and enable the XDS522A.

Debug

Reset the entry point and run the program:

```
restart ↵
run ↵
```

lesson continues on the next page →

BTT

To see the contents of the trace buffer, select Update from the Trace menu.

The best way to verify your results is to set up the trace filter to display only the samples that are labeled _call and _xcall in the TRACE window. Once you set up your filter, your TRACE window should look something like this:

```
TRACE
8000: EAdr  EDta  MAdr  MDta  Label
7FFF: 20A6  8AA0  030C  2083  _xcall:  POPD *+
7FA8: 205A  8AA0  0308  2025  _call:   POPD *+
7FA7: 20A6  8AA0  030C  2083  _xcall:  POPD *+
7F50: 205A  8AA0  0308  2025  _call:   POPD *+
7F4F: 20A6  8AA0  030C  2083  _xcall:  POPD *+
7EF8: 205A  8AA0  0308  2025  _call:   POPD *+
7EF7: 20A6  8AA0  030C  2083  _xcall:  POPD *+
7EA0: 205A  8AA0  0308  2025  _call:   POPD *+
```

Here is what happened:

1) The system encountered three occurrences of Event A (_call). With each occurrence of Event A, Counter 1 decremented by 1. On the third occurrence, Counter 1 reached 0. When Counter 1 reached 0, the sequencer advanced to the next level.

2) After the third occurrence of Event A, the system encountered Event B (_xcall). This qualified as the TSQ event and as the Level-2 event:

   - ❑ Event B qualified as the TSQ event, because it was the trace-start event *and* it occurred between the Level-1 and Level-2 events. You can verify that tracing started with Event B by looking at the top of your TRACE window.
   - ❑ Event B was also set up as the Level-2 event. Because the Level-1 event had already been detected, this occurrence of Event B qualified as the Level-2 event. With this occurrence of Event B, the sequence was detected and the system began looking for the Level-1 event again.

3) The system encountered Event A, which qualified as the trace-stop event. Event A appears in the TRACE window as sample number 7FA8.

4) The next occurrence of Event B (the second occurrence) was ignored. Because the system had not encountered the Level-1 event again, this occurrence of Event B did not qualify as the TSQ event or as the Level-2 event.

5) The for loop repeated and the system continued collecting samples in this manner until the trace buffer was full.

Table 11–2 shows the status of the trace buffer, the sequencer, and Counter 1 *after* each occurrence of Events A and B. The highlighted lines indicate where tracing occurred.

Table 11–2. Status After Events A and B Occur

	After this event occurs	Trace status	Sequencer status	Counter status
Before execution		Trace has not started	Looking for Level-1 event	Current value is 3
For loop starts	Event A (_call)	Trace has not started	Looking for Level-1 event	Current value is 2
	Event A (_call)	Trace has not started	Looking for Level-1 event	Current value is 1
	Event A (_call)	Trace has not started	Level-1 event is detected	Current value is 0
	Event B (_xcall)	Tracing started (TSQ event)	Level-2 event is detected	Current value is 3
	Event A (_call)	Tracing stopped	Looking for Level-1 event	Current value is 2
	Event B (_xcall)	Tracing has not restarted	Looking for Level-1 event	Current value is 3
For loop restarts	Event A (_call)	Tracing has not restarted	Looking for Level-1 event	Current value is 2
	Event A (_call)	Tracing has not restarted	Looking for Level-1 event	Current value is 1
	Event A (_call)	Tracing has not restarted	Level-1 event is detected	Current value is 0
	Event B (_xcall)	Tracing restarted (TSQ event)	Level-2 event is detected	Current value is 3
	Event A (_call)	Tracing stopped	Looking for Level-1 event	Current value is 2
	Event B (_xcall)	Tracing has not restarted	Looking for Level-1 event	Current value is 3

Note: TSQ and the Sample Program

In this lesson, you could have achieved the same results without using TSQ. Instead of using TSQ, you could have set up tracing to start when the sequence was detected. The purpose of this lesson was to show you how to use TSQ, but the lesson was limited by the sample program. Keep in mind that you may have situations in your own code where you must use TSQ to achieve the results you want.

11.5 Summary

Congratulations! In this chapter, you learned how to do the following:

- ❏ Detect a sequence of events
- ❏ Reset the sequencer on an event
- ❏ Use the trace start qualifier (TSQ)

If you feel uncomfortable in performing any of these tasks, take some time to review the material in this chapter.

If you feel comfortable with these tasks and the tasks covered in the previous chapters, you're ready to use the XDS522A emulation system on your own target system. Keep in mind that the XDS522A emulation system includes online help, which you can use whenever you get stuck or forget how to set something up. You can also use the information in Appendix B, *Troubleshooting*, to help you if you experience problems.

Closing the system

If you want to close the system at this time, follow these directions:

- ❏ To close the BTT software, select Exit from the File menu.
- ❏ To close the debugger, from the COMMAND window, enter:

  quit ↵

To start the software again, follow the instructions in Section 4.4, *Starting a Debugging/Testing Session*, on page 4-8.

Part I
Overview

Part II
Tutorial

Part III
User Reference

Part IV
Appendixes

Part III

Chapter 12

Defining Events

The XDS522A is an event-driven tool that allows you to monitor activity on the buses in the SE device:

- ❑ Program address bus
- ❑ Program data bus
- ❑ Data address bus
- ❑ Data data bus

The XDS522A detects the execution of instructions on these buses. In addition, you can use up to 16 external probes to monitor other parts of your target system that are not visible using the SE device.

This chapter discusses how the EVENTS box works and describes how to specify the conditions that define events. This chapter also explains how to monitor a target system using the 16 color-coded external probes.

Topic **Page**

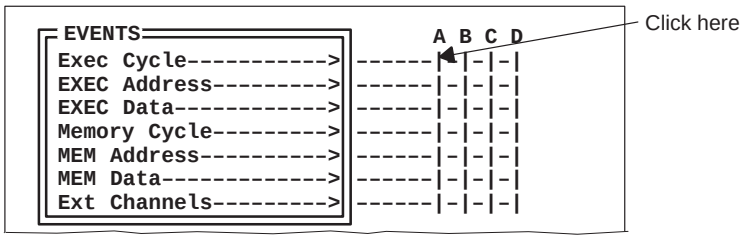
12.1 How the EVENTS Box Works

The EVENTS box allows you to set up one or more conditions that define up to four unique events (A–D). Multiple conditions placed on a vertical line in the Events box are logically ANDed. Conditions bring up dialog boxes that allow you to define those events more narrowly. The particular events that you are able to set up using the XDS522A are discussed later in this chapter.

How to define an event

To set up the conditions to define an event:

1) Point to the intersection of that condition and the event (A–D) that you want to set up. For example, if you want to define Event A as an Execution Cycle type (flattened mode), point to the intersection of A and Exec Cycle.



2) Click on the intersection. A dialog box appears at the intersection. To deselect an intersection, click on the 0; the 0 disappears.

3) Enter the appropriate information if a dialog box appears.

   Events (such as Exec Cycle) have dialog boxes that allow you to define those events more specifically. These events and their dialog boxes are discussed later in this chapter.

What the EVENTS box fields allow you to define

The events you can define depend on whether you are using flattened or unflattened mode. You can define these events in flattened mode:

- ❑ Execution cycles
- ❑ Execution addresses
- ❑ Execution data
- ❑ External channels
- ❑ Memory cycles
- ❑ Memory addresses
- ❑ Memory data

You can define these events in unflattened mode:

- ❑ Program bus cycles
- ❑ Program bus addresses
- ❑ Program bus data
- ❑ External channels
- ❑ Data bus cycles
- ❑ Data bus addresses
- ❑ Data bus data

Table 12–1 lists the fields in the EVENTS box and gives a brief description of what each field allows you to define.

Table 12–1. EVENTS Box Fields in Flattened Mode

Use this field...	To define an event as...	Page
Exec Cycle	Any type of execution cycle activity	12-4
EXEC Address	A specific execution address or a range of execution addresses	12-10
EXEC Data	A specific execution data value or a range of execution data values	12-10
Memory Cycle	Any type of memory cycle activity	12-5
MEM Address	A specific memory address or a range of memory addresses	12-10
MEM Data	A specific memory data value or a range of memory data values	12-10
Ext Channels	Readings from the 16 color-coded external probes attached to a target board	12-17

Table 12–2. EVENTS Box Fields in Unflattened Mode

Use this field...	To define an event as...	Page
PROG Bus Cycle	Any combination of instructions and data access on the program bus	12-7
PROG Address	A specific program address or a range of program addresses	12-10
PROG Data	A specific program bus data value or a range of program bus data values	12-10
DATA Bus Cycle	Any activity on the data bus	12-8
DATA Address	A specific data bus address or a range of data bus addresses	12-10
DATA Data	A specific data bus data value or a range of data bus data values	12-10
Ext Channels	Readings from the 16 color-coded external probes attached to a target board	12-17

The PROG and EXEC labels on the EVENTS box fields are equivalent, as well as the DATA and MEM/Memory labels. Both refer to the program and data buses. The different labels distinguish between unflattened and flattened modes.

12.2 Defining an Event as Cycle Activity in Flattened Mode

The XDS522A includes a pipeline flattener that filters out unexecuted instructions and aligns the 'C2xx display of prefetched data and instructions with the appropriate execution cycle. The flattener mode that you select affects how you define events. The *About the TMS320C2xx pipeline flattener* subsection on page 1-10 explains the flattener modes.

Execution cycle activity

To define an event as activity in the execution cycle, click on the intersection of Exec Cycle and an event. Figure 12–1 shows the Execution Cycle dialog box that appears. You can select up to two conditions (2nd word and any other option). Table 12–3 describes the types of execution cycle activity.

Figure 12–1. Execution Cycle Dialog Box

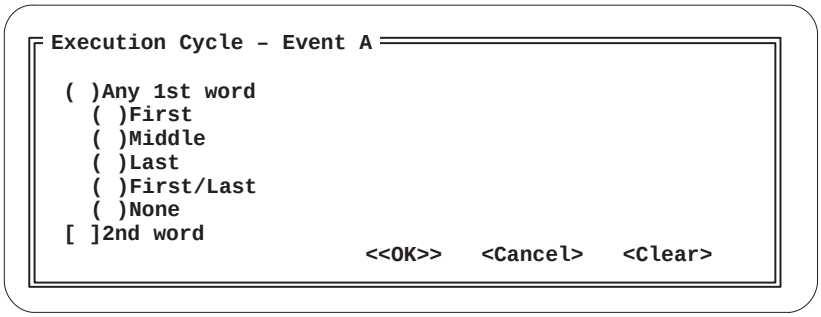


Table 12–3. Execution Cycle Dialog Box Options

Use this option ..	To define the conditions for an event as...
Any 1st word	Any first word of an instruction
First	First instruction after a discontinuity
Middle	Middle instruction with no discontinuity before or after it
Last	Last instruction before a discontinuity; normally, this is a branch or call, but in the case of an interrupt, it could be any instruction
First/Last	Instruction in which one discontinuity follows another, such as a branch to a branch instruction
None†	Any second word of a two-word instruction
2nd word†	Second word of an instruction. This option can be ANDed with one of the 1st word options.

† None is the default selection for this dialog box. You can deselect it by selecting one of the other 1st word options. Selecting 2nd word requires you to select a 1st word option; if you do not want to AND a 1st word option with 2nd word, select 2nd word and None.

Memory cycle activity

To define an event as memory cycle activity, click on the intersection of an event and Memory Cycle. Figure 12–1 shows the Memory Cycle dialog box that appears. The columns of the Memory Cycle dialog box work together to specify an event. The left column lists the possible activities in the memory cycle. The right column lists the types of memory.

To specify a memory cycle activity, choose a type of memory from the right column and choose the activity that is occurring in that type of memory from the left column. For example, if you want to define an event as a write to data memory, you would select Data from the right (memory) column and Write from the left (activity) column.

There are fourteen possible types of memory cycle activities that you can define with this dialog box. These are listed in Table 12–4.

Figure 12–2. Memory Cycle Dialog Box

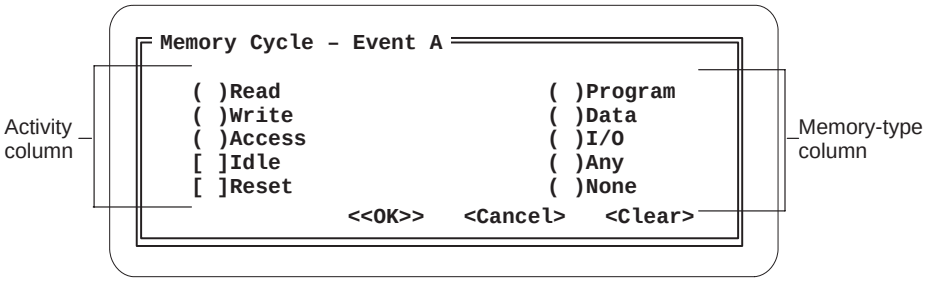


Table 12–4. Defining Memory Cycle Activities

To define this event...	Use this left column option...	With this right column option...
Read from program memory	Read	Program
Write to program memory	Write	Program
Any program memory access	Access	Program
Read from data memory	Read	Data
Write to data memory	Write	Data
Any access to data memory	Access	Data
Read from I/O port	Read	I/O
Write to I/O port	Write	I/O
Any I/O port access	Access	I/O
Any read	Read	Any
Any write	Write	Any
Any access	Access	Any
Processor low-power mode†	Idle	None
Processor reset†	Reset	None

† These options can be ORed with other events or with each other.

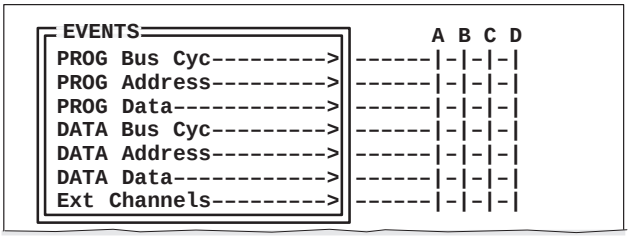
Note: Use None Option to Define Idle or Reset Without Defining Other Accesses

To define an event as just an Idle or just a reset without ORing them with any other events, you must select the None option in the right column of the dialog box. When you select None, the Access option in the left column is automatically selected. This allows you to select Idle, Reset, or both without selecting any type of access such as a data read, etc.

12.3 Defining an Event as Cycle Activity in Unflattened Mode

When you choose the unflattened mode, the XDS522A presents instructions as they are being decoded before they are sent to the CPU. Unexecuted instructions are included. Data and instructions are not aligned with the appropriate execution cycle. Figure 12–3 shows the Events box in unflattened mode.

Figure 12–3. Events Box in Unflattened Mode



Program bus cycle activity

To define an event as activity on the program bus, click on the intersection of PROG Bus Cyc and an event. Figure 12–4 shows the Program Bus Cycle dialog box that appears. All options in the dialog box can be ORed. This allows you to select up to eight possible conditions to define an event. Table 12–5 describes these types of program bus activity.

Figure 12–4. Program Bus Cycle Dialog Box

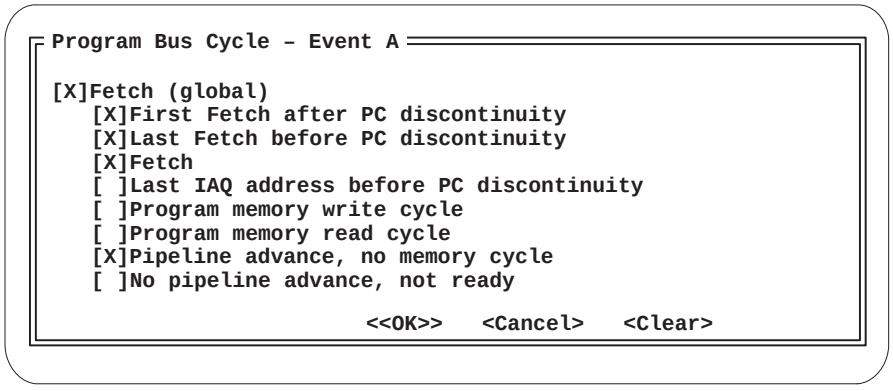


Table 12–5. Defining Program Bus Cycle Activities

Use this option...	To define this event...
Fetch (global)†	Any type of fetch
First Fetch after PC discontinuity	First program bus access after an interrupt, restore, branch, call, or trap
Last Fetch before PC discontinuity	Last program bus access before an interrupt, restore, branch, call, or trap
Fetch	Any program bus access that is not immediately followed by or preceded by a discontinuity
Last IAQ address before PC discontinuity	Last instruction acquisition address before an interrupt, restore, branch, call, or trap
Program memory write cycle	Any cycle that writes to program memory
Program memory read cycle	Any cycle that reads from program memory
Pipeline advance, no memory cycle	Next instruction in the pipeline without memory access
No pipeline advance, not ready	Cycle in which the pipeline does not advance because the CPU is waiting for the external memory interface

† Selecting this option automatically selects all types of fetches in the dialog box. See Figure 12–4.

Data bus cycle activity

To define an event as activity on the data bus, click on the intersection of DATA Bus Cyc and an event. Figure 12–5 shows the Data Bus Cycle dialog box that appears. All options in the dialog box can be ORed. This allows you to select up to eight possible conditions to define an event. Table 12–6 describes these types of data bus activity.

Figure 12–5. Data Bus Cycle Dialog Box

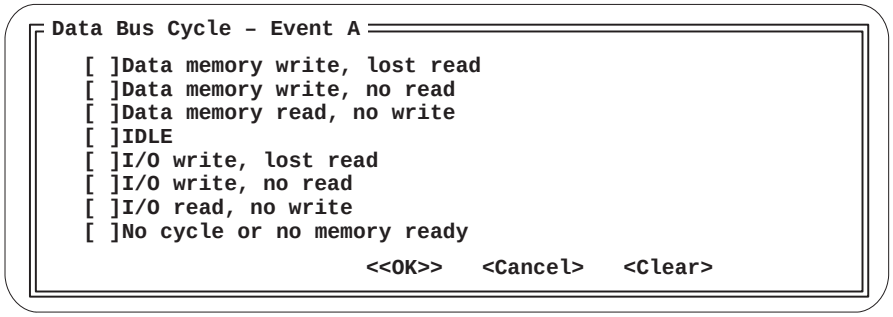


Table 12–6. Defining Data Bus Cycle Activities

Use this option...	To define an event as...
Data memory write, lost read	A write in data memory when the read is lost
Data memory write, no read	A write in data memory when no read occurs
Data memory read, no write	A read in data memory when no write occurs
IDLE	The processor is in low-power mode
I/O write, lost read	A write to I/O when the read is lost
I/O write, no read	A write to I/O when no read occurs
I/O read, no write	An I/O read when no write occurs
No cycle or no memory ready	No activity occurs on the data bus, or data memory is not ready for access

Note: Select Multiple Options to Ensure Capturing a Write

If a read and write occur in the same cycle, the write is given priority. Because it is difficult to predict when the XDS522A will be unable to detect a read, the best way to capture a write is to select both of the following:

- ❑ Data memory write, lost read
- ❑ Data memory write, no read

To capture an I/O write, select both of these:

- ❑ I/O write, lost read
- ❑ I/O write, no read

12.4 Defining an Event as an Address or Data Value

If you are using flattened mode, the XDS522A allows you to define events as:

- ❑ Execution addresses
- ❑ Execution data
- ❑ Memory addresses
- ❑ Memory data

If you are using unflattened mode, the you can define events as:

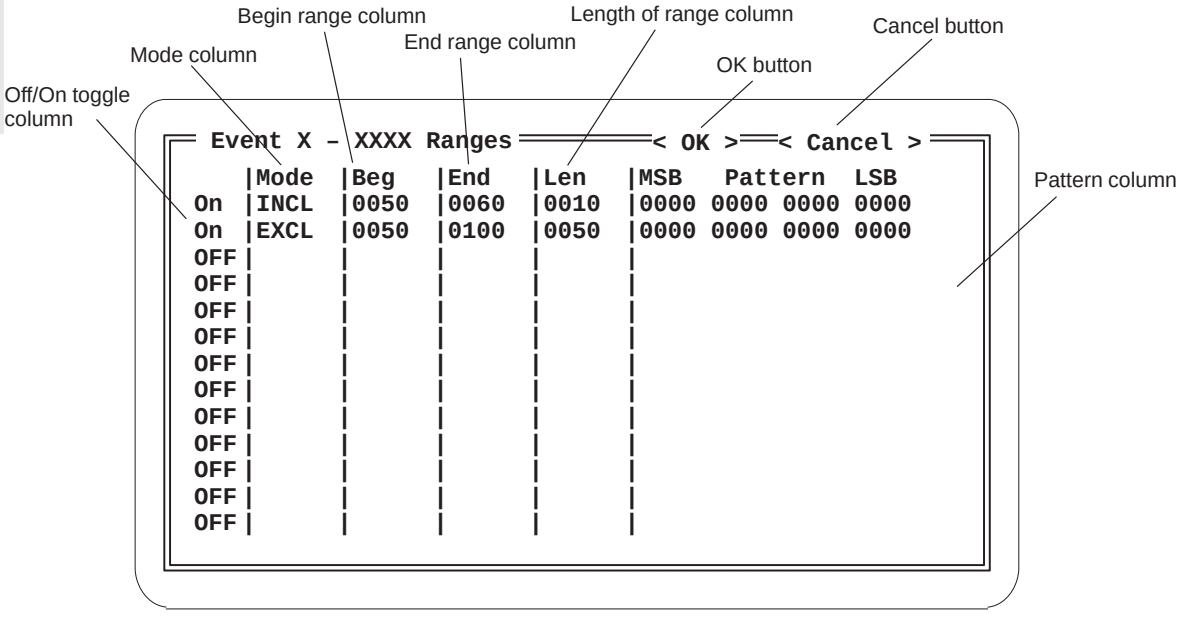
- ❑ Program bus addresses
- ❑ Program bus data
- ❑ Data bus addresses
- ❑ Data bus data

How the address or data value event dialog box works

When you define an event as an address or data value by clicking on the intersection of an event (A–D) and an address or data value condition, the dialog box in Figure 12–6 appears. This dialog box allows you to specify the conditions for the event in four ways:

- ❑ As a specific address or data value
- ❑ As a range of addresses or data values
- ❑ As a masked-bit pattern
- ❑ As a symbol address (for addresses only)

Figure 12–6. Address or Data Value Event Dialog Box



The Beg, End, and Len columns in the dialog box automatically update to reflect the information that you enter in the other columns. For example, if you enter a beginning and ending range for an event in the Beg and End columns, the Length column automatically updates to reflect the length of the range that you entered.

The rows in the dialog box are logically ORed. You can specify up to twenty specific addresses or data values, ranges, masked-bit patterns, or symbol addresses to be ORed for one event. Table 12–7 lists the fields for an address or data value dialog box and gives a brief description of each field.

Table 12–7. Address and Data Value Dialog Box Fields

Use this field...	To...
Off/On toggle	Activate the Mode, Beg, End, Len, and Pattern fields for each row
Mode	Specify whether a range is inclusive or exclusive ❑ Inclusive means that the event is defined as all the addresses or data values inside the range, including the ends of the range ❑ Exclusive means that the event is defined as all the addresses or data values outside the specified range, excluding the ends of the range
Begin range	Enter the beginning address or data value for a range
End range	Enter the ending address or data value for a range
Length of range	Enter the length of the range based on a particular address or data value
Pattern	Specify a masked-bit pattern for a range
OK	Accept the inputs you have made and dismiss the dialog box
Cancel	Return the dialog box to its previous configuration and dismiss the dialog box

Note: Clear Each Dialog Box Before You Redefine an Event

When you are ready to redefine an event, you must clear the information in the dialog box either manually or by using a Clear button if one is available. Once you have entered values in any BTT dialog box and clicked OK, these values remain in the dialog box until you change them. Even if the connection symbol is not visible in the EVENTS box, the information for that event remains in the dialog box until you clear the box or change the information. Note that although the information remains in the dialog box, it does not affect the XDS522A unless there is a connection symbol at that intersection in the EVENTS box.

Defining an event as a specific address or data value

The address and data value dialog boxes allow you to set up a single, specific value as an event. To set up a single, specific value, you must set up the dialog box as though you are defining the event as a range with one value that both begins and ends the range.

To define an event as a single, specific, address or data value:

1) Click on the intersection of an event (A–D) and the address or data condition that you want to specify. The appropriate address or data value dialog box appears.

   For example, if you want to define Event A as a particular address in the execution cycle, you click on the intersection of EXEC Address and A.

2) Click one of the rows in the On/Off column to turn On that row.

3) Ensure that the Mode column is set to INCL (inclusive). If the Mode column is set to EXCL, click once on EXCL to toggle it to INCL.

4) In the Beg column, type in the specific address or data value that you want to define as the event. The End column automatically updates with the value that you enter in the Beg column. The value in the Len column updates to 0001.

5) Click OK to define the event and dismiss the dialog box.

Because you set the value as both the beginning and end of the range and the length of the range is one, the event is defined as a single, specific data or address value.

Defining an event as a specific range of addresses or data values

The XDS522A allows you to define an event by setting up a beginning and ending address or data value for a range. To define an event as a range that begins and ends on specific data values or addresses (including the end points of the range):

1) Click on the intersection of an event (A–D) and the address or data condition that you want to specify. The appropriate address or data value dialog box appears.

2) Click one of the rows in the On/Off column to turn On that row.

3) Ensure that the Mode column is set to INCL (inclusive). If the Mode column is set to EXCL, click once on EXCL to toggle it to INCL.

4) In the Beg column, type in the beginning address or data value for the range that you want to define as the event.

5) In the End column, type in the ending address or data value for the range that you want to define as an event.

   The Len column automatically updates to reflect the length of the range between the two values that you have entered.

6) Click OK to define the event and dismiss the dialog box.

Defining an event as a specific number of addresses or data values

You can define an event as a range based on a beginning address or data value and a specified length for the range. To define an event as a range that begins on a specific data or address value and extends to a particular length:

1) Click on the intersection of an event (A–D) and the address or data condition that you want to specify. The appropriate address or data value dialog box appears.

2) Click one of the rows in the On/Off column to turn On that row.

3) Ensure that the Mode column is set to INCL (inclusive). If the Mode column is set to EXCL, click once on EXCL to toggle it to INCL.

4) In the Beg column, type in the beginning address or data value for the range that you want to define as the event.

5) In the Len column, type in the hexadecimal number of addresses or data values that you want in the range.

   The End column updates to reflect the ending value for the range based on the length you type in the Len column.

6) Click OK to define the event and dismiss the dialog box.

The event is defined as the beginning value that you entered in the Beg column through the number of addresses or data values you entered in the Len column.

Defining an event as addresses or data values outside of a specified range

If you want to define an event as all addresses or data values with the exception of a particular range, you can use the EXCL mode in the address or data value dialog box. To define an event as any values outside a specified range (excluding the end points of the range):

1) Click on the intersection of an event (A–D) and the address or data condition that you want to specify. The appropriate address or data value dialog box appears.
2) Click one of the rows in the On/Off column to turn On that row.
3) Ensure that the Mode column is set to EXCL (exclusive). If the Mode column is set to INCL, click once on INCL to toggle it to EXCL.
4) Type in the appropriate values for the range in the Beg, End, or Len columns.
5) Click OK to define the event and dismiss the dialog box.

Defining an address as a symbol address

When entering the beginning or ending address for a range, or the value for a single address or single data value, you can enter the symbol address rather than the numeric equivalent. To use symbol addresses, you must load the symbol table by using the Target Code option in the File menu or the TRGLOAD command on the command line.

To enter an address as a symbol address:

1) Click on the intersection of an event (A–D) and the address or data condition that you want to specify. The appropriate address or data value dialog box appears.
2) Click one of the rows in the On/Off column to turn On that row.
3) In the Mode column, select EXCL (exclusive) or INCL (inclusive) as appropriate.
4) Place the cursor in the appropriate Beg column or End column, depending on whether you want the symbol address to begin or end the range.
5) Press SPACE. A Get Symbol Address dialog box appears.

```
Get Symbol Address
   Case sensitivity:  ( )On  (*)Off
   Symbol:[_xcall                                  ]
                                    <<OK>>  <Cancel>
```

6) Fill in the fields in the Get Symbol Address dialog box.

   a) In the Case sensitivity field, select whether or not the symbol address is case sensitive by selecting On or Off .

   b) In the Symbol field, type in the symbol address.

   c) Click OK to set the symbol address and dismiss the Get Symbol Address dialog box.

   The numeric equivalent of the symbol address appears in the Beg or End column.

7) Enter values or symbol addresses in the other columns of the address ranges dialog box as appropriate.

8) Click OK to define the event and dismiss the dialog box.

Masking bits within a specified range

Unlike the Pattern column in the Search and Filter dialog boxes, the Pattern column in the address and data range dialog boxes does not automatically update when you enter values in the other columns. You can use the Pattern column to mask bits within a specified range.

To define a masked-bit range, first, you must specify a range in the Beg and End columns. Then, in the Pattern column, you set up the masked-bit pattern that you want to see within that range. The Pattern column is a binary representation of a hexadecimal address or data value. The Pattern column and the Beg and End columns work together to define an event as specific addresses or data values within a range.

For example, you could define an event as all the even addresses or data values within a specified range:

1) Click on the intersection of an event (A–D) and the address or data condition that you want to specify. The appropriate address or data value dialog box appears.

2) Click one of the rows in the On/Off column to turn On that row.

3) Ensure that the Mode column is set to INCL (inclusive). If the Mode column is set to EXCL, click once on EXCL to toggle it to INCL.

4) Type in values or symbol addresses in the Beg, End, or Len columns of the address ranges dialog box as appropriate.

5) Toggle the least significant bit (LSB) to 0 to define the event as all even addresses or data values within the range. Make sure that all the other bits are toggled to Xs (for don't care bits).

6) Click OK to define the event and dismiss the dialog box.

The event is defined as all the even addresses or data values within the range that you specified in the Beg and End columns.

12.5 Monitoring Your Target System With External Channels

You can use the interface adapter pod to monitor parts of your target system that you cannot view using the SE device. The external-channel connector, shown on the interface adapter pod in Figure 12–7, provides the connections to the 16 color-coded external probes that you connect to your target system. Table 12–8 lists the test lead color for each channel.

Figure 12–7. External-Channel Connector on the Interface Adapter Pod

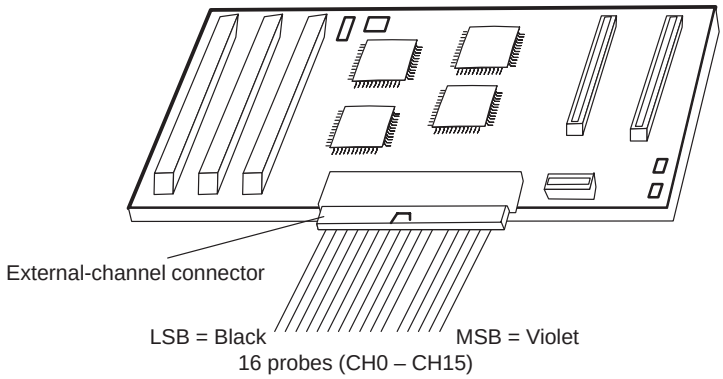


Table 12–8. External-Channel Pin Assignments

Channel		J1 Pin #	Color	Channel		J1 Pin #	Color
LSB	0	2	Black		8	10	Black
	1	3	Brown		9	11	Brown
	2	4	Red		10	12	Red
	3	5	Orange		11	13	Orange
	4	6	Yellow		12	14	Yellow
	5	7	Green		13	15	Green
	6	8	Blue		14	16	Blue
	7	9	Violet	MSB	15	17	Violet

How the External Ranges dialog box works

Figure 12–8 shows the External Ranges dialog box that appears when you click on the intersection of *Ext Channels* and an event. Table 12–9 lists the fields for the External Ranges dialog box and gives a brief description of each field.

Figure 12–8. External Ranges Dialog Box

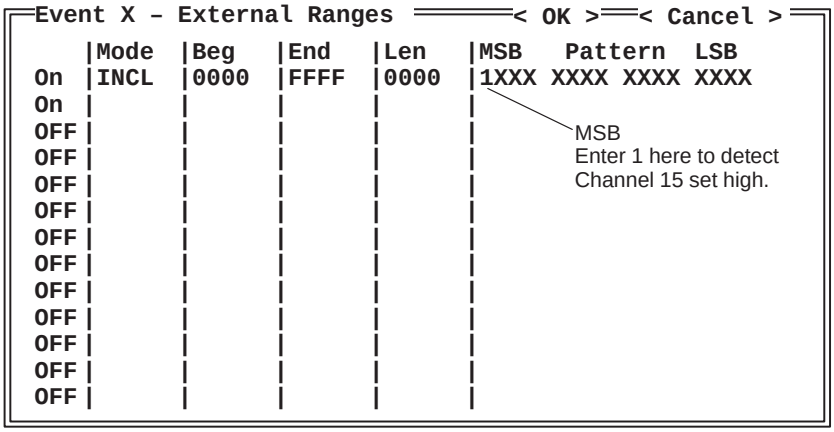


Table 12–9. External Ranges Dialog Box Fields

Use this field...	To...
Off/On toggle	Activate the Mode, Beg, End, Len, and Pattern fields for each row
Mode column	Specify whether a range is inclusive or exclusive ❑ Inclusive means that the event is defined as all data values inside the range, including the ends of the range ❑ Exclusive means that the event is defined as all the data values outside the specified range, excluding the ends of the range
Begin range column	Enter the beginning value for a range
End range column	Enter the ending value for a range
Length of range column	Enter the length of the range based on a particular channel
Pattern column	Specify a bit pattern for a range or combination of external channels ❑ Entering 1 for the bit detects a high signal on the corresponding channel ❑ Entering 0 for the bit detects a low signal on the corresponding channel ❑ Entering X for the bit masks the corresponding channel
OK button	Accept the inputs you have made and dismiss the dialog box
Cancel button	Return the dialog box to its previous configuration and dismiss the dialog box

Monitoring a combination of external channels

When you define an event based on external-channel activity, you can specify a bit pattern that indicates which channels you want to monitor and whether you want to detect a low or high signal on those channels. If you want to monitor multiple channels for one event, you can define logical AND and OR operations in the External Ranges dialog box:

- ❏ To use a logical AND operation, mark multiple bits on the same row.
- ❏ To use a logical OR operation, mark individual bits on multiple rows.

For example, if you want to define an event as high signals on both channels 5 and 7 *or* low signals on both channels 2 and 13, follow these steps:

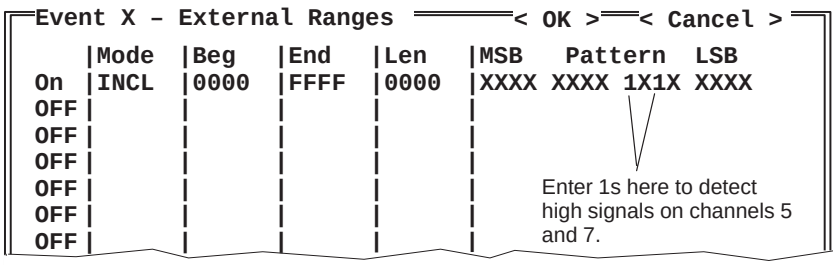
1) Click on the intersection of an event (A–D) and Ext Channels. The External Ranges dialog box appears.

2) Click a row in the On/Off toggle column to turn on the Mode, Beg, End, Len, and Pattern columns for that row.

3) In the Pattern column, set up the bit pattern for the first combination of external channels:

   a) Locate the two bits in the bit pattern that correspond to external channels 5 and 7.

   b) Set channels 5 and 7 to detect a high signal by toggling the corresponding bits from X to 1 as shown:



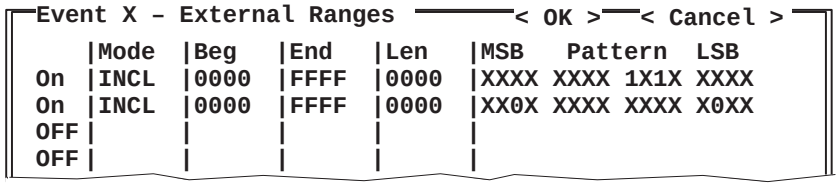
If you select OK now, the event is defined as channels 5 and 7 simultaneously detecting a high signal. For this example, you want to define the event as detecting high signals on both channels 5 and 7 *or* detecting low signals on both channels 2 and 13.

4) In the Pattern column, set up the bit pattern for channels 2 and 13 in another row to create the OR condition:

   a) Click a row in the On/Off toggle column to turn on the Mode, Beg, End, Len, and Pattern columns for that row.

   b) Locate the two bits in the bit pattern that correspond to external channels 2 and 13.

   c) Set channels 2 and 13 to detect a low signal by toggling the corresponding bits from X to 0. The dialog box should be set up as shown below.



5) Click OK to define the event and dismiss the dialog box.

The event is defined as detecting either high signals on both channels 5 and 7 *or* low signals on both channels 2 and 13.

Chapter 13

Collecting Trace Samples and Using the TRACE Window

A basic feature of the XDS522A emulation system is the ability to trace the execution of code. The XDS522A can collect trace samples and store them in a *trace buffer*, which can hold up to 32 768 (0x8000) samples. You can view the contents of the trace buffer in the TRACE window.

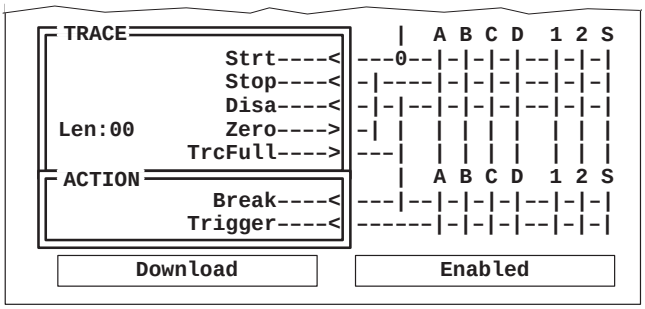
This chapter describes how to set up the system to collect trace samples, how to stop or disable tracing, and how to display the trace samples in the TRACE window, and how to view different parts of the TRACE window.

Topic **Page**

13.1 Collecting Trace Samples

The TRACE box in the BTT setup window (see Figure 13–1) allows you to control when the XDS522A collects trace samples.

Figure 13–1. TRACE Box



With the TRACE box, you can set up the XDS522A to do the following:

- *Start* collecting trace samples with one or more of these conditions:
  - As soon as your code begins running
  - When a particular event occurs
  - When a counter reaches zero
  - When a sequence of events completes
- *Stop* collecting trace samples with one or more of these conditions:
  - When a particular event occurs
  - When a counter reaches zero
  - When a sequence of events completes
  - When the XDS522A collects a specific number of trace samples

  When a *stop* condition occurs, the XDS522A stops writing to the trace buffer until the start tracing event occurs again. If the XDS522A encounters another start-tracing event, it resumes tracing.

- You can *disable* the collection of trace samples with one or more of these conditions:
  - When a particular event occurs
  - When a counter reaches zero
  - When a sequence of events completes
  - When the trace buffer is full
  - When the XDS522A collects a specific number of trace samples

  When a *disable* condition occurs, the XDS522A disables tracing. Once tracing is disabled, tracing does not reoccur, even if the XDS522A detects another occurrence of the start-tracing event. This is useful if you have a condition that is difficult to capture and you want to make sure that the trace samples for that condition are preserved after you obtain them.

  You can reenable tracing by clicking on *TrcOn=F8*, by pressing F8 , or by clicking on the *Restart* menu button while the target is still running. For more information about reenabling tracing, see the *Reenabling after a disable* subsection on page 13-7.

Setting up the system to collect trace samples

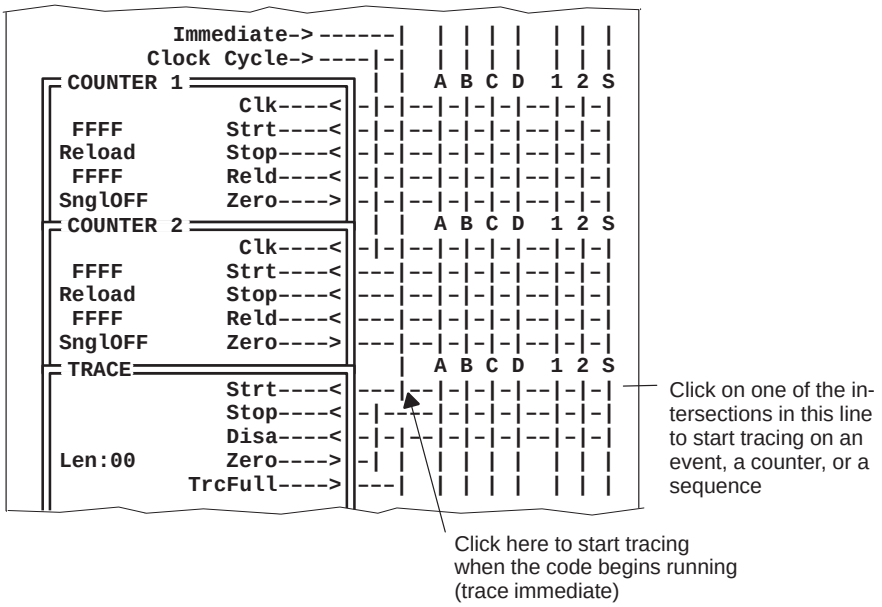
The TRACE box of the BTT setup window allows you to specify when the XDS522A should collect trace samples. To set up the system to collect trace samples, do the following:

1) Unless you want to collect samples as soon as your code begins running, define the event that you want to use to start collecting samples. Your event can be:

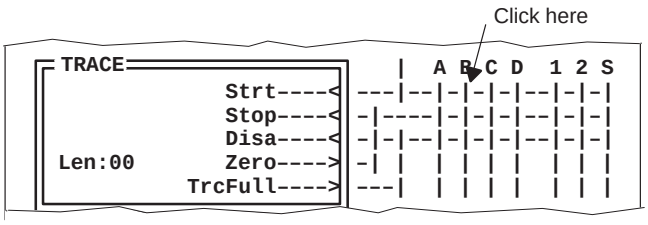
   - ❑ Any of Events A through D
   - ❑ A counter reaches 0
   - ❑ A sequence completion

   For information about defining events, see Chapter 12, *Defining Events.* For information about counters and the sequencer, see Chapter 15, *Counters*, and Chapter 16, *Sequencer.*

2) Set up the TRACE box to collect trace samples by clicking on the intersection of *Strt* and the condition that you want to monitor:



For example, to start collecting trace samples when Event B occurs, click on the intersection of *Strt* and Event B:



Collecting samples until the trace buffer is full

The trace buffer can hold 32 768 (0x8000) samples. You can use the *trace buffer full* condition to generate an action. When the trace buffer is full, you can do the following:

- ❏ Disable tracing
- ❏ Generate a hardware breakpoint

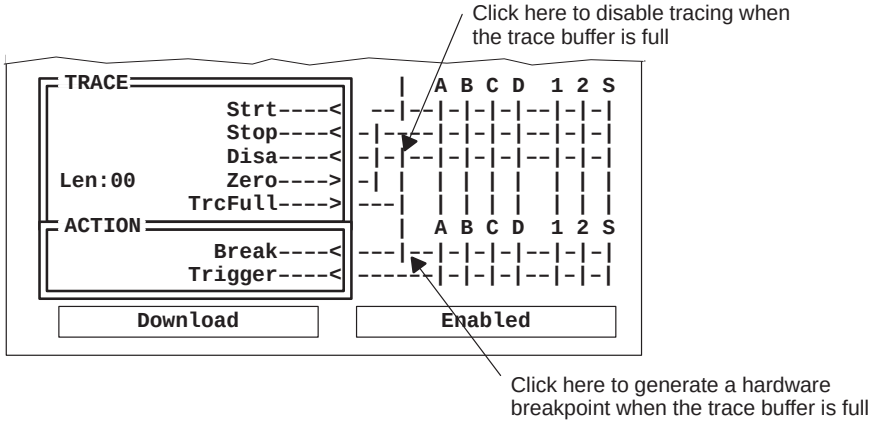
To set up the XDS522A to collect samples until the trace buffer is full, follow these steps:

1) Define the event that you want to use as a start event.

2) Set up the action that you want to occur when the trace buffer is full:

   a) If you want to disable tracing when the trace buffer is full, click on the intersection of *TrcFull* and *Disa*.

   b) If you want to generate a hardware breakpoint when the trace buffer is full, click on the intersection of *TrcFull* and *Break*.



3) Download the XDS522A configuration and be sure that the XDS522A is enabled.

4) If you want to generate a hardware breakpoint, set up the debugger's on-chip analysis module to recognize and generate hardware breakpoints. For information about setting up the analysis module, see the *TMS320C2xx C Source Debugger User's Guide*.

5) Start your application.

Collecting a specific number of trace samples

The XDS522A tracing feature allows you to collect a specific number of trace samples. The *Len* field in the TRACE box allows you to specify the number of additional samples that you want to collect after a certain condition occurs. You enter the length value as a hexadecimal number.

Once the XDS522A detects the start condition that you specify, it begins with the length value and internally decrements that value by 1 for each sample it collects. If the XDS522A detects the start condition again before the count reaches zero, it reloads the length value that you specified and starts decrementing again. Each time the start condition is detected, the XDS522A reloads the length value that you specified and starts decrementing.

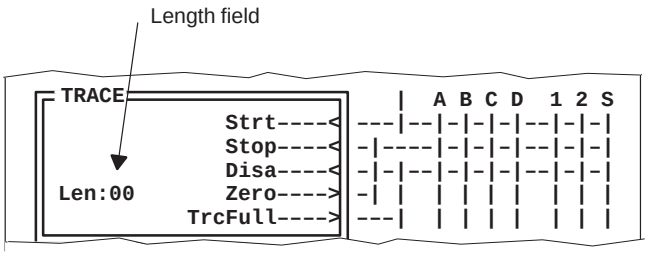
Once the count reaches zero, the XDS522A stops or disables tracing.

- If the XDS522A detects only one start condition and does not reload the length value, the trace buffer contains Length + 1 trace samples.
- If the XDS522A detects the start condition two or more times before the count reaches zero, the trace buffer contains Length + 1 trace samples *plus* the samples collected before the length value was last reloaded.

To set up the XDS522A to collect a specific number of trace samples, follow these steps:

1) Define the event that you want to use as a start event.

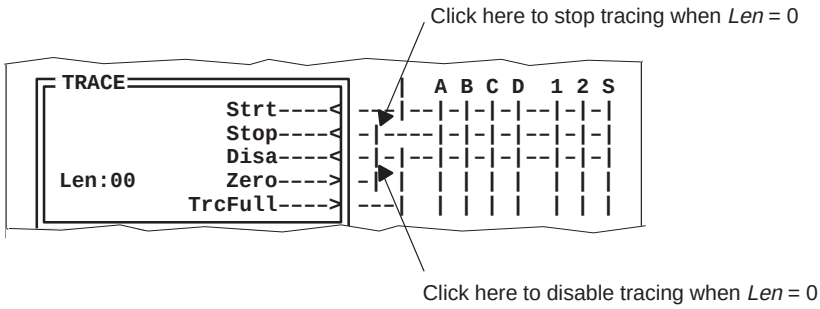
2) Set up the Len field to trace a specific samples after the start event occurs.



3) Set up the action that you want to occur when *Len* = 0:

   a) If you want to stop tracing when *Len* = 0, click on the intersection of *Zero* and *Stop*.

   b) If you want to disable tracing when *Len* = 0, click on the intersection of *Zero* and *Disa*.

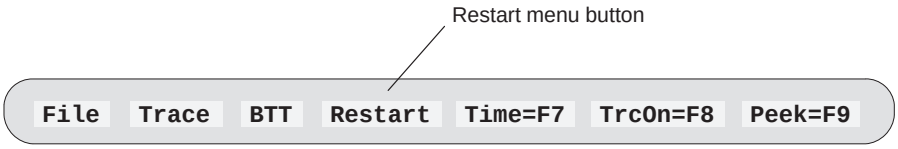


4) Download the XDS522A configuration and be sure that the XDS522A is enabled.

5) Start your application.

Reenabling after a disable

Once tracing is disabled, tracing does not occur, even if the XDS522A detects another occurrence of the start-tracing event. To reenable tracing without downloading a new XDS522A configuration, use the *Restart* menu button:



The Restart menu button restores the BTT setup window to its last downloaded configuration and starts tracing.

You can also click on the TrcOn/Off menu button or press F8 to turn tracing on and off.

- When the TrcOn=F8 menu button is visible, tracing is on. Click on TrcOn=F8 or press F8 to disable tracing (even if you have configured the BTT setup window to collect trace samples indefinitely). The menu button changes to TrcOff=F8, telling you that tracing is currently disabled.
- When the TrcOff=F8 menu button is visible, tracing is off. Click on TrcOff=F8 or press F8 to enable tracing and to display the contents of the trace buffer in the TRACE window.

13.2 Flushing or Accumulating Trace Samples

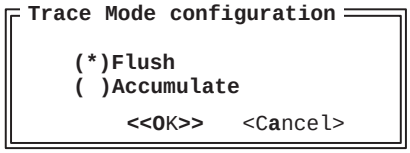
When you set up the XDS522A to collect trace samples, you can choose to overwrite the contents of the trace buffer or append new samples to the current trace buffer:

- To overwrite the current contents of the trace buffer, use the *trace flush* tracing mode. This is the default tracing mode when you invoke the BTT software.
- To append new trace samples to the current trace buffer, use the *trace accumulate* tracing mode. This mode prevents previous trace samples from being overwritten.

Select the tracing mode that you want to use *before* you set up the XDS522A to collect trace samples. To select a tracing mode, follow these steps:

1) From the Trace menu, select Mode.

   This displays the Trace Mode configuration dialog box:



2) Select the tracing mode that you want to use.

3) Click on OK.

13.3 Updating the TRACE Window With the Contents of the Trace Buffer

After XDS522A collects trace samples, you can view the contents of the trace buffer in the TRACE window. There are several ways to update the TRACE window:

- While the XDS522A is collecting trace samples, click on *Peek=F9* in the menu bar, press F9, or enter PEEK from the COMMAND window.

  Peek updates the TRACE window with one screen of the samples that have been collected. Peek does not disable tracing; you can use Peek multiple times while the XDS522A is collecting samples to update the TRACE window with the latest contents of the trace buffer.

- While the XDS522A is collecting trace samples or when it has stopped collecting trace samples, select Update from the Trace menu or enter TRACEUPDATE from the COMMAND window.

  Update displays the latest trace samples stored in the trace buffer *and* disables tracing. To reenable tracing, click on the TrcOff=F8 menu button or press F8.

When you use Peek or Update to update the TRACE window while the XDS522A is collecting samples, the trace buffer is temporarily disabled from accumulating new samples during the update. Moreover, you could loose a few samples while the XDS522A updates the TRACE window.

13.4 Moving Through the TRACE Window

Once you have collected trace samples and updated the TRACE window with those samples, you can move through the TRACE window to view the samples.

Using key sequences to move through the samples

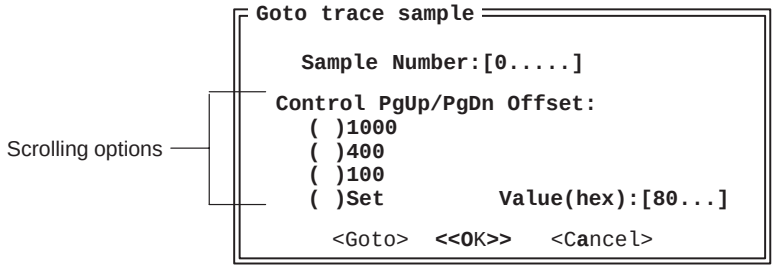
When the TRACE window is active, the BTT software allows you to move through the trace samples that you collected.

- ❏ To scroll through the newest samples, one window length at a time, use PAGE DOWN.
- ❏ To scroll through the oldest samples, one window length at a time, use PAGE UP.
- ❏ To adjust the TRACE window's contents so that the newest sample is shown in the window, use END.
- ❏ To adjust the TRACE window's contents so that the oldest sample is shown in the window, use HOME.
- ❏ To move the cursor up, one line at a time, use ↑.
- ❏ To move the cursor down, one line at a time, use ↓.

You can set up the BTT software to scroll through a specific number of samples (instead of a window length of samples). To do so, follow these steps:

1) Select Goto from the Trace menu.

   This displays the Goto trace sample dialog box:



2) To select a specific number of samples to scroll in the TRACE window, configure one of the the scrolling options:

   - ❏ Click next to 1000 to scroll 0x1000 samples at a time.
   - ❏ Click next to 400 to scroll 0x0400 samples at a time.

- ❑ Click next to 100 to scroll 0x0100 samples at a time.
- ❑ Click next to Set to specify a hexadecimal value other than 1000, 400, or 100; specify the value in the Value field.

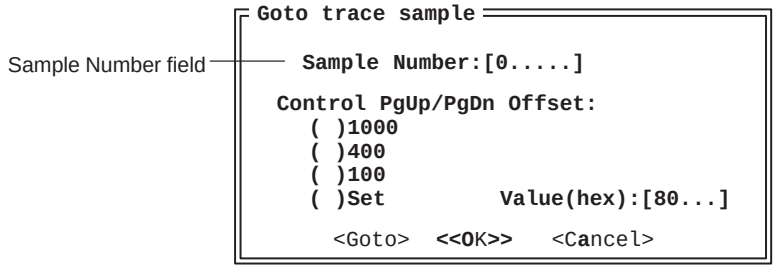
Once you configure a scrolling option, you can use CONTROL PAGE DOWN and CONTROL PAGE UP to page through the newest and the oldest samples, respectively, in increments equal to the value that you've chosen.

Displaying a specific trace sample

The BTT software allows you to display a specific trace sample. To do so, follow these steps:

1) Select Goto from the Trace menu.

   This displays the Goto trace sample dialog box:



2) In the Sample Number field, enter the number of the sample that you want to display.

3) Click on Goto.

13.5 Understanding the Parts of the TRACE Window

The TRACE window displays relevant information about the samples that you collect. Figure 13–2 illustrates the TRACE window in flattened mode. Table 13–1 describes the information in the TRACE window columns. Table 13–1 also lists the short and long versions of the headings that you can display in the columns of the TRACE window. If you are very familiar with cycle types, you probably want to use the short version. If you are not familiar with cycle types, you might want to use the long version. The default for all cycle types is the long version.

You can configure the information in the TRACE window with the Trace configuration dialog box. For more information about configuring the TRACE window, see Section 13.6 on page 13-16.

Figure 13–2. TRACE Window

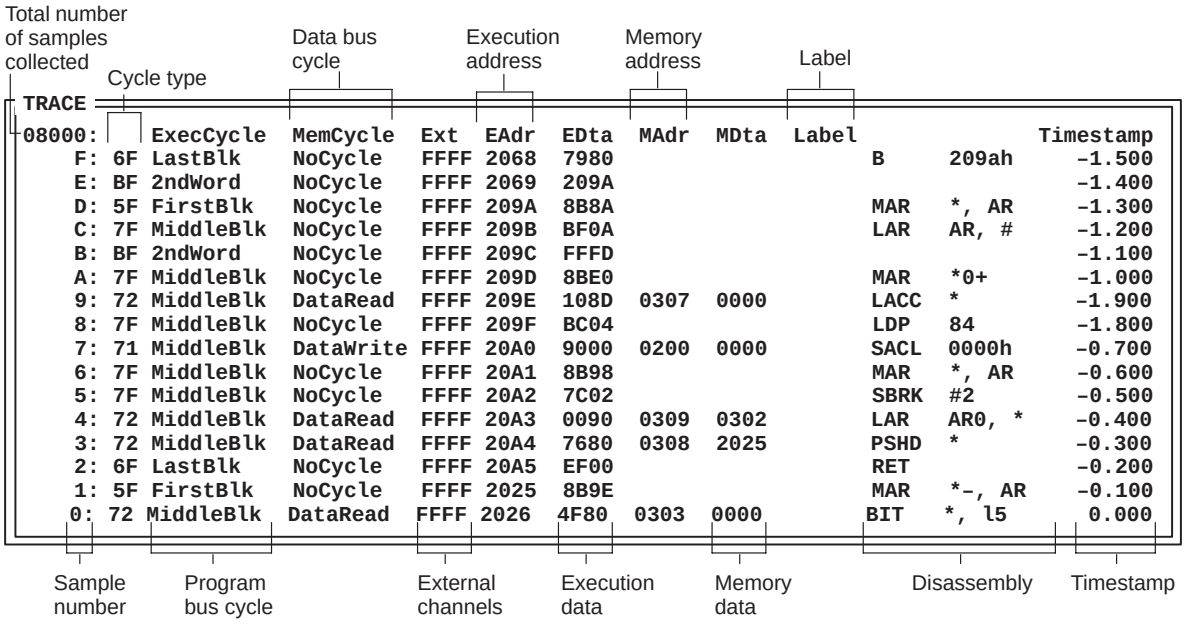


Table 13–1. TRACE Window Column Descriptions

Column	Column Heading	Column contents
Sample number	5-digit hexadecimal number that shows the total number of samples	The number for each sample in the trace buffer. The newest sample is always numbered 0.
Cycle type	❑ Short: No heading ❑ Long: CycleTyp	The binary or hexadecimal numeric code corresponding to the cycle type
Execution/program cycle	*Flattened version:* ❑ Short: Ex–MC ❑ Long: ExecCycle *Unflattened version:* ❑ Short: PC–DC ❑ Long: ProgCycle	Description of program bus activity for each sample
Memory/data cycle	*Flattened version:* ❑ Short: Ex–MC ❑ Long: MemCycle *Unflattened version:* ❑ Short: PC–DC ❑ Long: DataCycle	Description of data bus activity for each sample
External channels	Ext	The data from the 16 external channels
Execution/program address	*Flattened version:* EAdr *Unflattened version:* PAdr	The data on the 16-bit program/execution address bus
Execution/program data	*Flattened version:* EDta *Unflattened version:* PDta	The data on the 16-bit program/execution data bus
Memory/data address	*Flattened version:* MAdr *Unflattened version:* DAdr	The data on the 16-bit data/memory address bus
Memory/data data	*Flattened version:* MDta *Unflattened version:* DDta	The data on the 16-bit data/memory data bus
Label	Label	Symbols that are used to identify a section of code
Disassembly	No heading	The program disassembly code
Timestamp	Timestamp	48-bit hardware timestamp for each sample. Can be displayed in Absolute, Relative, or Delta time

About the mnemonics in the cycle columns

For the execution/program cycle and the memory/data cycle columns, the XDS522A uses codes and mnemonics to describe the cycle type. Table 13–2 lists the codes and mnemonics that appear in the the cycle columns of the TRACE window when you collect samples in the flattened mode.

Table 13–2. Mnemonics for TMS320C2xx With Flattener

(a) Execution-cycle mnemonics (bits 7 through 4 in the Cycle Type column)

	Mnemonics		
Code	**Short**	**Long**	**Cycle Type**
01 00	FL–	First/Last	Last instruction of one block and first instruction of next
01 01	Ft–	FirstBlk	First instruction after a discontinuity
01 10	Lt–	LastBlk	Last instruction before a discontinuity
01 11	Mi–	MiddleBlk	Instruction that occurred neither before nor after a discontinuity
	Il–	Illegal	Invalid bus activity
10 11	2w–	2ndWord	Second word of an instruction
11 11	No–	NoInstr	No instruction executed

(b) Memory-cycle mnemonics (bits 3 through 0 in the Cycle Type column)

	Mnemonics		
Code	**Short**	**Long**	**Cycle Type**
000X	DW	DataWrite	Write in data memory
0010	DR	DataRead	Read in data memory
0011	PW	ProgWrite	Write in program memory
010x	IW	IOWrite	I/O write
0110	IR	IORead	I/O read
0111	PR	ProRead	Read in program memory
	NC	NoCycle	No data access
1000	RS	ResetStart	Processor started reset
1001	RE	ResetEnd	Processor completed reset
1010	IS	IdleStart	Processor entered low-power, idle mode
1011	IE	IdleEnd	Processor left low-power, idle mode

Table 13–3 lists the codes and mnemonics that appear in the the cycle columns of the TRACE window when you collect samples in the unflattened mode.

Table 13–3. Mnemonics for TMS320C2xx Without Flattener

(a) Program bus mnemonics (bits 5 through 3 in the Cycle Type column)

	Mnemonics		
Code	**Short**	**Long**	**Cycle Type**
0 0 0	<D–	Fetch<Dis	First fetch after PC discontinuity
0 0 1	>D–	Fetch>Dis	Last fetch before PC discontinuity
0 1 0	Fe–	Fetch	Fetch
0 1 1	ID–	IAQ>Dis	Last instruction acquisition address before PC discontinuity
1 0 0	PW–	PWrite	Program memory write cycle
1 0 1	PR–	PRead	Program memory read cycle
1 1 0	Pi–	PipeAdv	Pipeline advance, no memory cycle
1 1 1	No–	NoCycle	No pipeline advance, not ready

(b) Data bus mnemonics (bits 2 through 0 in the Cycle Type column)

	Mnemonics		
Code	**Short**	**Long**	**Cycle Type**
0 0 0	DL	DWrLostRd	Data memory write, lost read
0 0 1	DW	DWrNoRd	Data memory write, no read
0 1 0	DR	DRdNoWr	Data memory read, no write
0 1 1	Id	Idle	Processor in low power mode
1 0 0	IL	IOWrLostRead	I/O write, lost read
1 0 1	IW	IOWrNoRd	I/O write, no read
1 1 0	IR	IORdNoWr	I/O read, no write
1 1 1	No	NoCycle	No cycle or no memory ready

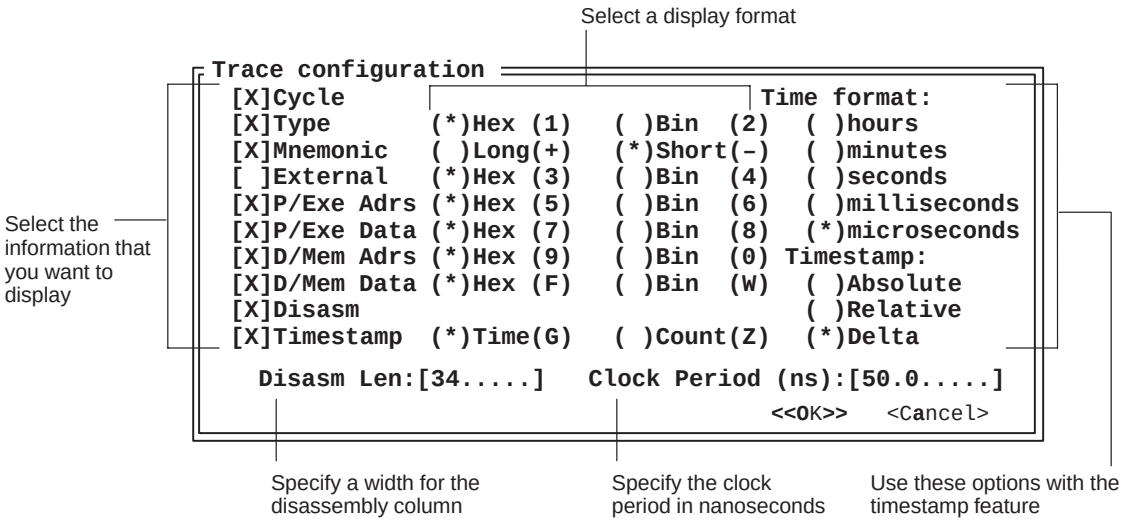
13.6 Displaying or Hiding Information About Trace Samples

The large amount of information stored for each trace sample can make your display difficult to read. You can hide some of the fields in the TRACE window.

To display or hide the fields in the TRACE window, follow these steps:

1) From the Trace menu, select Config.

   This displays the Trace configuration dialog box:



2) Select or deselect the options that you want to display or hide by clicking next to the option name. For more information about the specific options, refer to the *XDS522A Emulation System Online Help*.

3) When you are finished, click on OK.

Tips for displaying information in the TRACE window

Turning on all of the fields in the TRACE window will probably make your display difficult to read. Use the following tips to manage the content of your TRACE window:

- The XDS522A can display the cycle type as a number (binary or hexadecimal) or as a mnemonic. The numbers listed in the CycleTyp column of the TRACE window correspond to the mnemonics shown in the ExecCycle/ProgCycle and MemCycle/DataCycle columns (as shown in Table 13–2 and Table 13–3).

  To conserve space in the TRACE window, turn off either the numeric listing or the mnemonic listing by deselecting either *Type* or *Mnemonic* in the Trace configuration dialog box. The *Type* or *Mnemonic* options control when both the execution and data cycle columns are displayed.

- You can save space in the TRACE window by displaying the short versions of the cycle type mnemonics.
- When choosing a format for displaying information, keep the following in mind:
  - If you plan to use bit patterns in the dialog boxes (for example, in the Filter config dialog box), it's useful to use the binary formats in the TRACE window.
  - If you plan to use the state and mask values in the dialog boxes, it's useful to use the hexadecimal formats in the TRACE window.

Saving the TRACE window configuration

When you change the TRACE window configuration, the XDS522A saves the changes automatically and recalls the new configuration when you reinvoke the BTT software.

If you want to save a specific TRACE window configuration, use the TRACECFGSAVE command. The syntax for this command is:

tracecfgsave *filename*

This command saves the current TRACE window configuration in *filename*.

To load a previously saved configuration, use the TRACECFGLOAD command. The syntax for this command is:

tracecfgload *filename*

13.7 Saving and Loading the Contents of the Trace Buffer

You can save the contents of the trace buffer to a file. This is useful if you want to reload the trace buffer with the contents of previous trace or if you want to use the trace information in text format.

Determining the file format

You can save the trace buffer in one of two formats: binary or text.

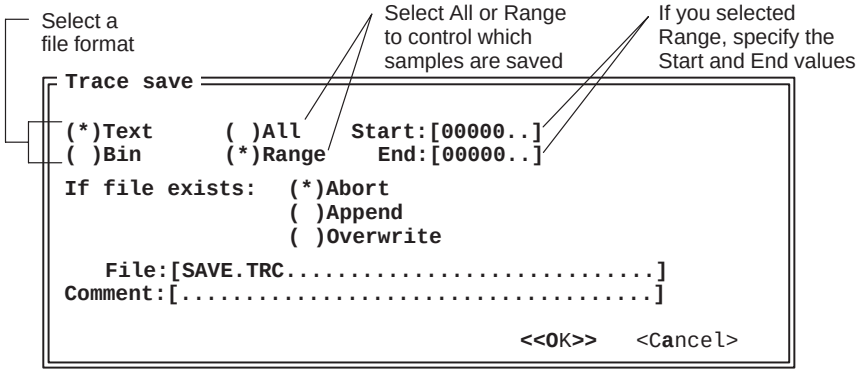
- Use *binary* format if you plan to reload the trace buffer contents. When you save the files as binary, the XDS522A saves *all* of the information about each trace sample, including information in the columns that you might have hidden in the TRACE window.
- Use *text* format if you do not need to reload the trace buffer and you want to view trace samples in another application or print them out. When you save the file as text, the XDS522A saves *only* the columns of information that are currently displayed in the TRACE window. For example, if you changed the TRACE window configuration to hide the disassembly listing, the disassembly will not appear in your saved text file.

Saving the contents of the trace buffer

To save the current contents of the trace buffer, follow these steps:

1) Select Save from the Trace menu.

   This displays the Trace save dialog box:



2) Select a file format:
   - Select *Text* if you want to use the file in text format.
   - Select *Bin* if you want to reload the trace buffer contents.

3) Select the number of samples that you want to save:

   - ❑ Select *All* if you want to save all of the samples in the trace buffer.
   - ❑ Select *Range* if you want to save a range of samples. Use the *Start* and *End* fields to specify the range.

   You can save the entire trace buffer to a file. However, if the the trace buffer is full when you save it, the save might take several minutes to complete. Save a smaller range of samples to avoid this problem.

4) Select how you want BTT software to handle file overwrites:

   - ❑ Select *Abort* to cancel the save if a file of the same name exists.
   - ❑ Select *Append* to add the current trace samples to the end of an existing file.
   - ❑ Select *Overwrite* to save over the contents of an existing file.

5) Enter a filename in the *File* field.

6) If you want to add a 33-character comment to the saved file, use the *Comment* field. You can use this comment to describe the trace buffer contents, which will help you distinguish among several saved sets of trace samples. You can view the file with your usual text editor; the comment is shown at the beginning of the file.

7) Click on OK.

You can also save the trace buffer by using a command. The syntax for the TRACESAVE command is:

tracesave *filename*, *start*, *end*, { **y** | **n** } , { **y** | **n** }

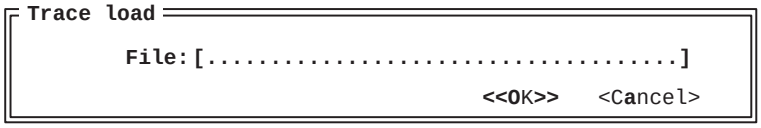
- ❑ *filename* names the file for saving the trace samples.
- ❑ *start* indicates the number of the first sample that you want to save.
- ❑ *end* indicates the number of the last sample that you want to save; this value must be smaller than the start value.
- ❑ The first set of { **y** | **n** } indicates how you want to save the file: **y** indicates text mode; **n** indicates binary mode.
- ❑ The second set of { **y** | **n** } indicates whether you want to append or overwrite an existing file; **y** indicates append/overwrite; **n** indicates overwrite.

Loading the contents of the trace buffer

To load the a previously saved trace buffer, follow these steps:

1) Select Load from the Trace menu.

   This displays the Trace load dialog box:



2) Enter a filename in the *File* field. The filename that you name must be saved in binary format.

3) Click on OK.

Chapter 14

Viewing Trace Samples

Once you have collected samples in the trace buffer, you can use the filter, search, and timestamp features to change the way you view the samples in the trace buffer.

The filter and search features help you to isolate specific samples or sets of samples. The timestamp feature allows you gather precise timing information to help you debug your code.

Topic **Page**

14.1 Filtering

You can filter a collection of samples to show only the samples that you are interested in seeing. Use the filter feature when you want to see all of the occurrences of a specific sample (or set of samples) at a glance.

After filtering, the TRACE window shows only the trace samples that you are interested in seeing. The unwanted samples are not discarded during filtering, only masked (or hidden) from view. You can see the rest of the samples at any time by disabling the filter.

You can filter for:

- An 8-bit cycle type (represented in the Control row of the Filter config dialog box)
- A program or execution bus address
- A program or execution bus data value
- A data or memory bus address
- A data or memory bus data value
- Readings from any of the 16 external probes
- The high word of the 48-bit timestamp
- The middle word of the 48-bit timestamp
- The low word of the 48-bit timestamp

How the Filter config dialog box works

Use the Filter option in the Trace menu to display the Filter config dialog box to filter out unwanted samples from the display. The parts of the Filter config dialog box are called out in Figure 14–1. Table 14–1 provides descriptions for each part of the dialog box.

Figure 14–1. Filter Config Dialog Box

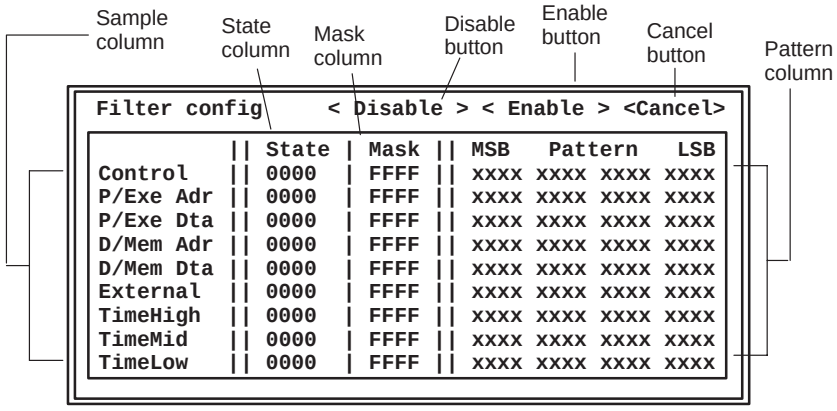


Table 14–1. Filter Config Dialog Box Field Descriptions

Use this part of the dialog box...	To...
Sample column	Select the type of sample you want to see in the TRACE window
State column	Enter the hexadecimal value of the specific sample you want to see in the TRACE window
Mask column	Enter a hexadecimal value to mask bits that do not have to match a particular value
Pattern column	❑ Enter 1s, 0s, or Xs (for don't care bits) to specify a bit pattern ❑ View or change the State and Mask information in binary format
Disable button	Disable the current filter; dismiss dialog box
Enable button	Accept and implement the current filter; dismiss dialog box
Cancel button†	Clear the most recent changes to the filter and return it to the previous filter (if any); dismiss dialog box

† The Cancel button does not disable the filter.

The State, Mask, and Pattern columns work together to filter for a particular sample or group of samples. Values entered in any of these columns automatically update the values in the other columns.

You can also set up a filter by entering a bit pattern in the Pattern column. You can use Xs (for don't care bits) to mask the bits that you do not want to see.

You can narrow your filter by setting up a combination of the options in the Filter config dialog box; the rows are logically ANDed.

How to set up a filter

Once you have updated the contents of the TRACE window (see Section 13.3 on page 13-9), you can filter out the samples that you don't want to see.

In the State column of the Filter config dialog box, type the hexadecimal value of the cycle type, address, data value, external channel, or 16-bit word of the timestamp that you want to display in the TRACE window.

For example, you might want to filter for a specific symbol address, _xcall. To filter for all the occurrences of _xcall, you would set up the filter as follows:

1) Press the HOME key to ensure that you begin filtering from the beginning of the trace buffer.
2) Select Filter from the Trace menu. This displays the Filter config dialog box.
3) Place the cursor in the *P/Exe Adr* row.
4) Press the SPACE key to display the Get Symbol Address dialog box.
5) In the Get Symbol Address dialog box, enter the symbol name:

```
Get Symbol Address
   Case sensitivity:  ( )On  (*)Off

   Symbol:[_xcall                                    ]

                                    <<OK>>   <Cancel>
```

6) Click on OK. This dismisses the Get Symbol Address dialog box. The Filter config dialog box should look something like this:

```
Filter config            < Disable > < Enable > <Cancel>

          || State | Mask || MSB   Pattern   LSB
Control   || 0000  | FFFF || xxxx xxxx xxxx xxxx
P/Exe Adr || 206A  | 0000 || 0010 0000 0110 1010
P/Exe Dta || 0000  | FFFF || xxxx xxxx xxxx xxxx
D/Mem Adr || 0000  | FFFF || xxxx xxxx xxxx xxxx
D/Mem Dta || 0000  | FFFF || xxxx xxxx xxxx xxxx
```

7) Click on the Enable button to enable the filter and dismiss the dialog box.

All occurrences of _xcall in the trace buffer are displayed in the TRACE window.

If your sample is not in the trace buffer, this message appears in the COMMAND window:

```
Trace entry not found
```

You might also see this message if you didn't start the filter from the top of the TRACE window.

How to set up a filter using a mask

If you want to mask certain bits in a filter, enter in the Mask column the hexadecimal mask value for the bits you want the XDS522A to ignore. Fs in the Mask column tell the XDS522A to ignore those bits and filter for the unmasked values in the State column. The Pattern column automatically changes to reflect the values in the State and Mask columns.

You can use the mask to define a range for the filter or to mask specific bits. For example, if you collect some samples and want to filter for any data reads in a specific memory block (memory addresses 0x0300 through 0x030F), you set up the filter as follows:

1) Press the HOME key to ensure that you begin filtering from the beginning of the trace buffer.

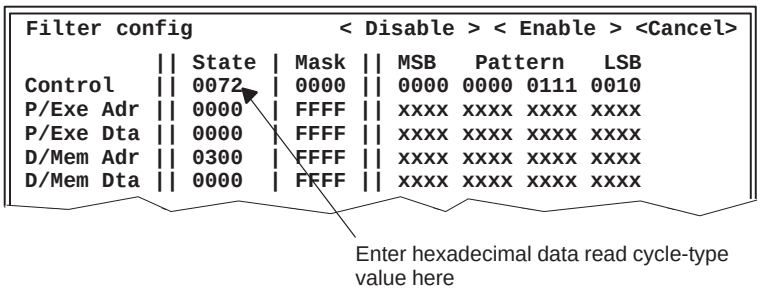
2) Select Filter from the Trace menu. This displays the Filter config dialog box.

3) Identify the hexadecimal value that represents data reads by looking at the CycleTyp column in the TRACE window.

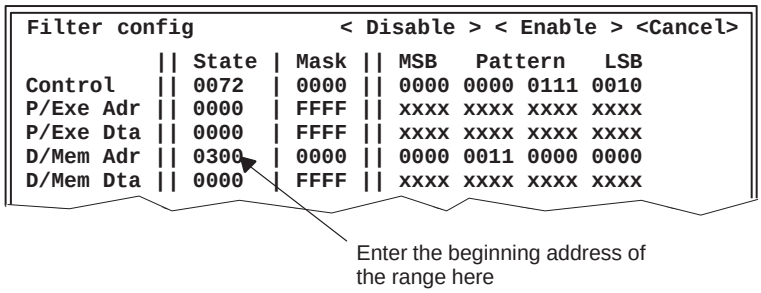
   To configure the TRACE window to display the cycle type in hexadecimal format, see Section 13.6 on page 13-16.

4) Click on the zeros in the *State* column of the Control row. This location is where you will enter the hexadecimal code for the data reads on the memory bus that you want to keep in the TRACE window.

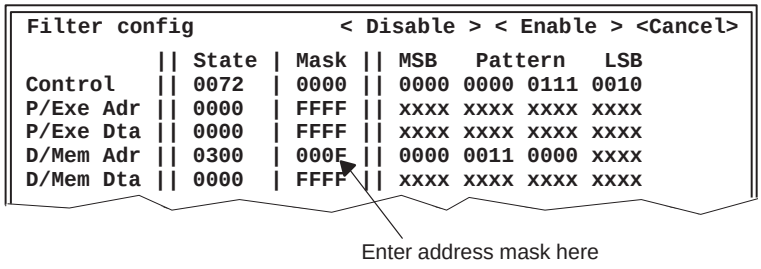
5) Enter **0072**, the hexadecimal cycle-type value representing data reads in this example.



6) In the State column for the D/Mem Adr row, enter **0300** as the initial memory address for the range of memory addresses you want to filter. Notice that the Mask column of the D/Mem Adr row is filled with zeros and the bit pattern is automatically entered. The Filter config dialog box should now look like this:



If you enabled the filter at this point, all the data reads for memory address 0x0300 would be displayed in the TRACE window. However, you want to see all the data reads for memory address locations 0x0300 through 0x030F. Use the Mask column to display more than one memory address.



7) Enter **000F** in the Mask column for the D/Mem Adr row. This value masks the last four bits, which are the only bits that change within the range of 0x0300 through 0x030F.

   The XDS522A compares the values in the State and Mask columns. The zeros in the Mask column tell the XDS522A *not* to mask the first three digits in the State column. The F in the Mask column tells the XDS522A that any value up to F is to be included in the filtered samples.

8) Click the Enable button. This dismisses the Filter config dialog box and applies the filter to the trace samples in the TRACE window.

The samples that appear in the TRACE window are all the data reads within the memory addresses 0x0300 through 0x30F.

If your sample is not in the trace buffer or not in the direction in which you are searching, this message appears in the COMMAND window:

```
Trace entry not found
```

How to set up a filter using a bit pattern

You can also set up a filter by entering a bit pattern in the Pattern column. The Pattern column is a binary representation of the State and Mask columns. When you enter a bit pattern in the Pattern column, the State and Mask columns are updated automatically.

To enter a bit pattern, you can either move the cursor into the Pattern column and type in the binary bit pattern, or use the mouse to toggle the individual bits in this order: x (for don't care bits), 0, 1.

For example, you might want to filter for all the data writes with odd program or execution addresses. You would set up the filter like this:

1) Press the HOME key to ensure that you begin filtering from the beginning of the trace buffer.

2) Select Filter from the Trace menu. This displays the Filter config dialog box.

3) Set up the filter for data writes.

   a) Identify the binary value that represents data writes by looking at the CycleTyp column in the TRACE window.

      To configure the TRACE window to display the cycle type in binary format, see Section 13.6 on page 13-16.

   b) In the Control row, enter the binary cycle-type value for data writes in the Pattern column. The Filter config dialog box should look something like this:

```
Filter config            < Disable > < Enable > <Cancel>

          || State | Mask || MSB   Pattern   LSB
Control   || 0071  | 0000 || 0000 0000 0111 0001
P/Exe Adr || 0000  | FFFF || xxxx xxxx xxxx xxxx
P/Exe Dta || 0000  | FFFF || xxxx xxxx xxxx xxxx
```

   If you enable the filter now, the XDS522A would filter all the data writes and display them in the TRACE window. However, you want to filter for all the data writes that have odd program or execution addresses.

4) Set up the filter for odd program or execution addresses.

   a) In the P/Exe Adr row, place the cursor in the Pattern column.

   b) Toggle the least significant bit (LSB) to 1. The State and Mask columns automatically update. The Filter config dialog box should look like this:

```
Filter config            < Disable > < Enable > <Cancel>

           || State | Mask || MSB   Pattern   LSB
Control    || 0071  | 0000 || 0000 0000 0111 0001
P/Exe Adr  || 0001  | FFFE || xxxx xxxx xxxx xxx1
P/Exe Dta  || 0000  | FFFF || xxxx xxxx xxxx xxxx
```

   If the LSB is set to 1, this turns on the 1s place in the binary pattern. This ensures that only odd values are retained in the filter. If you wanted to retain only even addresses, you would set the LSB to 0.

5) Click the Enable button. This dismisses the Filter config dialog box and applies the filter to the trace samples in the TRACE window.

All the data writes that have odd program or execution addresses are displayed in the TRACE window.

If your sample is not in the trace buffer, this message appears in the COMMAND window:

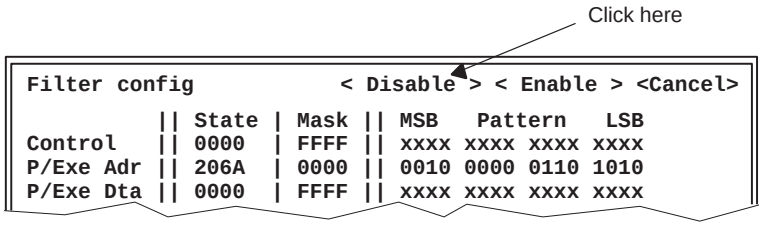
```
Trace entry not found
```

Disabling a filter

If you want to disable the filter and return all of the samples to the TRACE window:

1) Select Filter from the Trace menu.

2) Click on Disable.



This dismisses the Filter config dialog box; the TRACE window displays all of the samples again.

Even after a disable, the Filter config dialog box retains all the information from the last filter. To start a new filter, you must clear out the values from the previous filter. To clear the previous filter, be sure the State column is filled with 0s and the Mask column is filled with Fs. The Pattern column will automatically clear when these values are entered in the State and Mask columns.

14.2 Searching for a Specific Trace Sample

The XDS522A emulation system has a search feature that you can use to search for specific samples in the TRACE window.

You can search for:

- A cycle type (represented in the Control row of the Search config dialog box)
- A program or execution bus address
- A program or execution bus data value
- A data or memory bus address
- A data or memory bus data value
- Readings from any of the 16 external probes
- The high word of the 48-bit timestamp
- The middle word of the 48-bit timestamp
- The low word of the 48-bit timestamp

How the Search config dialog box works

Use the Search option in the Trace menu to display the Search config dialog box to search for a specific sample. Figure 14–2 calls out the parts of the Search config dialog box. Table 14–2 gives a description of each of the Search config dialog box fields.

Figure 14–2. Search Config Dialog Box

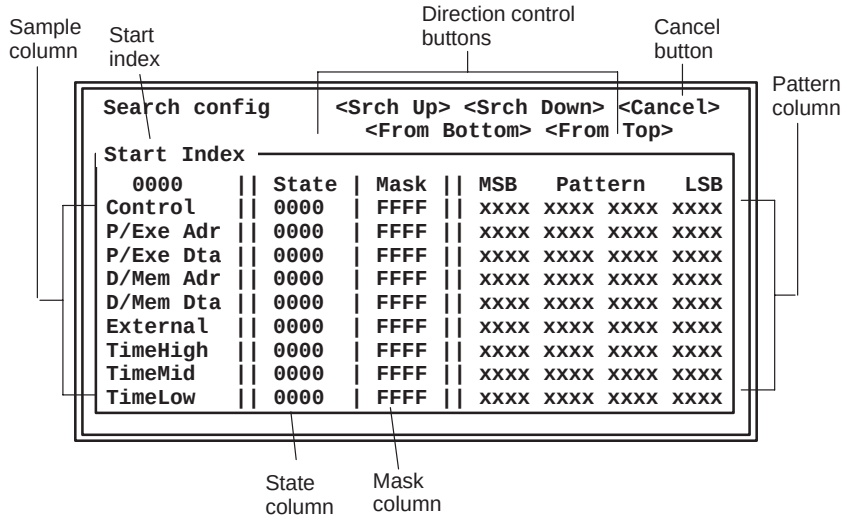


Table 14–2. Search Config Dialog Box Field Descriptions

Use this part of the dialog box...	To...
Sample column	Select the type of sample you want to see in the TRACE window
Start index	Enter the number of the sample on which you want the search to begin. The default start index is 0000.
State column	Enter the hexadecimal value of the specific sample that you want to see in the TRACE window
Mask column	Enter a hexadecimal value to mask bits that do not have to match a particular value
Pattern column	❑ Enter 1s, 0s, or Xs (for don't care bits to specify a bit pattern ❑ View or change the State and Mask information in binary format
Direction control buttons	Determine the direction and origin of the search
Srch Up	Search the trace samples that were collected before the trace sample specified in the Start Index field; dismiss dialog box
Srch Down†	Search the trace samples that were collected after the trace sample specified in the Start Index field; dismiss dialog box
From Bottom‡	Search the trace buffer from the newest samples to the oldest samples; dismiss dialog box
From Top‡	Search the trace buffer from the oldest samples to the newest samples; dismiss dialog box
Cancel button	Clear most recent changes to the filter and return it to the previous filter; dismiss the dialog box

† This option is invalid if the Start Index is 0000.
‡ These options ignore any values entered in the Start Index field

The State, Mask, and Pattern columns work together to search for a particular sample or type of sample. Values entered in any of these columns automatically update the values in the other columns.

You can also set up a search by entering a bit pattern in the Pattern column. You can use Xs (for don't care bits) to mask the bits that you do not want to see.

You can narrow your search by setting up a combination of the options in the Search config dialog box; the rows are logically ANDed.

How to search for a specific sample

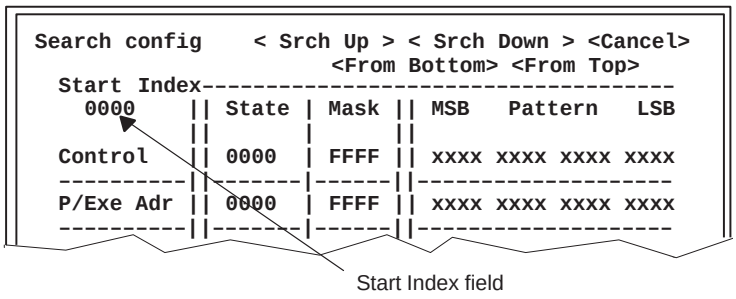
Once you have updated the contents of the TRACE window (see Section 13.3 on page 13-9), you can search the TRACE window for a specific sample:

1) From the Trace menu, select Search.

   This displays the Search config dialog box, in which you set up specific search criteria.

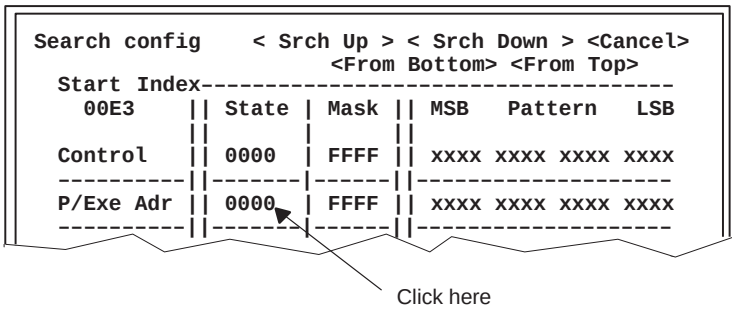
2) Set up the start index by typing the hexadecimal value for the sample number on which you want your search to begin. (This example will use 00E3 for the Start Index.)

   This allows you to limit your search when you do not want to find every occurrence of a sample in the trace buffer. By default, the start index is set to 0000. Note that if the Start Index is set to 0000, you cannot search down (as there are no samples below 0000).



3) Specify the criteria to search for in the trace buffer. For example, if you want to find a sample with a particular symbol address, you would set up the Search config dialog box in the following manner:

   a) Click on 0000 in the State column of the address row. This location is where you enter the program or execution address of the sample that you want to search for.



   b) Press the SPACE key. This displays the Get Symbol Address dialog box.

c) In the Get Symbol Address dialog box, enter the symbol name:

```
Get Symbol Address
   Case sensitivity:  ( )On  (*)Off

   Symbol:[_call                                    ]

                                    <<OK>>   <Cancel>
```

d) Click on OK. This dismisses the Get Symbol Address dialog box. The Search config dialog box should look like this:

```
Search config    < Srch Up > < Srch Down > <Cancel>
                          <From Bottom> <From Top>
  Start Index--------------------------------------
    00E3    || State | Mask || MSB   Pattern   LSB
            ||       |      ||
  Control   || 0000  | FFFF || xxxx xxxx xxxx xxxx
  ----------||-------|------||--------------------
  P/Exe Adr || 205A  | 0000 || 0010 0000 0101 1010
  ----------||-------|------||--------------------
```

The Mask column of the P/Exe Adr row is filled with zeros. You can use the Mask column to mask specific bits. By masking bits, you tell the system to ignore those bits when it searches for a bit pattern.

The XDS522A compares the values in the State and Mask columns. The zeros in the Mask column tell the XDS522A *not* to mask any of the digits in the State column.

4) Select the appropriate search direction button.

Select Srch Up or Srch Down if you want to search the samples before or after the sample number in the Start Index. The From Bottom and From Top buttons search the entire trace buffer regardless of the Start Index. Note that these buttons ignore any value entered in the Start Index.

When you select a search direction button, the Search config dialog box is dismissed and the sample or samples you are searching for appears at the top of the TRACE window.

If your sample is not in the trace buffer or not in the direction in which you are searching, this message appears in the COMMAND window:

```
Trace entry not found
```

14.3 Using the Timestamp to Gather Timing Information

The XDS522A allows you to gather timing information to determine when a trace sample is collected, when one sample is collected in relation to another sample, how long it takes a loop to execute, etc.

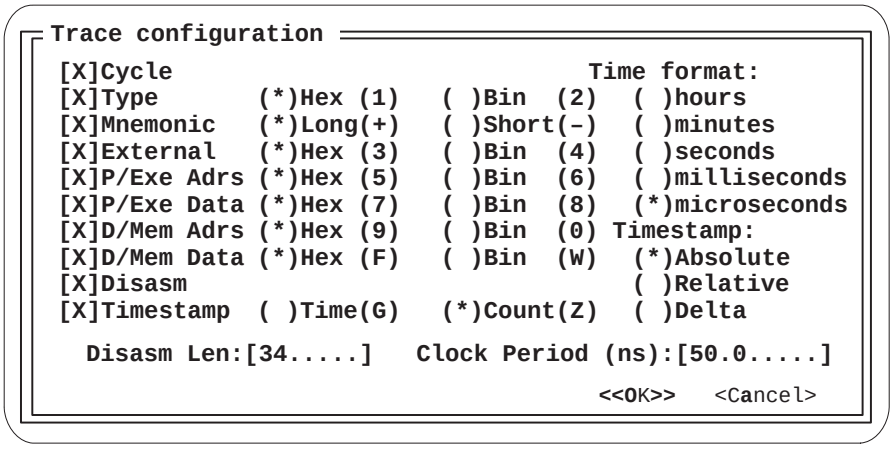
The XDS522A uses the base cycle clock to generate a timestamp. The base clock determines how many cycles per second the CPU processes (assuming the CPU is not in power-down mode).

Displaying the timestamp in the TRACE window

You have many alternatives in setting up the timestamp display in the TRACE window. For example, you can display the timestamp in hours, minutes, seconds, milliseconds, or microseconds.

Follow these instructions to configure your timestamp for display:

1) Select Config from the Trace menu. This displays the Trace configuration dialog box:



2) Because the TRACE window cannot display all of the information that you can set up at one time, you must limit the display when you use the timestamp option by deselecting any options that you will not be using. If you do not limit the display, the timestamp will not appear in your TRACE window.

3) Be sure the *Timestamp* option is selected.

4) Select one of the following options:

Time	Displays units of time
Count	Displays the number of clock cycles

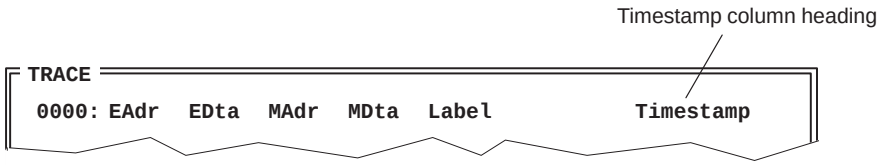
5) If you chose *Time* in step 4, you need to choose the format in which you want the time displayed—hours, minutes, seconds, milliseconds, or microseconds.

6) Select a display mode for your timestamp:

Absolute	Shows the time relative to the first sample that you collected
Relative	Shows the time relative to a sample that you select by clicking on that sample in the TRACE window
Delta	Shows the time relative to the previous sample in the TRACE window

7) Click on OK. This dismisses the Trace configuration dialog box.

Notice that the column headings in your TRACE window have automatically been updated with your new configuration:

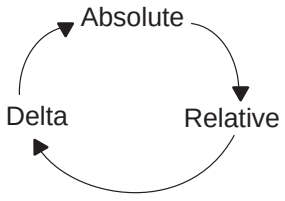


You can change any of the timestamp options and view the updated results in the TRACE window without having to rerun the program.

Toggling through the timestamp display modes

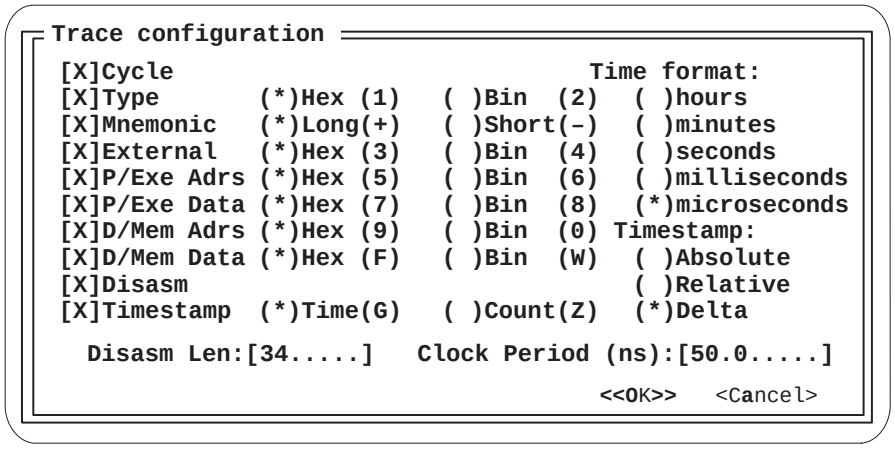
You can quickly change your display mode by toggling through the three display modes (Absolute, Relative, and Delta). To toggle through the display modes, click on the Time menu button or press F7.

The mode that you see first depends on the configuration of the timestamp in the Trace configuration dialog box. For example, if the timestamp configuration is set to Absolute, toggling the display mode changes the timestamp to Relative, then Delta, then back to Absolute again:



Determining your timestamp display mode

Until you become familiar with the three timestamp display modes, you may not recognize which mode you are displaying at any given time. If this happens, select Config from the Trace menu to display the Trace configuration dialog box. The Timestamp portion of the Trace configuration dialog box displays the current mode.



Chapter 15

Counters

The BTT software provides you with two 16-bit counters that can be configured to interact with other parts of the BTT setup window.

This chapter explains the operation of the counters, how to clock them using various qualifiers, how to use both counters to configure a 32-bit counter, and how to configure a counter to behave as a watchdog timer.

Topic **Page**

15.1 How the Counters Work

The XDS522A emulation system provides you with two 16-bit counters, Counter 1 and Counter 2. Having two counters offers you the convenience of setting up the system several ways: you can use one counter at a time, both counters at once, or one counter's output as input for the other counter.

How you control the counters

You can use the counters to count clock cycles, events, and other BTT functions. You can also define a 32-bit counter by configuring the zero output of one counter as an input to the other. Table 15–1 describes how the counters work.

Table 15–1. How You Can Control the Counters

You can control…	**By specifying**…
What they count	❑ Clock cycles ❑ Events ❑ How many times a sequence completes ❑ How many times the other counter reaches 0
When they start	❑ Immediate ❑ Event A, B, C, or D ❑ When the other counter reaches 0 ❑ When the specified sequence completes
Their initial values	❑ The default initial value of 0xFFFF ❑ A value you enter in the Initial value field
When they stop	❑ Event A, B, C, or D ❑ When the other counter reaches 0 ❑ When the specified sequence completes
When they reload	❑ Event A, B, C, or D ❑ When the other counter reaches 0 ❑ When the specified sequence completes
What happens next	❑ Advance to the next sequence level ❑ Decrement, start, stop, or reload the other counter ❑ Start, stop, or disable tracing ❑ Generate a hardware breakpoint ❑ Generate a trigger pulse

Precedence of counter inputs

If inputs to the counters occur in the same clock cycle, the order of precedence is as follows:

1) Start
2) Reload
3) Clock
4) Stop

What the counter fields do

Each of the two 16-bit counters has nine fields that allow you to control the operation of the counters and how the counters interact with other parts of the BTT setup window. Figure 15–1 labels each counter field. For a description of each field see Table 15–2.

Figure 15–1. Counter Fields in the COUNTER Box

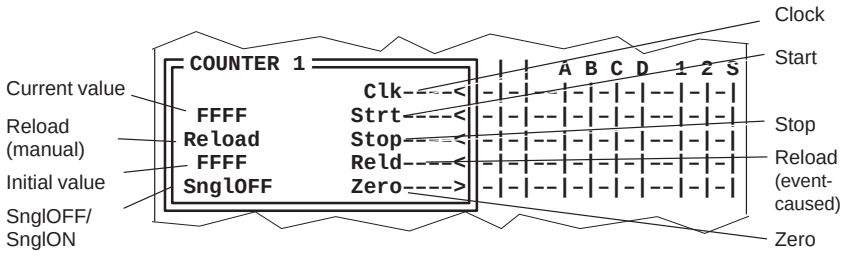


Table 15–2. Counter Box Field Descriptions

If you want to...	Use this field	To...
Set up what you want to count	Clk	❑ Count clock cycles ❑ Count any of Events A–D ❑ Count the number of times the other counter decrements to 0 ❑ Count completed sequences
Start the counter	Strt	❑ Start Immediate ❑ Start on any of Events A–D ❑ Start when the other counter decrements to 0 ❑ Start when the sequence completes
Stop the counter	Stop	❑ Stop on any of Events A–D ❑ Stop when the other counter decrements to 0 ❑ Stop when the sequence completes
Reload the counter with its initial value when an event or other condition occurs	Reld	❑ Reload on any of Events A–D ❑ Reload when the other counter decrements to 0 ❑ Reload when the sequence completes
Manually reload the counter with its initial value	Reload	Reload the counter with its initial value after the program has stopped running
See the current value of the counter	FFFF (current value)	
Set up whether or not the counter decrements multiple times	SnglON/SnglOFF	❑ Decrement to 0 once (SnglON) ❑ Decrement to 0 multiple times (SnglOFF)
Set up an initial value for the counter	FFFF (initial value)	Count a specific number of: ❑ Clock cycles ❑ Events ❑ Times the other counter decrements to 0 ❑ Sequence completions
Set up an output action for the counter	Zero	❑ Advance to the next sequence level ❑ Decrement, start, stop, or reload the other counter ❑ Start, stop, or disable tracing ❑ Generate a hardware breakpoint ❑ Generate a trigger pulse

15.2 Counting a Specific Number of Events

You can use the counters to count a specific number of occurrences of one event. Perhaps you want to count a specific number of occurrences of Event A, then stop tracing to look at that part of your program.

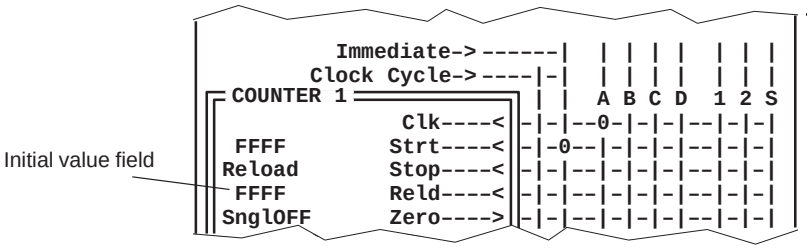
1) Set up Event A and any other events you plan to monitor. For information on setting up events, see Chapter 12, *Defining Events.*

2) Set up Counter 1 to be clocked by Event A by clicking on the intersection of *Clk* and *A.*

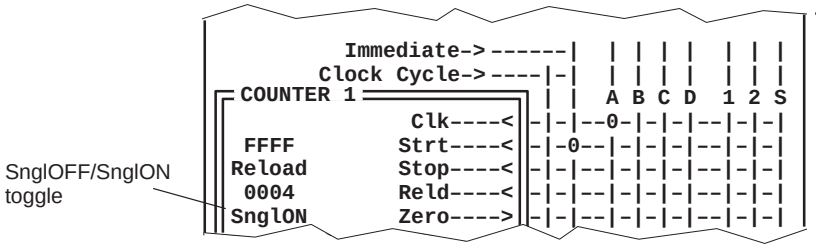
3) Set up your start event in the COUNTER 1 box by clicking on the intersection of *Strt* and any of the following possible start events:

   - ❑ Immediate
   - ❑ Any of Events A through D
   - ❑ When the other counter reaches 0
   - ❑ When the specified sequence completes

4) Set the initial value for the counter by clicking in the Initial value field. Type in the hexadecimal number you want for the initial value. The counter begins decrementing from this value.



5) Set up Counter 1 as a single-shot counter by toggling the SnglOFF/SnglON toggle to SnglON:



SnglON makes the counter a single-shot counter. The counter decrements to 0 once, then stops.

6) Set up the counter so that its output causes the BTT to perform some action when the counter decrements to zero. For this example, you want to stop tracing when you have counted a specific number of occurrences of Event A. To stop tracing when the counter decrements to zero, in the TRACE box, click on the intersection of *Disa* and *1*. Other possible output actions include:

   - ❑ Starting the second counter
   - ❑ Advancing the sequencer
   - ❑ Starting or disabling tracing
   - ❑ Generating a hardware breakpoint
   - ❑ Generating a trigger pulse

7) Download the BTT setup window configuration and be sure that the XDS522A is enabled.

8) Start your application.

In this example, when the start event occurs, the counter begins counting every occurrence of Event A. When the counter counts the desired number of occurrences of Event A, its value is 0, which is the condition you defined to disable tracing.

Note: Counting More Than One Event

You can count two distinct events by using both counters. Set up Counter 1 for one event and Counter 2 for the second event.

15.3 Counting Clock Cycles

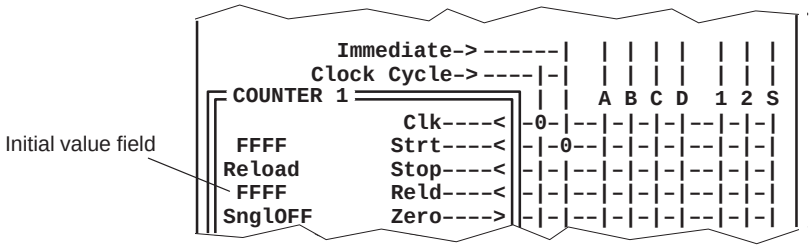
You can use the counters to count clock cycles.

1) Set up Counter 1 to be clocked by clock cycles by clicking on the intersection of *Clock Cycle* and *Clk*.

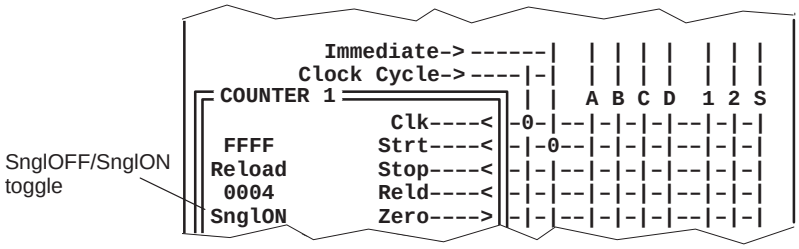
2) Set up your start event in the COUNTER 1 box by clicking on the intersection of *Strt* and any of the following possible start events:

   - ❑ Immediate
   - ❑ Any of Events A through D
   - ❑ When the other counter reaches 0
   - ❑ When the specified sequence completes

3) Set the initial value for the counter by clicking in the Initial value field. Type in the hexadecimal number you want for the initial value. The counter begins decrementing from this value.



4) Set up Counter 1 as a single-shot counter by toggling the SnglOFF/SnglON toggle to SnglON:



5) Set up the counter so that its output causes the BTT to perform some action when the counter decrements to zero. Possible output actions include:

   - ❏ Starting the second counter
   - ❏ Advancing the sequencer
   - ❏ Starting, stopping, or disabling tracing
   - ❏ Generating a hardware breakpoint
   - ❏ Generating a trigger pulse

6) Download the BTT setup window configuration and be sure that the XDS522A is enabled.

7) Start your application.

15.4 Counting the Number of Times a Sequence Completes

The counters can accept input from the sequencer. This allows you to count the number of times a sequence completes. To set up the counter to count sequence completions:

1) Define the events that you want to use in a sequence. For information on defining events, see Chapter 12, *Defining Events*.

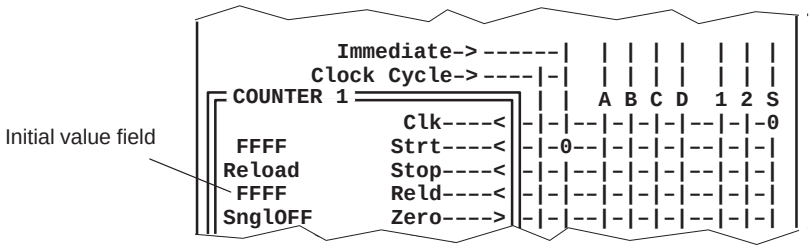
2) In the SEQUENCER box, set up the sequence of events you want the counter to be clocked by. See Chapter 16, *Sequencer*, for information on how to set up a sequence.

3) In the COUNTER box, set up the counter to be clocked by sequencer completions by clicking on the intersection of *Clk* and *S*.

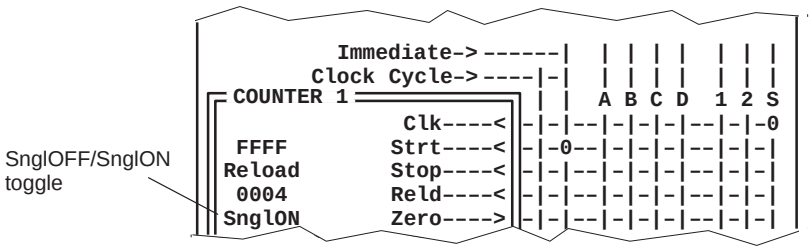
4) Set up your start event in the COUNTER box by clicking on the intersection of *Strt* and any of the following possible start events:

   - ❑ Immediate
   - ❑ Any of Events A through D
   - ❑ When the other counter reaches 0
   - ❑ When the specified sequence completes

5) Set the initial value for the counter by clicking in the Initial value field. Type in the hexadecimal number you want for the initial value. The counter begins decrementing from this value.



6) Set up the counter to use either SingleOff or SingleOn by toggling the SnglOFF/SnglON toggle.



7) Set up the counter so that its output causes the BTT to perform some action when the counter decrements to zero. Possible output actions include:

   - ❑ Starting the second counter
   - ❑ Advancing the sequencer
   - ❑ Starting, stopping, or disabling tracing
   - ❑ Generating a hardware breakpoint
   - ❑ Generating a trigger pulse

8) Download the BTT setup window configuration and be sure that the XDS522A is enabled.

9) Start your application.

15.5 Configuring a 32-Bit Counter

You can use the two 16-bit counters together to create one 32-bit counter. By using one counter to count the other, you can count FFFF × FFFF clock cycles, events, or completed sequences. To set up a 32-bit counter, you must configure both Counter 1 and Counter 2.

Setting up COUNTER 1

To set up Counter 1 as part of a 32-bit counter:

1) Define the events you plan to use with the counters. For information on setting up events, see Chapter 12, *Defining Events*.

2) Set up Counter 1 to clock any of:

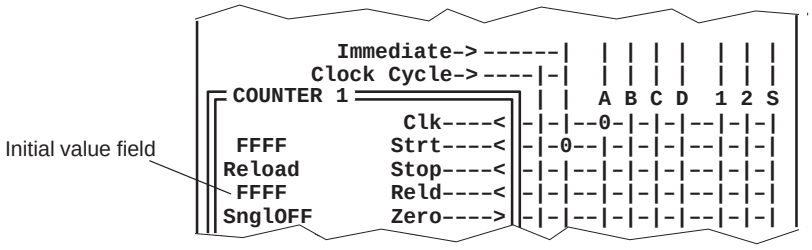
   - ❑ Clock cycles
   - ❑ Events A–D
   - ❑ Completed sequences

   Click on the intersection of *Clk* and any of the above qualifiers.

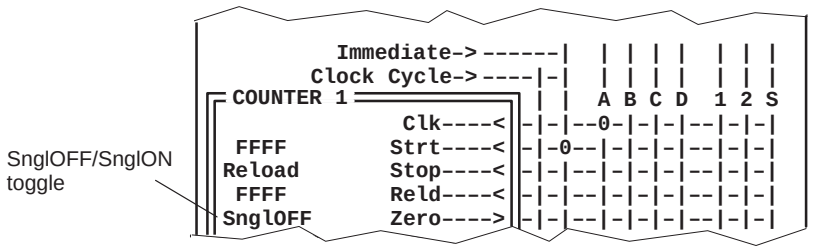
3) Set up your start condition for Counter 1 by clicking on the intersection of *Strt* and any of the following possible start events:

   - ❑ Immediate
   - ❑ Any of Events A through D
   - ❑ When the specified sequence completes

4) Set up the initial value for Counter 1 to FFFF.



5) Set up Counter 1 as a continuously running counter by toggling the SnglOFF/SnglON toggle to SnglOFF:

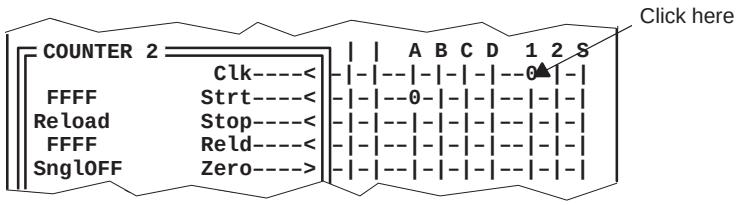


Setting up COUNTER 2

To set up Counter 2 as part of a 32-bit counter:

1) Set up a start event for Counter 2.

2) Set up Counter 2 to clock on Counter 1 by clicking the intersection of Clk and 1.



3) Set up the initial value for Counter 2 to FFFF.

4) Set up Counter 2 so that its output causes the BTT to perform some action when the counter decrements to zero. Possible output actions include:

- ❑ Starting, stopping, or disabling tracing
- ❑ Advancing the sequencer
- ❑ Generating a hardware breakpoint
- ❑ Generating a trigger pulse

5) Download the BTT setup window configuration and be sure that the XDS522A is enabled.

6) Start your application.

15.6 Configuring a Counter to Behave as a Watchdog Timer

You can use a BTT counter to configure a watchdog timer, which is useful in a real-time application to do either of the following:

- ❑ Ensure that a time-critical deadline is always met
- ❑ Generate a signal if the processor is not executing the code correctly

Setting up the debugger

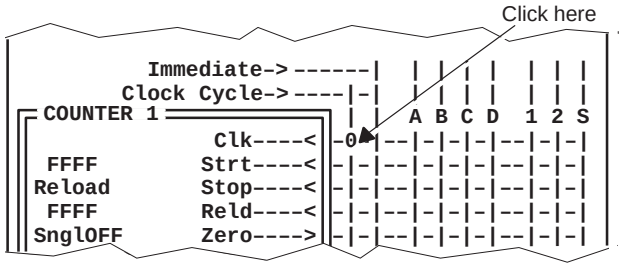
For the counter to be able to simulate a watchdog timer, the debugger's on-chip analysis module must be set up to recognize and generate hardware breakpoints. For more information, see the *Setting up the analysis module to generate breakpoints* subsection on page 3-6.

Setting up the BTT counter

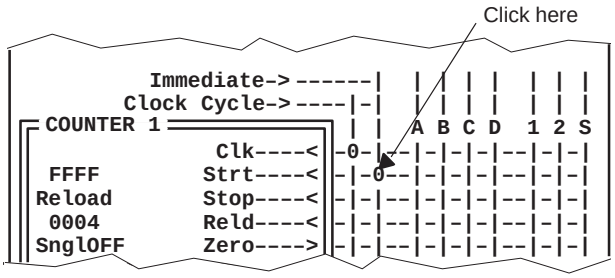
To configure either counter to behave as a watchdog timer:

1) Define the events you plan to "watchdog." For information on setting up events, see Chapter 12, *Events*.

2) Set up the counter to be clocked on clock cycles by clicking on the intersection of *Clk* and *Clock Cycle*.



3) Set up your start condition as Immediate by clicking on the intersection of *Strt* and *Immediate*.

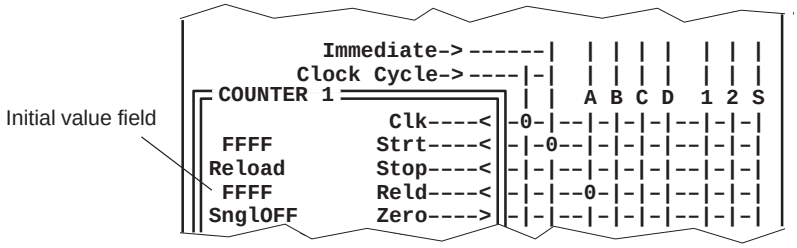


4) Set up the reload condition for the watchdog timer. The reload condition should be the event you want to "watchdog." This event can be:

   - ❑ Any of Events A through D
   - ❑ When the other counter reaches 0
   - ❑ When the specified sequence completes

   Click on the intersection of *Reld* and the reload condition.

5) Set up the reset value for the watchdog timer. In the initial value field, enter the maximum number of clock cycles that should occur before the reload condition occurs.



6) Set up the counter to generate a hardware breakpoint if it decrements to 0. In the ACTION box, click on the intersection of *Break* and *1* or *2* (depending on which counter you use as the watchdog timer). The breakpoint halts the processor.

   If your application does not make it feasible for you to stop the processor, you can set a trigger. Click on the intersection of *Trigger* and *1* or *2*.

7) Download the BTT setup window configuration and be sure that the XDS522A is enabled.

8) Start your application.

If the "watchdog" event does not occur within the count specified, a hardware breakpoint (or trigger pulse) occurs.

Chapter 16

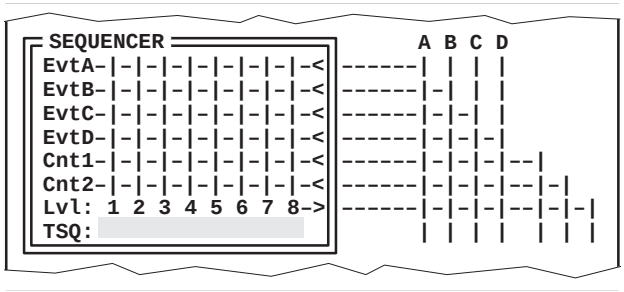
Sequencer

When you want to detect when a series of events occur in a specified order, use the sequencer. This chapter describes how you can set up the sequencer to detect a series of events and how to use the sequencer output to control other parts of the XDS522A emulation system.

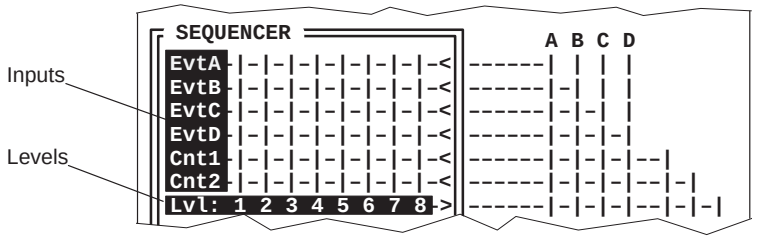
Topic **Page**

16.1 How the Sequencer Works

To set up a sequence of events for the XDS522A to look for, use the SEQUENCER box in the BTT setup window:



The sequencer operates as an 8-level state machine. Up to six inputs are ORed to determine whether the sequencer should advance to the next level:



A sequence can be defined with any of these qualifiers:

- ❏ One or more events (A, B, C, or D)
- ❏ One or both counters equal 0

You specify the sequence by assigning a level to each input. The first input in the sequence is Level 1, the second is Level 2, and so on. The system starts by searching for the qualifier that you have set up at Level 1—your Level-1 event. Then, it searches for your Level-2 event. The system continues searching until all of the levels that you have defined have been detected. Then, the system begins searching for the Level-1 event again.

When the sequence is detected, you can do any of the following:

- ❏ Start, stop, decrement, or reload a counter
- ❏ Start, stop, or disable tracing
- ❏ Generate a hardware breakpoint
- ❏ Generate a trigger pulse

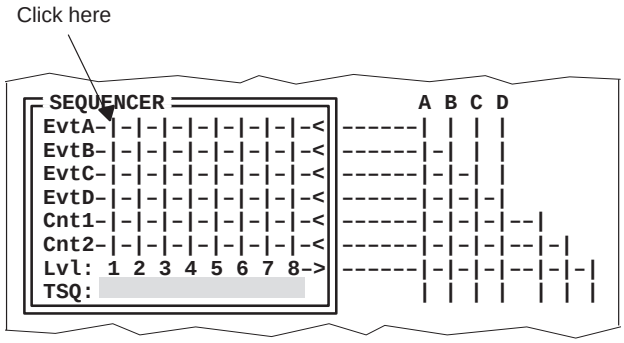
16.2 Detecting a Sequence

Use the sequencer to detect when qualifiers occur in a specified order. These qualifiers can be an occurrence of Event A, B, C or D or when one of the counters equals 0.

Setting up a sequence

To set up a sequence, start by defining the qualifiers that you want to use in the sequence. Then, set up the sequencer to look for the qualifiers in the order that you specify. For example, to set up the system to look for Event A, followed by Event B, followed by Counter 1 = 0, do the following:

1) Set up Event A as the first event in the sequence, the Level-1 event, by clicking on the intersection of *EvtA* and 1.

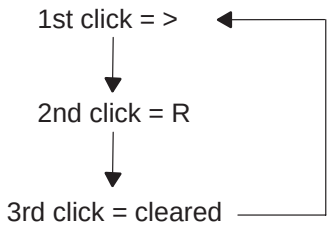


A sequence connection symbol (>) appears at the intersection.

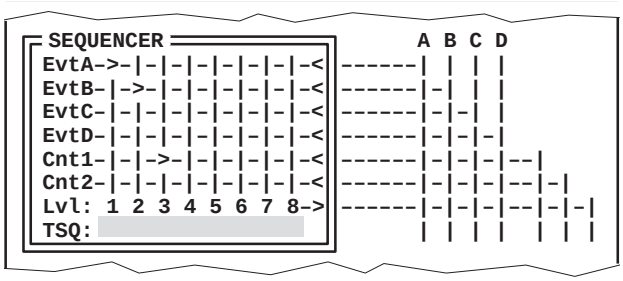
2) Set up Event B as the second event in the sequence by clicking on the intersection of *EvtB* and 2.

3) Set up Counter 1 = 0 as the third event in the sequence by clicking on the intersection of *Cnt1* and 3.

If you accidentally click on an intersection twice, a reset connection symbol (R) appears. If this happens, click on the intersection two more times to replace the reset connection symbol with the sequence connection symbol. (For more information about the reset connection symbol, see Section 16.4.)



In this example, the SEQUENCER box would be set up like this:



This setup tells the system to do the following:

1) Look for the Level-1 event (Event A).
2) Look for the Level-2 event (Event B).
3) Look for the Level-3 event (Counter 1 = 0).

After Counter 1 decrements to 0, the sequence is detected.

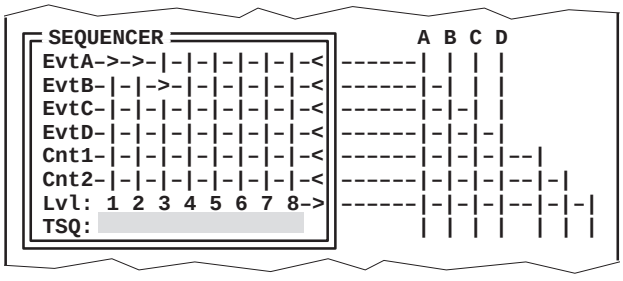
Note: Intervening Events are Ignored

The sequencer searches for a particular sequence of events without regard to what happens between the defined events. In this example, if the counter decrements to 0 after Event A and before Event B, that occurrence of the counter decrementing to 0 is ignored. The system waits until Event B occurs before it looks for the counter to decrement to 0.

Part III

Using the same event more than once in a sequence

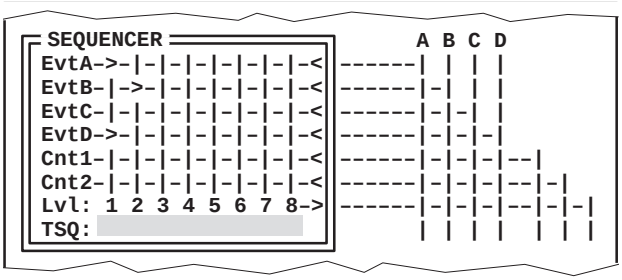
You can configure the sequencer to look for the same event at different levels. For example, you could set up the following sequence: Event A, followed by Event A, followed by Event B. Your SEQUENCER box would look like this:



The system would look for two occurrences of Event A followed by one occurrence of Event B.

Setting up the sequencer to look for either of two events

Because the sequencer inputs are ORed, you can set up two events on the same level; the sequencer looks for one event or the other. For example, if you want to look for Event A or Event D followed by Event B, you would set up your SEQUENCER box like this:



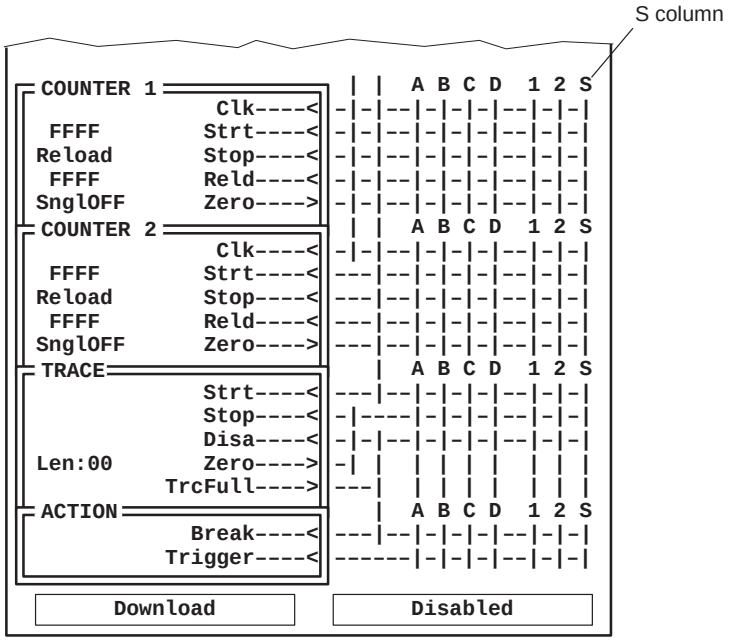
First, the system would look for Event A or Event D. Then, it would look for Event B.

16.3 After the Sequence Is Detected

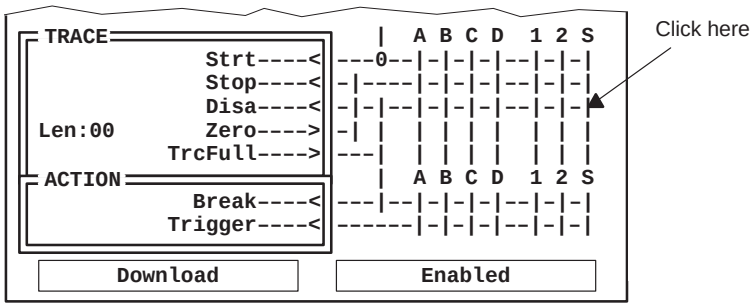
When the sequence is detected, you can do any of the following:

- Decrement, start, stop, or reload a counter
- Start, stop, or disable tracing
- Generate a hardware breakpoint
- Generate a trigger pulse

To set up the completion of the sequence to do any of these things, use the S column in the BTT window:



For example, to disable tracing when the sequencer has detected the defined sequence, click on the intersection of *Disa* and S:



A connection symbol (0) appears at the intersection.

16.4 Resetting the Sequencer

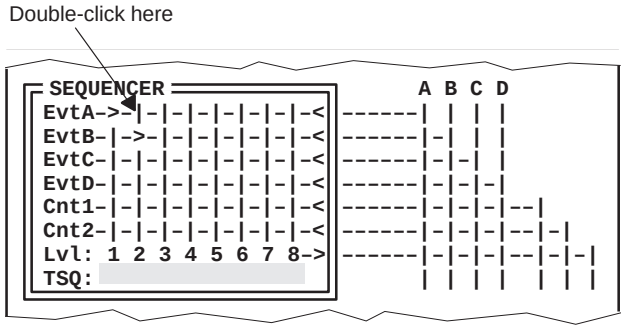
Any input (Event A, B, C, D or when one of the counters equals 0) can reset the sequencer. Resetting the sequencer means that the sequencer returns to Level 1 and begins monitoring events from that level.

For example, you want to set up the system to look for Event A, followed by Event B. If Event A occurs a second time before Event B occurs, you want to tell the system to start the sequencer over again. This is how you would set up the system to detect that series of events:

1) Set up Event A as the Level-1 event.

2) Set up Event B as the Level-2 event.

3) Even though you have already set up Event A as the Level-1 event, you can also set it up as the reset event by double-clicking on the intersection of *EvtA* and 2.



A reset connection symbol (R) appears at the intersection of *EvtA* and 2.

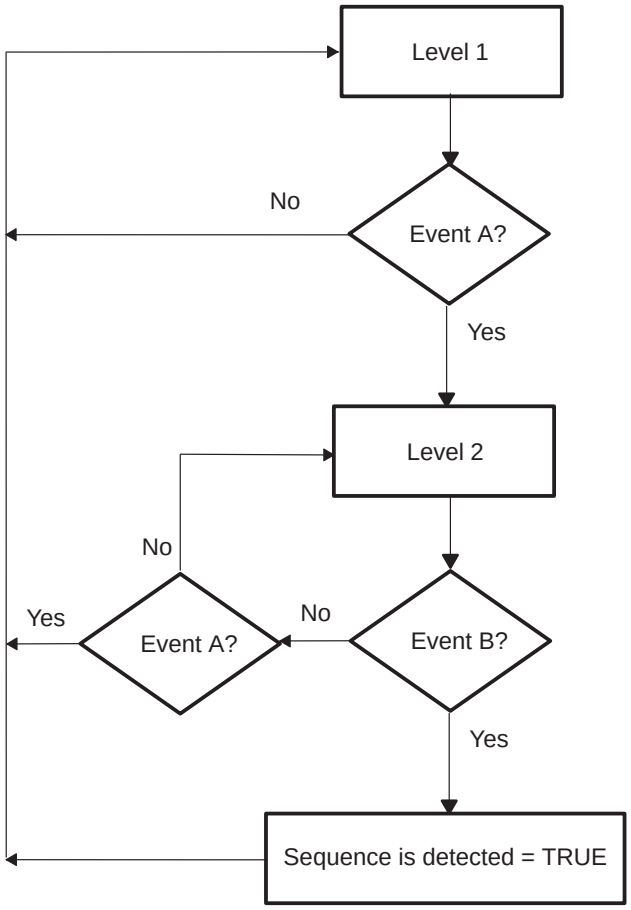
This setup tells the system to do the following:

1) Look for the Level-1 event.

2) Look for the Level-2 event. If the reset event occurs before the Level-2 event, start looking for the Level-1 event again.

Figure 16–1 illustrates how this setup works.

Figure 16–1. Sequencer Reset Operation

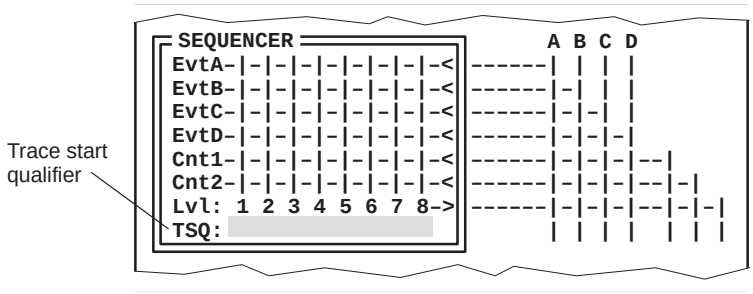


The precedence of sequencer inputs

If the reset event occurs at the same time as the event that advances the sequence, the latter has priority.

16.5 Starting Trace During a Specified Sequencer Level

The sequencer includes a trace start qualifier (TSQ). Use the TSQ to set up a condition that allows trace to start only during specified sequencer levels.

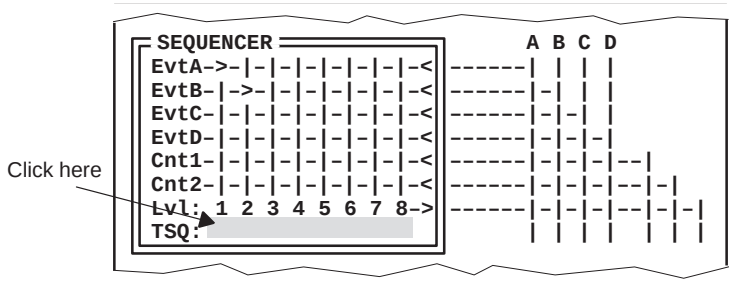


By default, TSQ is enabled for all levels of the sequencer. As a result, trace can start at any level. The TSQ works in conjunction with the sequencer to add additional trace control. Use the TSQ when you want tracing to start only during specified sequencer levels. For example, you might want to start tracing on Event C, but only when Event C occurs between Events A and B:

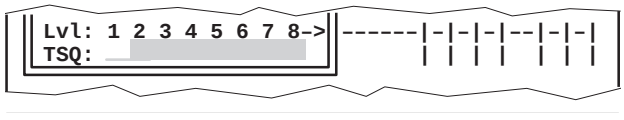
1) Set up Event A as the Level-1 event.

2) Set up Event B as the Level-2 event.

3) Set up the TSQ on Level 2 by clicking on the intersection of 1 and TSQ:



Level 1 should be the only level that is *not* highlighted in the TSQ row:



4) In the TRACE box, set up tracing to start on Event C. That makes Event C the TSQ event.

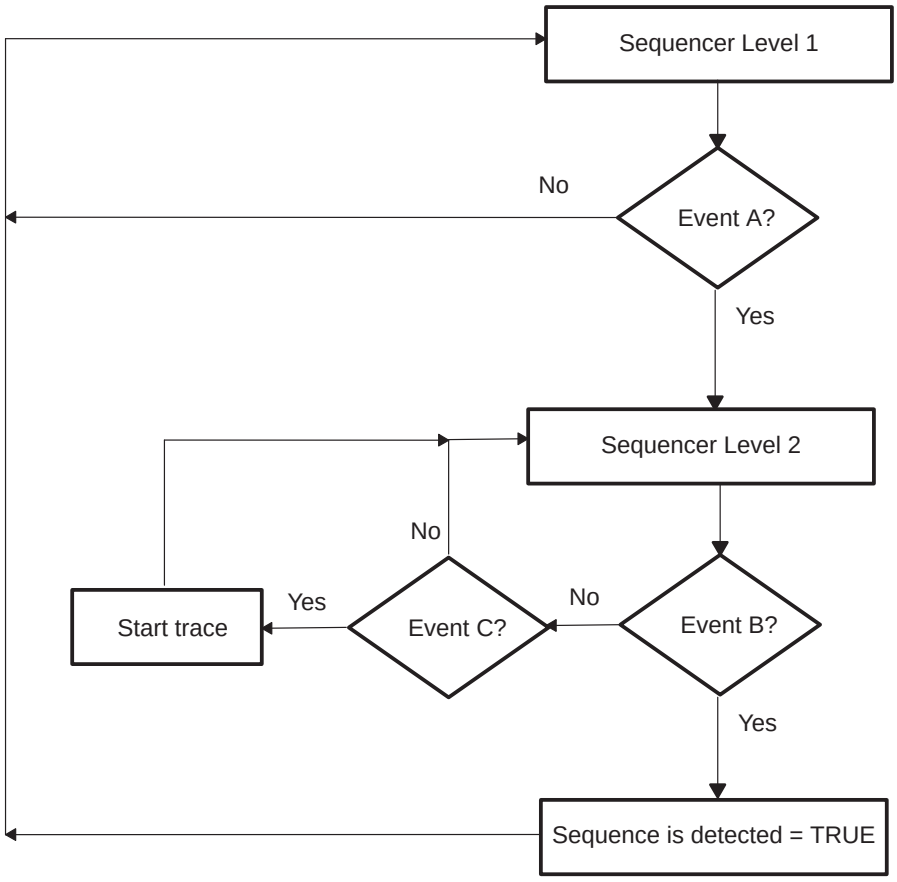
This setup tells the system to do the following:

1) Look for the Level-1 event (Event A).

2) Look for the Level-2 event (Event B). If the TSQ event, Event C, occurs between the Level-1 event and the Level-2 event, start tracing on Event C.

Figure 16–2 illustrates how this setup works.

Figure 16–2. TSQ Operation



Chapter 17

Breakpoints and Triggers

The XDS522A allows you to monitor activities of the CPU. When a specified event occurs, the XDS522A can generate a hardware breakpoint or generate a trigger pulse. You use the ACTION box in the BTT setup window to tell the XDS522A when you want to generate a hardware breakpoint or trigger pulse.

Topic **Page**

17.1 Hardware Breakpoints

The BTT software allows you to define when the XDS522A will generate a hardware breakpoint. The XDS522A monitors events as they occur in real time; when a specified event occurs, the XDS522A can force a breakpoint on the EMU0 pin of the target device.

You can generate a breakpoint with one or more of these conditions:

- ❑ When the trace buffer is full
- ❑ When event A, B, C, or D occurs
- ❑ When a counter reaches zero
- ❑ When a sequence of events completes

Generating a hardware breakpoint

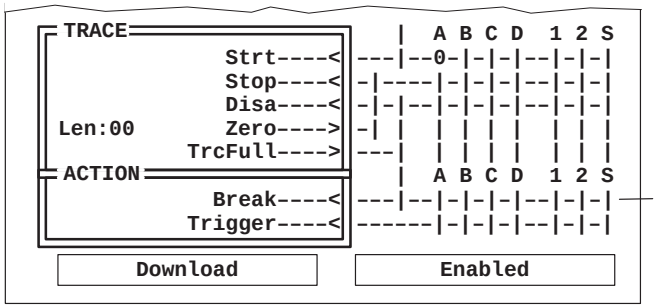
The ACTION box of the BTT setup window allows you to specify what the XDS522A should do on detection of a certain set of conditions. You use the ACTION box to direct the XDS522A to generate a hardware breakpoint.

To generate a hardware breakpoint, do the following:

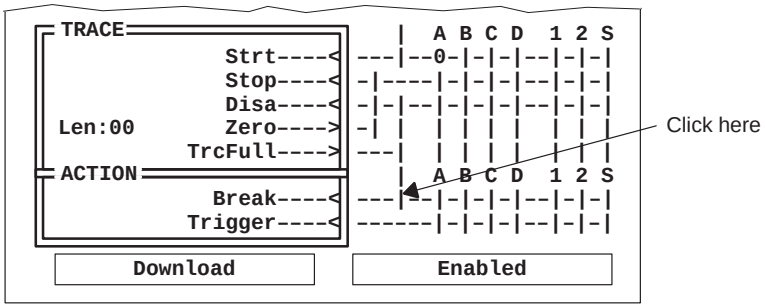
1) Unless you want to generate a breakpoint when the trace buffer is full, define the event that you want to use to generate the hardware breakpoint. Your event can be:

   - ❑ Any of Events A through D
   - ❑ A counter reaches 0
   - ❑ A sequence completion

2) Set up the ACTION box to generate a hardware breakpoint by clicking on the intersection of *Break* and the condition that you want to monitor:



For example, to generate a hardware breakpoint when the trace buffer is full, click on the intersection of *Break* and *TrcFull*:



If you click on more than one intersection in the *Break* line, the conditions are logically ORed.

3) Download the XDS522A configuration and be sure that the XDS522A is enabled.

4) Set up the debugger's on-chip analysis module to recognize and generate hardware breakpoints. For information about setting up the analysis module, see the *TMS320C2xx C Source Debugger User's Guide*.

5) Start your application.

Note: Latency Associated With Hardware Breakpoints

Because of the BTT pipeline architecture and the inherent delays of the EMU0 line on the target device, a latency of approximately eight bus clock cycles is associated with a hardware breakpoint generated by the XDS522A. The number of clock cycles associated with the latency depends on the type of the event that signals a breakpoint.

17.2 Triggers

The trigger functionality of the XDS522A is useful for synchronizing oscilloscopes or other test equipment. You have two options for generating trigger pulses:

- You can generate a high-to-low trigger pulse on the BNC connector:
  - When Event A, B, C, or D occurs
  - When a counter reaches zero
  - When a sequence of events completes

  The XDS522A holds this pulse active for at least one bus cycle. The BNC connector, shown in Figure 17–1 (a), generates this pulse *only* if you mark one or more trigger intersections in the ACTION box of the BTT setup window.

- You can generate an individual trigger pulse (low-to-high) on a trigger-connector pin:
  - When Event A, B, C, or D occurs
  - When a counter reaches zero
  - When a sequence of events completes

  Each of these trigger outputs is assigned to a trigger-connector pin on the front of the XDS522A chassis. These pulses are *always* generated on the trigger-connector, regardless of what you mark in the ACTION box of the BTT setup window.

  Figure 17–1 (b) shows the location of the trigger connector on the front of the XDS522A chassis and the arrangement of the triggers.

Figure 17–1. XDS522A Chassis

(a) Rear view

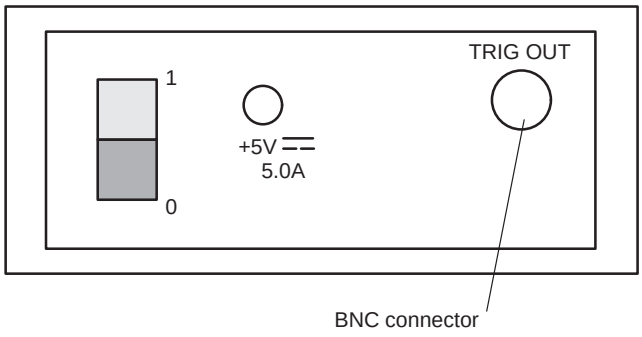
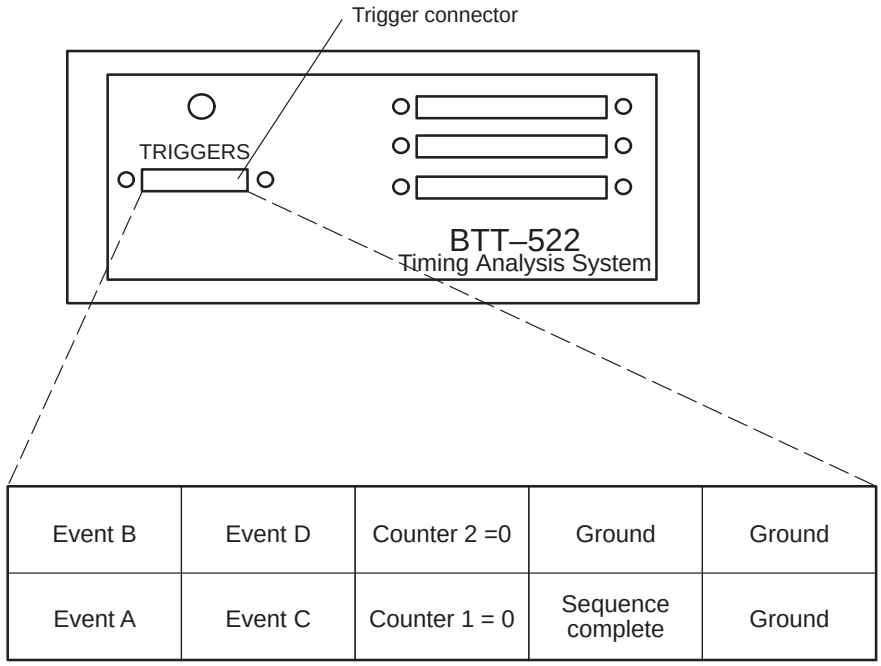


Figure 17–1. XDS522A Chassis (Continued)

(b) Front view



Event B	Event D	Counter 2 =0	Ground	Ground
Event A	Event C	Counter 1 = 0	Sequence complete	Ground

Note: Latency Associated With Triggers

For time-critical applications, keep in mind that the BTT pipeline architecture causes a latency of two bus clock cycles when the XDS522A generates a trigger pulse.

Generating a trigger pulse

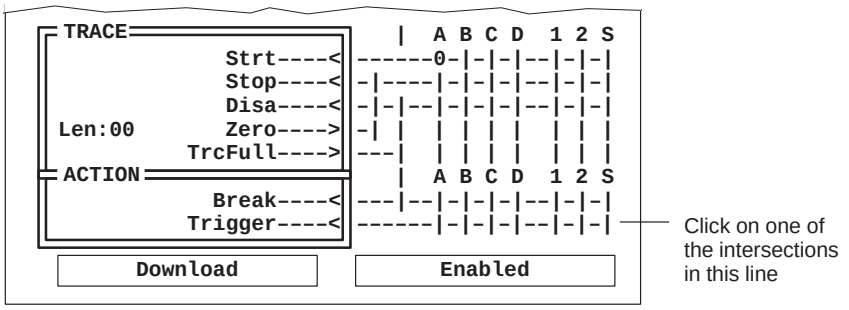
The ACTION box of the BTT setup window allows you to specify what the XDS522A should do on detection of a certain set of conditions. You use the ACTION box to direct the XDS522A to generate a trigger pulse on the BNC connector.

To generate a trigger pulse on the BNC connector, do the following:

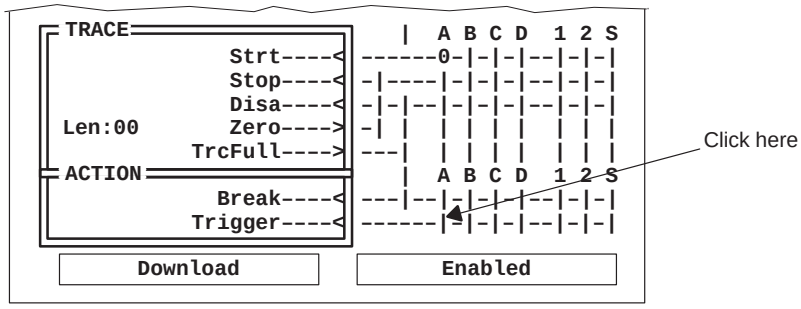
1) Define the event that you want to use to generate the trigger pulse. Your event can be:

   - ❑ Any of Events A through D
   - ❑ When a counter reaches 0
   - ❑ When the specified sequence completes

2) Set up the ACTION box to generate a trigger pulse by clicking on the intersection of *Trigger* and the condition that you want to monitor:



For example, to generate a trigger pulse when Event A occurs, click on the intersection of *Trigger* and Event A:



If you click on more than one intersection in the *Trigger* line, the conditions are logically ORed.

3) Download the XDS522A configuration and be sure that the XDS522A is enabled.

4) Start your application.

Part I
Overview

Part II
Tutorial

Part III
User Reference

Part IV
Appendixes

Part IV

Appendix A

XDS522A Emulation System Commands

This appendix describes commands that you can use as alternatives to selecting menu options and using dialog boxes to manage the XDS522A functionality. You can enter these commands from the BTT software's COMMAND window. You can also include these commands in a batch file that you execute with the TAKE command.

Topic **Page**

A.1 Defining Your Own Command Strings

The BTT software provides a shorthand method of entering often-used commands or command sequences. This processing is called *aliasing*. Aliasing enables you to define an alias name for the command(s) and then enter the alias name as if it were a BTT software command.

To do this, use the ALIAS command. The syntax for this command is:

alias [*alias name* [, "*command string*"]]

The primary purpose of the ALIAS command is to associate the *alias name* with the command that you've supplied as the *command string*. However, the ALIAS command is versatile and can be used in several ways:

- **Aliasing several commands.** The *command string* can contain more than one command—just separate the commands with semicolons. For example:

  `alias startdb, "trgsend restart;trgsend run"` ↵

  Now you could enter STARTDB instead of the TRGSEND commands listed within the quotation marks.

- **Supplying parameters to the command string.** The *command string* can define parameters that you'll supply later. To do this, use a percent sign and a number (%1) to represent the parameter that will be filled in later. The numbers should be consecutive (%1, %2, %3) unless you plan to reuse the same parameter value for multiple commands.

- **Listing all aliases.** To display a list of all the defined aliases, enter the ALIAS command with no parameters. The BTT software lists the aliases and their definitions in the COMMAND window.

- **Finding the definition of an alias.** If you know an alias name but are not sure of its current definition, enter the ALIAS command with just an alias name. The BTT software displays the definition in the COMMAND window.

- **Nesting alias definitions.** You can include a defined alias name in the *command string* of another alias definition. This is especially useful when the command string would be longer than the BTT software command line.

- **Redefining an alias.** To redefine an alias, reenter the ALIAS command with the same alias name and a new command string.

To delete a single alias, use the UNALIAS command:

unalias *alias name*

To delete *all* aliases, enter the UNALIAS command with an asterisk instead of an alias name:

`unalias *` ↵

Note that the * symbol *does not* work as a wildcard.

Notes: Considerations When Using Alias Definitions

1) Alias definitions are lost when you exit the BTT software. If you want to reuse aliases, define them in a batch file. You can execute that batch file with the TAKE command (for more information, see Section A.2)
2) Individual commands within a command string are limited to an expanded length of 132 characters. The expanded length of the command includes the length of any substituted parameter values.

The aliasing feature for the BTT software is the same as that for the debugger. For more information about aliasing, see the *TMS320C2xx C Source Debugger User's Guide*.

A.2 Creating a Batch File

When you invoke the BTT software, the software looks for an initialization batch file. That file contains special memory-mapping commands and other initialization commands that the BTT software needs to work properly.

You can create your own batch file for commands that you want to enter at one time. A batch file is useful for tasks such as defining aliases that you want to reuse, setting up your screen configuration, invoking the debugger, loading the symbol table, or any other task that you want to do each time you invoke the BTT software.

Example A–1 shows a sample batch file that you can create. You can create the batch file in any text editor (such as the Windows NotePad).

Example A–1. Sample Batch File for Use With the BTT Software

```
echo Starting debugger
trgexe emu2xxwm.exe sample –n CPU_TRG –t se_init.cmd –b –@

echo Loading screen configuration
sconfig myconfig.cfg

echo Defining aliases
alias startdebug, ”trgexe emu2xxwm.exe sample –n CPU_TRG –t se_init.cmd –b –@”
alias restdb, ”trgsend restart”
alias rundb, ”trgsend run”

echo Loading symbol table
trgload sample.out,n
```

Once you create a batch file, you can execute it by entering the following from the BTT software's COMMAND window:

take *batch filename*

The BTT software reads and executes the commands in the *batch filename*.

Echoing strings in a batch file

When executing a batch file, you can display a string to the COMMAND window by including the ECHO command in your batch file. The syntax for the command is:

echo *string*

This displays the *string* in the display area of the COMMAND window.

For example, you might want to document what is happening during the execution of a certain batch file. To do this, you could use a line such as the following in your batch file to indicate that you are invoking the debugger:

```
echo Invoking the debugger
```

(Notice that the string is not enclosed in quotes.)

When you execute the batch file, the following message appears:

```
.
.
Invoking the debugger
.
.
```

Any leading blanks in your string are removed when the ECHO command is executed.

Controlling command execution in a batch file

In batch files, you can control the flow of XDS522A commands. You can choose to execute XDS522A commands conditionally or set up a looping situation by using IF/ELSE/ENDIF or LOOP/ENDLOOP, respectively.

- To execute XDS522A commands conditionally in a batch file, use the IF/ELSE/ENDIF commands. The syntax is:

  if *Boolean expression*
  commands
  [**else**
  commands]
  endif

- To set up a looping situation to execute XDS522A commands in a batch file, use the LOOP/ENDLOOP commands. The syntax is:

  loop *expression*
  commands
  endloop

  These looping commands evaluate as follows:

  - If you use an *expression* that is not Boolean, the XDS522A evaluates the expression as a loop count.
  - If you use a Boolean *expression*, the XDS522A executes the command repeatedly as long as the expression is true.

The IF/ELSE/ENDIF and LOOP/ENDLOOP commands work with the following conditions:

- You can use conditional and looping commands only in a batch file.
- You must enter each XDS522A command on a separate line in the file.
- You can't nest conditional and looping commands within the same file.

A.3 Functional List of Commands

This section lists the tasks that you can perform from the COMMAND window along with the command and a page reference where you can find the syntax and a description. The commands are not case sensitive.

Function	Command	Page
Clear the BTT setup window	bclear	A-8
Conditionally execute XDS522A commands in a batch file (valid only in a batch file)	if/else/endif	A-15
Configure the filter for the TRACE window	cfgfilter	A-11
Configure the search in the TRACE window	cfgsearch	A-12
Configure the TRACE window	cfgtrace	A-13
Define a custom command string	alias	A-8
Delete an alias definition	unalias	A-22
Disable the XDS522A operation	bdisable	A-8
Display a string to the COMMAND window while executing a batch file (valid only in a batch file)	echo	A-14
Display the newest trace samples	peek	A-17
Download the configuration to the XDS522A	bdownload	A-9
Enable the XDS522A operation	benable	A-9
Execute a batch file	take	A-18
Exit the BTT software	quit	A-17
Filter the TRACE window	tracefilter	A-19
Force a reload of the firmware	load	A-15
Invoke the debugger from the BTT software	trgexe	A-20
Load a saved screen configuration	sconfig	A-17
Load a saved TRACE window configuration	tracecfgload	A-18
Load an XDS522A configuration from a file	bload	A-9
Load the target code symbol table	trgload	A-20
Loop XDS522A commands in a batch file (valid only in a batch file)	loop/endloop	A-16

Function	Command	Page
Make a window active	win	A-22
Move the active window	move	A-16
Record the information shown in the display area of the COMMAND window	dlog	A-14
Reload the firmware	bttload	A-9
Run the target/wait for the debug mode	trgrunwait	A-21
Save the contents of the TRACE window to a file	tracesave	A-19
Save the current screen configuration to a file	ssave	A-18
Save the current TRACE window configuration to a file	tracecfgsave	A-18
Save the current XDS522A configuration to a file	bsave	A-9
Search the TRACE window	tracesearch	A-20
Send a command to the debugger	trgsend	A-21
Set the test options	bttoptions	A-10
Set the tracing mode (flush or accumulate)	traceflush	A-19
Single-step through the target application and wait for the debug mode	trgstepwait	A-21
Size the active window	size	A-17
Stop executing the target application	trgstop	A-22
Update the contents of the TRACE window	traceupdate	A-20
Zoom the active window	zoom	A-22

A.4 Alphabetical List of Commands

alias — *Define Custom Command String*

Syntax **alias** [*alias name* [**, "***command string***"**]]

Menu selection none

Description Use the ALIAS command to define customized command strings. You can associate one or more BTT software commands with a single *alias name*. Include as many commands in the *command string* as you like, as long you separate them with semicolons and enclose the entire string of commands in quotation marks. You can also identify command parameters by a percent sign followed by a number (%1, %2, etc.). The total number of characters for an individual command (expanded to include parameter values) is limited to 132.

Previously defined alias names can be included as part of the definition for a new alias.

To find the current definition of an alias, enter the ALIAS command with the *alias name* only. To see a list of all defined aliases, enter the ALIAS command with no parameters.

bclear — *Clear BTT Setup Window*

Syntax **bclear**

Menu selection BTT→Clear

Description Use the BCLEAR command to clear the contents of the BTT setup window. This allows you to define a new configuration.

bdisable — *Disable XDS522A Operation*

Syntax **bdisable**

Menu selection BTT→Disable

Description Use the BDISABLE command to temporarily disable the operation of the XDS522A. You can also perform this task by clicking on the Enabled button at the bottom of the BTT setup window.

bdownload — *Download Configuration to XDS522A*

Syntax **bdownload**

Menu selection BTT→Download

Description Use the BDOWNLOAD command to download a configuration to the XDS522A. You can also perform this task by clicking on the Download button at the bottom of the BTT setup window.

benable — *Enable XDS522A Operation*

Syntax **benable**

Menu selection BTT→Enable

Description Use the BENABLE command to enable the operation of the XDS522A. You can also perform this task by clicking on the Disabled button at the bottom of the BTT setup window.

bload — *Load XDS522A Configuration*

Syntax **bload** *filename*

Menu selection BTT→Load

Description Use the BLOAD command to load an XDS522A configuration from a file. The configuration is one that you previously defined in the BTT setup window and saved with the BSAVE command.

bsave — *Save XDS522A Configuration*

Syntax **bsave** *filename*

Menu selection BTT→Save

Description Use the BSAVE command to save XDS522A the configuration in the BTT set-up window to a file.

bttload — *Reload Firmware*

Syntax **bttload** *filename*

Menu selection none

Description Use the BTTLOAD command to reload the firmware. When you enter BTTLOAD, the XDS522A reloads the firmware, after checking to see if the firmware is already loaded. Use this command after you have confirmed that the firmware is *not* operating correctly.

bttoptions — *Set Test Options*

Syntax **bttoptions** *value*

Menu selection none

Description Use the BTTOPTIONS command to set the test options. You supply a hexadecimal value that toggles the default setting of a particular test option. The test options and settings are listed here:

Test Option	Default Setting	Use this *value* to toggle the default setting
Represent inactive cycle values as '....' in the TRACE window	ON	0x1
Display error log on the screen	OFF	0x2
Display firmware interface messages	OFF	0x4
Allow debugging firmware code (halt interface)	OFF	0x8
Disable display of trace buffer address	ON	0x10

For example, if you wanted to display your error log on the screen, you would enter:

```
bttoptions 0x2 ↵
```

You can change the default value of more than one test option. To do so, enter the sum of the values for each option. For example, if you wanted to display firmware interface messages (0x0004) and turn off the display of inactive cycle values in the TRACE window (0x0001), you would enter the command with the value 0x0005:

```
bttoptions 0x5 ↵
```

To revert to the default settings for all test options, use 0 as the *value*:

```
bttoptions 0 ↵
```

cfgfilter — *Configure Filter for TRACE Window*

Syntax

cfgfilter *option, state, mask*

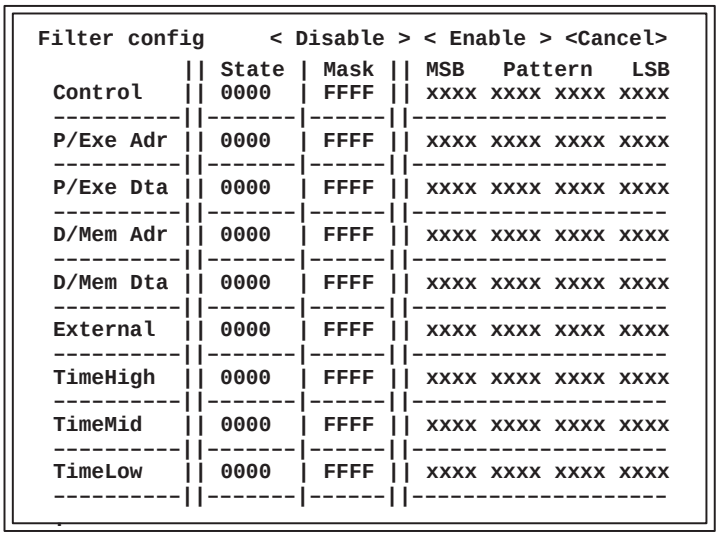
Menu selection

Trace→Filter

Description

Use the CFGFILTER command to define a filter for the data in the TRACE window. After you enter this command with the state and mask values for each option that you want to use in the filter (each specified with a hexadecimal value), you must enter the TRACEFILTER command to enable the filter that you have defined.

The following valid values for the *option* parameter are listed next to the corresponding field in the Filter config dialog box.



control
progaddr
progdata
dataaddr
datadata
external
timehigh
timemid
timelow

cfgsearch — *Configure Search in TRACE Window*

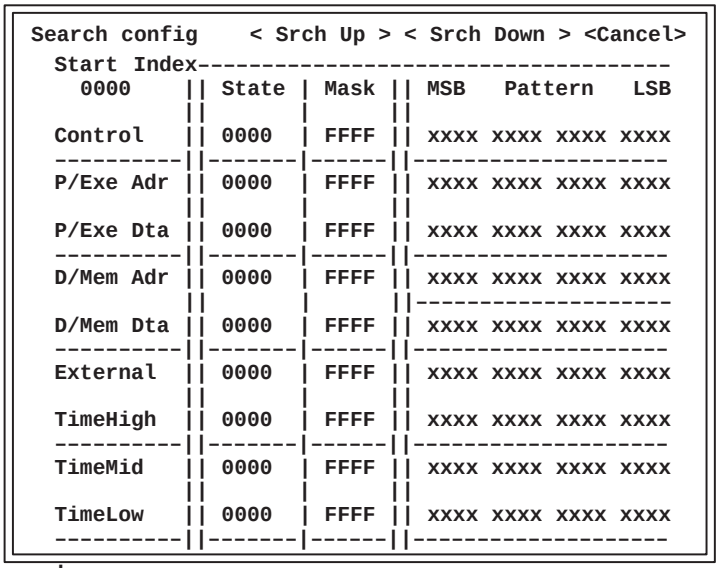
Syntax **cfgsearch** *option, state, mask*

Menu selection Trace→Search

Description Use the CFGSEARCH command to search the TRACE window for specific samples. After you enter the starting index and the state and mask for each option that you want to use in the search, you must enter the TRACESEARCH command.

The valid entries for the *option* parameter are listed below next to the corresponding field in the Search config dialog box.

startindex
control
progaddr
progdata
dataaddr
datadata
external
timehigh
timemid
timelow



cfgtrace — *Configure TRACE Window*

Syntax

cfgtrace *option, value*

Menu selection

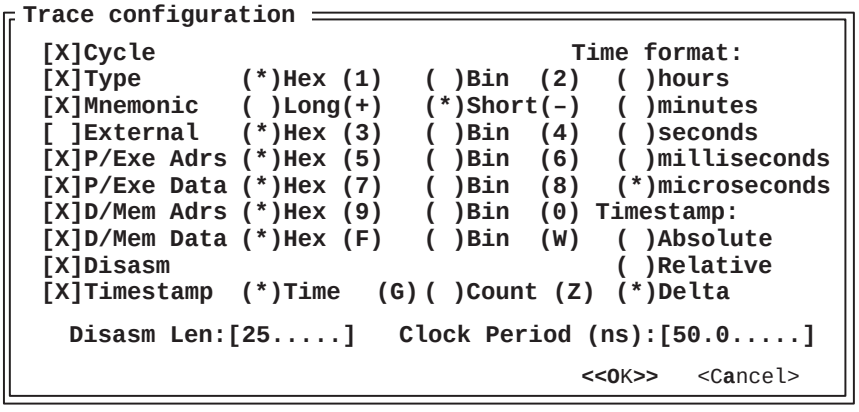
Trace→Config

Description

Use the CFGTRACE command to specify which columns of information you want to appear in the TRACE window.

The valid entries for the *option* and *value* parameters are listed below. These parameters correspond to fields in the Trace configuration dialog box.

Option	Values	Option	Value
cycle	,{ **off** | **on** }		
type	,{ **off** | **on** | **hex** | **bin** }	**hours**	**,on**
mnemonic	,{ **off** | **on** | **short** | **long** }	**minutes**	**,on**
external	,{ **off** | **on** | **hex** | **bin** }	**seconds**	**,on**
paddress	,{ **off** | **on** | **hex** | **bin** }	**milliseconds**	**,on**
pdata	,{ **off** | **on** | **hex** | **bin** }	**microseconds**	**,on**
daddress	,{ **off** | **on** | **hex** | **bin** }		
ddata	,{ **off** | **on** | **hex** | **bin** }	**absolute**	**,on**
disasm	,{ **off** | **on** | **hex** | **bin** }	**relative**	**,on**
timestamp	,{ **off** | **on** | **count** | **time** }	**delta**	**,on**
disasmlen	,*value*	**clockperiod**	,*float_value*



dlog — *Record Display Window*

Syntax

dlog *filename* [**,{a** | **w}**]
or
dlog close

Menu selection

none

Description

The DLOG command allows you to record the information displayed in the COMMAND window into a log file.

- To begin recording the information shown in the display area of the COMMAND window, use:

  dlog *filename*

  Log files can be executed with the TAKE command. When you use DLOG to record the information from the display area into a log file called *filename*, the BTT software automatically precedes all error or progress messages and command results with a semicolon to turn them into comments. This way, you can easily reexecute the commands in your log file by using the TAKE command.

- To end the recording session, enter:

  `dlog close` ↵

If necessary, you can write over existing log files or append additional information to existing files. The optional parameters of the DLOG command control how existing log files are used:

- **Appending to an existing file.** Use the **a** parameter to open an existing file to which to append the information in the display area.
- **Writing over an existing file.** Use the **w** parameter to open an existing file to write over the current contents of the file. Note that this is the default action if you specify an existing filename without using either the **a** or **w** options; you will lose the contents of an existing file if you don't use the append (a) option.

echo — *Echo String to Display Area*

Syntax

echo *string*

Menu selection

none

Description

The ECHO command displays *string* in the display area of the COMMAND window. Any leading blanks in your command string are removed when the ECHO command is executed. You can execute the ECHO command only in a batch file.

if/else/endif — *Conditionally Execute XDS522A Commands*

Syntax

if *expression*
commands
[**else**
commands]
endif

Menu selection

none

Description

Use the IF/ELSE/ENDIF commands to execute XDS522A commands conditionally in a batch file. If the *expression* is nonzero, the XDS522A executes the commands between the IF and the ELSE or ENDIF. Note that the ELSE portion of the command sequence is optional.

The conditional commands work with the following provisions:

- ❏ You can use conditional commands only in a batch file.
- ❏ You must enter each XDS522A command on a separate line in the file.
- ❏ You can't nest conditional commands within the same batch file.

load — *Reload Firmware*

Syntax

load *filename*

Menu selection

none

Description

Use the LOAD command to force a reload of the firmware. Use this command after you have confirmed that the firmware is *not* operating correctly.

loop/endloop — *Loop XDS522A Commands*

Syntax

loop *expression*
commands
endloop

Menu selection

none

Description

Use the LOOP/ENDLOOP commands set up a looping situation in a batch file. These looping commands evaluate as follows:

- If you use an *expression* that is not Boolean, the XDS522A evaluates the expression as a loop count.
- If you use a Boolean *expression*, the XDS522A executes the command repeatedly as long as the expression is true.

The LOOP/ENDLOOP commands work under the following conditions:

- You can use LOOP/ENDLOOP commands only in a batch file.
- You must enter each XDS522A command on a separate line in the file.
- You can't nest LOOP/ENDLOOP commands within the same file.

move — *Move Active Window*

Syntax

move [*X position*, *Y position* [, *width*, *length*]]

Menu selection

none

Description

Use the MOVE command to move the active window to the specified XY position. If you choose, you can resize the window while you move it (see the SIZE command on page A-17 for valid *width* and *length* values). You can use the MOVE command in one of two ways:

- By supplying a specific *X position* and *Y position*
- By omitting the *X position* and *Y position* parameters and using function keys to interactively move the window

You can move a window by defining a new XY position for the window's upper left corner. Valid X and Y positions depend on the screen size and the window size. X positions are valid if the X position plus the window width in characters is less than or equal to the screen width in characters. Y positions are valid if the Y position plus the window height is less than or equal to the screen height in lines.

peek — *Display Newest Samples*

Syntax **peek**

Menu selection Peek=F9

Description Use the PEEK command to take a snapshot of the newest trace samples and display them in the TRACE window. This command is valid only if tracing is on.

quit — *Exit BTT Software*

Syntax **quit**

Menu selection File→Exit

Description Use the QUIT command to exit the BTT software.

sconfig — *Load Screen Configuration*

Syntax **sconfig** *filename*

Menu selection none

Description Use the SCONFIG command to restore the display to a specified configuration. This restores the window positions and window sizes that were saved with the SSAVE command into *filename*.

size — *Size Active Window*

Syntax **size** [*width*, *length*]

Menu selection none

Description Use the SIZE command to change the size of the active window. You can use the SIZE command in one of two ways:

- By supplying a specific *width* and *length*
- By omitting the *width* and *length* parameters and using function keys to interactively resize the window

Valid values for the width and length depend on the screen size and the window position on the screen. If the window is in the upper left corner of the screen, the maximum size of the window is the same as the screen size minus one line. (The extra line is needed for the menu bar.) For example, if the screen size is 80 characters by 25 lines, the largest window size is 80 characters by 24 lines.

If a window is in the middle of the display, you can't size it to the maximum height and width—you can size it only to the right and bottom screen borders.

ssave — *Save Screen Configuration*

Syntax **ssave** [*filename*]

Menu selection none

Description Use the SSAVE command to save the current screen configuration to a file. This saves the window positions and window sizes. If you omit the *filename*, the BTT software names the file init.clr.

take — *Execute Batch File*

Syntax **take** *batch filename*

Menu selection none

Description Use the TAKE command to tell the BTT software to read and execute commands from a batch file. The *batch filename* parameter identifies the file that contains the commands. If you don't supply a pathname as part of the filename, the BTT software first looks in the current directory and then searches directories named with the D_DIR environment variable.

tracecfgload — *Load TRACE Window Configuration*

Syntax **tracecfgload** *filename*

Menu selection none

Description Use the TRACECFGLOAD command to load a TRACE window configuration that you saved with the TRACECFGSAVE command.

tracecfgsave — *Save TRACE Window Configuration*

Syntax **tracecfgsave** *filename*

Menu selection none

Description Use the TRACECFGSAVE command to save the current TRACE window configuration. You can use the Trace → Config menu option or or the CFGTRACE command to configure your TRACE window and use the TRACECFGSAVE command to save that configuration.

To load a saved configuration, use the TRACECFGLOAD command.

tracefilter — *Filter TRACE Window*

Syntax **tracefilter** { **on** | **off** }

Menu selection none

Description Use the TRACEFILTER command to limit the display in the TRACE window. Use this command after you have entered all of the filter parameters that you defined with the CFGFILTER command. This command corresponds to the Disable and Enable buttons in the Filter config dialog box.

traceflush — *Set Tracing Mode*

Syntax **traceflush** { **on** | **off** }

Menu selection ModeFlush

Description Use the TRACEFLUSH ON command to flush the trace buffer when tracing is on and you want to start collecting new samples (trace flush). Use the TRACEFLUSH OFF command to continue accumulating samples in the trace buffer (trace accumulate).

tracesave — *Trace Contents of TRACE Window*

Syntax **tracesave** *filename*, *start*, *end*, { **y** | **n** } , { **y** | **n** }

Menu selection Trace→Save

Description Use the TRACESAVE command to save the current trace samples to a file.

- ❏ *filename* names the file for saving the trace samples.
- ❏ *start* indicates the number of the first sample that you want to save.
- ❏ *end* indicates the number of the last sample that you want to save; this value must be smaller than the start value.
- ❏ The first set of { **y** | **n** } indicates how you want to save the file: **y** indicates text mode; **n** indicates binary mode.
- ❏ The second set of { **y** | **n** } indicates whether you want to append or overwrite an existing file; **y** indicates append/overwrite; **n** indicates overwrite.

tracesearch — *Search TRACE Window*

Syntax **tracesearch** { **up** | **down** }

Menu selection none

Description Use the TRACESEARCH command to search the TRACE window for samples that you defined with the CFGSEARCH command. This command corresponds to the Srch up and Srch down buttons in the Search config dialog box.

traceupdate — *Update Contents of TRACE Window*

Syntax **traceupdate**

Menu selection Trace→Update

Description Use the TRACEUPDATE command to constantly update the TRACE window with the newest samples. This command functions as a continuous PEEK command when tracing is on. To disable this trace-update function, press ESC .

trgexe — *Invoke Debugger*

Syntax **trgexe** *debugger executable* [*filename*] *options* **–n** *processor name* **–@**

Menu selection none

Description Use the TRGEXE command to invoke the C source debugger; specify all command-line options that you normally use to start the application.

trgload — *Load Target Code Symbol Table*

Syntax **trgload** *filename*, { **yes** | **no** }

Menu selection File→Target Code

Description To ensure the accuracy of the symbolic disassembly in the TRACE window, use the TRGLOAD command to load the target code after you reassemble and load the application source in the debugger. The **yes** or **no** parameter indicates whether or not you want to append the symbols to the current set.

If you want to clear the current symbols, enter **null** as the *filename*.

trgrunwait — *Run Target / Wait for Debug Mode*

Syntax **trgrunwait**

Menu selection none

Description Use the TRGRUNWAIT command to start the target application and wait until the processor finishes running due to a software breakpoint or an analysis break event. This command freezes the user interface. To cancel the wait mode, press ESC in either the BTT application window or the debugger application window.

trgsend — *Send Command to Debugger*

Syntax **trgsend** *debugger command*

Menu selection none

Description Use the TRGSEND command to send a debugger command to the debugger. In order for the debugger to recognize the TRGSEND command, you must invoke the debugger with the –@ command-line option.

The *debugger command* that you specify is sent directly to the debugger's command interpreter. Any progress or error messages generated by the debugger are not shown in the BTT software's COMMAND window.

When you use the TRGSEND command, control is immediately returned to the command line of the BTT software—the BTT software does not wait for the debugger to finish executing the command.

trgstepwait — *Single-Step Target Application / Wait for Debug Mode*

Syntax **trgstepwait**

Menu selection none

Description Use the TRGSTEPWAIT command to single-step through the target application and wait until the processor finishes stepping due to a software breakpoint or an analysis break event. This command freezes the user interface. To cancel the wait mode, press ESC in the BTT application window.

trgstop — *Stop Executing Target Application*

Syntax **trgstop**

Menu selection none

Description Use the TRGSTOP command to send an escape sequence to the processor to stop executing the target application.

unalias — *Delete Alias Definition*

Syntax **unalias** *alias name*
unalias *****

Menu selection none

Description Use the UNALIAS command to delete defined aliases.

- ❏ To delete a single alias, enter the UNALIAS command with an alias name.
- ❏ To delete all aliases, enter an asterisk instead of an alias name. Note that the * symbol *does not* work as a wildcard.

win — *Make a Window Active*

Syntax **win** *WINDOW NAME*

Menu selection none

Description Use the WIN command to make a window active. Note that the *WINDOW NAME* is in uppercase (matching the name exactly as displayed). You can spell out the entire window name, but you really need to specify only enough letters to identify the window.

zoom — *Zoom Active Window*

Syntax **zoom**

Menu selection none

Description Use the ZOOM command to make the active window as large as possible. To "unzoom" a window, enter the ZOOM command a second time; this returns the window to its prezoom size and position.

Appendix B

Troubleshooting

This appendix describes solutions to common problems that users experience with the XDS522A emulation system. This appendix also includes a listing of the progress and error messages that the BTT software might display in the COMMAND window.

Topic **Page**

B.1 Solutions to Common Problems

This section provides some troubleshooting tips that you can use when working with the XDS522A emulation system. Situations are described, followed by actions that you can try to work through the problem that you are experiencing.

XDS522A is not doing what you expect

Situation When you run the program, the XDS522A does not do what you expect. You can tell that the debugger is working because you are looking at the program in the DISASSEMBLY window. You can tell that the BTT software is up because all three of its windows are displayed.

Try This

- ❑ Check to see if the XDS522A is enabled by looking at the bottom right corner of the BTT setup window. If it displays the word *Disabled*, click on the word to enable the XDS522A. Then, download the XDS522A configuration again and rerun your program.
- ❑ If the XDS522A is already enabled, try downloading the XDS522A configuration again. Remember, if the Download button is yellow, the current configuration has not been downloaded.

Events are not being detected

Situation The XDS522A does not seem to detect the event that you defined.

Try This

- ❑ Check to see how you set up the events in the BTT setup window. Sometimes the problem is just one misplaced connection symbol, so check all of your connections.
- ❑ A common mistake is mixing up EXEC data and EXEC address in the EVENTS box or setting MEM (memory) information when you meant to set EXEC information (or vice versa). Check to see how you set up the events in the EVENTS box of the BTT setup window. Verify the beginning and ending address fields and the mask value, if appropriate.
- ❑ Download the XDS522A configuration again.

Samples are not being collected

Situation The XDS522A is not collecting trace samples when you expect it to do so.

Try This

- ❑ Be sure that you told the XDS522A to start writing to the trace buffer (by enabling *Strt* in the TRACE box). Then, download the XDS522A configuration again and rerun your program.
- ❑ You might have set up a trace start qualifier (TSQ) condition that is not being met. If you do *not* want to use a TSQ condition, make sure the entire TSQ row in the SEQUENCER box is selected (highlighted). If you want to use a TSQ condition, make sure your TSQ condition is valid.

  For more information about TSQ, see Section 16.5, *Starting Trace During a Specified Sequencer Level,* on page 16-9.

Samples are collected but not displayed

Situation Nothing displays in the TRACE window even though the trace status indicator (the number in the upper right corner of the BTT application window) has a number in it other than zero.

Try This

- ❑ Check to see if you set something up in the trace filter by selecting Filter from the Trace menu. You might have set up a filter in a previous session and forgot to disable it for this session. To disable the filtering feature, use the File→Filter command to bring up the Filter config dialog box. Then, click on Disable.
- ❑ Click on *TrcOn=F8* in the menu bar or press the F8 key. When you click on *TrcOn=F8* or press F8, the TRACE window updates with the contents of the trace buffer.

Breakpoints are not executing

Situation The XDS522A is not generating a hardware breakpoint when you expect it to do so.

Try This Be sure that you have set up the debugger's analysis module to recognize and generate hardware breakpoints. Review the steps on page 3-6 or see the *TMS320C2xx C Source Debugger User's Guide.* Also, be sure that the analysis module is enabled.

Counter will not stop

Situation The counter keeps reloading and does not count down to zero.

Try This Be sure that the last line of the COUNTER box is set to SnglON. This ensures that the counter goes to zero one time and does not restart. If it is set to SnglOFF, click on the word SnglOFF to turn on the single-shot counter feature. Then, download the XDS522A configuration again and rerun your program.

For more information about the SnglOFF/SnglON toggle, see Section 10.3, *Using a Single-Shot Counter*, on page 10-10.

Symbol name is not recognized

Situation The XDS522A does not recognize a symbol name that you enter in a dialog box.

Try This Be sure that the symbol-table portion of the object file is loaded into the XDS522A. Use the File→Target Code command or the TRGLOAD command.

For more information about loading the symbol-table portion of the object file into the XDS522A, see Section 3.2, *Invoking the BTT Software*, on page 3-3.

Filter feature is not working properly

Situation You are using the trace filter feature, but the XDS522A does not show you the samples that you expect. (For example, it shows no samples or one sample and you think it should be showing many samples.)

Try This

- ❑ In the Filter config dialog box, check the State column. Make sure you entered the value(s) that you want to use in the filter in the correct row(s). A common mistake is mixing up the P/Exe Adr row with the P/Exe Dta row or mixing up the D/Mem Adr row with the D/Mem Dta row.

  Also, check the Mask column. Make sure you are not masking a value that you want the system to look for:

  - With the exception of the row(s) that you want to use in the filter, all of the rows in the State column should be filled with zeros.
  - With the exception of the row(s) that you want to use in the filter, all of the rows in the Mask column should be filled with the letter F.
  - With the exception of the row(s) that you want to use in the filter, all of the rows in the Pattern column should be filled with the letter x.

- ❑ Disable the filter. With the TRACE window active, press HOME so that the first sample that was collected is displayed. Then, reenable the filter.

For more information about filtering, see 8.2, *Filtering Samples Out of the TRACE Window*, on page 8-6.

Search feature is not working properly

Situation You are using the search feature, but the XDS522A cannot seem to find the sample that you are looking for.

Try This In the Search config dialog box, check the State column. Make sure you entered the value(s) that you want to use in the search in the correct row(s). A common mistake is mixing up the P/Exe Adr row with the P/Exe Dta row or mixing up the D/Mem Adr row with the D/Mem Dta row.

Also, check the Mask column. Make sure you are not masking a value that you want the system to look for:

- ❏ With the exception of the row(s) that you want to use in the search, all of the rows in the State column should be filled with zeros.
- ❏ With the exception of the row(s) that you want to use in the search, all of the rows in the Mask column should be filled with the letter F.
- ❏ With the exception of the row(s) that you want to use in the search, all of the rows in the Pattern column should be filled with the letter x.

For more information about searching, see 8.1, *Searching for a Sample in the TRACE Window*, on page 8-2.

BTT software does not recognize commands

Situation You are trying to enter an XDS522A command, but the BTT software does not recognize your keyboard inputs.

Try This Check to see if the BTT setup window is active (it is active when it has the double-lined borders). When the BTT setup window is active, commands are not recognized in the COMMAND window (unlike the functionality of the C source debugger). Use F6 to cycle through the windows and make the COMMAND window active.

B.2 Summary of BTT Software Messages

This section contains an alphabetical listing of the progress and error messages you might see in the COMMAND window. Each message contains both a description of the situation that causes the message and an action to take if the message indicates a problem or error.

BTT configuration downloaded

Description The XDS522A finished downloading the latest configuration in the BTT setup window.

Action None required; this is normal XDS522A behavior.

BTT configuration loaded from *filename*

Description The XDS522A loaded a configuration that you specified with the BLOAD or BTT→Load command.

Action Download the configuration by clicking on the word Download in the BTT setup window, by selecting Download from the BTT menu, or by entering BDOWNLOAD from the COMMAND window.

BTT configuration saved: *filename*

Description You used the BSAVE or BTT→Save command to save the XDS522A configuration to a file.

Action None required; this is normal XDS522A behavior.

BTT configuration saved for flattened mode

Description You're using the XDS522A in unflattened mode, but the XDS522A configuration that you tried to load with the BLOAD or BTT→Load command was saved in a flattened mode.

Action Reenter the command using a configuration file saved in unflattened mode.

BTT configuration saved for non-flattened mode

Description You're using the XDS522A in a flattened mode, but the XDS522A configuration that you tried to load with the BLOAD or BTT→Load command was saved in unflattened mode.

Action Reenter the command using a configuration file saved in a flattened mode (flat first word or flat multi word).

BTT disabled

Description You disabled the XDS522A by entering the BDISABLE or BTT→Disable command.

Action None required; this is normal system behavior.

BTT enabled

Description You enabled the XDS522A by entering the BENABLE or BTT→Enable command.

Action None required; this is normal system behavior.

C

Cannot append in binary mode

Description You tried to save a file as binary using the TRACESAVE or Trace→Save command, and you selected the append saving option.

Action When saving the trace buffer in binary, use the abort/cancel or overwrite saving option.

Can not create file *filename*

Description There is an invalid character in the filename that you entered with the BSAVE, BTT→Save, TRACESAVE, Trace→Save, or TRACECFGSAVE command.

Action Enter a filename that contains only alphanumeric characters.

Can not detect clock period

Description There is a problem with your firmware, or the XDS522A cannot detect the clock in your target system.

Action

- ❑ Modify the btt522a.cmd initialization file by replacing the BTTLOAD command with LOAD. This forces a reload of the firmware.
- ❑ Check the JP5 jumper settings on the XDS511 emulator:
  - ■ To use the XDS511 clock, ensure pins 7 and 8 are connected and pins 5 and 6 are not connected.
  - ■ To use the target clock, remove the jumper from pins 7 and 8 and connect pins 5 and 6.

  Never connect both sets of pins at the same time. Doing so results in a power-ground short, which typically blows a fuse.
- ❑ Ensure that the oscillator on the XDS511 emulator is seated correctly.

Can not load file *filename*

Description The XDS522A could not find the *filename* that you specified with the TRGLOAD command.

Action

- ❑ Be sure that the file exists as named.
- ❑ Be sure that the file resides in the current directory or in one of the directories specified by the D_DIR environment variable.

Can not locate symbol

Description The XDS522A could not find the symbol name that you entered.

Action

- ❑ Be sure that the symbol-table portion of the object file that you are using is loaded into the XDS522A. Use the File→TargetCode command or the TRGLOAD command.
- ❑ Be sure that the symbol name that you specified is listed in the symbol table.

Cannot open config file

Description The XDS522A could not find the file that you specified with the SCONFIG command.

Action

- ❑ Be sure that the file exists as named.
- ❑ Be sure that the file resides in the current directory or in one of the directories specified by the D_DIR environment variable.

Can not open file: *filename*

Description The XDS522A could not find the *filename* that you specified with the TRACECFGLOAD, File→TargetCode, Trace→Load, or BTT→Load command.

Action

- ❑ Be sure that the file exists as named.
- ❑ Be sure that the file resides in the current directory or in one of the directories specified by the D_DIR environment variable.

Cannot open save file

Description There is an invalid character in the filename that you entered with the SSAVE command.

Action Enter a filename that contains only alphanumeric characters.

Can not read from BTT memory

Description The XDS522A lost power during your testing/debugging session.

Action

- ❑ Close the BTT software and turn off power to the XDS522A chassis.
- ❑ Check all of your hardware connections and cabling.
- ❑ Turn on the power to the XDS522A chassis.
- ❑ Execute EMURST by double-clicking on the Emurst icon.

Can not write to BTT memory

Description The XDS522A lost power during your testing/debugging session.

Action

- ❑ Close the BTT software and turn off power to the XDS522A chassis.
- ❑ Check all of your hardware connections and cabling.
- ❑ Turn on the power to the XDS522A chassis.
- ❑ Execute EMURST by double-clicking on the Emurst icon.

Clock period value error

Description The clock period that you entered in the Trace Config dialog box was invalid.

Action Reenter the clock period using a value from 0.1 ns to 100 ms.

Command "*cmd*" not found

Description The BTT software didn't recognize the command that you typed.

Action Reenter the correct command. See Appendix A, *XDS522A Emulation System Commands*, for a list of valid commands.

D

Disassembly Length Error

Description The number that you entered in the *Disasm Len* field of the Trace Config dialog box was invalid.

Action Enter a whole number that is less than or equal to 80.

F

File already exists

Description This message displays in the following situations:

- ❑ The file that you specified with the Trace→Save command already exists, and you selected *Abort* in the Trace save dialog box.
- ❑ The filename that you specified with the Trace→Save command contains an invalid character.

Action

- ❑ Use a unique filename or rename the existing file.
- ❑ Select *Append* or *Overwrite* in the Trace save dialog box.
- ❑ Enter a filename that contains only alphanumeric characters.

File not found

Description The XDS522A could not find the file that you specified with the BTTLOAD or LOAD command.

Action

- ❑ Be sure that the file exists as named.
- ❑ Be sure that the file resides in the current directory or in one of the directories specified by the D_DIR environment variable.

I

I/F comm error. Error UNKNOWN

Description The XDS522A is having trouble communicating with the ’C2xx processor.

Action

- ❑ Close the BTT software and turn off power to the XDS522A chassis.
- ❑ Check all of your hardware connections and cabling.
- ❑ Turn on the power to the XDS522A chassis.
- ❑ Execute EMURST by double-clicking on the Emurst icon.

Invalid config option *option*

Description The option that you named with the CFGTRACE command was correct, but the value for that option was incorrect.

Action Reenter the command with a valid value. For a list of valid options and values, see page A-13.

Invalid filter option: *option*

Description The option that you entered with the TRACEFILTER command was incorrect.

Action Use one of these commands: **tracefilter on** or **tracefilter off**.

Invalid option: *option*

Description The option that you named with the CFGFILTER, CFGSEARCH, or CFGTRACE command was invalid.

Action Reenter the command with a valid option name. For a list of valid options and values, see page A-11 for the CFGFILTER command, page A-12 for the CFGSEARCH command, or page A-13 for the CFGTRACE command.

Invalid options pattern

Description The option value that you entered with the BTTOPTIONS command was invalid.

Action Reenter the command with a value from 0x0 to 0x10.

Invalid search mask *value*

Description The mask value that you used with the CFGFILTER or CFGSEARCH command was invalid.

Action Reenter the command using a 16-bit hexadecimal mask value.

Invalid search option: *option*

Description The option that you entered with the TRACESEARCH command was incorrect.

Action Use one of these commands: **tracesearch up** or **tracesearch down**.

Invalid search value *value*

Description The option that you named with the CFGFILTER or CFGSEARCH command was correct, but the value for that option was incorrect.

Action Reenter the command with a valid value. For a list of valid options and values, see page A-11 for the CFGFILTER command or page A-12 for the CFGSEARCH command.

Invalid structure size in *filename*

Description The filename that you entered with the TRACECFGLOAD command was not a trace configuration file.

Action Reenter the command with a trace configuration file that you saved with the TRACECFGSAVE command.

Invalid Trace Mode option: *option*; TRACEFLUSH ON|OFF

Description The option that you entered with the TRACEFILTER command was incorrect.

Action Use one of these commands: **tracefilter on** or **tracefilter off**.

N

Name "*name*" not found

Description The start or end value that you entered with the TRACESAVE command was invalid.

Action Use a hexadecimal number for the start and end value, and be sure to prefix the number with **0x**.

P

Press 'Esc' to abort

Description The XDS522A is saving the contents of the trace buffer.

Action None required; if you want to interrupt the save, press ESC.

Press Esc to end WAIT

Description You entered the TRGRUNWAIT or TRGSTEPWAIT command, and the XDS522A is waiting for the processor to stop running.

Action None required; however, you can cancel the command and return control to the BTT software by pressing ESC.

S

Save to *filename* aborted

Description The XDS522A was saving the contents of the trace buffer, and you pressed ESC.

Action None required. If you want to save the trace buffer, reenter the TRACESAVE or Trace→Save command.

Saving *number* samples to file: *filename*

Description You used the TRACESAVE command to save *number* of trace samples to file *filename*.

Action None required; this is normal XDS522A behavior.

Syntax error

Description The start or end value that you entered with the TRACESAVE command was invalid.

Action Use a hexadecimal number for the start and end value and be sure to prefix the number with **0x**.

T

Target code loaded from *filename*

Description The XDS522A loaded the symbol table associated with *filename.*

Action None required; this is normal system behavior.

Target in debug mode

Description You attempted to update the contents of the TRACE window while the processor was stopped. When the processor stops running, the XDS522A updates the TRACE window automatically. Any further attempt to update the TRACE window while the processor is stopped results in this error message.

Action Update the contents of the TRACE window only when the processor is running. The program status indicator in the upper right corner of the BTT application window shows the status of the processor (Running or Stopped).

TO' on Msg Rcv – *value*, Seq *n*

Description The XDS522A is having trouble communicating with the target system. This message might display several times. While the message is displaying, your system will be locked up.

Action

- ❑ Close the BTT software and turn off power to the XDS522A chassis.
- ❑ Check all of your hardware connections and cabling.
- ❑ Turn on the power to the XDS522A chassis.
- ❑ Execute EMURST by double-clicking on the Emurst icon.

Trace buffer flush – DISABLED

Description You entered the TRACEFLUSH OFF command to turn on trace accumulate mode (and disable trace flush mode).

Action None required; this is normal XDS522A behavior.

Trace buffer flush – ENABLED

Description You entered the TRACEFLUSH ON command to turn on trace flush mode.

Action None required; this is normal XDS522A behavior.

Trace buffer saved: *filename*

Description You used the Trace→Save command to save trace samples to a file.

Action None required; this is normal XDS522A behavior.

Trace entry found –– *sample number*

Description The search event that you defined was found, and the XDS522A displayed the sample number associated with that event.

Action None required; this is normal XDS522A behavior.

Trace entry not found

Description This message displays when you are filtering or searching:

- ❑ Filtering: The filter condition that you defined is not in the trace buffer. This message might display if you didn't start the filter from the top of the TRACE window.
- ❑ Searching: The search condition that you defined is either not in the trace buffer or not in the direction in which you are searching.

Action

- ❑ Filtering:
  - ■ Check your filter condition.
  - ■ Disable the filter. With the TRACE window active, press HOME so that the first sample that was collected is displayed. Then, reenable the filter.

  For more information about filtering, see Section 14.1, *Filtering*, on page 14-2.
- ❑ Searching:
  - ■ Check your search condition.
  - ■ If you are searching up or down (using *Srch Up* or *Srch Down*), check your search index.

  For more information about searching, see Section 14.2, *Searching for a Specific Trace Sample*, on page 14-11.

Trace number error

Description The value that you entered in the *Sample Number* field of the Goto trace sample dialog box is invalid. You also see this message when you try to go to a specific trace sample while the XDS522A is collecting trace samples.

Action

- ❑ Enter a hexadecimal value from 0x0 to 0x7FFF.
- ❑ Be sure that tracing is disabled when you use the Trace→Goto command to go to specific trace sample.

Trace PgUp/PgDn offset value error

Description The value that you specified in the *Value(hex)* field of the Goto trace sample dialog box is invalid.

Action Enter a hexadecimal value that is less than or equal to 0x10000.

W

Waiting for target RUN–HALT

Description You entered the TRGRUNWAIT command, and the XDS522A is waiting for the processor to stop running.

Action None required; however, you can cancel the command and return control to the BTT software by pressing ESC.

Waiting for target to STEP

Description You entered the TRGSTEPWAIT command, and the XDS522A is waiting for the processor to stop running.

Action None required; however, you can cancel the command and return control to the BTT software by pressing ESC.

Appendix C

Glossary

A

Absolute mode: A timestamp display option that shows the time relative to the first sample that was collected.

B

breakpoint: A debugging mechanism that allows you to predefine an explicit stopping point in your program that causes the processor to stop executing. There are two types of breakpoints: a *software breakpoint* is set with the C source debugger; this breakpoint is not set in ROM and is controlled by the debugger only. A *hardware breakpoint* is set in ROM (using a combination of the BTT software and the C source debugger); this breakpoint allows you to stop the processor on certain program accesses or on low levels on the EMU0 pin.

BTT: *Breakpoint, tracing, and timing*. A software tool that adds breakpoint, tracing, and timing functionality to the C source debugger.

C

clock: A device that generates periodic signals used for synchronization.

counter: A feature of the XDS522A emulation system that counts the number of occurrences of an event or sequence of events or clock cycles. There are two counters in the XDS522A.

D

Delta mode: A timestamp display option that shows the time relative to the previous sample in the TRACE window.

E

event: A predefined condition that is used as an input to the functional control blocks of the XDS522A (such as the sequencer, counters, and trace control). All functional control blocks respond to events that you define.

F

fetch: Locating and loading instructions or data from storage.

flat first word: A flattened mode that shows only the address of the first word of a two-word instruction.

flat multi word: A flattened mode that shows the addresses of the first and second words in a two-word instruction.

flattened mode: An XDS522A mode that enables the flattener features.

flattener: A tool that filters out unexecuted instructions so that only executed cycles are collected in the trace buffer and aligns the data with the corresponding execution cycle.

I

IAQ: *Instruction acquisition*

M

memory map: A description of memory space partitioned into functional blocks.

mask: To ignore specific bits within an address, data, or external-signal value.

P

pipeline: A method of executing instructions in an assembly-line fashion rather than one at a time.

pipeline flattener: See *flattener*

program status indicator: A field in the upper right corner of the BTT application window that indicates the status of the processor (running or stopped).

Q

qualifier: A condition that modifies or limits the definition of an event.

R

real-time mode: A debugging mode that allows you to debug your code *without* stopping the processor; this allows time-critical interrupts to be serviced when the trap occurs.

Relative mode: A timestamp display option that shows the time relative to a sample that you select in the TRACE window.

S

sample: Program-execution information recorded at a discrete interval of time.

sequencer: A feature of the XDS522A emulation system that allows you to search for events that occur in the order you specify.

single-off mode: The XDS522A counter mode that allows you to configure the counter to decrement multiple times.

single-on mode: The XDS522A counter mode that prevents a counter from being restarted.

single-shot counter: A special feature of the XDS522A counters that prevents a counter from being restarted.

stop mode: A debugging mode that causes the processor to halt at a software breakpoint so that you can inspect the register contents at that particular point in the code.

T

timestamp: A 48-bit time display for each sample. The timestamp displays in the TRACE window. It can be displayed in absolute, relative, or delta time.

trace: Record the execution of a program, showing the sequence of instructions executed.

trace accumulate mode: A tracing mode in which previous samples in the trace buffer are retained when a new trace occurs.

trace buffer: A listing of samples collected by the XDS522A. The trace buffer contents are shown in the TRACE window.

trace flush mode: A tracing mode in which previous samples in the trace buffer are discarded (overwritten) when a new trace occurs.

trace immediate: A condition in which the XDS522A starts collecting trace samples as soon as an application begins running.

trace sample: See *sample*

trace start qualifier (TSQ): An XDS522A feature that allows you to set up a trace condition in which trace can start during a specified sequencer state only.

trace status indicator: A number field in the upper right corner of the BTT application window that tells you whether or not the XDS522A is collecting trace samples.

trigger: A pulse that is used to start an action in another circuit.

U

unflattened mode: An XDS522A mode that disables the flattener features. See also *flattener*

W

watchdog timer: A special feature of the XDS522A counters. A watchdog timer ensures that a time-critical deadline is always met and generates a signal if the processor is not executing the main loop correctly.

X

XDS522A emulation system: An analysis and debugging tool that adds breakpoint, tracing, and timing functionality to traditional debugging tools, allowing you to monitor and record bus- and cycle-type activity, gather precise instruction-timing information, and generate hardware breakpoints.

Index

C

D

E

F

G

H

I

J

K

L

M

N

P

Q

R

S

T

U

V

W

X

Z