ARTISAN®
TECHNOLOGY GROUP

Your **definitive** source for quality pre-owned equipment.

Artisan Technology Group
(217) 352-9330 | sales@artisantg.com | artisantg.com

Full-service, independent repair center
with experienced engineers and technicians on staff.

We buy your excess, underutilized, and idle equipment
along with credit for buybacks and trade-ins.

Custom engineering
so your equipment works exactly as you specify.

- Critical and expedited services
- In stock / Ready-to-ship
- Leasing / Rentals / Demos
- ITAR-certified secure asset solutions

Expert team | Trust guarantee | 100% satisfaction

All trademarks, brand names, and brands appearing herein are the property of their respective owners.

Find the *Astronics / EADS / Talon Instruments SR101* at our website: *Click HERE*

PROGRAMMING MANUAL

Talon Instruments™

SR192

SCPI Command Language

Publication Date: 04/24/06
Publication Number: SRPM120 Rev. A
Instrument Part Number: SR192

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

THANK YOU FOR PURCHASING THIS EADS NORTH AMERICA TEST AND SERVICES PRODUCT

For this product, or any other EADS North America Test and Services a division of EADS North America, Inc. ("EADS North America Test and Services") product that incorporates software drivers, you may access our web site to verify and/or download the latest driver versions. The web address for driver downloads is:

http://www.eads-nadefense.com/downloads

If you have any questions about software driver downloads or our privacy policy, please contact us at:

info@eads-nadefense.com

WARRANTY STATEMENT

All EADS North America Test and Services products are designed to exacting standards and manufactured in full compliance to EADS AS9100B Quality Management System processes.

This warranty does not apply to defects resulting from any modification(s) of any product or part without EADS North America Test and Services express written consent, or misuse of any product or part. The warranty also does not apply to fuses, software, non-rechargeable batteries, damage from battery leakage, or problems arising from normal wear, such as mechanical relay life, or failure to follow instructions.

This warranty is in lieu of all other warranties, expressed or implied, including any implied warranty of merchantability or fitness for a particular use. The remedies provided herein are buyer's sole and exclusive remedies.

For the specific terms of your standard warranty, contact EADS North America Test and Services Customer Support. Please have the following information available to facilitate service.

1. Product serial number
2. Product model number
3. Your company and contact information

You may contact Customer Support by:

E-Mail:	Helpdesk@eads-nadefense.com	
Telephone:	+1 800 722 3262	(USA)
Fax:	+1 949 859 7139	(USA)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

RETURN OF PRODUCT

Authorization is required from EADS North America Test and Services before you send us your product or sub-assembly for service or calibration. Call or contact Customer Support at 1-800-722-3262 or 1-949-859-8999 or via fax at 1-949-859-7139. We can also be reached at: helpdesk@eads-nadefense.com.

If the original packing material is unavailable, ship the product or sub-assembly in an ESD shielding bag and use appropriate packing materials to surround and protect the product.

PROPRIETARY NOTICE

This manual is copyright by EADS North America Test and Services, a division of EADS North America, Inc. Printed in the United States of America. All rights reserved. This book or parts thereof may not be reproduced in any form without written permission of the publisher

This document and the technical data herein disclosed, are proprietary to EADS North America Test and Services, and shall not, without express written permission of EADS North America Test and Services, be used in whole or in part to solicit quotations from a competitive source or used for manufacture by anyone other than EADS North America Test and Services. The information herein has been developed at private expense, and may only be used for operation and maintenance reference purposes or for purposes of engineering evaluation and incorporation into technical specifications and other documents which specify procurement of products from EADS North America Test and Services.

DISCLAIMER

Buyer acknowledges and agrees that it is responsible for the operation of the goods purchased and should ensure that they are used properly and in accordance with this document and any other instructions provided by Seller. EADS North America Test and Services products are not specifically designed, manufactured or intended to be used as parts, assemblies or components in planning, construction, maintenance or operation of a nuclear facility, or in life support or safety critical applications in which the failure of the EADS North America Test and Services product could create a situation where personal injury or death could occur. Should Buyer purchase EADS North America Test and Services product for such unintended application, Buyer shall indemnify and hold EADS North America Test and Services, its officers, employees, subsidiaries, affiliates and distributors harmless against all claims arising out of a claim for personal injury or death associated with such unintended use.

FOR YOUR SAFETY

Before undertaking any troubleshooting, maintenance or exploratory procedure, read carefully the **WARNINGS** and **CAUTION** notices.



This equipment contains voltage hazardous to human life and safety, and is capable of inflicting personal injury.



If this instrument is to be powered from the AC line (mains) through an autotransformer, ensure the common connector is connected to the neutral (earth pole) of the power supply.



Before operating the unit, ensure the conductor (green wire) is connected to the ground (earth) conductor of the power outlet. Do not use a two-conductor extension cord or a three-prong/two-prong adapter. This will defeat the protective feature of the third conductor in the power cord.



Maintenance and calibration procedures sometimes call for operation of the unit with power applied and protective covers removed. Read the procedures and heed warnings to avoid “live” circuit points.

Before operating this instrument:

1. Ensure the proper fuse is in place for the power source to operate.

2. Ensure all other devices connected to or in proximity to this instrument are properly grounded or connected to the protective third-wire earth ground.

If the instrument:

- fails to operate satisfactorily
- shows visible damage
- has been stored under unfavorable conditions
- has sustained stress

Do not operate until performance is checked by qualified personnel.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

DOCUMENT CHANGE HISTORY

Revision	Date	Description of Change
A	06/12/2009	Document Control release

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

This page was intentionally left blank.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table of Contents

Introduction

1.1 Manual Layout 1-1

SCPI Programming

2.1 Overview 2-1
2.1.1 SR192 SCPI Programming Steps 2-1
2.1.2 SR192 Plug&Play/VISA 2-1
2.1.3 Programming Language 2-2
2.2 Programming Examples 2-2
2.2.1 Initialize the VXI Session 2-3
2.2.2 Configuring SR192 settings 2-4
2.2.3 Route the I/O Channels 2-5
2.2.4 Program the timing module 2-6
2.2.5 Program the Pattern Memory 2-8
2.2.6 Define and Program the Execution Sequence 2-10
2.2.7 Enable Output Drivers 2-11
2.2.8 Run (Execute) the Defined Sequences 2-11
2.2.9 Analyze the Execution Results 2-12
2.2.10 Close the VXI Session 2-13

Command Reference

3.1 SCPI Fundamentals 3-1
3.1.1 SCPI Command Structure 3-1
3.1.1.1 Command Header Definition 3-1
3.1.1.1.1 IEEE 488.2 Common Command Header 3-1
3.1.1.1.2 Talon SCPI Instrument Command Header 3-1
3.1.1.2 Parameter List Definition 3-2
3.1.1.2.1 Parameter List Symbols 3-2
3.1.1.2.2 Parameter List Types 3-2
3.1.2 Events and Queries 3-3
3.2 Calculate Subsystem 3-4
3.2.1 CALCulate:CRC? 3-4
3.2.2 CALCulate:EMEMory 3-4
3.2.2.1 CALCulate:EMEMory:ADDRess? 3-5
3.2.2.2 CALCulate:EMEMory:COUNt? 3-5
3.2.3 CALCulate:PCRC? 3-6
3.3 Channel Subsystem 3-7
3.3.1 CHANnel:MODE 3-7
3.3.2 CHANnel:REFerence 3-7
3.4 Execute Subsystem 3-9
3.4.1 EXECute:FIELd 3-9
3.4.2 EXECute:MODE 3-9
3.4.3 EXECute:SEQuence 3-10
3.4.4 EXECute[:TIMing] 3-10
3.5 Input Subsystem 3-11
3.5.1 INPut:ADELay 3-11
3.5.2 INPut:MSTRobe 3-11
3.5.3 INPut:PROBe 3-12
3.5.3.1 INPut:PROBe:MODE 3-12
3.5.3.2 INPut:PROBe:ORDelay 3-13
3.5.3.3 INPut:PROBe:REFerence 3-13

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.5.3.3.1 INPut:PROBe:REFerence:HIGH[:DATA] . . . 3-13
3.5.3.3.2 INPut:PROBe:REFerence[:LOGic] . . . 3-13
3.5.3.3.3 INPut:PROBe:REFerence:LOW[:DATA] . . . 3-14
3.5.3.4 INPut:PROBe:STATus? . . . 3-14
3.5.3.5 INPut:PROBe:STRobe . . . 3-15
3.5.3.5.1 INPut:PROBe:STRobe:DELay . . . 3-15
3.5.3.5.2 INPut:PROBe:STRobe[:SOURce] . . . 3-15
3.5.3.6 INPut:PROBe:SWITch . . . 3-16
3.5.3.7 INPut:PROBe:VERsion? . . . 3-16
3.5.4 INPut:REFerence . . . 3-17
3.5.4.1 INPut:REFerence:AUTO . . . 3-17
3.5.4.2 INPut:REFerence:HIGH . . . 3-17
3.5.4.2.1 INPut:REFerence:HIGH[:DATA] . . . 3-17
3.5.4.2.2 INPut:REFerence:HIGH:INCRement . . . 3-18
3.5.4.3 INPut:REFerence:LOW . . . 3-18
3.5.4.3.1 INPut:REFerence:LOW[:DATA] . . . 3-18
3.5.4.3.2 INPut:REFerence:LOW:INCRement . . . 3-18
3.5.4.4 INPut:REFerence:SELect . . . 3-19
3.5.4.5 INPut:REFerence:UPDate . . . 3-19
3.5.5 INPut:STRobe . . . 3-19
3.5.5.1 INPut:STRobe:DELay . . . 3-19
3.5.5.2 INPut:STRobe[:SOURce] . . . 3-20
3.6 Measure Subsystem . . . 3-21
3.6.1 MEASure:VOLTage? . . . 3-21
3.7 Module Subsystem . . . 3-22
3.7.1 MODule:LINK . . . 3-22
3.7.2 MODule:SELect . . . 3-22
3.7.3 MODule:STATus? . . . 3-23
3.8 Output Subsystem . . . 3-25
3.8.1 OUTPut:CHANnel . . . 3-25
3.8.1.1 OUTPut:CHANnel:AUTO . . . 3-25
3.8.1.2 OUTPut:CHANnel[:STATe] . . . 3-26
3.8.2 OUTPut:CLK<a> . . . 3-26
3.8.2.1 OUTPut:CLK<a>:DELay . . . 3-27
3.8.2.2 OUTPut:CLK<a>:REFerence:HIGH[:DATA] . . . 3-27
3.8.2.3 OUTPut:CLK<a>:REFerence:LOW[:DATA] . . . 3-27
3.8.2.4 OUTPut:CLK<a>[:STATe] . . . 3-28
3.8.3 OUTPut:ENABle . . . 3-28
3.8.3.1 OUTPut:ENABle:DELay . . . 3-28
3.8.3.2 OUTPut:ENABle[:SOURce] . . . 3-29
3.8.4 OUTPut:MASTer . . . 3-29
3.8.5 OUTPut:PGMClk<n> . . . 3-30
3.8.5.1 OUTPut:PGMClk<n>[:DATA] . . . 3-30
3.8.5.2 OUTPut:PGMClk<n>:REFerence . . . 3-30
3.8.5.3 OUTPut:PGMClk<n>:RESet . . . 3-31
3.8.6 OUTPut:REFerence . . . 3-31
3.8.6.1 OUTPut:REFerence:AUTO . . . 3-31
3.8.6.2 OUTPut:REFerence:HIGH . . . 3-31
3.8.6.2.1 OUTPut:REFerence:HIGH[:DATA] . . . 3-32
3.8.6.2.2 OUTPut:REFerence:HIGH:INCRement . . . 3-32
3.8.6.3 OUTPut:REFerence:LOW . . . 3-32
3.8.6.3.1 OUTPut:REFerence:LOW[:DATA] . . . 3-32
3.8.6.3.2 OUTPut:REFerence:LOW:INCRement . . . 3-33
3.8.6.3.3 OUTPut:REFerence:ISR[:DATA] . . . 3-33
3.8.6.3.4 OUTPut:REFerence:ISR:INCRement . . . 3-33

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.8.6.4 OUTPut:REFerence:SELect . . . 3-34
3.8.6.5 OUTPut:REFerence:UPDate . . . 3-34
3.8.7 OUTPut:REGister . . . 3-34
3.8.7.1 OUTPut:REGister:SOURce . . . 3-34
3.8.7.2 OUTPut:REGister[:STATe] . . . 3-35
3.8.8 OUTPut:RESet . . . 3-35
3.8.9 OUTPut:SYNC . . . 3-36
3.8.9.1 OUTPut:SYNC:POSition . . . 3-36
3.8.9.2 OUTPut:SYNC[:STATe] . . . 3-36
3.8.10 OUTPut:TIMing . . . 3-37
3.8.10.1 OUTPut:TIMing[:STATe] . . . 3-37
3.8.11 OUTPut:TTLTrg <n> . . . 3-37
3.8.11.1 OUTPut:TTLTrg <n>[:STATe] . . . 3-37
3.9 Route Subsystem . . . 3-39
3.9.1 ROUTe:PATH Subtree . . . 3-39
3.9.1.1 ROUTe:PATH:CATalog? . . . 3-39
3.9.1.2 ROUTe:PATH:DEFine . . . 3-39
3.9.1.3 ROUTe:PATH:DELete . . . 3-40
3.9.1.3.1 ROUTe:PATH:DELete[:NAME] . . . 3-40
3.9.1.3.2 ROUTe:PATH:DELete:ALL . . . 3-40
3.10 Sequence Subsystem . . . 3-41
3.10.1 SEQuence:BRANch? . . . 3-41
3.10.2 SEQuence:DEFine . . . 3-42
3.10.3 SEQuence:DELete . . . 3-42
3.10.3.1 SEQuence:DELete[:NAME] . . . 3-42
3.10.3.2 SEQuence:DELete:ALL . . . 3-43
3.10.4 SEQuence:DIRectory? . . . 3-43
3.10.5 SEQuence:GOSub . . . 3-43
3.10.6 SEQuence:INITialize . . . 3-43
3.10.7 SEQuence:JUMP . . . 3-44
3.10.8 SEQuence:LOOP . . . 3-44
3.10.9 SEQuence:RESet . . . 3-45
3.10.10 SEQuence:STOP . . . 3-45
3.10.11 SEQuence:TABLe . . . 3-45
3.10.12 SEQuence:TIMing . . . 3-46
3.11 Status Subsystem . . . 3-47
3.11.1 STATus:OPERation . . . 3-48
3.11.1.1 STATus:OPERation:CONDition? . . . 3-49
3.11.1.2 STATus:OPERation:ENABle . . . 3-49
3.11.1.3 STATus:OPERation[:EVENt]? . . . 3-49
3.11.1.4 STATus:OPERation:INSTrument . . . 3-50
3.11.1.4.1 STATus:OPERation:INSTrument:CONDition? . . . 3-50
3.11.1.4.2 STATus:OPERation:INSTrument:ENABle . . . 3-50
3.11.1.4.3 STATus:OPERation:INSTrument[:EVENt]? . . . 3-51
3.11.1.4.4 STATus:OPERation:INSTrument:ISUMmary<n> . . . 3-51
3.11.1.4.4.1 STATus:OPERation:INSTrument:ISUMmary<n>:CONDition? . . . 3-51
3.11.1.4.4.2 STATus:OPERation:INSTrument:ISUMmary<n>:ENABle . . . 3-52
3.11.1.4.4.3 STATus:OPERation:INSTrument:ISUMmary<n>[:EVENt]? . . . 3-52
3.11.2 STATus:QUEStionable . . . 3-52
3.11.2.1 STATus:QUEStionable[:EVENt]? . . . 3-52
3.11.2.2 STATus:QUEStionable:CONDition? . . . 3-53
3.11.2.3 STATus:QUEStionable:ENABle . . . 3-53
3.11.3 :PRESet . . . 3-53
3.12 System Subsystem . . . 3-54
3.12.1 SYSTem:ERRor? . . . 3-54

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.12.2 SYSTem:VERSion? . . . 3-55
3.13 Table Subsystem . . . 3-56
3.13.1 TABLe[:DATA] . . . 3-56
3.13.2 TABLe:DEFine . . . 3-57
3.13.3 TABLe:DELete . . . 3-57
3.13.3.1 TABLe:DELete:ALL . . . 3-57
3.13.3.2 TABLe:DELete[:NAME] . . . 3-57
3.13.4 TABLe:DIRectory? . . . 3-58
3.13.5 TABLe:FREE? . . . 3-58
3.13.6 TABLe:JENable . . . 3-58
3.13.7 TABLe:MEMory . . . 3-59
3.13.7.1 TABLe:MEMory:DATA . . . 3-59
3.13.7.2 TABLe:MEMory:FILL . . . 3-60
3.13.7.3 TABLe:MEMory:WORD . . . 3-60
3.13.8 TABLe:SELect . . . 3-61
3.14 Timing Subsystem . . . 3-62
3.14.1 TIMing:CELL . . . 3-62
3.14.2 TIMing[:DATA] . . . 3-64
3.14.3 TIMing:DEFine . . . 3-65
3.14.4 TIMing:DELete . . . 3-66
3.14.4.1 TIMing:DELete:ALL . . . 3-66
3.14.4.2 TABLe:DELete[:NAME] . . . 3-66
3.14.5 TIMing:DIRectory? . . . 3-67
3.14.6 TIMing:PAGE . . . 3-67
3.14.7 TIMing:SETup . . . 3-67
3.14.7.1 TIMing:SETup:CLOCk . . . 3-67
3.14.7.2 TIMing:SETup:CTIMeout . . . 3-68
3.14.7.3 TIMing:SETup:DELay . . . 3-68
3.14.7.4 TIMing:SETup:TSINput2 . . . 3-69
3.14.8 TIMing:TEST . . . 3-69
3.14.8.1 TIMing:TEST:CELL . . . 3-69
3.14.8.2 TIMing:TEST:DELay . . . 3-70
3.14.8.3 TIMing:TEST:LEVel . . . 3-70
3.14.8.4 TIMing:TEST:STRobe . . . 3-70
3.14.8.5 TIMing:TEST:ERRor . . . 3-71
3.14.8.6 TIMing:TEST:COMPare . . . 3-71
3.14.8.7 TIMing:TEST:RESet . . . 3-72
3.15 IEEE 488.2 Common Commands . . . 3-73
3.15.1 *CLS (Clear Status) . . . 3-73
3.15.2 *ESE (Event Status Enable) . . . 3-73
3.15.3 *ESE? (Event Status Enable Query) . . . 3-73
3.15.4 *ESR? (Event Status Register Query) . . . 3-73
3.15.5 *IDN? (Identification Query) . . . 3-74
3.15.6 *OPC (Operation Complete Command) . . . 3-74
3.15.7 *OPC? (Operation Complete Query) . . . 3-75
3.15.8 *RST (Reset Command) . . . 3-75
3.15.9 *SRE (Service Request Enable Command) . . . 3-75
3.15.10 *SRE? (Service Request Enable Query) . . . 3-75
3.15.11 *STB? - Read Status Byte Query (section 10.36) . . . 3-76
3.15.12 *TST? (Self Test Query) . . . 3-76
3.15.13 *WAI (Wait to Continue Command) (section 10.39) . . . 3-77
3.16 VXI WS Interface Commands . . . 3-78
3.16.1 ANOP - Abort Normal Operation . . . 3-78
3.16.2 AMC - Asynchronous Mode Control . . . 3-78
3.16.3 BNOP - Begin Normal Operation . . . 3-79

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.16.4 BAV - Byte Available . . . 3-80
3.16.5 BRQ - Byte Request . . . 3-80
3.16.6 CEV - Control Event . . . 3-80
3.16.7 CLE - Clear . . . 3-81
3.16.8 ENOP - End Normal Operation . . . 3-81
3.16.9 RPR - Read Protocol . . . 3-82
3.16.10 RPE - Read Protocol Error . . . 3-82
3.16.11 RSTB - Read Status Byte . . . 3-83
3.16.12 PFSH - Program Flash . . . 3-83
3.16.13 FTST - Full Ram Test . . . 3-83

Command Summary

1 SCPI MANDATED COMMANDS . . . A-4
2 IEEE COMMON COMMANDS . . . A-4
3 LOW LEVEL WS INTERFACE COMMANDS . . . A-4

Guide to Bus Emulation

1 Introduction . . . B-1
2 Word Generator Overview . . . B-1
2.1 Tables . . . B-2
2.2 Table Looping . . . B-2
2.3 Idle Cycle . . . B-2
3 BUS EMULATION OVERVIEW . . . B-2
3.1 Fields . . . B-3
3.2 Field Timing . . . B-4
3.3 Timing Signals . . . B-4
4 BUS EMULATION TESTING . . . B-5

Worksheets

Improving SR192 Access

1 VXI COMMUNICATION LAYERS . . . D-1
1.1 Register Based Devices . . . D-1
1.2 Message Based Devices . . . D-1
2 THE WORD SERIAL PROTOCOL . . . D-1
2.1 The Word Serial Bottleneck . . . D-2
3 SPEEDING UP TABLE TRANSFERS . . . D-2

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

List of Figures

Figure 3-1 SCPI Header Hierarchy 3-1
Figure 3-2 Error Memory Address to Table Mapping 3-5
Figure 3-3 Status Subsystem Registers 3-48
Figure 3-4 TABLe:DATA Programming Byte Order 3-56
Figure B-1 192 Channels by 128K Words B-1
Figure B-2 192 Channels by 128K Consolidated B-2
Figure B-3 Word Generator Tables B-2
Figure B-4 Table Looping B-2
Figure B-5 Word Timing Slices B-3
Figure B-6 Bus Structure B-3
Figure B-7 Inside Word Transfers B-3
Figure B-8 Field Timing B-4
Figure B-9 Improved Bus Structure B-4
Figure B-10 Testing Input Signals B-5
Figure B-11 Bus Cycle Timing B-5
Figure B-12 Sample UUT B-6
Figure D-1 VXI Communication Layers D-1
Figure D-2 Byte Transfer Protocol Example D-2

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

1 Introduction

Commercial computer-controlled test instruments introduced in the 1960's used a wide variety of nonstandard, proprietary interfaces and communication protocols. In 1975, the Institute of Electrical and Electronic Engineers approved IEEE 488-1975. IEEE 488 defined a standard electrical and mechanical interface for connectors and cables. It also defined handshaking, addressing and general protocol for transmitting individual bytes of data to and from instruments and computers. This standard has been updated and is now IEEE 488.1-1987.

Although it solved the problem of how to send bytes of data between instruments and computers, IEEE 488 did not specify the data bytes' meanings. Instrument manufacturers freely invented new commands as they developed new instruments. The format of data returned from instruments varied as well. By the early 1980's, work began on additional standards to specify how to interpret data sent via IEEE 488.

In 1987, the IEEE released IEEE 488.2-1987, Codes, Formats, Protocols and Common Commands for Use with IEEE 488.1-1987. This standard defined the roles of instruments and controllers in a measurement system and a structured scheme for communication. In particular, IEEE 488.2 described how to send commands to instruments and how to send responses to controllers. It defined some frequently used "housekeeping" commands explicitly, but each instrument manufacturer was left with the task of naming any other types of command and defining their effect. IEEE-488.2 specified how certain types of features should be implemented if they were included in an instrument. It generally did not specify which features or commands should be implemented for a particular instrument. Thus, it was possible that two similar instruments could each conform to IEEE488.2, yet they could have an entirely different command set.

Standard Commands for Programmable Instruments (SCPI) is the new instrument command language for controlling instruments that goes beyond IEEE488.2 to address a wide variety of instrument functions in a standard manner. SCPI promotes consistency, from the remote programming standpoint, between instruments of the same class and between instruments with the same functional capability. For a given measurement function such as frequency or voltage, SCPI defines the specific command set that is available for that function. Thus, two oscilloscopes made by different manufacturers could be used to make frequency measurements in the same way. It is also possible for a SCPI counter to make a frequency measurement using the same commands as an oscilloscope.

SCPI commands are easy to learn, self-explanatory and account for both novice and expert programmer's usage. Once familiar with the organization and structure of SCPI, considerable efficiency gains can be achieved during control program development, independent of the control program language selected.

1.1 Manual Layout

This manual contains SCPI programming and command reference information on the SR192.

The layout of this manual is in three sections described below:

1. Introduction — This section.
2. SCPI Programming — Programming examples using SR192 SCPI commands.
3. Command Reference — SR192 SCPI command reference.

In addition, five appendices are included:

A. Command Summary
B. Guide to Bus Emulation
C. Worksheets
D. Speeding Up SR192 Access

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

2 SCPI Programming

This section describes programming the SR192 digital resource module using the SCPI remote command language.

2.1 Overview

The following sections describe SR192 programming concepts using SCPI commands and the VISA driver.

2.1.1 SR192 SCPI Programming Steps

Programming the SR192 is comprised of ten steps listed below:

Step 1	INITIALIZE the VXI session and establish a communication channel with the SR192.
Step 2	CONFIGURE the SR192 global hardware selections for your application.
Step 3	ROUTE the channels by identifying the physical channels with group names.
Step 4	Define the timing parameters(TIMING SET(s)) to govern the I/O transfers.
Step 5	Define the data TABLES and populate as appropriate.
Step 6	Define the SEQUENCE(s) of operation if using multiple timing patterns .
Step 7	Enable output drivers (SOURCE) .
Step 8	Issue SETUP And EXECUTE commands.
Step 9	Utilize STATUS and POST PROCESS functions to evaluate/analyze results .
	(Repeat steps 5 through 9 for each set of unique patterns.)
Step 10	CLOSE the VXI programming session.

2.1.2 SR192 Plug&Play/VISA

The VXIplug&play Systems Alliance was formed on September 22, 1993 with the goal of increasing end-user success and multi vendor interoperability for VXIbus systems. To achieve this goal, VXIplug&play defines and implements new levels of standardization to simplify multi vendor VXI system integration to benefit both end-users and vendors. As a result, VXIplug&play products are easy to use, thanks to new standards for both hardware and software.

At the heart of these standards is the Virtual Instrument Software Architecture, or VISA, the I/O software standard on which all VXI plug&play software components are based. In the past, there were many different I/O software products to control GPIB and VXI. Software written with these various libraries supplied by individual vendors was directly and uniquely tied to the hardware these vendors produced. The VISA standard, endorsed by over 35 of the largest instrumentation companies in the industry including Tektronix, Hewlett-Packard, and National Instruments, unifies the industry to make software interoperable, reusable, and able to stand the test of time.

The following VISA Plug & Play functions will be used in the examples to communicate and control the SR192.

```
ViStatus viOpenDefaultRM(ViPSession sesn)
ViStatus viOpen(ViSession sesn, ViRsrc rsrcName, ViAccessMode accessMode, ViUInt32 timeout, ViPSession vi)
ViStatus viWrite(ViSession vi, ViBuf buf, ViUInt32 count, ViPUInt32 retCount)
ViStatus viRead(ViSession vi, ViPBuf buf, ViUInt32 count, ViPUInt32 retCount)
```

Refer to the VISA documentation supplied by the Slot O manufacturer for the descriptions of the functions listed above.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

2.1.3 Programming Language

The examples listed in this section will be shown using the standard “C” programming language, however any programming language may be used that supports VISA.

2.2 Programming Examples

The following main program is used to call each of the ten steps which are coded as functions:

```
/*
      File name:          ex1.prj
      Date:               11/19/98
      Description:  SCPI programming manual example number one.
                          Initialize the SR192 and check for installed TSA and DRA1 module.
*/
#include <ansi_c.h>
#include <visa.h>

/* Prototype declarations */
ViStatus sr192_init(void);
ViStatus sr192_config(void);
ViStatus sr192_route(void);
ViStatus sr192_timing(void);
ViStatus sr192_table(void);
ViStatus sr192_seq(void);
ViStatus sr192_drven(void);
ViStatus sr192_exec(void);
ViStatus sr192_post(void);
ViStatus sr192_close(void);

/* Global Declarations */
static ViSession SR;
static ViSession RM;
ViStatus status;
ViChar    response[1024];

/* Main function definition */
int main (int argc, char *argv[])
{

      /* Step one, initialize the SR192 */
      status = sr192_init();

      /* Step two, configure the SR192 */
      if (status == VI_SUCCESS)
            status = sr192_config();

      /* Step three, route the SR192 channels */
      if (status == VI_SUCCESS)
            status = sr192_route();

      /* Step four, define the SR192 timing */
      if (status == VI_SUCCESS)
            status = sr192_timing();

      /* Step five, define the SR192 patterns */
      if (status == VI_SUCCESS)
            status = sr192_table();

      /* Step six, define the SR192 sequence */
      if (status == VI_SUCCESS)
            status = sr192_seq();

      /* Step seven, enable the output drivers */
      if (status == VI_SUCCESS)
            status = sr192_drven();

      /* Step eight, start sequence */
      if (status == VI_SUCCESS)
            status = sr192_exec();

      /* Step nine, analyze results */
      if (status == VI_SUCCESS)
            status = sr192_post();

      /* Step ten, close session */
      if (status == VI_SUCCESS)
            status = sr192_close();
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The following sections list the functions for each of the ten steps listed in section 2.1.1.

The following SR192 hardware was used for the examples:

1. TSA SR101
2. DRA1 SR115
3. All other I/O and timing module slots empty.
4. CPU firmware revision 1.31

This example programs the static ram memory on the 20251 Training Pod (Rev. B).

2.2.1 Initialize the VXI Session

The first step in programming the SR192 is to establish a VISA session.

The following code illustrates initializing a SR192 as well as checking for errors and verifying the I/O modules installed in the system.

```
/* Function to initialize the SR192 installed at logical address 2 */
ViStatus sr192_init(void)
{
	ViUInt32 retcnt;
	ViInt32 cnt;

	/* Open session to the resource manager, report if error and exit. */
	if ((status = viOpenDefaultRM (&RM))) {
		printf("Not able to open session to resource manager; status = %x\n", status);
		exit(0);
	}

	/* Open session to SR192 with logical address 2. */
	if ((status = viOpen (RM, "VXI::2", VI_NULL, VI_NULL, &SR))) {
		printf("Not able to open session to SR192 at LA2; status = %x\n", status);
		exit(0);
	}

	/* Query the SR192 id. */
	if ((status = viWrite (SR, (ViBuf)"*IDN?\n", 5, &retcnt))) {
		printf("Error sending *IDN? to SR192 at LA2; status = %x\n", status);
		exit(0);
	}

	/* Read the *IDN? query results. */
	if ((status = viRead (SR, (ViPBuf)response, 100, &retcnt))) {
		printf("Error reading the *IDN? Query results from the SR192 at LA2; status = %x\n", status);
		exit(0);
	}
	response[retcnt] = VI_NULL;
	printf("*IDN? query returned %d bytes; %s\n", retcnt, response);

	/* Check TSA timing module slot position for installed modules.
	   Select the module then send the error query command.  Selecting
	   a slot position when a module is not installed generates an
	   execution error. */
	if ((status = viWrite (SR, (ViBuf)"MODULE:SELECT TSA;:SYSTEM:ERROR?", 32, &retcnt))) {
		printf("Error selecting TSA to SR192 at LA2; status = %x\n", status);
		exit(0);
	}

	/* Read the error query results. */
	if ((status = viRead (SR, (ViPBuf)response, 100, &retcnt))) {
		printf("Error reading the error query results from the SR192 at LA2; status = %x\n", status);
		exit(0);
	}
	response[retcnt] = VI_NULL;
	printf("SYSTEM:ERROR? query returned %d bytes; %s\n", retcnt, response);

	/* Check return string to see if module installed. */
	cnt = atoi(response);
	if (cnt == 0) { /* "0, No error" returned by SYSTEM:ERROR? query */
		/* Send module status query */
		if ((status = viWrite (SR, (ViBuf)"MODULE:STATUS?", 14, &retcnt))) {
			printf("Error sending TSA status query to SR192 at LA2; status = %x\n", status);
			exit(0);
		}

		/* Read the module status query results. */
		if ((status = viRead (SR, (ViPBuf)response, 100, &retcnt))) {
			printf("Error reading the TSA status query results from the SR192 at LA2; status = %x\n", status);
			exit(0);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
        }
        response[retcnt] = VI_NULL;
        printf("MODULE:STATUS? query returned %d bytes; %s\n", retcnt, response);

        /* upper byte is module id, check module reference manual. */
        cnt = atoi(response);
        printf("TSA module id: %x ", cnt >> 8);

        /* Check bit 0 of the return string to see if module passed self test. */
        if (cnt & 1)
            printf("passed selftest\n");
        else
            printf("failed selftest\n");
    } else
        printf("TSA module not installed");

    return status;
}
```

2.2.2 Configuring SR192 settings

The SR192 configuration settings include:

1. Master mode state,
2. TSA link state.
3. TSB link state.
4. Voltage group reference levels.

The following code illustrates setting the SR192 as independent. The example program is written for a system where TSB is not installed and fixed TTL I/O modules are installed. The SCPI commands to program the TSB link as well as the voltage group references are shown as comments.

```
/* Function to program the SR192 configuration settings */
ViStatus sr192_config(void)
{
    ViUInt32 retcnt;
    ViInt32 cnt;

    /* set the SR192 as independent */
    if ((status = viWrite (SR, (ViBuf)"OUTPUT:MASTER OFF", 17, &retcnt))) {
        printf("Error sending output master command to SR192 at LA2; status = %x\n", status);
            exit(0);
    }

    /* Check for command error */
    if ((status = viWrite (SR, (ViBuf)"SYSTEM:ERROR?", 13, &retcnt))) {
        printf("Error sending the error query command to SR192 at LA2; status = %x\n", status);
            exit(0);
    }
    /* read response */
    if ((status = viRead (SR, (ViPBuf)response, 100, &retcnt))) {
        printf("Error reading the error query results from the SR192 at LA2; status = %x\n", status);
        exit(0);
    }
    response[retcnt] = VI_NULL;
    printf("SYSTEM:ERROR? query returned %d bytes; %s\n", retcnt, response);
    cnt = atoi(response);
    if (cnt != 0)
        printf("Error sending OUTPUT:MASTER command\n");

    /*
        If TSB installed then program TSA, TSB link on or off with the
        following commands:

        MODULE:SELECT TSB
        MODULE:LINK ON  (links TSA and TSB)
        MODULE:LINK OFF (TSA and TSB not linked)
    */

    /*
        If variable voltage I/O modules installed, program the input/output
        references for each voltage group with the following commands:

        OUTPUT:REFERENCE:SELECT VGRP1
        OUTPUT:REFERENCE:HIGH 5.0
        OUTPUT:REFERENCE:LOW 0.0
        INPUT:REFERENCE:SELECT VGRP1
        INPUT:REFERENCE:HIGH 2.4
        INPUT:REFERENCE:LOW 0.8
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
            (repeat for VGRP2 and VGRP3)
            OUTPUT:REFERENCE:UPDATE  (programs the DACs with the new values)
        */

        return status;
}
```

2.2.3 Route the I/O Channels

Routing the I/O channels allows the user to create logical channel groups to program or query the I/O control settings as well as the pattern memory. The physical routing of all I/O channels to the front panel is fixed.

The following code creates and programs two groups, ADDR_BUS and DATA_BUS. Each group is eight channels.

```
/* Function to route the I/O channel of the SR192 installed at logical address 2 */
ViStatus sr192_route(void)
{
        ViUInt32 retcnt;
        ViInt32 cnt;

        /* Create first group: ADDR_BUS */
        if ((status = viWrite (SR, (ViBuf)"ROUTE:PATH:DEFINE ADDR_BUS,(@1:8)", 33, &retcnt))) {
                printf("Error sending route path define (ADDR_BUS) command to SR192 at LA2; status = %x\n", status);
                        exit(0);
        }

        /* Create next group: DATA_BUS */
        if ((status = viWrite (SR, (ViBuf)"ROUTE:PATH:DEFINE DATA_BUS,(@9:16)", 34, &retcnt))) {
                printf("Error sending route path define (DATA_BUS) command to SR192 at LA2; status = %x\n", status);
                        exit(0);
        }

        /* Check for command error */
        if ((status = viWrite (SR, (ViBuf)"SYSTEM:ERROR?", 13, &retcnt))) {
                printf("Error sending error query command to SR192 at LA2; status = %x\n", status);
                exit(0);
        }
        /* read response */
        if ((status = viRead (SR, (ViPBuf)response, 100, &retcnt))) {
                printf("Error reading the error query results from the SR192 at LA2; status = %x\n", status);
                exit(0);
        }
        response[retcnt] = VI_NULL;
        printf("SYSTEM:ERROR? query returned %d bytes; %s\n", retcnt, response);
        cnt = atoi(response);
        if (cnt != 0)
                printf("Error sending ROUTE:PATH:DEFINE commands\n");

        /* Program the output register state for both groups. */
        if ((status = viWrite (SR, (ViBuf)"OUTPUT:REGISTER:STATE ADDR_BUS,ON", 33, &retcnt))) {
                printf("Error sending output register command (ADDR_BUS) to SR192 at LA2; status = %x\n", status);
                exit(0);
        }
        if ((status = viWrite (SR, (ViBuf)"OUTPUT:REGISTER:STATE DATA_BUS,ON", 33, &retcnt))) {
                printf("Error sending output register command (DATA_BUS) to SR192 at LA2; status = %x\n", status);
                exit(0);
        }

        /* Program the output register strobe source for both groups. */
        if ((status = viWrite (SR, (ViBuf)"OUTPUT:REGISTER:SOURCE ADDR_BUS,STIM_LOAD", 41, &retcnt))) {
                printf("Error sending output register source command (ADDR_BUS) to SR192 at LA2; status = %x\n", status);
                exit(0);
        }
        if ((status = viWrite (SR, (ViBuf)"OUTPUT:REGISTER:SOURCE DATA_BUS,STIM_LOAD", 41, &retcnt))) {
                printf("Error sending output register source command (DATA_BUS) to SR192 at LA2; status = %x\n", status);
                exit(0);
        }

        /* Program the output enable source for both groups. */
        if ((status = viWrite (SR, (ViBuf)"OUTPUT:ENABLE:SOURCE ADDR_BUS,TSENABLE1", 39, &retcnt))) {
                printf("Error sending output enable source command (ADDR_BUS) to SR192 at LA2; status = %x\n", status);
                exit(0);
        }
        if ((status = viWrite (SR, (ViBuf)"OUTPUT:ENABLE:SOURCE DATA_BUS,TSENABLE2", 39, &retcnt))) {
                printf("Error sending output enable source command (DATA_BUS) to SR192 at LA2; status = %x\n", status);
                exit(0);
        }
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
    /* Program the output enable delay for both groups. */
    if ((status = viWrite (SR, (ViBuf)"OUTPUT:ENABLE:DELAY ADDR_BUS,0", 31, &retcnt))) {
        printf("Error sending output enable delay command (ADDR_BUS) to SR192 at LA2; status = %x\n", status);
        exit(0);
    }
    if ((status = viWrite (SR, (ViBuf)"OUTPUT:ENABLE:DELAY DATA_BUS,0", 31, &retcnt))) {
        printf("Error sending output enable delay command (DATA_BUS) to SR192 at LA2; status = %x\n", status);
        exit(0);
    }

    /* Program the input strobe source for both groups. */
    if ((status = viWrite (SR, (ViBuf)"INPUT:STROBE:SOURCE ADDR_BUS,TSSTROBE1", 38, &retcnt))) {
        printf("Error sending input strobe source command (ADDR_BUS) to SR192 at LA2; status = %x\n", status);
        exit(0);
    }
    if ((status = viWrite (SR, (ViBuf)"INPUT:STROBE:SOURCE DATA_BUS,TSSTROBE2", 38, &retcnt))) {
        printf("Error sending input strobe source command (DATA_BUS) to SR192 at LA2; status = %x\n", status);
        exit(0);
    }

    /* Program the input strobe delay for both groups. */
    if ((status = viWrite (SR, (ViBuf)"INPUT:STROBE:DELAY ADDR_BUS,0", 29, &retcnt))) {
        printf("Error sending input strobe delay command (ADDR_BUS) to SR192 at LA2; status = %x\n", status);
        exit(0);
    }
    if ((status = viWrite (SR, (ViBuf)"INPUT:STROBE:DELAY DATA_BUS,0", 29, &retcnt))) {
        printf("Error sending input strobe delay command (ADDR_BUS) to SR192 at LA2; status = %x\n", status);
        exit(0);
    }

    /* Program the algorithmic mode for both groups. */
    if ((status = viWrite (SR, (ViBuf)"CHANNEL:MODE ADDR_BUS,HOLD", 26, &retcnt))) {
        printf("Error sending channel mode command (ADDR_BUS) to SR192 at LA2; status = %x\n", status);
        exit(0);
    }
    if ((status = viWrite (SR, (ViBuf)"CHANNEL:MODE DATA_BUS,HOLD", 26, &retcnt))) {
        printf("Error sending channel mode command (DATA_BUS) to SR192 at LA2; status = %x\n", status);
        exit(0);
    }

    /* Check for command errors */
    if ((status = viWrite (SR, (ViBuf)"SYSTEM:ERROR?", 13, &retcnt))) {
        printf("Error sending system error query command to SR192 at LA2; status = %x\n", status);
        exit(0);
    }
    /* read response */
    if ((status = viRead (SR, (ViPBuf)response, 100, &retcnt))) {
        printf("Error reading the error query results from the SR192 at LA2; status = %x\n", status);
        exit(0);
    }
    response[retcnt] = VI_NULL;
    printf("SYSTEM:ERROR? query returned %d bytes; %s\n", retcnt, response);
    cnt = atoi(response);
    if (cnt != 0)
        printf("Error sending group control commands\n");

    return status;
}
```

2.2.4 Program the timing module

Programming the timing module consists of the following:

1. Clock source
2. Delay value
3. Cycle timeout count
4. TSINPUT2 test mode
5. Timing cycle memory

The following code programs the timing parameters as well as creates two timing cycles, WRITE and READ.

```
/* Function to define and program the timing module of the SR192 installed at logical address 2 */
ViStatus sr192_timing(void)
{
    ViUInt32 retcnt;
    ViInt32 cnt;
    ViUInt8 write_data[12] = {0xff, 0xfe,       /* SR_CLK low */
                              0xff, 0xfb,       /* STIM_LOAD low */
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
                                0xf7, 0xe7,     /* TSENABLE1, TSENABLE2 and TSOUT1 low */
                                0xf7, 0xe7,     /* TSENABLE1, TSENABLE2 and TSOUT1 low */
                                0xf7, 0xe7,     /* TSENABLE1, TSENABLE2 and TSOUT1 low */
                                0x7f, 0xff};    /* LSTBIT low */
ViUInt8 read_data[16] = {0xff, 0xfe,    /* SR_CLK low */
                                0xff, 0xfb,     /* STIM_LOAD low */
                                0xff, 0xf7,     /* TSENABLE1 low */
                                0xff, 0xf7,     /* TSENABLE1 low */
                                0xff, 0xf7,     /* TSENABLE1 low */
                                0xff, 0xf7,     /* TSENABLE1 low */
                                0xff, 0x97,     /* TSENABLE1, TSSTROBE1 and TSSTROBE2 low */
                                0x7f, 0xff};    /* LSTBIT low */

/* Program the timing module parameters */
if ((status = viWrite (SR, (ViBuf)"MODULE:SELECT TSA", 17, &retcnt))) {
        printf("Error sending module select command to SR192 at LA2; status = %x\n", status);
        exit(0);
}

if ((status = viWrite (SR, (ViBuf)"TIMING:SETUP:CLOCK 10", 21, &retcnt))) {
        printf("Error sending clock source command to SR192 at LA2; status = %x\n", status);
        exit(0);
}

if ((status = viWrite (SR, (ViBuf)"TIMING:SETUP:DELAY 255", 22, &retcnt))) {
        printf("Error sending delay value command to SR192 at LA2; status = %x\n", status);
        exit(0);
}

if ((status = viWrite (SR, (ViBuf)"TIMING:SETUP:CTIMEOUT 50", 24, &retcnt))) {
        printf("Error sending cycle timeout command to SR192 at LA2; status = %x\n", status);
        exit(0);
}

if ((status = viWrite (SR, (ViBuf)"TIMING:SETUP:TSINPUT2 LEVEL", 27, &retcnt))) {
        printf("Error sending tsinput2 mode command to SR192 at LA2; status = %x\n", status);
        exit(0);
}

/* Check for command error */
if ((status = viWrite (SR, (ViBuf)"SYSTEM:ERROR?", 13, &retcnt))) {
        printf("Error sending system error query command to SR192 at LA2; status = %x\n", status);
        exit(0);
}
/* read response */
if ((status = viRead (SR, (ViPBuf)response, 100, &retcnt))) {
        printf("Error reading the error query results from the SR192 at LA2; status = %x\n", status);
        exit(0);
}
response[retcnt] = VI_NULL;
printf("SYSTEM:ERROR? query returned %d bytes; %s\n", retcnt, response);
cnt = atoi(response);
if (cnt != 0)
        printf("Error sending timing parameter commands\n");

/* Define the timing cycles */
if ((status = viWrite (SR, (ViBuf)"TIMING:DEFINE WRITE,6", 21, &retcnt))) {
        printf("Error sending timing define (WRITE) command to SR192 at LA2; status = %x\n", status);
        exit(0);
}

if ((status = viWrite (SR, (ViBuf)"TIMING:DEFINE READ,8", 20, &retcnt))) {
        printf("Error sending timing define (READ) command to SR192 at LA2; status = %x\n", status);
        exit(0);
}

/* Check for command error */
if ((status = viWrite (SR, (ViBuf)"SYSTEM:ERROR?", 13, &retcnt))) {
        printf("Error sending system error query command to SR192 at LA2; status = %x\n", status);
        exit(0);
}
/* read response */
if ((status = viRead (SR, (ViPBuf)response, 100, &retcnt))) {
        printf("Error reading the error query results from the SR192 at LA2; status = %x\n", status);
        exit(0);
}
response[retcnt] = VI_NULL;
printf("SYSTEM:ERROR? query returned %d bytes; %s\n", retcnt, response);
cnt = atoi(response);
if (cnt != 0)
        printf("Error sending timing define commands\n");
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
	/* program timing cycle memory */
	sprintf(response, "TIMING:DATA WRITE,#212");
	memcpy(&response[22], write_data, 12);
	if ((status = viWrite (SR, (ViBuf)response, 34, &retcnt))) {
		printf("Error sending timing data (WRITE) command to SR192 at LA2; status = %x\n", status);
		exit(0);
	}

	sprintf(response, "TIMING:DATA READ,#216");
	memcpy(&response[21], read_data, 16);
	if ((status = viWrite (SR, (ViBuf)response, 37, &retcnt))) {
		printf("Error sending timing data (READ) command to SR192 at LA2; status = %x\n", status);
		exit(0);
	}

	/* Check for command error */
	if ((status = viWrite (SR, (ViBuf)"SYSTEM:ERROR?", 13, &retcnt))) {
		printf("Error sending system error query command to SR192 at LA2; status = %x\n", status);
		exit(0);
	}
	/* read response */
	if ((status = viRead (SR, (ViPBuf)response, 100, &retcnt))) {
		printf("Error reading the error query results from the SR192 at LA2; status = %x\n", status);
		exit(0);
	}
	response[retcnt] = VI_NULL;
	printf("SYSTEM:ERROR? query returned %d bytes; %s\n", retcnt, response);
	cnt = atoi(response);
	if (cnt != 0)
		printf("Error programming  timing set memory\n");

	return status;
}
```

2.2.5 Program the Pattern Memory

The pattern memory must be programmed the stimulus data (OUTPUT and TRISTATE) as well as the response data (MASK and EXPECT)

The following code creates two tables, SR_NRTC and SR_RTC. Both tables contain sixteen patterns. SR_NRTC does not program the real time compare.

```
	/* Function to define and program the SR192 pattern memory */
	ViStatus sr192_table(void)
	{
		ViUInt32 retcnt;
		ViInt32 cnt;
		ViUInt8 output_data[16] =  {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15};

		/* Define a table (SR_NRTC) to hold the stimulus/response patterns, no real time compare */
		if ((status = viWrite (SR, (ViBuf)"TABLE:DEFINE SR_NRTC,16", 23, &retcnt))) {
			printf("Error sending table define command to SR192 at LA2; status = %x\n", status);
			exit(0);
		}

		/* Check for command error */
		if ((status = viWrite (SR, (ViBuf)"SYSTEM:ERROR?", 13, &retcnt))) {
			printf("Error sending system error query command to SR192 at LA2; status = %x\n", status);
			exit(0);
		}
		/* read response */
		if ((status = viRead (SR, (ViPBuf)response, 100, &retcnt))) {
			printf("Error reading the error query results from the SR192 at LA2; status = %x\n", status);
			exit(0);
		}
		cnt = atoi(response);
		if (cnt != 0) {
			printf("Error sending table define (SR_NRTC) command\n");
			response[retcnt] = VI_NULL;
			printf("SYSTEM:ERROR? query returned %d bytes; %s\n", retcnt, response);
		}

		/* program output pattern memory for ADDR_BUS */
		if ((status = viWrite (SR, (ViBuf)"TABLE:SELECT OUTPUT", 19, &retcnt))) {
			printf("Error sending table select command to SR192 at LA2; status = %x\n", status);
			exit(0);
		}

		sprintf(response, "TABLE:MEMORY:DATA SR_NRTC,ADDR_BUS,#216");
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
memcpy(&response[39], output_data, 16);
if ((status = viWrite (SR, (ViBuf)response, 55, &retcnt))) {
        printf("Error sending table data (ADDR_BUS output) command to SR192 at LA2; status = %x\n", status);
        exit(0);
}

/* program output pattern memory for DATA_BUS to walking one pattern */
/* set 1st word to 1 */
if ((status = viWrite (SR, (ViBuf)"TABLE:MEMORY:WORD SR_NRTC,DATA_BUS,1,1", 38, &retcnt))) {
        printf("Error sending table memory word command to SR192 at LA2; status = %x\n", status);
        exit(0);
}

/* program remaining words using rotate fill (walking one ) */
if ((status = viWrite (SR, (ViBuf)"TABLE:MEMORY:FILL SR_NRTC,DATA_BUS,ROTATE,1,1", 45, &retcnt))) {
        printf("Error sending table memory fill command to SR192 at LA2; status = %x\n", status);
        exit(0);
}

/* program tristate pattern memory using compliment fill */
if ((status = viWrite (SR, (ViBuf)"TABLE:SELECT TRISTATE", 21, &retcnt))) {
        printf("Error sending table select command to SR192 at LA2; status = %x\n", status);
        exit(0);
}

if ((status = viWrite (SR, (ViBuf)"TABLE:MEMORY:FILL SR_NRTC,ADDR_BUS,COMPLIMENT,1", 47, &retcnt))) {
        printf("Error sending table memory fill command (ADDR_BUS tristate) to SR192 at LA2; status = %x\n", status);
        exit(0);
}

if ((status = viWrite (SR, (ViBuf)"TABLE:MEMORY:FILL SR_NRTC,DATA_BUS,COMPLIMENT,1", 47, &retcnt))) {
        printf("Error sending table memory fill command (DATA_BUS tristate) to SR192 at LA2; status = %x\n", status);
        exit(0);
}

/* Check for command error */
if ((status = viWrite (SR, (ViBuf)"SYSTEM:ERROR?", 13, &retcnt))) {
        printf("Error sending system error query command to SR192 at LA2; status = %x\n", status);
        exit(0);
}
/* read response */
if ((status = viRead (SR, (ViPBuf)response, 100, &retcnt))) {
        printf("Error reading the error query results from the SR192 at LA2; status = %x\n", status);
        exit(0);
}
cnt = atoi(response);
if (cnt != 0) {
        printf("Error programming SR_NRTC pattern memory\n");
        response[retcnt] = VI_NULL;
        printf("SYSTEM:ERROR? query returned %d bytes; %s\n", retcnt, response);
}

/* Define a table (SR_RTC) to hold the stimulus/response patterns, real time compare */
/* copy stimulus data from SR_NRTC then program response memory */
if ((status = viWrite (SR, (ViBuf)"TABLE:DEFINE SR_RTC,SR_NRTC", 27, &retcnt))) {
        printf("Error sending table define command to SR192 at LA2; status = %x\n", status);
        exit(0);
}

/* Check for command error */
if ((status = viWrite (SR, (ViBuf)"SYSTEM:ERROR?", 13, &retcnt))) {
        printf("Error sending system error query command to SR192 at LA2; status = %x\n", status);
        exit(0);
}
/* read response */
if ((status = viRead (SR, (ViPBuf)response, 100, &retcnt))) {
        printf("Error reading the error query results from the SR192 at LA2; status = %x\n", status);
        exit(0);
}
cnt = atoi(response);
if (cnt != 0) {
        printf("Error creating (SR_RTC) table\n");
        response[retcnt] = VI_NULL;
        printf("SYSTEM:ERROR? query returned %d bytes; %s\n", retcnt, response);
}

/* program expect pattern memory for ADDR_BUS */
if ((status = viWrite (SR, (ViBuf)"TABLE:SELECT EXPECT", 19, &retcnt))) {
        printf("Error sending table select command to SR192 at LA2; status = %x\n", status);
        exit(0);
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
        /* expect should match output */
        sprintf(response, "TABLE:MEMORY:DATA SR_RTC,ADDR_BUS,#216");
        memcpy(&response[38], output_data, 16);
        if ((status = viWrite (SR, (ViBuf)response, 54, &retcnt))) {
                printf("Error sending table data (ADDR_BUS expect) command to SR192 at LA2; status = %x\n", status);
                exit(0);
        }

        /* program output pattern memory for DATA_BUS to walking one pattern */
        /* set 1st word to 1 */
        if ((status = viWrite (SR, (ViBuf)"TABLE:MEMORY:WORD SR_RTC,DATA_BUS,1,1", 37, &retcnt))) {
                printf("Error sending table memory word command to SR192 at LA2; status = %x\n", status);
                exit(0);
        }

        /* program remaining words using rotate fill (walking one ) */
        if ((status = viWrite (SR, (ViBuf)"TABLE:MEMORY:FILL SR_RTC,DATA_BUS,ROTATE,1,1", 44, &retcnt))) {
                printf("Error sending table memory fill command to SR192 at LA2; status = %x\n", status);
                exit(0);
        }

        /* program mask pattern memory using compliment fill */
        if ((status = viWrite (SR, (ViBuf)"TABLE:SELECT MASK", 17, &retcnt))) {
                printf("Error sending table select command to SR192 at LA2; status = %x\n", status);
                exit(0);
        }

        if ((status = viWrite (SR, (ViBuf)"TABLE:MEMORY:FILL SR_RTC,ADDR_BUS,COMPLIMENT,1", 46,  &retcnt))) {
                printf("Error sending table memory fill command (ADDR_BUS mask) to SR192 at LA2; status = %x\n", status);
                exit(0);
        }

        if ((status = viWrite (SR, (ViBuf)"TABLE:MEMORY:FILL SR_RTC,DATA_BUS,COMPLIMENT,1", 46, &retcnt))) {
                printf("Error sending table memory fill command (DATA_BUS mask) to SR192 at LA2; status = %x\n", status);
                exit(0);
        }

        /* Check for command error */
        if ((status = viWrite (SR, (ViBuf)"SYSTEM:ERROR?", 13, &retcnt))) {
                printf("Error sending system error query command to SR192 at LA2; status = %x\n", status);
                exit(0);
        }
        /* read response */
        if ((status = viRead (SR, (ViPBuf)response, 100, &retcnt))) {
                printf("Error reading the error query results from the SR192 at LA2; status = %x\n", status);
                exit(0);
        }
        cnt = atoi(response);
        if (cnt != 0) {
                printf("Error programming SR_RTC pattern memory\n");
                response[retcnt] = VI_NULL;
                printf("SYSTEM:ERROR? query returned %d bytes; %s\n", retcnt, response);
        }

        return status;
}
```

2.2.6 Define and Program the Execution Sequence

The following code defines three sequences, “WRITE_RAM”, “READ_RAM”, and “RW_RAM”. The first two sequences do not utilize the real time compare.

```
        /* Function to create execution sequences */
        ViStatus sr192_seq(void)
        {
                ViUInt32 retcnt;
                ViInt32 cnt;

                /* Define a sequence (WRITE_RAM) to program data into the RAM, no real time compare */
                if ((status = viWrite (SR, (ViBuf)"SEQUENCE:DEFINE WRITE_RAM,WRITE,SR_NRTC", 39, &retcnt))) {
                        printf("Error sending sequence define command to SR192 at LA2; status = %x\n", status);
                        exit(0);
                }

                /* Define a sequence (READ_RAM) to read data from the RAM, no real time compare */
                if ((status = viWrite (SR, (ViBuf)"SEQUENCE:DEFINE READ_RAM,READ,SR_NRTC", 37, &retcnt))) {
                        printf("Error sending sequence define command to SR192 at LA2; status = %x\n", status);
                        exit(0);
                }
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
        /* Define a sequence (RW_RAM) to program then read data from the RAM, real time compare */
        if ((status = viWrite (SR, (ViBuf)"SEQUENCE:DEFINE WR_RAM,WRITE,SR_RTC,READ,SR_RTC", 47, &retcnt))) {
                printf("Error sending sequence define command to SR192 at LA2; status = %x\n", status);
                exit(0);
        }

        /* Check for command error */
        if ((status = viWrite (SR, (ViBuf)"SYSTEM:ERROR?", 13, &retcnt))) {
                printf("Error sending system error query command to SR192 at LA2; status = %x\n", status);
                exit(0);
        }
        /* read response */
        if ((status = viRead (SR, (ViPBuf)response, 100, &retcnt))) {
                printf("Error reading the error query results from the SR192 at LA2; status = %x\n", status);
                exit(0);
        }
        cnt = atoi(response);
        if (cnt != 0) {
                printf("Error creating sequences\n");
                response[retcnt] = VI_NULL;
                printf("SYSTEM:ERROR? query returned %d bytes; %s\n", retcnt, response);
        }

        return status;
}
```

2.2.7 Enable Output Drivers

The following code enable the output drivers of the SR192. On power up the drivers are all disabled. The timing and control output drivers are controled by the “OUTPUT:TIMING:STATE” command. The I/O module channels are controlled by three conditions, “OUTPUT:CHANNEL:STATE” command, TRISTATE memory level (section 2.2.5) and the timing set group enable signal (section 2.2.4).

The following code enables the timing and channel output drivers, the channel drivers will still be disabled until the sequence is executed.

```
/* Function to enable the SR192 output drivers. */
ViStatus sr192_drven(void)
{
        ViUInt32 retcnt;
        ViInt32 cnt;

        /* Enable both the timing and channel output drivers */
        if ((status = viWrite (SR, (ViBuf)"OUTPUT:TIMING:STATE ON", 22, &retcnt))) {
                printf("Error sending output timing state command to SR192 at LA2; status = %x\n", status);
                exit(0);
        }

        if ((status = viWrite (SR, (ViBuf)"OUTPUT:CHANNEL:STATE ON", 23, &retcnt))) {
                printf("Error sending output channel state command to SR192 at LA2; status = %x\n", status);
                exit(0);
        }

        /* Check for command error */
        if ((status = viWrite (SR, (ViBuf)"SYSTEM:ERROR?", 13, &retcnt))) {
                printf("Error sending system error query command to SR192 at LA2; status = %x\n", status);
                exit(0);
        }
        /* read response */
        if ((status = viRead (SR, (ViPBuf)response, 100, &retcnt))) {
                printf("Error reading the error query results from the SR192 at LA2; status = %x\n", status);
                exit(0);
        }
        cnt = atoi(response);
        if (cnt != 0) {
                printf("Error turning on the output drivers\n");
                response[retcnt] = VI_NULL;
                printf("SYSTEM:ERROR? query returned %d bytes; %s\n", retcnt, response);
        }

        return status;
}
```

2.2.8 Run (Execute) the Defined Sequences

The following code sets the run mode to single and executes each of the defined sequences.

```
/* Function to execute the SR192 sequences */
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
ViStatus sr192_exec(void)
{
	ViUInt32 retcnt;
	ViInt32 cnt;

	/* set the single mode */
	if ((status = viWrite (SR, (ViBuf)"EXECUTE:MODE SINGLE", 19, &retcnt))) {
		printf("Error sending execute mode command to SR192 at LA2; status = %x\n", status);
		exit(0);
	}

	if ((status = viWrite (SR, (ViBuf)"EXECUTE:SEQUENCE WRITE_RAM", 26, &retcnt))) {
		printf("Error sending execute sequence (WRITE_RAM) command to SR192 at LA2; status = %x\n", status);
		exit(0);
	}

	if ((status = viWrite (SR, (ViBuf)"EXECUTE:SEQUENCE READ_RAM", 25, &retcnt))) {
		printf("Error sending execute sequence (READ_RAM) command to SR192 at LA2; status = %x\n", status);
		exit(0);
	}

	if ((status = viWrite (SR, (ViBuf)"EXECUTE:SEQUENCE WR_RAM", 23, &retcnt))) {
		printf("Error sending execute sequence (WR_RAM) command to SR192 at LA2; status = %x\n", status);
		exit(0);
	}

	/* Check for command error */
	if ((status = viWrite (SR, (ViBuf)"SYSTEM:ERROR?", 13, &retcnt))) {
		printf("Error sending system error query command to SR192 at LA2; status = %x\n", status);
		exit(0);
	}
	/* read response */
	if ((status = viRead (SR, (ViPBuf)response, 100, &retcnt))) {
		printf("Error reading the error query results from the SR192 at LA2; status = %x\n", status);
		exit(0);
	}
	cnt = atoi(response);
	if (cnt != 0) {
		printf("Executing the sequences\n");
		response[retcnt] = VI_NULL;
		printf("SYSTEM:ERROR? query returned %d bytes; %s\n", retcnt, response);
	}

	return status;
}
```

2.2.9 Analyze the Execution Results

The following code analyzes the execution results.

The non-real time compare table is verified by performing a CRC on the response data. The returned CRC can be compared against a known good CRC value.

The real time compare table is verified by checking the error counter. If the error count is not zero then an error has been recorded.

```
/* Function to post process the sequence results */
ViStatus sr192_post(void)
{
	ViUInt32 retcnt;
	ViInt32 cnt;

	/* Get CRC of table SR_NRTC group DATA_BUS */
	if ((status = viWrite (SR, (ViBuf)"CALCULATE:CRC? SR_NRTC,DATA_BUS,0", 33, &retcnt))) {
		printf("Error sending CRC command to SR192 at LA2; status = %x\n", status);
		exit(0);
	}

	/* read response */
	if ((status = viRead (SR, (ViPBuf)response, 100, &retcnt))) {
		printf("Error reading the error query results from the SR192 at LA2; status = %x\n", status);
		exit(0);
	}
	response[retcnt] = VI_NULL;
	printf("CALCULATE:CRC? query returned %d bytes; %x\n", retcnt, strtoul (response, VI_NULL, 10));

	/* Get error count of table SR_NRTC group DATA_BUS */
	if ((status = viWrite (SR, (ViBuf)"CALCULATE:EMEMORY:COUNT?", 24, &retcnt))) {
		printf("Error sending error count query to SR192 at LA2; status = %x\n", status);
		exit(0);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
        }

        /* read response */
        if ((status = viRead (SR, (ViPBuf)response, 100, &retcnt))) {
                printf("Error reading the error count query results from the SR192 at LA2; status = %x\n", status);
                exit(0);
        }
        response[retcnt] = VI_NULL;
        printf("Real time error results = %s error(s)\n", response);

        return status;
}
```

2.2.10 Close the VXI Session

The following code closes the communication channel to the SR192.

```
/* Function to close the SR192 and resource manager sessions. */
ViStatus sr192_close(void)
{
        /* Close the SR192 session */
        if ((status = viClose(SR))) {
                printf("Error closing the SR192 session at LA2; status = %x\n", status);
                exit(0);
        }

        /* Close the resource manager session */
        if ((status = viClose(RM))) {
                printf("Error closing the resource manager session; status = %x\n", status);
                exit(0);
        }
        return status;
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3 Command Reference

The following sections describe the SCPI commands for the SR192.

3.1 SCPI Fundamentals

SCPI formats (version 1994.0) are used in this instrument to provide a general purpose programming structure.

3.1.1 SCPI Command Structure

The basic structure of a SCPI command is:

<COMMAND_HEADER> <PARAMETER_LIST>

3.1.1.1 Command Header Definition

The <COMMAND_HEADER> consists of two distinct types, IEEE 488.2 common commands and Talon SCPI instrument commands.

3.1.1.1.1 IEEE 488.2 Common Command Header

A common command header is a command keyword preceded by an asterisk (*). The command keyword is a string of ASCII characters that represent a specific command.

Examples:

*RST
*IDN?

3.1.1.1.2 Talon SCPI Instrument Command Header

An instrument control header is a hierarchical command keyword list separated by colons (:). Figure 3-1 illustrates the SR192 command subsystems:

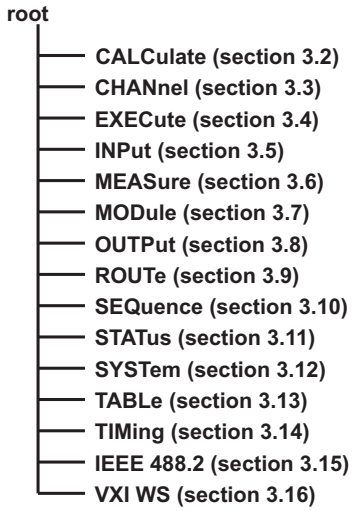


Figure 3-1 SCPI Header Hierarchy

Each instrument control command keyword is a string of ASCII characters that represent a specific command or subsystem.

The upper case letters of each keyword are the short form of the keyword, the upper and lower case letters are the long form of the keyword. Only the entire short or long form of each keyword is valid. Command keywords enclosed within square brackets [ ] are default and may be excluded from the header.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Examples:

OUTPut:TTLTrg1:STATe OFF
OUTP:TTLT1 OFF
TABLe:DELete:ALL
TABL:DEL:ALL

3.1.1.2 Parameter List Definition

The parameter list defines the settings for the specified command. Some commands may not require any parameters (i.e. TABLe:DELete:ALL) whereas other commands may have several parameters.

3.1.1.2.1 Parameter List Symbols

The following symbols are used in the parameter list definitions:

Comma ,	Parameter separator: If a command has more than one parameter then each parameter is separated by a comma.
Square Brackets [ ]	Optional parameter: The square brackets indicate an optional parameter entry.
Curly Brackets { }	Repeating parameter: The curly brackets represent one or more repeating parameter(s) separated by commas.
Parenthesis ( )	Parameter option: The parentheses will enclose a list of parameter choices separated by vertical lines.
Vertical Line |	See Parenthesis above.

3.1.1.2.2 Parameter List Types

The following list describes the parameter list types and their restrictions where applicable:

Name
User-defined alphanumeric character string used to define a timing set, table, group or sequence. The first character in the string must be an alpha character. Maximum length is ten characters. The underscore character is allowed.

Ascii
Pre-defined list of ASCII strings used to represent a valid setting. For example, in the “EXEC:MODE <mode>”, <mode> can be RESet, STOP, SINGle, LOOP or CONTinuous.

Arb Block
Represents Definite Length Arbitrary Block Data as described by the IEEE 488.2 standard. The <block> parameter is used to transfer 8-bit binary data and has the following format:

#<n><size><byte data>

Where:

<n>	Single ASCII number from 1 to 9 and defines how many digits are in <size>.
<size>	The number of bytes that are in <byte data> .
<byte data>	Stream of eight bit data.

The following example defines a ten byte transfer:

#210$byte_1 byte_2 byte_3$ . . . $byte_9 byte_{10}$

Boolean
This parameter is used as a shorthand for ON | OFF | 0 | 1. ON | 1 specifies a true setting and OFF | 0 specifies a false setting.

Channel list
This parameter is used to associate a physical channel to a logical group. The format of the channel list is:

(@<channel_spec>)

Where:

<channel_spec>	Comma separated list of either single channels or a channel range. Single channels are identified as a single number from 1 to 192. A channel range is two single channels separated by a colon ':'.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Examples:

(@1,2,3)	Assigns channel 1, 2, & 3 as a single group
(@1:3)	Performs the same operation as above
(@1:8,17:24)	Creates a sixteen channel group.

Channel groups are programmed/queried with the lowest numbered channel assigned as the least significant bit regardless of order assigned in the channel list.

Numeric — Represents a formatted numeric entry. Numeric values can be entered as ASCII, hexadecimal (#H), octal (#Q) or binary (#B).

Examples:

ASCII	256, 69, 42, 2.54, -3.89, 10.35e5
Hexadecimal	#H100, #H45, #H2A
Octal	#Q400, #Q105, #Q52
Binary	#B100000000, #B1000101

Subsequence — An SR192 execution sequence is defined by one or more timing cycle, table groups. Each timing cycle, table group is called a “subsequence”. A specific subsequence is specified by one of following two methods:

<seq_name>,<offset>

Or

<seq_addr>

The <seq_name>,<offset> form identifies a subsequence by specifying an offset from a defined sequence. The <seq_name> is a “Ascii” type and has a valid range described above under “Ascii”. The <offset> parameter is a “Numeric” type and has a valid range from 1 to the size of the sequence, i.e., the number of timing cycle, table pairs.

The <seq_addr> form identifies a subsequence by specifying the physical address of the subsequence. The <seq_addr> is a “Numeric” type and has a valid range from 0 to 131071.

3.1.2 Events and Queries

All commands, unless otherwise noted, have an event form and a query form.

The event form of a command programs a setting or sets a mode, i.e., performs an event. For example the command “TABLE:DEFINE DATA,1024" defines a table in memory named “DATA” and allocates 1024 words of memory space set to zeros.

The query form returns the associated event’s current setting or value. A query is a command header with a question mark appended. For example the query “TABLE:DEFINE? DATA” returns the name,size and the offset address of the first word within the SR192's A32 memory of table named “DATA”.

All ASCII numeric results are returned in decimal.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.2 Calculate Subsystem

The CALCulate subsystem is used to perform post acquisition table analysis.

KEYWORD	PARAMETER LIST
CALCulate	
:CRC?	<table_name>,<group_name>,<seed>[,<mask>]
:EMEMory	
:ADDRess?	<numeric>
:COUNt?	
:PCRC?	<table_name>

3.2.1 CALCulate:CRC?

The SR192 will execute a CRC polynomial on the specified table and channel group.

SYNTAX

CALCulate:CRC? <table_name>,<group_name>,<seed>[,<mask>]

PARAMETER LIST

Parameter	Type	Values	Description	Default
<table_name>	Name	See section 3.1.1.2.2	The table name identifies the memory locations to perform the CRC on.	NA
<group_name>	Name	See section 3.1.1.2.2	The group name identifies the channels to perform the CRC on.	
<seed>	Numeric	0-4,294,967,295 (0_h-$FFFFFFFF_h$)	The seed is used to accumulate multiple CRC results into a single CRC.	
[<mask>]	Numeric	0-4,294,967,295 (0_h-$FFFFFFFF_h$)	The mask is used to disable specific channels in the channel group from the CRC calculation.	$FFFFFFFF_h$

RETURNED PARAMETER SYNTAX

<crc>

Returned Parameter	Type	Value	Description
<crc>	Numeric	0-4,294,967,295 (0_h-$FFFFFFFF_h$)	Calculated CRC for the response data specified by <table_name> and <group_name>.

COMMENTS

1. The CRC is generated from the “RESPONSE” data from five memory I/O modules and the “RECORD” data from three memory modules.
2. A zero in each bit position of the <mask> disables the corresponding bit of the channel group. For Example:
   Channel group = 17:32;
   <mask> = #hFF0F.
   Causes a CRC to be generated on channels 17 through 20 and 25 through 32, channels 21 through 24 are masked off.
3. A “Parameter error” will be generated if <table_name> or <group_name> are not defined.
4. A “Settings conflict” will be generated if <group_name> has more than 32 channels.
5. Response data is only available when the timing module is idle or reset.

3.2.2 CALCulate:EMEMory

The EMEMory subtree allows the operator to query results of an execution sequence from the error memory logic. The error memory logic consists of an error counter and an error address ram. When a transfer from the SR192 does not match the expected results the address of the transfer is written into the RAM and the error counter is incremented. Thus a sequence execution with no errors will return a count of zero.

The error memory counter and address ram is synchronized with the TSA timing module only. When TSB is not linked with TSA no errors will be recorded in the error memory for TSB transfers.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.2.2.1 CALCulate:EMEMory:ADDRess?

The ADDRess command returns an value from the error memory address ram located on the SR192 motherboard. The value returned represents a physical address location of a transfer that caused an error during sequence execution.

SYNTAX

CALCulate:EMEMory:ADDRess? <error_number>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<error_number>	Numeric	0-262,144 (1_h-40000_h)	This parameter selects which error to query the error address. This number will be between "1" and the returned error count. A "0" returns the present memory address.	NA

RETURNED PARAMETER SYNTAX

<EMA>

Returned Parameter	Type	Value	Description
<EMA>	Numeric	0-4,294,967,295 (0_h-$FFFFFFFF_h$)	Error Memory Address. Physical address of the error number specified by <error_number>.

COMMENTS

1. The returned <EMA> can be used to query the error memory to determine which channel(s) caused the real-time error. Figure 3-2 shows the relationship between the <EMA> and table memory.

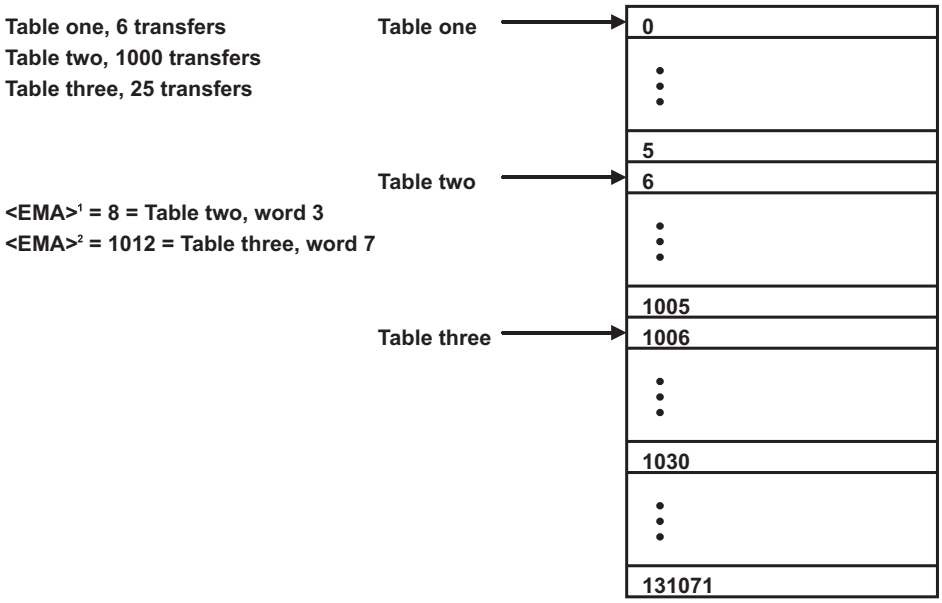


Figure 3-2 Error Memory Address to Table Mapping

2. The starting address of each table is returned in both the "TABLE:DIRECTORY?" and "TABLE:DEFINE?" commands.

3.2.2.2 CALCulate:EMEMory:COUNt?

This command returns the number of table transfers that did not match the expected response.

SYNTAX

CALCulate:EMEMory:COUNt?

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

RETURNED PARAMETER SYNTAX

<count>

Returned Parameter	Type	Value	Description
<count>	Numeric	0-262,143 (0_h-$3FFFF_h$)	The number of transfers from the most recent execution sequence that generated a real time error.

COMMENTS

1. The error count is cleared prior to sequence execution and when the timing module is reset.

3.2.3 CALCulate:PCRC?

The SR192 will execute a CRC polynomial on the specified tables probe memory.

SYNTAX

CALCulate:PCRC? <table_name>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<table_name>	Name	See section 3.1.1.2.2	The table name identifies the memory locations to perform the probe CRC on.	NA

RETURNED PARAMETER SYNTAX

<crc>

Returned Parameter	Type	Value	Description
<crc>	Numeric	0-4,294,967,295 (0_h-$FFFFFFFF_h$)	Calculated CRC for the probe data specified by <table_name>.

COMMENTS

1. A “Parameter error” will be generated if <table_name> is not defined.
2. A “Execution error” will be generated if the SR211 probe pod is not installed.
3. Probe data is only available when the TSA timing module is idle or reset.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.3 Channel Subsystem

The CHANnel subsystem is used to program/query the channel group mode of the SR192's algorithmic I/O modules.

KEYWORD	PARAMETER LIST
CHANnel	
:MODE	<group_name>,<mode>
:REFerence	<group_name>,<reference>

3.3.1 CHANnel:MODE

The MODE command is used to select the channel mode for Talon's algorithmic I/O modules.

SYNTAX

CHANnel:MODE <group_name>,<mode>
CHANnel:MODE? <group_name>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<group_name>	Name	See section 3.1.1.2.2	The group name identifies the channels to program or query.	NA
<mode>	Ascii	MULTiplex	Input/Output mode that allows 50MHz data rates.	HOLD
		SERial	Input/Output mode that allows serial stimulus/response.	
		HOLD	Output mode for latching output data.	
		RTO	Output Return To One.	
		RTZ	Output Return To Zero.	
		RTC	Output Return To Compliment.	
		INCRement1	Output increment by one.	
		INCRement2	Output increment by two.	
		INCRement4	Output increment by two.	
		INCRement8	Output increment by eight.	

RETURNED PARAMETER SYNTAX

<mode>

Returned Parameter	Type	Value	Description
<mode>	Ascii	MULTiplex	Input/Output mode that allows 50MHz data rates.
		SERial	Input/Output mode that allows serial stimulus/response.
		HOLD	Output mode for latching output data.
		RTO	Output Return To One.
		RTZ	Output Return To Zero.
		RTC	Output Return To Compliment.
		INCRement1	Output increment by one.
		INCRement2	Output increment by two.
		INCRement4	Output increment by four.
		INCRement8	Output increment by eight.

COMMENTS

1. The SERial and INCRement modes require that the specified channel group be a multiple of eight and located in adjacent I/O module slots in order for shifting and carry signals to be routed properly.
2. Channel modes are assigned in groups of eight channels.
3. HOLD is assigned to new channel groups.

3.3.2 CHANnel:REFerence

This command is used to select the voltage reference for Talon's dual reference modules.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SYNTAX

CHANnel:MODE <group_name>,<reference>
CHANnel:REFerence? <group_name>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<group_name>	Name	See section 3.1.1.2.2	The group name identifies the channels to program or query.	NA
<reference>	Ascii	REFA	Selects reference A (VGRP1 signals).	REFA
		REFB	Selects reference B (VGRP2/VGRP3 signals).	

RETURNED PARAMETER SYNTAX

<reference>

Returned Parameter	Type	Value	Description
<reference>	Ascii	REFA	Reference A (VGRP1 signals) selected.
		REFB	Reference B (VGRP2/VGRP3 signals) selected.

COMMENTS

1. REFA is assigned to new channel groups
2. Reference selection is assigned in groups of eight channels.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.4 Execute Subsystem

The EXECute subsystem allows the operator to program and execute SR192 dynamic and static I/O.

KEYWORD	PARAMETER LIST
EXECute	
:FIELd	<group_name>,{<numeric>}
:MODE	<mode>
:SEQuence	[<seq_name>] [<address>,<cycle_name>] [{<cycle_name>,<table_name>}]
[:TIMing]	[<cycle_name>,<table_name>] [<cycle_name>,<address>,<size>]

3.4.1 EXECute:FIELd

This command allows the operator to program the output data for the static IO modules.

SYNTAX

EXECute:FIELd <group_name>,{<numeric>}
EXECute:FIELd? <group_name>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<group_name>	Name	See section 3.1.1.2.2	The group name identifies the channels to program or query.	NA
{<numeric>}	Numeric	0-4,294,967,295 (0_h-$FFFFFFFF_h$)	One or more data values separated by commas that represent the output data.	

RETURNED PARAMETER SYNTAX

{<numeric>}

Returned Parameter	Type	Value	Description
{<numeric>}	Numeric	0-4,294,967,295 (0_h-$FFFFFFFF_h$)	One or more data values separated by commas that represent the input data.

COMMENTS

1. Each <numeric> value represents up to 32 static channels, <numeric> LSB maps to the channel group LSB.
2. If the channel group is greater than 32 channels additional <numeric> values are required, i.e., (<channels 64 to 33>,<channels 32 to 1>).
3. A “Parameter error” will be generated if the <group_name> is not a static I/O type.

3.4.2 EXECute:MODE

The MODE command defines the run and stop modes of the SR192 timing modules.

SYNTAX

EXECute:MODE <mode>[,<numeric>]

PARAMETER LIST

Parameter	Type	Values	Description	Default
<mode>	Ascii	RESet	Resets the selected timing module.	RESet
		STOP	Sends a stop signal to the selected timing module.	
		SINGle	Sets the run mode to single.	
		LOOP	Sets the run mode to loop <numeric> times.	
		CONTinuous	Sets the run mode to continuous.	
[,<numeric>]	Numeric	SR100 0-65535 1_h-$FFFF_h$	Loop count, 0 = continuous.	NA
		SR101 0-32768 1_h-$7FFF_h$		

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

COMMENTS

1. The RESet mode stops the selected timing module and disables the idle sequence so that the timing memory can be programmed.
2. The STOP mode will stop sequence execution at the end of the next I/O transfer.

3.4.3 EXECute:SEQuence

This command initializes and runs a dynamic I/O sequence.

SYNTAX

EXECute:SEQuence [<seq_name>]
EXECute:SEQuence [<address>,<cycle_name>]
EXECute:SEQuence [{<cycle_name>,<table_name>}]

PARAMETER LIST

Parameter	Type	Values	Description	Default
<seq_name>	Name	See section 3.1.1.2.2	The pre defined sequence definition to execute.	NA
<address>	Numeric	0-131071 0_h-1FFFF$_h$	Physical sequence address to execute.	
<cycle_name>	Name	See section 3.1.1.2.2	Identifies a timing cycle definition.	
<table_name>	Name	See section 3.1.1.2.2	Identifies I/O module stimulus/response memory locations.	

COMMENTS

1. If the parameter list is omitted then the previous sequence will be executed.
2. A “Settings conflict” error will be generated if a timing module is not selected.
3. Up to four pairs of <cycle_name>,<table_name> pairs can be programmed and executed.

3.4.4 EXECute[:TIMing]

This command initializes and runs a single timing/data sequence.

SYNTAX

EXECute[:TIMing] [<cycle_name>,<table_name>]
EXECute[:TIMing] [<cycle_name>,<address>,<size>]

PARAMETER LIST

Parameter	Type	Values	Description	Default
<cycle_name>	Name	See section 3.1.1.2.2	Identifies a timing cycle definition.	NA
<table_name>	Name	See section 3.1.1.2.2	Identifies I/O module stimulus/response memory locations.	
<address>	Numeric	0-131071 0_h-1FFFF$_h$	Physical table address to execute.	
<size>	Numeric	1-131072 0_h-1FFFF$_h$	The number of stimulus/response transfers to execute.	

COMMENTS

1. If the parameter list is omitted then the previous timing definition will be executed.
2. A “Settings conflict” error will be generated if a timing module is not selected.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.5 Input Subsystem

The INPut subsystem controls the input ports and resources of the SR192.

KEYWORD	PARAMETER LIST
INPut	
:ADELay	<boolean>
:MSTRobe	<group_name>,<multiplex_strobe>
:PROBe	
:MODE	<mode>
:ORDelay	<ord_state>
:REFerence	
:HIGH	
[:DATA]	<vih_value>
[:LOGic]	<logic>
:LOW	
[:DATA]	<vil_value>
:STATus?	
:STRObe	
:DELay	<delay>
[:SOURce]	<probe_strobe>
:SWITch	<switch>,<function>
:VERSion?	
:REFerence	
:AUTO	<boolean>
:HIGH	
[:DATA]	<vih_value>
:INCRement	<increment_count>
:LOW	
[:DATA]	<vil_value>
:INCRement	<increment_count>
:SELect	[<voltasge_group>]
:UPDate	
:STRobe	
:DELay	<group_name>,<delay>
[:SOURce]	<group_name>[,<input_strobe>]

3.5.1 INPut:ADELay

This command allows response data to be recorded during the next output stimulus word. This mode requires the ADEL_CLK timing set signal to be programmed to register the response memory address.

SYNTAX

INPut:ADELay <adelay_state>
INPut:ADELay?

PARAMETER LIST

Parameter	Type	Values	Description	Default
<adelay_state>	Boolean	OFF | 0	Sets address delay mode off.	OFF
		ON | 1	Sets address delay mode on.	

RETURNED PARAMETER SYNTAX

<adelay_state>

Parameter	Type	Value	Description
<adelay_state>	Numeric	0	Selected timing modules address delay state is turned off.
		1	Selected timing modules address delay state is turned on.

COMMENTS

1. A “Settings conflict” error will be generated if a timing module is not selected.

3.5.2 INPut:MSTRobe

This command is used to specify the serial/multiplex data strobe for algorithmic I/O modules.

SYNTAX

INPut:MSTRobe <group_name>,<signal_name>

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

INPut:MSTRobe? <group_name>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<group_name>	Name	See section 3.1.1.2.2	The group name identifies the channels to program or query.	NA
<signal_name>	Ascii	TSSTrobe1	Selects timing module strobe 1.	TSSTrobe1
		TSSTrobe2	Selects timing module strobe 2.	
		FCNTl1	Selects front panel “FCNTL1” signal.	
		FCNTl2	Selects front panel “FCNTL2” signal.	

RETURNED PARAMETER SYNTAX

<signal_name>

Returned Parameter	Type	Value	Description
<signal_name>	Ascii	TSST1	Timing module TSSTROBE1 selected.
		TSST2	Timing module TSSTROBE2 selected.
		FCNT1	Front panel FCNTL1 selected.
		FCNT2	Front panel FCNTL2 selected.

COMMENTS

1. For channel groups assigned as MULTiplex, the falling edge of MSTROBE records the lower nibble of each eight bit group.
2. For channel groups assigned as SERial, the falling edge of MSTROBE records the most significant bit of the group.
3. SR114 and SR115 I/O modules cannot select the TSSTROBE2 or FCNTL2.

3.5.3 INPut:PROBe

The PROBe subsystem allows the user to program and query the SR211 probe functions and results.

3.5.3.1 INPut:PROBe:MODE

The MODE command defines the run mode of the SR211 probe pod.

SYNTAX

INPut:PROBE:MODE <mode>
INPut:PROBe:MODE?

PARAMETER LIST

Parameter	Type	Values	Description	Default
<mode>	Ascii	AUTO	Enables the node detect mode enabled, i.e., LED indicators on the SR211 pod are active.	MANual
		ACQuire	Disables the node detect mode, OPEN LED on probe pod flashes.	
		PULSe	Places the SR211 probe into a standby status, i.e., node and pulse detect disabled and LEDs off.	
		MANual	Set the SR211 manual mode.	
		RESet	Resets the references to zero and places the SR211 into MANual mode.	

RETURNED PARAMETER SYNTAX

<mode>

Parameter	Type	Value	Description
<mode>	Ascii	AUTO	AUTO mode is set.
		ACQ	ACQuire mode os set.
		PUL	PULse mode is set.
		MAN	MANual mode is set.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

COMMENTS

1. A “Execution error” will be generated if the SR211 pod is not installed.

3.5.3.2 INPut:PROBe:ORDelay

This command allows SR211 output register delay mode to be enabled or disabled.

SYNTAX

INPut:PROBe:ORDelay <ordelay_state>
INPut:PROBe:ORDelay?

PARAMETER LIST

Parameter	Type	Values	Description	Default
<ordelay_state>	Boolean	OFF | 0	Sets output register delay mode off.	OFF
		ON | 1	Sets output register delay mode on.	

RETURNED PARAMETER SYNTAX

<ordelay_state>

Parameter	Type	Value	Description
<ordelay_state>	Numeric	0	SR211 output register delay state is turned off.
		1	SR211 output register delay state is turned on.

COMMENTS

1. A “Execution error” error will be generated if a timing module is not selected.
2. The output register delay is used to align the probe memory with I/O module memory.

3.5.3.3 INPut:PROBe:REFerence

This subsystem allows the user to program or query the SR211 probe reference levels.

3.5.3.3.1 INPut:PROBe:REFerence:HIGH[:DATA]

This command sets the high reference levels for the SR211 pods GOOD0/GOOD1 receiver.

SYNTAX

INPut:PROBE:REFerence:HIGH[:DATA] <vih>
INPut:PROBe:REFerence:HIGH[:DATA]?

PARAMETER LIST

Parameter	Type	Values	Description	Default
<vih>	Numeric	-10.0 to +10.0	Programs the GOOD0/GOOD1 high level.	0.0

RETURNED PARAMETER SYNTAX

<vih>

Parameter	Type	Value	Description
<vih>	Numeric	-10.0 to +10.0	SR211 high reference setting.

COMMENTS

1. A “Execution error” will be generated if the SR211 pod is not installed.

3.5.3.3.2 INPut:PROBe:REFerence[:LOGic]

This command sets the high and low reference levels for the SR211 pods GOOD0/GOOD1 receiver.

SYNTAX

INPut:PROBE:REFerence[:LOGic] <logic>

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

PARAMETER LIST

Parameter	Type	Values	Description	Default
<logic>	Ascii	TTL	Programs the GOOD0/GOOD1 high and low levels to TTL. VIH = +2.4V, VIL = +0.8V.	NA
		ECL	Programs the GOOD0/GOOD1 high and low levels to ECL. VIH = -0.87V, VIL = -1.48V.	
		LV	Programs the GOOD0/GOOD1 high and low levels to LV. VIH = +2.0V, VIL = +0.8V.	

COMMENTS

1. A “Execution error” will be generated if the SR211 pod is not installed.

3.5.3.3.3 INPut:PROBe:REFerence:LOW[:DATA]

This command sets the low reference levels for the SR211 pods GOOD0/GOOD1 receiver.

SYNTAX

INPut:PROBE:REFerence:LOW[:DATA] <vil>
INPut:PROBe:REFerence:LOW[:DATA]?

PARAMETER LIST

Parameter	Type	Values	Description	Default
<vih>	Numeric	-10.0 to +10.0	Programs the GOOD0/GOOD1 high level.	0.0

RETURNED PARAMETER SYNTAX

<vil>

Parameter	Type	Value	Description
<vil>	Numeric	-10.0 to +10.0	SR211 low reference setting.

COMMENTS

1. A “Execution error” will be generated if the SR211 pod is not installed.

3.5.3.4 INPut:PROBe:STATus?

This command returns the SR211 probe pod status.

SYNTAX

INPut:PROBe:STATus?

RETURNED PARAMETER SYNTAX

<status>

Parameter	Type	Value	Description
<status>	Numeric	32767 to -32768 (0_h-$FFFF_h$)	SR211 probe pod status code

COMMENTS

1. The upper byte of the status code is the module ID, refer to the specific modules reference manual, appendix B for the unique ID code. The table below lists the bit definition of the lower byte of the status code for the different module types.

Status Code Bit Definition															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NU	LS2	LS1	LPRB	NU	S2	S1	PRB	NU		NODE	P1	P0	HI-Z	G1	G0

Bit Description

G0	GOOD0 signal; 0 = false, 1 = true.
G1	GOOD1 signal; 0 = false, 1 = true.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

HI-Z	Tristate signal; 0 = false, 1 = true.
P0	Pulse low signal; 0 = false, 1 = true.
P1	Pulse high signal; 0 = false, 1 = true.
NODE	Node detect signal; 0 = false, 1 = true.
PRB	Probe switch depressed; 0 = false, 1 = true.
S1	S1 switch depressed; 0 = false, 1 = true.
S2	S2 switch depressed; 0 = false, 1 = true.
LPRB	Probe switch transition flag; 0 = false, 1 = true.
LS1	S1 switch transition flag; 0 = false, 1 = true.
LS2	S2 switch transition flag; 0 = false, 1 = true.
NU	Not used bits, always 0.

2. The switch transition (LBRP, LS1, LS2) and pulse (P0, P1) flags are reset by the query and “INPut:PROBe:MODE RESet” command.
3. A “Execution error” will be generated if the SR211 pod is not installed.

3.5.3.5 INPut:PROBe:STRobe

This subsystem allows the user to program and query the SR211 input probe signal.

3.5.3.5.1 INPut:PROBe:STRobe:DELay

This command is used to specify the serial/multiplex data strobe for algorithmic I/O modules.

SYNTAX

INPut:PROBe:STRobe:DELay <delay>
INPut:PROBe:STRobe[:DELay]?

PARAMETER LIST

Parameter	Type	Values	Description	Default
<delay>	Numeric	0 to 7	Selects strobe delay from 0 to 28ns in increments of 4ns.	0

RETURNED PARAMETER SYNTAX

<delay>

Returned Parameter	Type	Value	Description
<delay>	Numeric	0 to 7	Selected probe strobe delay.

COMMENTS

1. A “Execution error” will be generated if the SR211 pod is not installed.

3.5.3.5.2 INPut:PROBe:STRobe[:SOURce]

This command is used to specify the SR211 probe strobe signal.

SYNTAX

INPut:PROBe:STRobe[:SOURce] <strobe>
INPut:PROBe:STRobe[:SOURce]?

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

PARAMETER LIST

Parameter	Type	Values	Description	Default
<strobe>	Ascii	TSSTrobe1	Selects timing module “A” strobe 1.	TSOUT5
		TSSTrobe2	Selects timing module “A” strobe 2.	
		FCNTl1	Selects front panel “FCNTL1” signal.	
		FCNTl2	Selects front panel “FCNTL2” signal.	
		TSOut5	Selects timing module “A” TSOUT5.	
		PRBDat	Selects front panel PRBDAT.	

RETURNED PARAMETER SYNTAX

<strobe>

Returned Parameter	Type	Value	Description
<strobe>	Ascii	TSST1	Timing module “A” TSSTROBE1 selected.
		TSST2	Timing module “A” TSSTROBE2 selected.
		FCNT1	Front panel FCNTL1 selected.
		FCNT2	Front panel FCNTL2 selected.
		TSO5	Timing module “A” TSOUT5 selected.
		PRBD	Front panel PRBDAT selected.

COMMENTS

1. A “Execution error” will be generated if the SR211 pod is not installed.

3.5.3.6 INPut:PROBe:SWITch

This command is used to program the SR211 switch function.

SYNTAX

INPut:PROBe:SWITch <switch>,<function>
INPut:PROBe:SWITch? <switch>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<switch>	Ascii	PROBE	The selected switch to program or query.	All switches set to NONE.
		S1		
		S2		
<function>	Ascii	NONE	Selected switch set has no function.	
		AUTO	Set the SR211 to AUTO mode.	
		ACQuire	Set the SR211 to ACQuire mode.	
		PULSe	Set the SR211 to PULSe mode.	

RETURNED PARAMETER SYNTAX

<function>

Returned Parameter	Type	Value	Description
<function>	Ascii	NONE	Selected switch has no function assigned.
		AUTO	Selected switch assigned AUTO mode.
		ACQ	Selected switch assigned ACQuire mode.
		PULS	Selected switch assigned PULSe mode.

COMMENTS

1. A “Execution error” will be generated if the SR211 pod is not installed.

3.5.3.7 INPut:PROBe:VERsion?

This command returns the SR211 probe pods installed firmware version.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SYNTAX

INPut:PROBe:VERSion?

RETURNED PARAMETER SYNTAX

<version>

Returned Parameter	Type	Value	Description
<version>	Numeric	0.00 to 9.99	SR211 firmware version.

COMMENTS

1. A “Execution error” will be generated if the SR211 pod is not installed.

3.5.4 INPut:REFerence

The REFerence subsystem provides the commands for programming and querying the reference voltages to be used by the DAC module. These references are used by the variable voltage I/O modules and TSIO.

3.5.4.1 INPut:REFerence:AUTO

This command allows the user to program the update mode of the SR192 reference signals, auto or manual.

SYNTAX

INPut:REFerence:AUTO <auto_state>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<auto_state>	Boolean	OFF | 0	Sets the reference update mode to manual.	OFF
		ON | 1	Sets the reference update mode to auto.	

COMMENTS

1. The manual reference mode requires the “INPut:REFerence:UPDate” command to trigger the new references.

3.5.4.2 INPut:REFerence:HIGH

The HIGH subtree allows the operator to program the input high voltages for the three SR192 voltage groups.

3.5.4.2.1 INPut:REFerence:HIGH[:DATA]

This commands allows the operator to program the input high threshold for the selected group.

SYNTAX

INPut:REFerence:HIGH[:DATA] <vih>
INPut:REFerence:HIGH[:DATA]?

PARAMETER LIST

Parameter	Type	Values	Description	Default
<vih>	Numeric	-5.0 to +15.0	Programs the input high level for the variable voltage modules.	0.0

RETURNED PARAMETER SYNTAX

<vih>

Parameter	Type	Value	Description
<vih>	Numeric	-5.0 to +15.0	Reference high reference setting.

COMMENTS

1. The voltage group is selected by the “INPut:REFerence:SELect” command.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.5.4.2.2 INPut:REFerence:HIGH:INCRement

This command allows the operator to increment the current value of the input high level.

SYNTAX

INPut:REFerence:HIGH:INCRement <count>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<count>	Numeric	-1500 to 1500	The number of counts to increase/decrease the input high level of the selected group.	0.0

COMMENTS

1. The voltage group is selected by the “INPut:REFerence:SELect” command.
2. Each <count> is approximately 20mV for the SR110 DAC module.
3. Each <count> is 10mV for the SR210 DAC/MFC module.

3.5.4.3 INPut:REFerence:LOW

The LOW subtree allows the operator to program the input low voltages for the three SR192 voltage groups.

3.5.4.3.1 INPut:REFerence:LOW[:DATA]

This commands allows the operator to program the input low threshold for the selected group.

SYNTAX

INPut:REFerence:LOW[:DATA] <vil>
INPut:REFerence:LOW[:DATA]?

PARAMETER LIST

Parameter	Type	Values	Description	Default
<vil>	Numeric	-15.0 to +5.0	Programs the input low level for the variable voltage modules.	0.0

RETURNED PARAMETER SYNTAX

<vil>

Parameter	Type	Value	Description
<vil>	Numeric	-15.0 to +5.0	Reference low reference setting.

COMMENTS

1. The voltage group is selected by the “INPut:REFerence:SELect” command.

3.5.4.3.2 INPut:REFerence:LOW:INCRement

This command allows the operator to increment the current value of the input low level.

SYNTAX

INPut:REFerence:LOW:INCRement <count>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<count>	Numeric	-1500 to 1500	The number of counts to increase/decrease the input low level of the selected group.	0.0

COMMENTS

1. The voltage group is selected by the “INPut:REFerence:SELect” command.
2. Each <count> is approximately 20mV for the SR110 DAC module.
3. Each <count> is 10mV for the SR210 DAC/MFC module.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.5.4.4 INPut:REFerence:SELect

This command is used to select one of the three voltage groups for programming.

SYNTAX

INPut:REFerence:SELect <group>
INPut:REFerence:SELect?

PARAMETER LIST

Parameter	Type	Values	Description	Default
<group>	Ascii	VGRP1	Program voltage group one input references.	VGRP1
		VGRP2	Program voltage group two input references.	
		VGRP3	Program voltage group three input references.	
		ALL	Program all input references.	

RETURNED PARAMETER SYNTAX

<group>

Returned Parameter	Type	Value	Description
<group>	Ascii	VGRP1	Voltage group one selected.
		VGRP2	Voltage group two selected.
		VGRP3	Voltage group three selected.
		ALL	All voltage groups selected.

COMMENTS

None

3.5.4.5 INPut:REFerence:UPDate

This command updates the variable voltage reference signals when the “INPut:REFerence:AUTO” setting has been turned off.

SYNTAX

INPut:REFerence:UPDate

COMMENTS

1. The voltage group is selected by the “INPut:REFerence:SELect” command.

3.5.5 INPut:STRobe

This subsystem allows the user to program or query the input strobe source and delay.

3.5.5.1 INPut:STRobe:DELay

This command is used to program the input strobe delay.

SYNTAX

INPut:STRobe:DELay <group_name>,<delay>
INPut:STRobe:DELay? <group_name>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<group_name>	Name	See section 3.1.1.2.2	The group name identifies the channels to program or query.	NA
<delay>	Numeric	0 to 3	Delay value; 0 = none, 1 = 5ns, 2 = 10ns, 3 = 15ns	0

RETURNED PARAMETER SYNTAX

<delay>

Returned Parameter	Type	Value	Description
<delay>	Numeric	0 to 3	Programmed delay value.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

COMMENTS

1. Not all SR192 I/O module support the input strobe delay.

3.5.5.2 INPut:STRobe[:SOURce]

This command is used to specify the input data strobe for the I/O modules.

SYNTAX

INPut:STRobe[:SOURce] <group_name>,<signal_name>
INPut:STRobe[:SOURce]? <group_name>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<group_name>	Name	See section 3.1.1.2.2	The group name identifies the channels to program or query.	NA
<signal_name>	Ascii	TSSTrobe1	Selects timing module strobe 1.	See comment 1 below.
		TSSTrobe2	Selects timing module strobe 2.	
		FCNTl1	Selects front panel “FCNTL1” signal.	
		FCNTl2	Selects front panel “FCNTL2” signal.	
		CSTRobe	Selects front panel “CSTROBE” signal.	
		TRANsparent	Selects unregistered input mode.	

RETURNED PARAMETER SYNTAX

<signal_name>

Returned Parameter	Type	Value	Description
<signal_name>	Ascii	TSST1	Timing module TSSTROBE1 selected.
		TSST2	Timing module TSSTROBE2 selected.
		FCNT1	Front panel FCNTL1 selected.
		FCNT2	Front panel FCNTL2 selected.
		CSTRobe	Front panel CSTROBE selected.
		TRANsparent	Transparent mode selected.

COMMENTS

1. Default input strobe setting is based on module type; static I/O modules select “TRANsparent”, dynamic non-algorithmic I/O modules select “TSSTROBE1”, dynamic algorithmic I/O modules select “TSSTROBE2”.
2. Not all SR192 I/O modules support every strobe source setting. Refer to the specific I/O module reference manual for valid choices. Selecting a non supported source will generate a “Parameter error”.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.6 Measure Subsystem

The MEASure subsystem is used to measure voltages through the SR211 probe pod.

COMMAND HEADER **PARAMETER LIST**

MEASure
:VOLTage?

3.6.1 MEASure:VOLTage?

Returns the voltage at the SR211 probe tip.

SYNTAX

MEASure:VOLTage?

RETURNED PARAMETER SYNTAX

<voltage>

Returned Parameter	Type	Value	Description
<voltage>	Numeric	-10.0 to +10.0	Probe tip voltage.

COMMENTS

1. A “Execution error” will be generated if the SR211 pod is not installed.
2. ”999.99” will be returned if the voltage is out of range.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.7 Module Subsystem

The MODule subsystem is used to select and configure individual modules installed in the SR192 motherboard.

COMMAND HEADER	PARAMETER LIST
MODule	
:LINK	<link_state>
[:SELect]	<module>
:STATus?	

3.7.1 MODule:LINK

Allows the operator to link timing module control signals together.

TSA can be linked with the external control signals from the front panel J7 connector.

TSB can be linked to TSA's selected control signals.

Linking the control signals allows multiple timing sets to be synchronized as well as multiple SR192's.

SYNTAX

MODule:LINK <link_state>
MODule:LINK?

PARAMETER LIST

Parameter	Type	Values	Description	Default
<link_state>	Boolean	OFF | 0	Sets control signals to internal.	OFF
		ON | 1	Sets control signals to external.	

RETURNED PARAMETER SYNTAX

<link_state>

Returned Parameter	Type	Value	Description
<link_state>	Numeric	0	Selected timing modules link state is turned off.
		1	Selected timing modules link state is turned on.

COMMENTS

1. The link command is only valid for TSA or TSB.
2. Turning on TSB's link state will cause TSA to be selected after the LINK command has executed.

3.7.2 MODule:SELect

Allows the operator to choose one of the installed modules in order to program or query settings.

SYNTAX

MODule:SELect <module>
MODule:SELect?

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

PARAMETER LIST

Parameter	Type	Values	Description	Default
<module>	Ascii	TSA	Selects module in TSA slot.	TSA
		TSB	Selects module in TSB slot.	
		DAC	Selects module in DAC/MFC slot.	
		DRA1	Selects module in DRA1 slot.	
		DRA2	Selects module in DRA2 slot.	
		DRA3	Selects module in DRA3 slot.	
		DRA4	Selects module in DRA4 slot.	
		DRA5	Selects module in DRA5 slot.	
		DRA6	Selects module in DRA6 slot.	
		DRB1	Selects module in DRB1 slot.	
		DRB2	Selects module in DRB2 slot.	
		DRB3	Selects module in DRB3 slot.	
		DRB4	Selects module in DRB4 slot.	
		DRB5	Selects module in DRB5 slot.	
		DRB6	Selects module in DRB6 slot.	

RETURNED PARAMETER SYNTAX

<module>

Returned Parameter	Type	Value	Description
<module>	Ascii string	TSA	Module in TSA slot selected.
		TSB	Module in TSB slot selected.
		DAC	Module in DAC/MFC slot selected.
		DRA1	Module in DRA1 slot selected.
		DRA2	Module in DRA2 slot selected.
		DRA3	Module in DRA3 slot selected.
		DRA4	Module in DRA4 slot selected.
		DRA5	Module in DRA5 slot selected.
		DRA6	Module in DRA6 slot selected.
		DRB1	Module in DRB1 slot selected.
		DRB2	Module in DRB2 slot selected.
		DRB3	Module in DRB3 slot selected.
		DRB4	Module in DRB4 slot selected.
		DRB5	Module in DRB5 slot selected.
		DRB6	Module in DRB6 slot selected.

COMMENTS

1. The module selection is retained for any follow-on programming commands until a new selection is made.
2. If a module is selected that is not installed a setting conflict error will occur and the previous module will be selected.

3.7.3 MODule:STATus?

Returns the selected modules status information.

SYNTAX

MODule:STATus?

RETURNED PARAMETER SYNTAX

<status>

Parameter	Type	Value	Description
<status>	Numeric	32767 to -32768 (0_h-$FFFF_h$)	Module status code

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

COMMENTS

The upper byte of the status code is the module ID, refer to the specific modules reference manual, appendix B for the unique ID code. The table below lists the bit definition of the lower byte of the status code for the different module types.

SR192 Module Type	Lower Byte Status Code Bit Definition							
	7	6	5	4	3	2	1	0
Timing	Command Done	NU	Error	Timeout	Wait	Run	Idle	Pass/Fail
Dynamic I/O	NU	PWR	NU					
Static I/O		PWR	TSSTROBE2	TSSTROBE1	FCNTL2	FCNTL1	CSTROBE	
DAC/MFC	NU					SR211 pass	SR211 Installed	

All Modules Bit Description

Pass/Fail	Module self test flag; 0 = failed self test, 1 = passed self test.

Timing Module Bit Description

Command Done	Asynchronous command done; 0 = done, 1 = not done.
Error	Error flag; 0 = no error, 1 = error.
Timeout	Input trigger/handshake timeout flag; 0 = no timeout, 1 = timeout.
Wait	Waiting for trigger/handshake flag; 0 = waiting, 1 = not waiting.
Run	Sequence running flag; 0 = running, 1 = idle/reset.
Idle	Idle enabled flag; 0 = idle enabled, 1 = idle not enabled.

Dynamic I/O Module Bit Description

PWR	Driver power enabled flag; 0 = power off, 1 = power on.

Static I/O Module Bit Description

CSTROBE	CSTROBE signal detect, reset when static I/O executed; 0 = not active, 1 = falling edge detected.
FCNTL1	FCNTL1 signal detect, reset when static I/O executed; 0 = not active, 1 = falling edge detected.
FCNTL2	FCNTL2 signal detect, reset when static I/O executed; 0 = not active, 1 = falling edge detected.
TSSTROBE1	TSSTROBE1 signal detect, reset when static I/O executed; 0 = not active, 1 = falling edge detected.
TSSTROBE2	TSSTROBE2 signal detect, reset when static I/O executed; 0 = not active, 1 = falling edge detected.

DAC/MFC Bit Description

SR211 Installed	SR211 pod detect flag; 0 = not installed, 1 = pod detected.
SR211 Self Test	SR211 self test flag; 0 = fail, 1 = pass.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.8 Output Subsystem

The OUTPut subsystem controls the output ports of the SR192.

KEYWORD	PARAMETER LIST
OUTPut	
:CHANnel	
:AUTO	<boolean>
[:STATe]	<boolean>
:CLK<a>	
:DELay	<delay>
:REFerence	
:HIGH	
[:DATA]	<clk_voh>
:LOW	
[:DATA]	<clk_vol>
[:STATe]	<clk_state>
:ENABle	
:DELay	<group_name>,<delay>
[:SOURce]	<group_name>,<output_enable>
:MASTer	<boolean>
:PGMClk<n>	
[:DATA]	<frequency>
:REFerence	<source>
:RESet	
:REFerence	
:AUTO	<boolean>
:HIGH	
[:DATA]	<voh>
:INCRement	<count>
:LOW	
[:DATA]	<vol>
:INCRement	<count>
:ISR	
[:DATA]	<isr>
:INCRement	<increment_count>
:SELect	[<voltage_group>]
:UPDate	
:REGister	
:SOURce	<group_name>,<output_strobe>
[:STATe]	<group_name>,<boolean>
:RESet	<mode>[,<duration>]
:SYNC	
[:STATe]	<boolean>
:POSition	<table_name>,<word_number>
:TIMing	
[:STATe]	<boolean>
:TTLTrg<n>	
[:STATe]	<boolean>

3.8.1 OUTPut:CHANnel

The CHANnel subsystem allows the user the ability to control the enable state of the I/O drivers. The user can select when the output drivers are enabled. The drivers can be enabled always, when the idle state is active, or when a sequence is running.

3.8.1.1 OUTPut:CHANnel:AUTO

The SR192 I/O module drivers require three signals in order to be enabled; PON signal, group enable signal and tristate memory signal. The :AUTO command in conjunction with the :STATe command controls when the PON signal is set true.

SYNTAX

OUTPut:CHANnel:AUTO <auto_state>
OUTPut:CHANnel:AUTO?

PARAMETER LIST

Parameter	Type	Values	Description	PON/*RST Default
<auto_state>	Boolean	OFF | 0	Sets auto state to OFF.	OFF
		ON | 1	Sets auto state to ON.	

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

RETURNED PARAMETER SYNTAX

<auto_state>

Returned Parameter	Type	Value	Description
<auto_state>	Numeric	0	Channel auto turned off.
		1	Channel auto turned on.

COMMENTS

1. If the OUTPut:CHANnel:STATe state is OFF then:

OUTPut:CHANnel:AUTO	PON Description
OFF	PON signal false always.
ON	PON signal true when sequence running.

2. If the OUTPut:CHANnel:STATe state is ON then:

OUTPut:CHANnel:AUTO	PON Description
OFF	PON signal true always.
ON	PON signal true when sequence or idle running.

3.8.1.2 OUTPut:CHANnel[:STATe]

The SR192 I/O module output drivers require three signals in order to be enabled; PON signal, group enable signal and tristate memory signal. The STATe command in conjunction with the :AUTO command controls when the PON signal is set true.

SYNTAX

OUTPut:CHANnel[:STATe] <chan_state>
OUTPut:CHANnel:STATe?

PARAMETER LIST

Parameter	Type	Values	Description	Default
<chan_state>	Boolean	OFF | 0	Sets channel state to OFF.	OFF
		ON | 1	Sets channel state to ON.	

RETURNED PARAMETER SYNTAX

<chan_state>

Parameter	Type	Value	Description
<chan_state>	Numeric	0	Channel state turned off.
		1	Channel state turned on.

COMMENTS

1. If the OUTPut:CHANnel:AUTO state is OFF then:

OUTPut:CHANnel:STATe	PON Description
OFF	PON signal false always.
ON	PON signal true always.

2. If the OUTPut:CHANnel:AUTO state is ON then:

OUTPut:CHANnel:STATe	PON Description
OFF	PON signal true when sequence running.
ON	PON signal true when sequence or idle running.

3.8.2 OUTPut:CLK<a>

The CLK<a> subsystem allows the user to program the front panel SMA clock signals.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SYNTAX

CLK<a>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<a>	Ascii	A	Selects CLKA to program/query	NA
		B	Selects CLKB to program/query	
		C	Selects CLKC to program/query	

3.8.2.1 OUTPut:CLK<a>:DELay

This command allows the user to program the delay for CLKA and CLKB.

SYNTAX

OUTPut:CLKA:DELay <delay>
OUTPut:CLKB:DELay <delay>
OUTPut:CLKA:DELay?
OUTPut:CLKB:DELay?

PARAMETER LIST

Parameter	Type	Values	Description	Default
<delay>	Numeric	0 to 7	Delay setting; 1 = 5ns, 2 = 10ns, .. ,7 = 35ns	0

RETURNED PARAMETER SYNTAX

<delay>

Returned Parameter	Type	Value	Description
<delay>	Numeric	0 to 7	Programmed delay value.

COMMENTS

None

3.8.2.2 OUTPut:CLK<a>:REFerence:HIGH[:DATA]

This command allows the user to program the output high level for CLKC.

SYNTAX

OUTPut:CLKC:REFerence:HIGH[:DATA] <voh>
OUTPut:CLKC:REFerence:HIGH[:DATA]?

PARAMETER LIST

Parameter	Type	Values	Description	Default
<voh>	Numeric	-8.0 to 8.0	CLKC output high level.	0.0

RETURNED PARAMETER SYNTAX

<voh>

Returned Parameter	Type	Value	Description
<voh>	Numeric	-8.0 to 8.0	CLKC output high level.

COMMENTS

None

3.8.2.3 OUTPut:CLK<a>:REFerence:LOW[:DATA]

This command allows the user to program the output low level for CLKC.

SYNTAX

OUTPut:CLKC:REFerence:LOW[:DATA] <vol>

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

OUTPut:CLKC:REFerence:LOW[:DATA]?

PARAMETER LIST

Parameter	Type	Values	Description	Default
<vol>	Numeric	-8.0 to 8.0	CLKC output low level.	0.0

RETURNED PARAMETER SYNTAX

<vol>

Returned Parameter	Type	Value	Description
<vol>	Numeric	-8.0 to 8.0	CLKC output low level.

COMMENTS

None

3.8.2.4 OUTPut:CLK<a>[:STATe]

This command is used to enable the front panel clock drivers.

SYNTAX

OUTPut:CLKA[:STATe] <clk_state>
OUTPut:CLKB[:STATe] <clk_state>
OUTPut:CLKC[:STATe] <clk_state>
OUTPut:CLKA[:STATe]?
OUTPut:CLKB[:STATe]?
OUTPut:CLKC[:STATe]?

PARAMETER LIST

Parameter	Type	Values	Description	Default
<clk_state>	Boolean	OFF | 0	Sets clock driver state to OFF.	OFF
		ON | 1	Sets clock driver state to ON.	

RETURNED PARAMETER SYNTAX

<clk_state>

Returned Parameter	Type	Value	Description
<clk_state>	Numeric	0	Clock state auto turned off.
		1	Clock state auto turned on.

COMMENTS

None

3.8.3 OUTPut:ENABle

This subsystem allows the user to program or query the output group enable source and delay.

3.8.3.1 OUTPut:ENABle:DELay

This command is used to program the output group enable delay.

SYNTAX

OUTPut:ENABle:DELay <group_name>,<delay>
OUTPut:ENABle:DELay? <group_name>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<group_name>	Name	See section 3.1.1.2.2	The group name identifies the channels to program or query.	NA
<delay>	Numeric	0 to 3	Delay value; 0 = none, 1 = 5ns, 2 = 10ns, 3 = 15ns	0

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

RETURNED PARAMETER SYNTAX

<delay>

Returned Parameter	Type	Value	Description
<delay>	Numeric	0 to 3	Programmed delay value.

COMMENTS

1. Not all SR192 I/O module support the output group enable delay.

3.8.3.2 OUTPut:ENABle[:SOURce]

This command is used to specify the output group enable for the I/O modules.

SYNTAX

OUTPut:ENABle[:SOURce] <group_name>,<enable>
OUTPut:ENABle[:SOURce]? <group_name>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<group_name>	Name	See section 3.1.1.2.2	The group name identifies the channels to program or query.	NA
<enable>	Ascii	TSENable1	Selects timing module enable 1.	See comment 1 below.
		TSENable2	Selects timing module enable 2.	
		FCNTl1	Selects front panel “FCNTL1” signal.	
		FCNTl2	Selects front panel “FCNTL2” signal.	
		CSTRobe	Selects front panel “CSTROBE” signal.	
		ALWays	Group enable always true.	
		NEVer	Group enable never true.	

RETURNED PARAMETER SYNTAX

<enable>

Returned Parameter	Type	Value	Description
<strobe>	Ascii	TSEN1	Timing module TSENABLE1 selected.
		TSEN2	Timing module TSENABLE2 selected.
		FCNT1	Front panel FCNTL1 selected.
		FCNT2	Front panel FCNTL2 selected.
		CSTR	Front panel CSTROBE selected.
		ALW	Always selected.
		NEV	Never selected.

COMMENTS

1. Default input strobe setting is based on module type; static I/O modules select “NEVer”, dynamic I/O modules select “TSENABLE1”.
2. Not all SR192 I/O modules support every enable source setting. Refer to the specific I/O module reference manual for valid choices. Selecting a non supported source will generate a “Parameter error”.

3.8.4 OUTPut:MASTer

This command allows an operator to set a particular SR192 to serve as master when multiple SR192's are used as a single system.

SYNTAX

OUTPut:MASTer <state>
OUTPut:MASTer?

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

PARAMETER LIST

Parameter	Type	Values	Description	Default
<state>	Boolean	OFF | 0	Disables the master mode.	OFF
		ON | 1	Enable the master mode.	

RETURNED PARAMETER SYNTAX

<state>

Parameter	Type	Value	Valid Range
<state>	Numeric	0	Master mode turned off.
		1	Master mode turned on.

COMMENTS

1. The link command is only valid for TSA or TSB.
2. Turning on TSB's link state will cause TSA to be selected after the LINK command has executed.

3.8.5 OUTPut:PGMClk<n>

This subsystem allows the user to program the two programmable clock generators.

SYNTAX

OUTPut:PGMClk<n>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<n>	Numeric	1	Selects PGMCLK1 to program/query	NA
		2	Selects PGMCLK2 to program/query	

3.8.5.1 OUTPut:PGMClk<n>[:DATA]

This command allows the user to program the programmable clock frequency.

SYNTAX

OUTPut:PGMClk<n>:DATA <frequency>
OUTPut:PGMClk<n>[:DATA]?

PARAMETER LIST

Parameter	Type	Values	Description	Default
<frequency>	Numeric	0 to 50000000	Frequency in Hz.	0

RETURNED PARAMETER SYNTAX

<frequency>

Returned Parameter	Type	Value	Description
<frequency>	Numeric	0 to 50000000	Programmed frequency value in Hz.

COMMENTS

None

3.8.5.2 OUTPut:PGMClk<n>:REFerence

This command allows the user to program the programmable clock generator reference clock source.

SYNTAX

OUTPut:PGMClk1:REFerence <source>
OUTPut:PGMClk1:REFerence?

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

PARAMETER LIST

Parameter	Type	Values	Description	Default
<source>	Ascii	INTernal	Selects the internal reference clock.	INTernal
		EXTernal	Selects the front panel “CLKREF” SMA	

RETURNED PARAMETER SYNTAX

<source>

Returned Parameter	Type	Value	Description
<source>	Ascii	INT	Internal reference clock selected.
		EXT	External reference clock selected.

COMMENTS

None

3.8.5.3 OUTPut:PGMClk<n>:RESet

This command resets both clock generators.

SYNTAX

OUTPut:PGMClk1:RESet

3.8.6 OUTPut:REFerence

The REFerence subsystem provides the commands for programming and querying the reference voltages to be used by the DAC module. These references are used by the variable voltage I/O modules and TSIO.

3.8.6.1 OUTPut:REFerence:AUTO

This command allows the user to program the update mode of the SR192 reference signals, auto or manual.

SYNTAX

OUTPut:REFerence:AUTO <auto_state>
OUTPut:REFerence:AUTO?

PARAMETER LIST

Parameter	Type	Values	Description	Default
<auto_state>	Boolean	OFF | 0	Sets the reference update mode to manual.	OFF
		ON | 1	Sets the reference update mode to auto.	

COMMENTS

1. The manual reference mode requires the “OUTPut:REFerence:UPDate” command to trigger the new references.

RETURNED PARAMETER SYNTAX

<auto_state>

Parameter	Type	Value	Description
<auto_state>	Numeric	0	Auto reference update state turned off.
		1	Auto reference update state turned on.

COMMENTS

None

3.8.6.2 OUTPut:REFerence:HIGH

The HIGH subtree allows the operator to program the output high voltages for the three SR192 voltage groups.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.8.6.2.1 OUTPut:REFerence:HIGH[:DATA]

This commands allows the operator to program the output high threshold for the selected group.

SYNTAX

OUTPut:REFerence:HIGH[:DATA] <voh>
OUTPut:REFerence:HIGH[:DATA]?

PARAMETER LIST

Parameter	Type	Values	Description	Default
<voh>	Numeric	-5.0 to +15.0	Programs the output high level for the variable voltage modules.	0.0

RETURNED PARAMETER SYNTAX

<voh>

Returned Parameter	Type	Value	Description
<voh>	Numeric	-5.0 to +15.0	Reference high reference setting.

COMMENTS

1. The voltage group is selected by the “OUTPut:REFerence:SELect” command.

3.8.6.2.2 OUTPut:REFerence:HIGH:INCRement

This command allows the operator to increment the current value of the output high level.

SYNTAX

OUTPut:REFerence:HIGH:INCRement <count>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<count>	Numeric	-1500 to 1500	The number of counts to increase/decrease the output high level of the selected group.	0.0

COMMENTS

1. The voltage group is selected by the “OUTPut:REFerence:SELect” command.
2. Each <count> is approximately 20mV for the SR110 DAC module.
3. Each <count> is 10mV for the SR210 DAC/MFC module.

3.8.6.3 OUTPut:REFerence:LOW

The LOW subtree allows the operator to program the output low voltages for the three SR192 voltage groups.

3.8.6.3.1 OUTPut:REFerence:LOW[:DATA]

This commands allows the operator to program the output low threshold for the selected group.

SYNTAX

OUTPut:REFerence:LOW[:DATA] <vol>
OUTPut:REFerence:LOW[:DATA]?

PARAMETER LIST

Parameter	Type	Values	Description	Default
<vol>	Numeric	-15.0 to +5.0	Programs the output low level for the variable voltage modules.	0.0

RETURNED PARAMETER SYNTAX

<vol>

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Returned Parameter	Type	Value	Description
<vol>	Numeric	-15.0 to +5.0	Reference low reference setting.

COMMENTS

1. The voltage group is selected by the “OUTPut:REFerence:SELect” command.

3.8.6.3.2 OUTPut:REFerence:LOW:INCRement

This command allows the operator to increment the current value of the output low level.

SYNTAX

OUTPut:REFerence:LOW:INCRement <count>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<count>	Numeric	-1500 to 1500	The number of counts to increase/decrease the output low level of the selected group.	0.0

COMMENTS

1. The voltage group is selected by the “OUTPut:REFerence:SELect” command.
2. Each <count> is approximately 20mV for the SR110 DAC module.
3. Each <count> is 10mV for the SR210 DAC/MFC module.

3.8.6.3.3 OUTPut:REFerence:ISR[:DATA]

This commands allows the operator to program the output ISR slew rate for the selected group.

SYNTAX

OUTPut:REFerence:ISR[:DATA] <vol>
OUTPut:REFerence:ISR[:DATA]?

PARAMETER LIST

Parameter	Type	Values	Description	Default
<isr>	Numeric	0.0 to +5.0	Programs the output ISR slew rate for the variable voltage modules.	2.0

RETURNED PARAMETER SYNTAX

<isr>

Returned Parameter	Type	Value	Description
<isr>	Numeric	0.0 to +5.0	ISR reference setting.

COMMENTS

1. The voltage group is selected by the “OUTPut:REFerence:SELect” command.

3.8.6.3.4 OUTPut:REFerence:ISR:INCRement

This command allows the operator to increment the current value of the output ISR level.

SYNTAX

OUTPut:REFerence:ISR:INCRement <count>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<count>	Numeric	-1000 to 1000	The number of counts to increase/decrease the output ISR level of the selected group.	NA

COMMENTS

1. The voltage group is selected by the “OUTPut:REFerence:SELect” command.
2. Each <count> is approximately 5mV for the SR110 DAC module.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3. Each <count> is 10mV for the SR210 DAC/MFC module.

3.8.6.4 OUTPut:REFerence:SELect

This command is used to select one of the three voltage groups for programming.

SYNTAX

OUTPut:REFerence:SELect <group>
OUTPut:REFerence:SELect?

PARAMETER LIST

Parameter	Type	Values	Description	Default
<group>	Ascii	VGRP1	Program voltage group one output references.	VGRP1
		VGRP2	Program voltage group two output references.	
		VGRP3	Program voltage group three output references.	
		ALL	Program all output references.	

RETURNED PARAMETER SYNTAX

<group>

Returned Parameter	Type	Value	Description
<group>	Ascii	VGRP1	Voltage group one selected.
		VGRP2	Voltage group two selected.
		VGRP3	Voltage group three selected.
		ALL	All voltage groups selected.

COMMENTS

None

3.8.6.5 OUTPut:REFerence:UPDate

This command updates the variable voltage reference signals when the “output:REFerence:AUTO” setting has been turned off.

SYNTAX

OUTPut:REFerence:UPDate

COMMENTS

1. The voltage group is selected by the “OUTPut:REFerence:SELect” command.

3.8.7 OUTPut:REGister

This subsystem allows the user to program or query the output register strobe source and state.

3.8.7.1 OUTPut:REGister:SOURce

This command is used to specify the output register signal for the I/O modules.

SYNTAX

OUTPut:REGister:SOURce <group_name>,<source>
OUTPut:REGister:SOURce? <group_name>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<group_name>	Name	See section 3.1.1.2.2	The group name identifies the channels to program or query.	NA

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Parameter	Type	Values	Description	Default
<source>	Ascii	STIM_LOAD	Selects timing module “STIM_LOAD” signal	See comment 1 below.
		TSSTrobe1	Selects timing module “TSSTROBE1” signal.	
		TSSTrobe2	Selects timing module “TSSTROBE2” signal.	
		FCNTl1	Selects front panel “FCNTL1” signal.	
		FCNTl2	Selects front panel “FCNTL2” signal.	
		CSTRobe	Selects front panel “CSTROBE” signal.	

RETURNED PARAMETER SYNTAX

<source>

Returned Parameter	Type	Value	Description
<source>	Ascii	STIM_LOAD	Timing module “STIM_LOAD” selected.
		TSST1	Timing module “TSSTROBE1” selected.
		TSST2	Timing module “TSSTROBE2” selected.
		FCNT1	Front panel “FCNTL1” selected.
		FCNT2	Front panel “FCNTL2” selected.
		CSTR	Front panel “CSTROBE” selected.

COMMENTS

1. Dynamic I/O modules can only select “STIM_LOAD”.
2. Not all SR192 I/O modules support every register source setting. Refer to the specific I/O module reference manual for valid choices. Selecting a non supported source will generate a “Parameter error”.

3.8.7.2 OUTPut:REGister[:STATe]

This command is used to enable/disable the output register.

SYNTAX

OUTPut:REGister[:STATe] <group_name>,<state>
OUTPut:REGister[:STATe]? <group_name>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<group_name>	Name	See section 3.1.1.2.2	The group name identifies the channels to program or query.	NA
<state>	Boolean	0 | OFF	Disables the output register mode.	See comment 1 below.
		1 | ON	Enables the output register mode.	

RETURNED PARAMETER SYNTAX

<state>

Returned Parameter	Type	Value	Description
<state>	Numeric	0	Output register mode disabled.
		1	Output register mode enabled.

COMMENTS

1. Static and non-algorithmic I/O modules disables the output register when the channel group is created. Algorithmic I/O modules enable the output register.

3.8.8 OUTPut:RESet

This command allows the user to generate a pulse on the front panel “UUTRST” pin.

SYNTAX

:OUTPut:RESet <mode>[,<duration>]

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

PARAMETER LIST

Parameter	Type	Values	Description	Default
<mode>	Ascii	LOW	Sets the static level of “UUTRST” to TTL low.	HIGH
		HIGH	Sets the static level of “UUTRST” to TTL high.	
		PULSe	Pulses the “UUTRST” for <duration>	
[,<duration>]	Numeric	1 to 10	Length of the pulse in 100ms increments.	NA

COMMENTS

1. If the “UUTRST” is set “LOW” than “PULSe” will generate a high pulse.
2. If the “UUTRST” is set “HIGH” than “PULSe” will generate a low pulse.

3.8.9 OUTPut:SYNC

The SYNC subtree allows the operator to define and enable a sync pulse during a specific word execution.

3.8.9.1 OUTPut:SYNC:POSition

This command allows the operator to define a sync trigger pulse to be generated at a specific I/O word.

SYNTAX

:OUTPut:SYNC:POSition <table_name>,<offset>
:OUTPut:SYNC:POSition?

PARAMETER LIST

Parameter	Type	Values	Description	Default
<table_name>	Name	See section 3.1.1.2.2	The table name identifies the I/O vectors where a sync trigger will be programmed.	NA
<offset>	Numeric	1 to 131072	The specific word number within the specified table where the sync trigger will be programmed.	NA

RETURNED PARAMETER SYNTAX

<offset>

Returned Parameter	Type	Value	Description
<offset>	Numeric	0-131071	The physical word address where the sync trigger is programmed.

COMMENTS

1. The sync trigger cannot be programmed while the timing module is running.

3.8.9.2 OUTPut:SYNC[:STATe]

The STATe command selects whether or not the SR192 drives the sync pulse output low.

SYNTAX

OUTPut:SYNC[:STATe] <state>
OUTPut:SYNC:STATe?

PARAMETER LIST

Parameter	Type	Values	Description	Default
<state>	Boolean	0 | OFF	Disables the sync trigger output.	OFF
		1 | ON	Enables the sync trigger output.	

RETURNED PARAMETER SYNTAX

<state>

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Returned Parameter	Type	Value	Description
<state>	Numeric	0	Sync trigger output disabled.
		1	Sync trigger output enabled.

COMMENTS

None

3.8.10 OUTPut:TIMing

The TIMing subtree allows the user to control the driver state of the timing set output signals for both TSA and TSB timing modules.

3.8.10.1 OUTPut:TIMing[:STATe]

This command allows the user to enable/disable the front panel timing signals.

SYNTAX

OUTPut:TIMing[:STATe] <state>
OUTPut:TIMing[:STATe]?

PARAMETER LIST

Parameter	Type	Values	Description	Default
<state>	Boolean	0 | OFF	Disables the timing output signals.	OFF
		1 | ON	Enables the timing output signals.	

RETURNED PARAMETER SYNTAX

<state>

Returned Parameter	Type	Value	Description
<state>	Numeric	0	Front panel timing signals disabled.
		1	Front panel timing signals enabled.

COMMENTS

None

3.8.11 OUTPut:TTLTrg <n>

The TTLTrg subtree controls the driving of the eight VXI backplane trigger lines. A numeric suffix <n> is required to identify the particular VXI trigger line, TTLTRG0 - TTLTRG7.

SYNTAX

OUTPut:TTLTrg<n>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<n>	Numeric	0 to 7	Selects a specific VXI backplane TTLTRG signal.	NA

3.8.11.1 OUTPut:TTLTrg <n>[:STATe]

This command allows the user to enable/disable the VXI backplane trigger drive.

SYNTAX

OUTPut:TTLTrg<n>[:STATe] <state>
OUTPut:TTLTrg<n>[:STATe]?

PARAMETER LIST

Parameter	Type	Values	Description	Default
<state>	Boolean	0 | OFF	Disables the specified “TTLTRG<n>” trigger output.	OFF
		1 | ON	Enables the specified “TTLTRG<n>” trigger output.	

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

RETURNED PARAMETER SYNTAX

<state>

Returned Parameter	Type	Value	Description
<state>	Numeric	0	Selected “TTLTRG” backplane output disabled.
		1	Selected “TTLTRG” backplane output enabled.

COMMENTS

None

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.9 Route Subsystem

The ROUTe subsystem allows the operator to name and define logical groupings of the I/O channels on the SR192. This is accomplished by naming a group and assigning the selected I/O channels to the group. Other commands may then be used to assign control signals such as input strobes and output enables to the logical group name.

KEYWORD **PARAMETER**

```
ROUTe
    :PATH
        :CATalog?
        :DEFine        <group_name>,<channel_list>
        :DELete
            [:NAME]    <group_name>
            :ALL
```

3.9.1 ROUTe:PATH Subtree

The PATH subtree allows the user to define and assign the I/O channels to groups.

3.9.1.1 ROUTe:PATH:CATalog?

This command will return all the defined channel groups as a comma separated list.

SYNTAX

ROUTe:PATH:CATalog?

RETURNED PARAMETER SYNTAX

{“<group_name>”}

Returned Parameter	Type	Value	Description
“<group_name>”	Name	See section 3.1.1.2.2	Comma separated list of all the defined channel groups.

COMMENTS

1. The empty string (”“) will be returned if no channel groups are defined.

3.9.1.2 ROUTe:PATH:DEFine

Used to group channels together so that control signals as well as memory locations can be programmed or queried.

SYNTAX

ROUTe:PATH:DEFine <group_name>,<channel_list>
ROUTe:PATH:DEFine? <group_name>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<group_name>	Name	See section 3.1.1.2.2	Group name to be defined or queried.	No groups defined
<channel_list>	Channel List		List of channel to assign to <group_name>.	

RETURNED PARAMETER SYNTAX

“<channel_list>”

Returned Parameter	Type	Values	Description
<channel_list>	Channel List	See section 3.1.1.2.2	The channel list assigned to the specified group name.

COMMENTS

1. A channel may only be assigned to one group.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

2. Only channels from the same module type can be grouped together.
3. Creating a channel group does not modify the contents of the memory for the affected channel(s).
4. If the group specified by <group_name> has not been defined then the empty string (“”) will be returned.

EXAMPLE

ROUTe:PATH:DEFine ADDR,(@1:32)
(Defines a channel group named ADDR and assigns channels 1 to 32 to ADDR.)
ROUTe:PATH:DEFine? ADDR
(Returns: ‘(@1:32)’)

3.9.1.3 ROUTe:PATH:DELete

The DELete subtree allows the user to delete channels that have been defined and free the I/O channels to be assigned to another group.

3.9.1.3.1 ROUTe:PATH:DELete[:NAME]

This command allows the user to delete a specific channel group name and free the channels.

SYNTAX

ROUTe:PATH:DELete <group_name>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<group_name>	Name	See section 3.1.1.2.2	Group to delete.	NA

COMMENTS

1. Deleting a channel group does not modify the contents of the memory for the affected channel(s).

3.9.1.3.2 ROUTe:PATH:DELete:ALL

This command allows the user to delete all the defined channel groups.

SYNTAX

ROUTe:PATH:DELete:ALL

COMMENTS

1. Deleting a channel group does not modify the contents of the memory for the affected channel(s).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.10 Sequence Subsystem

The SEQuence subsystem allows the operator to program and query the sequence memory.

KEYWORD	PARAMETER LIST
SEQuence	
:BRANch?	<seq_id>
:DEFine	<seq_name>,{<cycle_name>,<table_name>}
	<seq_name>,{<cycle_name>,<table_name>,<loop>}
	<seq_name>,<size>[,<cycle_name>]
:DELete	
[:NAME]	<seq_name>
:ALL	
:DIRectory?	
:GOSub	<from>,<to>[,<condition>]
:INITialize	<seq_addr>,<size>[,<cycle_name>]
:JUMP	<from>,<to>[,<condition>]
:LOOP	<sub_seq>,<loop>
:RESet	<seq_def>
:STOP	<seq_def>,<boolean>
:TABLe	<seq_def>,<table_name>
	<seq_def>,<table_addr>
:TIMing	<seq_def>,<next_cycle_name>[,<branch_cycle_name>]

3.10.1 SEQuence:BRANch?

This command is used to query the sequence memory for jump/gosub commands.

SYNTAX

SEQuence:BRANch? <sub_seq>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<sub_seq>	Subsequence	See section 3.1.1.2.2	Subsequence to query.	NA

RETURNED PARAMETER SYNTAX

<branch_type>[,<branch_mode>,<branch_seq>]

Returned Parameter	Type	Value	Description
<branch_type>	Ascii	RES	No sequence branch defined.
		JUMP	Sequence jump defined.
		GOS	Sequence gosub defined.
<branch_mode>	Ascii	UNC	Unconditional jump/gosub.
		JEN	Conditional jump/gosub if “JENable” true.
		ERR	Conditional jump/gosub if “ERRor” true.
		CTIM	Conditional jump/gosub if “CTIMeout” true.
		TSIN1,LOW	Conditional jump/gosub if “TSINput1” low.
		TSIN1,HIGH	Conditional jump/gosub if “TSINput1” high.
		TSIN2,LOW	Conditional jump/gosub if “TSINput2” low.
		TSIN2,HIGH	Conditional jump/gosub if “TSINput2” high.
<branch_seq>	Subsequence	See section 3.1.1.2.2	Subsequence to jump/gosub to.

COMMENTS

1. If <branch_type> is “RES” then the <branch_mode> and <seq_def> parameters are omitted.
2. The <seq_name>,<offset> form of <branch_seq> will be used unless the subsequence address does not exist within a defined sequence.
3. The SR100 timing module does not support sequence branching and will generate an “Execution error”.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.10.2 SEQuence:DEFine

This command allows the user to program a SR192 execution sequence. A execution sequence is made up of one or more timing cycle, table pairs called a subsequence.

SYNTAX

SEQuence:DEFine <seq_name>,{<cycle_name>,<table_name>}
SEQuence:DEFine <seq_name>,{<cycle_name>,<table_name>,<loop>}
SEQuence:DEFine <seq_name>,<size>[,<cycle_name>]
SEQuence:DEFine? <seq_name>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<seq_name>	Name	See section 3.1.1.2.2	Sequence name to define or query.	NA
<cycle_name>			Cycle timing for the specified subsequence.	
<table_name>			Table for the specified subsequence.	
<loop>	Numeric	1 to 32768	Loop count for the specified subsequence.	1
<size>	Numeric	1 to 131072	The number of subsequences.	See comment 3

RETURNED PARAMETER SYNTAX

SR100 Timing Module: "<seq_name>"[,{<cycle_name>,<table_name>}]
SR101 Timing Module: "<seq_name>"[,<size>,<offset>]

Returned Parameter	Type	Value	Description
<seq_name>	Name	See section 3.1.1.2.2	Name of the sequence.
<cycle_name>			Cycle name defined in the SR100 subsequence.
<table_name>			Table name defined in the SR100 subsequence.
<size>	Numeric	1 to 131071	The number of subsequences in the SR101 sequence.
<offset>		1 to 131071	The SR101 sequence memory offset where the sequence starts.

COMMENTS

1. The SR100 timing module only supports the first defined syntax of this command.
2. The SR100 timing module can have from 1 to 4 timing cycle, table pairs. The SR101 timing module can have from 1 to 131072 timing cycle, table pairs.
3. The third form of the command is used to initialize a large block of subsequences, each subsequence will be assigned the specified timing cycle, IDLE is omitted.
4. The empty ("",0,0) will be returned if the sequence is not defined.

3.10.3 SEQuence:DELete

The DELete subtree allows the user to delete execution sequences that have been defined.

3.10.3.1 SEQuence:DELete[:NAME]

This command allows the user to delete a specific sequence name and free the sequence memory assigned to it.

SYNTAX

SEQuence:DELete[:NAME] <seq_name>

PARAMETER LIST

Parameter	Type	Value	Description	Default
<seq_name>	Name	See section 3.1.1.2.2	Sequence to delete.	NA

COMMENTS

1. All JUMP/GOSUB references to the sequence to be deleted must be reset or a "Settings conflict" error will be generated.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.10.3.2 SEQuence:DELete:ALL

This command allows the user to delete all the defined channel groups.

SYNTAX

SEQuence:DELete:ALL

3.10.4 SEQuence:DIRectory?

The DIRectory command returns a listing of all the defined sequences.

SYNTAX

SEQuence:DIRectory?

RETURNED PARAMETER SYNTAX

{"<seq_name>"}

Returned Parameter	Type	Value	Description
<seq_name>	Name	See section 3.1.1.2.2	Comma separated list of all the defined sequences.

COMMENTS

1. The empty string ("") will be returned if no sequences are defined.

3.10.5 SEQuence:GOSub

The gosub command programs the SR101's branch memory as well as setting the subroutine bit in the control register. Subroutines unlike jumps will return to the sequence after completion.

SYNTAX

SEQuence:GOSub <from>,<to>[,<condition>]

PARAMETER LIST

Parameter	Type	Value	Description	Default
<from>	Subsequence	See section 3.1.1.2.2	Subsequence location to program.	NA
<to>	Subsequence	See section 3.1.1.2.2	Branch to subsequence.	
<condition>	Ascii	JENable	Branch if jump enable flag true.	NA
		ERRor	Branch if error flag true.	
		CTIMeout	Branch if cycle timeout flag true.	
		TSINput1,LOW	Branch if front panel "TSINPUT1" signal logic low.	
		TSINput1,HIGH	Branch if front panel "TSINPUT1" signal logic high.	
		TSINput2,LOW	Branch if front panel "TSINPUT2" signal logic low.	
		TSINput2,HIGH	Branch if front panel "TSINPUT2" signal logic high.	

COMMENTS

1. If <condition> is omitted then the gosub branch will be unconditional.
2. Conditional gosub branches require the jump enable qualifier be programmed in order for a branch to occur.

3.10.6 SEQuence:INITialize

This command is used to initialize the sequence memory.

SYNTAX

SEQuence:INITialize <seq_addr>,<size>[,<cycle_name>]

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

PARAMETER LIST

Parameter	Type	Value	Description	Default
<seq_addr>	Numeric	1 to 131071	Beginning subsequence location to program.	NA
<size>	Numeric	1 to 131071	Number of subsequence locations to program.	
<cycle_name>	Name	See section 3.1.1.2.2	Timing cycle to program each subsequence.	NA

COMMENTS

1. If <cycle_name> is omitted then the “IDLE” cycle is programmed for each subsequence.
2. The word count for each subsequence is set to one.
3. The table address is set to the subsequence address, i.e., subsequence address hex 1A is programmed with table address hex 1A, subsequence address hex 1B is programmed with table address hex 1B, etc.
4. Loop count for each subsequence is set to one.
5. Jump/Gosub logic reset.

3.10.7 SEQuence:JUMP

The jump command programs the SR101's branch memory as well as resetting the subroutine bit in the control register.

SYNTAX

SEQuence:JUMP <from>,<to>[,<condition>]

PARAMETER LIST

Parameter	Type	Value	Description	Default
<from>	Subsequence	See section 3.1.1.2.2	Subsequence location to program.	NA
<to>	Subsequence	See section 3.1.1.2.2	Branch to subsequence.	
<condition>	Ascii	JENable	Branch if jump enable flag true.	NA
		ERROr	Branch if error flag true.	
		CTIMeout	Branch if cycle timeout flag true.	
		TSINput1,LOW	Branch if front panel “TSINPUT1” signal logic low.	
		TSINput1,HIGH	Branch if front panel “TSINPUT1” signal logic high.	
		TSINput2,LOW	Branch if front panel “TSINPUT2” signal logic low.	
		TSINput2,HIGH	Branch if front panel “TSINPUT2” signal logic high.	

COMMENTS

1. If <condition> is omitted then the jump branch will be unconditional.
2. Conditional gosub branches require the jump enable qualifier be programmed in order for a branch to occur.

3.10.8 SEQuence:LOOP

The loop command programs the sequence memory loop count for the selected subsequence.

SYNTAX

SEQuence:LOOP <sub_seq>,<loop>
SEQuence:LOOP? <sub_seq>

PARAMETER LIST

Parameter	Type	Value	Description	Default
<sub_seq>	Subsequence	See section 3.1.1.2.2	Subsequence location to program.	NA
<loop>	Numeric	1 to 32768	Loop count to program the subsequence.	

RETURNED PARAMETER SYNTAX

<loop>

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Returned Parameter	Type	Value	Description
<loop>	Numeric	1 to 32768	Programmed loop count for the specified subsequence.

COMMENTS

None

3.10.9 SEQuence:RESet

This command resets the specified subsequence branch logic.

SYNTAX

SEQuence:RESet <sub_seq>

PARAMETER LIST

Parameter	Type	Value	Description	Default
<sub_seq>	Subsequence	See section 3.1.1.2.2	Subsequence location to program.	NA

3.10.10 SEQuence:STOP

The STOP command programs the sequence memory branch control to stop at the end of the current sequence and return to IDLE.

SYNTAX

SEQuence:STOP <sub_seq>

PARAMETER LIST

Parameter	Type	Value	Description	Default
<sub_seq>	Subsequence	See section 3.1.1.2.2	Subsequence location to program.	NA

3.10.11 SEQuence:TABLe

The table command programs the sequence memory table address.

SYNTAX

SEQuence:TABLe <sub_seq>,<table_name>
SEQuence:TABLe <sub_seq>,<table_addr>
SEQuence:TABLe? <sub_seq>

PARAMETER LIST

Parameter	Type	Value	Description	Default
<sub_seq>	Subsequence	See section 3.1.1.2.2	Subsequence location to program or query.	NA
<table_name>	Name	See section 3.1.1.2.2	Selected table to assign to the subsequence.	NA
<table_addr>	Numeric	0 to 131071	Physical table address to assign to the subsequence.	NA

RETURNED PARAMETER SYNTAX

<table_name>,<table_addr>

Returned Parameter	Type	Value	Description
<table_name>	Name	See section 3.1.1.2.2	Name of the table assigned to the subsequence.
<table_addr>	Numeric	0 to 131071	Physical offset of the assigned table.

COMMENTS

1. If the assigned table address does not map to a defined table then the empty string (”“) will be returned for <table_name>.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.10.12 SEQuence:TIMing

The timing command programs the sequence memory timing selection.

SYNTAX

SEQuence:TIMing <sub_seq>,<ts>[,<jump>]
SEQuence:TIMing? <sub_seq>

PARAMETER LIST

Parameter	Type	Value	Description	Default
<sub_seq>	Subsequence	See section 3.1.1.2.2	Subsequence location to program or query.	NA
<ts>	Name	See section 3.1.1.2.2	Selected timing cycle to assign to the subsequence.	NA
<jump>	Name	See section 3.1.1.2.2	Selected timing cycle to assign is a branch is taken.	NA

RETURNED PARAMETER SYNTAX

<ts>,<jump>

Returned Parameter	Type	Value	Description
<ts>	Name	See section 3.1.1.2.2	Name of the timing cycle assigned to the subsequence.
<jump>	Name	See section 3.1.1.2.2	Name of the timing cycle assigned as the branch timing cycle.

COMMENTS

1. If the <seq_name>,<offset> form of <sub_seq> is used then the <ts> will be the timing cycle for the specified subsequence offset.
2. If the <seq_addr> form of <sub_seq> is used then the <ts> will be timing cycle for the next higher <seq_addr>.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.11 Status Subsystem

The STATus subsystem allows the user to program and query the SCPI status reporting registers, OPERation and QUEStionable. The SCPI status registers are composed of three elements, the condition register, the event register and enable register described below.

CONDition: The condition register contains the current state of the status bits.

EVENt: The event register contains the positive transition status of every bit in the condition register, i.e., a one in bit 8 of the event register indicates a positive transition of bit 8 in the condition register. The event register is cleared when queried by the user or by the "*CLS" command.

ENABle: The enable register is used to report positive transitions in the condition register. Specific bits can be enabled or disabled by changing the contents of the enable register.

Summary bits are generated and cascaded down to the IEEE 488.2 status register. The following table describes all the summary bits in the status reporting structure

Summary Bit	Description
Event Summary (STB bit 5)	Set when one or more bits are set in the event status register (ESR) and the corresponding bit in the event status enable register (ESE) is set high.
Operation Summary (STB bit 7)	Set when one or more bits are set in the operation event register (STATus:OPERation:EVENt) and the corresponding bit in the operation enable register (STATus:OPERation:ENABle) is set high.
Instrument Summary (STATus:OPERation:CONDition bit 13)	Set when one or more bits are set in the operation instrument event register (STATus:OPERation:INSTrument:EVENt) and the corresponding bit in the operation enable register (STATus:OPERation:INSTrument:ENABle) is set high.
ISUM<n> (STATus:OPERation:INSTrument:CONDition bit <n>)	Set when one or more bits are set in the operation instrument isummary<n> event register (STATus:OPERation:INSTrument:ISUMmary<n>:EVENt) and the corresponding bit in the operation enable register (STATus:OPERation:INSTrument:ISUMmary<n>:ENABle) is set high. The following lists the mapping of modules to ISUMmary<n>: TSA ISUMmary1 TSB ISUMmary2 DRA1 ISUMmary3 DRA2 ISUMmary4 DRA3 ISUMmary5 DRA4 ISUMmary6 DRA5 ISUMmary7 DRA6 ISUMmary8 DRB1 ISUMmary9 DRB2 ISUMmary10 DRB3 ISUMmary11 DRB4 ISUMmary12 DRB5 ISUMmary13 DRB6 ISUMmary14

NOTE

The QUEStionable registers are included for SCPI compatibility. These registers are not used by the SR192.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Figure 3-3 below illustrates the SCPI and IEEE 488.2 status reporting registers.

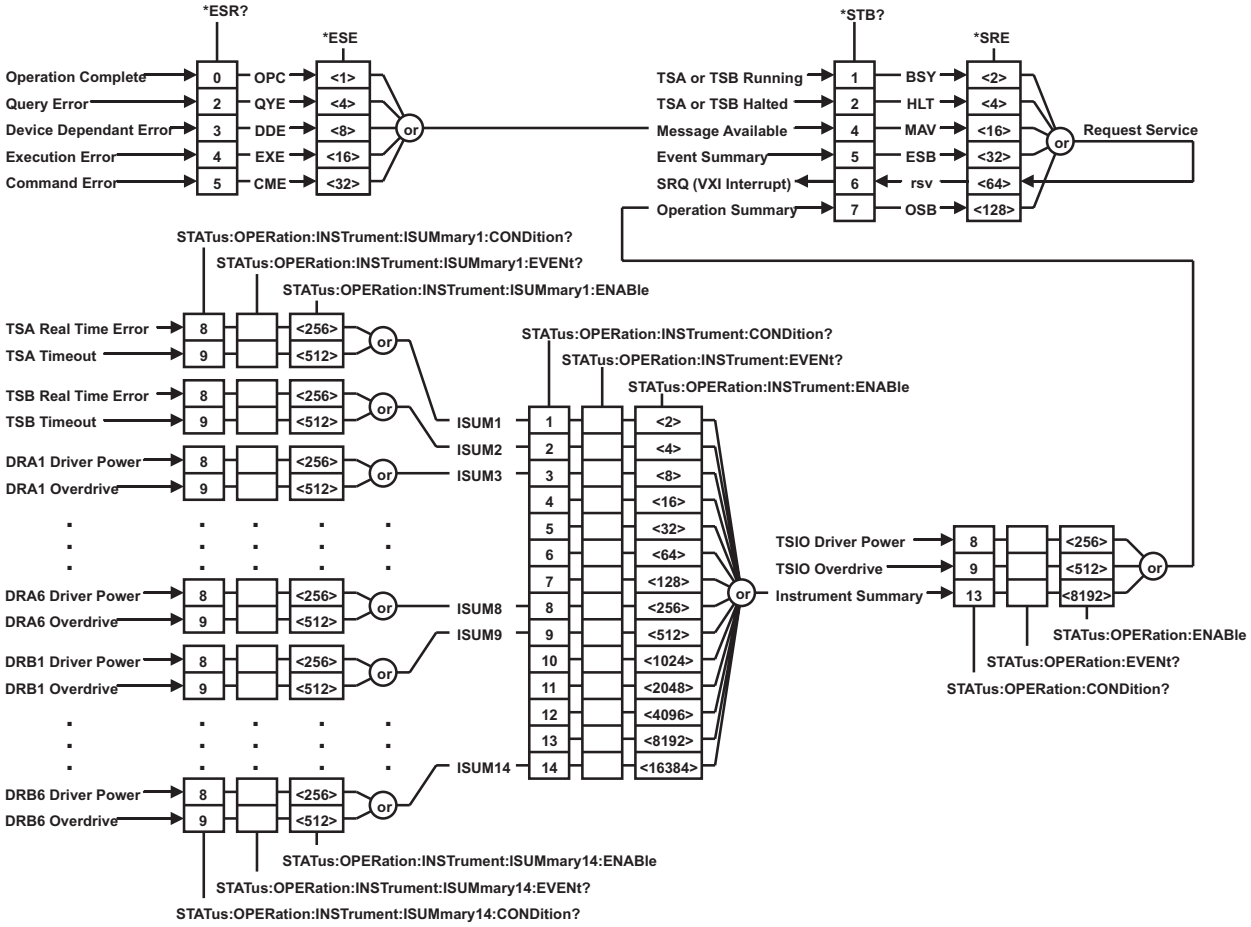


Figure 3-3 Status Subsystem Registers

KEYWORD	PARAMETER LIST
STATus	
:OPERation	
:CONDition?	
:ENABle	<numeric>
[:EVENt]?	
:INSTrument	
:CONDition?	
:ENABle	<numeric>
[:EVENt]?	
:ISUMmary<n>	
:CONDition?	
:ENABle	<numeric>
[:EVENt]?	
:PRESet	
:QUEStionable	
[:EVENt]?	
:CONDition?	
:ENABle	<numeric>

3.11.1 STATus:OPERation

The operation subtree enables the user to program and query the operation registers for the SR192.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.11.1.1 STATus:OPERation:CONDition?

This command returns the content of the condition register and is non-destructive.

SYNTAX

STATus:OPERation:CONDition?

RETURNED PARAMETER SYNTAX

<condition>

Parameter	Type	Value	Description
<condition>	Numeric	32767 to -32768 (0_h-$FFFF_h$)	Operation condition register.

COMMENTS

1. Only three bits will be set by the SR192:

Condition Register Bit Definition															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NU		Instrument Summary	NU			Overdrive	Power On	NU							

3.11.1.2 STATus:OPERation:ENABle

This command allows the operator to program which bits in the operation event register that will set the operation summary bit in the IEEE 488.2 status register of the SR192, refer to section 3.15.11.

SYNTAX

STATus:OPERation:ENABle <mask>
STATus:OPERation:ENABle?

PARAMETER LIST

Parameter	Type	Values	Description	Default
<mask>	Numeric	32767 to -32768 (0_h-$FFFF_h$)	A one in a bit position enables the corresponding bit in the operation event register to generate the operation summary bit.	0

RETURNED PARAMETER SYNTAX

<mask>

Parameter	Type	Value	Description
<mask>	Numeric	32767 to -32768 (0_h-$FFFF_h$)	Contents of the operation enable register.

COMMENTS

None

3.11.1.3 STATus:OPERation[:EVENt]?

This command returns the contents of the operation event register. The operation event register is set on a positive transition of an operation condition register bit. The operation event register is cleared when read.

SYNTAX

STATus:OPERation[:EVENt]?

RETURNED PARAMETER SYNTAX

<event>

Parameter	Type	Value	Description
<event>	Numeric	32767 to -32768 (0_h-$FFFF_h$)	Operation event register.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

COMMENTS

1. The following table describes the data returned by the SR192:

Event Register Bit Definition															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NU		Instrument Summary	NU			Overdrive	Power On	NU							

2. The contents of the operation event register are cleared when queried.
3. All event registers are cleared with a *CLS command.

3.11.1.4 STATus:OPERation:INSTrument

The INSTrument subtree allows the operator to query and set the INSTrument condition, event and enable registers.

3.11.1.4.1 STATus:OPERation:INSTrument:CONDition?

This command returns the content of the instrument condition register and is non-destructive.

SYNTAX

STATus:OPERation:INSTrument:CONDition?

RETURNED PARAMETER SYNTAX

<condition>

Parameter	Type	Value	Description
<condition>	Numeric	32767 to -32768 (0_h-$FFFF_h$)	Instrument condition register.

COMMENTS

1. The following lists the module mapping of the instrument condition register.

Instrument Condition Register Bit Definition															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NU	DRB6	DRB5	DRB4	DRB3	DRB2	DRB1	DRA6	DRA5	DRA4	DRA3	DRA2	DRA1	TSB	TSA	NU

3.11.1.4.2 STATus:OPERation:INSTrument:ENABle

This command allows the operator to program which bits in the instrument event register that will set the instrument summary bit in the operation condition register of the SR192, refer to section 3.11.1.1.

SYNTAX

STATus:OPERation:INSTrument:ENABle <mask>
STATus:OPERation:INSTrument:ENABle?

PARAMETER LIST

Parameter	Type	Values	Description	Default
<mask>	Numeric	32767 to -32768 (0_h-$FFFF_h$)	A one in each bit position enables the corresponding bit in the instrument event register to generate the instrument summary bit.	0

RETURNED PARAMETER SYNTAX

<mask>

Parameter	Type	Value	Description
<mask>	Numeric	32767 to -32768 (0_h-$FFFF_h$)	Contents of the instrument enable register.

COMMENTS

None

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.11.1.4.3 STATus:OPERation:INSTrument[:EVENt]?

This command returns the contents of the instrument event register. The instrument event register is set on a positive transition of a instrument condition register bit.

SYNTAX

STATus:OPERation:INSTrument[:EVENt]?

RETURNED PARAMETER SYNTAX

<event>

Parameter	Type	Value	Description
<event>	Numeric	32767 to -32768 (0_h-$FFFF_h$)	Instrument event register.

COMMENTS

1. The following table describes the data returned by the SR192:

Instrument Event Register Bit Definition															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NU	DRB6	DRB5	DRB4	DRB3	DRB2	DRB1	DRA6	DRA5	DRA4	DRA3	DRA2	DRA1	TSB	TSA	NU

2. The contents of the instrument event register are cleared when queried.
3. All event registers are cleared with a *CLS command.

3.11.1.4.4 STATus:OPERation:INSTrument:ISUMmary<n>

This subsystem allows the user to program or query the ISUMmary condition, event and enable registers.

SYNTAX

ISUMmary<n>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<n>	Numeric	1 to 14	Each SR192 module is mapped to a instrument summary bit.	NA

3.11.1.4.4.1 STATus:OPERation:INSTrument:ISUMmary<n>:CONDition?

This command returns the contents of the isummary condition register and is non-destructive.

SYNTAX

STATus:OPERation:INSTrument:ISUMmary<n>:CONDition?

RETURNED PARAMETER SYNTAX

<condition>

Parameter	Type	Value	Description
<condition>	Numeric	32767 to -32768 (0_h-$FFFF_h$)	Isummary condition register.

COMMENTS

1. Only two bits will be set by the SR192:

Module	Instrument Isummary Condition Register Bit Definition															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Timing	NU						Timeout	Real Time Error	NU							
I/O Module							Overdrive	Power On								

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.11.1.4.4.2 STATus:OPERation:INSTrument:ISUMmary<n>:ENABle

This command allows the operator to program which bits in the operation isummary event register that will set the summary bit in the instrument condition register of the SR192, refer to section 3.11.1.4.1.

SYNTAX

STATus:OPERation:INSTrument:ISUMmary<n>:ENABle <mask>
STATus:OPERation:INSTrument:ISUMmary<n>:ENABle?

PARAMETER LIST

Parameter	Type	Values	Description	Default
<mask>	Numeric	32767 to -32768 (0_h-$FFFF_h$)	A one in each bit position enables the corresponding bit in the isummary event register to generate the summary bit in the instrument condition register.	0

RETURNED PARAMETER SYNTAX

<mask>

Parameter	Type	Value	Description
<mask>	Numeric	32767 to -32768 (0_h-$FFFF_h$)	Contents of the isummary enable register.

COMMENTS

None

3.11.1.4.4.3 STATus:OPERation:INSTrument:ISUMmary<n>[:EVENt]?

This command returns the contents of the isummary event register. The isummary event register is set on a positive transition of the isummary condition register bit.

SYNTAX

STATus:OPERation:INSTrument:ISUMmary<n>[:EVENt]?

RETURNED PARAMETER SYNTAX

<event>

Parameter	Type	Value	Description
<event>	Numeric	32767 to -32768 (0_h-$FFFF_h$)	Isummary event register.

COMMENTS

1. The following table describes the data returned by the SR192:

Module	Instrument Isummary Event Register Bit Definition															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Timing	NU						Timeout	Real Time Error	NU							
I/O							Overdrive	Power On								

2. The contents of the isummary event register are cleared when queried.
3. All event registers are cleared with a *CLS command.

3.11.2 STATus:QUEStionable

The QUEStionable subtree is included for SCPI compatibility.

All queries will return zero.

3.11.2.1 STATus:QUEStionable[:EVENt]?

SYNTAX

STATus:QUEStionable[:EVENt]?

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

RETURNED PARAMETER SYNTAX

Returns zero.

3.11.2.2 STATus:QUEStionable:CONDition?

SYNTAX

STATus:QUEStionable:CONDition?

RETURNED PARAMETER SYNTAX

Returns zero.

3.11.2.3 STATus:QUEStionable:ENABle

SYNTAX

STATus:QUEStionable:ENABle <mask>
STATus:QUEStionable:ENABle?

PARAMETER LIST

Parameter	Type	Values	Description	Default
<mask>	Numeric	32767 to -32768 (0_h-$FFFF_h$)	Not used.	0

RETURNED PARAMETER SYNTAX

<mask>

Parameter	Type	Value	Description
<mask>	Numeric	32767 to -32768 (0_h-$FFFF_h$)	Contents of the questionable enable register.

COMMENTS

None

3.11.3 :PRESet

The PRESet command resets the enable registers to zero for both the SCPI and IEEE 488.2.

SYNTAX

STATus:PRESet

COMMENTS

1. The PRESet command does not affect any of the event registers.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.12 System Subsystem

This subsystem is used to query system errors as well as the SCPI version.

KEYWORD **PARAMETER LIST**

SYSTem
:ERRor?
:VERSion?

3.12.1 SYSTem:ERRor?

This command returns an error code/message from the error queue.

SYNTAX

SYSTem:ERRor?

RETURNED PARAMETER SYNTAX

<error_number>,"<message>[;<extension>]"

Parameter	Type	Value	Description
<error_number>	Numeric	32767 to -32768 (0_h-$FFFF_h$)	Error number, negative numbers are SCPI defined and positive numbers are SR192 dependent errors.
<message>	Ascii	See comment 4	Error message.
<extension>	Ascii	See comment 5	Error message extension.

COMMENTS

1. The error queue is a first in first out list of error messages. If the error queue overflows the last error in the queue is replaced with the following error: -350,"Queue overflow"
2. When all the errors have been read from the queue, (i.e. error queue is empty), the error query will return the following: 0,"No error"
3. Error numbers are segmented into the following categories:

Range	Description
-100 to -199	Command errors, sets the CME bit in the IEEE 488.2 event register.
-200 to -299	Execution errors, sets the EXE bit in the IEEE 488.2 event register.
-300 to -399	Device dependant error, sets the DDE bit in the IEEE 488.2 event register.
-400 to -499	Query error, sets QYE bit in the IEEE 488.2 event register.

4. The following is a list of all the SR192's error messages:

Returned Message	Cause
0,"No error"	Error queue empty.
-100,"Command error"	Parser failed to recognize command header.
-102,"Syntax error"	Parser failed to recognize command keyword.
-108,"Parameter not allowed"	Parser received too many parameters.
-109,"Missing parameter"	Parser received too few parameters.
-160,"Block data error"	Error sending/receiving <block> data type.
-200,"Execution error"	A valid command could not be executed.
-220,"Parameter error"	An error in the parameter list was detected.
-221,"Settings conflict"	A command could not be executed due to the current SR192 settings.
-311,"Memory error"	SR192 unable to allocate memory required for the command.
-350,"Queue Overflow"	Too many errors in the error queue.
-400,"Query error"	User attempted to read data from an empty output buffer or data from a previous query was not read and subsequently deleted.

5. The following is a list of all the SR192's error message extensions:

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Message Extension	Cause
“Field data not available while BUSY”	Attempt to program or query I/O module memory while the controlling timing module is running, i.e., not idle or reset.
“Sync cannot be set while BUSY”	Attempt to program the sync pulse trigger while the controlling timing module is running, i.e., not idle or reset.
“Error stopping emulator”	Attempt to stop or reset a timing sequence without TS_CLK active.
“Error starting timing set”	Attempt to start a run or idle sequence without TS_CLK active.
“Timing data not available while BUSY or IDLE”	Attempt to program or query the timing cell memory while the timing module is running or idle, i.e., not reset.
“Error setting timing set clock”	Attempt to select a new TS_CLK source without TS_CLK active.
“Cannot test strobe in the last cell”	Attempt to program a edge in the last cell of a timing set.
“Synchronizing timing modules”	Attempt to synchronize TSA and TSB without TS_CLK active.
“Sequence data not available while BUSY”	Attempt to program or query the sequence memory while the timing module is running, i.e., not idle or reset.

3.12.2 SYSTem:VERSion?

This command returns the SCPI version that the SR192 complies with.

SYNTAX

SYSTem:VERSion?

RETURNED PARAMETER SYNTAX

<version>

Parameter	Type	Value	Description
<version>	Numeric	1994.0	SR192 SCPI version.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.13 Table Subsystem

The TABLe subsystem is used to define and edit tables.

KEYWORD	PARAMETER LIST
TABLe	
[:DATA]	<table_name>,<block>
:DEFine	<table_name>,(<size> | <table_copy>)
:DELete	
:ALL	
[:NAME]	<table_name>
:DIRectory?	
:FREE?	
:JENable	<table_name>, (ALL | NONE |<offset>,<boolean>)
:MEMory	
:DATA	<table_name>,<group_name>,<block>
:FILL	<table_name>,<group_name>,<pattern>,<offset>[,<numeric>]
:WORD	<table_name>,<group_name>,<offset>,<data>{,<data>}
:SELect	<memory>

3.13.1 TABLe[:DATA]

This command programs the contents of the I/O module memory.

SYNTAX

TABLe[:DATA] <table_name>,<block>
TABLe[:DATA]? <table_name>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<table_name>	Name	See section 3.1.1.2.2	Selected table to program or query.	NA
<block>	Arb Block	See section 3.1.1.2.2	Data to program into I/O module memory.	NA

RETURNED PARAMETER SYNTAX

<block>

Parameter	Type	Value	Description
<block>	Arb Block	See section 3.1.1.2.2	Data from the selected table dynamic I/O module memory.

COMMENTS

1. The memory is programmed starting from the last dynamic I/O module installed in the selected timing module group. The following figure illustrates the byte order programming of the block data into the I/O module memory.

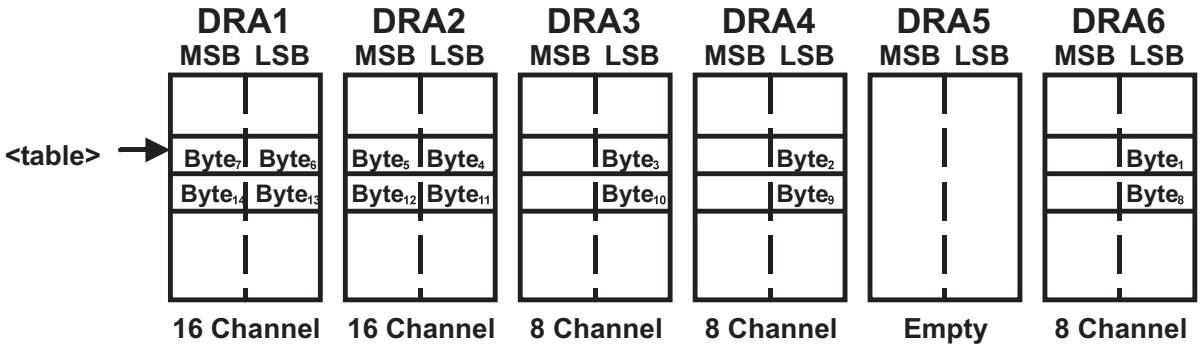


TABLe:DATA <table>,#214Byte$_1$Byte$_2$ . . . Byte$_{14}$

Figure 3-4 TABLe:DATA Programming Byte Order

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

2. The memory is queried starting from the last dynamic I/O module installed in TSB timing group if TSA and TSB are linked.

3.13.2 TABLe:DEFine

This command allocates and initializes a new table.

SYNTAX

TABLe:DEFine <table_name>,<size>
TABLe:DEFine <table_name>,<table_copy>
TABLe:DEFine? <table_name>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<table_name>	Name	See section 3.1.1.2.2	Table to define or query.	NA
<size>	Numeric	1 to 131072	Number of I/O addresses.	
<table_copy>	Name	See section 3.1.1.2.2	Table to copy.	

RETURNED PARAMETER SYNTAX

"<table_name>",<size>,<offset>

Returned Parameter	Type	Value	Description
<table_name>	Name	See section 3.1.1.2.2	Name of the table.
<size>	Numeric	1 to 131071	The number of I/O address.
<offset>		1 to 131071	The address memory offset where the table starts.

COMMENTS

1. If the "<table_name>,<size>" form of the command is used then the contents of the new tables memory will be initialized to the following:
   - OUTPut Programmed low.
   - TRIState Programmed high.
   - EXPect Programmed low.
   - MASK Programmed high.
   - RECord Not initialized.
2. If the "<table_name>,<table_copy>" form of the command is used then the size and contents of the new table memory will be copied from the specified copy table.
3. The string (""",0,0) will be returned if the table is not defined.

3.13.3 TABLe:DELete

This subtree allows the operator to delete tables and de-allocate the I/O addresses assigned to them. Tables defined after the deleted table are realigned to prevent fragmentation.

3.13.3.1 TABLe:DELete:ALL

This command deletes all the defined table(s) and frees system memory required to manage the tables.

SYNTAX

TABLe:DELete:ALL

3.13.3.2 TABLe:DELete[:NAME]

This command deletes the specified table and de-allocates its field memory.

SYNTAX

TABLe:DELete[:NAME] <table_name>

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

PARAMETER LIST

Parameter	Type	Values	Description	Default
<table_name>	Name	See section 3.1.1.2.2	Selected table to delete.	NA

COMMENTS

1. In order to prevent fragmentation tables defined after the deleted table will be realigned. The time required to realign the tables will depend on the size of the tables defined after the deleted table, the greater the size the longer realignment will take..

3.13.4 TABLe:DIRectory?

Returns a directory listing of all the defined table names.

SYNTAX

TABLe:DIRectory?

RETURNED PARAMETER SYNTAX

{"<table_name>",<size>,<offset>}

Returned Parameter	Type	Value	Description
<table_name>	Name	See section 3.1.1.2.2	Name of the table.
<size>	Numeric	1 to 131071	The number of I/O address.
<offset>		1 to 131071	The address memory offset where the table starts.

COMMENTS

1. The string (""",0,0) will be returned if no table are defined.

3.13.5 TABLe:FREE?

This command returns the amount of used and free I/O memory addresses.

SYNTAX

TABLe:FREE?

RETURNED PARAMETER SYNTAX

<used>,<free>

Returned Parameter	Type	Value	Description
<used>	Numeric	0 to 131072	Number of used I/O memory address.
<free>	Numeric	0 to 131072	Number of available I/O memory address.

3.13.6 TABLe:JENable

This command programs the jump enable bit in the table memory. The jump enable bit allows the operator to enable/disable conditional JUMP and GOSUB branches in the SR101 timing module.

SYNTAX

TABLe:JENable <table_name>,ALL
TABLe:JENable <table_name>,NONE
TABLe:JENable <table_name>,<offset>,<state>
TABLe:JENable? <table_name>,ALL
TABLe:JENable? <table_name>,NONE
TABLe:JENable? <table_name>,<word_#>,<state>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<table_name>	Name	See section 3.1.1.2.2	Name of the table to program or query.	

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Parameter	Type	Values	Description	Default
<offset>	Numeric	1 to 131072	Table index to program/query.	
<state>	Boolean	OFF | 0	Sets jump enable bit off.	OFF
		ON | 1	Sets jump enable on.	

RETURNED PARAMETER SYNTAX

<state>

Returned Parameter	Type	Value	Description
<state>	Numeric	0 or 1	The state of the query, 0 = false and 1 = true.

COMMENTS

1. The “ALL” parameter enables the jump enable bit true for all words in the table.
2. The “NONE” parameter disables the jump enable bit true for all words in the table.
3. A specific word can be enabled/disabled by entering the offset followed by ON or OFF to enable/disable respectively.
4. “TABL:JEN? PATT1,NONE” will returns “1” if all jump enable bits in table PATT1 are disabled and returns “0” if one or more bits are enabled.)
   “TABL:JEN? PATT1,1” will return “1” if the jump enable bit for word one in table PATT1 is enabled and “0” if disabled.

3.13.7 TABLe:MEMory

This subtree allows the operator to load data or execute fill functions to the memory selected.

3.13.7.1 TABLe:MEMory:DATA

This command allows the user to program the specified channels of a selected table.

SYNTAX

TABLe:MEMory:DATA <table_name>,<group_name>,<block>
TABLe:MEMory:DATA? <table_name>,<group_name>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<table_name>	Name	See section 3.1.1.2.2	Name of the table to program or query.	OUTPut
<group_name>	Name		Name of the group to program.	
<block>	Arb Block		Data to program into table group.	

RETURNED PARAMETER SYNTAX

<block>

Parameter	Type	Value	Description
<block>	Arb Block	See section 3.1.1.2.2	Data from the selected table group.

COMMENTS

1. A byte is required for each 8 channels or portion thereof (i.e. a 20 channel group will require 3 bytes, a 30 channel group will require 4 bytes). The following illustrates the bit order programming of the block data into the group memory.

   ROUTe:PATH:DEFine ADDR,(@17:36)
   TABLe:MEMory:DATA <table>,ADDR,#214$Byte_1 Byte_2 \ldots Byte_6$

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

	ADDR																			
Group Channel Number	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
Front Panel Channel Number	CH36	CH35	CH34	CH33	CH32	CH31	CH30	CH29	CH28	CH27	CH26	CH25	CH24	CH23	CH22	CH21	CH20	CH19	CH18	CH17
<table> word 1	$Byte_1$				$Byte_2$								$Byte_3$							
	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
<table> word 2	$Byte_4$				$Byte_5$								$Byte_6$							
	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

2. The specific I/O memory to be programmed is selected by the “TABLe:SELect” command.
3. The data is returned identical to the “TABLe:MEMory:DATA” command.
4. Static I/O module groups are not queried by this command.

3.13.7.2 TABLe:MEMory:FILL

The FILL command executes the fill pattern on the memory previously selected and the table and channel group specified in the command.

SYNTAX

TABLe:MEMory:FILL <table_name>,<group_name>,<fill_pattern>,<offset>[,<fill_parameter>]

PARAMETER LIST

Parameter	Type	Values	Description	Default
<table_name>	Name	See section 3.1.1.2.2	Name of the table to program.	
<group_name>	Name		Name of the group to program.	
<fill_pattern>	Ascii	COMPlement	Perform a ones complement on the contents of each word.	
		INCRement	Successive words are formed by adding the <fill_parameter> to the current word and then updating the current word for the next increment. The range of <fill_parameter> is from -128 to 127.	
		RAMP	The initial word is set to zero. Successive words are formed by setting succeeding bit positions to one until all bit positions are one. Then zeros are written starting with the MSB.	
		RANDom	The <fill_parameter> is used as the initial word value. A pseudo random number is generated from the initial word and stored in successive words. The range of <fill_parameter> is from -32768 to 32767.	
		ROTate	Successive words are generated by left shifting the current word by the <fill_parameter> and then updating the current word for the next rotate operation. The range of the <fill_parameter> is from 1 to the number of channels minus one.	
		REPeat	Repeats the first word throughout the tables memory.	
		TOGGle	Successive words are generated by performing a ones complement on the current word and then updating the current word for the next complement operation.	
<offset>	Numeric	1 to 131072	Table index for the fill function.	
<fill_parameter>	Numeric	See <fill_pattern>	Fill pattern parameter.	

COMMENTS

1. The execution time for fill functions vary according to table size, number of channels and the selected fill pattern.

3.13.7.3 TABLe:MEMory:WORD

This command allows the user to program a specific group of channels within table.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SYNTAX

TABLe:MEMory:WORD <table_name>,<group_name>,<offset>,{<data>}
TABLe:MEMory:WORD? <table_name>,<group_name>,<offset>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<table_name>	Name	See section 3.1.1.2.2	Name of the table to program.	
<group_name>	Name		Name of the group to program.	
<offset>	Numeric	1 to 131072	Table index to program data into.	
<data>	Numeric	0-4,294,967,295 (0_h-$FFFFFFFF_h$)	Data to program into table group.	

RETURNED PARAMETER SYNTAX

{<data>}

Parameter	Type	Value	Description
<data>	Numeric	0-4,294,967,295 (0_h-$FFFFFFFF_h$)	Data from the selected table group.

COMMENTS

1. Each <data> element contains data for up to 32 channels. Groups that exceed 32 channels require an additional <data> value separated by a comma for each additional 32 channels.
2. The data will be mapped to the groups channels with the least significant bit mapped to the lowest channel number.

3.13.8 TABLe:SELect

This command allows the user to select a I/O module memory for further programming/query commands.

SYNTAX

TABLe:SELect <memory>
TABLe:SELect?

PARAMETER LIST

Parameter	Type	Values	Description	Default
<memory>	Ascii	OUTPut	Selects the output memory.	OUTPut
		TRIState	Selects the tristate memory.	
		EXPect	Selects the expect memory.	
		MASK	Selects the mask memory.	
		RECord	Selects the record memory.	
		ERRor	Selects the error memory.	
		RESPonse	Selects the response memory.	

RETURNED PARAMETER SYNTAX

<memory>

Parameter	Type	Values	Description
<memory>	Ascii	OUTPut	Output memory selected.
		TRIState	Tristate memory selected.
		EXPect	Expect memory selected.
		MASK	Mask memory selected.
		RECord	Record memory selected.
		ERRor	Error memory selected.
		RESPonse	Response memory selected.

COMMENTS

None

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.14 Timing Subsystem

The TIMing subsystem allows the operator to set and edit the timing setup selections and the sixteen timing sets for each of the four timing set pages.

KEYWORD	PARAMETER LIST
TIMing	
:CELL	<cycle_name>,<cell>,<data>
[:DATA]	<cycle_name>,<block>
:DEFine	<cycle_name>,(<size> | <cycle_copy>)
:DELete	
:ALL	
[:NAME]	<cycle_name>
:DIRectory?	
:PAGE	[<page>]
:SETup	
:CLOCk	<clock>
:CTIMeout	<timeout_count>
:DELay	<delay_count>
:TSINput2	<test_type>
:TEST	
:CELL?	<cycle_name>,<cell>
:COMPare	<cycle_name>,<cell>
:DELay	<cycle_name>,<cell>
:ERRor	<cycle_name>,<cell>
:LEVel	<cycle_name>,<signal>,<level>,<cell>
:RESet	<cycle_name>,<cell>
:STRobe	<cycle_name>,<level>,<cell>

3.14.1 TIMing:CELL

This command allows the user to program timing cell data.

SYNTAX

TIMing:CELL <cycle_name>,<offset>,<data>
TIMing:CELL? <cycle_name>,<offset>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<cycle_name>	Name	See section 3.1.1.2.2	Name of the timing cycle to program or query.	
<offset>	Numeric	1 to 256	Timing cell index to program data into.	
<data>	Numeric	0-8,388,607 ($0-7FFFFF_h$)	Data to program into timing cell.	

RETURNED PARAMETER SYNTAX

{<data>}

Parameter	Type	Value	Description
<data>	Numeric	0-8,388,607	Data from the selected timing cycle cell.

COMMENTS

1. Each <data> element contains data for a single cell of a timing cycle. Each cell is active for at least one period of the timing set clock (TS_CLK).
2. The contents of <data> for a single timing module, (TSB not linked to TSA) is an ASCII formatted 12 bit word defined below:

Bit #	Mnemonic	Description
0	SR_CLK	Falling edge increments the pattern address to the I/O modules.
		Rising edge samples the sequence branch condition.
1	ADEL_CLK	Falling edge latches the response pattern address when the address delay mode is enabled, see section 3.5.1

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Bit #	Mnemonic	Description
2	STIM_LOAD	Falling edge latches stimulus data into the output register and the response pattern address for non-algorithmic I/O modules.
		Active low enables data from the stimulus memory into the output register on the next TS_CLK rising edge for algorithmic I/O modules.
		Rising edge latches the response pattern address when the output register mode is enabled for algorithmic I/O modules.
3	TSENABLE1	One of two signals used to enable/disable I/O module stimulus data, low enables stimulus.
4	TSENABLE2	One of two signals used to enable/disable I/O module stimulus data, low enables stimulus.
5	TSSTROBE1	One of two signals used to strobe I/O module response data, falling edge latches response.
6	TSSTROBE2	One of two signals used to strobe I/O module response data, falling edge latches response.
7	TSOUT1	General purpose front panel output trigger/handshake signal level.
8	TSOUT2	General purpose front panel output trigger/handshake signal level.
9	TSOUT3	General purpose front panel output trigger/handshake signal level.
10	TSOUT4	General purpose front panel output trigger/handshake signal level.
		Active low enables the algorithmic function on the next TS_CLK rising edge for algorithmic I/O modules.
11	TSOUT5	General purpose front panel output trigger/handshake signal level.

3. The contents of <data> for synchronized timing modules, (TSB linked to TSA), is an ASCII formatted 24 bit word defined below:

Bit #	Mnemonic	Description
0	SR_CLK	Falling edge increments the pattern address to the I/O modules.
		Rising edge samples the sequence branch condition.
1	ADEL_CLKA	Falling edge latches the response pattern address for channels 1 to 96 when the address delay mode is enabled, see section 3.5.1
2	ADEL_CLKB	Falling edge latches the response pattern address for channels 97 to 192 when the address delay mode is enabled, see section 3.5.1
3	STIM_LOADA	Falling edge latches stimulus data into the output register and the response pattern address for non-algorithmic I/O modules for channels 1 to 96.
		Active low enables data from the stimulus memory into the output register on the next TS_CLK rising edge for algorithmic I/O modules for channels 1 to 96.
		Rising edge latches the response pattern address when the output register mode is enabled for algorithmic I/O modules for channels 1 to 96.
4	STIM_LOADB	Falling edge latches stimulus data into the output register and the response pattern address for non-algorithmic I/O modules for channels 1 to 96.
		Active low enables data from the stimulus memory into the output register on the next TS_CLK rising edge for algorithmic I/O modules for channels 1 to 96.
		Rising edge latches the response pattern address when the output register mode is enabled for algorithmic I/O modules for channels 1 to 96.
5	TSENABLE1A	One of two signals used to enable/disable I/O module stimulus data for channels 1 to 96, low enables stimulus.
6	TSENABLE2A	One of two signals used to enable/disable I/O module stimulus data for channels 1 to 96, low enables stimulus.
7	TSENABLE1B	One of two signals used to enable/disable I/O module stimulus data for channels 97 to 192, low enables stimulus.
8	TSENABLE2B	One of two signals used to enable/disable I/O module stimulus data for channels 97 to 192, low enables stimulus.
9	TSSTROBE1A	One of two signals used to strobe I/O module response data for channels 1 to 96, falling edge latches response.
10	TSSTROBE2A	One of two signals used to strobe I/O module response data for channels 1 to 96, falling edge latches response.
11	TSSTROBE1B	One of two signals used to strobe I/O module response data for channels 97 to 192, falling edge latches response.
12	TSSTROBE1B	One of two signals used to strobe I/O module response data for channels 97 to 192, falling edge latches response.
13	TSOUT1A	General purpose front panel output trigger/handshake signal level.
14	TSOUT2A	General purpose front panel output trigger/handshake signal level.
15	TSOUT3A	General purpose front panel output trigger/handshake signal level.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Bit #	Mnemonic	Description
16	TSOUT4A	General purpose front panel output trigger/handshake signal level.
		Active low enables the algorithmic function for channels 1 to 96 on the next TS_CLK rising edge for algorithmic I/O modules.
17	TSOUT5A	General purpose front panel output trigger/handshake signal level.
18	TSOUT1B	General purpose front panel output trigger/handshake signal level.
19	TSOUT2B	General purpose front panel output trigger/handshake signal level.
20	TSOUT3B	General purpose front panel output trigger/handshake signal level.
21	TSOUT4B	General purpose front panel output trigger/handshake signal level.
		Active low enables the algorithmic function for channels 97 to 192 on the next TS_CLK rising edge for algorithmic I/O modules.
22	TSOUT5B	General purpose front panel output trigger/handshake signal level.
23	NU	Not used.

4. Groups that exceed 32 channels require an additional <data> value separated by a comma for each additional 32 channels.
5. The data will be mapped to the groups channels with the least significant bit mapped to the lowest channel number. The most significant bits will be truncated if the bit stream exceeds the number of channels in the group.
6. Each timing cycle must have at least one SR_CLK pulse for proper operation.
7. If the address delay mode is enabled then each timing cycle must have at least one ADEL_CLK pulse for proper operation, see section 3.5.1.
8. If the output register mode is enabled then each timing cycle must have at least one STIM_LOAD pulse for proper operation.
9. The timing module must be reset when programming the cell memory, see section 3.4.2.

3.14.2 TIMing[:DATA]

This command is used to program the specified timing memory using definite length arbitrary block format.

SYNTAX

TIMing[:DATA] <cycle_name>,<block>
TIMing:DATA? <cycle_name>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<cycle_name>	Name	See section 3.1.1.2.2	Name of the timing cycle to program.	
<block>	Arb Block		Timing cell data.	

RETURNED PARAMETER SYNTAX

{<block>}

Parameter	Type	Value	Description
<block>	Arb Block	See section 3.1.1.2.2	Data from the selected timing cycle cell.

COMMENTS

1. For a single timing module, (TSB not linked to TSA), every cell is defined by two bytes illustrated below:
   TIMing:DEFine T1,3
   TIMing:DATA T1,#16$Byte_1Byte_2Byte_3Byte_4Byte_5Byte_6$

Cell #	Block Data															
1	$Byte_1$								$Byte_2$							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Cell #	Block Data															
2	$Byte_3$								$Byte_4$							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
3	$Byte_5$								$Byte_6$							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

The bit definition is described below:

Bit #	Mnemonic	Description
0-11	See section 3.14.1, comment 2.	
12	INSTR0	Instruction bit 0.
13	INSTR1	Instruction bit 1.
14	INSTR2	Instruction bit 2.
15	LSTBIT	Last cell flag, low = last timing cell.

2. For synchronized timing modules, (TSB linked to TSA), every cell is defined by four bytes illustrated below:
   TIMing:DEFine T1,2
   TIMing:DATA T1,#h18$Byte_1Byte_2Byte_3Byte_4Byte_5Byte_6Byte_7Byte_8$

Cell #	Block Data																											
1	$Byte_1$								$Byte_2$								$Byte_3$								$Byte_4$			
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
2	$Byte_5$								$Byte_6$								$Byte_7$								$Byte_8$			
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

The bit definition is described below:

Bit #	Mnemonic	Description
0-22	See section 3.14.1, comment 3.	
23	INSTR0	Instruction bit 0.
24	INSTR1	Instruction bit 1.
25	INSTR2	Instruction bit 2.
26	LSTBIT	Last cell flag, low = last timing cell.

3. The INSTR<n> bits define the test input parameters for the handshake control. The format of the data is described in the following the following table:

INSTR2	INSTR1	INSTR0	Test Description
0	0	0	WAIT FOR TSINP1 LOW
0	0	1	WAIT FOR TSINP1 HIGH
0	1	0	WAIT FOR TSINP2 LOW/FALLING
0	1	1	WAIT FOR TSINP2 HIGH/RISING
1	0	0	WAIT ON ERROR
1	0	1	WAIT UNTIL COMPARE
1	1	0	WAIT FOR DELAY
1	1	1	NO WAIT

4. The timing module must be reset when reading the cell memory, see section 3.4.2.

3.14.3 TIMing:DEFine

This command allocates and initializes a new timing cycle.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SYNTAX

TIMing:DEFine <cycle_name>,<size>
TIMing:DEFine <cycle_name>,<cycle_copy>
TIMing:DEFine? <cycle_name>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<cycle_name>	Name	See section 3.1.1.2.2	Name to assign to the timing cycle.	NA
<size>	Numeric	2 to 256	Number of cells.	
<cycle_copy>	Name	See section 3.1.1.2.2	Timing cycle to copy.	

RETURNED PARAMETER SYNTAX

"<cycle_name>",<size>,<offset>

Returned Parameter	Type	Value	Description
<cycle_name>	Name	See section 3.1.1.2.2	Name of the timing cycle.
<size>	Numeric	1 to 256	The number of cells.
<offset>		0 to 8191	The memory offset where the timing cycle starts.

COMMENTS

1. If the "<cycle_name>,<size>" form of the command is used then the contents of the new tables memory will be initialized to the following:
   1st Cell All bits high except SR_CLK.
   All others All bits set high.
2. If the "<cycle_name>,<cycle_copy>" form of the command is used then the size and contents of the new timing cycle memory will be copied from the specified timing cycle.
3. The timing module must be reset when programming the cell memory, see section 3.4.2.
4. The string (""",0,0) will be returned if the timing cycle is not defined.

3.14.4 TIMing:DELete

This subtree allows the operator to delete timing cycles.

3.14.4.1 TIMing:DELete:ALL

This command deletes all the defined timing cycles and initialize the default "IDLE" cycle.

SYNTAX

TIMing:DELete:ALL

COMMENTS

1. The timing module must be reset when programming the cell memory, see section 3.4.2.

3.14.4.2 TABLe:DELete[:NAME]

This command deletes the specified timing cycle.

SYNTAX

TIMing:DELete[:NAME] <cycle_name>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<cycle_name>	Name	See section 3.1.1.2.2	Selected timing cycle to delete.	NA

COMMENTS

1. The default "IDLE" timing cycle cannot be deleted.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

2. The timing module must be reset when programming the cell memory, see section 3.4.2.

3.14.5 TIMing:DIRectory?

Returns a directory listing of all the defined timing cycle names.

SYNTAX

TIMing:DIRectory?

RETURNED PARAMETER SYNTAX

{"<cycle_name>",<size>,<offset>}

Returned Parameter	Type	Value	Description
<cycle_name>	Name	See section 3.1.1.2.2	Name of the timing cycle.
<size>	Numeric	2 to 256	The number of cell.
<offset>		0 to 8191	The memory offset where the timing cycle starts.

COMMENTS

1. The string (""",0,0) will be returned if no timing cycles are defined.

3.14.6 TIMing:PAGE

This command allows the operator to select the from one of four pages. Each page contains sixteen timing cycles.

SYNTAX

TIMing:PAGE <page>
TIMing:PAGE?

PARAMETER LIST

Parameter	Type	Values	Description	Default
<page>	Numeric	1 to 4	Timing page to make active.	1

RETURNED PARAMETER SYNTAX

<page>

Returned Parameter	Type	Value	Description
<page>	Numeric	1 to 4	Current page number.

COMMENTS

1. Every timing page contains its own default "IDLE" cycle.
2. The timing module must be reset when selecting a new timing page, see section 3.4.2.

3.14.7 TIMing:SETup

The SETup subtree allows the operator to program or query the timing module parameters, clock source, delay value, timeout and input trigger two mode.

3.14.7.1 TIMing:SETup:CLOCk

This command sets the clock source for the timing module. Each timing cell is at least one period of the selected clock.

SYNTAX

TIMing:SETup:CLOCk <clock>
TIMing:SETup:CLOCk?

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

PARAMETER LIST

Parameter	Type	Values	Description	Default
<clock>	Ascii	10	Selects the internal 10MHz clock.	10
		20	Selects the internal 20MHz clock.	
		50	Selects the internal 50MHz clock.	
		EXTernal1	Selects the front panel “EXTCLK1” clock.	
		EXTernal2	Selects the front panel “EXTCLK2” clock.	
		PGMClk1	Selects the internal “PGMCLK1” clock.	
		PGMClk2	Selects the internal “PGMCLK2” clock.	

RETURNED PARAMETER SYNTAX

<clock>

Returned Parameter	Type	Value	Description
<clock>	Ascii	10	10MHz internal clock selected.
		20	20MHz internal clock selected.
		50	50MHz internal clock selected.
		EXTERNAL1	“EXTCLK1” front panel clock selected.
		EXTERNAL2	“EXTCLK2” front panel clock selected.
		PGMCLK1	“PGMCLK1” internal clock selected.
		PGMCLK2	“PGMCLK2” internal clock selected.

COMMENTS

None

3.14.7.2 TIMing:SETup:CTIMeout

This command allows the operator to specify the number of clocks before a cycle timeout is flagged while waiting for an input trigger/handshake.

SYNTAX

TIMing:SETup:CTIMeout <timeout>
TIMing:SETup:CTIMeout?

PARAMETER LIST

Parameter	Type	Values	Description	Default
<timeout>	Numeric	0 to 32768	Timeout count.	0

RETURNED PARAMETER SYNTAX

<timeout>

Returned Parameter	Type	Value	Description
<timeout>	Numeric	0 to 32768	Current cycle timeout setting.

COMMENTS

1. Timeouts are signaled to the timing module status (section 3.7.3) and the instrument summary registers (3.11.1.4.4).
2. A cycle timeout indicates that a timing set test condition never occurred within the specified number of clocks.
3. A timeout value of zero disables the cycle timeout logic.
4. The maximum timeout count for the SR100 timing module is 128.

3.14.7.3 TIMing:SETup:DELay

This command allows the operator to program the number of TS_CLKs that each test delay test cell will take for the specified cycle.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SYNTAX

TIMing:SETup:DELay <delay>
TIMing:SETup:DELay?

PARAMETER LIST

Parameter	Type	Values	Description	Default
<delay>	Numeric	0 to 32768	Delay count.	0

RETURNED PARAMETER SYNTAX

<delay>

Returned Parameter	Type	Value	Description
<delay>	Numeric	0 to 32768	Current delay count setting.

COMMENTS

1. A delay count of zero disables the delay logic.
2. The maximum delay count for the SR100 timing module is 128.

3.14.7.4 TIMing:SETup:TSINput2

This command allows the user to set the input trigger two mode as a level test or an edge test input.

SYNTAX

TIMing:SETup:TSINput2 <mode>
TIMing:SETup:TSINput2?

PARAMETER LIST

Parameter	Type	Values	Description	Default
<mode>	Ascii	LEVel	Selects the level test mode.	LEV
		EDGE	Selects the edge test mode.	

RETURNED PARAMETER SYNTAX

<mode>

Returned Parameter	Type	Value	Description
<mode>	Ascii	LEV	Level test mode selected.
		EDGE	Edge test mode selected.

COMMENTS

None

3.14.8 TIMing:TEST

The TEST subtree allows the operator to program or query test cell into the timing cycle memory.

3.14.8.1 TIMing:TEST:CELL

This command queries the timing memory for the programmed test condition.

SYNTAX

TIMing:TEST:CELL? <cycle_name>,<cell>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<cycle_name>	Name	See section 3.1.1.2.2	Selected timing cycle to query.	
<cell>	Numeric	1 to 256	Cell number to query.	

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

RETURNED PARAMETER SYNTAX

<condition>[,<level>]

Parameter	Type	Value	Description
<condition>	Ascii	RES	No test programmed.
		TSIN1	Input trigger one test for <level>.
		TSIN2	Input trigger two test for <level>.
		ERR	Test for I/O module real time error.
		COMP	Test for I/O module real time compare.
		DEL	Delay cell programmed.
<level>	Ascii	LOW	Test for low level or edge on the specified input trigger.
		HIGH	Test for high level or edge on the specified input trigger.

COMMENTS

1. The timing module must be reset when reading the timing memory, see section 3.4.2.

3.14.8.2 TIMing:TEST:DELay

This command programs a test delay into the timing memory of the specified timing cycle and cell.

SYNTAX

TIMing:TEST:DELay <cycle_name>,<cell>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<cycle_name>	Name	See section 3.1.1.2.2	Selected timing cycle to program.	
<cell>	Numeric	1 to 256	Cell number to insert test.	

COMMENTS

1. The value of the delay is programmed using the “TIMing:SETup:DELay” command.
2. The timing module must be reset when programming the timing memory, see section 3.4.2.

3.14.8.3 TIMing:TEST:LEVel

This command programs a handshake/input trigger level test into the timing memory.

SYNTAX

TIMing:TEST:LEVel <cycle_name>,<signal>,<level>,<cell>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<cycle_name>	Name	See section 3.1.1.2.2	Selected timing cycle to program.	
<signal>	Ascii	TSINput1	Select front panel “TSINPUT1” to test.	
		TSINput2	Select front panel “TSINPUT2” to test.	
<level>	Ascii	HIGH	Test selected signal and continue if high.	
		LOW	Test selected signal and continue if low.	
<cell>	Numeric	1 to 256	Cell number to insert delay.	

COMMENTS

1. If “TSINPUT2” is selected then the input trigger two mode must be set to LEVel, see section 3.14.7.4.
2. The timing module must be reset when programming the timing memory, see section 3.4.2.

3.14.8.4 TIMing:TEST:STRobe

This command allows the user to program a input trigger/handshake edge transition test into the timing memory.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SYNTAX

TIMing:TEST:STRobe <cycle_name>,<level>,<cell>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<cycle_name>	Name	See section 3.1.1.2.2	Selected timing cycle to program.	
<level>	Ascii	HIGH	Test selected signal and continue if low to high transition.	
		LOW	Test selected signal and continue if high to low transition.	
<cell>	Numeric	1 to 255	Cell number to insert test.	

COMMENTS

1. Only the front panel “TSINPUT2” signal can be tested for edge transitions.
2. The input trigger two mode must be set to EDGE, see section 3.14.7.4.
3. The edge test circuitry is reset during the last cell of a timing set and the cell after a valid test edge is detected, therefore the following rules apply to programming edge tests:
   A. An edge test in the last cell is not allowed.
   B. Edge tests of the same level are not allowed in consecutive cells.

3.14.8.5 TIMing:TEST:ERRor

This command programs a test for non-compare or error into the timing memory.

SYNTAX

TIMing:TEST:ERRor <cycle_name>,<cell>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<cycle_name>	Name	See section 3.1.1.2.2	Selected timing cycle to program.	
<cell>	Numeric	1 to 256	Cell number to insert test.	

COMMENTS

1. The error signal that is tested is generated from the I/O modules and is valid only after the input strobe has occurred (i.e, the test for error should occur after the input strobe).
2. If the cycle timeout is set to zero then when an error is detected the timing cycle will halt until re-set, see section 3.4.2.

3.14.8.6 TIMing:TEST:COMPare

This command programs a test for non-compare or error into the timing memory.

SYNTAX

TIMing:TEST:COMPare <cycle_name>,<cell>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<cycle_name>	Name	See section 3.1.1.2.2	Selected timing cycle to program.	
<cell>	Numeric	1 to 256	Cell number to insert test.	

COMMENTS

1. The compare signal that is tested is generated from the I/O modules and does not require a strobe.
2. The compare test is not affected by the cycle timeout logic.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.14.8.7 TIMing:TEST:RESet

This command removes a previously defined test cell from the timing memory.

SYNTAX

TIMing:TEST:RESet <cycle_name>,<cell>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<cycle_name>	Name	See section 3.1.1.2.2	Selected timing cycle to program.	
<cell>	Numeric	1 to 256	Cell number to reset test.	

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.15 IEEE 488.2 Common Commands

The SR192 responds to the following Common commands (for further information refer to the IEEE 488.2-1987 specification.).

3.15.1 *CLS (Clear Status)

The clear status command clears status data structures and sets all pending operation flags false.

SYNTAX

*CLS

COMMENTS

1. IEEE 488.2-1987 section 10.3.

3.15.2 *ESE (Event Status Enable)

The event status enable command sets the event status enable register mask enabling the operator to select specific bits to be passed through to the Event Status summary Bit (ESB) of the Status Register. Refer to section 3.11 for a description of the status reporting structures of the SR192.

SYNTAX

*ESE <mask>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<mask>	Numeric	0 to 255 (0_h-FF_h)	A one in each bit position enables the corresponding bit in the event status register to generate the summary bit in the status register.	0

COMMENTS

1. IEEE 488.2-1987 section 10.10.

3.15.3 *ESE? (Event Status Enable Query)

The event status enable query allows the programmer to determine the current contents of the event status enable register.

SYNTAX

*ESE?

RETURNED PARAMETER SYNTAX

<mask>

Returned Parameter	Type	Value	Description
<mask>	Numeric	0 to 255	Event status enable register contents.

COMMENTS

1. IEEE 488.2-1987 section 10.11.

3.15.4 *ESR? (Event Status Register Query)

The event status register query allows the programmer to determine the current contents of the event status register.

RETURNED PARAMETER SYNTAX

<event>

Parameter	Type	Value	Description
<event>	Numeric	0 to 255 (0_h-FF_h)	Event status register.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

COMMENTS

1. The following table describes the contents of the event status register:

Event Status Register Bit Definition							
7	6	5	4	3	2	1	0
PON	URQ	CME	EXE	DDE	QYE	RQC	OPC

Event Status Bit Description

OPC	Operate Complete; Set by the "*OPC" command when the pending operation flag is false. The pending operation flag is set when a timing module sequence is executed and is reset when the sequence is done.
RQC	Request Control; Not used by the SR192.
QYE	Query Error; Set when a query error occurs, i.e., Attempt to read an empty output buffer or sending another command before responding to a previous query command.
DDE	Device Dependent Error; Set when an SR192 specific error occurs, i.e., memory error.
EXE	Execution Error; Set when the SR192 could not execute a valid command, i.e., settings conflict.
CME	Command Error; Set when the SR192 command parser detects and error, i.e., syntax error.
URQ	User Request; Not used by the SR192.
PON	Power On, Not used by the SR192.

2. Reading the Standard Event Status Register clears it.
3. IEEE 488.2-1987 section 10.12.

3.15.5 *IDN? (Identification Query)

Returns the SR192 instrument identification string.

SYNTAX

*IDN?

RETURNED PARAMETER SYNTAX

TALON INSTRUMENTS,SR192,0,<firmware>

Returned Parameter	Type	Value	Description
<firmware>	Numeric	0.00 to 9.99	Current SR192 firmware revision.

COMMENTS

1. IEEE 488.2-1987 section 10.14.

3.15.6 *OPC (Operation Complete Command)

The operation complete command sets the "OPC" bit in the event status register when the pending operation flag is false.

SYNTAX

*OPC

COMMENTS

1. The pending operation flag is set true when a timing module sequence is executed and is set false when the sequence is done.
2. Either TSA or TSB must be selected prior to sending the "*OPC" command.
3. IEEE 488.2-1987 section 10.18.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.15.7 *OPC? (Operation Complete Query)

The operation complete query places an ASCII character 1 into the device's Output Queue when all pending selected device operations have been finished.

SYNTAX

*OPC?

RETURNED PARAMETER SYNTAX

1

COMMENTS

1. The pending operation flag is set true when a timing module sequence is executed and is set false when the sequence is done.
2. Either TSA or TSB must be selected prior to sending the "*OPC?" command.
3. IEEE 488.2-1987 section 10.18.

3.15.8 *RST (Reset Command)

The reset command performs a device reset.

SYNTAX

*RST

COMMENTS

1. The RST command does the following:
   A. Sets the device-specific functions to a known state.
   B. Sets pending operation flag for both timing modules false.
   C. Clears any data from the output buffer.
2. IEEE 488.2-1987 section 10.32.

3.15.9 *SRE (Service Request Enable Command)

The service request enable command sets the service request enable register mask bits permitting the programmer to select which messages in the status byte register may cause service requests (VXI interrupt).

SYNTAX

*SRE <mask>

PARAMETER LIST

Parameter	Type	Values	Description	Default
<mask>	Numeric	0 to 255 (0_h-FF_h)	A one in each bit position enables the corresponding bit in the status byte register to generate a VXI backplane interrupt.	0

COMMENTS

1. IEEE 488.2-1987 section 10.34.

3.15.10 *SRE? (Service Request Enable Query)

The service request enable query allows the programmer to determine the current contents of the service request enable register.

SYNTAX

*SRE?

RETURNED PARAMETER SYNTAX

<mask>

Returned Parameter	Type	Value	Description
<mask>	Numeric	0 to 255	Service request enable register contents.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

COMMENTS

1. IEEE 488.2-1987 section 10.35.

The response will be the enable value from 0 through 63 or 128 through 191.

3.15.11 *STB? - Read Status Byte Query (section 10.36)

The status byte query allows the programmer to read the contents of the status byte.

SYNTAX

*STB?

RETURNED PARAMETER SYNTAX

<status>

Parameter	Type	Value	Description
<status>	Numeric	0 to 255 (0_h-FF_h)	Status byte register.

COMMENTS

1. The following table describes the contents of the event status register:

Event Status Register Bit Definition							
7	6	5	4	3	2	1	0
OSB	rsv	ESB	MAV	QSB	HLT	BSY	NU

Status Byte Bit Description

NU	Not Used.
BSY	Busy; Set true when either SR192 timing module is running.
HLT	Halt; Set true when either SR192 timing module has halted on an I/O module error.
QSB	Questionable Summary Bit; Not used by the SR192.
MAV	Message available; Set when the SR192 has data in its output buffer.
ESB	Event Summary Bit; Set when any of the masked bits in the event status register are set.
rsv	Request service; Set when any of the masked bits in the status byte register are set. Causes a VXI backplane interrupt.
OSB	Operation Summary Bit; Set when any of the masked bits in the operation event register are set.

2. IEEE 488.2-1987 section 10.36.

3.15.12 *TST? (Self Test Query)

The self test query causes an internal self-test and places a response into the Output Queue indicating whether or not the device completed the test without any detected errors. This query should be the first command after power-up.

SYNTAX

*TST?

RETURNED PARAMETER SYNTAX

<result>

Parameter	Type	Value	Description
<result>	Numeric	0 to 255 (0_h-FF_h)	Self test result data.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

COMMENTS

1. The following table describes the contents of the returned result.

Self Test Result Bit Definition, 0 = Pass															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	DRB6	DRB5	DRB4	DRB3	DRB2	DRB1	DRA6	DRA5	DRA4	DRA3	DRA2	DRA1	TSB	TSA	DAC

2. Device settings return to power up defaults.
3. IEEE 488.2-1987 section 10.38.

3.15.13 *WAI (Wait to Continue Command) (section 10.39)

The wait to continue command will prevent the device from executing any further commands or queries until the pending operation flag is set false.

SYNTAX

*WAI

COMMENTS

1. The pending operation flag is set true when a timing module sequence is executed and is set false when the sequence is done.
2. Either TSA or TSB must be selected prior to sending the “*WAI” command.
3. IEEE 488.2-1987 section 10.32.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.16 VXI WS Interface Commands

The SR192 responds to the following low level VXIbus word serial interface commands. These are issued by its commander. Refer to the VXIbus specification for further information.

3.16.1 ANOP - Abort Normal Operation

The Abort Normal Operation command is used to cause a device to cease normal operation. Upon receipt of this command the SR192 returns to its default configuration, aborting all operations. The aborted state is defined as follows: Pending interrupts are unasserted. No interrupts may be asserted. The device is in a generally inactive state and is ready to accept commands.

The syntax of the Abort Normal Operation command is defined in the following table.

ANOP Command															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	1	0	0	0	1	1	1	1	1	1	1	1

When the abort operation has completed (the SR192 is in the aborted state), response data is placed in the Data Low register in the following format:

ANOP Command Response															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

3.16.2 AMC - Asynchronous Mode Control

The Asynchronous Mode Control command is used by a Commander to direct the path of events and responses. The syntax of this command is defined in the following table.

AMC Command															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	0	1	0	0	0	X				Resp. En	Event En	Resp. Mode	Event Mode

The bits have the following meanings:

X (Don't care):	Don't care. The value written to this bit has no effect.
Resp. En:	A zero (0) enables generation of responses. A one (1) disables generation of responses.
Event En:	A zero (0) enables generation of events. A one (1) disables generation of events.
Resp. Mode:	A one (1) indicates that responses should be sent as signals. A zero (0) indicates that responses should be sent as interrupts. This bit is meaningful only if the Resp En* bit is zero (0).
Event Mode:	A one (1) indicates that events should be sent as signals. A zero (0) indicates that events should be sent as interrupts. This bit is meaningful only if the Event En* bit is zero (0).

The result data is placed in the Data Low register with the following format. This result is a confirmation/denial of the command:

AMC Command Response															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Status				1	1	1	1	1	1	1	1	Resp. En	Event En	Resp. Mode	Event Mode

The bits have the following meanings:

Status: This field indicates the execution status of the command. It may have the following values:

F_{16}:	The command successfully completed as expected.
7_{16}:	Command failed – A requested option is not supported.

Resp. En: A zero (0) indicates that the generation of responses is enabled. A one (1) indicates that the generation of responses is disabled.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Event En: A zero (0) indicates that the generation of events is enabled. A one (1) indicates that the generation of events is disabled.

Resp. Mode: A one (1) indicates that responses are being sent as signals. A zero (0) indicates that responses are being sent as interrupts. This bit is meaningful only if the Resp En bit is zero (0).

Event Mode: A one (1) indicates that events are being sent as signals. A zero (0) indicates that events are being sent as interrupts. This bit is meaningful only if the Event En bit is zero (0).

3.16.3 BNOP - Begin Normal Operation

The Begin Normal Operation command notifies the SR192 that it can begin normal operation now. A Commander may not initiate any VMEbus operations after power-up until it receives the Begin Normal Operation command. It then sends the Begin Normal Operation command to all of its current Message Based Servants and begins normal execution of commands and/or application programs. The Top_Level field of the Begin Normal Operation command is provided to inform a device whether or not it is a top level Commander. The syntax of this command is defined in the following table.

BNOP Command															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	0	Top Level	1	1	1	1	1	1	1	1

The bits have the following meanings:

Top Level: A one (1) in this field indicates that the device is a top level Commander. A zero (0) indicates that it is a Servant to another device.

When the begin operation has completed, response data is placed in the Data Low register in the following format:

BNOP Command Response															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Status				State				Logical Address							

The bits have the following meanings:

Status: This field indicates the execution status of the command. It may have the following values:

F_{16}: The Begin Normal Operation command has been successfully executed. The value FE16 is reported in the Logical Address field.

7_{16}: The Commander device is not able to command a Servant. The Servant's Logical Address is reported in the Logical Address field.

6_{16}: The Commander device is not able to configure a Servent device. The Servants Logical Address is reported in the Logical Address field.

5_{16}: The Commander device timed out when waiting for response from a Servant device. The Servant's Logical Address is reported in the Logical Address field.

4_{16}: The device is not able to successfully initialize itself. The value FE_{16} is reported in the Logical Address field.

3_{16}: The device is not able to be a top level Commander. The value FE_{16} is reported in the Logical Address field.

2_{16}: The device must be a top level Commander. The value FE16 is reported in the Logical Address field.

1_{16}: An undefined error was caused. The value FE_{16} is reported in the Logical Address field.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

State: This field indicates the state of this device and its sub-tree. It may have the following values:

- 3_{16}: This device and all of its Servant tree are in the CONFIGURE sub-state.
- 7_{16}: This device is in the CONFIGURE sub-state, however al least one device in its Servant tree is in the NORMAL OPERATION sub-state.
- B_{16}: This device is in the NORMAL OPERATION sub-state, however at least one device in its Servant tree is in the CONFIGURE sub-state.
- F_{16}: This device and all of its Servant tree are I the NORMAL OPERATION sub-state.

Logical Address: This field contains the Logical Address corresponding to the status field values.

3.16.4 BAV - Byte Available

The Byte Available command is used by a Commander to send a byte of data to a Servant device. The END field identifies that this is the last byte of the message. The syntax of this command is defined in the following table.

BAV Command															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	1	1	1	END	Datum							

3.16.5 BRQ - Byte Request

The Byte Request command is used by a Commander to read a byte of data from a Servant device. The syntax of this command is defined in the following table.

BRQ Command															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	1	1	1	1	0	1	1	1	1	1	1	1	1

The result data is placed in the Data Low register with the following format. The END field is used to indicate the last byte of the message.

BRQ Command Response															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	END	Datum							

3.16.6 CEV - Control Event

The Control Event command is used by a Commander to selectively enable the generation of events by a Servant. A one (1) in the enable field enables the generation of the specific event. A zero (0) in the enable field disables the generation of the specific event. The syntax of this command is defined in the following table.

CEV Command															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	0	1	1	1	1	Enable	Event						

Event: These bits (6-0) are the identifying bits of the event being enabled/disabled.

- $7D_{16}$: Request True; Sent when the SR192 request service bit (rsv) in the status byte is set true.
- $7C_{16}$: Request False; Sent when the SR192 request service bit (rsv) in the status byte is set false.

The device returns the following data in the Data Low register:

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

CEV Command Response															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Status				1	1	1	1	1	1	1	1	1	1	1	0

The bits have the following meanings:

Status: This field indicates the execution status of the command. It may have the following values:

F_{16}: The command successfully completed.

7_{16}: Command failed – The event referenced is not generated by this device.

3.16.7 CLE - Clear

The Clear command is used by a Commander to caused a Servant device to clear the VXIbus interface and any pending operations. Any initiated operations in the receiving device are undisturbed. The syntax of this command is defined in the following table.

CLE Command															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

3.16.8 ENOP - End Normal Operation

The End Normal Operation command is used to cause a Commander to cease normal operation in an orderly manner. Upon receipt of this command the Commander issues the End Normal Operation command to all of its Message Based Servants. The Commander is responsible for halting all its Servants. The “ended” state is defined as follows: All VMEbus Master activity is halted. Pending interrupts are unasserted. The device is in a generally inactive state and is ready to accept commands.

The syntax of the End Normal Operation command is defined in the following table.

ENOP Command															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	1	0	0	1	1	1	1	1	1	1	1	1

Upon completion of the ENOP command, the SR192 places the following result in the data low register:

ENOP Command Response															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Status				State				Logical Address							

The bits have the following meanings:

Status: This field indicates the execution status of the command. It may have the following values:

F_{16}: The End Normal Operation command has been successfully executed. The value FE16 is reported in the Logical Address field.

7_{16}: The device was already in the CONFIGURE sub-state. The value FE16 is reported in the Logical Address field.

6_{16}: The Commander device timed out when waiting for the response from a Servant device. The Servants Logical Address is reported in the Logical Address field.

5_{16}: The device is not able to end its operation I a consistent manner. The value FE16 is reported in the Logical Address field.

4_{16}: The Commander device is not able to deactivate a Servant. The Servants Logical Address is reported in the Logical Address field.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3_{16}: An undefined error was caused. The value FE16 is reported in the Logical Address field.

State: This field indicates the state of this device and its sub-tree. It may have the following values:

F16: This device and all of its Servant tree are in the CONFIGURE sub-state.

B_{16}: This device is in the CONFIGURE sub-state, however at least one device in its Servant tree is in the NORMAL OPERATION sub-state.

9_{16}: This device is in the CONFIGURE sub-state, however the sub-states of the devices in its Servant tree are unknown.

7_{16}: This device is in the NORMAL OPERATION sub-state, however at least one device in its Servant tree is in the CONFIGURE sub-state.

3_{16}: This device and all of its Servant tree are in the NORMAL OPERATION sub-state.

Logical Address: This field contains the Logical Address corresponding to the status field values. The SR192 ends its normal operation in an orderly manner without any time limit constraints. Incoming signals are no longer processed. The ready bit of the status register is reset to 0. The current SR192 configuration is maintained.

3.16.9 RPR - Read Protocol

The VXIbus commander uses this command to find out what protocols, in addition to the Word Serial protocol, that the SR192 supports. The SR192 supports I4.

RPR Command															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1

Upon completion of the RPR command, the SR192 places the following result in the data low register:

RPR Command Response															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	RSVD	RG	EG	0	PI	PH	TRG	I4	I	ELW	LW

3.16.10 RPE - Read Protocol Error

The VXIbus commander uses this command to query the cause of the last protocol error.

RPE Command															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	1	1	0	1	1	1	1	1	1	1	1	1

Upon completion of the RPE command, the SR192 places the following result in the data low register:

RPE Command Response															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	1	1	1	1	1	1	Error Code		

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.16.11 RSTB - Read Status Byte

This command is used by the commander to read the status word from a servant device.

RSTB Command															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1

Upon completion of the RSTB command, the SR192 places the following result in the data low register:

RSTB Command Response															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	1	Status Byte							

3.16.12 PFSH - Program Flash

This command re-programs the contents of the flash ram with the contents of the static ram.

PFSH Command															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Upon completion of the PFSH command, the SR192 places the following result in the data low register:

PFSH Command Response															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Result															

Result:

8000_{16} Program Flash Command Complete.
0001_{16} Program Flash Failed; New System Not Loaded.
0003_{16} Program Flash Failed; New System Checksum Error.
0008_{16} Program Flash Failed; Vpp low at chip.
0010_{16} Program Flash Failed; Word program error.
0020_{16} Program Flash Failed; Block erase error.
0040_{16} Program Flash Failed; Erase suspended.

3.16.13 FTST - Full Ram Test

This command initiates a full ram test on the I/O modules. If the test passes a hex 8000 code will be returned, otherwise a 1 will be returned. A module that does not pass the FTST command will set bit 0 of its module status to a zero. Each module take approximately 3 to 7 seconds for a full ram test depending on the specific module.

FTST Command															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Upon completion of the FTST command, the SR192 places the following result in the data low register:

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

FTST Command Response															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Result															

Result:

8000_{16} Full Ram Test Passed.
0001_{16} Full Ram Test Failed, Query the module status (section 3.7.3) for module pass/fail.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Appendix A Command Summary

KEYWORD	PARAMETER FORM
CALCulate	
:CRC?	<table_name>,<group_name>,<seed>[,<mask>]
:EMEMory	
:ADDRess?	<numeric>
:COUNt?	
:PCRC?	<table_name>
CHANnel	
:MODE	<group_name>,<mode>
:REFerence	<group_name>,<reference>
EXECute	
:FIELd	<group_name>,{<numeric>}
:MODE	<mode>
:SEQuence	[<seq_name>] [<address>,<cycle_name>] [{<cycle_name>,<table_name>}]
[:TIMing]	[<cycle_name>,<table_name>] [<cycle_name>,<address>,<size>]
INPut	
:ADELay	<boolean>
:MSTRobe	<group_name>,<multiplex_strobe>
:PROBe	
:MODE	<mode>
:ORDelay	<ord_state>
:REFerence	
:HIGH	
[:DATA]	<vih_value>
[:LOGic]	<logic>
:LOW	
[:DATA]	<vil_value>
:STATus?	
:STROBe	
:DELay	<delay>
[:SOURce]	<probe_strobe>
:SWITch	<switch>,<function>
:VERSion?	
:REFerence	
:AUTO	<boolean>
:HIGH	
[:DATA]	<vih_value>
:INCRement	<increment_count>
:LOW	
[:DATA]	<vil_value>
:INCRement	<increment_count>
:SELect	[<voltasge_group>]
:UPDate	
:STRobe	
:DELay	<group_name>,<delay>
[:SOURce]	<group_name>[,<input_strobe>]
MEASure	

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
    :VOLTage?
MODule
    :LINK                   <link_state>
    [:SELect]               <module>
    :STATus?
OUTPut
    :CHANnel
        :AUTO               <boolean>
        [:STATe]            <boolean>
    :CLK<a>
        :DELay              <delay>
        :REFerence
            :HIGH
                [:DATA]     <clk_voh>
            :LOW
                [:DATA]     <clk_vol>
        [:STATe]            <clk_state>
    :ENABle
        :DELay              <group_name>,<delay>
        [:SOURce]           <group_name>,<output_enable>
    :MASTer                 <boolean>
    :PGMClk<n>
        [:DATA]             <frequency>
        :REFerence          <source>
        :RESet
    :REFerence
        :AUTO               <boolean>
        :HIGH
            [:DATA]         <voh>
            :INCRement      <count>
        :LOW
            [:DATA]         <vol>
            :INCRement      <count>
        :ISR
            [:DATA]         <isr>
            :INCRement      <increment_count>
        :SELect             [<voltage_group>]
        :UPDate
    :REGister
        :SOURce             <group_name>,<output_strobe>
        [:STATe]            <group_name>,<boolean>
    :RESet                  <mode>[,<duration>]
    :SYNC
        [:STATe]            <boolean>
        :POSition           <table_name>,<word_number>
    :TIMing
        [:STATe]            <boolean>
    :TTLTrg<n>
        [:STATe]            <boolean>
ROUTe
    :PATH
        :CATalog?
        :DEFine             <group_name>,<channel_list>
        :DELete
            [:NAME]         <group_name>
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
            :ALL
SEQuence
    :BRANch?            <seq_id>
    :DEFine             <seq_name>,{<cycle_name>,<table_name>}
                        <seq_name>,{<cycle_name>,<table_name>,<loop>}
                        <seq_name>,<size>[,<cycle_name>]
    :DELete
        [:NAME]         <seq_name>
        :ALL
    :DIRectory?
    :GOSub              <from>,<to>[,<condition>]
    :INITialize         <seq_addr>,<size>[,<cycle_name>]
    :JUMP               <from>,<to>[,<condition>]
    :LOOP               <sub_seq>,<loop>
    :RESet              <seq_def>
    :STOP               <seq_def>,<boolean>
    :TABLe              <seq_def>,<table_name>
                        <seq_def>,<table_addr>
    :TIMing             <seq_def>,<next_cycle_name>[,<branch_cycle_name>]
TABLe
    [:DATA]             <table_name>,<block>
    :DEFine             <table_name>,(<size> | <table_copy>
    :DELete
        :ALL
        [:NAME]         <table_name>
    :DIRectory?
    :FREE?
    :JENable            <table_name>, (ALLï NONEú<offset>,<boolean>)
    :MEMory
        :DATA           <table_name>,<group_name>,<block>
        :FILL           <table_name>,<group_name>,<pattern>,<offset>[,<numeric>]
        :WORD           <table_name>,<group_name>,<offset>,<data>{,<data>}
    :SELect             <memory>
TIMing
    :CELL               <cycle_name>,<cell>,<data>
    [:DATA]             <cycle_name>,<block>
    :DEFine             <cycle_name>,(<size> | <cycle_copy>)
    :DELete
         :ALL
        [:NAME]         <cycle_name>
    :DIRectory?
    :PAGE               [<page>]
    :SETup
        :CLOCk          <clock>
        :CTIMeout       <timeout_count>
        :DELay          <delay_count>
        :TSINput2       <test_type>
    :TEST
        :CELL?          <cycle_name>,<cell>
        :COMPare        <cycle_name>,<cell>
        :DELay          <cycle_name>,<cell>
        :ERRor          <cycle_name>,<cell>
        :LEVel          <cycle_name>,<signal>,<level>,<cell>
        :RESet          <cycle_name>,<cell>
        :STRobe         <cycle_name>,<level>,<cell>
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

1 SCPI MANDATED COMMANDS

KEYWORD	PARAMETER FORM
STATus	
:OPERation	
[:EVENt]?	
:CONDition?	
:ENABle	<mask_value>
:INSTrument	
[:EVENt]?	
:CONDition?	
:ENABle	<mask_value>
:ISUMmary<n>	
[:EVENt]?	
:CONDition?	
:ENABle	<mask_value>
:QUEStionable	
[:EVENt]	
:CONDition?	
:ENABle	<mask_value>
:PRESet	
SYSTem	
:ERRor?	
:VERSion?	

2 IEEE COMMON COMMANDS

*CLS	Clear Status Command
*ESE	Event Status Enable Command
*ESE?	Event Status Enable Query
*ESR?	Event Status Register Query
*IDN?	Identification Query
*OPC	Operation Complete Command
*OPC?	Operation Complete Query
*RST	Reset Command
*SRE	Service Request Enable Command
*SRE?	Service Request Enable Query
*STB?	Status Byte Query
*TST?	Self Test Query
*WAI	Wait to Continue Command

3 LOW LEVEL WS INTERFACE COMMANDS

AMC	Asynchronous Mode Control
ANO	Abort Normal Operation
BNO	Begin Normal Operation
BAV	Byte Available
BRQ	Byte Request
CEV	Control Event
CLE	Clear
ENO	End Normal Operation
RPER	Read Protocol Error
RPR	Read Protocol
RSTB	Read Status Byte
PFSH	Program Flash
FTST	Full RAM Test

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Appendix B Guide to Bus Emulation

1 Introduction

Bus Emulation is a technology which enables an engineer to simulate all the signal characteristics of a given interface. Programmable bus emulation allows the user to software configure the programmable bus emulator to simulate literally an infinite number of digital interfaces.

The term 'bus emulation' generally implies microprocessor and/or bus structured interfaces. However, to infer that Talon's programmable bus emulator simulates only these categories of interfaces would be a substantial understatement. While the programmable bus emulator is ideally suited to simulate a VME bus, Multibus, SCSI bus, 80486, or 68030 bus structured interface, the emulator can also simulate almost any digital interface; interfaces such as serial communication interfaces, state sequencer interfaces or any user defined digital interface. Indeed, most digital interfaces can be considered to be a subset of a bus structured interface.

2 Word Generator Overview

Since "bus emulator" is a relatively new technology, this appendix will first describe the basic concept of a word generator. This will establish a baseline for the bus emulation description. The reader should note that the SR192 is capable of executing all the concepts which are described in this document.

A basic digital word or pattern generator is composed of a number of I/O channels where the group of channels define a single "word". Each channel is further described by a serial data stream. When the Talon SR192 is looked at from this viewpoint, its word generator capacity is 192 I/O channels with a 128K word depth, see figure B-1.

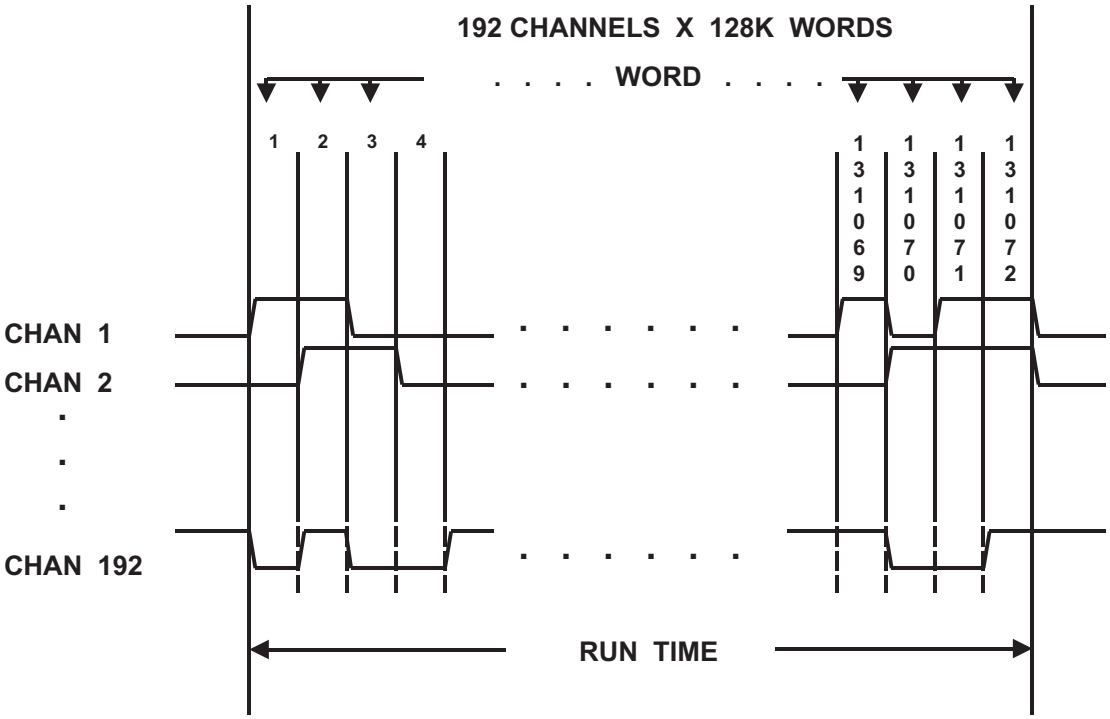


Figure B-1 192 Channels by 128K Words

In a run mode, the data would be output across the channels starting with word 1 and continuing through word 128K, generating a continuous stream of data. The time required to execute the entire table is referred to as RUN TIME.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

For illustrations' sake we will consolidate figure B-1 into the abbreviated figure B-2 below.

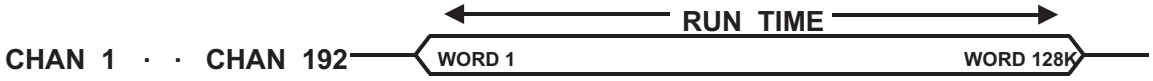


Figure B-2 192 Channels by 128K Consolidated

2.1 Tables

When using a word generator, it is desirable to subdivide the entire word generator memory into smaller sections, each section dedicated to a specific UUT operation. The SR192 refers to the divided memory as tables, see figure B-3. The SR192 can be divided into multiple tables. The table length can be individually configured from 1 word to the full 128K memory depth. The RUN TIME is the time required to execute all the tables called a sequence. The execution of the tables is continuous with zero dead time between tables.



Figure B-3 Word Generator Tables

2.2 Table Looping

The ability to loop a sequence of tables is often desirable in order to create a repeating stimulus to the UUT. The SR192 allows a sequence of tables to be looped from 1 to 64K times, see figure B-4. In addition, tables can be looped continuously. A command from the VXI Slot 0 controller is required to exit the loop. Again, the RUN TIME is the amount of time to complete the entire sequence of tables.

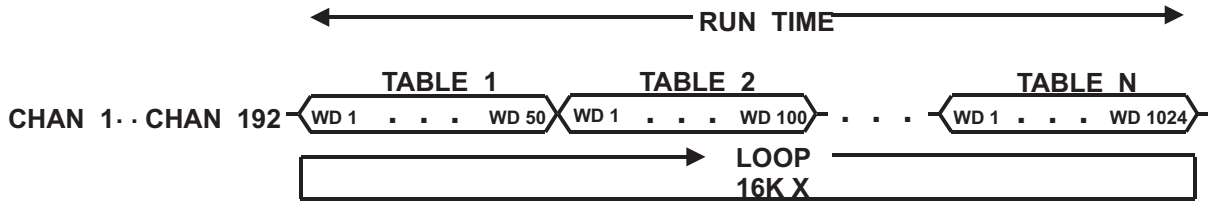


Figure B-4 Table Looping

2.3 Idle Cycle

The SR192 incorporates a unique function which is not found on present data generators, this feature being the idle cycle. The idle cycle allows the user to define a timing and data sequence prior to and after the "run time", see figure C-5. This idle cycle can be programmed such that no signal activity is seen by the UUT, or a complete repetitive timing cycle with associated data patterns could be defined.

One of the key functions of the idle cycle is the ability of the VXI controller to download new data tables, loop count values, and sequence of operation while the idle cycle is running. This feature enables a UUT to continue to receive required timing signals while new data is defined for the next operation.

3 BUS EMULATION OVERVIEW

In general, word generator parameters describe the sequencing of data from one word to the next, and

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

from one table of data to the next table. Bus emulation describes what happens “inside” each word. For example, let’s take a look at what’s happening inside word 2 of table 1, see figure B-7.

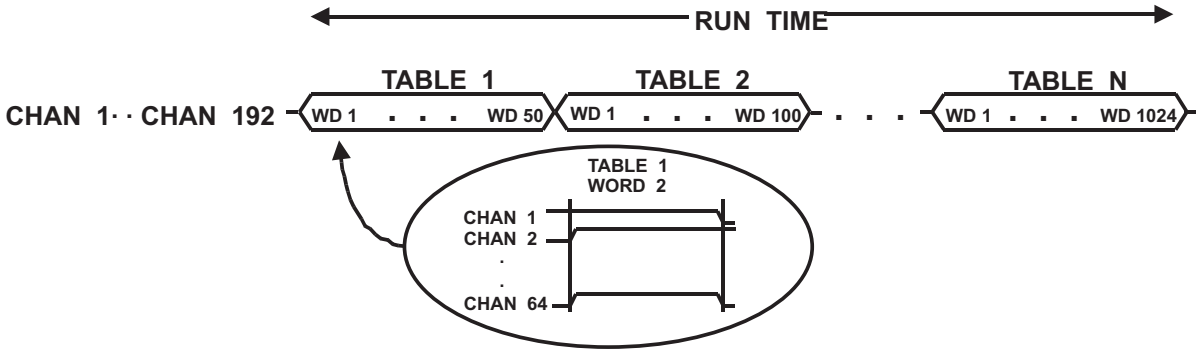


Figure B-7 Inside Word Transfers

The SR192 allows an individual word to be divided into timing increments with resolution of 20 ns per increment. Figure B-5 depicts word 2 of table 1, where word 2 has a total period of 200ns.

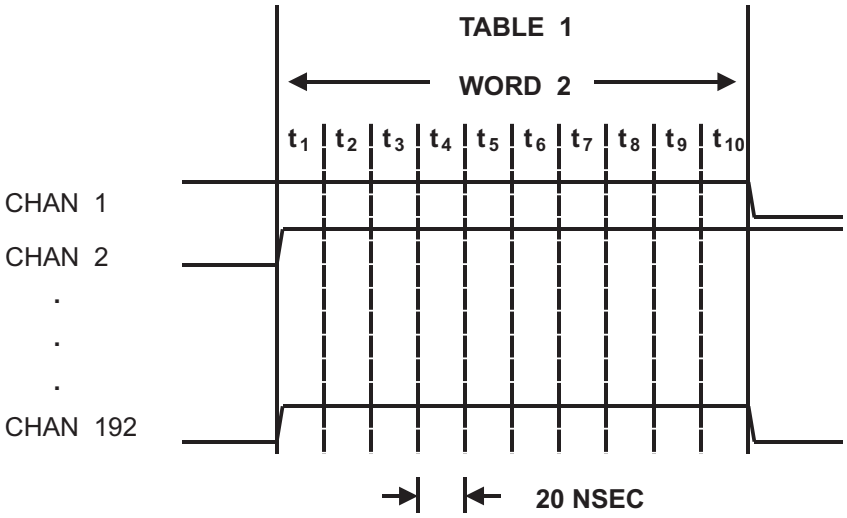


Figure B-5 Word Timing Slices

3.1 Fields

Figure B-6 again depicts word 2 of table 1. However, in this figure we will assign 64 of the 192 channels into two fields, an address field and a data field. This configuration will allow our example to follow a more traditional bus structure interface.

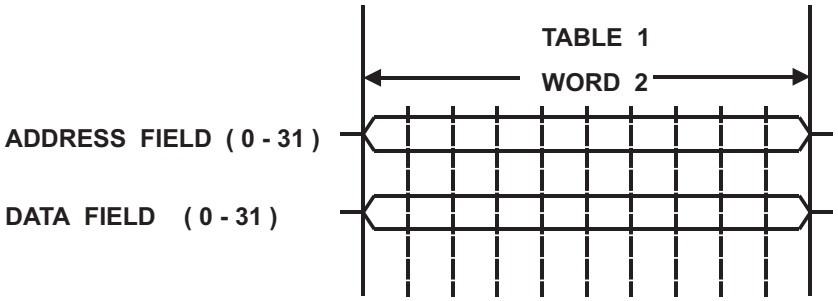


Figure B-6 Bus Structure

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

In most applications, the signals comprising a word in a field are not active during the entire word period. As shown in figure B-9, the address field period may be active from t2 (20ns) through t9 (180ns), while the data field is active from t4 (60ns) through t8 (160ns).

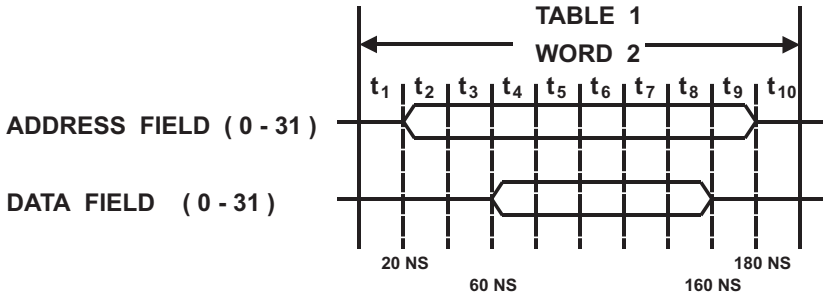


Figure B-9 Improved Bus Structure

3.2 Field Timing

In order to achieve the “bus signals” depicted in figure C-10, the buses require control signals. In particular, they require an ADDRESS ENABLE and DATA ENABLE signal, figure B-8. These signals are generated from a separate timing generator, nomenclated the “timing set”.

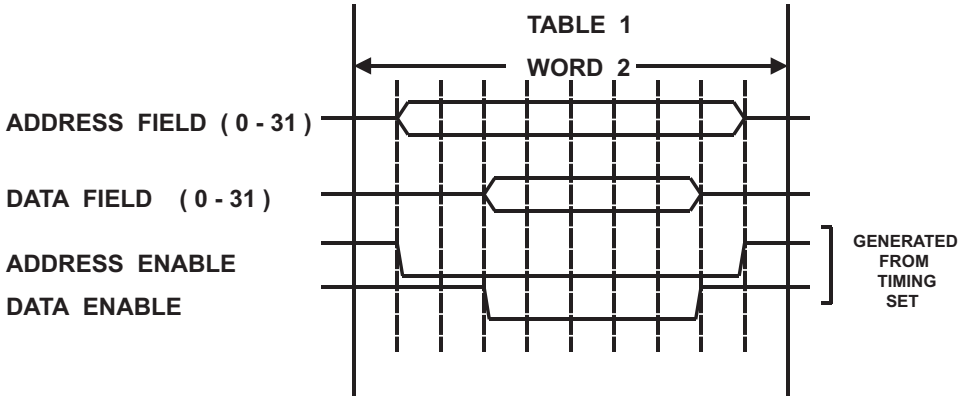


Figure B-8 Field Timing

3.3 Timing Signals

Typically, the UUT requires additional control signals which further define the “bus cycle”. figure B-11

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

describes a UUT R/W-, a UUT Data Enable and a UUT Data Strobe signal. These signals are also generated by the SR192 timing set.

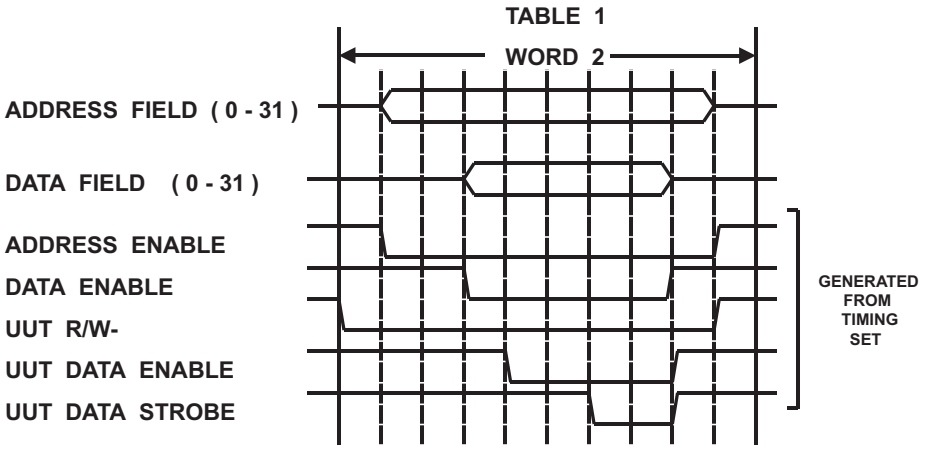


Figure B-11 Bus Cycle Timing

Once active data has been placed on the data bus and the data enable signal has been asserted, there must be a means to determine that the UUT is ready for data. The timing set must have the ability to test the UUT READY signal and enter a “WAIT” state until the test condition is true. This sequence is often referred to as a “handshake” sequence.

Figure B-10 indicates that the timing set will remain in state t5 until the UUT READY- signal goes low. Once the UUT READY- = Low condition is met, the timing set will continue. If the test condition does not occur in a specified time, a timeout condition will force the continuation of the timing set and inform the VXI controller that a timeout occurred. The timeout logic can be disabled, which in turn will cause the SR192 to remain in state t5 until commanded to complete by the VXI controller.

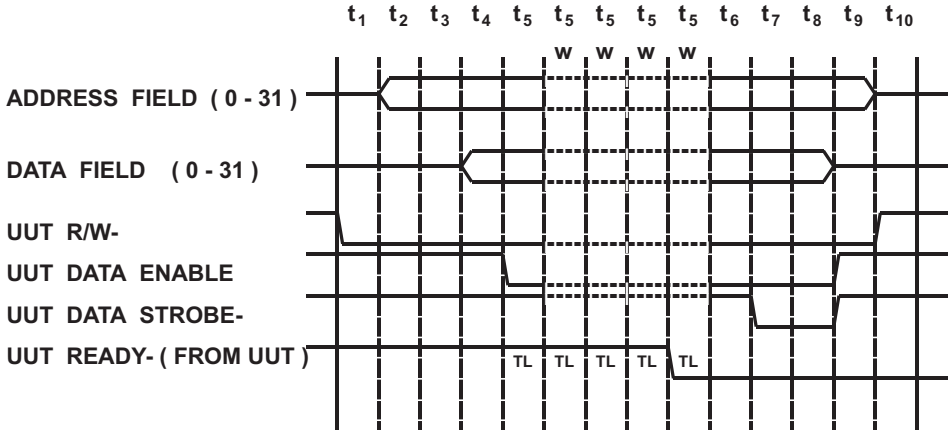


Figure B-10 Testing Input Signals

The SR192 incorporates 64 timing sets, each timing set programmed to simulate a particular bus cycle type or UUT timing cycle.

4 BUS EMULATION TESTING

Bus Emulation Testing is a test method in which the unit under test (UUT) is exercised, in real time, through all of its functions. This is accomplished by precisely simulating all the interfaces into and out of

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

the UUT as well as transferring functional data to/from the UUT. Successful bus emulation testing requires independent bus emulators for each interface resident on the UUT.

figure B-12 depicts a typical UUT. The UUT has two interfaces, a VME interface and a SCSI bus interface. Successful test of the UUT requires two bus emulators, each providing a real time simulation of the address, data, and control lines of the respective bus, as well as the ability to transmit and receive functional data to/from the UUT.

Each SR192 has the capability of emulating two independent interfaces with up to 96 channels each.

This UUT also incorporates a 68020 microprocessor. If it is desirable to emulate the 68020, a second bus emulator module would be installed to simulate this function. The UUT would then be exercised, in real time, through all its functions.

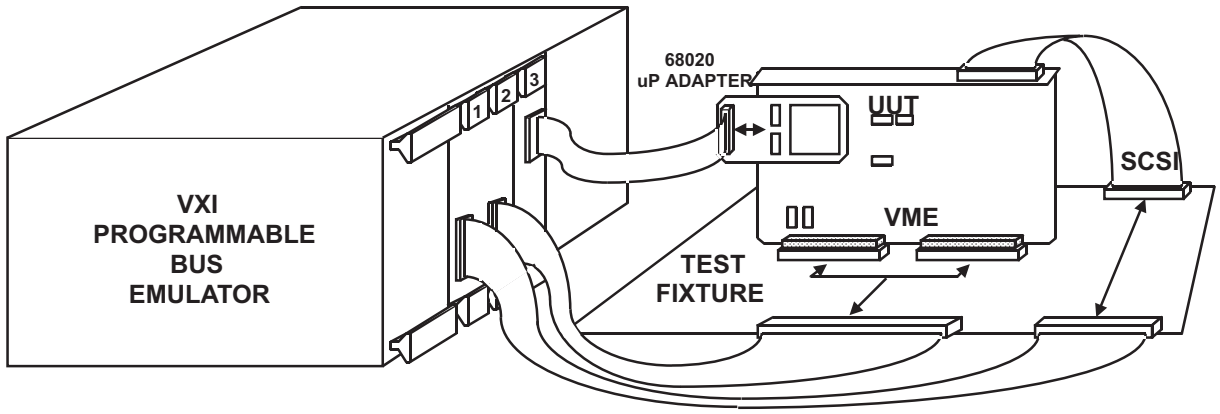


Figure B-12 Sample UUT

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Appendix C Worksheets

The following worksheets are included:

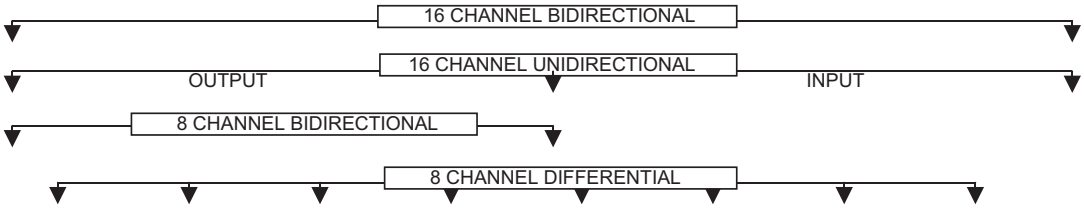
1. Channel Group.
2. Independent Timing Set.
3. Linked Timing Set.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SR192 CHANNEL GROUP WORKSHEET

Group Name:_____ Channels:_____

SLOT **CHANNELS**



DRA1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
DRA2	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
DRA3	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
DRA4	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
DRA5	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
DRA6	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96
DRB1	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112
DRB2	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128
DRB3	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144
DRB4	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160
DRB5	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176
DRB6	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192

SCPI COMMANDS for SIGNAL ASSIGNMENTS

ROUTe:PATH:DEFine _____,_____
<group_name> <channel_list>

CHANnel:MODE _____,_____
<group_name> HOLD | SERial | MULTiplex | RTO | RTZ | RTC | INCRement<1 | 2 | 4 | 8>

OUTPut:ENABle _____,_____
<group_name> TSEN(1,2) | FCNTl(1,2) | CSTRobe | ALWays | NEVer

OUTPut:ENABle:DELay _____,_____
<group_name> 0 - 3

OUTPut:REGister:STATe _____,_____
<group_name> ON | OFF

OUTPut:REGister:SOURce _____,_____
<group_name> STIM_LOAD | TSSTr(1,2) | FCNTl(1,2) | CSTRobe

INPut:STRobe _____,_____
<group_name> TSSTr(1,2) | FCNTl(1,2) | CSTRobe | TRANsparent

INPut:STRobe:DELay _____,_____
<group_name> 0 - 3

INPut:MSTRobe _____,_____
<group_name> TSSTr(1,2) | FCNTl(1,2)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SR192 TIMING SET WORKSHEET

Timing Set Name_____ Sheet___of___

CELL	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0
SR_CLK (Aor B)																				
BIT 0																				
ADEL_CLK (A or B)																				
BIT 1																				
STIM_LOAD (A or B)																				
BIT 2																				
TSENable1 (A or B)																				
BIT 3																				
LOW ORDER BITS																				
TSENable2 (A or B)																				
BIT 4																				
TSSTrobe1 (A or B)																				
BIT 5																				
TSSTrobe2 (A or B)																				
BIT 6																				
TSOUT1 (A or B)																				
BIT 7																				
MIDDLE BITS																				
TSOUT2 (A or B)																				
BIT 8																				
TSOUT3 (A or B)																				
BIT 9																				
TSOUT4 (A or B)																				
BIT 10																				
TSOUT5 (A or B)																				
BIT 11																				
HIGH ORDER BITS																				
TEST CONDITIONS																				
TSINput1 (A or B),(H or L)																				
TSINput2 (A or B),(H,L,＼ or ／)																				
RESPONSE_ERROR																				
RESPONSE_COMPARE																				
Insert Delay																				

BIN---HEX

0000---0
0001---1
0010---2
0011---3
0100---4
0101---5
0110---6
0111---7
1000---8
1001---9
1010---A
1011---B
1100---C
1101---D
1110---E
1111---F

NOTE: Default values are high or "1".

3/99

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SR192 LINKED TIMING SET WORKSHEET

Timing Set Name _____ Sheet ___ of ___

CELL	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
SR_CLK A															
BIT 0															
ADEL_CLK A															
BIT 1															
ADEL_CLK B															
BIT 2															
STIM_LOAD A															
BIT 3															
STIM_LOAD B															
BIT 4															
TSENable1A															
BIT 5															
TSENable2A															
BIT 6															
TSENable1B															
BIT 7															
BYTE 1															
TSENable2B															
BIT 8															
TSSTrobe1A															
BIT 9															
TSSTrobe2A															
BIT 10															
TSSTrobe1B															
BIT 11															
TSSTrobe2B															
BIT 12															
TSOUT 1A															
BIT 13															
TSOUT 2A															
BIT 14															
TSOUT 3A															
BIT 15															
BYTE 2															
TSOUT 4A															
BIT 16															
TSOUT 5A															
BIT 17															
TSOUT 1B															
BIT 18															
TSOUT 2B															
BIT 19															
TSOUT 3B															
BIT 20															
TSOUT 4B															
BIT 21															
TSOUT 5B															
BIT 22															
BIT 23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
BYTE3															
TEST CONDITIONS															
TSINput1 (H or L)															
TSINput2 (H,L, ⟍ or ⟋)															
RESPONSE_ERROR															
RESPONSE_COMPARE															
Insert Delay															

BIN---HEX

0000---0
0001---1
0010---2
0011---3
0100---4
0101---5
0110---6
0111---7
1000---8
1001---9
1010---A
1011---B
1100---C
1101---D
1110---E
1111---F

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Appendix D Improving SR192 Access

1 VXI COMMUNICATION LAYERS

The following figure describes the different communication layers that can be used to program the SR192's field and timing memories.

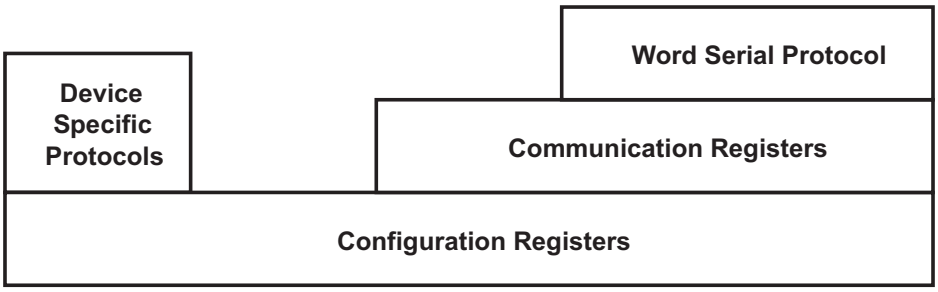


Figure D-1 VXI Communication Layers

The lowest level of communication of any VXIbus device is the VME bus protocol. The VME bus protocol is defined by generic bus cycles (READ, WRITE, etc.) that are controlled by hardware on each device.

1.1 Register Based Devices

The VXIbus protocol defines a specific addressing method as well as a standard set of registers called the Configuration Registers. VXIbus devices who only support these registers are called "Register Based Devices". Data/commands are sent to these registers via VMEbus WRITE cycles. Data/results are received from these registers via VMEbus READ cycles. Register based devices can be programmed quickly because their control functions are closer to the hardware. Register based devices generally require a custom driver because there is no standard set of rules governing programming or setup.

1.2 Message Based Devices

An optional device class that the SR192 supports is called "Message Based". Message based devices define an additional set of registers called the Communication Registers. As with the Configuration Registers data is sent/received to these registers via VMEbus WRITE/READ cycles. Message Based devices are required to support a minimum set of rules and communication capability. Additionally, several manufacturers have developed a command language called SCPI (Standard Command for Programmable Instruments) in order to standardize commands for instruments with similar functionality. Message based devices can be programmed with drivers supplied with the Slot 0 resource manager, however, Message Based device communication is performed a character at a time and involves several protocols as described below.

2 THE WORD SERIAL PROTOCOL

All Message Based device must support the Word Serial protocol. The Word Serial protocol defines a set of commands (WS Commands) and device functionality. Two of the message based communication registers are of particular interest, the Data Low and Response registers. These registers are used to send/receive WS commands and data.

The Response Register contains two bits, Read Ready (RRDY) and Write Ready (WRDY), that handshakes WS commands/responses to the Data Low register. The Response Register also contains two additional bits, Data Input Ready (DIR) and Data Output Ready (DOR), that are used to handshake message strings to/from the SR192's command parser buffer via the Byte Transfer protocol.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The following figure illustrates the processing involved with a Slot 0 resource manager sending an EXEC command to a SR192.

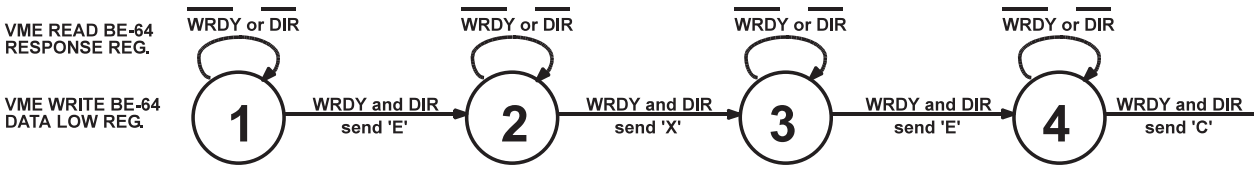


Figure D-2 Byte Transfer Protocol Example

Step 1 — Resource manager reads the SR192's RESPONSE register until both WRDY and DIR are both true. When WRDY and DIR are both true, the resource manager sends the 'E' to the SR192 by sending a Byte Available Word Serial command to the DATA LOW register.

Step 2 — Resource manager reads the SR192's RESPONSE register until both WRDY and DIR are both true. When WRDY and DIR are both true, the resource manager sends the 'X' to the SR192 by sending a Byte Available Word Serial command to the DATA LOW register.

Step 3 — Resource manager reads the SR192's RESPONSE register until both WRDY and DIR are both true. When WRDY and DIR are both true, the resource manager sends the 'E' to the SR192 by sending a Byte Available Word Serial command to the DATA LOW register.

Step 4 — Resource manager reads the SR192's RESPONSE register until both WRDY and DIR are both true. When WRDY and DIR are both true, the resource manager sends the 'C' to the SR192 by sending a Byte Available Word Serial command to the DATA LOW register.

2.1 The Word Serial Bottleneck

As can be seen in the figure above there are two layers of protocol as well as a command parser when sending Word Serial messages. This overhead averages to about 2 milliseconds per eight bit byte for the SR192. Binary block data transfers have a 500 microsecond overhead because they bypass the command parser.

Talon recognized the possible bottleneck of data that could occur with the Word Serial Protocol. Therefore the Field and Timing memory were placed within the VXIbus A32 address space so that they could be programmed without using the Word Serial Protocol. Using the A32 memory, sixteen bit words can be read/written approximately every 1 microsecond using direct VME bus READ and WRITE cycles.

3 SPEEDING UP TABLE TRANSFERS.

Using the preceding numbers above, the following rules can be used to speed up data throughput to/from the SR192.

1. Program the field memory using the A32 memory. 32K 64 bit words works out to 128K 16 bit words. 128K words X 1 microsecond/word = 128 milliseconds (best case).
2. Use the fill functions to define the data. The SR192 can program the field memory faster than the SCPI commands. 32K field increment = ~2 seconds.
3. Program the fields separately if one field can be filled using a fill pattern or if one field is set to input.
4. Use the DATA commands instead of the WORD commands. The DATA command requires fewer bytes and bypasses the parser for the data block. TABL:DATA using a 32K table = ~ 256K bytes X 500 microseconds/byte = ~128 seconds (2 min.). TABL:WORD using a 32K table = ~ 1.4Mbytes X 2 millisecond/byte = ~2,800 seconds (46 min).
5. Use the CALC:CRC command to verify large tables results.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group is an independent supplier of quality pre-owned equipment

Gold-standard solutions

Extend the life of your critical industrial, commercial, and military systems with our superior service and support.

We buy equipment

Planning to upgrade your current equipment? Have surplus equipment taking up shelf space? We'll give it a new home.

Learn more!

Visit us at **artisantg.com** for more info on price quotes, drivers, technical specifications, manuals, and documentation.

Artisan Scientific Corporation dba Artisan Technology Group is not an affiliate, representative, or authorized distributor for any manufacturer listed herein.

We're here to make your life easier. How can we help you today?

(217) 352-9330 | sales@artisantg.com | artisantg.com

ARTISAN® TECHNOLOGY GROUP