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 *Tektronix VX5790A* at our website: ***Click HERE***



This manual was downloaded from Artisan Scientific Corporation.

If you are interested in other electrical T&M, optical T&M, and/or laboratory equipment please visit us on the web at http://www.artisan-scientific.com.



VX5790 Arbitrary Waveform Generator Users Manual

PRELIMINARY DOCUMENTATION

Preliminary documentation is subject to change without notice. Every effort is made to ensure that preliminary documentation is as accurate as possible; however, it may contain errors, omissions, or information that no longer applies.

10/16/89

Tektronix®
COMMITTED TO EXCELLENCE

Copyright ©1989 Tektronix, Inc. All rights reserved. Contents of this publication may not be reproduce in any form without the written permission of Tektronix, Inc.

Products of Tektronix, Inc. and its subsidiaries are covered by U.S. and foreign patents and/or pending patents.

Printed in the U.S.A. Specification and price change privileges are reserved.

Tektronix, Inc.
VXI Products
Jack Murdock Park
P.O. Box 3500
Vancouver, WA 98607-3500

061-3754-00
First Printing OCTOBER 1989

Preface

Purpose of this Document

This manual provides the information necessary to install, configure, and operate the VX5790 Arbitrary Waveform Generator (AWG).

Organization of this Manual

This manual consists of the following sections and appendices:

Section 1. General Information — provides a brief overview of the VX5790 and its standard and optional accessories. This section also includes the VX5790 specifications.

Section 2. Preparation for Use — provides information about how to configure and install the VX5790.

Section 3. Operation — describes the connectors used and the power-up sequence of the VX5790.

Section 4. Rear-Panel Interface — describes the connectors on the VX5790 rear panel.

Section 5. Introduction to Programming — contains introductory information about the command sets.

Section 6. Instrument Control Command Dictionary — contains an alphabetical and functional listing of the instrument control commands and a description of each command.

Section 7. System Command Dictionary — contains information about the IEEE 488, VX5790 Slot 0 Resource Manager, and Word Serial Protocol commands.

Section 8. ID-ROM Information — contains information about the ID-ROM and its associated command set.

Section 9. Calibration — provides information on the calibration of the AWG.

Appendix A. ASCII and GPIB Code Chart

Appendix B. VXIbus Glossary

Notational Conventions

Active Signal States

An asterisk (*) following a signal mnemonic (as in ACFAIL*) denotes the signal is active when in its low state.

A signal mnemonic without an asterisk (*) denotes the signal is active when in its high state.

Number Base

Unless otherwise noted, all numbers are assumed to be decimal (base 10).

Binary numbers (base 2) are followed by the character "b" (for example: 1101b).

Hexadecimal numbers (base 16) are followed by the character "h" (for example: 97FFh).

Octal numbers (base 8) are followed by the character "o" (for example: 307o).

Related Publications

The following documents on related subjects may be useful in efficient use of the AWG:

- *VXIbus System Specification, Version 1.3, July 14, 1989*
- *VMEbus Specification Manual, Revision C.1, October, 1985*
- *ANSI/IEEE Standard 1014-1987, IEEE Standard for a Versatile Backplane Bus: VMEbus*

Table of Contents

Preface

Purpose of this Document i
Organization of this Manual i
Notational Conventions ii
Active Signal States ii
Number Base ii
Related Publications ii

Section 1 General Information

Introduction 1-1
Description 1-1
Specifications 1-2
Performance Conditions 1-2

Section 2 Preparation for Use

Configuration 2-1
Setting the Logical Address 2-1
Configuring the Backplane Jumpers 2-2
Installation 2-4
Module Installation Using Injector/Ejector Handles 2-4
Module Removal Using Injector/Ejector Handles 2-5

Section 3 Operation

Connectors and Indicators 3-1
Connectors 3-1
Indicators 3-1
Diagnostics and Calibration 3-2
Background Information 3-2
SYSFAIL* Signal 3-2
Front-panel Indicators 3-4
Diagnostic Routines 3-4
Calibration 3-4
Diagnostic/Calibration Commands 3-5
Diagnostic Routine Use 3-5
Kernel Test 3-5
Self Test 3-6
Operational Considerations 3-6
Non-Fatal Error Messages 3-6
Fatal Instrument Module Errors 3-6
Kernel Failure 3-6
Hardware Failure 3-6
Calibration 3-6
Triggering 3-7
Trigger Protocols 3-7
SYNC Protocol 3-7
ESTST Protocol 3-7
SYNC100 Protocol 3-8
STAR Bus 3-8

Section 4 Rear-Panel Interface

Introduction 4-1

Section 5 Introduction to Programming

Introduction 5-1
Command Notation 5-1
Program Message 5-2
ASCII Code 5-3
Syntactic Delimiters 5-3
White Space 5-4
Special Characters 5-4
Program Data 5-4
- Numeric Program Data Value Range 5-4
- Decimal Numeric Program Data 5-5
- Non-Decimal Numeric Program Data 5-5
- String Program Data 5-6
- Arbitrary Block Program Data 5-6

Command Formats 5-7
- Simple Command Program Header 5-8
- Simple Query Program Header 5-8
- Compound Command Program Header 5-9
- Compound Query Program Header 5-9
- Common Command Program Header 5-10
- Common Query Program Header 5-10
- Multiple-Command Format With Same 1st Program Mnemonic . 5-10

Case Sensitivity 5-11
Command Guidelines 5-11
- Waveform Data Memory 5-11
- Program Memory 5-13

Command Types 5-15
Command Dictionary Layout 5-15

Section 6 Instrument Control Command Dictionary

Introduction 6-1
- Alphabetical Command Listing 6-1
- Functional Command Listing 6-3

Section 7 System Command Dictionary

Introduction 7-1
Alphabetical Listing 7-3

Section 8 ID-ROM (NVRAM) Information

Introduction 8-1
ID-ROM Commands 8-1
Error Codes 8-2
Data Object Identifier Mnemonics 8-2
Additional ID-ROM Contents 8-4
Alphabetical List of ID-ROM Commands 8-4

Section 9 Calibration

Introduction 9-1
Calibration Commands 9-2
The Calibration Algorithm 9-3
Possible Reasons for Poor Calibration 9-4
Half-LSB Output Accuracy 9-6

Appendix A ASCII/GPIB Code Chart

Appendix B VXIbus Glossary

List of Illustrations

Figure	Title	Page
	VX5790 Arbitrary Waveform Generator	x
2-1	Injector/Ejector Mechanism	2-5
3-1	AWG Diagnostic Dependencies	3-3
3-2	ESTST Protocol	3-7
3-3	SYNC100 Protocol	3-8
5-1	Command/Query Flowchart	5-7
5-2	Waveform Makeup	5-11
5-3	Waveform Memory Organization	5-12
5-4	Program Memory Organization	5-14
6-1	Pulse Definitions	6-37
6-2	X and Y Parameters	6-49
A-1	ASCII/GPIB Chart	A-1

List of Tables

Table	Title	Page
1-1	Environmental Specifications	1-3
1-2	Electrical Specifications	1-4
1-3	Mechanical Specifications	1-7
2-1	Bus Grant Jumpers	2-3
2-2	Interrupt Acknowledge Jumpers	2-3
4-1	P1 Pinout	4-2
4-2	P2 Pinout	4-3
4-3	P3 Pinout	4-4
5-1	Waveform Memory Term Definitions	5-12
5-2	Program Memory Term Definitions	5-14
7-1	IEEE 488 Commands Supported by the VX5790	7-2
7-2	VX5520 Commands Supported by the VX5790	7-2
8-1	ID-ROM Error Codes	8-2
8-2	ID-ROM Data Object Identifier Mnemonics	8-2

Operators Safety Summary

The general safety information in this summary is for both operating and servicing personnel. Specific warnings and cautions are found throughout the manual where they apply, and may not appear in this summary.

TERMS

Terms in This Manual

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

CAUTION statements identify conditions or practices that could result in damage to the module or other property.

Terms as Marked on Module

DANGER indicates a personal injury hazard immediately accessible as one reads the marking.

CAUTION indicates a personal injury hazard not immediately accessible as one reads the markings, or a hazard to property, including the module itself.

SYMBOLS

Symbols in This Manual



This symbol indicates where applicable cautionary or other information is to be found.



This symbol indicates where special explanatory information is included in the manual. There is no caution or danger associated with the information.

Symbols as Marked on Module



DANGER - High Voltage.



Protective ground (earth) terminal.



ATTENTION - Refer to manual.



Refer to manual before using.

Power Source

This module is intended to operate in a mainframe whose power source does not apply more than 250 V rms between the supply conductors or between either supply conductor and ground. A protective ground connection through the grounding conductor in the power cord(s) is essential for safe operation.

Grounding the Module

This module is grounded through the grounding connector of the mainframe power cord(s). To avoid electrical shock, plug the mainframe power cord(s) into a properly wired receptacle before connecting to the module connectors. A protective ground connection through the mainframe is essential for safe operation.

Danger Arising From Loss of Ground

Upon loss of the protective-ground connection, all accessible conductive parts can render an electric shock.

Use the Proper Fuse

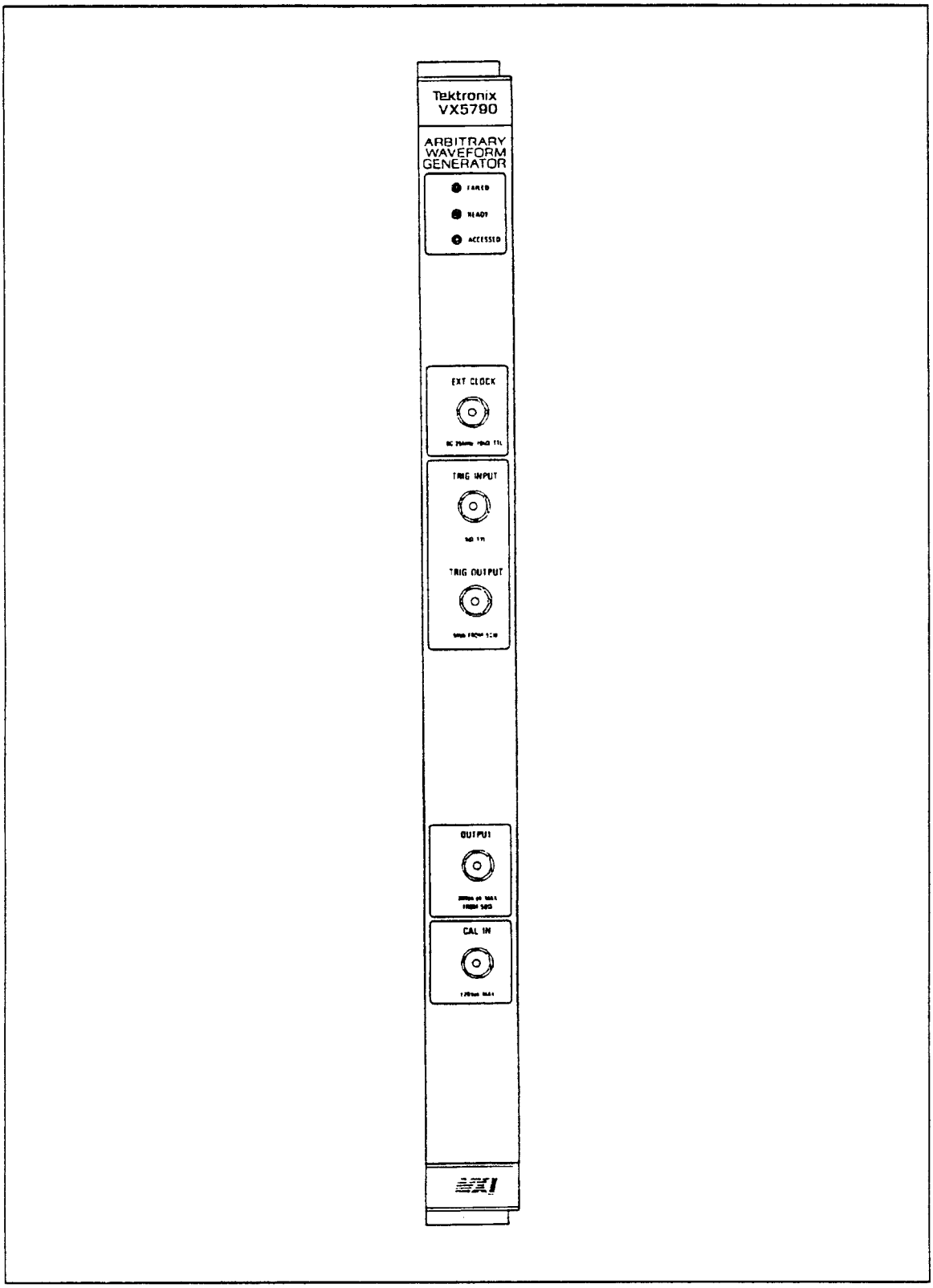
To avoid fire hazard, use only fuses specified in the module parts list. A replacement fuse must meet the type, voltage rating, and current rating specifications required for the fuse that it replaces.

Do Not Operate in Explosive Atmosphere

To avoid explosion, do not operate the module in an explosive atmosphere.

Do Not Remove Covers or Panels

To avoid personal injury, the module covers should be removed only by qualified service personnel. Do not operate the module without covers and panels properly installed.



VX5790 Arbitrary Waveform Generator

Section 1
General Information

Introduction

Description

The VX5790 Arbitrary Waveform Generator is a single-wide D-size module having the following features:

- Custom waveforms of up to 100 MS/s (10 ns/point resolution)
- Waveforms from 0.01 — 25 MHz
- 10 bits of amplitude resolution
- Flexible waveform memory structure
- 128K data points of waveform memory
- Single channel
- A wide variety of trigger sources and modes
- Output disable
- 3 selectable filters

The waveform memory of the VX5790 is organized as data packets. Packets can be set to any size from 24 — 2048 10-bit words, and sequences of data packets can be combined into custom waveforms.

The VX5790 can be programmed to generate an output trigger at any data point in a waveform. This trigger can be routed to the VXIbus via a STAR line, to a VXIbus ECLTRG or TTLTRG line, and is available on the front panel BNC connector.

The VX5790 accepts input triggers from the VXIbus STAR lines, VXIbus ECLTRG or TTLTRG lines, from software, the front panel BNC connector, SYNC100 Trigger Protocol, or the ESTST Trigger Protocol.

The VX5790 supports both VXIbus Word Serial and Shared Memory Protocols for communications with other VXIbus devices. Software codes and command formats conform to IEEE 488.2 and VXIbus System Specification Revision 1.2.

Specifications

Performance Conditions

The performance characteristics listed in the specifications table in this appendix are valid within the following limits:

- The AWG power-on diagnostics tests must have been performed successfully.
- The AWG must have warmed up for 20 minutes or more.
- The AWG must be operating in an environment within the limits described in the environmental specifications.

Any condition unique to a particular characteristic but not listed above is stated as part of that characteristic.

The electrical and environmental performance limits, together with the related validation procedure, comprise a complete statement of the electrical and environmental performance of a calibrated AWG.

Table 1-1. Environmental Specifications

Characteristic	Description
Temperature:	
Operating	0 — 50° C (32 — 122° F)
Non-operating	-40 — +71° C (-40 — +160° F)
Cooling	Circulating air flow sufficient to dissipate 100W
Humidity:	
Below 30° C	95 ± 5%, non-condensing
Above 30° C:	75 ± 5%, non-condensing
Above 40° C	45 ± 5%, non-condensing
Altitude:	
Operating	3.05 km (10,000'). Derate maximum operating temperature 1° C for each 300 m (1000') above 1.5 km (5000')
Non-operating	4.7 km (15,000')
Vibration:	
Non-operating	Withstands 0.33 mm (0.013") p-p; 5 — 55 Hz sine wave, 15 minutes each axis; and 10 minutes each axis at resonance or 55 Hz
Shock:	
Non-operating	Withstands 30 g, halfsine, 11 ms duration, 3 shocks in each direction along 3 major axes, 18 total shocks
Bench Handling:	
Non-operating	12 drops from 45°, 4", or equilibrium, whichever occurs first
Transportation	Qualified under National Safe Transit Association Preshipment Test Procedures 1A-B-1 and 1A-B-2
Electromagnetic Compatibility	Within limits of FCC Regulations, Part 15, Subpart J, Class A. Complies with VDE 0871, Category B. Complies with MIL STD 461B, Part 7 (CS01, CS02, CS06, RS03-limited to 1 GHz, RE02, CE01, CE03)
Electrostatic Immunity while Operating:	
Non-interruptive	15 kV, 100Ω in series with 500 pF
Non-destructive	20 kV, 100Ω in series with 500 pF

Note

In general, the Mainframe will meet MIL-T-28800D, Class 5 environmental requirements. Testing is accomplished using procedures contained in Tektronix Standards for Class 5, which, in some instances, are stricter than MIL-T-28800D, Class 5.

Table 1-2. Electrical Specifications

Characteristic	Performance
FREQUENCY	
Range	0.01 Hz — 25 MHz
Filters	
Type	5 pole Butterworth, low pass
Cutoff frequencies	390 KHz, 3.1 MHz, 25 MHz
OUTPUT	
Amplitude	
Range	25 mV — 20 V p-p into open 12.5 mV — 10 V p-p into 50 Ω
Resolution	3 digits + 2 mV
Accuracy	± (0.75% + 5 mV + 0.05% offset)
Stability	± (0.5% + 4 mV + 0.1% offset)
Amplitude Flatness	< ± 0.2 dB, dc — 100 KHz < ± 1 dB, 100KHz — 25 MHz
Offset	
Range	-10 — +10 V into open -5 — +5 V into 50 Ω
Resolution	2 mV
Accuracy	± (0.5% + 6 mV + 0.05%)
Stability	± (0.5% + 4 mV)
CAL IN Connector	
Input Signal Range	± 20 V
Input Impedance	≥ 100 KΩ in parallel with 30 pF
Impedance	
Output On	50 Ω ± 1%
Output Off	50 Ω ± 1% to ground
Waveform	
Resolution	10 bits
Integral Linearity	± 0.3%
Differential Linearity	± 0.5 LSB
Settling Time	≤ 50 ns to 1%
Sinewave Distortion	
THD + N	≤ 0.2% 10 Hz — 100 KHz
Harmonics	-45 dB 100 KHz — 2 MHz -30 dB 2 — 25 MHz
Square Wave Performance	
Transition Time	≤ 8 ns
Aberrations	± (5% + 5 mV) into 50 Ω
Symmetry	

Table 1-2. Electrical Specifications (cont.)

Characteristic	Performance
WAVEFORM MEMORY	
Memory Size	128K X 10 bits (volatile)
Organization	
Packet Size	24 — 2048 data points in multiples of 8 data points
Packet Scans	Each packet can be scanned 1 — 256 times
Number of Packets in a Sequence	1 — 2048
CLOCK	
CLK100 Clock	
Clock to Signal Delay	25 ns maximum (with divide ratio = 1)
External BNC	
Signal Level	TTL Level (0 — 5.5 V)
Clock Frequency	dc — 25 MHz
Duty Cycle	50% ± 20%
Clock to Signal Delay	35 ns maximum (with divide ratio = 1)
Input Impedance	≥ 1 KΩ
STARX Clock	
Signal Level	10 KH ECL level, as specified by the *VXIbus Specification*
Clock Frequency	dc — 100 MHz
Duty Cycle	50% ± 20%
Clock to Signal delay	25 ns maximum (with divide ratio = 1)
Internal Divider	
Divide Ratios	2^n where n = 0 — 24
Clock Range using divider and 100 MHz clock	5.96 Hz — 100 MHz
Data Point Duration using divider and 100 MHz clock	10 ns — 167 ms
INPUT TRIGGERS	
Software Trigger	Word Serial Command
External BNC	
Signal Level	TTL level (0 — 5.5 V)
Input Impedance	≥ 1 KΩ
Minimum Pulse Width	10 ns
Maximum Trigger to Signal delay	$40\ ns + \frac{1}{f_{SOURCE\ CLOCK}}$
STARY	
Signal Level	10 KH ECL level, as specified by the *VXIbus Specification*
Minimum Pulse Width	10 ns
Maximum Trigger to Signal delay	$35\ ns + \frac{1}{f_{SOURCE\ CLOCK}}$

Table 1-2. Electrical Specifications (cont.)

Characteristic	Performance
INPUT TRIGGERS (cont.)	
ECLTRGn - Synchronous Protocol	
Signal Level	10 KH ECL level, as specified in the *VXIbus Specification*
Minimum Pulse Width	8 ns
Maximum Trigger to Signal delay	$35\ ns + \frac{1}{f_{SOURCE\ CLOCK}}$
TTLTRGn - Synchronous Protocol	
Signal Level	TTL level, as specified by the *VXIbus Specification*
Minimum Pulse Width	30 ns
Maximum Trigger to Signal delay	$80\ ns + \frac{1}{f_{SOURCE\ CLOCK}}$
ESTST Trigger	
Maximum Trigger to Signal delay	$50\ ns + \frac{1}{f_{SOURCE\ CLOCK}}$
SYNC100 Trigger	
Maximum Trigger to Signal delay	$50\ ns + \frac{1}{f_{SOURCE\ CLOCK}}$
OUTPUT TRIGGERS	
Output Trigger Duration	$4 \times \frac{1}{f_{DIVIDED\ CLOCK}}$
External BNC	
Signal Level	-0.5 — 2.2 V sourced from 50 Ω
Output Impedance	50 Ω ±10 %
Maximum Signal to Trigger delay	-9 ns
STARY	
Signal Level	10 KΩ ECL level, as specified by the *VXIbus Specification*
Maximum Signal to Trigger delay	-9 ns
ECLTRGn - Synchronous Protocol	
Signal Level	10 KΩ ECL level, as specified by the *VXIbus Specification*
Maximum Trigger to Signal delay	-9 ns
TTLTRGn - Synchronous Protocol	
Signal Level	TTL level, as specified by the *VXIbus Specification*
Maximum Trigger to Signal delay	45 ns
VXIbus Extended Start/Stop	
Maximum CLK100 edge to Signal Start Delay	40 ns
Maximum CLK100 edge to Signal Stop Delay	30 ns

Table 1-2. Electrical Specifications (cont.)

Characteristic	Performance	
POWER		
Current Requirements	Ipm Max	Idm Max
+5 V	4.4 A	? A
-5.2 V	4.6 A	? A
-2 V	1.8 A	? A
+24 V	0.5 A	? A
-24 V	0.5 A	? A
Power Consumption		
Maximum	70 W	

Table 1-3. Mechanical Specifications

Characteristic	Description
Overall Dimensions	
Height	366.7 mm (14.437")
Width	30.18 mm (1.188")
Depth	345.03 mm (13.584")
Weight	
Net	≈ ? kg (? lbs)
Shipping	≈ ? kg (? lbs)
Cooling	Forced air circulation. Airflow direction is from the base of the unit to the top. Required airflow should exceed ? CFM.

Section 2
Preparation for Use

Configuration

The AWG requires no user configuration other than setting its logical address. Although there are several jumpers on the AWG board, these are for use by service personnel.

Setting the Logical Address

The AWG uses an 8-bit, binary-coded, DIP switch (SW1) located at the top of the rear half of the PCB to set its Logical Address. It is accessible by removing the right side panel (when facing the front panel). To remove the side panel, unlock the six 1/4-turn fasteners that attach the panel to the module chassis. These fasteners unlock when turned 1/4 turn counterclockwise from their locked position, which is indicated when their screwdriver slot is inline with the small plastic insert adjacent to the fastener. When replacing the panel, be sure to lock all of the 1/4-turn fasteners by turning them 1/4 turn clockwise.

Each VXI device must be assigned a unique 8-bit Logical Address in the range 0 — 255. Of these addresses, 0 is reserved for a Slot 0 module and 255 is reserved for Dynamic Auto-Configuration. When the Logical Address is set to 255 and a Tektronix Slot 0 module is installed and operating as the resource manager, that module will automatically set the logical address for the VX5790.

The VX5790 uses an 8-bit, binary coded, DIP switch (SW1) to set this address. The ON position of the switch corresponds to a logical 0; the OFF position corresponds to a logical 1. The factory default setting for all positions is FFh (Logical Address 255).

Installation

The AWG may be inserted into any empty slot in the Mainframe except Slot 0. To install the AWG, align the printed circuit board edges with the guides in the Mainframe Chassis and gently slide it in until the edge connectors on the back edge of the board contact the VXIbus connectors on the Backplane. At that point, press firmly at the top and bottom of the AWG front panel to seat the connectors. Then, install the two screws that secure the AWG front panel to the Mainframe Chassis.

Module Installation Using Injector/Ejector Handles

The following procedure is for installing the module into a Tektronix Mainframe. If the module is being installed in another mainframe, the procedure may need to be modified. Refer to Figure 2-1 while reading this procedure.

To install the module using the Injector/Ejector Handles:

1. Ensure that both the rear-panel power ON/OFF switches and front-panel ON/STANDBY switch on the Mainframe are turned to OFF and STANDBY, respectively.

2. Insert the module into the Mainframe top and bottom module guides for the slot(s) where the module is to be installed and push it partially into the Mainframe. Then pull out the Injector/Ejector Handles to their fully extended position (see the INJECT and EJECT views of Figure 2-1), and push the upper handle fully up and lower handle fully down.

3. Push the module into the Mainframe as far as it will go without forcing it. Then simultaneously pull the upper Injector/Ejector Handle fully down and the lower handle fully up to INJECT the module.

4. Ensure the module is fully seated into its VXI connectors and the front panel is flush with the front of the Mainframe chassis. Then install the top and bottom module retainer screws to secure the module in the Mainframe. Push in the Injector/Ejectors Handles and set them to a level horizontal position (see NORMAL view in Figure 2-1).

Table 2-1. Bus Grant Jumper Straps

Slot	BG0	BG1	BG2	BG3
1	J1011	J1012	J1013	J1014
2	J1021	J1022	J1023	J1024
3	J1026	J1027	J1028	J1029
4	J1031	J1032	J1033	J1034
5	J1036	J1037	J1038	J1039
6	J1041	J1042	J1043	J1044
7	J1046	J1047	J1048	J1049
8	J1051	J1052	J1053	J1054
9	J1056	J1057	J1058	J1059
10	J1061	J1062	J1063	J1064
11	J1066	J1067	J1068	J1069
12	J1071	J1072	J1073	J1074

Table 2-2. Interrupt Acknowledge Jumper Straps

Slot	IACK Connector
0	J2010
1	J2015
2	J2020
3	J2025
4	J2030
5	J2035
6	J2040
7	J2045
8	J2050
9	J2055
10	J2060
11	J2065
12	J2070

Configuring the Backplane Jumpers

NOTE

The following instructions pertain to Tektronix Mainframes. If you are installing the AWG in a different mainframe, refer to the Instruction Manual for that mainframe for the correct jumper strap numbers.

VXIbus mainframes contain daisy-chain jumper straps for the Bus Grant (BG0 — BG3) and Interrupt Acknowledge (IACK) signals. Tektronix Mainframes are shipped with all jumpers installed. All jumper straps must be removed for slots that have cards installed.

NOTE

If your are using a Tektronix Mainframe, the names of the jumper straps (BG0 — BG3 & IACK) are printed on the circuit board facing the front of the mainframe. These jumpers are accessed from the front of the mainframe.

Remove the IACK jumper straps for the slot in which the AWG will be installed. Retain the straps for possible future reconfiguration.

The Bus Grant jumper straps and their associated component numbers are listed in Table 2-1; the Interrupt Acknowledge jumper straps and their associated component numbers are listed in Table 2-2.

Module Removal Using Injector/Ejector Handles

The following procedure is for removing the module from a Tektronix Mainframe. If the module is being removed from another mainframe, the procedure may need to be modified. Refer to Figure 2-1 while reading this procedure.

To remove the module using the Injector/Ejector Handles:

1. Ensure that both the rear-panel power ON/OFF switches and front-panel ON/STANDBY switch on the Mainframe are turned to OFF and STANDBY, respectively, and the top and bottom module retainer screws are removed.

2. Pull out the Injector/Ejector Handles to their fully extended position (see the INJECT and EJECT views of Figure 2-1). Then simultaneously push the upper Injector/Ejector Handle fully up and the lower handle fully down to EJECT the module. Once the module is ejected, it can be removed from the Mainframe.

3. Push in the Injector/Ejectors Handles and set them to a level horizontal position (see NORMAL view in Figure 2-1).

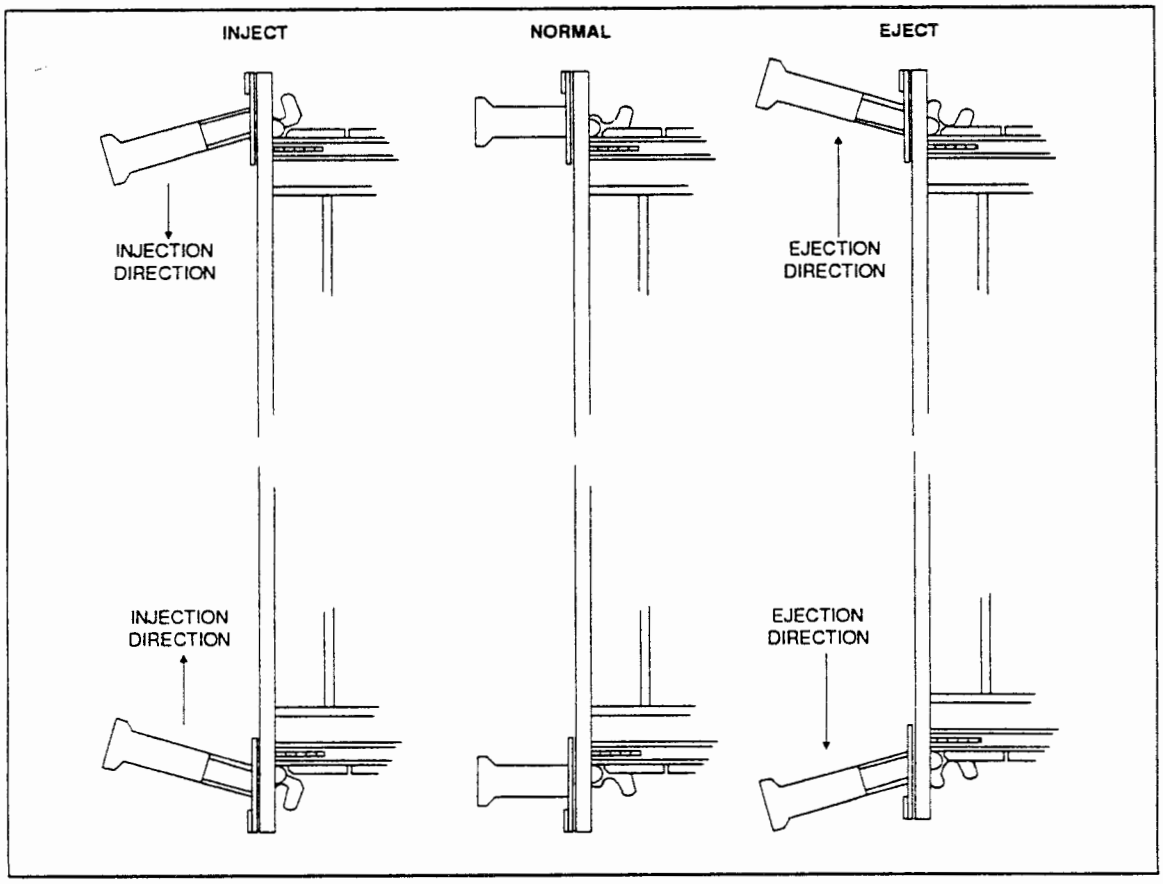


Figure 2-1. Injector/Ejector Mechanism

Section 3
Operation

Connectors and Indicators

The following paragraphs describe the AWG connectors and indicators.

Connectors

There are five BNC connectors on the front panel to allow connection of external clock, trigger input, trigger output, waveform output, and calibration input signals.

- *EXT CLOCK.* Provides for connecting an external TTL compatible data point clock signal to the AWG to be used as an alternate to the 100 MHz CLK100 system clock signal. Input signal amplitude range is 0 — 5 V; the input frequency range is 0 — 25 MHz. The rising edge of the incoming clock signal is used as the trigger.
- *TRIG INPUT.* Provides for connecting an external TTL trigger signal to the AWG trigger input circuits.
- *TRIG OUTPUT.* Provides a TTL version of the AWG trigger output signal to external equipment.
- *OUTPUT.* Provides the arbitrary waveform generated by the AWG to external equipment. The maximum output open circuit signal amplitude is 20 V_{p-p}.
- *CAL IN.* Provides for connecting an external 10 V reference for use in calibration.

Indicators

There are three LED indicators on the AWG front panel: FAILED, READY, and ACCESSED.

- *FAILED.* A red indicator that illuminates when the kernel is inoperative or running power-up tests, and the SYSFAIL* line is asserted.
- *READY.* A green indicator that illuminates when the AWG is operating properly.
- *ACCESSED.* An amber indicator that illuminates when the MODID line is asserted by the Slot 0 device, or flickers when the AWG is engaged in communication via the VXIbus.

Diagnostics and Calibration

The diagnostic routines provide information regarding AWG functionality. In the event of failure, these routines provide information to assist in identifying the AWG circuits not functioning properly. However, these routines isolate to the functional level only and thus do not provide direct information about failed components. The objective of these diagnostics is to decrease the time to isolate to a functional area or circuit block of the device. Figure 3-1 shows that some AWG circuits depend upon other circuits. If one function of the hardware is not operating, those functions depending on it may not produce the expected result even if operating properly.

The calibration routine verifies and adjusts the basic analog calibration of the AWG.

Background Information

First-line service strategy for VXIbus products calls for replacement of failed instrument modules. (An instrument module is defined as a complete functional assembly that plugs into the VXIbus Mainframe; for example, the AWG.) If a VXIbus Subsystem fails, the diagnostic and calibration routines point to the faulty module, which should be replaced. The functions described in the following paragraphs support such failure detection and correction.

SYSFAIL* Signal

For purposes of VXIbus products, the SYSFAIL* signal asserted by an instrument module indicates to the Resource Manager that the instrument kernel has not passed a self-test.

A lighted front-panel FAILED indicator signifies to the user that the SYSFAIL* signal is asserted.

Note

Because an extended test might reveal an error that may be ignored for operations currently in progress, the extended tests described herein are classified as background diagnostics and do not report such errors via the SYSFAIL signal. Instead, these errors are reported through the message-based (word-serial or shared-memory) protocol. Therefore, the SYSFAIL* signal is used only when an instrument is non-responsive (that is, its kernel has failed).*

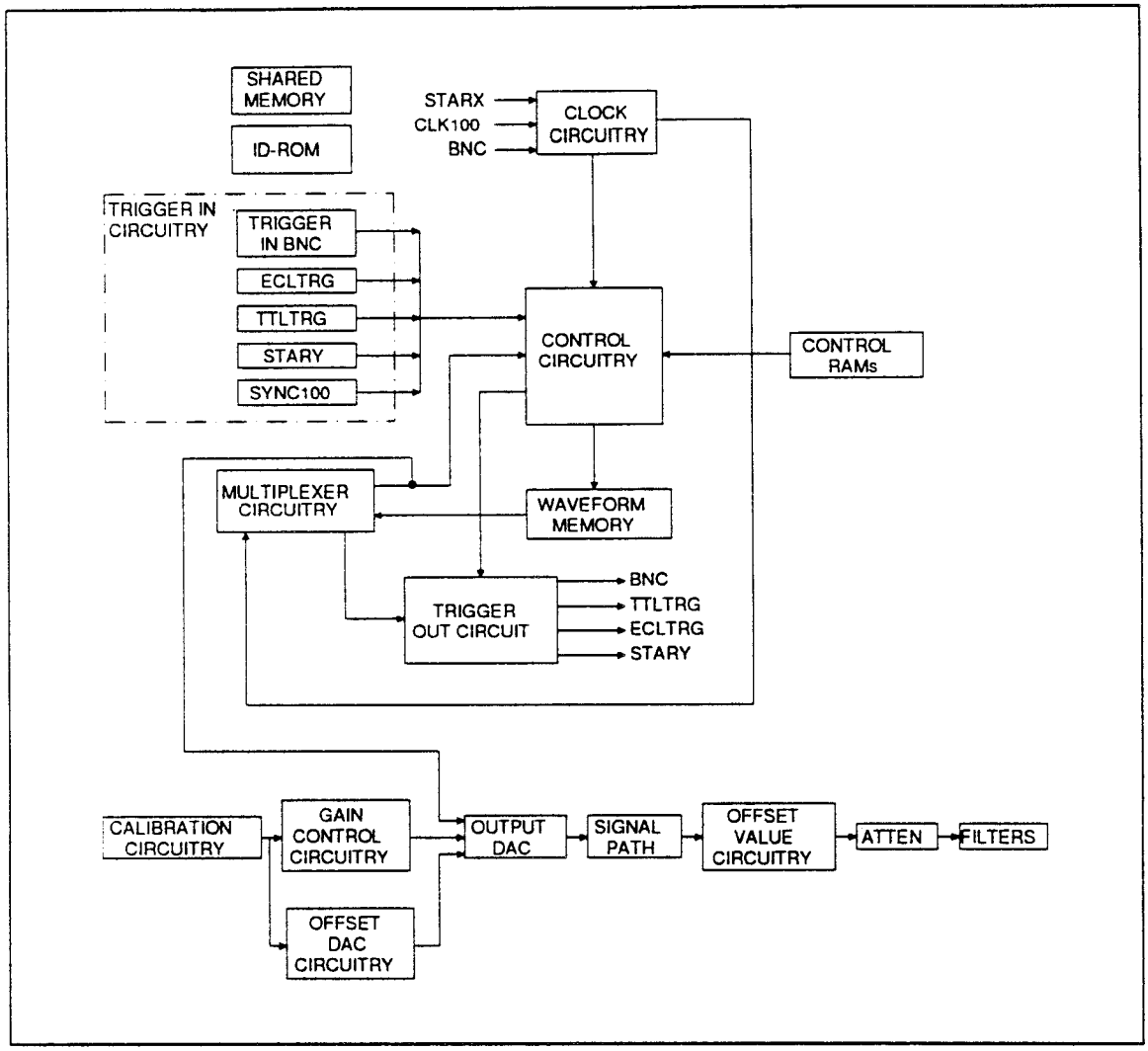


Figure 3-1. AWG Diagnostic Dependencies

Front-panel Indicators

In many instances, the front-panel indicators provide sufficient information to isolate an instrument module failure. A lighted indicator signifies: the kernel is inoperative (red), the module is engaged in communication via the VXIbus (amber), or the module is operating properly (green).

Diagnostic Routines

Diagnostic routines for the AWG consist of the kernel test and the self test.

The kernel test routine verifies that the instrument module processor and ancillary circuits (not including the ID-ROM) are capable of supporting the operating system. It is performed automatically upon power-up.

The self-test routine verifies that the instrument module hardware is capable of proper operation. Self-test can be invoked at any time via the **TST?* high-level command.

The **TST?* command invokes an executive routine called the Test Monitor, which is responsible for executing the self tests, storing any errors in the ID-ROM, and reporting those errors to the module's commander.

Note

It is important to remember that, for diagnostic purposes, the kernel is not a part of the AWG hardware. Kernel failures are reported immediately to the user via a lighted front panel FAILED indicator and via the VXIbus and the SYSFAIL signal.*

Results of the self-test are written to ID-ROM in the form of error codes and the number of failures is transmitted to its commander for error reporting. The **TST?* query and response are transmitted via the synchronous communication channel.

Calibration

The VX5790 has an internal 10 V reference used in conjunction with a calibration digital-to-analog converter (DAC) and comparator to directly measure the output voltages.

The instrument self-calibration routines control the attenuators, output DAC, offset DAC, gain DAC, and calibration DAC to achieve the output voltages specified by the user.

Self-calibration is invoked by the user through the **CAL?* command or automatically on power-up and whenever the instrument amplitude, offset, or impedance settings are changed.

The user can obtain maximum accuracy from the VX5790 by performing a self-calibration (using the **CAL?* command) at any time.

Diagnostic/Calibration Commands

The commands and queries used for diagnostics and calibration are described in Sections 6,7, and 8. These include the following:

*TST? or TST?	Invokes instrument module self-test.
*CAL? or CAL?	Invokes instrument module self-calibration.
CALMODE	Allows the user to select the accuracy-versus-time settings that best suit the application.
GNV?	Extracts calibration error information from the ID-ROM.
MAX?	Enables the user to verify dc calibration.
MID?	Enables the user to verify dc calibration.
MIN?	Enables the user to verify dc calibration.
STERR?	Retrieves error information.

Diagnostic Routine Use

The following paragraphs briefly describe the power-up and self test diagnostic routines and provide information on operational considerations.

Kernel Test

The power-up kernel test sequence starts when the SYSRESET* and ACFAIL* signals become unasserted. This causes the AWG processor to execute the kernel test. For purposes of power-up diagnostic testing, the kernel test routine performs circuit tests verifying the capability of the local processor and its ancillary circuits to support the operating system. If the kernel passes its test, it boots the operating system. The processor then de-asserts the SYSFAIL* signal, extinguishes the red FAILED (kernel inoperative) front-panel indicator, and lights the green READY (module operative) indicator.

Notes

The kernel test is actually more extensive than would be required ordinarily for simple power-up tests. It is designed to include significant tools to assist the service person in determining the cause of the kernel failure.

It is important to remember that, for diagnostic purposes, the kernel is not considered a part of the AWG hardware. Kernel failures are reported immediately to the user via a lighted front-panel FAILED indicator and to the module's commander via the VXIbus and the SYSFAIL signal.*

Self Test

The **TST?* command invokes the self-test, which performs various circuit tests to verify that the AWG hardware is capable of proper operation. When self-test has been completed (regardless of errors), the processor writes the results of the self-test to ID-ROM in the form of error codes and transmits the number of failures to the Resource Manager (for error reporting to the external system).

Note

Like the kernel test, the self test is actually more extensive than would be required ordinarily for simple test purposes. It is designed to include significant tools to assist the service person in determining the cause of a failure.

Operational Considerations

The following paragraphs include information on failure during normal operation and on calibration.

Non-Fatal Error Messages

Once the AWG processor has completed the power-up sequence and communication channels have been established, error reporting via the high-level communication channel may be accomplished. The AWG produces error codes and messages when errors occur. These are returned to the external system.

Fatal Instrument Module Errors

When a fatal error (that is, an error from which recovery is impossible) occurs in the AWG, an error code representing "system error" is stored in ID-ROM along with system status information. The AWG then asserts the SYSFAIL* signal and the AWG processor halts. After restarting the system, the *STERR?* command can then be used to retrieve the failure information. (The *STERR?* command recognizes that, if the first error is a "system error", the information following is necessarily status information and not more error codes.)

Kernel Failure

If the kernel fails during normal operation, the watch-dog timer times out and asserts the SYSFAIL* signal, illuminating the front-panel FAILED indicator.

Hardware Failure

If the AWG hardware fails during normal operation, that failure may be detected by operational (functional) diagnostics or by the self-test routine (invoked by the **TST?* command). Error messages generated by the AWG are transmitted to the module's commander.

Calibration

Calibration may be verified or optimized during normal operation by invoking the **CAL?* (or *CAL?*) command with appropriate arguments.

Triggering

Trigger Protocols

The VX5790 supports four trigger protocols: SYNC, ESTST (Extended STart/STop), SYNC100, and STAR Bus. The SYNC, SYNC100, and ESTST protocols are implemented using the TTLTRG* and ECLTRG lines, while the STAR Bus uses the STARX and STARY lines for triggering.

SYNC Protocol

SYNC protocol is a single source multi-acceptor trigger implemented by using one TTLTRG* or ECLTRG trigger line. The maximum repetition rate using a TTLTRG* line is 12.5 MHz, and the repetition rate while using an ECLTRG line is 62.5 MHz.

ESTST Protocol

Figure 3-2 illustrates the VXIbus resources and their relative timing used to implement ESTST protocol. While ESTST is not precisely a triggering function, its purpose of ESTST protocol is to be able to start or stop the operation of an instrument or group of instruments synchronously with the rising edge of the CLK100 clock signal. As shown in Figure 3-2, the SYNC100 signal identifies a unique clock edge to indicate when the start or stop is to be instigated. One of the ECLTRG trigger lines is designated for the instruments controlled by the ESTST protocol. This ECLTRG line is used as a qualifier to allow multiple groups of instruments within the same subsystem to be independently started or stopped. A qualifier is required for each group of instruments since both CLK100 and SYNC100 are shared resources, common to all instrument modules.

The combination of CLK100, SYNC100, and the designated ECLTRG line transmit an specified instant of time to the instrument modules for start or stop. A designated TTLTRG* line indicates the start or stop state. When this line is asserted (low) it indicates the start state; when unasserted (high) the stop state is indicated.

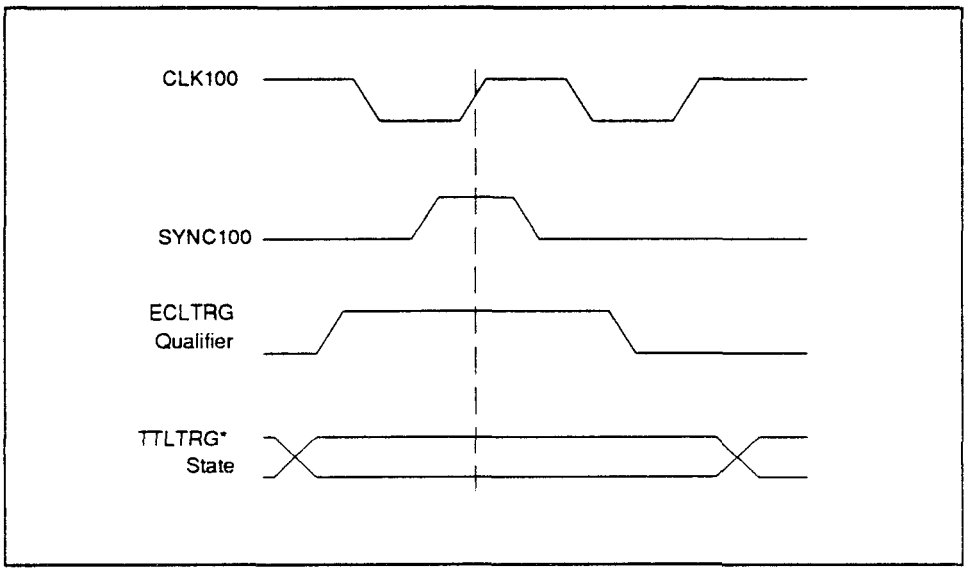


Figure 3-2. ESTST Protocol

SYNC100 Protocol

The SYNC100 protocol is a subset of the ESTST protocol allowing synchronous triggering of multiple instrument modules. Figure 3-3 illustrates how SYNC100 protocol uses the same resources as ESTST protocol, with the exception of using a TTLTRG* line to indicate the state. In the SYNC100 protocol, CLK100, SYNC100, and an ECLTRG qualifier combine to transmit a point in time to a group of instruments, causing them to trigger synchronously.

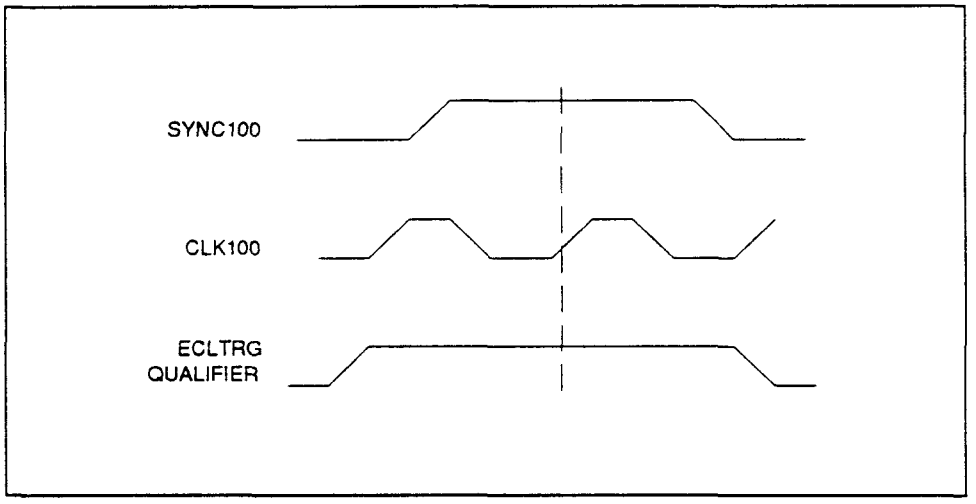


Figure 3-3. SYNC100 Protocol

STAR Bus

The STAR Bus is made up of two bidirectional differential ECL lines (STARX and STARY) connected to each module slot in the VXI Mainframe. The STAR configuration can be described as a 12-point star with the Slot 0 Controller in the center.

The STARX and STARY lines have a maximum timing skew of 2 ns between any STAR signals, and a maximum delay of 5 ns between the Slot 0 Controller and any module. Thus, the STAR Bus can route high speed trigger signals to any module in the subsystem.

Section 4
Rear-Panel Interface

Introduction

The AWG rear-panel interface consists of VXIbus connectors P1, P2, and P3. Those pins used by the AWG on P1 and the inner row (Row B) on P2 are configured as defined in the *VMEbus Specification, IEEE Standard 1014*. Those pins used by the AWG on the outer rows of P2 (Rows A and C) and P3 are configured as defined in the *VXIbus System Specification, Revision 1.3, dated July 14, 1988*. The pin assignments used by the AWG on these three connectors are listed in Tables 4-1, 4-2, and 4-3. For detailed information regarding the signals on the pins, refer to the *VMEbus Standard* and the *VXIbus Specification* referenced above.

Table 4-1. P1 Pinout
(nc = no connection)

Pin Number	Row A Signal Mnemonic	Row B Signal Mnemonic	Row C Signal Mnemonic
1	D00	nc	D08
2	D01	nc	D09
3	D02	ACFAIL*	D10
4	D03	BG0IN*	D11
5	D04	BG0OUT*	D12
6	D05	BG1IN*	D13
7	D06	BG1OUT*	D14
8	D07	BG2IN*	D15
9	GND	BG2OUT*	GND
10	nc	BG3IN*	SYSFAIL*
11	GND	BG3OUT*	BERR*
12	DS1*	nc	SYSRESET*
13	DS0*	nc	nc
14	WRITE*	nc	AM5
15	GND	nc	A23
16	DTACK*	AM0	A22
17	GND	AM1	A21
18	AS*	AM2	A20
19	GND	AM3	A19
20	IACK*	GND	A18
21	IACKIN*	nc	A17
22	IACKOUT*	nc	A16
23	AM4	GND	A15
24	A07	IRQ7*	A14
25	A06	IRQ6*	A13
26	A05	IRQ5*	A12
27	A04	IRQ4*	A11
28	A03	IRQ3*	A10
29	A02	IRQ2*	A09
30	A01	IRQ1*	A08
31	nc	nc	nc
32	+5V	+5V	+5V

Table 4-2. P2 Pinout
(nc = no connection)

Pin Number	Row A Signal Mnemonic	Row B Signal Mnemonic	Row C Signal Mnemonic
1	ECLTRG0	+5V	nc
2	-2V	GND	nc
3	ECLTRG1	nc	GND
4	GND	nc	-5.2V
5	nc	nc	nc
6	nc	nc	nc
7	-5.2V	nc	GND
8	nc	nc	nc
9	nc	nc	nc
10	GND	nc	GND
11	nc	nc	nc
12	nc	GND	nc
13	-5.2V	+5V	-2V
14	nc	nc	nc
15	nc	nc	nc
16	GND	nc	GND
17	nc	nc	nc
18	nc	nc	nc
19	-5.2V	nc	-5.2V
20	nc	nc	nc
21	nc	nc	nc
22	GND	GND	GND
23	TTLTRG0*	nc	TTLTRG1*
24	TTLTRG2*	nc	TTLTRG3*
25	+5V	nc	GND
26	TTLTRG4*	nc	TTLTRG5*
27	TTLTRG6*	nc	TTLTRG7*
28	GND	nc	GND
29	nc	nc	nc
30	MODID	nc	GND
31	GND	GND	nc
32	nc	+5V	nc

Table 4-3. P3 Pinout
(nc = no connection)

Pin Number	Row A Signal Mnemonic	Row B Signal Mnemonic	Row C Signal Mnemonic
1	ECLTRG2	+24V	nc
2	GND	-24V	nc
3	ECLTRG3	GND	nc
4	-2V	nc	+5V
5	ECLTRG4	-5.2V	nc
6	GND	nc	GND
7	ECLTRG5	+5V	-5.2V
8	-2V	GND	GND
9	nc	+5V	nc
10	nc	nc	nc
11	nc	nc	nc
12	nc	GND	nc
13	nc	nc	nc
14	nc	nc	nc
15	nc	+5V	nc
16	nc	nc	nc
17	nc	nc	nc
18	nc	-2V	nc
19	nc	nc	nc
20	nc	nc	nc
21	nc	GND	nc
22	nc	nc	nc
23	nc	nc	nc
24	nc	+5V	nc
25	nc	nc	nc
26	nc	nc	nc
27	GND	GND	GND
28	STARX+	-5.2V	STARY+
29	STARX-	GND	STARY-
30	GND	-5.2V	-5.2V
31	CLK100+	-2V	SYNC100+
32	CLK100-	GND	SYNC100-

Section 5
Introduction to Programming

Introduction

This section contains introductory information about the programming commands for the VX5790. The actual command dictionaries are located in: *Section 6, Instrument Control Command Dictionary*; *Section 7, System Command Dictionary*; and *Section 8, ID-ROM Information*. Commands follow the formats and conventions established by the *ANSI/IEEE Std 488.2 — 1987, IEEE Standard Codes, Formats, Protocols, and Common Commands.*

Command Notation

Command format descriptions and descriptive tables use the following standard notation:

< >	Delimits a required, defined element to be provided by the user.
[ ]	Delimits an optional element.
[ ]...	Delimits an optional element that may be omitted, or repeated one or more times .
{ }	Delimits a group of elements from which one must be selected.
|	Delimits exclusive-OR selectable elements.
::=	Means "is defined as".
: (colon)	Compound Command Program Header Separator used to separate "device names" from Command/Query Program Headers, and to separate Compound Command Headers.
; (semicolon)	Program Message Unit Separator used to separate functional parts of compound commands.
, (comma)	Program Data Separator used to separate compound Program Data elements.

Program Message

A VXIbus Program Message is a sequence of zero or more Program Message Units separated by a Program Message Unit separator, the semicolon (;). A Program Message Unit represents a single command message, programming data, or a query message intended for a device such as the VX5790. A Command Message represents a single command or programming data sent from the external (or embedded) controller through an interface device to the VXIbus device, such as the VX5790. A Query Message represents a single query sent over this same path to the VX5790.

Each Program Message Unit contains a series of syntactic elements, which may include a Device Name, a Command/Query Program Header (the command) and the associated Program Data elements (the command parameters), or multiple Program Message Units separated by semicolons (;). For a more detailed explanation about Program Messages, refer to the ANSI/IEEE Std 488.2-1987.

After the automatic execution of its initial power-up and configuration sequence, the VX5790 relies on Program Messages received over the VXIbus to control its on-going operations. These messages may originate from an external controller, then be transferred through a serial (RS232) or IEEE 488 (GPIB) interface to a VXIbus Interface Device, such as a Tektronix VX5520 Slot 0 Resource Manager, which finally transfers the messages to the VX5790 over the VXIbus. Messages also may originate from an embedded controller (installed in the VXIbus Mainframe), which would transfer all messages to the VX5790 directly over the VXIbus. Each message is parsed by the Resource Manager to determine its destination, or action to be taken.

Note

Explanations in this section assume that messages are transferred to the VX5790 through a Tektronix VX5520.

Messages sent to the VX5790 through its commander (using either the AUX PORT or IEEE 488 port are of the form:

```
:<device name>:<command>
```

such as:

```
:DEV06:INST:AMP 12
```

where:

:DEV06 is the *< device name>* ; the rest of the message is the command and its parameters.

When the VX5790 is installed in another manufacturer's mainframe, the interface device in that mainframe, and the mainframe, must provide the command path and format from the external (or embedded) controller to the VX5790.

ASCII Code

The 7-bit ASCII code is used for the majority of syntactic elements and semantic definitions. However, 8-bit ASCII code is allowed in special block messages, such as Arbitrary Block Program Data.

Syntactic Delimiters

Syntactic elements in a Program Message Unit are delimited with colons, white space, commas, or semicolons.

Colons (:) typically delimit "<device names>" and device-dependent Command/Query Program Headers, such as in the following message:

```
:DEV06:INST:AMP 12
```

where *:DEV06* is the *< device name>* ; *:INST:AMP* is the Command Program Header.

White Space (defined below under the White Space topic) typically delimits device-dependent Command/Query Program Headers from Program Data, such as reserved word command names in a command set from device-dependent command arguments. In the example above, *:INST:AMP* is the Command Program Header and *12* is the Program Data.

Commas (,) typically delimit multiple Program Data elements (arguments).

Semicolons (;) typically delimit multiple Program Message Units and compound Command Program Header tree structures in commands suited for compound program headers.

White Space

White space is essential to a program, but is not a part of the program. Typically, it is used to separate certain syntactic elements in a command. In the VX5790, white space is defined the same as in the ANSI/IEEE Std 488.2-1987, which is as a single ASCII-encoded byte in the range ASCII 0-32 (decimal), except ASCII 10, which is the Line Feed (LF) or Newline (NL) character.

Special Characters

The VX5790 firmware defines Special Characters as the Line Feed (LF) or Newline (NL) character (ASCII 10), and all characters in the range ASCII 127 — 255. These characters may be used in Arbitrary Block Program Data without consequence; however, when used in any other context, the outcome is unpredictable.

Program Data

The types of Program Data recognized by the VX5790 are:

- Decimal Numeric
- Non-Decimal Numeric
- String
- Arbitrary Block

Numeric Program Data Value Range

The range of acceptable Program Data numeric values is limited to that allowed by a 32-bit signed binary number. If out-of-range values are entered, an error message is generated, and execution of the command, or the multiple message unit containing the error, is terminated.

Decimal Numeric Program Data

Decimal Numeric Program Data is defined the same as in ANSI/IEEE Std 488.2-1987, which is as three numeric representations NR1, NR2, and NR3. When any one of the three is acceptable, the flexible representation NRf is specified. The general description of NRf data is shown below.

Type	Format	Examples
NR1	Implicit-point (Integer)	1, +3, -2, +10, -20
NR2	Explicit-point-unscaled (Fixed Point)	1.2, +23.5, -0.15
NR3	Explicit-point-scaled (Floating Point)	1E+2, +3.36E-2, -1.02E+3

If NR1 is specified as the input value format, NRf is acceptable. Query response is NR1 if the input is NR1, otherwise it is NR2.

Non-Decimal Numeric Program Data

Non-Decimal Numeric Program Data is defined the same as in ANSI/IEEE Std 488.2-1987, which is as the three alternate radix representations: hexadecimal, octal, and binary. These are represented as follows:

Hexadecimal

#Hd..d

where:	H	is H or h.
	d	is 0..9, A..F, or a..f

Octal

#Qd..d

where:	Q	is Q or q.
	d	is 0..7.

Binary

#Bd..d

where:	B	is B or b.
	d	is 0 or 1

String Program Data

String Program Data, sometimes referred to as "string literals", "literals", or just "strings", is defined as:

```
"[<non-double quote character>]..."
```

such as:

"This is a string constant."

or

```
"0..127"
```

To include a double quote character in the string, insert an additional double quote character ahead of the double quote character in the string. For example, the string:

```
serial number "B010000"
```

would be sent as:

```
"serial number ""B010000"""
```

String constants may be of any length, up to the memory limits of the VX5790.

Arbitrary Block Program Data

Arbitrary block program data is defined as:

```
#<byte count digit><byte count>[<contiguous 8-bit data
byte>]...
```

where:

<byte count digit >::= a non-zero digit in the range ASCII 1-9 that defines the number of digits (bytes) in the *<byte count>* field that follows.

<byte count>::= any number of digits in the range ASCII 0-9 that define how many bytes are in the *<contiguous 8-bit data byte>* field that follows.

<contiguous 8-bit data byte>::= a *<byte count>* number of 8-bit bytes in the range ASCII 0-255 that define the message. Each byte defines one character.

An example is:

```
#16AB4ZLT
```

Command Formats

All commands follow the Functional Element Syntax formats for Command Message Units and Query Message Units as defined by *ANSI/IEEE Std 488.2-1987*. These formats are defined to have Simple Command Program Headers, Simple Query Program Headers, Compound Command Program Headers, Compound Query Program Headers, Common Command Program Headers, and Common Query Program Headers, which are further described with examples in the paragraphs that follow.

Figure 5-1 is a flowchart of the structure of the AWG commands and queries. While the figure shows the general structure and possible combinations, text in the following pages gives more specific command and query descriptions and examples.

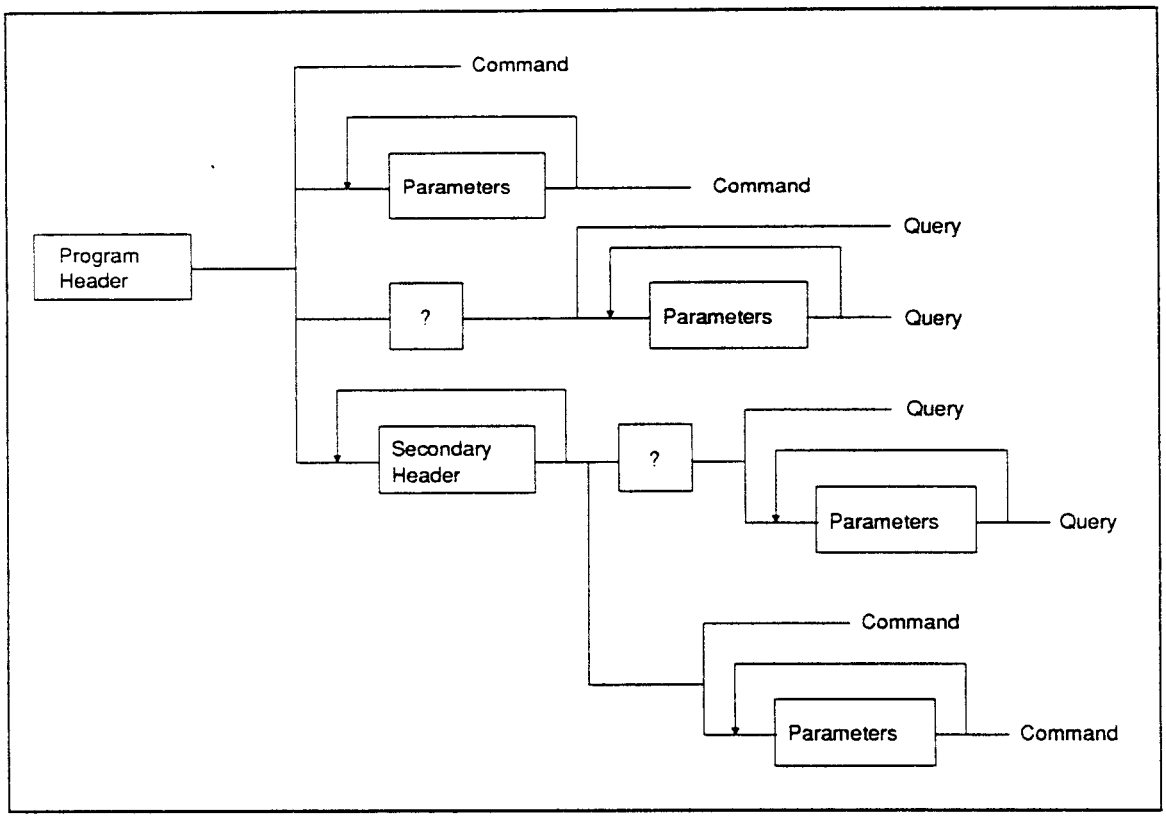


Figure 5-1. Command/Query Flowchart

Simple Command Program Header

A command with a Simple Command Program Header is one containing only one Program Mnemonic, or only one Program Mnemonic plus Program Data. Its message format is:

```
<Program Mnemonic> [<Program Data>]
```

such as:

```
CMDTRG
```

or

```
HEADER ON
```

Simple Query Program Header

A command with a Simple Query Program Header is one that contains only one Program Mnemonic followed by a question mark (?). Its message format is:

```
<1st Program Mnemonic>? [<Program Data>]
```

such as

```
ID?
```

Compound Command Program Header

A command with a Compound Command Program Header is one that contains multiple Program Mnemonics plus Program Data. Its message format is:

```
<1st Program Mnemonic>[:<2nd Program Mnemonic>]
```

or:

```
<1st Program Mnemonic>[:<2nd Program Mnemonic>] <Pro-
gram Data>
```

such as:

```
FUNC:TYPE SIN
```

or:

```
INST:AMP 12
```

The Compound Command Program Header is in the same format as embedded addresses. An example (when the VX5790 is used with a VX5520) is:

```
:DEV06:INST:AMP 12
```

where *DEV06* is the device name assigned to the VX5790 by the VX5520 during power-up configuration. When this message is received by the VX5520, it strips off the destination address (*:DEV06*) and sends the remaining message to the VX5790.

Compound Query Program Header

A command with a Compound Query Program Header is one containing multiple Program Mnemonics followed by a question mark (?). Its message format is:

```
<1st Program Mnemonic>:<2nd Program Mnemonic>? [<Pro-
gram Data>]
```

such as:

```
INST:FILTER?
```

or:

```
PAC:SHAPE?
```

Common Command Program Header

A command with a Common Command Program Header is one that precedes its Program Mnemonic with an asterisk (*). Its message format is:

```
<1st Program Mnemonic> [<Program Data>]
```

such as

```
*RCL 3
```

Common Query Program Header

A command with a Common Query Program Header is one that precedes its Program Mnemonic with an asterisk (*), and follows it with a question mark (?). Its message format is:

```
<1st Program Mnemonic>? [<Program Data>]
```

such as:

```
*TST?
```

Multiple-Command Format With Same 1st Program Mnemonic

The multiple-command message format for commands having the same 1st Program Mnemonic is:

```
<1st Program Mnemonic>:<2nd Program Mnemonic> [<Pro-
gram Data>] [;<2nd Program Mnemonic> [<Program
Data>]]...
```

such as:

```
INST:FILTER K390;AMP 0.5;OFF 10
```

In this multiple command, the *<1st Program Mnemonic>* is *INST*; the first *<2nd Program Mnemonic>* is *FILTER* ; the second *< 2nd Program Mnemonic>* is *AMP*, and the third *<2nd Program Mnemonic>* is *OFF*.

Case Sensitivity

The command set parser is insensitive to case; therefore, command mnemonics or Program Data may be entered in either upper-case or lower-case characters. Responses are returned in all uppercase, except for string data, which is returned in the same case as entered. For ease of recognition, command mnemonics, predefined parameter words (such as : ON and OFF), and decimal number formats (such as: NR1, NRf, etc.) are capitalized in the command set. Where the user must provide a parameter, the parameter name is set off with left and right angle symbols (<>), and have lower-case parameter names such as : <length>. String Program Data is case sensitive.

Command Guidelines

Before describing the details of the commands and the command structure, some background information is in order. The following paragraphs provide such information on the two kinds of memory (waveform data and program) and on the command structure.

Waveform Data Memory

Waveform data memory (WDM) is one of the two AWG memories. It is reserved for storage of waveform data; its capacity is 128K 10-bit words. Figure 5-2 shows an example of the makeup of a waveform, and Figure 5-3 shows the memory organization. Memory terms are defined in Table 5-1.

The WDM is partitioned into waveform data packets that are stored sequentially in memory using a dynamically partitioned format. The user allocates the memory while defining packets. This technique allows the user to alter the contents of packets freely without fear of losing other data, and requires very little memory management on the part of the user.

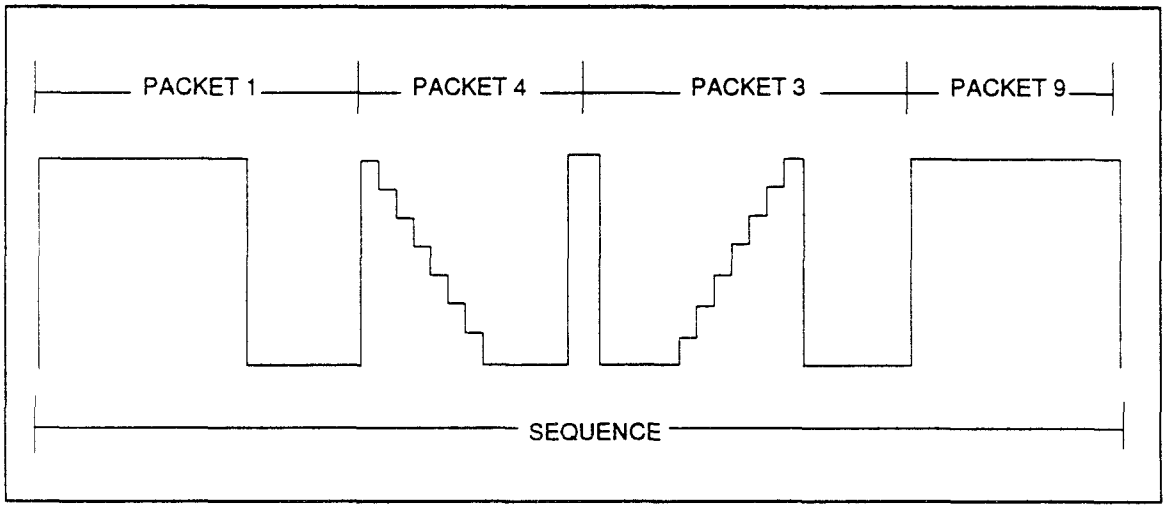


Figure 5-2. Waveform Makeup

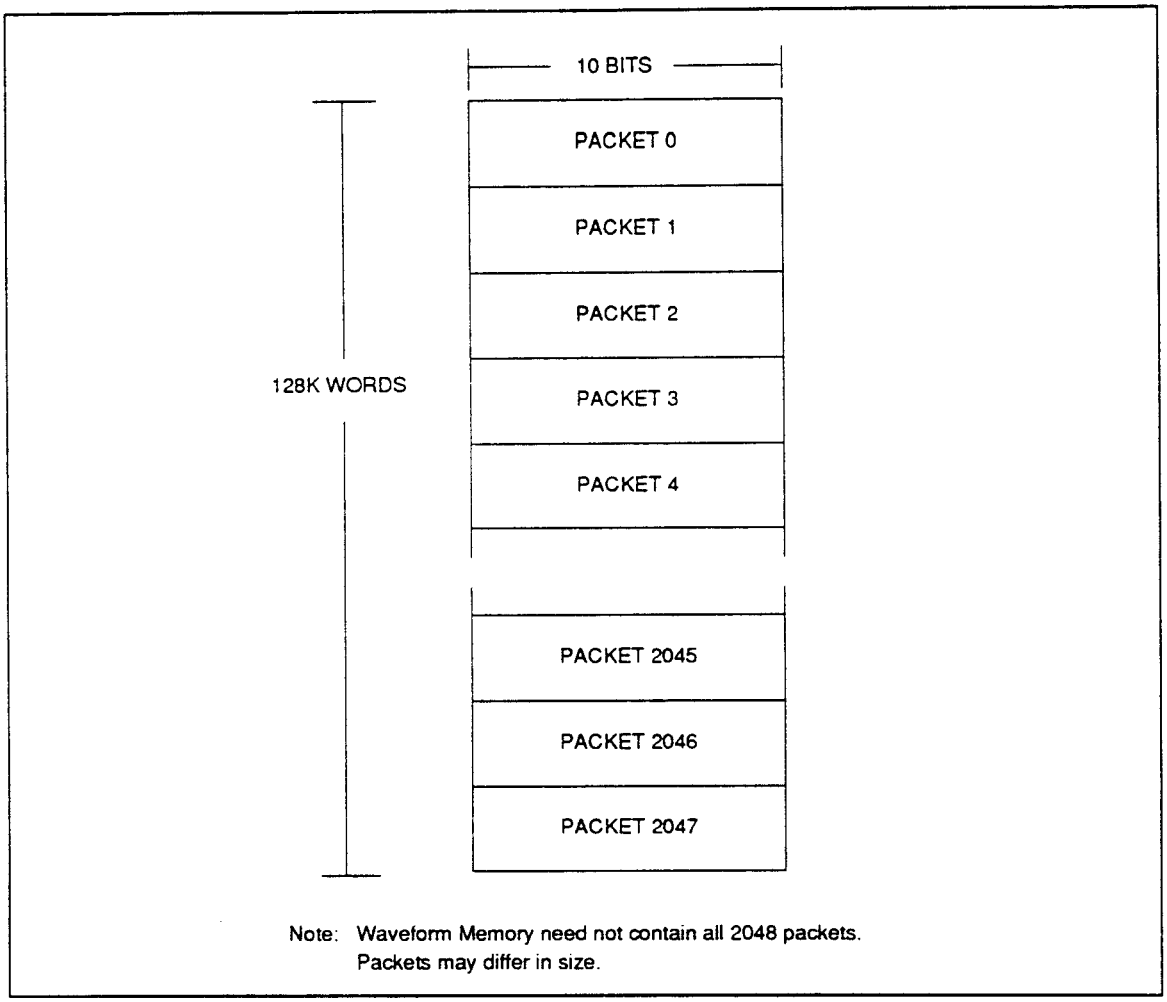


Figure 5-3. Waveform Memory Organization

Table 5-1. Waveform Memory Term Definitions

Term	Definition
data point	A 10-bit binary word that describes the amplitude at a specific point in a waveform.
packet	A block of binary data that describes a segment of a waveform; it consists of as few as 24 and as many as 2048 data points, with a step size of 8. The maximum number of packets is 2048.
sequence	A sequential list of packets, and the number of times each packet is to be scanned before proceeding on to the next packet.
8 point block	The smallest increment in a packet, consisting of 8 data points.

Program Memory

Program memory (PM) is the second of the two memories. It serves as a multipurpose storage area for data structures selected by the AWG user. Its capacity is 96K bytes. The memory terms are defined in Table 5-2, while Figure 5-4 shows the memory organization.

Of the 96K bytes allocated for user access, the space is initially divided into three major blocks of information: instrumentation setup, sequence information, and packet information. This allows for as many as 15 instrument setups, 7000 packet references distributed over 200 sequences and 2048 packets. All information in the blocks is addressable, objects (items) may be added or deleted, and all information (objects or characteristics of objects) is available for queries and changes.

Also note that all compound commands have a query opposite. The compound command *SEQ:INDEX <number>* selects that sequence. The query *SEQ:INDEX?* would thus yield the index of the currently selected sequence. For example, if sequence 3 were selected the response would be *SEQ:INDEX 3.*

Although the command set includes a query opposite for most commands, there are a few logical exceptions. For instance, the command *SEQ* does not exist. The query *SEQ?* yields all the information pertaining to the currently selected sequence. This is somewhat complex because an index number may be included with the command (for example, *SEQ? 3*). In such an instance, all of the characteristics held in the database for that particular object (sequence number 3) are then requested, but the object is not selected.

The previous examples describe the command structure semantics and illustrate how to examine and modify various objects. The only required facility not described is that of deletion. For this, the user can "zero" the data (i.e., *SEQ:ORDER #10* — see Section 6 for details).

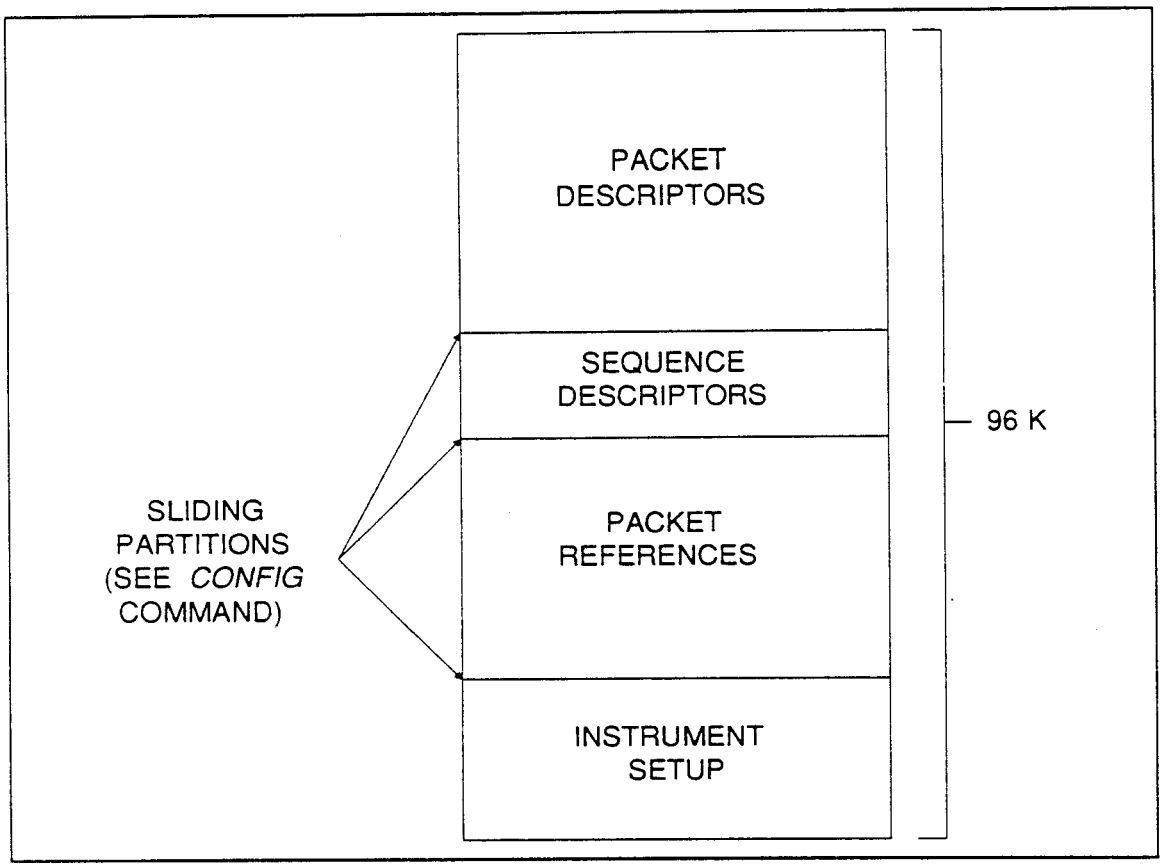


Figure 5-4. Program Memory Organization

Table 5-2. Program Memory Term Definitions

Term	Definition
packet index	The numerical identifier assigned to a packet; a number in the range 0 — 2047.
scan	One pass through a packet; a packet may be scanned as many as 256 times in a sequence.
sequence	An ordered group of packet references to be scanned for output.
sequence index	The numerical identifier assigned to a sequence; a number in the range 0 — 2047.
object	An item in a block of information; for example, a sequence, a packet description in a sequence list, etc.
marker	An electronic identifier that can be applied to a word within a packet within a sequence to allow for timing generation of an output trigger. If a packet is repeated (rescanned or re-referenced) the output trigger will be regenerated.
trigger out	A trigger output signal that occurs at the leading edge of the marker.
packet reference	An element of a sequence composed of a packet index and the number of times that packet is to be scanned.

Command Types

There are two command types described in the command dictionary sections: VX5790 Commands and System Commands.

Command Dictionary Layout

The Command Dictionary for the VX5790 firmware has a layout similar to many command dictionaries, but with some differences to accommodate the device specific capabilities of the VX5790. The following paragraphs describe the elements included in each Command Dictionary entry.

Each command is listed alphabetically at the top of a new page.

Purpose — Provides a brief description of what the command does.

Syntax — Shows the command format as it must appear at the VXIbus input to the VX5790. The command name, and some common Program Data elements (e.g., ON, OFF, NONE, etc.), are shown in all capital letters for readability only, as the firmware treats upper-case and lower-case letters the same. Mandatory user-defined entries are listed in lower-case letters and are enclosed in angle brackets (<>). Where the command descriptions include both command and query forms, there may be a separate Command and Query header titles.

Parameters — Describes the parameters values, including their allowable ranges, and any required delimiters, such as commas.

Description — Provides additional details about the command.

Examples — Provides one or more examples of actual command use including input parameters and returned values.

Related Commands — Lists other commands that may perform either parallel functions on different data, or different functions on the same data.

Section 6
Instrument Control Command Dictionary

Introduction

Instrument control commands are used to control the primary instrument functions. They can initialize the instrument, change its function settings, query for its status, initiate its self-calibration, create or download waveform packet information, etc.

Alphabetical Command Listing

Following is an alphabetical listing and a purpose statement for each instrument control command:

CALMODE	Allows the user to set the accuracy versus time to best suit the application.
CALMODE?	Returns the state established by *CALMODE*.
CMDARM	Arms the AWG for waveform generation.
CMDCLK	Allows the user to single-step the sequencer clock.
CMDTRG	Manually triggers the generation of a waveform output.
CONFIG	Permits reconfiguration of instrument memory partitions.
CONFIG?	Queries the partitioning of the AWG memory.
CURVE	Permits downloading full waveforms.
CURVE?	Permits uploading full waveforms.
DESE	Permits the setting of the Device Event Status Enable Register.
DESE?	Reads the Device Event Status Enable Register.
EVENT?	Provides notification to a non-Tektronix controller that an error has occurred on the AWG.
EVMSG?	Provides error messages to a non-Tektronix controller when errors occur on the AWG.
EXEC	Provides the same functionality as a *PREPARE* command, followed by a *CMDARM* command, followed by a *CMDTRG* command.

FACTORY	Sets instrument to factory power-up defaults.
FUNC	Selects on-board waveform generation settings.
FUNC?	Queries on-board waveform generation settings.
HEADER	Turns the command response headers on or off.
HEADER?	Queries whether the command response headers are turned on or off.
HELP?	Provides a list of valid AWG commands.
ID?	Queries the device type, hardware level, and firmware level.
INST	Sets the instrument operation parameters.
INST?	Queries the current operation parameters.
MAX?	Enables the user to verify dc calibration.
MEMAVAIL?	Returns the amount of waveform memory available to the user.
MID?	Enables the user to verify dc calibration.
MIN?	Enables the user to verify dc calibration.
OUTPUT	Enables/disables signal output to the front panel BNC connector.
OUTPUT?	Queries the *OUTPUT* setting.
PAC	Selects waveform packet descriptions.
PAC?	Queries for current packet descriptions.
PREPARE	Performs any operations necessary to operate the VX5790 in the mode described by the *INST* command.
PULS	Selects on-board pulse generation settings.
PULS?	Queries on-board pulse generation settings.
PURGESEQ	Removes all packets used by a specified sequence.
REPORT	Enables/disables various report functions.
REPORT?	Queries for the current status of the *REPORT* functions.
SEQ	Selects waveform packet sequence settings.
SEQ?	Queries for current packet sequence settings.

STATE?	Determines whether the AWG is running or stopped, and, if stopped, what stopped it.
STOP	Halts waveform generation operations.
WAVFRM?	The response to this query contains the information received from both the *WFMPRE?* and *CURVE?* queries separately.
WFMPRE	Sets the waveform preamble information.
WFMPRE?	Returns the waveform generation preamble information.

Functional Command Listing

In this listing, the instrument control commands are separated into three major groups: Instrument, Sequence and Packet, and Special Purpose; as follows:

Instrument
CONFIG
CONFIG?
EXEC
FUNC
FUNC?
INST
INST?
MEMAVAIL?
OUTPUT
OUTPUT?
PREPARE
PULS
PULS?
REPORT
REPORT?
STATE?
STOP

Sequence and Packet
CMDARM
CMDCLK
CMDTRG
CURVE
CURVE?
PAC
PAC?
PURGESEQ
SEQ
SEQ?
WAVFRM?
WFMPRE
WFMPRE?

Special Purpose
CALMODE
CALMODE?
DESE
DESE?
EVENT?
EVMSG?
FACTORY
HEADER
HEADER?
HELP?
ID?
MAX
MIN
MID

CALMODE, CALMODE?

CALMODE, CALMODE?

Purpose: *CALMODE* allows the user to set the accuracy-versus-time that best suits the application.

CALMODE? returns the state established by *CALMODE*.

Command Syntax:

```
CALMODE <PM>
```

where *<pm>* is one of the following:

```
DIAGOFF
DIAGON
EXTREF
FAST
INTREF
HOLDREF
NOCAL
```

Command Parameters:

`DIAGOFF`

Disable diagnostic mode.

`DIAGON`

Enable diagnostic mode.

`EXTREF`

Use an external voltage reference applied to the front-panel CAL IN BNC connector.

`FAST`

Fastest but least accurate mode.

`INTREF`

Use internal voltage reference. (More accurate than *FAST* mode.)

`HOLDREF`

Prevents updating of the calibration and DAC constants.

`NOCAL`

Recalls the calibration constants used for the last calibration. (This parameter is used only for service and maintenance.)

Query Syntax:

```
CALMODE?
```

Query Parameters: None

Description: *CALMODE* allows the user to set the trade-off between the accuracy desired and the time. *CALMODE* also allows the AWG to be placed in diagnostic mode. When in calibration mode, calibration errors are not logged into the ID-ROM and the *CAL?* command allows the acceptance of arguments to set the hardware to a specific state.

CALMODE? returns the current state (as set by *CALMODE*).

Related Commands: `*CAL`

CMDARM **CMDARM**

Purpose: Arms the instrument for waveform generation.

Syntax: `CMDARM`

Parameters: None

Description: This command arms the AWG sequencing hardware for waveform generation according to the setups in the most recent *PREPARE* command. The first valid trigger occurring after the *CMDARM* command begins waveform generation.

Related Commands: `CMDTRG, INST:TRIGTYPE, PREPARE, TRIGGER, *TRG`

CMDCLK **CMDCLK**

Purpose: Allows the user to single-step the sequencer clock.

Syntax: `CMDCLK [pm1]`

Parameters: `pm1`

The number of clock cycles desired (number of steps). The default value is 1.

Description: The *CMDCLK* allows the user to step the sequencer clock by a specified number of steps. The sequencer clock stops after the number of steps specified in pm1 is reached. If no pm1 is specified, the *CMDCLK* command uses the default of 1 step.

Related Commands: None

CMDTRG **CMDTRG**

Purpose: Manually triggers the generation of a waveform output.

Syntax: `CMDTRG`

Parameters: None

Description: This command asynchronously triggers the instrument to enable generation of an output waveform.

Related Commands: `*TRG, TRIGGER`

CONFIG, CONFIG?

CONFIG, CONFIG?

Purpose: *CONFIG* reconfigures the instrument memory partitions.

CONFIG? queries the setting of the instrument memory partitions.

Command Syntax:

```
CONFIG <pm1>,<pm2>,<pm3>,<pm4>
```

Command Parameters:

`<pm1>`

Specifies the number of instrument setups.

`<pm2>`

Specifies the number of packets.

`<pm3>`

Specifies the number of sequences.

`<pm4>`

Specifies the number of packet references.

Query Syntax:

```
CONFIG?
```

Query parameters: None

Description: The *CONFIG* command reconfigures the AWG user-memory partitions for specific applications. The default memory partitions are as follows:

- 5 instrument setups (0 — 4)
- 2048 packets (0 — 2047)
- 4 sequences (0 — 3)
- 4096 packet references

The *CONFIG* command is used to allow tradeoffs in the positioning of these memory partitions.

Once the user determines the specific needs for a certain application, *CONFIG* is used to set up the desired partitions in the following manner:

The total configurable memory space is 87 Kbytes and each element in memory takes up a specific amount of space:

- 36 bytes are used for each instrument setup
- 3 bytes are used per packet
- 12 bytes are used per sequence
- 3 bytes are used per packet reference

Thus, the following formula applies to the memory usage:

(number of instrument setups x 36) + (number of packets x 3) + (number of sequences x 12) + (number of packet references x 3) = 89088

With this formula, the default configuration uses 18,660 bytes of memory.

Note

The VX5790 has storage for 131,072 data points. Since the minimum length of a packet is 24 points and the maximum length is 2048, there is a minimum of 64 and a maximum of 5461 packets if all waveform memory is to be used. The CONFIG command allows underutilization of the waveform memory, in that it will allow the user to specify any whole number value (in the range of 1 — 5461) for the number of packets. (The default is 2048.)

There is a minimum requirement of one sequence, one packet reference, one packet, and one instrument setup.

Since packet and sequence index numbers begin with 0, when the user specifies a number of packets, the packets will be numbered from 0 to 1 less than the number specified. (For example, if 64 is specified, the packets will be numbered from 0 — 63.) This is also true for saved instrument setups.

Related Commands: None

CURVE, CURVE? **CURVE, CURVE?**

Purpose:

CURVE permits downloading waveform data.

CURVE? permits uploading waveform data.

Command Syntax: `CURVE <#arbitrary block data>|<comma separated decimal data>`

Command Parameters: `<#arbitrary block data>`

Curve data in the format specified by the *PAC:ENCDG*, *SEQ:ENCDG*, or *WFMPRE:ENCDG* commands (with the *BIN* or *HEX* parameters.)

`<comma separated decimal data>`

Curve data specified by the *PAC:ENCDG ASC*, *SEQ:ENCDG ASC*, or *WFMPRE:ENCDG ASC* commands.

Query Syntax: `CURVE?`

Query Parameters: None

Description: These commands download/return the curve data for a waveform. The *CURVE* command will use the currently selected sequence as its sequence, and will sequentially use the selected packets, and those whose numerical indexes follow, as its own until the data is stored.

Any packet data or sequence data existing when a *CURVE* command is issued will be discarded.

Note

The user must ascertain that sufficient memory is available for the curve data before issuing the CURVE command. No attempt is made to remove existing packet data until the packets are actually being overwritten.

Related Commands: `PAC:ENCDG, PAC:INDEX, SEQ:ENCDG, SEQ:INDEX, WAVFRM?, WFMPRE, WFMPRE?, WFMPRE:ENCDG`

DESE, DESE? **DESE, DESE?**

Purpose: *DESE* controls the setting and reading of the Device Event Status Enable Register.

DESE? reads the Device Event Status Enable Register.

Command Syntax:

```
DESE <register data>
```

Command Parameters:

`<register data>`

An integer in the range of 0 — 255. When expressed in binary, this integer represents the bit values of the Device Event Status Enable Register.

Query Syntax:

```
DESE?
```

Query Parameters: None

Description: The Device Event Status Enable Register is an 8-bit register with bit definitions corresponding to the Standard Event Status Register. Bits in the Device Event Status Enable Register (set using the *DESE* command) allow reporting through the Standard Event Status Register. Bits not set in the Device Event Status Enable Register disable reporting of that event.

The response to the *DESE?* query is and integer in the range of 0 — 255. When this integer is expressed in binary form, it corresponds to the bit values of the Device Event Status Enable Register.

Related Commands: None

EVENT? **EVENT?**

Purpose: Recalls an error message for a non-Tektronix controller.

Syntax: `EVENT?`

Parameters: None

Description: This command is used to obtain error messages when the VX5790 is used with a non-Tektronix Resource Manager or controller. The message returned is not the complete error message but a message stating that a certain type of error occurred. To obtain the actual error message, use the *EVMSG?* query.

Related Commands: `EVMSG?`

EVMSG? **EVMSG?**

Purpose: Recalls an error message for a non-Tektronix controller.

Syntax: EVMSG?

Parameters: None

Description: This command is used to obtain error messages when the VX5790 is used with a non-Tektronix Resource Manager or controller. The message returned is the exact error message, as issued by the VX5790 firmware.

Related Commands: EVENT?

EXEC

EXEC

Purpose: Provides the same functionality as a *PREPARE* command, followed by a *CMDARM* command, followed by a *CMDTRG* command.

Syntax: EXEC

Parameters: None

Description: None

Related Commands: PREPARE, CMDARM, CMDTRG

FACTORY **FACTORY**

Purpose: Sets the AWG to factory power-up defaults.

Syntax: `FACTORY`

Parameters: None

Description: The FACTORY command returns the instrument to its initial power-up state (after VXIbus communications have been initialized by the Resource Manager).

Example: `FACTORY`

Related Commands: None

FUNC, FUNC? **FUNC, FUNC?**

Purpose: Selects on-board waveform creation settings.

Command Syntax:

```
FUNC
FUNC:<pm1>
```

where *<pm1>* is one of the following:

```
CLK <1E-9 - 100000000> (100000000)
ERR <0 - 1000> (0.001)
FREQ <1 x 10^-9 - 50000000> (100000)
MAXPT <24 - 131072> (1000)
MINSAM <2 - 131072> (50)
PHASE <number>(0)
TYPE {RAM|SQR|SIN|TRI} (SIN)
```

Command Parameters:

`CLK <1E-9 - 100000000>`

Specifies the clock frequency (in Hertz) that will be used for waveform generation. (Specifying a clock frequency other than 100 MHz requires an external clock source.)

`ERR <0 - 1000>`

The percentage of error that can be tolerated in the creation of the desired waveform. (Typically, the larger the tolerated error percentage, the shorter the amount of time needed to create the waveform. Creating a waveform at the maximum accuracy of the VX5790 can take up to 5 minutes.)

`FREQ <1 x 10^-9 - 50000000>`

Specifies the frequency (in Hertz) of the waveform.

`MAXPT <24 - 131072>`

Specifies the maximum number of waveform memory points to be used in the creation of the waveform.

`MINSAM <2 - 131072>`

Specifies the minimum number of sample points per cycle.

`PHASE <number>`

Specifies the phase shift (in degrees) of the desired waveform. (Specifying a phase shift larger than 360° converts to a modulo 360 value.)

`TYPE {RAM|SQR|SIN|TRI}`

Specifies the type of waveform to be generated. Four types are allowed: RAM (ramp), SQR (square), SIN (sine), and TRI (triangular or sawtooth).

Query Syntax:

```
FUNC?
FUNC? STATS
FUNC:<pm2>
```

where *<pm2>* is one of the following:

```
ACTCYC?
ACTFREQ?
ACTPTS?
ACTSAM?
CLK?
ERR?
FREQ?
PHASE?
MAXPT?
MINSAM?
TYPE?
```

Query Parameters:

ACTCYC?

Returns the actual number of cycles in the current waveform.

ACTFREQ?

Returns the actual frequency of the generated waveform.

ACTPTS?

Returns the actual number of waveform memory points used in the generation of the current waveform.

ACTSAM?

Returns the actual number of samples per cycle used in the generation of the current waveform.

CLK?

Returns the current setting of the waveform generation clock frequency.

ERR?

Returns the current tolerated error percentage.

FREQ?

Returns the frequency of the current waveform.

PHASE?

Returns the phase shift of the current waveform (normalized to a value between 0 — 360°).

MAXPT?

Returns the current number of waveform memory points.

MINSAM?

Returns the current number of samples per cycle.

TYPE?

Returns the type of the current waveform.

Description: The *FUNC* command generates a waveform according to the specifications given by its parameters. (If the *FUNC* command is used without parameters, the AWG generates a waveform using the current settings of the parameters.)

The *FUNC* command uses sequential packets, beginning with the current selected packet and sequence. *FUNC* sets a marker at the first point in the first packet.

Note

Attempting to generate a waveform with too many points per cycle (too many samples specified for the clock frequency used) will generate an error.

FUNC? without additional parameters returns the values of *CLK*, *ERR*, *FREQ*, *MAXPT*, *MINSAM*, *PHASE*, and *TYPE*.

FUNC? STATS returns the values of *ACTCYC*, *ACTFREQ*, *ACTPTS*, and *ACTSAM*.

Related Commands: SEQ:MARKWORD, SEQ:MARKPAC

Purpose: *HEADER* turns the command response headers on/off.

HEADER? queries whether the command response headers are turned on or off.

Command Syntax:

```
HEADER <ON(1)|OFF(0)> (1)
```

Command Parameters:

`<ON(1)>`

Turns headers on.

`<OFF(0)>`

Turns headers off.

Query Syntax:

```
HEADER?
```

Query Parameters: None

Description: The *HEADER* command turns the command headers in a response message on or off. When the command headers are turned *OFF*, only the parameters are returned.

The *HEADER?* query returns a *1* if the headers are on or a *0* if they are off.

Command Example: In the following example, the headers are turned ON:

```
HEADER 1
```

Query Example: In the following example, the query returns a *0*, indicating that the headers are turned *OFF*:

```
HEADER?
0
```

Related Commands: None

HELP? **HELP?**

Purpose: Provides a list of VX5790 commands.

Syntax: `HELP?`

Parameters: None

Description: None

Related Commands: None

INST, INST? **INST, INST?**

Purpose: *INST:<pm1>* selects instrument settings.

INST? or *INST:< pm2> ?* returns current instrument settings.

Command Syntax:

```
INST:<pm1>[;<pm1>]*
```

where *< pm1>* is one of the following:

```
ADV {PAC|SEQ|FREERUN} (FREERUN)
AMP <0.0125 -20> (0.1)
CLKSRC {INT|EXT(BNC)|STARX} (INT)
CYCLE {SINGLE (BURST)|CONT} (CONT)
DIV <0 - 24> (0)
FILTER {NONE|K390|M3|M25} (NONE)
HARDTRIG {EXT(BNC)|ECLTRG0,1,2,3,4, or 5|
          TTLTRG0,1,2,3,4,5,6,or 7>|ESTST|
          SYNC,<ECLTRG0,1,2,3,4, or 5>|STARY} (EXT)
OFF <-10 -+10> (0)
PROTO {ESTST,<ECLTRG0,1,2,3,4, or 5>,
       <TTLTRG0,1,2,3,4,5,6, or 7>|NONE} (NONE)
TERM {HIGH|LOW} (HIGH)
TRIGOUT {NONE|EXT (BNC)|STARY|ECLTRG0, 1, 2, 3, 4, or 5|
     TTLTRG0, 1, 2, 3, 4, 5, 6, or 7}
         (NONE)
TRIGTYPE {HARDTRG|CMDTRG|EITHER} (EITHER)
```

Command Parameters:

`ADV {PAC|SEQ|FREERUN}`

Selects the type of packet advance mode.

`PAC` causes waveform generation to stop at the end of each packet.
`SEQ` causes waveform generation to stop at the end of each sequence.
`FREERUN` causes waveform generation to proceed nonstop.

`AMP <0.0125 - 20>`

Selects the peak-to-peak output signal amplitude in volts.

`CLKSRC {INT|EXT|CLK100|STARX}`

Selects the instrument clock source.

`INT` selects the internal system clock from the VXIbus (100 MHz).
`BNC` selects an external clock (in the range of 0 — 25 MHz) from the front-panel EXT CLOCK BNC connector.
`CLK100` selects the CLK100 signal from the VXIbus (100 MHz).
`STARX` selects whatever clock is routed from the Slot 0 module via the STARX line.

CYCLE {SINGLE|CONT}

Selects how many passes are made through the waveform generation packet sequence.

SINGLE causes one pass through the waveform generation packet sequence.
CONT causes the waveform packet sequence to be repeated continuously until a STOP command is received.

DIV <0 – 24>

Selects the value of *n* for a 2^n clock divider setting in steps of 1. (n = 0 gives a clock frequency of 100 MHz, n = 24 gives a clock frequency of 5.96 Hz.)

FILTER {NONE|K390|M3|M25}

Selects an output waveform filter.

NONE selects no output filtering.
K390 selects a 390 kHz filter.
M3 selects a 3.1 MHz filter.
M25 selects a 25 MHz filter.

HARDTRIG {EXT(BNC)|ECLTRG0,1,2,3,4, or 5|
TTLTRG0,1,2,3,4,5,6, or 7|ESTST|
SYNC,<ECLTRG0,1,2,3,4, or 5>|STARY}

Selects the trigger source.

EXT or BNC selects an external trigger from the front-panel TRIG INPUT BNC connector.
ECLTRG0,1,2,3,4, or 5 selects the ECLTRG trigger line to be used.
ESTST uses Extended Start as a trigger.
SYNC,<ECLTRG0,1,2,3,4, or 5> selects the SYNC100 trigger protocol and the trigger line to be used.
STARY selects the STARY lines for triggering.
TTLTRG0,1,2,3,4,5,6, or 7 selects the TTLTRG trigger line to be used.

OFF <-10 – +10>

Selects an output signal offset in the range -10 — + 10 V.

```
PROTO {ESTST,<ECLTRG0,1,2,3,4, or 5>
                <TTLTRG0,1,2,3,4,5,6, or 7>|NONE}
```

Specifies the desired protocol.

ESTST,<ECLTRG0,1,2,3,4, or 5>,<TTLTRG0,1,2,3,4,5,6, or 7> selects extended start/stop trigger protocol, and selects the trigger lines to be used.

NONE is no protocol specified.

```
TERM {HIGH|LOW}
```

Specifies the load impedance connected to the OUTPUT BNC.

HIGH specifies a load impedance of 1MΩ.

LOW specifies a load impedance of 50Ω.

```
TRIGOUT {NONE|EXT (BNC)|STARY|ECLTRG0, 1, 2, 3, 4, or 5|
        TTLTRG0, 1, 2, 3, 4, 5, 6, or 7}
```

Selects a destination for the output trigger.

NONE selects no trigger out.

EXT or BNC selects only the external BNC connector.

STARY selects the STARY line and the external BNC connector.

ECLTRG0, 1, 2, 3, 4, or 5 selects a specified ECLTRG line and the external BNC connector

TTLTRG0, 1, 2, 3, 4, 5, 6, or 7 selects a specified TTLTRG line and the external BNC connector.

```
TRIGTYPE {HARDTRG|CMDTRG|EITHER}
```

Selects the origin of the input trigger.

HARDTRG allows triggering only hardware triggers to be accepted.

CMDTRG allows triggering only by *TRG/CMDTRG*.

EITHER allows triggering by command triggers to override hardware triggers.

Query Syntax: INST?
INST:<pm2>?

where < *pm2*> is one of the following:

ADV?
AMP?
CLKSRC?
CYCLE?
DIV?
FILTER?
HARDTRIG?
OFF?
PROTO?
TERM?
TRIGOUT?
TRIGTYPE

Query Parameters:

ADV?

Returns the type of packet advancing mode selected.

AMP?

Returns the selected amplitude for the output signal.

CLKSRC?

Returns the selected clock source.

CYCLE?

Returns the *CYCLE* selection.

DIV?

Returns the value of *n* for the 2^n clock divider setting.

FILTER?

Returns the selected filter value.

HARDTRIG?

Returns the trigger source selected.

OFF?

Returns the value of the output offset.

PROTO?

Returns the specified protocol.

`TERM?`

Returns the value of the OUTPUT BNC impedance.

`TRIGOUT?`

Returns the selected destination of the output trigger.

`TRIGTYPE?`

Returns the selected origin for the input trigger.

Description: When an *INST?* is entered, an error message is generated if the setup does not exist (that is, has not been stored).

INST? used without parameters returns all information listed under the separate query parameters..

Related Commands: None

MAX? **MAX?**

Purpose: This query allows the user to verify dc calibration.

Syntax: `MAX?`

Parameters: None

Description: This query sets the output voltage to its maximum value (according to the settings of *INST:AMP* and *INST:OFF*) using the following formula:

$$offset + \frac{amplitude}{2}$$

The return is the voltage value that should be measured on the output BNC connector.

Note

Floating point errors from internal calculations may be present (i.e., 0.499 instead of 0.500).

Related Commands: `INST:AMP, INST:OFF, MID?, MIN?`

MEMAVAIL? **MEMAVAIL?**

Purpose: Returns the amount of waveform memory available.

Syntax: `MEMAVAIL?`

Parameters: None

Description: The MEMAVAIL? query returns the following message:

`WFM` *`number1`* `;P_REF` *`number2`*

where :

number1 = the number of points available in the waveform memory.

number2 = the number of packet references available.

Related Commands: `CONFIG`

MID? **MID?**

Purpose: This query allows the user to verify dc calibration.

Syntax: `MID?`

Parameters: None

Description: This query sets the output voltage to midrange (offset) and returns the voltage value that should be measured across the output BNC connector.

Note

Floating point errors from internal calculations may be present (i.e., 0.499 instead of 0.500).

Related Commands: `INST:OFF, MAX?, MIN?`

MIN? **MIN?**

Purpose: This query allows the user to verify dc calibration.

Syntax: `MIN?`

Parameters: None

Description: This query sets the output voltage to the minimum value using the following formula:

$$offset - \frac{amplitude}{2}$$

The return is the voltage value that should be measured across the output BNC connector.

Note

Floating point errors from internal calculations may be present (i.e., 0.499 instead of 0.500).

Related Commands: `INST:AMP, INST:OFF, MAX?, MID?`

OUTPUT, OUTPUT?

OUTPUT, OUTPUT?

Purpose: Enables/disables the signal to the front panel BNC connector.

Command Syntax:

```
OUTPUT {ON(1)|OFF(0)}
```

Command Parameters:

`ON (1)`

Enables signal output to the BNC connector.

`OFF (0)`

Disables signal output to the BNC connector.

Query Syntax:

```
OUTPUT?
```

Query Parameters: None

Description: *OUTPUT?* returns the setting of the *OUTPUT* command. A 1 is returned for the *ON* state; a 0 for the *OFF* state.

Note

When the output is turned off, the OUTPUT BNC is internally terminated to 50 Ω.

Related Commands: None

PAC, PAC? **PAC, PAC?**

Purpose:

PAC downloads packets of waveform data into the instrument.

PAC? or *PAC:<pm2>?* returns the waveform data from a specific packet.

Command Syntax:

```
PAC:<pm1>
```

where *<pm1>* is one of the following:

```
ENCDG {HEX|BIN|ASC}(HEX)
INDEX <0 -5461> (0)
LEN <0 -2048> (0)
SHAPE <#arbitrary block data> (#10)
```

Command Parameters:

`ENCDG {HEX|BIN|ASC}`

Specifies the data encoding desired. *HEX* specifies hexadecimal, *BIN* specifies binary, and *ASC* specifies ASCII characters.

`INDEX <0 - 5461>`

Selects an index number for the packet being created. The exact range depends upon the configuration set by the *CONFIG* command.

`LEN <0 - 2048>`

Specifies the length of the packet in number of points, with a step size of 8.

`SHAPE <waveform list>`

Downloads packet waveform data.

< waveform list> ::= An arbitrary fixed-length binary block of data

Query Syntax:

```
PAC?
PAC? <pm1>
PAC:<pm2>?
```

where *<pm2>* is one of the following:

```
ENCDG?
INDEX?
LEN?
SHAPE?
```

Query Parameters: <pm1>

Specifies a packet number. Returns the values of the *ENCDG*, *INDEX*, *LEN*, and *SHAPE* parameters for that packet.

<pm2>

ENCDG?

Returns the current data encoding.

INDEX?

Returns the index number of the currently selected packet.

LEN?

Returns the length of the packet in number of points.

SHAPE?

Requests the packet waveform data of the currently selected packet.

Description: The *PAC?* query command returns a complete description of the currently selected packet. If a packet specified by the *PAC?<pm1>* query does not exist (that is, has an index out of bounds), a warning message is generated.

The packets data contains a minimum of 24 and a maximum of 2048 data points. Each data point is two bytes long.

INDEX <number> — Results in the packet requested being selected. *<number>* is NR1 formatted with a step size of 1.

INDEX? — Produces the response pac:index <number> (with headers *ON*).

SHAPE <waveform list> — Permits the user to download a data list as packet waveform data to the AWG waveform memory.

SHAPE? — Permits the user to recall a packet waveform data list.

Related Commands: CONFIG, SEQ:ENCDG, WAVFRM?, WFMPRE:ENCDG

PREPARE **PREPARE**

Purpose: Performs any operations necessary to operate the VX5790 in the mode described by the *INST* command.

Syntax: `PREPARE`

Parameters: None

Description: None

Related Commands: `CMDTRG, CMDARM, INST`

PULS, PULS?

PULS, PULS?

Purpose: Sets pulse parameters.

Command Syntax:

```
PULS:<pm1>
```

where *<pm1>* is one of the following:

```
CLK <1E-9 -100000000> (100000000)
DUTY <0 -100> (50)
ERR <0 -1000> (0.001)
FTIME <seconds> (0)
MAXPT <24 -131072> (1000)
MINSAM <2 -131072> (50)
RTIME <seconds> (0)
WIDTH <number> (0.00005)
```

Command Parameters:

`CLK <1E-9 - 100000000>`

Specifies the clock frequency (in Hertz) that will be used for waveform generation. (Specifying a clock frequency other than 100 MHz requires an external clock source.)

`DUTY <0 - 100>`

The duty cycle of the waveform (see Figure 6-1).

`ERR <0 - 1000>`

The percentage of error that can be tolerated in the creation of the desired waveform. (Typically, the larger the tolerated error percentage, the shorter the amount of time needed to create the waveform. Creating a waveform at the maximum accuracy of the VX5790 can take up to 5 minutes.)

`FTIME <seconds>`

The fall time of the waveform (see Figure 6-1).

`MAXPT <24 - 131072>`

Specifies the maximum number of waveform memory points to be used in the creation of the waveform.

MINSAM <2 - 131072>

Specifies the minimum number of sample points per cycle.

RTIME <seconds>

The rise time (in seconds) of the waveform (see Figure 6-1).

WIDTH <number>

The pulse width in seconds. (see Figure 6-1).

Query Syntax:

```
PULS?
PULS? STATS
PULS:<pm2>
```

where *<pm2>* is one of the following:

```
ACTCYC?
ACTPTS?
ACTSAM?
ACTWIDTH?
CLK?
DUTY?
ERR?
FTIME?
MAXPT?
MINSAM?
RTIME?
WIDTH?
```

Query Parameters:

ACTCYC?

Returns the actual number of cycles in the current waveform.

ACTPTS?

Returns the actual number of waveform memory points used in the generation of the current waveform.

ACTSAM?

Returns the actual number of samples per cycle used in the generation of the current waveform.

ACTWIDTH?

Returns the actual pulse width of the generated waveform.

CLK?

Returns the current setting of the waveform generation clock frequency.

DUTY?

Returns the duty cycle of the waveform (see Figure 6-1).

ERR?

Returns the current tolerated error percentage.

FTIME?

Returns the fall time of the waveform (see Figure 6-1).

MAXPT?

Returns the current number of waveform memory points.

RTIME?

Returns the rise time of the waveform (see Figure 6-1).

WIDTH?

Returns the pulse width of the waveform (see Figure 6-1).

Description: The *PULS?* query used with no additional parameters returns the *CLK*, *DELAY*, *DUTY*, *ERR*, *FTIME*, *MAXPT*, *MINSAM*, *RTIME* and *WIDTH* values.

Using the *PULS? STATS* command returns the values of *ACTCYC*, *ACTPTS*, *ACTSAM*, and *ACTWIDTH*.

Related Commands: SEQ:MARKPAC, SEQ:MARKWORD

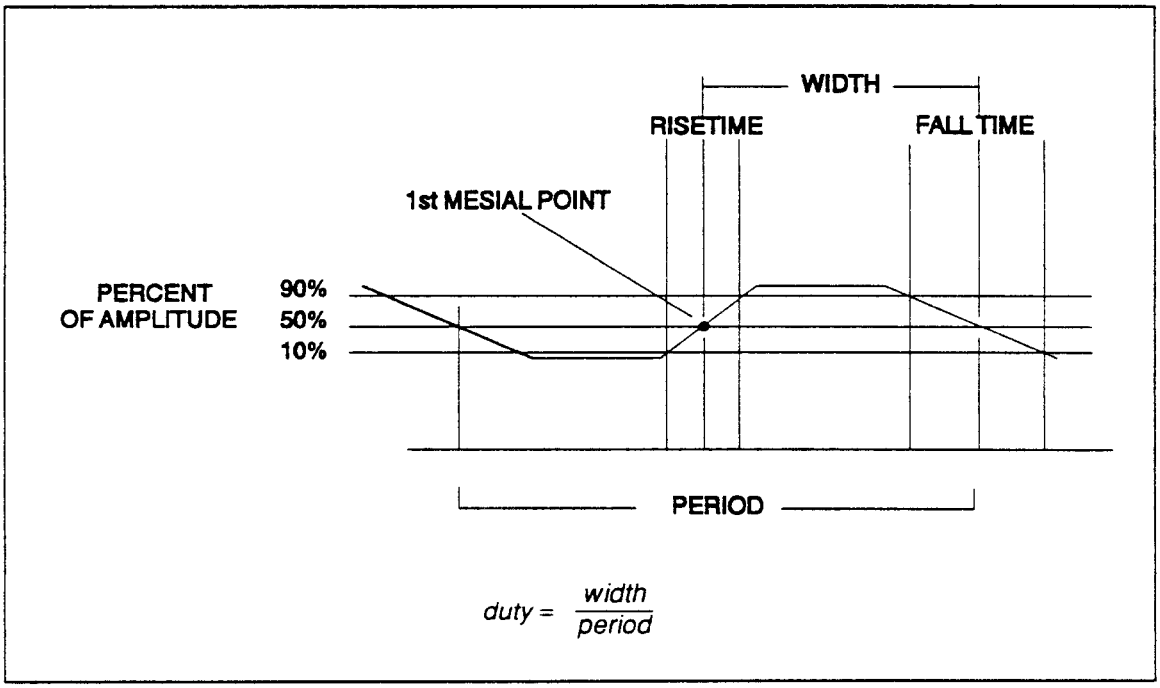


Figure 6-1. Pulse Definitions

PURGESEQ **PURGESEQ**

Purpose: Removes all packets related to a specific sequence.

Syntax: `PURGESEQ [pm1]`

Parameters: `pm1`

The sequence number where the packets are to be removed.

Description: If no *pm1* is specified, the *PURGESEQ* command will remove the packets from the current sequence.

Note

The PURGESEQ command does not check to see if a packet being purged is used in any additional sequences other than the one specified.

Related Commands: `SEQ:INDEX`

REPORT, REPORT? **REPORT, REPORT?**

Purpose: *REPORT enables or disables various error, event, and warning report functions.*

REPORT? or *REPORT:< pm2> ?* requests report settings.

Command Syntax: `REPORT:<pm1>`

where *<pm1>* is one of the following:

```
CMDERR <state> (ON)
CMDEVENT <state> (ON)
CMDWARN <state> (ON)
DEVERR <state> (ON)
DEVEVENT <state> (ON)
DEVWARN <state> (ON)
EXEERR <state> (ON)
EXEVENT <state> (ON)
EXWARN <state> (ON)
```

Command Parameters: In the following parameter descriptions, *<state>* can be either: *0* or *OFF* to disable the report, or, *1* or *ON* to enable the report.

`CMDERR <state>`

Enables or disables command error reporting.

`CMDEVENT <state>`

Enables or disables command event reporting.

`CMDWARN <state>`

Enables or disables command warning reporting.

`DEVERR <state>`

Enables or disables device error reporting.

`DEVEVENT <state>`

Enables or disables device event reporting.

`DEVWARN <state>`

Enables or disables device warning reporting.

`EXEERR <state>`

Enables or disables command execution error reporting.

`EXEVENT <state>`

Enables or disables command execution event reporting.

`EXWARN <state>`

Enables or disables command execution warning reporting.

Query Syntax:

```
REPORT?
REPORT:<pm2>
```

where *<pm2>* is one of the following:

```
CMDERR?
CMDEVENT?
CMDWARN?
DEVERR?
DEVEVENT?
DEVWARN?
EXEERR?
EXEVENT?
EXWARN?
```

Query Parameters:

`CMDERR`

Requests command error reporting status.

`CMDEVENT`

Requests command event reporting status.

`CMDWARN`

Requests command warning reporting status.

`DEVERR`

Requests device error reporting status.

`DEVEVENT`

Requests device event reporting status.

`DEVWARN`

Requests device warning reporting status.

`EXEERR`

Requests command execution error reporting status.

`EXEVENT`

Requests command execution event reporting status.

`EXWARN`

Requests command execution warning reporting status.

Command Description: When the reporting mode is set to *off*, no response is generated. When it is set to *on*, the results are as shown in the parameter definitions.

Query Description: If *REPORT?* is used without parameters, it requests all nine of the report settings (command error, event, and warning; device error, event, and warning; and command execution error, event, and warning), and produces a response for each.

If any optional *<pm2>* is selected, this command requests the report setting for that function only.

Related Commands: None

SEQ, SEQ? **SEQ: SEQ?**

Purpose: *SEQ* creates, selects, requests, or downloads packet information.

SEQ? or *SEQ:< number> ?* requests sequence description information.

Command Syntax:

```
SEQ:<pm1>
```

where *<pm1>* is one of the following:

```
ENCDG {HEX|BIN|ASC} (HEX)
INDEX <number> (0)
LEN <0 -2047> (0)
MARKPAC <0 -maximum number of packets> (0)
MARKWORD <0 - (length of packet -1)> (0)
ORDER <packet and scan list> (#10)
```

Command Parameters:

`ENCDG {HEX|BIN|ASC}`

Specifies the data encoding desired. *HEX* specifies hexadecimal, *BIN* specifies binary, and *ASC* specifies ASCII characters.

`INDEX <number>`

Creates and/or selects a sequence.

< number> ::= The index of the sequence description object in the list; the valid range is 0 — 100.

`LEN <0 - 2047>`

Specifies the number of packet references desired.

`MARKPAC <0 - maximum number of packets>`

Used to set a trigger out marker at a specific packet index in the sequence.

`MARKWORD <0 - (length of packet -1)>`

Specifies which word in a packet will receive the trigger out marker specified by *MARKPAC*.

`ORDER <packet and scan list>`

Downloads packet and repetition sequence list.

< packet and scan list> ::= An arbitrary block of packet sequencing data.

Query Syntax:

```
SEQ?
SEQ:<pm2>
```

where *<pm2>* is one of the following:

```
ENCDG?
INDEX?
LEN?
MARKPAC?
MARKWORD?
ORDER?
```

Query Parameters:

`ENCDG?`

Returns the type of data encoding currently specified.

`INDEX?`

Returns the number of the currently selected sequence.

`LEN?`

Returns the current number of packet references.

`MARKPAC?`

Returns the number of a packet index where a trigger out marker is currently set.

`MARKWORD?`

Returns the number of the word in the packet specified by *MARKPAC* where the trigger out marker is located.

`ORDER?`

Returns the current packet and repetition sequence list.

Description: `SEQ?` requests all information about the selected sequence.

`INDEX <number>` — Results in one of two operations, with the selection dependent upon whether the sequence object (specified to by *<number>*) exists. If the sequence does not exist, it is created, selected, and initialized. If the sequence already exists, it is selected. *<number>* is NR1 formatted with a step size of 1.

`ORDER <waveform list>` — Permits the user to download a block of data consisting of a list of packets and the number of times each packet will be scanned before advancing to the next packet in the sequence. The list includes a packet identifier (two bytes) and the number of scans (one byte) for each packet in the sequence.

Related Commands: `INST:TRIGOUT`

STATE? **STATE?**

Purpose: Reports whether the VX5790 is running or stopped.

Syntax: `STATE?`

Parameters: None

Description: The STATE? query will return one of four replies: *RUNNING*, *ESTST_STOP*, *NORMAL_STOP*, or *STOPMODE_STOP*.

RUNNING signifies that the AWG is currently generating a waveform.

ESTST_STOP signifies that the AWG has been halted by the ESTST trigger protocol.

NORMAL_STOP signifies that waveform generation has been completed, or the AWG has been halted by the *STOP* command.

STOPMODE_STOP signifies that the AWG has been halted by a hard trigger, or is awaiting a *CMDARM* command.

Related Commands: None

STOP **STOP**

Purpose: Terminates waveform generation operations.

Syntax: `STOP`

Parameters: None

Description: This command causes the AWG hardware to stop generating waveforms, however, the output signal remains on. The *CMDARM* command resets the instrument for further waveform generation.

Related Commands: None

WAVFRM? **WAVFRM?**

Purpose: The response to this query contains the information that would be received from both the *WFMPRE?* and *CURVE?* queries separately.

Syntax: `WAVFRM?`

Parameters: None

Description: *WAVFRM?* is a single query that combines the function and results of the *WFMPRE?* and *CURVE?* queries.

Related Commands: `CURVE?, WFMPRE?`

WFMPRE, WFMPRE? **WFMPRE, WFMPRE?**

Purpose: *WFMPRE* sets the waveform preamble information.

WFMPRE? or WFMPRE:<pm2>? returns the waveform generation preamble information.

Command Syntax: `WFMPRE:<pm1>`

where *<pm1>* is one of the following:

```
BIT_NR <number> (8)
BN_FMT {RI|RP} (RP)
BYT_NR <number> (1)
CRVCHK {NONE} (NONE)
ENCDG {BIN|HEX|ASC} (HEX)
PT_FMT {Y}
PT_OFF {0}
XINCR <seconds> (0.00000001)
XZERO <number> (0)
YMULT <number> (0.003906250)
YOFF <number> (0)
YZERO <number> (0)
```

Command Parameters:

`BIT_NR <number>`

The number of bits per data point. < *number*> = 1 — (8 x *BYT_ NR*).

`BN_FMT {RI|RP}`

Binary format (signed INT (RI), unsigned INT (RP)).

`BYT_NR <number>`

The number of bytes per data point (1 or 2).

`CRVCHK {NONE}`

NONE is the only allowable parameter.

`ENCDG {BIN|HEX|ASC}`

Enables data encoding in the specified format.

`PT_FMT`

Y is the only allowable parameter.

`PT_OFF`

0 is the only allowable parameter.

`XINCR <seconds>`

The time between samples.

`XZERO <number>`

The point of X origin (see Figure 6-2).

`YMULT <number>`

The Y amplitude point multiplier (see Figure 6-2).

`YOFF <number>`

The point of Y amplitude origin (see Figure 6-2).

`YZERO <number>`

The offset of the Y amplitude origin from 0 V (see Figure 6-2).

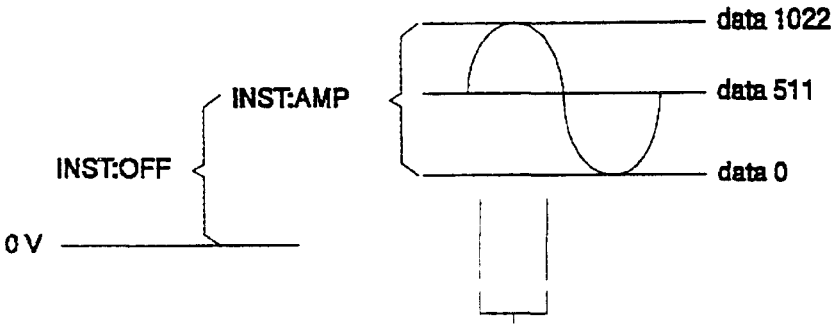
Query Syntax:

```
WFMPRE?
WFMPRE:<pm2>
```

where *<pm2>* is one of the following:

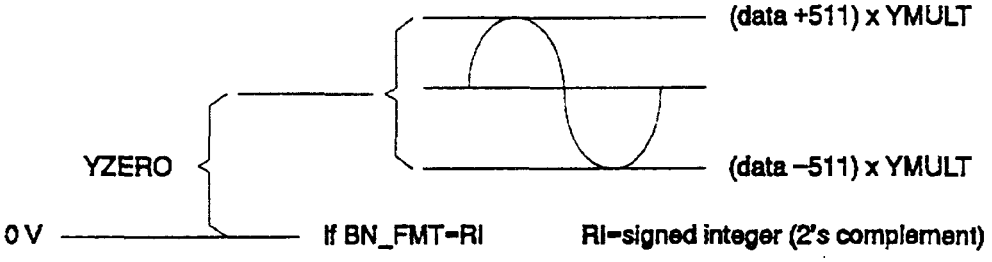
```
BIT_NR?
BN_FMT?
BYT_NR?
CRVCHK?
ENCDG?
PT_FMT?
PT_OFF?
XINCR?
XZERO?
YMULT?
YOFF?
YZERO?
```

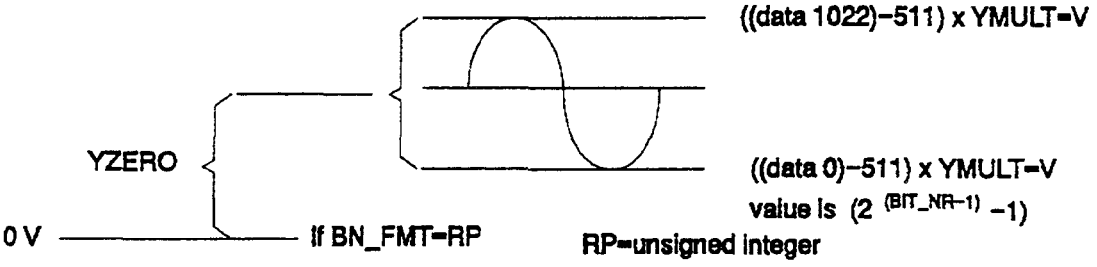
NATIVE INSTRUMENT



Sample rate set by INST:DIV and external/internal clock

TEK CODES AND FORMATS





Y_n = <XZERO value>+<XINCR value>(n –PT_OFF value>)
X_n = <YZERO value>+<YMULT value>(y_n–<YOFF value (0)>)

where n = the data point number (i.e., the 1st data point, n =0)

Figure 6-2. X and Y Parameters

Query Parameters:

BIT_NR?

Returns 1 — 8 x (*BYT_ NR*).

BN_FMT?

Returns "RI" or "RP".

BYT_NR?

Returns 1 or 2.

CRVCHK?

Returns "NONE" only.

ENCDG?

Returns BIN, HEX, or ASC.

PT_FMT?

Returns "Y" only.

PT_OFF?

Returns "0" only.

XINCR?

Returns the time in seconds between samples, using a clock frequency of 100 MHz and the current *DIV* setting.

XZERO?

Always returns 0 (see Figure 6-2).

YMULT?

Returns the *YMULT* value (see Figure 6-2).

YOFF?

Always returns 0 (see Figure 6-2).

YZERO?

Returns the value of the output offset (see Figure 6-2). (This is the same value as *INST:OFF*.

Description: The *WFMPRE* command allows the user to generate packets and sequences automatically.

Related Commands: INST:AMP, INST:DIV, INST:OFF

Section 7
System Command Dictionary

Introduction

Since the VX5790 is a VXI instrument rather than a GPIB instrument, it does not directly respond to GPIB commands except through the 488-VXIbus Interface Device. If the VX5790 is operated in an environment where the Host-to-VXI Subsystem interface is an IEEE 488.1 (GPIB) Interface, the 488-VXIbus Interface Device (such as the Tektronix VX5520) must be prepared to intercept and implement any IEEE 488.2 commands sent to the VX5790.

This subsection describes the IEEE 488.2 Common Commands and Low-Level VXIbus commands implemented by the VX5790. Commands prefixed with an asterisk (*) are IEEE 488.2 Common Commands. All others are classified as VXI Low-Level commands. These commands need to be read very closely, as an IEEE 488.2 Common Command received by the 488-VXIbus Interface Device may be implemented by sending an IEEE 488.2 Common Command.

When a VXIbus Interface Device (such as the Tektronix VX5520 Slot 0 Resource Manager) intercepts and implements the mandatory IEEE 488.2 Common Commands, it does so in one of the following ways:

- Accepts the command and rebroadcasts it to the VX5790.
- Accepts the command and processes it without sending it to the VX5790.
- Accepts the command and issues another command (may be a Word Serial Protocol, IEEE 488, Tektronix Instrument, or some other command) to the VX5790.

Table 7-1 indicates the command sent to the VX5790 by the VX5520 when it receives an IEEE 488 command. The command dictionary included later in this section describes the action taken by the VX5790 upon receipt of the command from the VX5520.

Table 7-1. IEEE 488 Commands Supported by the VX5790

IEEE 488 Command Sent to the VX5520	Command Sent over the VXIbus to the VX5790 by the VX5520
*CAL?	Rebroadcasts *CAL to the VX5790.
*CLS	Rebroadcasts *CLS.
*ESE	[1]
*ESE?	[1]
*ESR?	[1]
*OPC	Rebroadcasts *OPC.
*OPC?	Rebroadcasts *OPC?.
*PRC	[1]
*RCL	[1]
*RST	Rebroadcasts *RST.
*SAV	[1]
*SRE	[1]
*SRE?	[1]
*STB?	[1]
*TRG	Rebroadcasts *TRG.
*TST?	Rebroadcasts *TST? plus any arguments.

[1] IEEE 488.2 prohibits the prepending of a header to a common (*) command. Therefore, common commands targeted at a specific instrument can be sent to the VX5520 only through the *LOGADRRSEND* command.

The VX5520 commands supported by the VX5790, and the command communicated over the VXIbus to it, are listed in Table 7-2. The command dictionary included later in this section describes the action(s) taken by the VX5790 upon receipt of the IEEE 488 commands (prefixed with *), and the Tektronix WSP Halt command. Word Serial Protocol commands are implemented as defined in the VXIbus System Specification.

Table 7-2. VX5520 Commands Supported by the VX5790

Command Received by the VX5520	Command Sent over the VXIbus to the VX5790 by the VX5520
CLEAR "<device name>"	*CLS
DOWNGRADE "<device name>"	Sends ASCII DOWNGRADE message, which is executed in instrument firmware.
HALT "<device name>"	WSP *Clear* + WSP *Abort Normal Operations*
ID?	ID?
RESTART "<device name>"	WSP *Clear* + *RST
TRIGGER "<device name>"	
UPGRADE "<device name>"	Sends ASCII UPGRADE message, which is executed in instrument firmware.

To support the embedded addressing capability of the VX5520, each of the IEEE 488 Common Commands implemented by the VX5790 also has a version not prefixed with an asterisk (*). This non-prefixed version can be addressed directly to the VX5790 at any time. The action by the VX5790 is identical for both versions.

Alphabetical Listing

Following is an alphabetical listing and a purpose statement for each VX5790 IEEE 488.2 and Word Serial Protocol command.

***CAL?, CAL?** The response to this query depends on the setting specified by the previous *CALMODE* command.

***CLS, CLS** Clears instrument event registers and queues, except the output queue, and places a "1" in the output queue to indicate the **OPC?* idle state.

***ESE, ESE** Sets the Event Status Register bits.

***ESE?, ESE?** Queries the setting of the Event Status Enable Register bits.

***ESR?, ESR?** Reads the contents of the Event Status Enable Register.

***IDN?, IDN?** Query only. Returns the unique identification for the instrument.

***OPC, OPC** Causes the AWG to set the Operation Complete bit in the Standard Events Status Register when all operations are completed.

***OPC?, OPC?** Determines if all pending operations have been completed.

***PRC, PRC** **PRC* is a proposed IEEE 488.2 command that recalls and clears the saved setup values from non-volatile memory.

***RCL, RCL** Recalls instrument setup (settings made with the *INST* command only).

***RST, RST** Initializes the instrument to its power-up state (settings made with the *INST* command only), but doesn't alter the VXIbus interface data or change calibration constants.

***SAV,SAV** Saves current VX5790 state (settings made with the *INST* command only).

***SRE, SRE** Sets the bits in the Service Request Enable Register.

***SRE?, SRE?** Queries the bit settings of the Service Request Enable Register.

***STB?, STB?** Reads the Status Byte Register.

***TRG, TRG** Manually triggers the instrument asynchronously, overriding all other trigger sources.

***TST?, TST?** Causes the instrument to execute its power-up self-test routines, except the kernel verification tests, and reports any failures that occur.

***WAI, WAI**	Prevents the VX5790 from executing further commands or queries until the No-Operation-Pending flag is set.
CLEAR	A Word Serial Protocol command that clears the instrument's command parser, executes parsed commands, and clears the output queue, but doesn't change instrument settings.
DOWNGRADE	A Tektronix ASCII message sent from the VX5520 firmware to the instrument firmware to cause the instrument to change its communications protocol to Word Serial Protocol.
HALT	A Tektronix defined Word Serial Protocol command that asynchronously halts current operations.
ID?	An older Tektronix GPIB command that returns unique identification for the instrument.
RESTART	A Word Serial Protocol command that asynchronously halts current operations and restores all settings to default.
TRIGGER	A Word Serial Protocol command that provides a triggering signal to the VX5790 hardware.
UPGRADE	A Tektronix ASCII message sent from VX5520 firmware to the instrument firmware to cause the instrument to set its communications protocol to Shared Memory Protocol.

***CAL?, CAL?** ***CAL?, CAL?**

Purpose: The response to this query depends on the setting specified by the previous *CALMODE* command.

Syntax:
```
*CAL?
 CAL?
```

Parameters: None

Description: The response from this command depends on the state established by the *CALMODE* command (*DIAGON*/*DIAGOFF*).

If the AWG is in the Normal mode (*DIAGOFF*), the response will be the error code from the self-calibration routine. A zero (0) response indicates that the calibration was successful, while a non-zero response indicates that an error occurred and that the AWG output should be assumed to be out of specification.

If the AWG is in the Diagnostic mode (*DIAGON*), the response to the *CAL?* command will be as follows:

```
<amplitude spec>,<offset spec>,<impedance spec>,<output
spec>,<cal DAC 0V>,<cal DAC 10V>,<attenuation selected>,
<offset DAC value>,<gain DAC value>,<calmode>,<error code>
```

where:

`<amplitude spec>` is a floating point number from 0.0125 — 20.
`<offset spec>` is a floating point number from -10 — 10.
`<impedance spec>` is either 0 (high impedance) or 1 (low impedance).
`<output spec>` is either 0 (output off) or 1 (output on).
`<cal DAC 0V>` is the integer that, when loaded into the calibration DAC, produces a 0 V output.
`<cal DAC 10V>` is the integer that, when loaded into the calibration DAC, produces a 10 V output.
`<attenuation selected>` is a floating point number representing the user-selected attenuation.
`<offset DAC value>` is an integer that determines the offset for the current settings.
`<gain DAC value>` is an integer that determines the fine gain for the current settings.
`<calmode>` is 0 for internal reference; 1 for external reference; 2 for holding; 3 for fast; or 4 for nocal (see *CALMODE* command for further details).
`<error code>` is the error code (if any) from the calibration routines.

When in the diagnostic mode, the *CAL?* command will accept arguments in the form specified in the previous paragraph (*<amplitude spec>*, *<offset spec>*, etc.). When an argument is given, if a calibration error occurs for that argument, it will be stored in the ID-ROM in the object CEI (Calibration Error Information), and can be retrieved with the ID-ROM *GNV? CEI* command.

The calibration error codes are as follows:

`Error #8001` — Could not measure the 10 V reference.
`Error #8002` — If the *CALMODE* is *INTREF*, then the Digitizer is defective. If the *CALMODE* is *EXTREF*, then the voltage applied to the front panel CAL IN BNC connector out of the range of 8 — 12 V.
`Error #8003` — Could not measure the 0 V reference.
`Error #8004` — The 0 V reference is not 0 V.
`Error #8005` — Could not set the offset voltage.
`Error #8006` — Could not set the amplitude voltage.
`Error #8007` — AWG is in the Diagnostic mode; no self-calibration was performed.

Related Commands: `CALMODE`

***CLS, CLS** ***CLS, CLS**

Purpose: Clears all instrument event status registers, clears all queues, except the output queue, and places a "1" in the output queue to indicate the *OPC?* idle state.

Syntax:
```
*CLS
 CLS
```

Parameters: None

Description: The 488-VXIbus Interface Device issues this command whenever it receives a software *CLEAR* command.

Related Commands: None

***ESE, ESE, *ESE?, ESE?** ***ESE, ESE, *ESE?, ESE?**

Purpose: **ESE* sets the Event Status Enable Register bits.
**ESE?* queries the setting of the Event Status Enable Register bits.

Command Syntax:

```
*ESE <integer>
 ESE <integer>
```

Query Syntax:

```
*ESE?
 ESE?
```

Command Parameters: `<integer>`

An integer in the range of 0 — 255 that represents the desired setting of the Event Status Enable Register bits.

Query Parameters: None

Description: If an out-of-range integer is entered for the value of *<integer>* , an execution error is generated.

The **ESE?* query generates an NR1 integer response (in the range of 0 — 255) that corresponds to the bits set in the Event Status Enable Register.

Command Example: In the following example, the **ESE* command sets bits 1 and 2:

```
*ESE 3
```

Query Example: In the following example, the **ESE?* query determines that bits 1, 2, and 6 are set:

```
*ESE?
9
```

Related Commands: None

***ESR?, ESR?** ***ESR?, ESR?**

Purpose: Reads the contents of the Event Status Register.

Syntax:
```
*ESR?
 ESR?
```

Parameters: None

Description: The **ESR?* query clears the Event Status Register after reading. The response to this query is identical to that of **ESE?*

Example: In the following example, the **ESR?* query determines that bits 1, 2, and 6 are set:

```
*ESR?
9
```

Related Commands: None

***IDN?, IDN?** ***IDN?, IDN?**

Purpose: Query only. Returns the unique identification of the instrument.

Syntax:
```
*IDN?
 IDN?
```

Parameters: None

Description: This is the identify device query per the Tektronix Codes and Formats Standard. The information is reported in ASCII format and includes:

```
TEKTRONIX model number
Serial number
Firmware version number
```

This information is drawn from the VX5790 ID-ROM. Note that this is a query-only command; there is no associated statement-type command. For more detailed information on this command, refer to IEEE Standard 488.2.

Related Commands: None

*OPC, OPC, *OPC?, OPC?

*OPC, OPC, *OPC?, OPC?

Purpose: **OPC* causes the AWG to set the Operation Complete bit in the Standard Event Status Register when all operations are completed.

**OPC?* determines if all pending operations have been completed.

Command Syntax:
```
*OPC
 OPC
```

Command Parameters: None

Query Syntax:
```
*OPC?
 OPC?
```

Query Parameters: None

Description: When the **OPC?* query is issued, it remains in a waiting state until all operations are complete, then sets the Operation Complete bit in the Standard Event Status Register.

Related Commands: `*WAI`

***PRC, PRC** ***PRC, PRC**

Purpose: **PRC* is a proposed IEEE 488.2 command that recalls and clears the saved setup values from non-volatile memory.

Syntax:
```
*PRC
 PRC
```

Parameters: None

Description: The function of the **PRC* command is similar to that of the **RCL* command, but **PRC* reinitializes stored (*INST*) parameters after recalling the settings.

Related Commands: `*RCL, *RST, *SAV`

*RCL, RCL

*RCL, RCL

Purpose: Recalls instrument setup.

Syntax:
```
*RCL <number>
 RCL <number>
```

Parameters: <number>

A numeric value, 0 — 25 (Depending on the configuration set by the *CONFIG* command. The default is 5 setups, numbered from 0 — 4.).

Description: This command recalls the previously saved instrument setup identified by *<number>* . All information stated in *INST:<param>* commands is replaced with this recalled information. *<number>* is NR1 formatted with a step size of 1 and a range of 0 through 25.

Related Commands: CONFIG, *PRC, *RST, *SAV

***RST, RST** ***RST, RST**

Purpose: Resets the VX5790.

Syntax:
```
*RST
 RST
```

Parameters: None

Description: This command resets the VX5790 to the idle state, with no waveform being generated, and resets all instrument state variables to the default values. This command does not affect any packet or sequence data, or any stored instrument setups. However, selections do change. The packet, sequence, and instrument selections are changed to Packet 0, Sequence 0, etc.

Related Commands: `*PRC, *RCL, *SAV`

***SAV,SAV** ***SAV,SAV**

Purpose: Saves the current VX5790 state.

Syntax:
```
*SAV <number>
 SAV <number>
```

Parameters: `<number>`

A numeric value, 0 — 25 (depending on the configuration set by the *CONFIG* command); the destination location for the instrument setup information.

Description: This command saves the current state of the VX5790 (that is, all *INST:<param>* commands). Note that, when power is interrupted, all saved setups are lost. Setups are also lost when the VX5790 is reconfigured (that is, when the *CONFIG* command is used). *<number>* is NR1 formatted with a step size of 1 and a range of 0 through 25. It is user selectable; the default is no-save.

Related Commands: `CONFIG, *PRC, *RCL, *RST`

***SRE, SRE, *SRE?, SRE?** ***SRE, SRE, *SRE?, SRE?**

Purpose: **SRE* sets the bits in the Service Request Enable Register.

**SRE?* queries the bit settings of the Service Request Enable Register.

Command Syntax:
```
*SRE <number>
 SRE <number>
```

Command Parameters: <number>

Decimal numeric program data in the range of 0 — 255. This number, when rounded to an integer value and expressed in binary, represents the desired bit values of the Service Request Enable Register.

Query Syntax:
```
*SRE?
 SRE?
```

Query Parameters: None

Description: When using the **SRE* command, the value specified for Bit 6 will be ignored. (Bit 6 is always 0.)

The response to the **SRE?* query is an NR1 in the ranges of 0 — 63 or 128 — 191. When converted to binary, this number represents the current bit values of the Service Request Enable Register. Bit 6 of the Service Request Enable Register bit values will always be 0.

Related Commands: None

***STB?, STB?** ***STB?, STB?**

Purpose: Reads the status byte from the Status Byte Register.

Syntax:
```
*STB?
 STB?
```

Parameters: None

Description: The response to the **STB?* query is an NR1 that corresponds to the bits set in the Status Byte Register.

Example: In the following example, the **STB?* query indicates that bits 1 and 2 of the Status Byte Register are set:

```
*STB?
3
```

Related Commands: None

***TRG, TRG** ***TRG, TRG**

Purpose: Causes a trigger event to start waveform generation.

Syntax:
```
*TRG
 TRG
```

Parameters: None

Description: This asynchronous command causes a trigger pulse to be generated and sent to the waveform generation hardware. The trigger pulse can override the usual triggering operation. This command is the equivalent of the *CMDTRG* command.

Related Commands: `CMDTRG, TRIGGER`

***TST?, TST?** ***TST?, TST?**

Purpose: Executes power-up diagnostic routines.

Syntax:
```
*TST?
 TST?
```

Parameters: None

Description: This command causes the VX5790 to execute the internal diagnostic routines. (The diagnostic routines take ≈ 35 seconds to complete.)

Related Commands: None

***WAI, WAI** ***WAI, WAI**

Purpose: Prevents the VX5790 from executing further commands or queries until the No-Operation-Pending flag is set.

Syntax:
```
*WAI
 WAI
```

Parameters: None

Description: The *WAI command is used to prevent the VX5790 from overlapping or simultaneous executing separate commands. When the *WAI command is received, the VX5790 will not begin command execution of a following command until all current operations are completed.

Related Commands: `*OPC?`

CLEAR **CLEAR**

Purpose: A Word Serial Protocol command that clears the instrument's command parser and I/O communications registers.

Syntax: `CLEAR`

Parameters: None

Description: This command is received from the 488-VXIbus Interface Device when that device receives a **CLS* command, or an *SDC* or *DCL* control command over the GPIB. It also is received whenever the Slot 0 Module receives a software *RESTART* command addressed to the VX5790.

When the command is received, any command already parsed is executed; any command currently in, or waiting to be input to, the local parser is cleared; and the output queues are cleared. None of the instrument settings are changed.

Related Commands: `*CLS`

DOWNGRADE **DOWNGRADE**

Purpose: A Tektronix ASCII message sent by the VX5520 changing the VX5790 communications protocol from Shared Memory Protocol to Word Serial Protocol.

Parameters: None

Description: Some VXIbus modules have the capability of operating in either a Word Serial or a Shared Memory communications protocol (this capability is defined in the device's Protocol Register). When the VXIbus Slot 0 Resource Manager is a Tektronix VX5520, the communication capability of each device in the VXIbus system is known by the VX5520.

When the Tektronix VX5520 is the VXIbus Slot 0 Resource Manager, its power-up default communications protocol is Word Serial. For modules that have both protocols, the VX5520 provides an *UPGRADE* and *DOWNGRADE* command that can be used to change the communications protocol of specified devices. These command functions are implemented by Tektronix in the firmware of the VX5520 and other VXIbus modules. When either of these commands is received by the VX5520, and addressed to some Tektronix VXIbus module, the function is implemented by a message from the VX5520 firmware to the addressed device's firmware to change its communications protocol.

When the VX5520 receives a *DOWNGRADE* command that is addressed to the VX5790, the VX5520 firmware sends a message to the VX5790 firmware to change its communications protocol to Word Serial Protocol.

Related Command: `UPGRADE`

HALT **HALT**

Purpose: A Word Serial Protocol command causing an asynchronous halt to all current operations.

Syntax: `HALT`

Parameters: None

Description: This command is received from the Slot 0 Module whenever that module receives a *HALT* command addressed to the VX5790, or a *HALTALL* command through its interface communications channels (e.g., GPIB or RS232).

The command is processed asynchronously. If it is received during waveform generation, the generation is halted immediately regardless of the programmed sequence, and the data waveform memory segments will be in an unknown state. If it is received while the waveform generation hardware is idle, it has no effect.

Related Commands: None

ID? **ID?**

Purpose: Returns the unique identification of the instrument.

Syntax: `ID?`

Parameters: None

Description: This is an older Tektronix GPIB command used in previous software programs to query for the unique identification of an instrument. In newer instruments, it has been replaced with the IEEE 488.2 **IDN?* command. This command is implemented for compatibility with older Tektronix software. When the *ID?* query command is received by the VX5520 it is rebroadcast to the VX5790.

The returned value is:

```
ID TEK/<instrument name>,0,Vx.y
```

where *x.y* is the firmware version number.

Related Command(s): None

RESTART **RESTART**

Purpose: A Word Serial Protocol command that halts and restarts operations.

Syntax: `RESTART`

Parameters: None

Description: This command is issued by the Resource Manager to cause the VX5790 to stop all current operations and initiate a cold start. All data currently held by the VX5790 is lost, and the VXIbus command interface is cleared.

Related Commands: None

TRIGGER **TRIGGER**

Purpose: A Word Serial Protocol command providing a triggering signal.

Syntax: `TRIGGER`

Parameters: None

Description: This command is issued to the VX5790 by the Resource Manager to provide a triggering signal to the VX5790 hardware. The Resource Manager issues this command in response to a **TRG* or *GET* command from the system controller.

Note that the **TRG* command described above is the same as the VX5790 *TRG* command.

Related Commands: `*TRG, CMDTRG`

UPGRADE **UPGRADE**

Purpose: A Tektronix ASCII message sent by the VX5520 changing the VX5790 communications protocol from Word Serial Protocol to Shared Memory Protocol

Parameters: None

Description: Some VXIbus modules have the capability of operating in either a Word Serial or a Shared Memory communications protocol (this capability is defined in the device's Protocol Register). When the VXIbus Slot 0 Resource Manager is a Tektronix VX5520, the communication capability of each device in the VXIbus system is known by the VX5520.

When the Tektronix VX5520 is the VXIbus Slot 0 Resource Manager, its power-up default communications protocol is Word Serial. For modules having both protocols, the VX5520 provides an *UPGRADE* and *DOWNGRADE* command used to change the communications protocol of specified devices. These command functions are implemented by Tektronix in the firmware of the VX5520 and other VXIbus modules. When either of these commands is received by the VX5520, and addressed to some Tektronix VXIbus module, the function is implemented by a message from the VX5520 firmware to the addressed device's firmware to change its communications protocol.

When the VX5520 receives an *UPGRADE* command that is addressed to the VX5790, the VX5520 firmware sends a message to the VX5790 firmware to change its communications protocol to Shared Memory Protocol.

Related Command: DOWNGRADE

Section 8
ID-ROM (NVRAM) Information

Introduction

The ID-ROM is a general-purpose, non-volatile memory management scheme using NVRAM for storage of instrument/system specific information organized in data object cells.

NOTE

For diagnostic test purposes, the ID-ROM hardware is considered part of the instrument hardware, rather than part of the kernel.

ID-ROM Commands

When the ID-ROM was created, it was initialized with an *NNV* command. Then data objects were written into cells with the *SNV* command. When the data objects need to be used, they are read from the cells with a *GNV?* or *LNV?* command. Each command is described later in this section.

NOTE

The NNV command requires a password, which is available from the VX5790 Service Manual, Tektronix Customer Service, or your Tektronix Representative. This password is intended to prevent accidental erasure of the ID-ROM, rather than as a security measure. If you have the password, exercise caution when using it.

During the Power-up Kernel Test and Extended Self-Tests, error information is logged into the ID-ROM by the local processor. These errors can be read from the ID-ROM using the *STERR?* command.

Error Codes

When executing ID-ROM commands, certain nonspecific error codes may occur. Table 8-1 is a list of codes, with expanded meanings as pertains to the ID-ROM:

Table 8-1. ID-ROM Error Codes

Error Code	Error Message	ID-ROM Meaning
101	Invalid Character	ID-ROM Permission Denied
104	Data Type Error	ID-ROM Invalid Token
120	Numerical Data Error	ID-ROM Too Much Data
240	Hardware Error	ID-ROM Header Bad, Checksum Bad, or Memory Test Fail

Data Object Identifier Mnemonics

When used in an ID-ROM command, each data object is referenced by a 3-character, Tektronix assigned data object identifier mnemonic from Table 8-2.

Table 8-2. ID-ROM Data Object Identifier Mnemonics

Mnemonic	Accessibility	Description
ADJ	External	Component Last Adjusted
CCD	Internal	Calibration Constant Data (service use only)
CCM	Internal	Internal Calibration Constant (changed by self-calibration routine)
CDD	External	Calibration Due Date
CEI	Internal	Calibration Error Information
DLC	External	Date of Last Calibration
DLS	External	Date of Last Service
DOM	Protected	Date of Manufacture
ERR	Internal	Self-Test Errors
FAL	Internal	Number of Test Failures
HSI	External	Home System Identity
HSN	External	Home Slot Number
HWR	Protected	Hardware Revision Level
IDR	Protected	ID-ROM Device Type
IDS	Protected	ID-ROM Size (Bytes)
IND	Internal	Data Area During Power-Down

Accessibility Legend:

External - User may write data to these data objects at any time.

Internal - Only the local processor is allowed to update or write to these data objects.

Protected - Once data is written to these data objects (typically by Tektronix personnel), further writes are not allowed.

Table 8-2. ID-ROM Data Object Identifier Mnemonics (cont.)

Mnemonic	Accessibility	Description
INS	Internal	Settings During Power-Down
LLC	External	Location of Last Calibration
LLS	External	Location of Last Service
LMS	Protected	Local Memory Size
LPT	Internal	Local Processor Type
MID	Protected	Manufacturer Identity
MSN	Protected	Module Serial Number
MTY	Protected	Module Type
NAM	External	User's Name for Module
NSN	External	National Stock Number
OPT	External	Hardware Options
OWN	External	Owner's Name
PAS	Internal	Number of Self-Test Passes
PCC	Internal	Previous Calibration Constants
PER	Internal	Previous Self-Test Errors
PSC	External	Power Supply Capacity
PSL	External	Power Supply Loading
REP	External	Components Last Replaced
SCT	External	System Configuration Table
SDD	External	Service Due Date
SDT	Internal	Shutdown Delay Time
SER	External	Service Record
SPA	Internal	Serial Port Attributes
SSW	External	Soft Switches
STA	Internal	ID-ROM Wear-Out Status (%)
STD	External	NBS Standard Used for Calibration
STF	Internal	Self-Test Failure Record
TCC	Internal	Temperature Calibration Constant
TBN	External	TB Number
TMN	External	TM Number
UDS	External	User Defined Storage

Accessibility Legend:

External - User may write data to these data objects at any time.

Internal - Only the local processor is allowed to update or write to these data objects.

Protected - Once data is written to these data objects (typically by Tektronix personnel), further writes are not allowed.

Additional ID-ROM Contents

In addition to the data objects listed in Table 8-1, the ID-ROM also contains the following:

- A header address and test pattern that is used as a self-test/diagnostic tool to:
  - — determine the health of the data bus and address lines
  - — determine the correct byte ordering
  - — indicate that an ID-ROM is installed, initialized, and accessible.
- Values to indicate the size (in bytes) of the ID-ROM and each data object, which are used by internal routines.
- A power fail protection byte that normally contains a 0 until a write is performed to a data object. If the power fails during data object modification, this byte contains the code of the object that is probably corrupted.
- An ID-ROM Checksum for power-up error checking. A new checksum is generated after information is updated in the ID-ROM; however, the checksum is not updated if the current checksum is invalid.

Alphabetical List of ID-ROM Commands

Following is an alphabetical list of the ID-ROM commands and a brief description of each:

GNV? — Returns the contents of selected data object information from the ID-ROM.

LNV? — Returns a listing containing data object information for each data object having information in the ID-ROM.

NNV — Initializes the data structure of the ID-ROM.

SNV — Writes data into a specific data object in the ID-ROM.

STERR? — Selectively performs one of the following ID-ROM actions:

- — Returns the error codes and messages from the most recent self test
- — Returns the error codes and messages from a previously executed self test
- — Clears the ID-ROM error message buffer

GNV? **GNV?**

Purpose: Returns the data from the ID-ROM for a selected data object.

Syntax: `GNV? <data object id>[,<data object id>]*`

Parameters: `<data object id>`

A three character data object identifier mnemonic from Table 8-1.

Results: Returns a list of the data objects requested from the ID-ROM in the form:

```
<data object id>,<data>[,<data object id>,<data>]+
```

where *<data object id >* is one of the 3-character data object identifier mnemonics from Table 8-1, and *<data>* is the information for the data object.

Description: This command allows the user to query the ID-ROM for selected information data objects for the module.

Example: This example queries the VX5790 for its *MID* and *MTY* data objects (Manufacturer's Identity and Module Name), and shows the response.

```
GNV? mid,mty
Tektronix,VX5790
```

Related Commands: `SNV`

LNV? **LNV?**

Purpose: Returns a listing showing the data object identifier mnemonic and the size (in bytes) of the data in the ID-ROM.

Syntax: LNV?

Parameters: None

Results: Returns a list containing the data object identification mnemonic and its size (in bytes) in the following format:

```
<data object id>,<size>[,<data object id>,<size>]+
```

where *<data object id >* is one of the 3-character data object identifier mnemonics from Table 8-1, and *<size>* is an NR1 value indicating the number of bytes in the data object.

Description: The *LNV?* command provides the user a means to determine the size (in bytes) of the data for each data object in the ID-ROM. The amount of free space remaining in the ID-ROM can be calculated as follows:

```
<id-rom size>)-(3*(<# of data objects defined>)+
 (Σ <data object sizes>)+8
```

Example: This example queries the VX5790 for a list and the size of each data object stored in the ID-ROM, and shows part of a typical response.

```
LNV?
LNV ...,dom,7,err,100,per,100,mty,10,mid,7,ser,100
```

Related Commands: SNV

NNV **NNV**

Purpose: Initializes the data structure of the ID-ROM.

Syntax: `NNV <password>,<ID-ROM size>`

Parameters: `<password>`

A Tektronix assigned keyword. Rather than a security function, this keyword is used to prevent accidental erasure of the information written to the ID-ROM during manufacture. The *< password>* is available from the Service Manual, Tektronix Customer Service, or your Tektronix Representative.

`<ID-ROM size>`

An NR1 value in the range 16 — 65536 (in power-of-2 increments) indicating the number of bytes from the NVRAM being reserved for the ID-ROM.

Description: The NNV command allows the user to test and initialize a new NVRAM and assign ID-ROM space for use in a new, redesigned, or custom system.

Example: This example initializes the NVRAM and assigns 2048 bytes of it to ID-ROM.

```
NNV dummy,2048
```

Related Commands: `SNV`

SNV **SNV**

Purpose: Writes data into a specific data object in the ID-ROM.

Syntax: `SNV <data object id>[,<size>][,"<data>"]`

Parameters: `<data object id>`

A three character data object identifier mnemonic from Table 6-1.

`<size>`

An optional NR1 value representing the size (in bytes) to be reserved for the data object entry. When not included, *< size>* defaults to length of *< data>* .

`<data>`

An alphanumeric string representing the data object information.

Description: The SNV command allows the user to add data strings to individual data objects in the ID-ROM.

Example: The following example adds the data "Tektronix" to the *MID* data object (Manufacturer's Identity) in the ID-ROM.

```
SNV mid,20,"Tektronix"
```

Related Commands: `GNV?, LNV?`

STERR? **STERR?**

Purpose: Selectively performs one of the following ID-ROM actions:

- Returns the error codes and messages from the most recent self test
- Returns the error codes and messages from a previously executed self test
- Clears the ID-ROM error message buffer

Syntax: `STERR? [CLEAR|PREV]`

Parameters: `CLEAR`

An optional argument that clears all error codes and messages from the ID-ROM error message buffer.

`PREV`

An optional argument that returns self-test error codes and messages previously stored in the ID-ROM.

Description: The *STERR?* command without optional arguments allows the user to query the ID-ROM for the error codes and messages, if any, that resulted from the most current execution of the VX5790 self-test routines.

When the *CLEAR* argument is added to the command, all error codes and messages are cleared from the ID-ROM error code and message buffer.

When the *PREV* argument is added to the command, the error codes and messages from previous self-tests can be queried.

Example: The first example queries for current errors, while the second example queries for previous errors.

```
STERR?
8006, COULDNT SET AMPLITUDE

STERR? PREV
0, NONE
```

Related Commands: `TST?`

Section 9
Calibration

Introduction

The dc voltage accuracy of the VX5790 is determined by both hardware and operational firmware.

The instrument has a built-in 10 V reference that is used in conjunction with a calibration digitial-to-analog converter (DAC) and comparator to directly measure the output voltages. An external 10 V reference can be used to establish instrument calibration to an external reference.

The instrument self-calibration routines control the attenuators, output DAC, offset DAC, gain DAC and calibration DAC to achieve the output voltages specified by the user.

The operational firmware performing the calibration is invoked:

- On power-up
- Whenever the instrument is programmed for a different amplitude, offset or impedance level
- By the user in the form of the *CAL?* command.

The instrument will store temporary calibration constants with the instrument settings. These are volatile and will be lost when power is turned off. If calibration has been run on the current instrument settings and those settings have been stored, the temporary calibration constants will be stored as well. Thus, when the settings have been recalled, no calibration will be done, and the temporary cal constants will be reloaded into the hardware. Therefore, the user may prefer to load all the settings to be used, cal and save each of them and recall them as needed. This would eliminate the need for time consuming recalibrations for the same settings.

To obtain the best accuracy from the instrument, the user can perform a self-calibration (using the *CAL?* command) at any time. The *CALMODE* command allows the user to control the time versus accuracy trade-offs during instrument self-calibration.

Calibration Commands

The following commands relate to instrument calibration:

CALMODE

The purpose of this command is to allow the user to select the accuracy versus time trade-off that best suits the application. This command also allows the AWG to be placed in diagnostic mode. Refer to Section 6 for further information.

CALMODE?

This query returns the state established by the CALMODE command. Refer to Section 6 for further information.

CAL?

The response to this command is determined by CALMODE. Refer to Sections 6 and 7 for further information.

GNV?

This command extracts calibration error information from the ID-ROM. Refer to Section 8 for further information.

MAX?

This query helps the user to verify dc calibration. Refer to Section 6 for further information.

MID?

This query helps the user to verify dc calibration. Refer to Section 6 for further information.

MIN?

This query helps the user to verify dc calibration. Refer to Section 6 for further information.

The Calibration Algorithm

The calibration process is as follows:

1. Select approximate attenuation

   If the amplitude spec has changed since the last self-calibration, select the attenuators needed for the required output voltage.

2. Cal the Calibration DAC

   If *CALMODE* is not *HOLDREF* or *FAST*, find new values for the calibration DAC by measuring the 10 V point and the 0 V point. These values are used to compute a reference voltage for maximum output and midrange output for the adjustment of the amplitude and offset, respectively.

3. Adjust Offset

   Load the output and gain DACs to midscale. Load the calibration DAC with the value that represents the offset voltage that should result. Adjust the offset DAC until the output voltage is the same as the calibration DAC voltage.

4. Adjust fine gain DAC/Attenuation

   Hold the new value for the offset DAC. Load the output DAC to full scale output. Load the calibration DAC with the value that represents the offset voltage plus half the amplitude (i.e.: maximum output voltage). Adjust the gain DAC until the output voltage is the same as the calibration DAC voltage.

   This step may result in failure if the fine gain DAC runs out of adjustment. The action would then be to choose the next attenuator setting based on which direction the gain DAC exceeded its adjustment range. Choosing another attenuator is allowed until; minimum or maximum attenuation is reached, or, until a change in attenuation stepping direction is indicated.

5. Adjust Offset (Loop on Steps 4 and 5; five times to find best compromise)

   Adjust the offset DAC again as in Step 3 using the new value for the gain DAC. Repeat 5 times. Do not repeat if in the fast mode.

6. Manage Errors

   If there is a failure (and not in *DIAGON*) then save the results in the ID-ROM.

7. Return

   Report the status as determined by *CALMODE* (*DIAGON* | *DIAGOFF*)

   Return control to user

Possible Reasons for Poor Calibration

There are three general reasons for poor calibration:

- Hardware failure
- Calibration algorithm failure
- Verification error

The possible hardware failures are: reference voltage, calibration DAC, output DAC, gain DAC, offset DAC, attenuators, output relay and 1MΩ termination, power supplies, or a component failure that causes excessive interactions within the analog section.

The calibration algorithm could fail if sufficient noise interfered with the successive approximation method used to adjust the DACs.

Verification errors can include: faulty measurement device, improper termination, erroneous interpretation of the results, instrument operating at a temperature extreme, self-calibration has not been performed for an extended period of time.

Half-LSB Output Accuracy

The output DAC has 10 bits of resolution. The midpoint (half way between the maximum and minimum outputs) can never be accurately set if 1024 output levels are used, since the real midpoint is half way between 0X1FFh (511) and 0X200h (512). Thus, the error for the midpoint would be $\frac{amplitude}{half\ an\ LSB}$. For large amplitudes and small offsets, the offset error could be great.

The solution to this problem is to specify a "region of calibration" that has an odd number of levels. This is analogous to oscilloscopes where the graticule is the calibrated area, but the waveform can still be seen outside of the graticule. Thus, the region of calibration for the AWG will be from 0 — 1022 (one less than full scale), the midpoint being 511. Waveform generation software should consider this detail. The user is not prohibited from using the maximum level, thus it will still be a '10 bit' AWG.

The user will be able to obtain greater confidence in the accuracy of the AWG with this method, especially when measuring the offset (midpoint) error.

Appendix A
ASCII/GPIB Code Chart

B7 B6 B5 / BITS / B4 B3 B2 B1	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
	CONTROL		NUMBERS SYMBOLS		UPPER CASE		LOWER CASE	
0 0 0 0	0 NUL 0	20 DLE 16	40 SP 32 0	60 0 48	100 @ 64 0	120 P 80 16	140 ` 96 0	160 p 112 16
0 0 0 1	1 GTL SOH 1 1	21 LLO DC1 17	41 ! 33 1	61 1 49 17	101 A 65 1	121 Q 81 17	141 a 97 1	161 q 113 17
0 0 1 0	2 STX 2 2	22 DC2 18	42 " 34 2	62 2 50 18	102 B 66 2	122 R 82 18	142 b 98 2	162 r 114 18
0 0 1 1	3 ETX 3 3	23 DC3 19	43 # 35 3	63 3 51 19	103 C 67 3	123 S 83 19	143 c 99 3	163 s 115 19
0 1 0 0	4 SDC EOT 4 4	24 DCL DC4 20	44 $ 36 4	64 4 52 20	104 D 68 4	124 T 84 20	144 d 100 4	164 t 116 20
0 1 0 1	5 PPC ENQ 5 5	25 PPU NAK 21	45 % 37 5	65 5 53 21	105 E 69 5	125 U 85 21	145 e 101 5	165 u 117 21
0 1 1 0	6 ACK 6 6	26 SYN 22	46 & 38 6	66 6 54 22	106 F 70 6	126 V 86 22	146 f 102 6	166 v 118 22
0 1 1 1	7 BEL 7 7	27 ETB 23	47 ' 39 7	67 7 55 23	107 G 71 7	127 W 87 23	147 g 103 7	167 w 119 23
1 0 0 0	10 GET BS 8 8	30 SPE CAN 24	50 (40 8	70 8 56 24	110 H 72 8	130 X 88 24	150 h 104 8	170 x 120 24
1 0 0 1	11 TCT HT 9 9	31 SPD EM 25	51) 41 9	71 9 57 25	111 I 73 9	131 Y 89 25	151 i 105 9	171 y 121 25
1 0 1 0	12 LF A 10	32 SUB 1A 26	52 * 2A 42 10	72 : 3A 58 26	112 J 74 10	132 Z 90 26	152 j 106 10	172 z 122 26
1 0 1 1	13 VT B 11	33 ESC 1B 27	53 + 2B 43 11	73 ; 3B 59 27	113 K 75 11	133 [91 27	153 k 107 11	173 { 123 27
1 1 0 0	14 FF C 12	34 FS 1C 28	54 , 2C 44 12	74 < 3C 60 28	114 L 76 12	134 \ 92 28	154 l 108 12	174 | 124 28
1 1 0 1	15 CR D 13	35 GS 1D 29	55 - 2D 45 13	75 = 3D 61 29	115 M 77 13	135] 93 29	155 m 109 13	175 } 125 29
1 1 1 0	16 SO E 14	36 RS 1E 30	56 . 2E 46 14	76 > 3E 62 30	116 N 78 14	136 ^ 94 30	156 n 110 14	176 ~ 126 30
1 1 1 1	17 SI F 15	37 US 1F 31	57 / 2F 47 15	77 UNL ? 3F 63	117 O 79 15	137 UNT _ 95	157 o 111 15	177 DEL (RUBOUT) 7F 127
	ADDRESSED COMMANDS	UNIVERSAL COMMANDS	LISTEN ADDRESSES		TALK ADDRESSES		SECONDARY ADDRESSES OR COMMANDS (PPE)	(PPD)

KEY

octal	25 PPU	GPIB code
	NAK	ASCII character
hex	15 21	decimal

Tektronix
COMMITTED TO EXCELLENCE

REF: ANSI STD X3. 4-1977
IEEE STD 488-1978
ISO STD 646-1973

TEKTRONIX STD 062-5435-00 4 SEP 80
COPYRIGHT © 1979, 1980 TEKTRONIX, INC. ALL RIGHTS RESERVED.

Appendix B
VXIbus Glossary

This glossary contains terms defined according to their use in the VXIbus System. Although some terms may be used in other systems with different meanings, it is important to apply them only to VXIbus applications. In any instance in which a term is applicable only to a particular instrument module, that fact is so noted.

ACCESSED Indicator

An amber LED indicator that illuminates when the module identity is selected by the Resource Manager module, and flashes during any I/O operation for the module.

ACFAIL*

A VXIbus backplane line asserted under these conditions: 1) by the Mainframe Power Supply when a power failure has occurred (either ac line source or power supply malfunction) or 2) by the Mainframe front panel ON/STANDBY switch when it is switched to STANDBY.

A-Size Card

A VXIbus instrument module 100 x 160 x 20.32 mm (3.9 x 6.3 x 0.8"); it is the same size as a VMEbus single-high, short module.

Asynchronous Communication

Communications occurring outside the normal "command-response" cycle. Such communications may have a higher priority than synchronous communication.

Backplane

The printed circuit board mounted in a VXIbus Mainframe to provide the interface between VXIbus modules, and between the Mainframe power supplies and the modules.

B-Size Card

A VXIbus instrument module 233.4 x 160 x 20.32 mm (9.2 x 6.3 x 0.8"); it is the same size as a VMEbus double-high, short module.

Bus Arbitration

In the VMEbus interface, a system for resolving contention for service among VMEbus Master devices on the VMEbus.

Bus Timer

A functional module measuring the duration of each data transfer on the Data Transfer Bus (DTB) and terminating the DTB cycle if the duration is excessive. Without the termination capability of this module, a Bus Master attempt to transfer data to or from a non-existent Slave location could result in an infinitely long wait for the Slave response.

Client

In Shared Memory Protocol (SMP), the half of an SMP channel that does not control the shared memory buffers.

CLK10

A 10 MHz, individually buffered, differential ECL system clock sourced from Slot 0 and distributed to each slot (1 — 12) through P2. It is distributed to each module as as a single source, single destination signal with a matched delay of under eight (8) nanoseconds total.

CLK100

A 100 MHz, individually buffered, differential ECL system clock sourced from Slot 0 and distributed to each slot (1 — 12) through P3. It is synchronized to CLK10 and distributed to each module as a single source, single destination signal with a maximum timing skew of two (2) nanoseconds, and a matched delay of under eight (8) nanoseconds total.

Commander

A device controlling another device (a servant). A commander may be a servant of another commander.

Command

Any communication from a commander to a message based servant, consisting of a write to the servant's Data Low register, possibly preceded by a write to the data register.

Communication Registers

A set of device registers accessible to the commander of the device. Such registers are used for inter-device communications, and are required on all VXIbus message-based devices.

Configuration Registers

A set of registers allowing the system to identify a (module) device type, model, manufacturer, address space, and memory requirements. In order to support automatic system and memory configuration, the VXIbus Specification requires all VXIbus devices have a set of such registers, all accessible from P1 on the VXIbus.

C-Size Card

A VXIbus instrument module 340 x 233.4 x 30.48 mm (13.4 x 9.2 x 1.2").

Data Transfer Bus

One of four buses on the VMEbus backplane. The Data Transfer Bus allows Bus Masters to direct the transfer of binary data between Masters and Slaves.

DC FAIL Indicator

A red LED indicator that illuminates when a power fault is detected on the backplane.

Device Specific Protocol

A protocol for communication with a device not defined in the VXIbus specification.

D-Size Card

A VXIbus instrument module 340 x 366.7 x 30.48 mm (13.4 x 14.4 x 1.2").

DTB

See Data Transfer Bus.

DTB Arbiter

A functional module accepting bus requests from Requester modules and granting control of the DTB to one Requester at a time.

DUT

Device Under Test.

ECLTRG

Six single-ended ECL trigger lines (two on P2 and four on P3) functioning as inter-module timing resources, and bussed across the VXIbus subsystem backplane. Any module, including the Slot 0 module, may drive and receive information from these lines. These lines have an impedance of 50 Ω and the asserted state is logical High.

Embedded Address

An address in a communications protocol in which the destination of the message is included in the message.

ESTST

Extended STart/STop protocol; used to synchronize VXIbus modules.

Extended Device

A device having both VXIbus configuration registers and a subclass register. This category of device is intended to allow for definition of additional device types.

Extended Self Test

Any self test or diagnostic power-up routine executing after the initial kernel self-test program.

External System Controller

The host computer or other external controller exerting overall control of VXIbus operations.

FAILED Indicator

A red LED indicator that illuminates when a device on the VXIbus has detected an internal fault. This might result in the assertion of the SYSFAIL* line.

HI TEMP Indicator

A VX5520 Slot 0 Controller (only) LED indicator that illuminates to signify an excessive temperature condition on the module.

IACK Daisy Chain Driver

The circuit driving the VXIbus Interrupt Acknowledge daisy chain line that runs continuously through all installed modules or through jumpers across the backplane.

ID-ROM

An NVRAM storage area providing for non-volatile storage of various module attributes, such as device name, serial number, error codes, and calibration data, etc.

Instrument Module

A plug-in printed circuit board, with associated components and shields, that can be installed in a VXIbus mainframe. An instrument module can contain more than one device, and/or can occupy more than one mainframe slot.

Interface Device

A VXIbus device providing an interface to external equipment.

Interrupt Handler

A functional module detecting interrupt requests generated by Interrupters and responding to those requests by requesting status and identification information.

Interrupter

A device capable of asserting VMEbus interrupts and performing the interrupt acknowledge sequence.

IRQ

(Interrupt ReQuest) A VXIbus interrupt line asserted by an Interrupter to signify to the controller that a device on the bus requires service by the controller.

Local Bus

A daisy-chained bus connecting adjacent VXIbus slots.

Local Controller

The instrument module performing system control and external interfacing functions for the instrument modules in a VXIbus Mainframe or several Mainframes. See Resource Manager.

Local Processor

The processor on an instrument module.

Logical Address

An 8-bit number uniquely identifying each VXIbus Device in a system. It defines a device's A16 register address, and indicates Commander/Servant relationships.

Mainframe

A rigid framework providing mechanical support for modules inserted into a VXIbus backplane. It ensures connectors mate properly, adjacent modules do not contact each other, and modules do not disengage from the backplane due to vibration or shock. It also may provide mechanical support and housing for power supplies and their distribution wiring to the backplane.

Memory Device

A storage element (such as bubble memory, RAM, and ROM) having configuration registers and memory attributes (such as type and access time).

Message

A series of data bytes treated as a single communication element, with a well defined message body and terminator.

Message Based Device

A VXIbus device supporting VXI configuration and communication registers. Such devices support the Word Serial Protocol, and possibly other message-based protocols.

MODID Lines

Module/system identity lines.

Physical Address

The address assigned to a backplane slot during an access.

Power Monitor

A device monitoring backplane power and reporting fault conditions.

P1

The top-most backplane connector for a given module slot in a vertical Mainframe, such as the Tektronix VX1500. The left-most backplane connector for a given slot in a horizontal Mainframe.

P2

The middle backplane connector for a given slot in a Mainframe.

P3

The bottom backplane connector for a given module slot in a vertical Mainframe, such as the Tektronix VX1500. The right-most backplane connector for a given slot in a horizontal Mainframe.

Query

A form of command allowing for inquiry to obtain status or data.

READY Indicator

A green LED indicator that illuminates when the power-up diagnostic routines have been successfully completed. An internal failure or failure of +5 V power will extinguish this indicator.

Register Based Device

A VXIbus device supporting VXI register maps, but not high level VXIbus communication protocols.

Requester

A functional module residing on the same module as a Master or Interrupt Handler and requesting use of the DTB whenever its Master or Interrupt Handler requires it.

Resource Manager

A VXIbus device providing configuration management services such as address map configuration, determining system hierarchy, allocating shared system resources, performing system self-test diagnostics, and initializing system commanders.

Self Calibration

A routine verifying the basic calibration of the instrument module circuits, and capable of adjusting this calibration to compensate for variables such as temperature, power supply variations, and line losses..

Self Test

A set of routines testing the operational functionality of the the instrument module. These routines are performed on power-up, and on command.

Servant

A device controlled by a Commander. There are message-based and register-based servants.

Server

A shared memory device controlling the shared memory buffers used in a given Shared Memory Protocol (SMP) channel.

Shared Memory Protocol

A communications protocol using a block of memory accessible to both client and server. The memory block operates as a message buffer for communications.

Slot 0 Controller

See Slot 0 Module. Also see Resource Manager.

Slot 0 Module

A VXIbus device providing the minimum VXIbus Slot 0 services to Slots 1 — 12 (CLK10 and the MODID module identity lines), but that can also provide other VXIbus services such as CLK100, SYNC100, STARBUS, and trigger control.

SMP

See Shared Memory Protocol.

STARX

Two bidirectional, 50 Ω, differential ECL lines providing inter-module asynchronous communication. These pairs of time-delay matched lines connect Slot 0 to each of Slots 1 — 12 in a Mainframe. These lines have a well matched timing skew and less than five (5) nanoseconds of total timing delay.

STARY

Two bidirectional, 50 Ω, differential ECL lines providing inter-module asynchronous communication. These pairs of time-delay matched lines connect Slot 0 to each of Slots 1 — 12 in a Mainframe. These lines have a well matched timing skew and less than five (5) nanoseconds of total timing delay.

STST

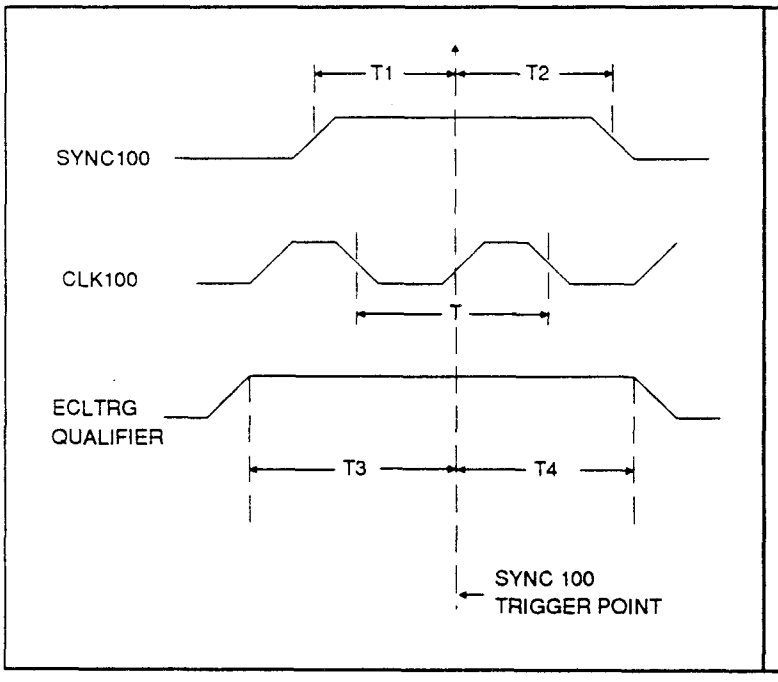
STart/STop protocol; used to synchronize modules.

SYNC100

SYNC100 is a Slot 0 signal used to synchronize multiple VXI devices with respect to a given rising edge of CLK100. The signal distributed to the devices is individually buffered to each slot, and is matched to less than 2 ns of skew between slots.

SYNC100 Trigger Protocol

SYNC100 trigger is a Tektronix designed protocol allowing multiple devices to receive a trigger on the same edge of the 100 MHz System Clock (CLK100). It utilizes the VXI SYNC100 signal to select the particular clock edge and an ECLTRG line as a qualifier to distinguish SYNC100 trigger protocol from ESTST protocol (see timing diagram below). This protocol allows device groups (such as multiple VX5260 Digitizers) to be triggered synchronously.



T = 10 ns
≥ 4.0 ns = T1 = ≤ 8.0 ns
≥ 3.5 ns = T2 = ≤ 8.0 ns
T3 ≤ 13.0 ns
T4 ≤ 11.0 ns

Synchronous Communications

A communications system following the "command-response" cycle model. In this model, a device issues a command to another device; the second device executes the command; then returns a response. Synchronous commands are executed in the order received.

SYSFAIL*

A signal line on the VMEbus used to indicate a failure by a device. The device failing asserts this line.

System Clock Driver

A functional module providing a 16 MHz timing signal on the Utility Bus.

System Hierarchy

The tree structure of the commander/servant relationships of all devices in the system at a given time. In the VXIbus structure, each servant has a commander. A commander may also have a commander.

Test Monitor

An executive routine responsible for executing the self tests, storing any errors in the ID-ROM, and reporting such errors to the Resource Manager.

Test Program

A program, executed on the system controller, controlling the execution of tests within the test system.

Test System

A collection of hardware and software modules operating in concert to test a target DUT.

TTLTRG

Open collector TTL lines used for inter-module timing and communication.

VXIbus Subsystem

One Mainframe with modules installed. The installed modules include one module in Slot 0 to perform the Slot 0 functions and a given complement of instrument modules in Slots 1 — 12. The subsystem also may include a Resource Manager.

Word Serial Protocol

A word oriented, bidirectional, serial protocol for VXIbus communications between message-based devices (devices that include communication registers in addition to configuration registers).

Word Serial Communications

Inter-device communications using the Word Serial Protocol.

WSP

See Word Serial Protocol.

10 MHz Clock

A 10 MHz timing reference. Also see CLK10.

100 MHz Clock

A 100 MHz clock synchronized to CLK10. Also see CLK100.

488-To-VXIbus Interface

A message based device providing for communication between the IEEE 488 bus and VXIbus instrument modules.

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