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 *Measurement Computing / IOtech DAC488HR/2* at our website: *Click HERE*



the smart approach to instrumentation

Iotech, Inc.
25971 Cannon Road Cleveland, OH 44146
Phone: (440) 439-4091 Fax: (440) 439-4093
E-mail: sales@iotech.com Internet: http:/www.iotech.com

DAC488HR User's Manual

P/n DAC488HR

© 1996-1998 by Iotech, Inc. Part No. DAC488HR

Printed in the United States of America.

Warranty

Your IOtech warranty is as stated on the *product warranty card*. You may contact IOtech by phone, fax machine, or e-mail in regard to warranty-related issues.
Phone: (440) 439-4091, fax: (440) 439-4093, email: sales@iotech.com

Limitation of Liability

IOtech, Inc. cannot be held liable for any damages resulting from the use or misuse of this product.

Copyright, Trademark, and Licensing Notice

All IOtech documentation, software, and hardware are copyright with all rights reserved. No part of this product may be copied, reproduced or transmitted by any mechanical, photographic, electronic, or other method without IOtech's prior written consent. IOtech product names are trademarked; other product names, as applicable, are trademarks of their respective holders. All supplied IOtech software (including miscellaneous support files, drivers, and sample programs) may only be used on one installation. You may make archival backup copies.

FCC Statement



IOtech devices emit radio frequency energy in levels compliant with Federal Communications Commission rules (Part 15) for Class A devices. If necessary, refer to the FCC booklet *How To Identify and Resolve Radio-TV Interference Problems* (stock # 004-000-00345-4) which is available from the U.S. Government Printing Office, Washington, D.C. 20402.

CE Notice



Many IOtech products carry the CE marker indicating they comply with the safety and emissions standards of the European Community. As applicable, we ship these products with a Declaration of Conformity stating which specifications and operating conditions apply.

Warnings and Cautions



Refer all service to qualified personnel. This caution symbol warns of possible personal injury or equipment damage under noted conditions. Follow all safety standards of professional practice and the recommendations in this manual. Using this equipment in ways other than described in this manual can present serious safety hazards or cause equipment damage.



This warning symbol is used in this manual or on the equipment to warn of possible injury or death from electrical shock under noted conditions.



This ESD caution symbol urges proper handling of equipment or components sensitive to damage from electrostatic discharge. Proper handling guidelines include the use of grounded anti-static mats and wrist straps, ESD-protective bags and cartons, and related procedures.

Specifications and Calibration

Specifications are subject to change without notice. Significant changes will be addressed in an addendum or revision to the manual. As applicable, IOtech calibrates its hardware products to published specifications. Periodic hardware calibration is not covered under the warranty and must be performed by qualified personnel as specified in this manual. Improper calibration procedures may void the warranty.

Quality Notice



IOtech has maintained ISO 9001 certification since 1996. Prior to shipment, we thoroughly test our products and review our documentation to assure the highest quality in all aspects. In a spirit of continuous improvement, IOtech welcomes your suggestions.

Table of Contents

1 Introduction 1.1

1.1 General Description 1.1

Figure 1.1: Block Diagram of DAC488HR System . . . 1.1

1.2 Available Accessories 1.2

1.3 Specifications 1.3

1.3.1 Compatibility with Previous Versions 1.5

2 Getting Started 2.1

2.1 Inspection 2.1

2.2 Configuration 2.1

2.2.1 Internal Settings 2.2

Figure 2.1: Internal Switch, Fuse and Jumper Locations 2.2

2.2.1.1 Line Voltage Selection 2.3

2.2.1.2 Digital Output Configuration 2.4

Figure 2.2: Output Configuration - TTL Compatible . . 2.4

Figure 2.3: High Voltage High Current Setting 2.4

2.2.2 External Settings 2.5

Figure 2.4: S1 Default Settings 2.5

2.2.2.1 IEEE 488 Bus Address Selection 2.6

2.2.2.2 Address Mode Selection 2.6

Figure 2.5: S1 Secondary Address Mode Setting 2.6

Figure 2.6: IEEE 488 Bus Address Settings 2.6

Figure 2.7: S1 Primary Address Mode Setting 2.6

2.2.2.2.1 Primary Addressing Mode 2.7

2.2.2.2.2 Secondary Addressing Mode 2.7

2.2.2.3 Calibration Enable 2.7

Figure 2.8: S1 Set for Calibration Enabled 2.7

Figure 2.9: S1 Set for Calibration Disabled 2.7

2.3 Hardware Installation 2.8

2.3.1 Rack Mount 2.8

2.3.2 Bench Top 2.8

Figure 2.10: Enclosure (Side View) 2.8

Figure 2.11: Rack Mounting Hardware (Top View) . . . 2.8

2.4 Front Panel Indicators 2.9
Figure 2.12: Front Panel Indicators 2.9
2.5 Power-up 2.10
3 DAC488HR Operation 3.1
3.1 DAC488HR Overview 3.1
Figure 3.1: DAC488HR Block Diagram 3.1
3.2 DAC488HR Commands 3.3
3.2.1 The Register-Based Command Set 3.3
3.3 Analog Output Ports 3.3
Figure 3.2: Pictorial View of DAC488HR 3.4
3.3.1 Pinouts 3.5
3.4 IEEE 488 Bus Addressing 3.5
Figure 3.3: Analog Output Connector Pinout 3.5
3.5 DAC488HR Data Buffers 3.6
3.5.1 Reading and Writing the Data Buffer 3.6
Figure 3.4: Secondary Addressing 3.6
Figure 3.5: Pre-Defined Functions 3.7
3.5.1.1 Data Buffer Location Pointer 3.8
3.5.1.2 Data Formats 3.8
3.5.2 Writing the Data Buffer 3.8
3.5.3 Data Buffer Output Control (Buffer Mode) 3.9
3.5.3.1 Linear Buffer Mode 3.9
Figure 3.6: Linear Buffer Mode 3.9
3.5.3.2 Complex Buffer Mode 3.10
3.5.3.2.1 Sequence Control Table 3.10
3.5.3 . . . . . 3.10
Figure 3.7: Complex Buffer Mode 3.10
3.5.3.2.2 Sequence Control Blocks 3.11
3.5.3.3 Changing Buffer Mode 3.11
3.5.3.4 Buffer Count 3.11
3.6 Update Clock 3.12
3.6.1 Internal Clock Settings 3.12
Figure 3.8: Update Clock Timing Diagram 3.12

3.6.2 External Clock Source 3.13
3.6.3 Update Modes 3.13
Figure 3.9: Synchronous Update Mode 3.13
3.7 Analog Output Port Triggering 3.14
3.7.1 Trigger Sources 3.14
Figure 3.10: Asynchronous Update Mode 3.14
3.7.1.1 External Trigger Source 3.15
3.7.1.2 Group Execute Trigger (GET) Source 3.15
3.7.1.3 Command Source 3.15
3.7.1.4 Interval Timer Trigger Source 3.15
3.7.2 Analog Output Port Trigger Processing 3.15
3.7.3 Trigger OUT 3.16
3.7.4 Delayed Trigger Output 3.16
Figure 3.11 Trigger Timing Diagram 3.16
3.8 Trigger Control Modes 3.17
Figure 3.12: Delayed Trigger Timing Diagram 3.17
Figure 3.13: Retriggered Delayed Trigger Timing Diagram 3.17
3.8.1 Direct Control 3.18
3.8.2 Step 3.18
3.8.3 Burst 3.18
3.8.4 Waveform 3.18
3.8.5 Immediate 3.18
3.8.6 Continuous 3.19
3.9 Digital Inputs 3.20
3.9.1 Digital Input Connector 3.20
3.9.1.1 8-Bit TTL Input Port (Pins 1 - 8) 3.20
3.9.1.2 Power (Pin 9) 3.20
Figure 3.14: Digital Input Connector Pinout 3.20
3.9.1.3 External Clock (Pin 13) 3.21
3.9.2 Trigger In BNC Connector 3.21
3.10 Digital Outputs 3.21
3.10.1 Digital Output Connector 3.21
Figure 3.15: Digital Output Connector Pinout 3.21

3.10.1.1 8-Bit TTL Output Port (Pins 1-8) 3.22
3.10.1.1.1 High Voltage/High Current Outputs 3.22
3.10.1.2 Power (Pin 9) 3.22
Figure 3.16: High Current/High Voltage Output Circuitry 3.22
3.10.1.3 Flyback (Pin 10) 3.23
3.10.1.4 Trigger Output (Pin 13) 3.23
3.10.1.5 Delayed Trigger Output (Pin 14) 3.23
Figure 3.17 Trigger Timing Diagram 3.23
3.10.1.6 Update Clock Output (Pin 15) 3.24
3.10.2 Trigger Out BNC Connector 3.24
3.11 User Defined System Defaults 3.24
3.12 IEEE 488 Bus Implementation 3.25
4 Programming the DAC488HR 4.1
4.1 Static DC Output 4.1
4.2 Predefined Waveform Generation 4.2
4.3 High Speed Buffer Loading 4.2
4.4 Single Channel Continuous Transfer Mode 4.6
4.5 Two Channel Continuous Transfer Mode 4.9
4.6 Synchronizing Multiple DAC488HR Units 4.13
Figure 4.1: Master/Slave Connector Pinouts 4.14
Figure 4.2: DAC488HR Master/Slave Cable Configuration 4.14
4.6.1 For Synchronous Update Modes (C1-C4) 4.15
4.6.2 For Continuous Update Modes (C5-C7) 4.18
4.7 Sample DAC488HR Programs 4.22
4.7.1 Binary Data Load Example 4.22
4.7.2 Continuous Data Transfer Example 4.24
4.7.3 Interleaved Data Transfer Example 4.26
4.7.4 Multiple Units (Continuous Update Mode) Example . . . 4.30
4.7.5 Multiple Units (Synchronous Update Mode) Example . . 4.33
5 Command Descriptions 5.1
5.1 Overview 5.1
5.2 Terminators 5.2
5.3 Command Interpretation 5.3

5.4 Command Execution Order 5.3
5.4.1 Deferred Command Execution Order 5.4
5.5 Syntax Rules 5.4
5.5.1 Case Sensitivity 5.5
5.5.2 Spaces 5.5
5.5.3 Voltage Values 5.5
5.5.4 Multiple Parameters 5.5
5.5.5 Command Strings 5.6
5.5.6 Execute (X) 5.6
5.5.7 Query Option 5.7
5.5.8 Fixed Formats 5.8
5.6 Serial Poll Status Byte Register 5.8
5.7 DAC488HR Serial Poll Model 5.10
Figure 5.1: DAC488HR Serial Poll Model 5.11
5.8 Event Status Register 5.12
5.9 Error Source Register 5.13
Figure 5.2: DAC488HR Status Reporting Registers . . . 5.13
Command Trigger (@@) 5.14
Reset (*R) 5.15
Buffer Mode (An) 5.16
Buffer Data (Bval) 5.17
Trigger Control Mode (Cn) 5.20
Figure 5.3: Stepped Mode Output 5.21
Figure 5.4: Burst Mode Output 5.21
Figure 5.5: Waveform Mode Output 5.22
Figure 5.6: Immediate Mode Output 5.22
Figure 5.7: Continuous Mode Output 5.24
Digital Output (Dn) 5.25
Error Query (E?) 5.26
Figure 5.8: DAC488HR Status Reporting Registers . . . 5.26
Format (Fn) 5.28
Update Source, Mode (Gn) 5.30
Figure 5.9: Synchronous Update Mode 5.31

Figure 5.10: Asynchronous Update Mode 5.32
Offset Calibration (Hn) 5.33
Update Divider (In) 5.34
Figure 5.11: Update Clock Timing Diagram 5.34
Gain Calibration (Jn) 5.36
Buffer Count (Kn) 5.37
Data Buffer Location Pointer (Ln) 5.38
SRQ Mask (Mn) 5.40
Event Mask (Nn) 5.41
Figure 5.12: DAC488HR Status Reporting Registers . . 5.42
Sequence Pointer (On) 5.43
Port Select (Pn) 5.45
Define Sequence Control Block (Ql,n,r) 5.46
Range (Rn) 5.48
Save/Restore (Sn) 5.50
Trigger Source (Tn) 5.53
Status (Un) 5.55
Voltage Output (Vval) 5.59
Waveform Load (Ww,l,max,min,d) 5.62
Figure 5.13: Pre-Defined Functions 5.63
Execute (X) 5.66
Interval Timer (Yn) 5.67
Trigger Delay (Zn) 5.68
Figure 5.14: Delayed Trigger Timing Diagram 5.68
Figure 5.15: Retriggered Delayed Trigger Timing Diagram 5.68

6 DAC488HR Calibration 6.1

6.1 Calibration Overview 6.1
6.2 Calibration Theory 6.1
Figure 6.1: Ideal D/A Converter Response 6.1
Figure 6.2: Deviation from Ideal Response 6.2
Figure 6.3: Analog Output DAC Circuitry 6.3
6.3 Calibration Procedure 6.4
6.3.1 Equipment Required 6.4

6.3.2 Calibration Setup 6.4
6.3.3 Calibration Method 6.5
Figure 6.4: Calibration Cable Wiring Diagram 6.5
6.3.3.1 Offset Calibration 6.6
6.3.3.2 Gain Calibration 6.8
6.4 Sample Calibration Program 6.9
A Glossary A.1
B Character Codes and IEEE Multiline Messages B.1
C Command Summary C.1

Introduction

1.1 General Description

The DAC488HR is a multiple output Digital to Analog Converter (DAC) for the IEEE 488 bus. It is available with two or four analog output ports. The two-port version is the DAC488HR/2; the four-port version is the DAC488HR/4.

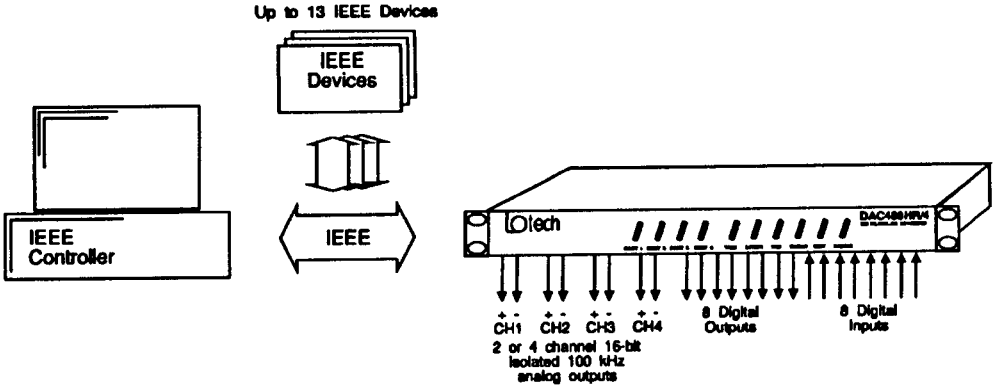


Figure 1.1: Block Diagram of DAC488HR System

Each analog output port uses a 16-bit D/A converter and is isolated from the IEEE 488 common and each of the other ports by up to 500 volts. Each analog output port has its own processor, providing waveform generation independent of the other ports. A common trigger can be used to step each port through a preprogrammed waveform. All analog output ports can be triggered from one of four different sources: an external TTL trigger, a trigger command, a Group Execute Trigger (GET) bus command, or an internally generated timebase. Six different trigger control modes are available to allow each analog output port to output data differently in response to a common trigger.

Full scale output of 1, 2, 5 or 10 volts, unipolar or bipolar, can be programmed for each analog output port. Data may be specified in volts or bits in increments of one part in 32,768 of full scale bipolar or one part in 65,536 of full scale unipolar.

Each analog output port maintains its own data buffer of 8K sample points. With the MEMX3 or MEMX4 memory expansion options, any analog output port may have 128K or 480K sample point buffers, respectively.

All configuration settings can be saved in non-volatile RAM (NVRAM) for use as the power-on defaults.

The DAC488HR provides eight TTL-level digital inputs and eight digital outputs. The digital outputs may be internally configured as either TTL-level or 100 mA @ 50 V to allow interfacing with solenoids, relays or other high-power devices.

Separate 15-pin D-shell connectors on the rear panel are provided for the digital input and digital output signals. +5 V and ground are also accessible on these connectors.

Throughout this manual, "DAC488HR" refers to either a DAC488HR/4 or a DAC488HR/2.

1.2 Available Accessories

Additional accessories that can be ordered for the DAC488HR include:

CA-7-1	1.5 foot IEEE 488 cable
CA-7-2	6 foot IEEE 488 cable
CA-7-3	6 foot shielded IEEE 488 cable
CA-7-4	6 foot reverse entry IEEE 488 cable
CA-89	6 foot shielded cable with male DB-9 connector for analog output
CA-90	6 foot shielded cable with male DB-15 connector for digital I/O
CN-20	Right angle IEEE 488 adapter, male and female
CN-22	IEEE 488 multi-tap bus strip, four female connectors in parallel
CN-23	IEEE 488 panel mount feed-through connector, male and female
DAC488HR-901	Additional user's manual
MEMX3	128K sample memory expansion for one analog output port. Factory installed on motherboard
MEMX4	480K sample memory expansion for one analog output port. Factory installed on motherboard
R4	Waveform generation and editing software for IBM PCs

1.3 Specifications

ANALOG OUTPUT

DC Output Voltage/Resolution:

Range	Unipolar		Bipolar	
	Output Volts	Resolution	Output Volts	Resolution
1 V	0-1	15.3µV	±1	30.5µV
2 V	0-2	30.5µV	±2	61µV
5 V	0-5	76.3µV	±5	153µV
10 V	0-10	153µV	±10	305µV

DC Output Current: ±10 mA maximum

Accuracy (25 ±5°C; I_{out} = 1 mA):

1V Range: ±(0.02% ±50µV)
2V Range: ±(0.02% ±100µV)
5V Range: ±(0.02% ±250µV)
10V Range: ±(0.02% ±500µV)

Polarity: Unipolar or Bipolar (software selectable)

Output Impedance: 10 Ω

Zero Offset: 50 to 500µV

Linearity: 0.005%

Differential Linearity: 0.001%

Update Rate: 100K samples/sec (maximum)

All channels out of the data buffer is 100k samples/sec maximum.
Any single channel from the IEEE 488 bus to output is 100k samples/sec maximum
Any two channels from the IEEE 488 bus to output is 50k samples/sec maximum.

Settling Time: 6µsec to 0.003% FSR

Temperature Coefficient: (±0.002% ±100µV)/°C; 0-20°C and 30-50°C.

Port to Port Isolation: 500 volts maximum, 10^5V-Hz.

Each Analog Output Port to Digital Low Isolation: 500 volts maximum, 10^5V-Hz.

Connectors: DB-9 (female) per analog output port. Mating connector supplied.

DIGITAL I/O

Digital Inputs:

8 TTL-level general purpose inputs
External TTL Trigger
External TTL clock input

Digital Outputs:

8 general purpose outputs, TTL-level or open collector with 100 mA @50 V drive
Delayed trigger output signal
Update clock output.

Connectors: DB-15 (female) per input and output digital port. Mating connector supplied.

IEEE 488 BUS SPECIFICATIONS

Interface Subsets: SH1, AH1, T6, TE6, L4, LE4, SR1, RL0, PP0, DC1, DT1, C0, E2.

Connector: Standard IEEE 488 connector with metric studs.

DATA STORAGE AND OUTPUT

Data Buffer: 8K sample standard (per analog output port)
128K sample with MEMX3 option or 480K sample with MEMX4 option.

Data Format: ASCII, integer decimal, hexadecimal, and binary.

Output Modes: Direct, step, burst, waveform, and continuous.

Trigger Sources: GET, Periodic Time Interval, External (TTL-level), and Command.

GENERAL

Power: Internally selectable 90-125 V or 210-250 V ac; 50/60 Hz; 20 VA Max.

Environment: 0 – 50°C; 0 to 95% RH (non-condensing).

Dimensions: 425mm wide x 45mm high x 305mm deep (16.75" x 1.75" x 12") without rack ears.

Weight: 17.38 kg. (7.9 lbs).

Controls: Power Switch (external), DIP switch for IEEE 488 bus address (external), line voltage selection switch (internal), calibration enable switch (external), digital output port configuration jumper (internal).

Logic Levels: Digital outputs drive two TTL loads or sink 100 mA (selectable using internal configuration jumper). The TTL inputs are capable of sinking 0.2 mA @ <0.4 V low or sourcing 20µA @ >2.7 V high. The high current/high voltage outputs can sink up to 100 mA at 50 V dc.

Output Resistance: 10 Ω.

Warm up time to rated accuracy: 1 hour

External Trigger Input: 1 line, TTL-level compatible.

Connectors: Analog DB-9S. Digital DB-15S. Mating connectors supplied.

IEEE 488-1978

Terminators: LF with EOI on output, forgiving on input

Connector: Standard IEEE 488 connector with metric studs.

Programmable: SRQ Mask, analog output Port Voltage, Gain, Offset, Digital I/O lines, Output Format, Internal Buffer, Mode.

General Indicators: LEDs for Talk, Listen, SRQ, Error, Test and Power.

1.3.1 Compatibility with Previous Versions

Note if using versions of DAC488HR software previous to 1.2, there may be conflicts in the use of the `W` and `F` commands. Refer to the Software Command Reference section of this manual for details on these specific commands.

Getting Started

2.1 Inspection

The DAC488HR was carefully inspected mechanically and electrically prior to shipment. After receiving the DAC488HR, carefully unpack all items from the shipping carton and check for any obvious signs of physical damage. Report any such damage found to the shipping agent immediately. Retain all shipping materials in the event that shipment back to the factory is necessary.

Every DAC488HR is shipped with the following items:

DAC488HR	Digital to Analog Converter
DAC488HR-901	User's manual
166-0600	Sample program disk in IBM format
CA-1	Power cable
FE-1	Rubber feet (4)
EN-6	Rack ears (2)
HA-41-6	Screws (4)
FU-1-.5	½ amp replacement fuse
FU-1-.375	⅜ amp replacement fuse
CN-18-9	DB-9P mating connector for analog output connectors (4) for DAC488HR/4; (2) for DAC488HR/2
CN-18-15	DB-15P mating connectors for digital input/output connectors (2)
If ordered:	
MEMX3	128K sample memory expansion for one analog output port - factory installed.
MEMX4	480K sample memory expansion for one analog output port - factory installed.

2.2 Configuration

To operate properly, the DAC488HR must be configured for the operating environment. Line voltage must be configured for 110 or 220 V ac to match the power being fed to the DAC488HR. The unit was shipped from the factory set for the operating voltage marked on the label placed over the power jack on the rear panel. The digital output port must be configured for TTL compatible or high voltage/high current outputs, depending on the user's application. The IEEE 488 bus address must be set to avoid conflict with other devices in the IEEE 488 system.

2.2.1 Internal Settings

The only internal settings on the DAC488HR are for the line voltage and the high voltage/high current digital output.

The locations of the jumpers and switches for these options are shown in Figure 2.1.

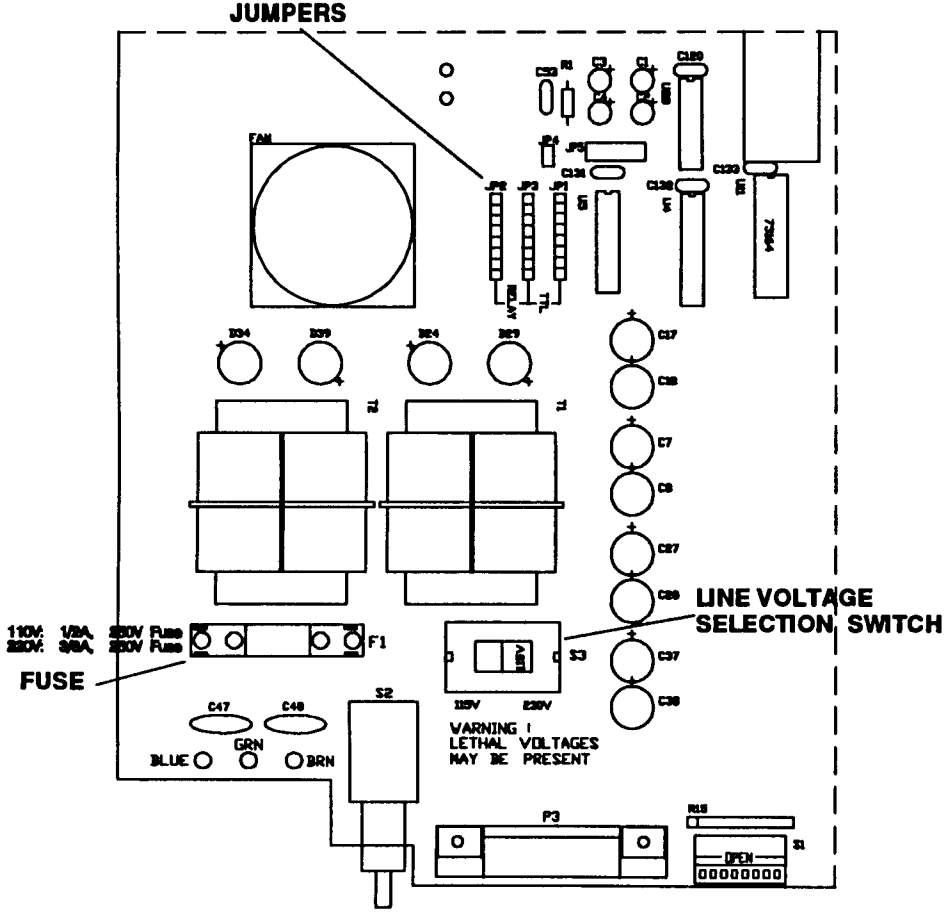


Figure 2.1: Internal Switch, Fuse and Jumper Locations

WARNING

Disconnect the power cord from the power line and from the DAC488HR prior to disassembly. Disconnect any cables prior to disassembly.

WARNING

Never open the DAC488HR case while it is connected to the ac line. Internal voltage potentials exist that could cause personal injury or death.

To access these settings, place the DAC488HR on a flat surface. Remove the six screws on top of the case and remove the top cover. Reverse this procedure to reassemble the unit.

2.2.1.1 Line Voltage Selection

The DAC488HR may be operated from 110 or 220 V ac. The unit was shipped from the factory set for the operating voltage marked on the label placed over the power jack on the rear panel. To change the operating voltage, change the setting of internal switch S3 according to the following instructions.

WARNING

The DAC488HR is intended for INDOOR USE ONLY. Failure to observe this warning could result in equipment failure, personal injury or death.

1. The line voltage selection switch (S3) is located next to the main power supply transformer (T1). Insert the tip of a small screwdriver into the slot of the switch and move the switch so the desired line voltage appears on the switch.
2. Install a power line fuse appropriate for the line voltage. The fuse is located next to the internal line voltage switch (S3). Gently pull upward on the plastic fuse housing to remove the entire housing with the fuse inside. Select a fuse with the proper rating (see table below).

Line Voltage	Fuse Type
90-125V	½A 250V, Slo Blo, 3AG
210-250V	⅜A 250V, Slo Blo, 3AG

CAUTION

A fuse with a rating higher than that specified may cause damage to the instrument. If the instrument repeatedly blows fuses, contact the factory.

3. Open the fuse housing by pushing up on the tab on the bottom of the housing.
4. Replace the fuse and close the housing. Insert the fuse housing into the fuse holder.
5. Make note of the new voltage setting for later reference and carefully re-assemble the unit.

2.2.1.2 Digital Output Configuration

The DAC488HR's digital outputs are factory set for TTL logic levels. They can also be configured as high voltage/high current outputs for interfacing with relays, lamps and solenoids. The high voltage/high current outputs can sink up to 100 mA at 50 V dc through the use of open collector drivers with integral diodes for inductive load transient suppression.

CAUTION

Do not connect external high level devices to the digital output lines unless they have first been configured for this purpose. Failure to do so may result in damage to the DAC488HR.

Once the digital output lines have been configured for high level operation, they can be used to drive devices such as relays, solenoids and displays. For example, a typical application may require an indicator light and a relay to be driven by the DAC488HR.

To configure the digital output lines for high voltage/high current, reposition the configuration jumper according to the following instructions. Section 3.10.1.1.1 contains more information on high voltage/high current digital outputs.

Three sockets and a configuration jumper are located next to the fan. This jumper is factory set to configure the digital output lines for TTL logic levels as shown in Figure 2.2.

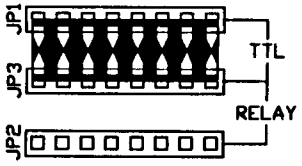


Figure 2.2: Output Configuration - TTL Compatible

To configure the digital output lines as high voltage high current outputs, remove the configuration jumper and reinsert it into sockets JP3 and JP2 as shown in Figure 2.3.

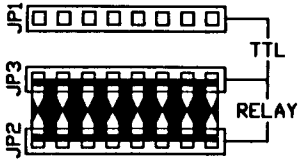


Figure 2.3: High Voltage High Current Setting

Make note of the new setting for later reference.

2.2.2 External Settings

The DAC488HR has one eight position switch (S1) accessible from the rear panel. This switch determines the unit's IEEE 488 bus address and its operating mode. The switch is read only during power on and should be set before applying power. Figure 2.4 shows the factory default setting for S1.

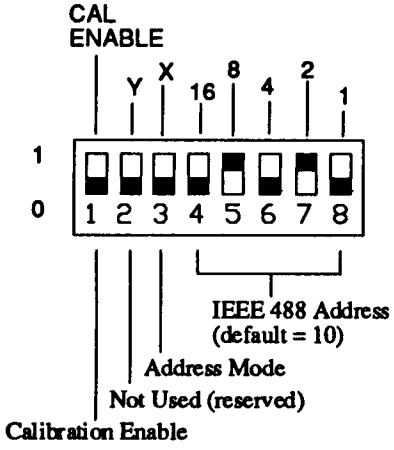


Figure 2.4: S1 Default Settings

To modify any of these defaults, disconnect the power cord from the power line and change the switch settings using a small screwdriver. The enclosure does not need to be opened to change the S1 settings.

2.2.2.1 IEEE 488 Bus Address Selection

The IEEE 488 bus address is set by switches S1-4 through S1-8 (located on rear of unit). The address can be set from 0 through 30 and is read only at power on. The address is selected by simple binary weighting. S1-8 is the least significant bit; S1-4 is the most significant bit. The factory default is address 10. Figure 2.6 shows the factory default IEEE 488 bus address setting.

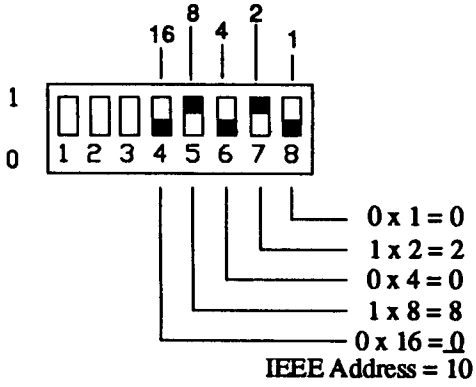


Figure 2.6: IEEE 488 Bus Address Settings

If address 31 is specified as the IEEE 488 bus address, address 30 will be used, since address 31 is reserved for the Unlisten (UNL) or the Untalk (UNT) command on the bus.

2.2.2.2 Address Mode Selection

The DAC488HR uses either primary or secondary addressing on the IEEE 488 bus. S1-3 (marked X on the rear panel of DAC488HR) selects between these two modes of operation. For more information on primary and secondary addressing, see below and section 3.4, IEEE 488 Bus Addressing.

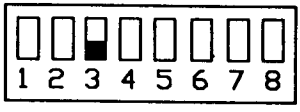


Figure 2.7: S1 Primary Address Mode Setting

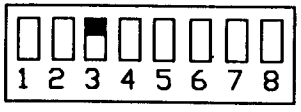


Figure 2.5: S1 Secondary Address Mode Setting

2.2.2.2.1 Primary Addressing Mode

If primary addressing mode is selected, the address is set by switches S1-4 through S1-8. The analog output port for data transfer is selected by the port command. Raw binary data can then be transferred using the protocol described on page 5.18.

2.2.2.2.2 Secondary Addressing Mode

If secondary addressing mode is selected, the DAC488HR occupies one primary address on the IEEE 488 bus, set by switches S1-4 through S1-8. Five secondary addresses are used in conjunction with the primary address to communicate with the DAC488HR. Secondary address 00 is used as the command channel to transfer command and status information.

Secondary addresses 01 through 04 transfer data directly to analog output ports 1 through 4, respectively. Addressing of a specific analog output port is inherent in the secondary address of the data channel that data are transferred to.

2.2.2.3 Calibration Enable

The calibration enable switch protects calibration constants held in non-volatile RAM (NVRAM). When enabled, the existing calibration constants for each analog output port may be stored in NVRAM. When disabled, the calibration constants can be changed, but any attempt to write them to NVRAM results in an error.

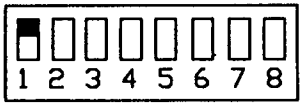


Figure 2.8: S1 Set for Calibration Enabled

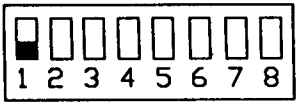


Figure 2.9: S1 Set for Calibration Disabled

2.3 Hardware Installation

Accessories for rack or bench use are included with the DAC488HR.

2.3.1 Rack Mount

If rack mount installation is required, remove the two plastic screws from the predrilled holes on each side of the unit. Only remove the screws from the set of holes that will be toward the front of the rack (the unit can be mounted with the front or rear panel facing the front of the rack fixture).

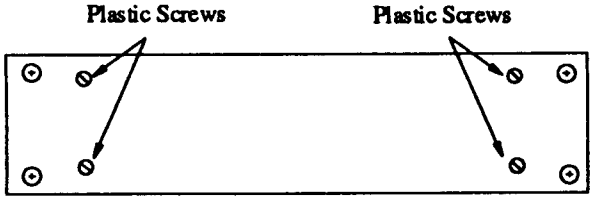


Figure 2.10: Enclosure (Side View)

Install the two rack ears using the enclosed screws.

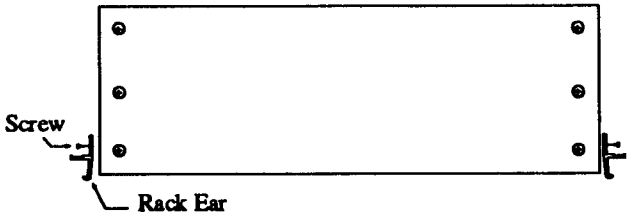


Figure 2.11: Rack Mounting Hardware (Top View)

2.3.2 Bench Top

If bench top installation is required, install the rubber feet on the bottom of the unit approximately one inch from each corner.

2.4 Front Panel Indicators

Ten indicator lights on the front panel display the status of the DAC488HR. The function of each indicator is described below. The DAC488HR/2 does not have indicators for Port 3 and Port 4.

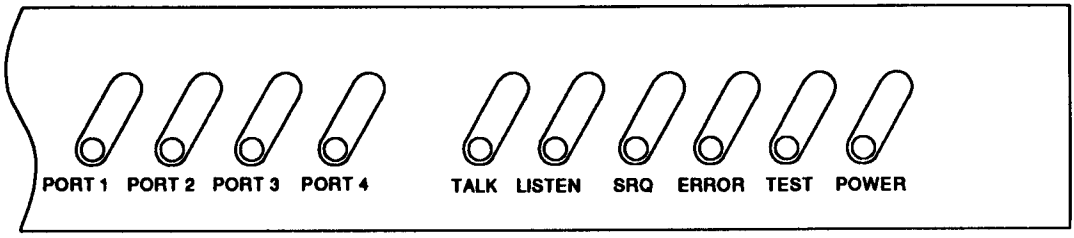


Figure 2.12: Front Panel Indicators

TALK	On when the DAC488HR is in the Talker Active state, off when the DAC488HR is in the Idle or Listener state.
LISTEN	On when the DAC488HR is in the Listener Active state, off when the DAC488HR is in the Idle or Talker state.
SRQ	On when the DAC488HR has generated a service request (SRQ), off when no SRQ is pending.
ERROR	On when an error has occurred, off when no error condition exists.
TEST	On when the DAC488HR is performing a power-on self-test. Off when the unit is operating normally and all self-tests pass.
POWER	On when power is applied to the DAC488HR and the power switch on the back panel is in the on position (depressed). Off if power is not present.
PORT 1 PORT 2 PORT 3 PORT 4	On when outputting a steady dc voltage. Flashing when in trigger mode. Off when its analog output port is set to 0 volts.

2.5 Power-up

At initial power-up, the DAC488HR performs automatic self-tests to ensure that it is fully functional. The indicator lights on the DAC488HR front panel show the progress of the self-tests as they are run and show failures if any occur. The tests include ROM and RAM tests on the main system unit. All indicator lights turn on and remain on until these tests are completed. If a ROM failure is detected, all indicator lights remain lit. If a RAM failure is detected, all indicator lights flash.

If no problems are found, the indicator lights go off and the DAC488HR will begin its power-up initialization. This will take approximately 5 seconds to complete, after which the DAC488HR is ready for normal operation.

DAC488HR Operation

3.1 DAC488HR Overview

The DAC488HR provides two or four independently programmable 16-bit digital to analog converters (DACs). Each analog output port is electrically isolated through the use of optoisolators. The ports can produce a maximum voltage range of ±10 V.

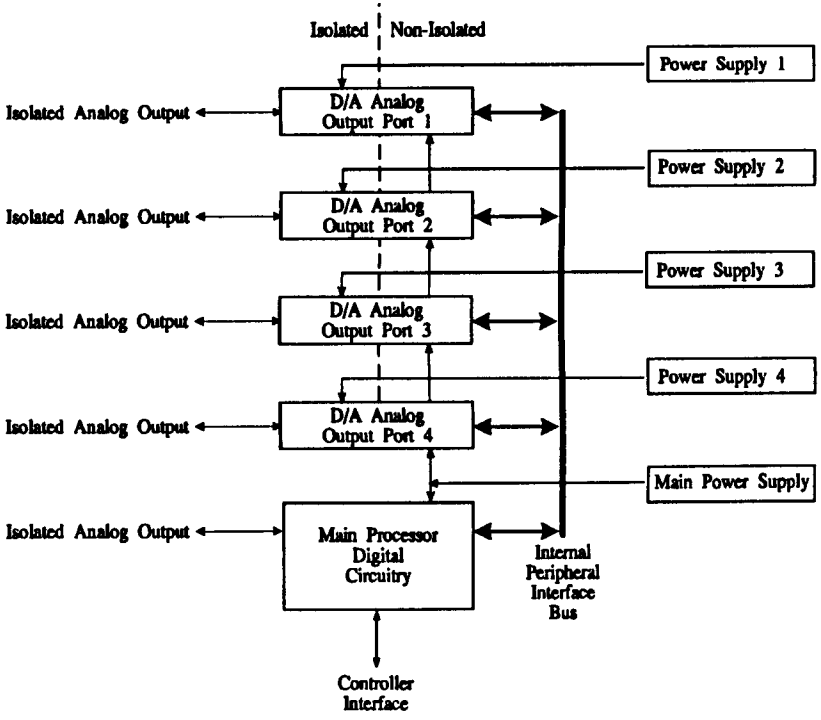


Figure 3.1: DAC488HR Block Diagram

Each analog output port has an internal data buffer capable of storing 8192 voltage values. The data buffer can be expanded to 128K voltage values when an analog output port has an MEMX3 option installed, or to 480K voltage values with an MEMX4 option installed. The buffer stores data voltage values to be output. Outputs are updated when a trigger event occurs or immediately upon receipt of a command. The DAC488HR offers pre-defined waveform functions that can be written automatically into the data buffer. The pre-defined waveform function loads one cycle of a sine wave, triangle wave or square wave.

Data transfer to and from the IEEE 488 bus can occur at rates up to 450 Kbytes/second for raw binary data. In addition, output of interleaved data for two analog output ports may be configured, allowing full 100 kHz updating of two analog output ports from the IEEE 488 bus simultaneously.

Each analog output port can respond to a trigger event. The specific response of each analog output port to a trigger event is controlled by the trigger control mode for each port. The voltage value at each analog output port changes synchronously in response to triggers.

The DAC488HR trigger control modes govern how data are output from the data buffer in response to a trigger. Each analog output port may be operated in a different mode. Data can be output as a single value, as the entire data buffer a specific number of times, as a burst of values from the buffer, or as continuous data from the IEEE 488 bus in response to each trigger event.

All analog output ports are updated simultaneously to guarantee synchronization. This is done through use of a common update clock. Commands allow selection of the external clock or the base frequency source and divisor of the frequency source to allow flexibility in the update rate for analog output update.

The method by which the DAC488HR responds to triggers is related to the update rate and is set by the update mode. Update mode can be asynchronous or synchronous to update clock generation. Synchronous update mode holds off the response to a trigger until the next occurrence of an update clock. When asynchronous update mode is selected, triggers are recognized as quickly as possible.

The DAC488HR provides eight digital inputs and eight digital outputs. The digital inputs are used to read up to eight TTL-level inputs. The digital outputs are used to control eight TTL-level outputs. The digital output lines can also be configured for high voltage/high current open-collector relay driver outputs for directly driving relays. The digital outputs are designed so that the power turn-on state is guaranteed. When configured for TTL compatible outputs, the outputs are always initialized to logic low upon power turn-on. When configured for high voltage/high current outputs, the outputs are always initialized as off (open), so no current flows through any relay coils attached to the DAC488HR digital outputs upon power turn-on.

User-defined system defaults and calibration constants are stored in battery-backed non-volatile RAM (NVRAM). This allows the DAC488HR to power-up in a state specified by the user.

3.2 DAC488HR Commands

Operation of the DAC488HR is accomplished using a set of character-based commands that configure the entire unit as well as each analog output port. The DAC488HR commands are divided into two groups: System commands and Port commands. System commands affect the operation of the entire unit and are not specific to a given analog output port. Port commands only affect the operation of the selected analog output port.

3.2.1 The Register-Based Command Set

IOtech peripheral products for the IEEE 488 bus use a register-based command set. This model is based upon the concept of "registers". Just as most peripheral chips have one or more registers that may be set to different values to control their operation, each command letter in this command set corresponds to an internal register. In general, each of these registers holds a single numeric value that is maintained at the last value set.

The DAC488HR is controlled by modifying the contents of these registers. The relationship between the contents of the registers and the actions taken by DAC488HR are described in Section 5.6.

3.3 Analog Output Ports

The DAC488HR can be thought of as multiple isolated IEEE 488 to analog converters. It has two or four 16-bit digital to analog output ports. Each analog output port has a low (L), high (H), and case ground line. The case ground line may be connected to the shield if a shielded cable is used.

CAUTION

The maximum common-mode input voltage (the voltage between output low and chassis ground) is 500 V peak. Exceeding this value may damage the DAC488HR.

Each analog output port is electrically isolated from the other ports and chassis common by optoisolators.

The voltage range for each analog output port is set separately to 1, 2, 5 or 10 volts full scale, unipolar or bipolar. Range cannot be specified on a per sample basis during buffer storage output of samples.

When the unipolar setting is selected, the range is 0 to + full scale and each bit has a value of 1/65,536 of the full scale range. Voltage is represented as a 16-bit unsigned binary number.

When the bipolar setting is selected, the range is – full scale to + full scale and each bit has a value of 1/32,768 of the full scale range. Voltage is represented as a 16-bit two's complement binary number.

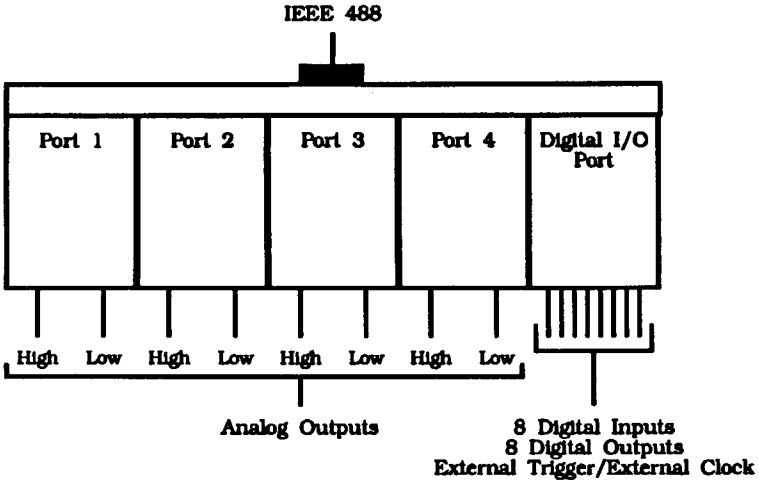


Figure 3.2: Pictorial View of DAC488HR

Each analog output port maintains its own separate data buffer of analog output values and responds in its own way to triggers. The trigger response of each analog output port is determined by the trigger control mode. The data buffer of output values is controlled separately for each analog output port (see Section 3.5 for more information on data buffers).

The analog output ports may be programmed in terms of volts or bits. The resolution per bit for each range is:

RANGE	RESOLUTION	
	Unipolar	**Bipolar**
1 V	15.3µV	30.5µV
2 V	30.5µV	61µV
5 V	76.3µV	153µV
10 V	153µV	305µV

Voltages may be programmed with decimal floating point numbers in the range of ±10.0000. They may also be programmed in bits using decimal numbers (range ±32,768 bipolar, 0-65,536 unipolar) or 16-bit hexadecimal two's complement numbers (range 0000 to FFFF).

3.3.1 Pinouts

The pinouts for each analog output port are shown in Figure 3.3. Each analog output port is capable of sourcing and sinking a maximum current of 10 mA.

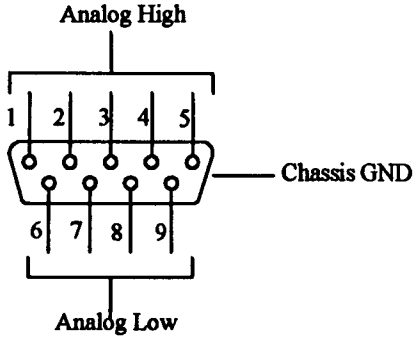


Figure 3.3: Analog Output Connector Pinout

3.4 IEEE 488 Bus Addressing

To allow communication with each channel, two different addressing modes are available: Primary and Secondary.

If primary addressing mode is selected, the address is set by switches `S1-4` through `S1-8`. The analog output port for data transfer is selected by the port command. Raw binary data can then be transferred using the protocol described on page 5.18.

> **If address 31 is specified as the IEEE 488 bus address, address 30 will be used, since address 31 is reserved for the Unlisten (UNL) or the Untalk (UNT) command on the bus.**

In Secondary addressing mode, one primary address on the IEEE 488 bus is used, and several secondary addresses are used. Communication with the DAC488HR over the IEEE 488 bus is accomplished using two different communication channels: one for commands and one for data. In general, when the DAC488HR is addressed to TALK or LISTEN on the command channel, the IEEE 488 controller is communicating with DAC488HR's command processor. When addressed to TALK or LISTEN on the data channel, the IEEE 488 controller is connected directly to one analog output port's data buffer.

One secondary address is dedicated to the command channel, and a single secondary address is dedicated to each analog output port for a data channel. In this way, the port command does not have to be used for selecting a specific analog output port; the addressing for a

specific analog output port is inherent in the secondary address used as a data channel. The secondary address for the command channel is 00. Secondary addresses 01 through 04 correspond to analog output ports 1 through 4 respectively.

For example, if the DAC488HR address switch is set to IEEE 488 bus address 10 and secondary addressing mode is enabled, the command channel will reside at IEEE 488 bus address 1000 and the data channel for analog output port 1 will reside at 1001, the data channel for analog output port 2 will reside at 1002, the data channel for analog output port 3 will reside at 1003 and the data channel for analog output port 4 will reside at 1004.

The choice between Primary and Secondary addressing modes depends on the IEEE 488

Primary Address = 10				
Command Secondary Address 00	Data Secondary Address 01 (Port 1)	Data Secondary Address 02 (Port 2)	Data Secondary Address 03 (Port 3)	Data Secondary Address 04 (Port 4)

Figure 3.4: Secondary Addressing

system in which the DAC488HR is being installed. In systems where the IEEE 488 controller software does not support secondary addressing, Primary addressing must be used. Secondary address mode is advantageous because addressing of a specific analog output port's data buffer is inherent in the secondary address selected, removing some extra steps when choosing a specific analog output port.

3.5 DAC488HR Data Buffers

The DAC488HR analog output ports each have a standard data buffer of 8192 samples. This buffer size can be expanded to 131,072 with MEMX3 option or to 491,520 with the MEMX4 option. Data are written from the IEEE 488 bus into the data buffer at a rate of up to 450 kBytes/s. The data buffers are not battery backed up.

3.5.1 Reading and Writing the Data Buffer

Data are written into the data buffer by setting the data buffer location pointer and writing to that location in the buffer. Data are read from the data buffer by setting the data buffer location pointer and reading from that location in the buffer.

The DAC488HR offers pre-defined waveform functions that can be written automatically into the data buffer starting at the data buffer location pointer. The pre-defined waveform function loads one cycle of a sine wave, triangle wave, or square wave.

The command for the pre-defined functions requires parameters for the length (# of samples), maximum and minimum values (as a percentage of full scale), and the symmetry (as a percentage of length) of the waveform. When data writing begins, the DAC488HR transfers

one cycle into the buffer starting at the present location of the data buffer location pointer according to these parameters. A sequence control block can then be used to force this cycle to repeat as often as desired by the user in the output waveform.

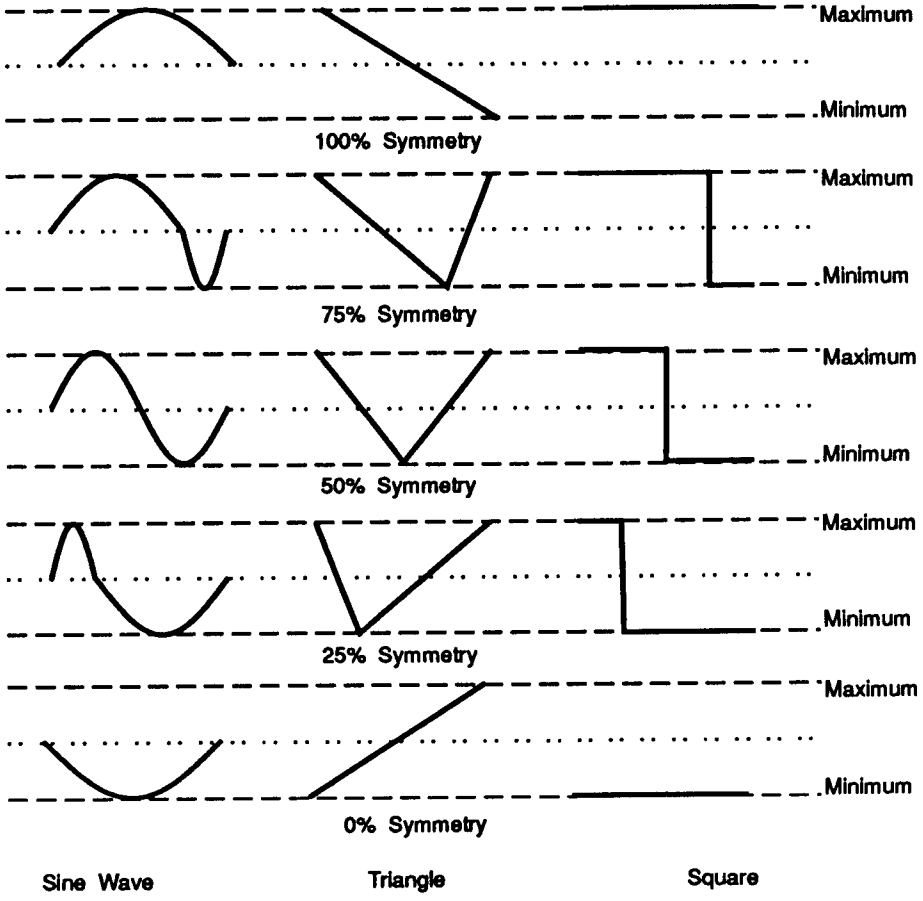


Figure 3.5: Pre-Defined Functions

3.5.1.1 Data Buffer Location Pointer

The data buffer location pointer points to a specific location in the data buffer. It is automatically incremented after each access, which allows successive data buffer locations to be written or examined without respecifying the pointer each time.

3.5.1.2 Data Formats

Voltage readings may be written to or read from an analog output port data buffer using one of four formats:

ASCII volts
Integer decimal
Hexadecimal
Binary

All of the formats except binary may be loaded through the command channel or the data channel. Binary data can only be loaded through the data channel (see IEEE 488 addressing, Section 3.4).

Binary format should be used for high speed data transfer or if a large number of readings must be written into the analog output port data buffer. Binary data transfer to or from the data buffer is terminated by the assertion of the EOI bus signal.

3.5.2 Writing the Data Buffer

Data are written to the data buffer by the buffer data command. The buffer data are written to the location pointed to by the data buffer location pointer. The format of data entered for writing into the data buffer is specified by the format command.

Successive locations may be written without respecifying the buffer data command for each value.

Data may be directed to the proper analog output port in one of two ways.

If the DAC488HR is set for primary addressing mode, the port select command must be used to select which analog output port's data buffer is written. Data points can then be written with the buffer data command.

If the DAC488HR is set for secondary addressing mode, the port select command can still be used through the command channel (secondary address 0) to select which analog output port's data buffer is written. Data points are then loaded with the buffer data command through the command channel (secondary address 0). The destination analog output port can also be specified implicitly by directing data to the secondary address associated with that analog output port. Secondary addresses 01 through 04 correspond to analog output ports 1 through 4, respectively.

3.5.3 Data Buffer Output Control (Buffer Mode)

After sample points are written into the data buffer, there are two means of outputting data:

Linear buffer mode
Complex buffer mode.

With linear buffer mode, the entire buffer is treated as one unit. It is not necessary to keep track of beginning or end addresses in the buffer or to specify any further information governing data buffer output. Complex buffer mode offers the full power of the DAC488HR buffer management, but adds complexity in the specification of how the buffer is output.

3.5.3.1 Linear Buffer Mode

Linear buffer mode eases buffer management for simple applications. It is intended for users who want to specify an arbitrary waveform, write the data into the analog data buffer and output it a specific number of times in response to a trigger. The entire buffer is treated as one unit. It is not necessary to keep track of beginning or end addresses in the buffer or specify any more information to govern data buffer output.

With linear buffer mode, buffer data are output starting from buffer location 0 until the last buffer location written. The last buffer location is the highest location of the data buffer written by the user. The buffer count defines how many times the entire buffer is to repeat in response to a trigger event.

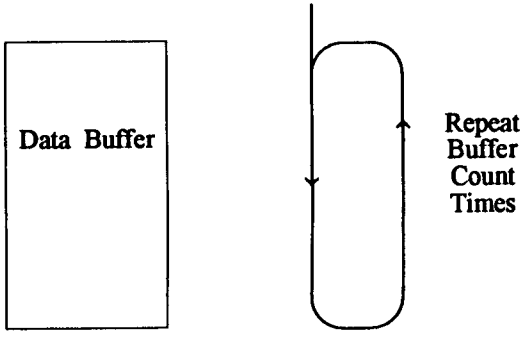


Figure 3.6: Linear Buffer Mode

3.5.3.2 Complex Buffer Mode

Complex buffer mode is intended for applications with complex or memory intensive waveforms or with multiple waveforms that fit in a large buffer with infrequent switching between them. Complex buffer mode conserves memory and allows switching between multiple waveforms stored in the same data buffer. Waveform segments used multiple times within a complex waveform output only need to be written and stored in the data buffer once. Complex buffer mode offers the full power of the DAC488HR buffer management, but adds complexity in the specification of how the buffer is output.

When complex buffer mode is used, it is not necessary to write more than one cycle of a periodic waveform, since that cycle may be repeated, or looped, many times. Multiple waveforms may be stored in the data buffer and the control table changed to output different waveforms from the same data buffer.

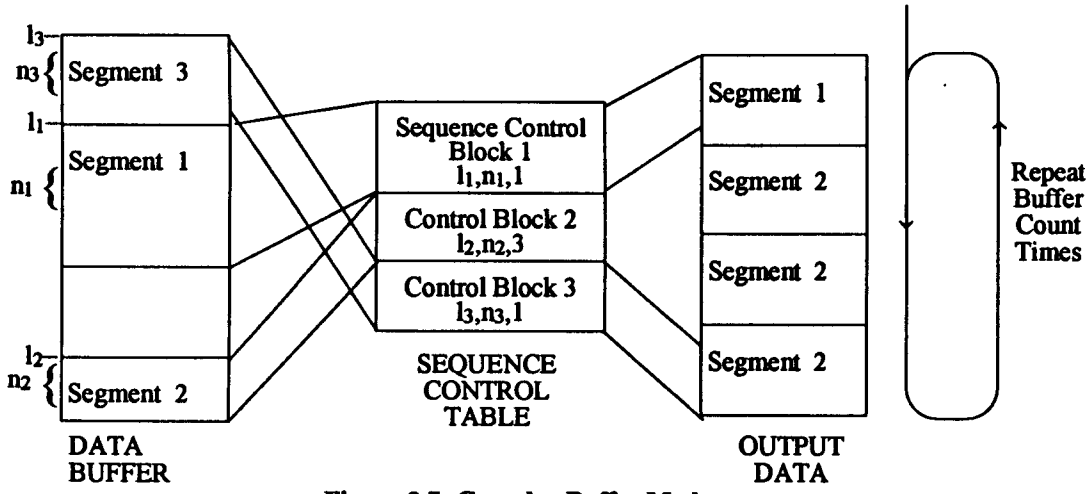


Figure 3.7: Complex Buffer Mode

3.5.3.2.1 Sequence Control Table

The sequence control table defines how data are output in complex buffer mode. This table contains structures called sequence control blocks that define the position and order of the data buffer segments to output and allow repeating, or looping, of segments of the Data Buffer.

Sequence control blocks are user-defined. Limited editing can be done to the sequence control table. Sequence control blocks may be defined, deleted, or inserted into the middle of an already defined sequence control table.

To completely save the state of an analog output port's data buffer when complex buffer control is used, it is necessary to save both the data buffer and the sequence control table. The sequence control table can be read from or written to using either integer decimal or hexadecimal format. All of the data formats can only be loaded through the command channel.

The size of the sequence control table is dependent on data buffer size.

Unit Memory	Data Buffer Size	Sequence Control Table Size
Base Unit	8192 samples	128 Sequence Control Blocks
MEMX3	128 Ksamples	2048 Sequence Control Blocks
MEMX4	480 Ksamples	2048 Sequence Control Blocks

3.5.3.2.2 Sequence Control Blocks

Sequence control blocks define arbitrary segments of the data buffer and control the order in which these segments are output. In complex buffer mode, it is not necessary to output data in the order that they appear in the data buffer. Instead, multiple sections of the buffer may be output in any order by defining the correct sequence control blocks.

Sequence control blocks also allow specified segments of the data buffer to be repeated, or looped, up to 65,535 times. Looping extends the usefulness and effective depth of the data buffer. It permits long periodic waveforms to be stored in the data buffer as one period of the waveform.

Sequence control blocks are defined by a port command. The starting point in the buffer, the length of the segment, and the number of times it is to be repeated are parameters of this command. The length of a repeated segment must be no less than 32 samples, and the repeat count can be no more than 65,535. Loop segments may overlap.

3.5.3.3 Changing Buffer Mode

Changing buffer modes after data have been written into the data buffer erases the contents of the sequence control table but leaves the data buffer undisturbed.

3.5.3.4 Buffer Count

In Linear Buffer Mode, the buffer count specifies the number of times that the data buffer is output before recognition of triggers is disabled by the analog output port.

In Complex Buffer Mode, the buffer count specifies the number of times that the sequence control table is executed before recognition of triggers is disabled by the analog output port. Depending on how the sequence control table is organized, each execution may involve all or part of the entire buffer.

3.6 Update Clock

The update clock controls when the analog output voltages start to change. It is used when updating the analog output ports. A single update clock is shared by all analog output ports to insure synchronization of output changes. The rising edge of the update clock initiates the change in analog output voltage.

The update rate is the time between update clocks. It is programmable by the user via system commands. A wide variety of update rates are provided to meet most application requirements.

The update clock is generated from one of four internal clock sources or from an external clock input. The internal clock sources offer several frequencies. The external clock input (up to 10 MHz) permits synchronization to an external frequency.

Synchronous and asynchronous update clock modes are provided to distinguish the way the analog output ports respond to triggers.

The update clock is buffered and routed to a digital output on the DAC488HR to allow the user to synchronize external equipment to the analog output port changes.

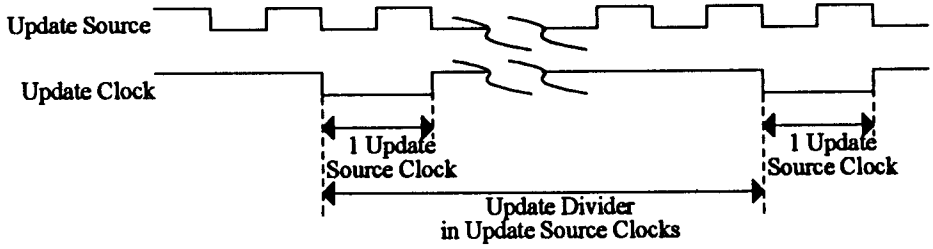


Figure 3.8: Update Clock Timing Diagram

3.6.1 Internal Clock Settings

The four internal clock settings meet a wide variety of applications, including CD and DAT testing.

When internal clock settings are used, a 16-bit divide counter follows the selected clock, allowing frequency division from 2 to 65,536. Typically, a 200 kHz clock source with a divide rate of two is used to output data at the full 100 kHz maximum rate of the D/A converters. The 16-bit counter also allows output updates at divisors of the 100 kHz rate (50 kHz, 33 kHz, 25 kHz, etc.)

Selection of the 2.8224 MHz clock with a divisor of 64 yields the standard audio CD sampling rate of 44.1 kHz. Using a divisor of 32 obtains a sample rate of 88.2 kHz for two times oversampling.

Selection of the 3.072 MHz clock with a divisor of 64 yields the standard audio DAT sampling rate of 48 kHz. Using a divisor of 32 obtains a sample rate of 96 kHz for two times oversampling.

3.6.2 External Clock Source

The external clock source allows synchronization of the analog output ports to an external frequency reference. External clock rates up to 5 MHz may be input and divided by the 16-bit divider. In no case may the resulting update rate exceed the 100 kHz rate of the analog output ports. The external clock can also be used directly as the update rate, providing the minimum external clock specifications are met. The minimum logic high time is 100ns and the minimum logic low time is 100ns.

An example of use of the external clock is to synchronize the DAC488HR to the IOtech ADC488-series Digitizers for stimulus/response applications.

3.6.3 Update Modes

The update rate can be generated in synchronous or asynchronous mode. The DAC488HR responds to triggers differently depending on the selected mode.

When synchronous update mode is used, triggers are recognized at the next regular occurrence of the update clock. Maximum trigger latency is equal to the update rate. Synchronous mode guarantees that no discontinuities in the update rate occur.

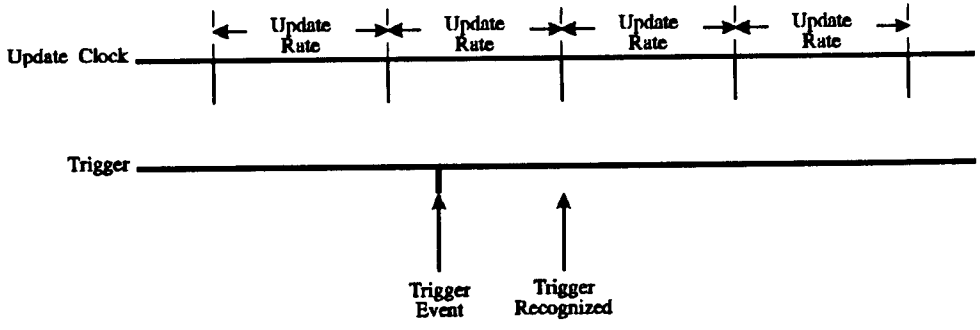


Figure 3.9: Synchronous Update Mode

When asynchronous update mode is used, triggers are recognized as quickly as possible. Update rate generation is resynchronized to the occurrence of the trigger event. The minimum trigger latency in this case is two periods of the update source rate. Thus, with an update

source of 5 MHz, asynchronous update mode, minimum trigger latency is 400 nanoseconds. The maximum trigger latency is equal to the time it takes to shift the next sample out to the analog circuitry. This is four microseconds.

In addition, further update clocks are resynchronized to the occurrence of the trigger event.

This mode minimizes trigger latency, but can cause discontinuity in the update rate of all analog output ports for every occurrence of a trigger event.

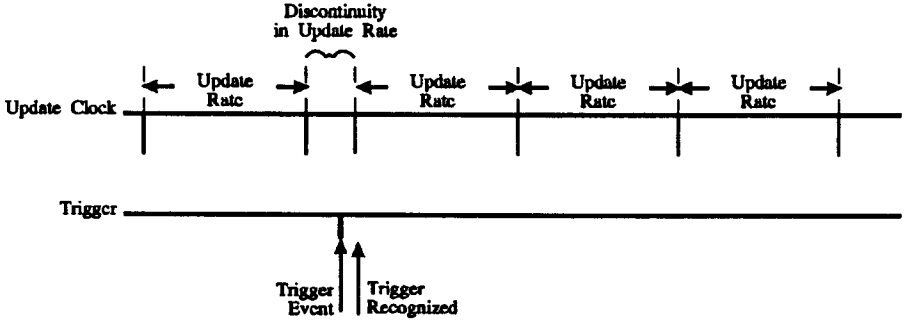


Figure 3.10: Asynchronous Update Mode

3.7 Analog Output Port Triggering

Triggering is the process of changing an analog output at the occurrence of an event. It provides a means of synchronizing multiple analog output ports for applications with critical time and phase relationships.

Trigger sources all work from the same trigger signal generated by the DAC488HR. When a trigger event occurs, the programmed voltages simultaneously appear at the analog output ports. Each analog output port can be independently programmed for the desired voltage and response to trigger events.

3.7.1 Trigger Sources

There are four standard trigger sources:

External
Group Execute Trigger (GET)
Command Trigger (@) and
Interval Timer trigger.

All analog output ports must work from the same selected trigger source, although the response to a trigger can be independently configured (see Section 3.8, Trigger Control Mode).

3.7.1.1 External Trigger Source

An external trigger is a TTL-level signal that may be applied as a trigger source for DAC488HR operation. The rising edge, falling edge, or both edges of the external trigger may be selected as the trigger events.

3.7.1.2 Group Execute Trigger (GET) Source

A Group Execute Trigger is generated when GET is issued on the IEEE 488 bus when the DAC488HR is a listener. It is normally used to synchronize DAC488HR actions with other devices on the IEEE 488 bus.

If GET is detected while the DAC488HR is busy processing commands and GET is specified as the trigger source, the GET trigger is held off on the IEEE 488 bus until command processing is complete. Once command processing is complete, the holdoff for GET is released on the IEEE 488 bus, and the DAC488HR responds.

3.7.1.3 Command Source

The command trigger is provided by issuing a specific command over the IEEE 488 bus to the DAC488HR. After the entire line has been processed and all necessary state changes made to the analog output port, a trigger is generated under program control to initiate the desired analog output.

3.7.1.4 Interval Timer Trigger Source

An interval timer trigger generates regular trigger events at periodic time intervals. The time interval is adjustable in one millisecond increments from 1 millisecond to 65,535 milliseconds.

3.7.2 Analog Output Port Trigger Processing

Although the trigger signal is common to all analog output ports, the response of each port is controlled individually by the trigger control mode.

A trigger is recognized by an analog output port at the next update clock. If a trigger event occurs before the analog output port has finished processing a previous one, the latter trigger event is ignored and a trigger overrun error is generated.

3.7.3 Trigger OUT

The recognized trigger signal is buffered and routed to a digital output (`Trigger OUT` pin) on the DAC488HR. This allows the user to synchronize external equipment to DAC488HR operation.

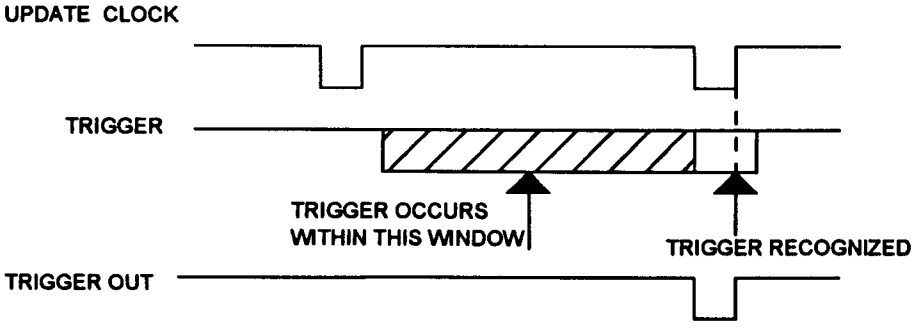


Figure 3.11 Trigger Timing Diagram

3.7.4 Delayed Trigger Output

A delayed trigger output is also provided. It is used with applications that require some delay or setup time after the application of a signal from the DAC488HR before meaningful data can be measured. The delayed trigger output signal is common to all analog output ports. The time delay is specified in update clocks to any value from 1 to 65,535 update clocks. The trigger delay time is the time between the update of an analog output port and the delayed trigger output. At a 100 kHz update rate, the delayed trigger time may be set in 10µsec increments from 10µsec to 0.65 seconds.

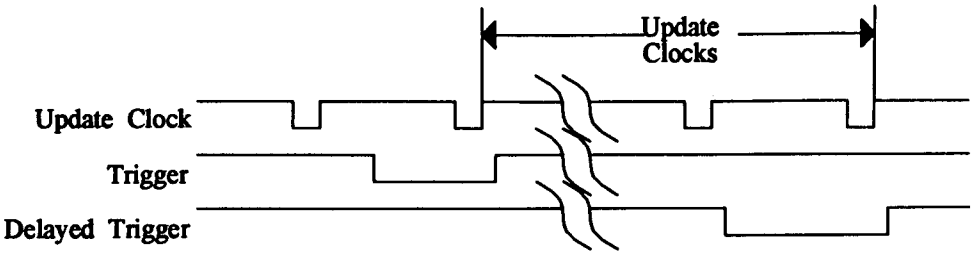


Figure 3.12: Delayed Trigger Timing Diagram

The delay is retriggerable, so if another valid trigger occurs before the trigger delay time has elapsed, the time delay is a full trigger delay after the second valid trigger.

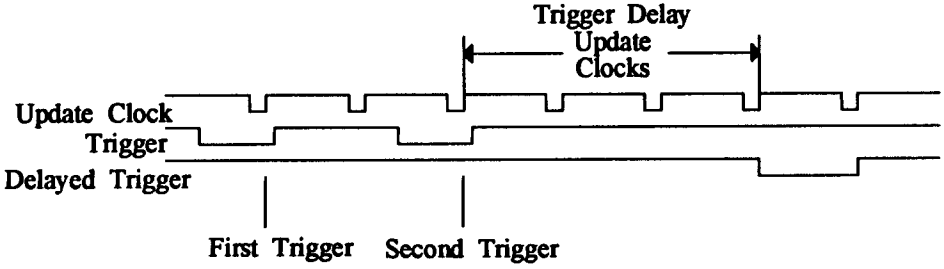


Figure 3.13: Retriggered Delayed Trigger Timing Diagram

3.8 Trigger Control Modes

The trigger control modes define how an analog output port responses to a trigger. Each analog output port maintains its own separate and distinct trigger control mode. The five available trigger control modes are:

Step
Burst
Waveform
Continuous and
Immediate.

Although not strictly a trigger control mode, direct control is included in the descriptions below because of its usefulness in conjunction with any trigger control mode.

3.8.1 Direct Control

In direct control mode, a voltage is output upon receipt of the voltage value command. Direct control is accomplished by selecting the analog output port and range and specifying the output voltage. One use of direct control mode is to output voltages from one or more analog output ports directly under program control from an IEEE 488 bus controller.

Direct control is useful with any of the trigger control modes in two different ways.

When an output voltage is set using the voltage value command prior to issuing a trigger control mode to an analog output port, the programmed output voltage is maintained until a trigger event is detected. Thus, direct control output can set an initial output voltage to be held until a trigger event occurs.

3.8.2 Step

Step mode causes an analog output port to output the next value from the data buffer each time a trigger is detected. After the data are output, trigger detection is rearmed. When the end of the defined data buffer is reached, the data buffer is repeated. When the buffer count is exhausted, the output is held at the last output value and trigger detection is disarmed.

3.8.3 Burst

Burst mode causes the data buffer to output data at the system update rate each time a trigger is detected. When the end of the data buffer is reached, the output is held at the last output value and trigger detection is rearmed. Triggers are accepted and the data buffer is repeated until the buffer count is exhausted.

After the buffer count is exhausted, the output is held at the last output value and trigger detection is disarmed. If the buffer count is set to 0, bursts are repeated forever in response to triggers.

3.8.4 Waveform

When waveform mode is selected, the data buffer outputs at the system update rate when a trigger is detected. When the end of the buffer is reached, it is repeated until the buffer count is exhausted. After the buffer count is exhausted, output is held at the last output value and trigger detection is disarmed. If buffer count is set to 0, the data buffer repeats forever.

3.8.5 Immediate

Immediate mode is similar to waveform mode, except that a trigger is not required to initiate the data buffer output. When the command to set this trigger control mode is issued, the data buffer is output at the system update rate. When the end of the buffer is reached, it

is repeated until the buffer count is exhausted. After the buffer count is exhausted, output is held at the last output value. If the buffer count is set to 0, the data buffer repeats forever.

3.8.6 Continuous

Continuous mode causes data to be output continuously from the IEEE 488 bus to one or two analog output ports. The output rate is the maximum of 100 Ksamples per second for one or two analog output ports. The data must be output in raw binary format to the correct secondary address corresponding to an analog output port. When outputting interleaved data to two analog output ports, raw binary data may be sent to the secondary address associated with either of the two active analog output ports. Interleaved data consist of two bytes of binary data for one port, then two bytes of binary data for the other, and so on. In order to set which analog output port captures which set of binary data, two different modes are provided (see trigger control mode command for details). Interleaving data is not possible with formats other than binary and continuous trigger control mode.

If an analog output port does not receive data at the update rate, output is held at the last received output value, output will resume when more data becomes available and a DMA underrun error is generated.

3.9 Digital Inputs

3.9.1 Digital Input Connector

A female DB-15 connector is provided on the rear panel of the DAC488HR for connection to the digital TTL inputs. Normal precautions should be taken to limit the input voltages to –0.3 to +7.0 volts. All input lines are referenced to digital ground (Pins 11, 12, and 15).

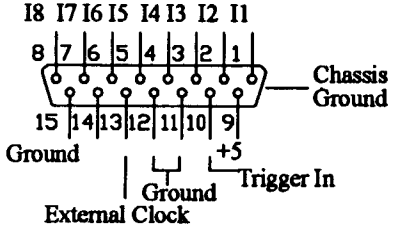


Figure 3.14: Digital Input Connector Pinout

3.9.1.1 8-Bit TTL Input Port (Pins 1 - 8)

Eight TTL-compatible digital input bits are provided for general system control. These eight bits may be read by issuing a status command. The TTL inputs require sinking 0.2 mA @ <0.4 V low or sourcing 20µA @ >2.7 V high.

3.9.1.2 Power (Pin 9)

A connection to the +5 V digital logic supply of the DAC488HR is also provided on this connector. If some minimum amount of logic is needed before the TTL inputs, this line may be used to power that logic.

It is recommended that no more than 50 mA be drawn from this pin. Care should also be taken that this line is not shorted to ground. Although the +5 V power supply is current-limited and no damage to the DAC488HR will result, a short to ground will keep the DAC488HR from operating.

3.9.1.3 External Clock (Pin 13)

An external TTL clock input is provided to allow the analog output port update rate to be set by an external update clock. (For more information on external clock source, refer to Section 3.6.2.)

3.9.2 Trigger In BNC Connector

An external TTL trigger input is provided on the rear panel through a BNC connector marked TRIGGER IN. This signal may be selected under program control as the trigger source. The rising, falling, or either edge can be selected for the trigger event. This input is also connected to Pin 10 on the Digital Input Connector.

3.10 Digital Outputs

3.10.1 Digital Output Connector

A female DB-15 connector is provided on the rear panel of the DAC488HR for connection of outputs to enable control and synchronization with internal DAC488HR processes.

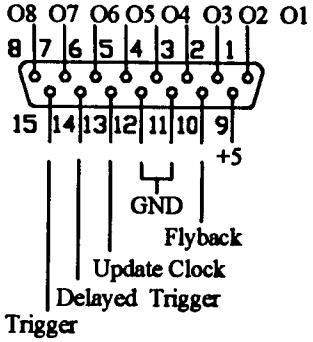


Figure 3.15: Digital Output Connector Pinout

3.10.1.1 8-Bit TTL Output Port (Pins 1-8)

An 8-bit digital output port is provided for general purpose control of application specific lines external to the DAC488HR. All eight lines may be individually set or cleared under program control.

3.10.1.1.1 High Voltage/High Current Outputs

In addition to interfacing with TTL logic levels, the digital output lines can be configured as high voltage/high current outputs. These outputs can sink up to 100 mA at 50 V dc through the use of open collector drivers with integral diodes for inductive load transient suppression. This allows for interfacing the digital outputs with relays, lamps and solenoids.

As shown in the schematic below, the gates driving the digital outputs (when configured for high voltage/high current operation) contain internal diodes to suppress inductive transients. The cathodes of these diodes are tied together and connected to the Flyback pin on the digital output port. By connecting the Flyback pin to the positive end of the supply driving the relays, the internal diodes are connected in parallel with the relay coils, preventing inductive spikes from damaging the internal circuitry of the DAC488HR. See Section 2.2.1.2 for instructions on changing to high voltage/high current outputs.

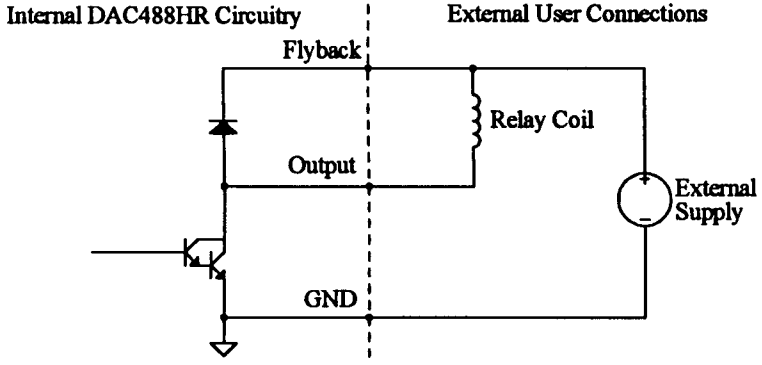


Figure 3.16: High Current/High Voltage Output Circuitry

3.10.1.2 Power (Pin 9)

A connection to the +5 V digital logic supply of the DAC488HR is also provided on this connector. If some minimum amount of logic is needed after the TTL outputs, this line may be used to power that logic.

It is recommended that no more than 50 mA be drawn from this pin. Care should also be taken that this line is not shorted to ground. Although the +5 V power supply is current-limited and no damage to the DAC488HR will result, a short to ground will prevent the DAC488HR from operating.

3.10.1.3 Flyback (Pin 10)

When used with relays or solenoids, the Flyback pin on the DB-15 digital output connector should be connected to the positive supply lead of the power supply used with the external devices being driven, as shown in Figure 3.16. See previous section on high voltage/high current outputs.

3.10.1.4 Trigger Output (Pin 13)

The trigger signal is also buffered and routed to a digital output pin (Trigger) on the DAC488HR. This allows the user to synchronize external equipment to DAC488HR operation.

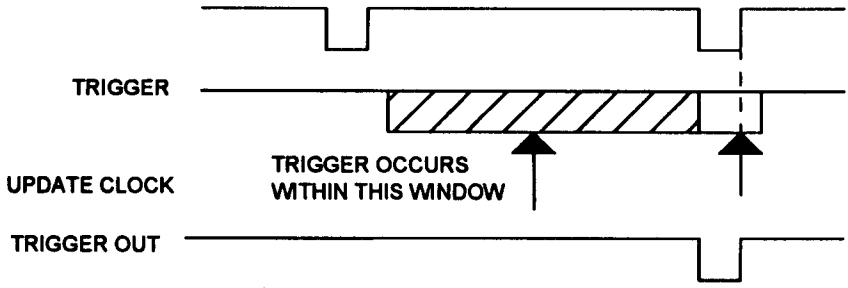


Figure 3.17 Trigger Timing Diagram

3.10.1.5 Delayed Trigger Output (Pin 14)

The delayed trigger output is used with applications that require some delay or setup time after the application of a signal from the DAC488HR before meaningful data can be measured. The delayed trigger output signal is common to all analog output ports. The time delay is

specified in update clocks to any value from 1 to 65,535 update clocks between when any analog output port reacts to a trigger and when the delayed trigger output goes true. At a 100 kHz update rate, this yields 10µsec increments from 10µsec to 0.65 seconds.

The delay is retriggerable, so if another valid trigger occurs before the trigger delay time has elapsed, the time delay is a full trigger delay after the second valid trigger.

3.10.1.6 Update Clock Output (Pin 15)

The Update Clock signal used by all analog output ports to change analog output values is buffered and presented to this pin. This signal may be used to synchronize external processes with changes in analog output voltage.

3.10.2 Trigger Out BNC Connector

The delayed trigger out is also routed to the BNC connector marked `Trigger Out` on the rear panel of the DAC488HR.

3.11 User Defined System Defaults

A command is provided to save the calibration constants in non-volatile RAM. The command may also be used to save a particular configuration of the DAC488HR as a power on default configuration. The system and port commands configure the DAC488HR for a particular application which can be saved as the power-on default configuration.

3.12 IEEE 488 Bus Implementation

The DAC488HR implements many of the capabilities defined by the IEEE 488 1978 specification. These are discussed in the following sections.

Commands Not Supported by DAC488HR

Those bus uniline and multiline commands that the DAC488HR does not support or respond to include:

Remote Enable (REN)	Parallel Poll (PP)
Go to Local (GTL)	Parallel Poll Configure (PPC)
Local Lockout (LLO)	Parallel Poll Unconfigure (PPU)
Take Control (TCT)	Parallel Poll Disable (PPD)

Commands Supported by DAC488HR

My Talk Address (MTA)

When the DAC488HR is addressed to talk on the command channel, it returns the message strings that have been requested by previous commands. If no requests for returned message strings have been issued, then no data are returned.

When the DAC488HR is addressed to talk on the data channel, data are returned from the selected analog output port's data buffer.

My Listen Address (MLA)

When the DAC488HR is addressed to listen on the command channel, it accepts characters from the active talker and interprets these characters as commands and command parameters. These commands are explained in Section 5.

When the DAC488HR is addressed to listen on the data channel, it accepts characters from the active talker and interprets them as data for the selected analog output port's data buffer.

Device Clear (DCL and SDC)

Device clear clears the command input buffer, the command response queue and any pending commands.

Group Execute Trigger (GET)

When the DAC488HR recognizes a GET on the IEEE 488 bus, and the Trigger Source command is set for GET, the programmed voltage or voltage sequence will be output on the analog output port.

Interface Clear (IFC)

IFC places the DAC488HR in the Talker/Listener Idle state.

Serial Poll Enable (SPE)

When Serial Poll is enabled, the DAC488HR sets itself to respond to a serial poll with its serial poll status byte if addressed to talk. When the serial poll status byte is accepted by the controller, any pending Service Requests (SRQs) are cleared. The DAC488HR continues to try to output its serial poll response until it is serial poll disabled by the controller.

Serial Poll Disable (SPD)

Disables the DAC488HR from responding to serial polls by the controller.

Unlisten (UNL)

UNL places the DAC488HR in the Listener Idle state.

Untalk (UNT)

UNT places the DAC488HR in the Talker Idle state.

Serial Poll Response

Whenever the DAC488HR generates a service request (SRQ), a serial poll returns a serial poll status byte of at least 64 (decimal), showing that the SRQ was generated by the DAC488HR. For complete details on SRQ generation, see Section 5.7, the Serial Poll model.

Note: To allow the DAC488HR to set SRQ on certain conditions, the Service Request Mask command must be enabled for each condition.

IEEE 488 Bus Output Terminators

Bus terminators are fixed and cannot be altered by the command set. The DAC488HR recognizes either linefeed, EOI, or both as an input terminator. The DAC488HR terminates data output with linefeed plus EOI.

Programming the DAC488HR

This section describes how to program the DAC488HR with an explanation of basic and advanced features and the use of walk-thru program examples. Although these examples are written in QuickBASIC, they can easily be ported to any desired language. All of the examples use Driver488/DRV as a basis for communicating on the IEEE 488 bus. A complete listing of example programs is included at the end of this chapter, as well as on the DAC488HR examples diskette.

4.1 Static DC Output

To begin programming, the DAC488HR should be initialized with the following command sequence:

```
PRINT #1,"OUTPUT 10;*RX"
```

The Driver488/DRV `OUTPUT` command is used to send commands to the specified device. In this case, the device is `10` which is the address for the DAC488HR. The `*R` command returns the DAC488HR to its power up state. Note the `*R` command will take a few seconds to complete.

After ga one second wait, the unit may be serial polled in order to determine if it has completed the initialization. The ready bit (0x04H) will be set when the DAC488HR is ready for additional commands. This step can easily be accomplished by using the following serial poll loop:

```
spoll = 0
WHILE NOT (spoll AND 4)
   PRINT #1,"SPOLL 10"
   INPUT #2, spoll
LOOP
```

Next a port and range for the desired voltage must be selected.

```
PRINT #1,"OUTPUT 10;P1X"
PRINT #1,"OUTPUT 10;R4X"
```

These commands specified the port as "port 1," and the range as "10 volt bipolar." Now a voltage can be programmed.

```
PRINT #1,"OUTPUT 10;V5X"
```

Five volts should now be present at the analog output terminals of port 1. In order to change the voltage, simply replace the `5` in the preceding command with the desired voltage.

4.2 Predefined Waveform Generation

The DAC488HR also has built in waveform generation capabilities for three common waveform types: Sine, Triangle and Square. Programming a standard waveform is actually quite simple. First, the waveform data must be generated. In the following example, a 60Hz sine wave will be defined. To begin, the desired port and output voltage range must be selected.

```
PRINT #1,"OUTPUT 10;P1X"
PRINT #1,"OUTPUT 10;R4X"
```

The desired waveform can now be created. The following command will generate a sine wave (W0) with a length of 1666 samples, 100% duty cycle and 50% symmetry. The output data rate defaults to 100KHz at power on. As just demonstrated, the voltage range was set to 10V bipolar.

```
PRINT #1,"OUTPUT 10;W0,1666,100,-100,50X"
```

Next the waveform repeat count must be specified. Since the waveform will be output indefinitely, the repeat count must be set to 0, as indicated below.

```
PRINT #1,"OUTPUT 10;K0X"
```

Finally, all that is necessary to initiate the waveform generation is a trigger event. For this example, the immediate trigger event will be used. This means that as soon as the DAC488HR receives the trigger command, it will begin waveform generation.

```
PRINT #1,"OUTPUT 10;C4X"
```

At this point there should be a 20 V p-p 60 Hz sine wave being generated by port 1.

4.3 High Speed Buffer Loading

Basic operations of the DAC488HR are fairly easy to understand and implement. However, the advanced features are what make the DAC488HR a much more powerful tool than what has been described so far. This section describes these features and shows how easy it is to implement them. Features that will be covered include:

- High speed binary waveform buffer loading
- Sequence definition
- Continuous binary transfers to one channel
- Continuous binary transfers to two channels

The DAC488HR offers a high-speed binary waveform buffer loading feature that allows large amounts of data to be loaded quickly via the IEEE 488 bus. The burst transfer rate is in excess of 400Kbytes/second. This allows large blocks of arbitrary waveform data to be sent to the selected port's waveform buffer in a matter of seconds.

Suppose that we were to generate our own arbitrary waveform on the IEEE controller and wanted to send it to a selected port's waveform buffer. The steps involved to complete this task are outlined on the following pages. (The complete program can be found in Section 4.7 of this manual and on the DAC488HR examples diskette as EXAMPLE1.BAS.)

The first series of lines will DIMension the necessary storage space needed by the program.

```
DIM SHARED wavebuffer(256) AS INTEGER
DIM SHARED seg1 AS STRING * 4
DIM SHARED off1 AS STRING * 4
DIM SHARED out1 AS STRING * 40
```

Next we must OPEN Driver488/DRV for communications.

```
OPEN "\DEV\IEEEOUT" FOR OUTPUT AS #1
IOCTL #1, "BREAK"
PRINT #1, "RESET"
OPEN "\DEV\IEEEIN" FOR INPUT AS #2
PRINT #1, "FILL ERROR"
```

As a precaution, the DAC488HR should be reset to a known power on condition. This is followed by a SPOLL loop to query the DAC488HR to see if it has finished its power on operations.

```
PRINT "Resetting DAC488HR..."
PRINT #1, "OUTPUT 10;*RX"
SLEEP 1, 'Wait one second
spoll = 0
WHILE NOT spoll AND 4
    PRINT #1, "SPOLL 10"
    INPUT #2, spoll
WEND
```

Once that has been accomplished, we can begin the actual programming. The next series of lines program the waveform output attributes. First, the update source and rate must be selected.

```
PRINT  #1, "OUTPUT 10;G0I50X"  'Set update source 5MHz,divisor 50
                               '(100K update)
```

Next the waveform buffer mode should be selected. There are two options for this attribute: 0 indicates linear buffer mode and 1 indicates complex buffer mode. In linear buffer mode, the DAC488HR will automatically generate the sequence that describes how the waveform is output. In this mode, a sequence is generated based on the last buffer location written to. This allows the user to write several different blocks of data into several areas of the waveform buffer without generating a sequence. As soon as the user selects the trigger mode to be used, the sequence will be written.

In complex buffer mode (a buffer mode of 1), it is up to the user to define the sequence(s) that will be used to describe the output of buffer data.

In this example, port 1 will be selected for linear buffer mode operation using the A0 command.

```
PRINT #1, "OUTPUT 10;P1XA0X"   'Select port 1, linear buffer mode
```

Next the range must be selected and then the buffer repeat count has to be defined.

```
PRINT #1, "OUTPUT 10;R4X"      'Select range 4
PRINT #1, "OUTPUT 10;K0X"      'Select infinite repeat count
```

Once the output attributes have been taken care of, we can define the data that will be used. For this example, we will generate a sine wave composed of 256 discrete steps.

```
PRINT "Calculating waveform..."
angle! = (6.28318 / 256)
FOR samples% = 1 TO 256
   points! = SIN(angle! * samples%)
   points! = points! * 32767
   wavebuffer(samples%) = INT(points!)
NEXT samples%
PRINT "Sending data to DAC488HR..."
```

At this point our sine wave data has been calculated, is in memory, and ready to transmit using the next command line. This statement requires some explanation. The #6; is a Driver488/DRV directive that indicates how many bytes of information following the semicolon will be sent to the selected device (10). The actual command to the DAC488HR is composed of three components. First the B tells the DAC488HR that buffer data is to follow. The 3 following the pound sign tells the DAC488HR that there are three digits which describe how many bytes of data will be transferred. The 512 tells the DAC488HR how many bytes are to be transferred. (No terminators can be sent with this command because the DAC488HR would input them as data bytes instead of terminators.)

```
PRINT#1, "OUTPUT 10 #6;B#3512"
```

Now we have to tell Driver488/DRV where the data can be found, and how much data we are going to move. The Driver488/DRV command format is explained as follows:

OUTPUT	Driver488/DRV command to output data
10	The device the data will be sent to
#512	Number of bytes to be sent to the selected device
BUFFER	Data will be sent directly from memory starting at the segment and offset addresses
SEGMENT:OFFSET	The addresses of the segment and offset

The command sequence is:

```
seg1 = HEX$(VARSEG(wavebuffer(1)))
off1 = HEX$(VARPTR(wavebuffer(1)))
PRINT "Sending waveform data to DAC488HR..."
out1 ="OUTPUT 10 #512 BUFFER &H" + seg1 + ":&H" + off1
PRINT #1, out1
```

Once the data has been sent, the transfer must be terminated properly. Again, this operation uses some specific DRIVER488/DRV syntax, as described in the chart below:

SEND	This Driver488/DRV command indicates that the next byte to be sent will be described as an ASCII number.
EOI	Informs Driver488/DRV to assert EOI with the next data byte transferred.
88	The ASCII character code for 'X'.

The command to terminate the transfer with a X EOI combination is:

```
PRINT#1, "SEND EOI 88"
```

Next we must set the trigger mode.

```
PRINT #1, "OUTPUT 10;C4X" 'Set trigger mode immediate
```

Upon reception of this command, the DAC488HR will automatically generate the sequence which will describe how the wavefom data will be output. This is caused by using the linear buffer mode which we previously selected. At this point, the DAC488HR will be generating the sine wave on port 1.

Waveform generation will continue until the user presses any key. The following lines will cause the program to wait for a key press.

```
PRINT "DAC is now generating waveform."
PRINT "Press any key to stop..."
WHILE INKEY$ = ""
WEND
```

Once the user has pressed a key, waveform output can be halted by entering the following command line.

```
PRINT #1, "OUTPUT 10;C0X" 'Cease waveform output.

END
```

4.4 Single Channel Continuous Transfer Mode

Continuing with the advanced features of the DAC488HR, this section covers how to use continuous binary transfers to send long arbitrary waveforms to a single DAC488HR port. There can be instances when it is necessary to send an arbitrary waveform to the DAC488HR which exceeds the available buffer space. One such example would be audio data which has been previously digitized using an analog-to-digital converter.

Since the existing buffer data cannot be updated while the DAC488HR is outputting data, this approach is not feasible. However, there is a mode available which allows a continuous stream of binary data to be sent directly to the DAC488HR. The only requirements are that the IEEE controller must be capable of maintaining a throughput rate greater than the update rate times 2.

For example, if the output rate of the arbitrary waveform is to be 50KHz, then the throughput rate should be grater than 100Kbytes/second. The reason for this is that there is a certain amount of overhead which occurs between the transmission of large blocks of binary data. This overhead must be compensated for by transfer rates greater than the waveform update rate.

The DAC488HR buffers approximately 4Kbytes of data internally when it is using continuous waveform mode. This 4K is split up into two 2K buffers. While one buffer is being output to the digital-to-analog converter, the other buffer is being filled. At an update rate of 100K, this buffer equals approximately 10 mS of waveform data.

If the data stream is interrupted for more than 10 mS, data starvation will occur. When the DAC488HR detects data starvation, the analog output level will be held at the last value that was sent. When new data becomes available, the waveform output will continue. Therefore, it is imperative that the IEEE controller be able to maintain data transfer rates.

The next example program (EXAMPLE2.BAS) shows how to implement continuous binary transfers.

As always, first DIMension all arrays and variables to be used.

```
DIM SHARED outbuffer(256) AS INTEGER
DIM SHARED seg1 AS STRING * 4
DIM SHARED off1 AS STRING * 4
DIM SHARED out1 AS STRING * 40
```

Next open Driver488/DRV for communications.

```
OPEN "\DEV\IEEEOUT" FOR OUTPUT AS #1
OPEN "\DEV\IEEEOUT" FOR BINARY AS #3
IOCTL #1, "BREAK"
PRINT #1, "RESET"
OPEN "\DEV\IEEEIN" FOR INPUT AS #2
PRINT #1, "FILL ERROR"
```

Reset the DAC488HR.

```
PRINT "Resetting DAC488HR..."
PRINT #1, "OUTPUT 10;*RX"
SLEEP 1, 'Wait one second
spoll = 0
WHILE NOT spoll AND 4
   PRINT #1, "SPOLL 10"
   INPUT #2, spoll
WEND
```

The waveform output attributes can now be set up.

```
PRINT #1, "OUTPUT 10;G0I100X" 'Select update source 5M, divisor 100
                              '(50K update rate)
PRINT #1, "OUTPUT 10;P1A0X"   'Select port 1, linear buffer mode
PRINT #1, "OUTPUT 10;P1R4X"   'Select range 4
PRINT #1, "OUTPUT 10;P1C5X"   'Select continuous update mode
```

Next we need to calculate waveform to be used.

```
PRINT "Calculating waveform..."
angle! = (6.28318 / 256)
FOR samples% = 1 TO 256
   points! = SIN(angle! * samples%)
   points! = points! * 32767
   outbuffer(samples%) = INT(points!)
NEXT samples%
PRINT "Sending data to DAC488HR..."
```

Finally, we need to tell the DAC488HR to expect continuous waveform data. The #0 command tells the DAC488HR that an undetermined amount of binary data is going to be sent. In order to send this command properly a Driver488/DRV specific command will be used. The #2; following the OUTPUT statement tells Driver488/DRV to send only the first two bytes following the semicolon. If just the OUTPUT statement had been used, terminator characters would have been sent along with the #0 command. This would result in incorrect data being sent to the DAC488HR.

```
PRINT #1, "OUTPUT 10 #2;#0"
```

Now we have to tell Driver488/DRV where the data can be found and how much data we are going to move. The Driver488/DRV command format is explained as follows:

`OUTPUT`	Driver488/DRV command to output data
`10`	The device the data will be sent to
`#512`	Number of bytes to be sent to the selected device.
`BUFFER`	Data will be sent directly from memory starting at the segment and offset addresses
`Segment: Offset`	The addresses of the segment and offset
`DMA`	Use Direct Memory Access. DMA must be used for this example. If DMA is not used, discontinuities may appear in the output waveform due to data starvation.
`CHR$(13) + CHR$(10)`	This is necessary because of the `PUT` command which is being used to send the data to Driver488/DRV. When `PRINT` is used, terminators are automatically appended to the end of the command string. However, when `PUT` is used, no terminators are appended. Without the `CHR$(13)` and `CHR$(10)`, Driver488/DRV would not know when the command had been completed.

The command sequence is:

```
seg1 = HEX$(VARSEG(outbuffer(1)))
off1 = HEX$(VARPTR(outbuffer(1)))
PRINT "Sending waveform data to DAC488HR."
PRINT "Press any key to halt..."
out1 = "OUTPUT10#512 BUFFER &H" + seg1 + ":&H" + off1 + "DMA" +_
    CHR$(13) + CHR$(10)
WHILE INKEY$ = ""
   PUT #3, , out1 'Use faster "PUT" command for less latency between
                  'bursts
WEND
```

Once the user presses a key, waveform output can then be halted by the following command:

```
PRINT #1, "SEND EOI 10"          'Terminate transfer with a LF EOI
                                  combination

PRINT #1, "OUTPUT 10;P1XC0X" 'Cease waveform output

END
```

If using either one channel or two channel continuous transfer modes (C5, C6 or C7), after terminating the transfer with a LF EOI it is necessary to wait for the unassertion of the trigger bit (1) in the Serial Poll register.

When the DAC488HR receives a LF EOI in continuous mode, the output is not discontinued immediately. The port continues to output the data remaining in the buffer. During this time, the DAC488HR will not respond to commands.

After the last sample in each of the daughterboards is output, the trigger bit becomes unasserted. At this point the DAC488HR can receive more commands.

In order to finish termination of the transfer, send the following string to the DAC488HR immediately after the unassertion of the trigger bit:

"P1c0X" where port 1 is used.

"P1c0XP2c0X" where ports 1 and 2 are used.

4.5 Two Channel Continuous Transfer Mode

The next DAC488HR programming topic is how to use the two channel continuous binary transfer mode. In this mode, data is sent to the two selected DAC488HR ports in an interleaved fashion. The order in which data is to be sent is as follows:

- First channel data word
- Second channel data word
- First channel data word data word
- Second channel data word
- Etc.

The rules concerning data starvation are the same as previously described in Section 4.4. If either channel detects data starvation, the analog output level will be held at the last received value, and will restart as soon as more data is available. In this example, EXAMPLE3.BAS, two waveforms will be calculated and sent to two selected channels of the DAC488HR at an update rate of 25KSamples/second.

First DIMension all arrays and variables to be used.

```
DIM SHARED outbuffer(8192) AS INTEGER
DIM SHARED wavebuffer(256) AS INTEGER
DIM SHARED seg1 AS STRING * 4
DIM SHARED off1 AS STRING * 4
DIM SHARED out1 AS STRING * 40
```

Now open Driver488/DRV for communications.

```
OPEN "\DEV\IEEEOUT" FOR OUTPUT AS #1
OPEN "\DEV\IEEEOUT" FOR BINARY AS #3
```

```
IOCTL #1, "BREAK"
PRINT #1, "RESET"
OPEN "\DEV\IEEEIN" FOR INPUT AS #2
PRINT #1, "FILL ERROR"
```

Next reset the DAC488HR.

```
PRINT "Resetting DAC488HR..."
PRINT #1, "OUTPUT 10;*RX"
SLEEP 1, 'Wait one second
spoll = 0
WHILE NOT spoll AND 4
   PRINT #1, "SPOLL 10"
   INPUT #2, spoll
WEND
```

Program the waveform output attributes.

```
PRINT #1,"OUTPUT 10;G0 I200X" 'Select update source 5M,divisor 200
                              '(25K)
PRINT #1, "OUTPUT 10;P1 A0X"  'Select port 1,linear buffer mode
PRINT #1, "OUTPUT 10;P2 A0X"  'Select port 2,linear buffer mode
PRINT #1, "OUTPUT 10;P1 R4X"  'Select port 1,range 4
PRINT #1, "OUTPUT 10;P2 R4X"  'Select port 2,range 4
PRINT #1, "OUTPUT 10;P1 C6X"  'Select port 1,interleave, take
                              'first sample
PRINT #1, "OUTPUT 10;P2 C7X"  'Select port 2,interleave, take
                              'next sample
```

Now we need to calculate waveforms to be used. The length of both waveforms will be 256 samples. Channel 1 will generate a sine wave; channel 2 will generate a triangle wave. Both waveforms will be output at 25 KHz. First calculate the sine wave.

```
PRINT "Calculating waveform 1..."
angle! = (6.28318 / 256)
FOR samples% = 1 TO 256
   points! = SIN(angle! * samples%)
   points! = points! * 32767
   wavebuffer(samples%) = INT(points!)
NEXT samples%
```

Once the waveform has been calculated, it must be placed into the output buffer in interleave order. This means that the waveform words will occupy every other element in the output buffer.

```
PRINT "Placing waveform 1 in output buffer..."
offset% = 1
FOR samples% = 1 TO 511 STEP 2
   outbuffer(samples%) = wavebuffer(offset%)
offset% = offset% + 1
NEXT samples%
```

Calculate the triangle wave data.

```
PRINT "Calculating waveform 2..."
baseline! = 0
stepsize! = 32768 / 64

FOR samples% = 1 TO 64
   wavebuffer(samples%) = INT(baseline!)
   baseline! = baseline! + stepsize!
NEXT samples%
baseline! = 32767
stepsize! = 65536 / 128

FOR samples% = 65 TO 192
   wavebuffer(samples%) = INT(baseline!)
   baseline! = baseline! - stepsize!
NEXT samples%
baseline! = -32768
stepsize! = 32768 / 64

FOR samples% = 193 TO 256
   wavebuffer(samples%) = INT(baseline!)
   baseline! = baseline! + stepsize!
NEXT samples%
```

When the triangle waveform has been calculated, it must be placed into the output buffer, interleaving with the data which describes the sine wave. The reason for placing multiple copies of the waveform in the output buffer is that sending larger blocks of data helps eliminate the effect of latency between data block transfers.

```
PRINT "Placing waveform 2 in output buffer..."
offset% = 1
FOR samples% = 2 TO 512 STEP 2
   outbuffer(samples%) = wavebuffer(offset%)
   offset% = offset% + 1
NEXT samples%

PRINT "Copying data to remainder of output buffer..."
```

```
FOR offset% = 512 TO 7680 STEP 512
   FOR samples% = 1 TO 512
         outbuffer(samples% + offset%) = outbuffer(samples%)
   NEXT samples%
NEXT offset%
PRINT "Sending data to DAC488HR..."
```

Finally, we need to tell the DAC488HR to expect continuous waveform data. The #0 command tells the DAC488HR that an undetermined amount of binary data is going to be sent. In order to send this command properly a Driver488/DRV specific command will be used. The #2; following the OUTPUT statement tells Driver488/DRV to send only the first two bytes following the semicolon. If just the OUTPUT statement had been used, terminator characters would have been sent along with the #0 command. This would result in incorrect data being sent to the DAC488HR.

```
PRINT #1, "OUTPUT 10 #2;#0"
```

Now we have to tell Driver488/DRV where the data can be found and how much data we are going to move. The Driver488/DRV command format is explained as follows:

OUTPUT	Driver488/DRV command to output data
10	The device data will be sent to.
#16384	Number of bytes to be sent to the selected device
BUFFER	Data will be sent directly from memory starting at the segment and offset addresses
Segment: Offset	The addresses of the segment and offset
DMA	Use Direct Memory Access. DMA must be used for this example. If DMA is not used, discontinuities may appear in the output waveform due to data starvation.
CHR$(13) + CHR$(10)	This is necessary because of the PUT command which is being used to send the data to Driver488/DRV. When PRINT is used, terminators are automatically appended to the end of the command string. However, when PUT is used, no terminators are appended. Without the CHR$(13) and CHR$(10), Driver488/DRV would not know when the command had been completed.

The command sequence is:

```
seg1 = HEX$(VARSEG(outbuffer(1)))
off1 = HEX$(VARPTR(outbuffer(1)))
PRINT "Sending waveform data to DAC488HR."
PRINT "Press any key to halt..."
```

```
out1 = "OUTPUT10#16384 BUFFER &H" + seg1 + ":&H" + off1 + "DMA" + _
        CHR$(13) + CHR$(10)

WHILE INKEY$ = ""
   PUT #3, , out1          'Use faster "PUT" to minimize DMA latency
                           'between bursts.
```

As soon as the user presses a key, waveform output can then be halted by the following command:

```
PRINT #1, "SEND EOI 10"          'Terminate transfer with LF EOI
                                 'combination

PRINT #1, "OUTPUT 10;P1X C0X" 'Cease waveform output on port 1

PRINT #1, "OUTPUT 10;P2X C0X" 'Cease waveform output on port 2

END
```

Note when using either one channel or two channel continuous transfer modes (C5, C6 or C7), after terminating the transfer with a LF EOI it is necessary to wait for the unassertion of the trigger bit (1) in the Serial Poll register.

When the DAC488HR receives a LF EOI in continuous mode, the output is not discontinued immediately. The port continues to output the data remaining in the buffer. During this time, the DAC488HR will not respond to commands.

After the last sample in each of the daughterboards is output, the trigger bit becomes unasserted. At this point the DAC488HR can receive more commands.

In order to finish termination of the transfer, send the following string to the DAC488HR immediately after the unassertion of the trigger bit:

"P1c0X" where port 1 is used.

"P1c0XP2c0X" where ports 1 and 2 are used.

4.6 Synchronizing Multiple DAC488HR Units

This section describes how to connect and synchronize the outputs of multiple DAC488HR units. Without the use of multiple DAC488HRs, the maximum number of channels that could accept and output synchronized arbitrary data from the IEEE bus is limited to two. To accomplish this synchronization, the DAC488HR provides an additional trigger mode command (T7). This command sets the update source to external with no divisor and sets the trigger source to external. In addition, all trigger and update clock synchronization is inhibited in order to prevent trigger latencies from occurring. Since trigger mode is completely independent of update mode, the T7 command can be used to synchronize multiple DAC488HRs for any of the update modes.

To begin, the DAC488HR units must be cabled together. The interface uses a simple three-wire cable, like IOtech's CA-112, to connect the first DAC488HR unit (master) to the second

DAC488HR unit (slave), etc. The pinouts used in the connector are detailed in the figure below.

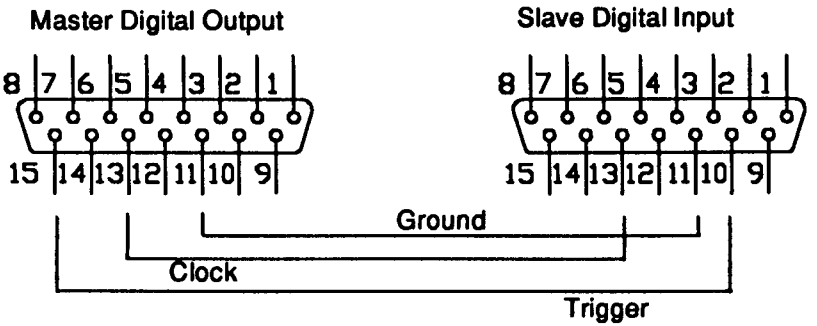


Figure 4.1: Master/Slave Connector Pinouts

The cables can then be daisy-chained (see Figure 4.2) between multiple DAC488HR units. Note that with this type of cabling, the propagation delays from each DAC488HR in the chain is additive. The delay between each unit is approximately 60nS.

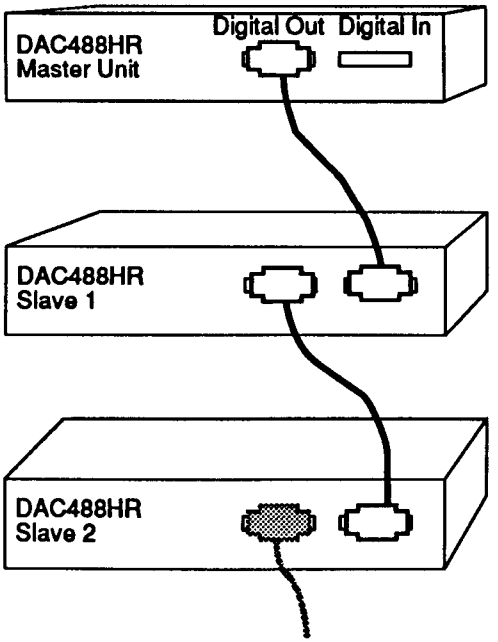


Figure 4.2: DAC488HR Master/Slave Cable Configuration

Once the first DAC488HR unit (master) and additional (slave) units are correctly cabled together, the user can output synchronous waveforms. The steps necessary to perform multiple DAC488HR synchronization for arbitrary waveforms vary depending on whether the update mode is synchronous or continuous. A listing for each is described below.

4.6.1 For Synchronous Update Modes (C1-C4)

For an application which requires synchronizing 8 analog output ports using data contained in each analog buffer, the following steps are required.

1. Connect two DAC488HRs with the master/slave cable.
2. Select ranges for all ports.
3. Select desired update source and divisor on master only.
4. Send data to all analog port buffers.
5. Program sequences (if using complex output mode in master or slave).
6. Select C3 mode for all analog ports (if using continuous update mode).
7. Select trigger mode 7 on slave.
8. Select desired trigger source on master.
9. Generate trigger.

The actual commands can be modified to accommodate other synchronous update modes or additional DAC488HR units. Refer to the example program below. This example will synchronize output on four DAC488HR output ports, two on one unit and two on another. Both units must be connected with a CA-112, or equivalent. The master unit is assumed to be at IEEE address 10, and the slave unit is assumed to be at IEEE address 11.

Open DRIVER488 for communications.

```
OPEN "\DEV\IEEEOUT" FOR OUTPUT AS #1
IOCTL #1, "BREAK"
PRINT #1, "RESET"
OPEN "\DEV\IEEEIN" FOR INPUT AS #2
PRINT #1, "FILL ERROR"
```

Reset both units.

```
PRINT "Resetting DAC488HR..."
PRINT #1, "OUTPUT 10;*RX"
PRINT #1, "OUTPUT 11;*RX"
```

Wait for both units to become ready.

```
SLEEP 2
```

```
spoll = 0
WHILE NOT spoll AND 4
 PRINT #1, "SPOLL 10"
 INPUT #2, spoll1
 PRINT #1, "SPOLL 11"
 INPUT #2, spoll2
 spoll = spoll1 AND spoll2
WEND

PRINT "Programming DAC488HR's for synchronous output..."

PRINT #1, "OUTPUT 10;P1X R4X" 'Select master port 1, range 4
PRINT #1, "OUTPUT 10;P2X R4X" 'Select master port 2, range 4
PRINT #1, "OUTPUT 11;P1X R4X" 'Select slave port 1, range 4
PRINT #1, "OUTPUT 11;P2X R4X" 'Select slave port 2, range 4
```

Fill the channel buffers with data.

```
PRINT #1, "OUTPUT 10;P1X"  'Select master port 1.
PRINT #1, "OUTPUT 10;B-10,-9,-8,-7,-6,-5,-4,-3,-2,-1 X"
PRINT #1, "OUTPUT 10;B 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 X"
PRINT #1, "OUTPUT 10;B-10,-9,-8,-7,-6,-5,-4,-3,-2,-1 X"
PRINT #1, "OUTPUT 10;B 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 X"

PRINT #1, "OUTPUT 10;P2X"  'Select master port 2.
PRINT #1, "OUTPUT 10;B-10,-9,-8,-7,-6,-5,-4,-3,-2,-1 X"
PRINT #1, "OUTPUT 10;B 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 X"
PRINT #1, "OUTPUT 10;B-10,-9,-8,-7,-6,-5,-4,-3,-2,-1 X"
PRINT #1, "OUTPUT 10;B 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 X"

PRINT #1, "OUTPUT 11;P1X"  'Select slave port 1.
PRINT #1, "OUTPUT 11;B-10,-9,-8,-7,-6,-5,-4,-3,-2,-1 X"
PRINT #1, "OUTPUT 11;B 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 X"
PRINT #1, "OUTPUT 11;B-10,-9,-8,-7,-6,-5,-4,-3,-2,-1 X"
PRINT #1, "OUTPUT 11;B 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 X"

PRINT #1, "OUTPUT 11;P2X"  'Select slave port 2.
PRINT #1, "OUTPUT 11;B-10,-9,-8,-7,-6,-5,-4,-3,-2,-1 X"
```

```
PRINT #1, "OUTPUT 11;B 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 X"
PRINT #1, "OUTPUT 11;B-10,-9,-8,-7,-6,-5,-4,-3,-2,-1 X"
PRINT #1, "OUTPUT 11;B 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 X"
```

Program update source, and divisor on master. Since the slave will be accepting trigger, and update information from the master, there is no need to program the slave's update source.

```
PRINT #1, "OUTPUT 10;G0 I50X" 'Select update source 5M, divisor 50
(100K)
```

Program update modes on both units.

```
PRINT #1, "OUTPUT 10;P1XC3X" 'Select port 1 on master, waveform up-
date mode.
PRINT #1, "OUTPUT 10;P2XC3X" 'Select port 2 on master, waveform up-
date mode.
PRINT #1, "OUTPUT 11;P1XC3X" 'Select port 1 on slave, waveform up-
date mode.
PRINT #1, "OUTPUT 11;P2XC3X" 'Select port 2 on slave, waveform up-
date mode.
```

Now program trigger source.

```
PRINT #1, "OUTPUT 10;T5X"    'Select trigger on @ for master.
PRINT #1, "OUTPUT 11;T7X"    'Select slave trigger on slave.
```

Wait for slave to become ready. This prevents the master from triggering before the slave is ready.

```
spoll% = 0
WHILE NOT spoll% AND 4
  PRINT #1, "SPOLL 11"
  INPUT #2, spoll%
WEND
```

Now trigger the master to begin outputting data on both master and both slave channels synchronously.

```
PRINT #1, "OUTPUT 10;@x"

PRINT "Both DAC488HR's are now outputting. Press any key to quit."

WHILE INKEY$ = ""
```

```
WEND
```

Disable update modes on both units.

```
PRINT #1, "OUTPUT 10;P1XC0X" 'Select port 1 on master, disabled update mode.
PRINT #1, "OUTPUT 10;P2XC0X" 'Select port 2 on master, disabled update mode.
PRINT #1, "OUTPUT 11;P1XC0X" 'Select port 1 on slave, disabled update mode.
PRINT #1, "OUTPUT 11;P2XC0X" 'Select port 2 on slave, disabled update mode.

END
```

4.6.2 For Continuous Update Modes (C5-C7)

For four channel (two per DAC488HR) arbitrary waveform generation, perform the following steps.

1. Set all ports and ranges for master and slave units.
2. Select update source and divisor on master.
3. Select update modes C6 and C7 on the selected ports for both units.
4. Select trigger source 7 on slave.
5. Start sending data blocks to all units—slaves first. Block size should be 4096 bytes per unit, per analog port.

Note there is no need to select a trigger source on the master because a trigger is automatically generated when the required amount of data has been transferred to the master. For each analog port involved in the continuous transfer, 4096 bytes of data must be transferred before output will begin. For example, when 4096 bytes are transferred to the master's analog port, it triggers out to slave 1 which triggers out to slave 2, etc. Again the commands can be modified to accommodate other continuous update modes or additional DAC488HR units.

To demonstrate these instructions, the following example program sends continuous binary data in interleaved fashion to four channels on two separate DAC488HR units. In this program (EXAMPLE4.BAS on the disk), the example waveforms are a ramp up of 64 points and a ramp down of 64 points for both DAC488HR units. All four waveforms will be output simultaneously assuming that the hardware connection is correct.

First DIMension buffer array to be used.

```
DIM buf%(1 TO 128)
```

Now open Driver488/DRV for communications.

```
OPEN "\DEV\IEEEOUT" FOR OUTPUT AS #1
IOCTL #1, "BREAK"
PRINT #1, "RESET"
OPEN "\DEV\IEEEIN" FOR INPUT AS #2
PRINT #1, "TERM IN LFEOI"
```

Next reset both DAC488HRs.

```
PRINT #1, "OUTPUT 10;*RX"          ' Reset Master
PRINT #1, "OUTPUT 11;*RX"          ' Reset Slave
PRINT "Resetting DAC488HR..."
SLEEP 1
```

Wait for the ready bit.

```
spoll = 0
WHILE NOT spoll AND 4
   PRINT #1, "SPOLL 10"
   INPUT #2, spoll1
   PRINT #1, "SPOLL 11"
   INPUT #2, spoll2
   spoll = spoll1 AND spoll2
WEND
```

Define the master clock of the DAC488HR at address 10 and slave mode for the DAC488HR at address 11.

```
PRINT #1, "OUTPUT 10;G3I40X"       ' Set Master Clock and divisor
PRINT #1, "OUTPUT 11;T7X"          ' Set Slave mode
```

Now we need to calculate waveforms to be used. For both DAC488HRs the example waveforms are a ramp up of 64 points and a ramp down of 64 points.

```
FOR i = 1 TO 64
   buf%((i * 2) - 1) = i * 32767! / 64     ' positive ramp
   buf%(i * 2) = (64 - i) * 32767! / 64    ' negative ramp
NEXT i
```

Once the waveform has been calculated, all ports must be set to continuous interleaved mode.

```
PRINT "Setting up the DAC488HRs for reception of continuous data..."
PRINT #1, "OUTPUT 10;P1R4X";       'Set port 1 on master to 10V
```

```
                                    'bipolar range.
PRINT #1, "OUTPUT 10;P2R4X";        'Set port 2 on master to 10V
                                    'bipolar range.
PRINT #1, "OUTPUT 11;P1R4X";        'Set port 1 on slave to 10V
                                    'bipolar range.
PRINT #1, "OUTPUT 11;P2R4X";        'Set port 2 on slave to 10V
                                    'bipolar range.
PRINT #1, "OUTPUT 10;P1C6X";        'Set port 1 on master to 10V
                                    'continuous, odd sample mode
PRINT #1, "OUTPUT 10;P2C7X";        'Set port 2 on master to 10V
                                    'continuous, even sample mode
PRINT #1, "OUTPUT 11;P1C6X";        'Set port 1 on slave to 10V
                                    'continuous, odd sample mode.
PRINT #1, "OUTPUT 11;P2C7X";        'Set port 2 on slave to 10V
                                    'continuous, even sample mode.
```

Finally, we need to tell the DAC488HR to expect continuous waveform data. As previously stated, the #0 command tells the DAC488HR that an undetermined amount of binary data is going to be sent. In order to send this command properly, a Driver488/DRV specific command will be used. The #2; following the OUTPUT statement tells Driver488/DRV to send only the first two characters following the semicolon. If just the OUTPUT statement had been used, terminator characters would have been sent along with the #0 command. This would result in incorrect data being sent to the DAC488HR.

```
PRINT #1, "OUTPUT 10#2;#0"          'Send 2 characters, '#' & '0',
                                    'to the DAC488HR at address 10
PRINT #1, "OUTPUT 11#2;#0"          'Send 2 characters, '#' & '0',
                                    'to the DAC488HR at address 11
```

Now we have to send binary data continuously to both DAC488HRs. In order for the trigger to function correctly, data will have to be sent to the slave first. Once the Master's port receives 4096 bytes of data, it triggers itself and the slaves and synchronizes all outputs.

The following lines construct strings which will be used to instruct Driver488/DRV. The driver uses Direct Memory Access to transfer the data directly from PC memory to the DAC488HR, using the Buffer command. The Buffer command has the following format:

```
OUTPUT address # number of bytes to transfer segment of data
buffer: offset of data buffer DMA
```

For this example program, the strings are as follows:

```
cmd1$ = "OUTPUT 10 #256 BUFFER" + STR$(VARSEG(buf%(1))) + ":" + _
  STR$(VARPTR(buf%(1))) + "DMA"
cmd2$ = "OUTPUT 11 #256 BUFFER" + STR$(VARSEG(buf%(1))) + ":" + _
  STR$(VARPTR(buf%(1))) + "DMA"
PRINT "Transmitting waveforms continuously. Press any key to
halt..."
WHILE INKEY$ = ""
   PRINT #1, cmd2$     ' Slave first
   PRINT #1, cmd1$     ' Master last
WEND
```

To terminate the transfer of both DAC488HRs, first send LF with EOI to both DAC488HRs:

```
PRINT #1, "SEND UNT UNL MTA LISTEN10 EOI 10"
PRINT #1, "SEND UNT UNL MTA LISTEN11 EOI 10"
```

Next shut down all four ports.

```
PRINT #1, "OUTPUT11;P1C0XP2C0X"
PRINT #1, "OUTPUT10;P1C0XP2C0X"
```

Now read the error code on the Master unit. This clears the DMA underrun error that will occur when the data transfer has been terminated.

```
PRINT #1, "OUTPUT10;E?X"
PRINT #1, "ENTER10"
INPUT #2, A$
PRINT "Master ERROR is :"; A$
```

Lastly, read the error code on the Slave unit.

```
PRINT #1, "OUTPUT11;E?X"
PRINT #1, "ENTER11"
INPUT #2, A$
PRINT "Slave ERROR is :"; A$
```

4.7 Sample DAC488HR Programs

Each of the example programs discussed in this chapter can be found on the DAC488HR diskette as well as on the following pages.

4.7.1 Binary Data Load Example

```
'EXAMPLE1.BAS
'DAC488HR binary data load example program
'REV 1.0
'
'This program demonstrates how to load a portion of a channel data
'buffer using binary transfers. The example waveform will be a sine
'wave of 256 points.
'

CLS

'First dimension all arrays and variables to be used

DIM SHARED wavebuffer(256) AS INTEGER
DIM SHARED seg1 AS STRING * 4
DIM SHARED off1 AS STRING * 4
DIM SHARED out1 AS STRING * 40

'Open DRIVER488/DRV for communications

OPEN "\DEV\IEEEOUT" FOR OUTPUT AS #1

IOCTL #1, "BREAK"
PRINT #1, "RESET"

OPEN "\DEV\IEEEIN" FOR INPUT AS #2
PRINT #1, "FILL ERROR"

PRINT "Resetting DAC488HR..."
PRINT #1, "OUTPUT 10;*RX"
SLEEP 1 'Wait for one second
spoll = 0

WHILE NOT spoll AND 4
   PRINT #1, "SPOLL 10"
   INPUT #2, spoll
WEND
```

```
PRINT #1, "OUTPUT 10;G0I50X" 'Set update source 5MHz,divisor 50
                             ' (100K update)
PRINT #1, "OUTPUT 10;P1A0X"  'Select port 1, linear buffer mode
PRINT #1, "OUTPUT 10;R4X"    'Select range 4
PRINT #1, "OUTPUT 10;K0X"    'Select infinite repeat count

'Now calculate waveform to be used. Length of waveform is 256
'samples.
'Channel 1 will generate a sine wave

PRINT "Calculating waveform..."

angle! = (6.28318 / 256)
FOR samples% = 1 TO 256
   points! = SIN(angle! * samples%)
   points! = points! * 32767
   wavebuffer(samples%) = INT(points!)
NEXT samples%

PRINT "Sending data to DAC488HR..."
'
'The next statement requires a little explanation. First, the B
'tells the DAC that buffer data is to follow. The 3 following the
'pound sign tells the DAC that there are three digits which describe
'how many BYTES of data will be transferred. The 512 tells the DAC how
'many BYTES are to be transferred.

PRINT #1, "OUTPUT 10 #6;B#3512"
'Define the segment and offset for the DMA output data buffer.
seg1 = HEX$(VARSEG(wavebuffer(1)))
off1 = HEX$(VARPTR(wavebuffer(1)))

PRINT "Sending waveform data to DAC488HR..."

out1 = "OUTPUT 10 #512 BUFFER &H" + seg1 + ":&H" + off1 + "DMA"

PRINT #1, out1

PRINT #1, "SEND EOI 88"   'Terminate transfer with a LF EOI
                          ' combination.

PRINT #1, "OUTPUT 10;C4X"'Set trigger mode immediate
```

```
PRINT "DAC is now generating waveform."
PRINT "Press any key to stop..."

WHILE INKEY$ = ""
WEND

PRINT #1, "OUTPUT 10;COX" 'Cease waveform output.

END
```

4.7.2 Continuous Data Transfer Example

```
'EXAMPLE2.BAS
'DAC488HR continuous data transfer example program
'REV 1.0
'
'This program demonstrates how to send continuous binary data to one
'channel of the DAC488HR. The example waveform is a sine wave of 256
'points. The waveform will be updated at 50KSamples/Second
'
'


CLS

'First dimension all arrays and variables to be used

DIM SHARED outbuffer(4096) AS INTEGER
DIM SHARED wavebuffer(256) AS INTEGER
DIM SHARED seg1 AS STRING * 4
DIM SHARED off1 AS STRING * 4
DIM SHARED out1 AS STRING * 40

'Open DRIVER488/DRV for communications

OPEN "\DEV\IEEEOUT" FOR OUTPUT AS #1
OPEN "\DEV\IEEEOUT" FOR BINARY AS #3

IOCTL #1, "BREAK"
PRINT #1, "RESET"
```

```
OPEN "\DEV\IEEEIN" FOR INPUT AS #2
PRINT #1, "FILL ERROR"

'Now program the DAC for reception of data
PRINT "Resetting DAC488HR..."
PRINT #1, "OUTPUT 10;*RX"
SLEEP 1 'Wait once second
spoll = 0

WHILE NOT spoll AND 4
   PRINT #1, "SPOLL 10"
   INPUT #2, spoll
WEND

PRINT "Programming DAC488HR continuous mode..."

PRINT #1, "OUTPUT 10;G0I100X" 'Select update source 5M, divisor 100
                              '(50K update rate)
PRINT #1, "OUTPUT 10;P1A0X"   'Select port 1, linear buffer mode.
PRINT #1, "OUTPUT 10;R4X"     'Select range 4
PRINT #1, "OUTPUT 10;C5X"     'Select continuous update mode


'Now calculate waveform to be used.

PRINT "Calculating waveform..."

angle! = (6.28318 / 256)
FOR samples% = 1 TO 256
   points! = SIN(angle! * samples%)
   points! = points! * 32767
   wavebuffer(samples%) = INT(points!)
NEXT samples%

'Now that the waveform has been calculated, it must be placed into
'the output buffer.

PRINT "Placing waveform in output buffer..."
```

```
FOR offset% = 0 TO 3840 STEP 256
   FOR samples% = 1 TO 256
         outbuffer(samples% + offset%) = wavebuffer(samples%)
   NEXT samples%
NEXT offset%


PRINT "Sending data to DAC488HR..."
PRINT #1, "OUTPUT 10 #2;#0"

'Define the segment and offset for the DMA output buffer
seg1 = HEX$(VARSEG(outbuffer(1)))
off1 = HEX$(VARPTR(outbuffer(1)))



PRINT "Sending waveform data to DAC488HR."
PRINT "Press any key to halt..."

out1 = "OUTPUT10#8192 BUFFER &H" + seg1 + ":&H" + off1 + "DMA" +_
       CHR$(13) + CHR$(10)

WHILE INKEY$ = ""
   PUT #3, , out1 'Use faster "PUT" command for less latency between
bursts
WEND

PRINT #1, "SEND EOI 10"        'Terminate transfer with a LF EOI combination
PRINT #1, "OUTPUT 10;P1XC0X" 'Cease waveform output

END
```

4.7.3 Interleaved Data Transfer Example

```
'EXAMPLE3.BAS
'DAC488HR Interleaved data transfer example program
'REV 1.0
'
'This example program demonstrates how to send continuous binary
'data in interleaved fashion to 2 channels on the DAC488HR. The
'example waveforms are a sine wave of 256 points, and a TRIANGLE wave
'of 256 points. Both waveforms will be output simultaneously at an
'update rate of 25KSamples/Channel/Second
'
'
```

```
CLS

'First dimension all arrays and variables to be used

DIM SHARED outbuffer(8192) AS INTEGER
DIM SHARED wavebuffer(256) AS INTEGER
DIM SHARED seg1 AS STRING * 4
DIM SHARED off1 AS STRING * 4
DIM SHARED out1 AS STRING * 40

'Open DRIVER488/DRV for communications

OPEN "\DEV\IEEEOUT" FOR OUTPUT AS #1
OPEN "\DEV\IEEEOUT" FOR BINARY AS #3

IOCTL #1, "BREAK"
PRINT #1, "RESET"

OPEN "\DEV\IEEEIN" FOR INPUT AS #2
PRINT #1, "FILL ERROR"

'Now program the DAC for reception of data
PRINT "Resetting DAC488HR..."
PRINT #1, "OUTPUT 10;*RX"
SLEEP 1 'Wait for one second
spoll = 0

WHILE NOT spoll AND 4
   PRINT #1, "SPOLL 10"
   INPUT #2, spoll
WEND

PRINT "Programming DAC488HR for interleave mode..."

PRINT #1, "OUTPUT 10;G0 I200X"      'Select update source 5M, divisor 200
                                    '(25K)
PRINT #1, "OUTPUT 10;P1 A0X"        'Select port 1, linear buffer mode
PRINT #1, "OUTPUT 10;P2 A0X"        'Select port 2, linear buffer mode
PRINT #1, "OUTPUT 10;P1 R4X"        'Select port 1, range 4
PRINT #1, "OUTPUT 10;P2 R4X"        'Select port 2, range 4
PRINT #1, "OUTPUT 10;P1 C6X"        'Select port 1, interleave, take odd
```

```
                                        'samples
PRINT #1, "OUTPUT 10;P2 C7X"  'Select port 2,interleave, take even
                                        'samples


'Now calculate waveforms to be used. Length of both waveforms is 256
'samples.
'Channel 1 will be generate a sine wave. Channel 2 will generate a
'triangle wave. Both waveforms will be output at 25 KHz.
'First calculate the sine wave.

PRINT "Calculating waveform 1..."

angle! = (6.28318 / 256)
FOR samples% = 1 TO 256
   points! = SIN(angle! * samples%)
   points! = points! * 32767
   wavebuffer(samples%) = INT(points!)
NEXT samples%

'Now that the waveform has been calculated, it must be placed into
'the output buffer.

PRINT "Placing waveform 1 in output buffer..."

offset% = 1
FOR samples% = 1 TO 511 STEP 2
   outbuffer(samples%) = wavebuffer(offset%)
   offset% = offset% + 1
NEXT samples%

'Now calculate the triangle wave

PRINT "Calculating waveform 2..."

baseline! = 0
stepsize! = 32768 / 64

FOR samples% = 1 TO 64
   wavebuffer(samples%) = INT(baseline!)
   baseline! = baseline! + stepsize!
NEXT samples%
```

```
baseline! = 32767
stepsize! = 65536 / 128

FOR samples% = 65 TO 192
   wavebuffer(samples%) = INT(baseline!)
   baseline! = baseline! - stepsize!
NEXT samples%

baseline! = -32768
stepsize! = 32768 / 64

FOR samples% = 193 TO 256
   wavebuffer(samples%) = INT(baseline!)
   baseline! = baseline! + stepsize!
NEXT samples%

'Now that the waveform has been calculated, it must be placed into
'the output buffer.

PRINT "Placing waveform 2 in output buffer..."

offset% = 1
FOR samples% = 2 TO 512 STEP 2
   outbuffer(samples%) = wavebuffer(offset%)
   offset% = offset% + 1
NEXT samples%

PRINT "Copying data to remainder of output buffer..."

FOR offset% = 512 TO 7680 STEP 512
   FOR samples% = 1 TO 512
         outbuffer(samples% + offset%) = outbuffer(samples%)
   NEXT samples%
NEXT offset%

PRINT "Sending data to DAC488HR..."
PRINT #1, "OUTPUT10#2;#0" 'Tell DAC to expect arbitrary length data

'Define the segment and offset for the DMA output buffer.
seg1 = HEX$(VARSEG(outbuffer(1)))
off1 = HEX$(VARPTR(outbuffer(1)))
```

```
PRINT "Sending waveform data to DAC488HR."
PRINT "Press any key to halt..."

out1 = "OUTPUT10#16384 BUFFER &H" + seg1 + ":&H" + off1 + "DMA" +_
       CHR$(13) + CHR$(10)

WHILE INKEY$ = ""
   PUT #3, , out1 'Use faster "PUT" to minimize DMA latency between
bursts
WEND

PRINT #1, "SEND EOI 10"          'Terminate transfer with LF EOI
                                 'combination
PRINT #1, "OUTPUT 10;P1 C0X" 'Cease waveform output on port 1
PRINT #1, "OUTPUT 10;P2 C0X" 'Cease waveform output on port 2

END
```

4.7.4 Multiple Units (Continuous Update Mode) Example

```
'EXAMPLES4.BAS
'DAC488HR Mater and Slave example program
'REV 1.0
'
'This example program demonstrates how to send continuous binary
'data in interleaved fashion to 4 channels on two separate DAC488HRs.
'The example waveforms are a ramp up of 64 points and a ramp down of
'64 points for both DAC488HRs. All four waveforms will be output
'simultaneously assuming that the hardware connection between both
'DACs is correctly done. This program is easily expandable to three
'or more DAC488HR
'
'


CLS
```

```
'Define array

DIM buf%(1 TO 128)


' Establish communications with DRIVER488

OPEN "\DEV\IEEEOUT" FOR OUTPUT AS #1
IOCTL #1, "BREAK"
PRINT #1, "RESET"
OPEN "\DEV\IEEEIN" FOR INPUT AS #2
PRINT #1, "TERM IN LFEOI"

'Reset both DAC488HRs

PRINT #1, "OUTPUT 10;*RX"    ' Reset Master
PRINT #1, "OUTPUT 11;*RX"    ' Reset Slave
PRINT "Resetting DAC488HR..."
SLEEP 1

spoll = 0
WHILE NOT spoll AND 4
   PRINT #1, "SPOLL 10"
   INPUT #2, spoll1
   PRINT #1, "SPOLL 11"
   INPUT #2, spoll2
   spoll = spoll1 AND spoll2
WEND


'Assign Master clock and slave mode according to hardware connec-
tions

PRINT #1, "OUTPUT 10;G3I40X"    ' Set Master Clock and divisor
PRINT #1, "OUTPUT 11;T7X"       ' Set Slave mode

'Calculate waveforms
```

```
FOR i = 1 TO 64
   buf%((i * 2) - 1) = i * 32767! / 64      ' positive ramp
   buf%(i * 2) = (64 - i) * 32767! / 64     ' negative ramp
NEXT i


'Set all ports to continuous interleaved mode

PRINT "Setting up the DAC488HRs for reception of continuous data..."
PRINT #1, "OUTPUT 10;P1R4X";          'Set port 1 on master to 10V
                                      'bipolar range.
PRINT #1, "OUTPUT 10;P2R4X";          'Set port 2 on master to 10V
                                      'bipolar range.
PRINT #1, "OUTPUT 11;P1R4X";          'Set port 1 on slave to 10V
                                      'bipolar range.
PRINT #1, "OUTPUT 11;P2R4X";          'Set port 2 on slave to 10V
                                      'bipolar range.
PRINT #1, "OUTPUT 10;P1C6X";          'Set port 1 on master to 10V
                                      'continuous, odd sample mode
PRINT #1, "OUTPUT 10;P2C7X";          'Set port 2 on master to 10V
                                      'continuous, even sample mode
PRINT #1, "OUTPUT 11;P1C6X";          'Set port 1 on slave to 10V
                                      'continuous, odd sample mode.
PRINT #1, "OUTPUT 11;P2C7X";          'Set port 2 on slave to 10V
                                      'continuous, even sample mode.


'Invoke Continuous transfer

PRINT #1, "OUTPUT 10#2;#0"
PRINT #1, "OUTPUT 11#2;#0"


'Send Binary data continuously to both DAC488HRs

cmd1$ = "OUTPUT 10 #256 BUFFER" + STR$(VARSEG(buf%(1))) + ":" +_
 STR$(VARPTR(buf%(1))) + "DMA"
cmd2$ = "OUTPUT 11 #256 BUFFER" + STR$(VARSEG(buf%(1))) + ":" +_
```

```
 STR$(VARPTR(buf%(1))) + "DMA"
PRINT "Transmitting waveforms continuously. Press any key to
halt..."
WHILE INKEY$ = ""
PRINT #1, cmd2$     ' Slave first
PRINT #1, cmd1$     ' Master last
WEND


'Terminate the transfer on both DAC488HRs

PRINT #1, "SEND UNT UNL MTA LISTEN10 EOI 10"
PRINT #1, "SEND UNT UNL MTA LISTEN11 EOI 10"

PRINT #1, "OUTPUT11;P1C0XP2C0X"
PRINT #1, "OUTPUT10;P1C0XP2C0X"

PRINT #1, "OUTPUT10;E?X"
PRINT #1, "ENTER10"
INPUT #2, A$
PRINT "Master ERROR is :"; A$

PRINT #1, "OUTPUT11;E?X"
PRINT #1, "ENTER11"
INPUT #2, A$
PRINT "Slave ERROR is :"; A$
```

4.7.5 Multiple Units (Synchronous Update Mode) Example

```
'EXAMPLE5.BAS
'DAC488HR Multiple DAC488HR synchronization example.
'REV 1.0
'
'This example program demonstrates how to synchronize multiple
'DAC488HR outputs. This example will synchronize output on four
'DAC488HR output ports, two on one unit and two on another. Both
'units must be connected with a CA-112, or equivalent. The 'Master'
'unit is assumed to be at IEEE address 10, and the 'Slave' unit is
'assumed to be at IEEE address 11.
```

```
CLS

'Open DRIVER488 for communications

OPEN "\DEV\IEEEOUT" FOR OUTPUT AS #1
IOCTL #1, "BREAK"
PRINT #1, "RESET"
OPEN "\DEV\IEEEIN" FOR INPUT AS #2
PRINT #1, "FILL ERROR"

'Reset both units

PRINT "Resetting DAC488HR..."
PRINT #1, "OUTPUT 10;*RX"
PRINT #1, "OUTPUT 11;*RX"

'Wait for both units to become ready

SLEEP 2
spoll = 0
WHILE NOT spoll AND 4
  PRINT #1, "SPOLL 10"
  INPUT #2, spoll1
  PRINT #1, "SPOLL 11"
  INPUT #2, spoll2
  spoll = spoll1 AND spoll2
WEND

PRINT "Programming DAC488HR's for synchronous output..."

PRINT #1, "OUTPUT 10;P1X R4X" 'Select master port 1, range 4
PRINT #1, "OUTPUT 10;P2X R4X" 'Select master port 2, range 4
PRINT #1, "OUTPUT 11;P1X R4X" 'Select slave port 1, range 4
PRINT #1, "OUTPUT 11;P2X R4X" 'Select slave port 2, range 4

'Now program output waveforms for ramp waves.

PRINT #1, "OUTPUT 10;P1X"   'Select master port 1.
```

```
PRINT #1, "OUTPUT 10;B-10,-9,-8,-7,-6,-5,-4,-3,-2,-1X"
PRINT #1, "OUTPUT 10;B 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10X"
PRINT #1, "OUTPUT 10;B-10,-9,-8,-7,-6,-5,-4,-3,-2,-1X"
PRINT #1, "OUTPUT 10;B 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10X"

PRINT #1, "OUTPUT 10;P2X"  'Select master port 2.
PRINT #1, "OUTPUT 10;B-10,-9,-8,-7,-6,-5,-4,-3,-2,-1X"
PRINT #1, "OUTPUT 10;B 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10X"
PRINT #1, "OUTPUT 10;B-10,-9,-8,-7,-6,-5,-4,-3,-2,-1X"
PRINT #1, "OUTPUT 10;B 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10X"

PRINT #1, "OUTPUT 11;P1X"  'Select slave port 1.
PRINT #1, "OUTPUT 11;B-10,-9,-8,-7,-6,-5,-4,-3,-2,-1X"
PRINT #1, "OUTPUT 11;B 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10X"
PRINT #1, "OUTPUT 11;B-10,-9,-8,-7,-6,-5,-4,-3,-2,-1X"
PRINT #1, "OUTPUT 11;B 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10X"

PRINT #1, "OUTPUT 11;P2X"  'Select slave port 2.
PRINT #1, "OUTPUT 11;B-10,-9,-8,-7,-6,-5,-4,-3,-2,-1X"
PRINT #1, "OUTPUT 11;B 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10X"
PRINT #1, "OUTPUT 11;B-10,-9,-8,-7,-6,-5,-4,-3,-2,-1X"
PRINT #1, "OUTPUT 11;B 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10X"

'Program update source, and divisor on master. Since the slave will
'be accepting trigger, and update information from the master, there
'is no need to program the slave's update source.

PRINT #1, "OUTPUT 10;G0 I50X" 'Select update source 5M, divisor 50
(100K)

'Program update modes on both units.

PRINT #1, "OUTPUT 10;P1XC3X" 'Select port 1 on master, waveform up-
date mode.
PRINT #1, "OUTPUT 10;P2XC3X" 'Select port 2 on master, waveform up-
date mode.
PRINT #1, "OUTPUT 11;P1XC3X" 'Select port 1 on slave, waveform up-
date mode.
```

```
PRINT #1, "OUTPUT 11;P2XC3X" 'Select port 2 on slave, waveform update mode.

'Now program trigger source.

PRINT #1, "OUTPUT 10;T5X"    'Select trigger on @ for master.
PRINT #1, "OUTPUT 11;T7X"    'Select slave trigger on slave.

'Wait for slave to become ready. This prevents the master from triggering
'before the slave is ready.

spoll% = 0
WHILE NOT spoll% AND 4
  PRINT #1, "SPOLL 11"
  INPUT #2, spoll%
WEND

'Now trigger the master, and begin outputting data.

PRINT #1, "OUTPUT 10;@x"

PRINT "Both DAC488HR's are now outputting. Press any key to quit."

WHILE INKEY$ = ""
WEND

'Disable update modes on both units.

PRINT #1, "OUTPUT 10;P1XC0X" 'Select port 1 on master, disabled update mode.
PRINT #1, "OUTPUT 10;P2XC0X" 'Select port 2 on master, disabled update mode.
PRINT #1, "OUTPUT 11;P1XC0X" 'Select port 1 on slave, disabled update mode.
PRINT #1, "OUTPUT 11;P2XC0X" 'Select port 2 on slave, disabled update mode.

END
```

Command Descriptions

5.1 Overview

The DAC488HR uses a register-based command set. Just as most peripheral chips have one or more registers that may be set to different values to control their operation, each command letter in this command set corresponds to an internal register. The DAC488HR is controlled by modifying the contents of its internal registers through the register-based bus commands. The relationship between the contents of the registers and the actions taken by DAC488HR are described in the command descriptions that follow in this section. In general, each of these registers holds a single numeric value that is maintained at the last value set.

There are two types of register-based commands. System commands configure features that act on all analog output ports simultaneously. Port commands configure parameters specific to each analog output port and are active for each analog output port separately.

The System commands are:

Reset (*R)
Command Trigger (@)
Digital Output (Dn)
Error Query (E?)
Format (Fn)
Update Source, Mode (Gn)
Update Divider (In)
SRQ Mask (Mn)
Event Mask (Nn)
Port Select (Pn)
Save/Restore (Sn)
Trigger Source (Tn)
Status (Un)
Execute (X)
Interval Timer (Yn)
Trigger Delay (Zn)

The analog output port is selected using the Port Select (Pn) command. The Port commands are:

Buffer Mode (An)
Buffer Data (Bval)
Trigger Control Mode (Cn)
Offset Calibration (Hn)
Gain Calibration (Jn)
Buffer Count (Kn)
Data Buffer Location Pointer (Ln)
Sequence Pointer (On)
Define Sequence Control Block (Ql,n,r)
Range (Rn)
Voltage Output (Vval)
Waveform Load (Ww,l,max,min,d)

Most commands consist of one alphabetic character followed by one or more numbers. The alphabetic character is the command and the number(s) are the command parameters.

The examples in this section use a personal computer functioning as an IEEE 488 bus controller, using the IOtech Personal488 PC/IEEE 488 board and associated driver software. All examples are given using BASIC. The DAC488HR bus address is set to 10 for all examples.

In order to establish communication with DRVR488 from BASIC, the following sequence must be used:

```
OPEN "\DEV\IEEEOUT" FOR OUTPUT AS #1
IOCTL#1,"BREAK"
PRINT#1,"RESET"
OPEN "\DEV\IEEEIN" FOR INPUT AS #2
```

All of the command examples assume the driver has been properly opened and reset by the above sequence, and that the DAC488HR is set to the factory defaults. The factory defaults are primary addressing with an IEEE 488 bus address of 10.

5.2 Terminators

Bus terminators are fixed and cannot be altered by the command set. The DAC488HR recognizes either linefeed, EOI, or a both as an input terminator. The DAC488HR terminates data output with linefeed plus EOI.

5.3 Command Interpretation

As commands are received by DAC488/HR, they are interpreted in the order in which they are received. Some commands are immediate, which means they immediately take effect. Other commands are deferred, and have no effect on device operation until the execute command (X) is interpreted.

An example of an immediate command is Port Select (Pn), which immediately chooses the port which is being referred to. The immediate commands are *R, Bval, On, Pn, Ql,n,r, Un and all queries.

An example of a deferred command is Trigger Source (Tn), which sets the trigger source used for all analog output ports when X is interpreted. As deferred commands are interpreted, their desired effects are recorded in internal temporary registers. As additional deferred command are interpreted, their effects are added to these registers, possibly overwriting earlier command's effects. Finally, when X is interpreted, the temporary registers are examined in the execution order described below. If two deferred commands that do not affect the same function are received before the execute command, they take effect in the execution order described below. If a deferred command is sent multiple times within a command line, the last occurance of the command will take precedence. (Note a command line is terminated by the X.) For example, if T1 T0 X is sent, the trigger source will be as specified by the T0 command. The T1 command is overridden and never takes effect.

If an error is detected during command processing, commands are ignored up through and including the next execute command. Thus any immediate commands after the error, as well as all deferred commands, are ignored. For example, the command line T1 O0 AA T3 P4 X containing the error AA only executes the O0, because it is an immediate command that occurred before the error. The deferred commands T1 and T3, and the immediate command P4 after the error, have no effect.

Deferred commands help reduce the effects of errors and improve synchronization of command execution. The primary advantage of deferred commands is that they are executed as a group, either all or none. If any errors occur, deferred commands have no effect and the device is left in a consistent state instead of a partially modified, inconsistent state.

The deferred commands are @, An, Cn, Fn, Gn, Hn, In, Jn, Kn, Ln, Mn, Nn, Rn, Sn, Tn, Vval, Ww,l,max,min,d, Yn and Zn.

5.4 Command Execution Order

The immediate commands (*R, Bval, On, Pn, Ql,n,r, Un and all queries) take effect immediately when they are interpreted. Even so, they must be followed by an X command to terminate the command string for correct operation. For example: P1 O0 X.

The deferred commands do not take effect until after the X is interpreted. At that time, they are executed in the following order, regardless of the order they are in the command string:

@, An, Cn, Fn, Gn, Hn, In, Jn, Kn, Ln, Mn, Nn, Rn, Sn, Tn, Vval, Ww,l,max,min,d, Yn and Zn.

5.4.1 Deferred Command Execution Order

First:

S0 or S2 (Recall setting from NVRAM)

Then for each daughter board in order:

F4 or F5 (Setting byte swapping)

An (Buffer mode)

Cn (Trigger control mode)

Kn (Buffer count)

Ln (Location Pointer)

Jn (Gain cal), Hn (Offset cal), and Rn (Range)

Vn (Output voltage)

Ww,l,max,min,d (Waveform generate)

Then the system commands:

Dn (Digital output)

Fn (Format)

Zn (Trigger Delay)

In (Update divider)

Gn (Update source), Tn (Trigger source)

Mn (SRQ mask)

Nn (ESB mask)

@ (Command trigger)

S1, S3 or S4

5.5 Syntax Rules

Commands are identified by a single letter (A through Z), at-sign (@) or an asterisk (*) followed by an single letter.

5.5.1 Case Sensitivity

Commands may be entered in upper or lower case.

Example:

`A0 X` Interpreted the same as `a0 x`

5.5.2 Spaces

White space, which consists of all ASCII values of 32 and below and includes the space, tab, new-line and carriage-return characters, is generally allowed anywhere between commands and command arguments. White space is not allowed in the middle of command options (for instance, `1 2 3` is not the same as `123`.)

Care must be taken when specifying options in hexadecimal. Because hexadecimal numbers may contain valid command letters, the user must be sure that hexadecimal values are separated properly from commands immediately following them. In this case, white space characters are significant. Thus, `B0FF F2 X` is valid, but `B0FFF2 X` will not result in the same action. In the second example, format is not set to decimal as intended, and the `B` value is set to `FFF2` hexadecimal.

Example:

`P1X` Interpreted the same as `P 1 X`

5.5.3 Voltage Values

Voltages are specified as `V#.#`. Thus, all of the following command strings cause +5.6 volts to be output on the selected analog output port.

```
V 5.6 X
V+5.6 X
V 5.6 X
```

5.5.4 Multiple Parameters

If more than one parameter is used for a command, they must be separated by a comma or white space.

Example:

```
Q4000,1000,10
```

or

```
Q4000  1000  10
```

Q1,n,r defines a Sequence Control Block in an analog output port's Sequence Control Table. The first parameter, 4000, defines the starting location in the data buffer. The second parameter, 1000, defines the length (in samples). The third parameter, 10, defines the number of times this block is to be repeated.

5.5.5 Command Strings

Commands may be sent individually or in a string with other commands.

Example:

These three lines of a program:

```
PRINT#1,"OUTPUT10;P1"
PRINT#1,"OUTPUT10;C0"
PRINT#1,"OUTPUT10;V2X"
```

have the same effect as the single line:

```
PRINT#1,"OUTPUT10;P1 C0 V2 X"
```

5.5.6 Execute (X)

Deferred commands are interpreted and processed as they are received. However, they require the Execute (X) command to be issued in order to be executed.

If multiple system commands are used in the same string, each use of the command must be followed by the Execute (X) command. The immediate Port Select (Pn) command and deferred port commands do not have to be followed by X when used in the same string.

Example:

To clear the SRQ mask and then set it for SRQ on trigger and SRQ on end of trigger sequence:

```
PRINT#1,"OUTPUT10;M000 X M003 X"
```

To program port 1 for 3 volts and port 2 for 5 volts using one command string:

```
PRINT#1,"OUTPUT10;P1 V3 P2 V5 X"
```

The immediate commands do not require an Execute command to be processed. For more detail on command types, refer to the full description of each command.

5.5.7 Query Option

Most of the commands offer a query (?) option. Query returns the present configuration or mode of a previously executed command. To use Query, follow the first letter of the command with a question mark (?). Any number of query commands may be combined into one string to create a specialized status command that returns only the information of interest for a given application. Query commands must be followed by the Execute (X) command to terminate the Query. The response may be retrieved after the query has been terminated by the X command. However, if another command or query is issued to the DAC488HR before the previous query is retrieved, a query error will result. In the examples below, the first includes a query error the second shows the proper method for performing multiple queries in a command line.

The first query example is:

```
PRINT#1,"OUTPUT10;P1 C? X R? X "
```

Select analog output port 1, query present trigger control mode, query present range.

```
PRINT#1,"ENTER10"
```
Read DAC488HR for the query response.

```
INPUT#2,A$
```
Read the response from Driver488.

```
PRINT A$
```
Screen shows C1.

```
PRINT#1,"OUTPUT10;U0 X"
```
Read the Event Status Register

```
PRINT#1,"ENTER10"
```
Read DAC488HR for the query response.

```
INPUT#2,A$
```
Read the response from Driver488.

```
PRINT A$
```
Screen shows 004 (query error).

The second query example is as follows:

```
PRINT#1,"OUTPUT10;P1 C? R? T? X"
```

	Select analog output port 1, query present trigger control mode, query present range, query trigger source.
`PRINT#1,"ENTER10"`	Read DAC488HR for the query response.
`INPUT#2,A$`	Read the response from Driver488.
`PRINT A$`	Screen shows `C1R2T5`.

5.5.8 Fixed Formats

Any query command or `Un` command returns a fixed format. For instance, any option that can range up to 65,535 always returns five digits, so zero would be returned as `00000`. In the following command descriptions, leading zeros are included. They are not, however, required when entering the command.

5.6 Serial Poll Status Byte Register

The Serial Poll Status Byte is sent when a serial poll (SPoll) command is received over the IEEE 488 bus from the active controller. Although these bits are always set to indicate the status of DAC488HR, they do not generate an SRQ on the IEEE 488 bus unless the corresponding enable bit in the Service Request Enable (SRE) register has been set with the `Mn` command. Below is a description of each bit in the Serial Poll Status Byte Register.

Bit Location	Value	Description
DIO1 (LSBit)	1	Triggered
DIO2	2	End of Trigger Sequence
DIO3	4	Ready
DIO4	8	Error
DIO5	16	Message Available
DIO6	32	Event Status register Bit (ESB)
DIO7	64	Service Request Bit
DIO8 (MSBit)	128	DMA underrun

All bits in the serial poll status byte register are cleared by either a `*R` command, which returns the DAC488HR to its power-up default conditions, or a read of the serial poll status byte register via the Status command `U0`. The ESB in the Serial Poll Status Register is also cleared by these operations. The Command Error, Execution Error and Device Dependent Error are also cleared with an Error Query (`E?`) command.

DIO1	Triggered	Set when the DAC488HR has sensed a valid trigger. It is cleared whenever a new Trigger Source command (Tn) is executed. This bit can only be used to catch the occurrence of the first valid trigger.
DIO2	End of Trigger Sequence	Set when all analog output ports that have been set to act upon a trigger by the Trigger Control Mode command (C1, C2 or C3) have been triggered and have completed their entire trigger sequence.
DIO3	Ready	Set when the DAC488HR is ready to process another command. It is cleared when the DAC488HR is processing a command line. This bit should be examined with a serial poll prior to issuing a new command line. This allows any detected errors to be traced to the specific command line containing the error. If all the setup information for a specific DAC488HR operation is included in one line, this bit also indicates when all processing is done and the X command is completed. This ensures that the DAC488HR is done processing all state changes before initiating any further activity.
DIO4	Error	Set whenever an error is detected (whenever the value returned by E? is not equal to zero). Cleared when E? is executed.
DIO5	Message Available	Set when the output queue is not empty. It is cleared when the output queue is empty. This bit reflects whether any command responses are still in the output queue.
DIO6	Event Status register Bit (ESB)	Reflects the logical OR of all the bits in the Event Status Register (ESR) ANDed with their equivalent enable bits in the Event Status Enable (ESE) register. If this bit is set, at least one bit in the ESR is set and has its corresponding enable bit in the ESE set. The status command U0 can be issued to read the ESR. See the following for more information on ESR and ESE.
DIO7	Service Request Bit (SRQ)	Set when the DAC488HR is requesting service. It is cleared when an SPoll is performed.
DIO8	DMA underrun	Set if a DMA underrun occurs (C5, 6, and 7 modes only).

5.7 DAC488HR Serial Poll Model

The Serial Poll (SPoll) model for DAC488HR uses four registers and an Output Queue. Their relationship to each other is shown in Figure 5.1.

The Serial Poll Status Byte is an 8-bit number returned when a SPoll is executed. The contents of the Service Request Enable Register, set by the SRQ Mask (Mn) command, determines which bit(s) in the Status Byte Register generate an SRQ on the IEEE 488 bus. A bit set in the Status Byte Register does not generate an SRQ on the IEEE 488 bus unless the corresponding bit in the Service Request Enable Register is set. Details on the bits in the Status Byte Register are discussed in the description of the Mn command.

The Event Status register Bit (ESB) in the Status Byte Register is the combined state of an 8-bit Event Status Register (ESR) and the Event Status Enable (ESE) register. The contents of the ESE, set by the Event Mask (Nn) command, determines which bit(s) in the ESR generate an ESB bit in the status byte register. ESB is not set in the Status Byte Register for a bit set in the ESR unless the corresponding enable bit in the ESE register is set. Details on the bits in the Event Status Register are in the Event Mask (Nn) command description.

The output queue contains all the responses to commands issued to the DAC488HR. It is a first-in, first-out queue. Whenever this queue is not empty, the Message Available (MAV) bit in the Status Byte Register is set to 1. The response queue is cleared by a Device Clear (DCL or SDC) command over the IEEE 488 bus, a *R command to the DAC488HR or by addressing the DAC488HR to talk and reading everything in the response queue.

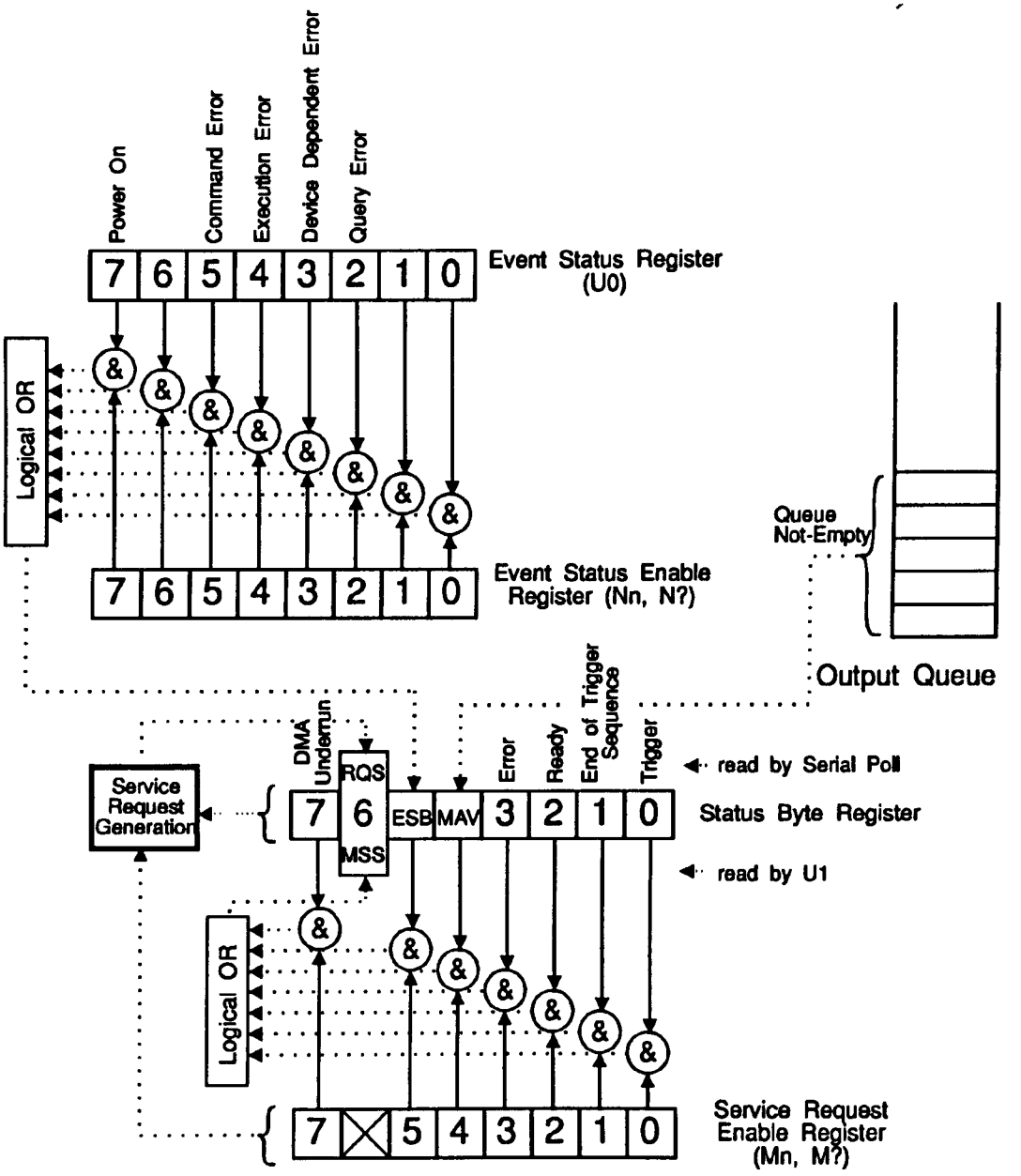


Figure 5.1: DAC488HR Serial Poll Model

5.8 Event Status Register

The Event Status Register (ESR) is another status register in the DAC488HR. Although the ESR bits are always set to indicate the status of DAC488HR, they do not generate an Event Status Bit (ESB) in the Serial Poll Status Register unless the corresponding enable bit in the Event Status Enable (ESE) register has been set with the Event Mask (Nn) command. Below is a description of the significance of each bit in the Event Status Register.

Bit Location	Values	Description
DIO1 (LSBit)	1	Unused
DIO2	2	Unused
DIO3	4	Query Error
DIO4	8	Device Dependent Error (Conflict)
DIO5	16	Execution Error
DIO6	32	Command Error
DIO7	64	Unused
DIO8 (MSBit)	128	Power On

DIO1	Unused	This bit always returns 0 (reserved for future use).
DIO2	Unused	This bit always returns 0 (reserved for future use).
DIO3	Query Error	Set when an attempt is made to read data from the output queue when no data are present or data in the output queue were lost. Data may be lost when too many data are requested to be buffered in the queue (for example, issuing multiple commands to return data without ever reading them).
DIO4	Device Dependent Error	Set when a conflict in programmed parameters is detected. This is also referred to as a conflict error.
DIO5	Execution Error	Set when a parameter exceeds valid limits for a particular command. This is also referred to as Invalid Device Dependent Command Option (IDDCO) error.
DIO6	Command Error	Set when an illegal command is sent to the DAC488HR. This is also referred to as Invalid Device Dependent Command (IDDC) error.
DIO7	Unused	This bit always returns 0 (reserved for future use).
DIO8	Power On	This bit is set on initial power-up of the DAC488HR.

All bits in the ESR are cleared by either a *R command, which returns the DAC488HR to its power-up default conditions, or a read of the ESR via status command U0. The ESB in the Serial Poll Status Register is also cleared by these operations.

5.9 Error Source Register

The error source register indicates which errors, if any have occurred. The individual errors are described in the E? command.

When an error occurs, it sets the appropriate bit in the error source register. This in turn sets a bit in the event status register as shown in Figure 5.2.

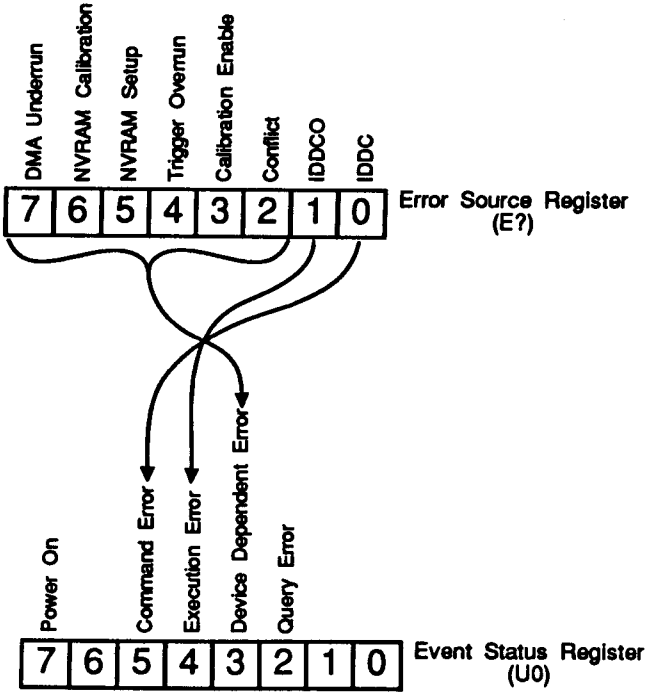


Figure 5.2: DAC488HR Status Reporting Registers

Command Trigger @

Type: System Command, Deferred

@ Generate a command trigger.

The command trigger generates a trigger for all analog output ports when the Trigger Source is set for a command trigger (T5). This command is not processed until an Execute (X) command is received.

The immediate trigger control mode (set by C4) does not require a trigger to start output of data on an analog output port, and therefore is not affected by a command trigger even if the Trigger Source is set to command trigger (T5).

Example:

This example assumes that the analog output port's data buffer has already been written with appropriate data values.

```
PRINT#1,"OUTPUT10;P1 C1 R2 T5 X"
```

Select analog output port 1, select stepped trigger control mode, select 2 volt bipolar range, select command trigger source.

```
PRINT#1,"OUTPUT10;@ X"
```

Trigger the DAC488HR.
The DAC488HR outputs the first value in analog output port 1's data buffer.

```
PRINT#1,"OUTPUT10;@ X"
```

Trigger the DAC488HR.
The DAC488HR outputs the second value in analog output port 1's data buffer.

Reset *R

Type: System Command, Immediate

*R Return DAC488HR to power up state.

This command has the same effect on the DAC488HR as removing and re-applying power. Before the reset is performed all ports are taken to a safe (0 voltage output) state. All calibration constants are returned to their factory defaults and the DAC488HR is returned to the state saved by the last S? command. All data in data buffers and sequence control tables in all analog output ports are erased.

Note because the *R command performs a full power-on reset, one second is required before you can communicate with the DAC488HR. A total of five seconds is required before normal operations can take place.

The IEEE 488 bus commands Device Clear (DCL and SDC) do not have this effect. They clear only the command input buffer, the output queue, any pending commands.

Example:

`PRINT#1,"OUTPUT10;*RX"`	Restore power-on settings to DAC488HR
`PRINT#1,"OUTPUT10;U2 X"`	Read back system settings
`PRINT#1,"ENTER10"`	Read DAC488HR for the response.
`INPUT#2,A$`	Read the response from Driver488.
`PRINT A$`	Screen shows: `D000F0G0I1M000N000T0Y00001Z00000` (factory power-on default).

Buffer Mode An

Type: Port Command, Deferred

A0	Linear buffer mode (default).
A1	Complex buffer mode.
A?	Return present buffer mode setting.

The buffer mode command sets the manner in which the data buffer and sequence control table for each analog output port handles data output in response to triggers.

Linear buffer mode is the factory default and is the simplest method of buffer control. When a trigger is executed, the analog output port starts outputting the data buffer starting at location 0, then linearly runs through all points in the data buffer until the highest location is written. This is repeated from location 0 for buffer count times. The analog output port keeps track of the highest data location loaded when data are written and uses that as the end point of the buffer.

Complex buffer mode is more difficult to use, but allows much more flexibility in the output of data. After the data buffer has been written with values to output, a sequence control table must be defined to determine which data points are to be output and in what order. The sequence control table includes a control block that defines an arbitrary starting point in the buffer, the length (in number of samples) to output, and a repeat count. See the Define Sequence Control Block command (Q1,n,r) for more details.

Changing the Buffer Mode clears the sequence control table.

Example:

`PRINT#1,"OUTPUT10;A0 X"`	Set linear buffer mode
`PRINT#1,"OUTPUT10,A? X"`	Query the buffer mode
`PRINT#1,"ENTER10"`	Read DAC488HR for the query response.
`INPUT#2,A$`	Read the response from Driver488.
`PRINT A$`	Screen shows A0.

Buffer Data Bval

Type: Port Command, Immediate

Bval	Writes data value val into the data buffer and advances the data buffer location pointer for the selected analog output port (format determined by Fn setting, location determined by Ln).
B?	Return the data value from the data buffer at the data buffer location pointer for the selected analog output port (format determined by Fn setting, location determined by Ln).

The Buffer Data command writes a voltage value to the internal data buffer. The buffer data are written to the location pointed to by the data buffer location pointer. The Data Buffer Location Pointer command (Ln) sets the pointer to the desired location. The data buffer location pointer automatically increments after each Buffer Data command is executed. The format for writing data into the data buffer is specified by the Format (Fn) command.

If the format command is set to F0 or F1, there is no need to suppress or add the leading + sign for positive values. The DAC488HR is forgiving in both cases on input. When querying voltage values, however, F0 always returns a + sign for positive values and F1 always returns a leading space for positive values.

Voltages are specified as V#.#. Thus, all of the following command strings cause +5.6 volts to be output on the selected analog output port.

```
V 5.6 X
V+5.6 X
V 5.6 X
```

If the format command is set to F2, all voltages are specified in decimal bits. A leading + or space is optional. When querying voltage values, neither a + nor a leading space is returned for positive values.

If the format command is set to F3, voltages are specified in hexadecimal bits. Signed 16-bit integer format is assumed, so leading + or – signs are not accepted. 0 to 7FFF are interpreted as 0 to 32,767 decimal. FFFF to 8000 are interpreted as –1 to –32,768 decimal when in bipolar mode. In unipolar mode, 0 to FFFF are interpreted as 0 to 65,535 bits.

Care must be taken when specifying options in hexadecimal. Because hexadecimal numbers may contain valid command letters, the user must be sure that hexadecimal values are separated properly from commands immediately following them. In this case, white space characters, which consist of all ASCII values of 32 and below and include the space, tab, new-line and carriage-return characters, are significant. Thus, B0FF F2 X is valid, but B0FFF2 X will not result in the same action. In the second example, format is not set to decimal as intended, and the B value is set to FFF2 hexadecimal.

Successive locations may be written with Bv,v,v,v,v. The last value must be followed by an X before any other commands are sent so subsequent commands are not perceived by the DAC488HR as data. The Buffer Data command is not deferred until the X command is received. As each data point is processed, the value is placed into the associated analog output port's data buffer immediately.

Data may be directed to the proper analog output port in one of two ways.

If the DAC488HR is set for primary addressing mode, the Port Select command (Pn) must be used to select which analog output port's data buffer is written. Data points can then be written with the Buffer Data command.

If the DAC488HR is set for secondary addressing mode, the Port Select command (Pn) can still be used through the command channel (secondary address 0) to select which analog output port's data buffer is written. Data points are then loaded with the Buffer Data command through the command channel (secondary address 0). The destination analog output port can also be specified implicitly by directing data to the secondary address associated with that analog output port. Secondary addresses 01 through 04 correspond to analog output ports 1 through 4, respectively.

Binary data may be transferred to the DAC488HR in primary addressing mode or through the command channel in secondary addressing mode. Binary data is indicated by a leading '#'. If the number of data bytes to be transferred is known, the byte count may be specified in the following format:

```
# <digit  count> <byte count>
```

Digit count refers to the number of digits in the byte count. For instance, to transfer 1000 bytes of data, the following command string should be issued to DAC488HR:

```
B#41000 (1000 bytes of binary data) X
```

If binary data of undetermined length is to be transferred, the digit count following '#' should be 0. In this case, the binary data must be terminated with a LF with EOI asserted. For instance, to transfer a random length of binary data to the DAC488HR, issue the following command string to the DAC488HR:

```
B#0 (arbitrary count of binary data) (LF EOI) X
```

Binary data transfers are not affected by the current format (F) setting.

Example of ASCII format transfer:

```
PRINT#1,"OUTPUT10;P1"
```

Select analog output port 1.

```
PRINT#1,"OUTPUT10;L0 F2 R2 X"
```

Set data buffer location pointer to 000000, set format to decimal bits, set range to ±2 volts.

```
PRINT#1,"OUTPUT10;B2000,32000,-1000 X"
```

Store 2000 in data buffer location 000000, store 32,000 in location 000001, store –1000 in location 000002. Note that the data buffer location pointer automatically increments.

```
PRINT#1,"OUTPUT10;B1500,-300,1000 X"
```

Store 1500 in data buffer location 000003, store -300 in location 000004, store 1000 in location 000005.

```
PRINT#1,"OUTPUT10;F3 X"
```
Set format to hexadecimal.

```
PRINT#1,"OUTPUT10;BFF,4000,3E8 X"
```

Store 255 in data buffer location 000006, store 16,384 in location 000007 and store 1000 in location 000008.

```
PRINT#1,"OUTPUT10;F1 X"
```
Set format to signed volts.

```
PRINT#1,"OUTPUT10;B-2.000 X"
```

Store -2 volts in data buffer location 000009.

```
PRINT#1,"OUTPUT10;L2 X B? X"
```

Set data buffer location pointer back to 000002.

```
PRINT#1,"ENTER10"
```
Query data value at that data buffer location.

```
INPUT#2,A$
```
Read the response from Driver488.

```
PRINT A$
```
Screen shows `B-0.06100`.

Example of binary transfer:

```
PRINT#1,"OUTPUT10;P1 X"
```
Select analog output port 1.

```
PRINT#1,"OUTPUT10;L0 F2 R2 X"
```

Set data buffer location pointer to 000000, set format to decimal bits, set range to ±2 volts.

```
PRINT#1,"OUTPUT10;B#0";
```
Send binary prefix, suppress line feed.

```
PRINT#1,"OUTPUT10#100
BUFFER &H20100"
```
Send 100 characters from memory at &H20100.

```
PRINT#1,"OUTPUT10;";CHR$(10)
```

Send line feed with EOI asserted to terminate transfer.

Trigger Control Mode Cn

Type: Port Command, Deferred

C0	Trigger response disabled (default).
C1	Step mode.
C2	Burst mode.
C3	Waveform mode.
C4	Immediate mode.
C5	Continuous Mode, one analog output port.
C6	Continuous Mode, two analog output ports, odd.
C7	Continuous Mode, two analog output ports, even.
C?	Returns existing control mode for selected analog output port.

The Trigger Control Mode command defines how each analog output port responds to a trigger in the output of its data.

Issuing any trigger control mode command stops the prior trigger mode activity and rearms the analog output port for the new mode. For example, if the DAC488HR is generating a waveform, it may be halted and rearmed by issuing the C3 command.

Note the port LED indicates the state of the analog output in the following manner. If the LED if OFF, then there is 0 Volts present on the output. If the LED is ON, then there is a non-zero voltage present.

In all C modes, the LED begins flashing immediately after the execution of the C command and is turned OFF when "C0" is executed.

The modes and their operation are described below.

Disabled (factory power-up default)

Any trigger response is completely disabled. If the analog output port is currently executing another trigger control mode, trigger processing is immediately stopped and the analog output voltage is held at the last value output. All analog output ports power up in this mode.

Step Mode

In Step mode, each trigger event causes the analog output port to output the next value from the data buffer. (If a trigger is detected while the output is taking place, a trigger overrun will occur.) Trigger detection is then rearmed. When the end of the defined data buffer is reached, the data buffer location pointer is reset to the beginning of the data buffer and the process is repeated buffer count times. After the buffer count is exhausted, trigger detection is disarmed, and the output is held at the last output value. If the buffer count is set to 0, the buffer is cycled through forever.

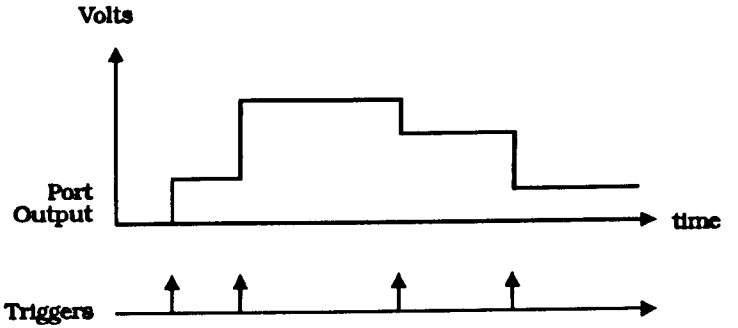


Figure 5.3: Stepped Mode Output

Burst Mode

In burst mode, the data buffer is output at the update rate each time a trigger is detected. When the end of the defined data buffer is reached, the analog output is held at the last output value and triggers are rearmed. Triggers are rearmed and the data buffer output until the buffer count is exhausted. After the buffer count is exhausted, trigger detection is disarmed, and the output is held at the last output value. If the buffer count is set to 0, the buffer is cycled through forever. If another trigger occurs before the entire data buffer is output, a trigger overrun error occurs.

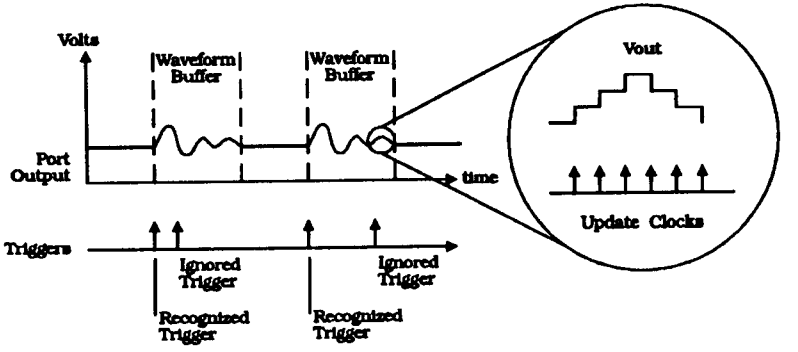


Figure 5.4: Burst Mode Output

Waveform Mode

In waveform mode, the data buffer is output at the update rate each time a trigger is detected. When the end of the defined data buffer is reached, it is repeated until the buffer count is exhausted. (If a trigger is encountered before the buffer count is exhausted, a trigger overrun will occur.) After the buffer count is exhausted, trigger detection is disarmed and the output is held at the last output value. If the buffer count is set to 0, the buffer is cycled through forever.

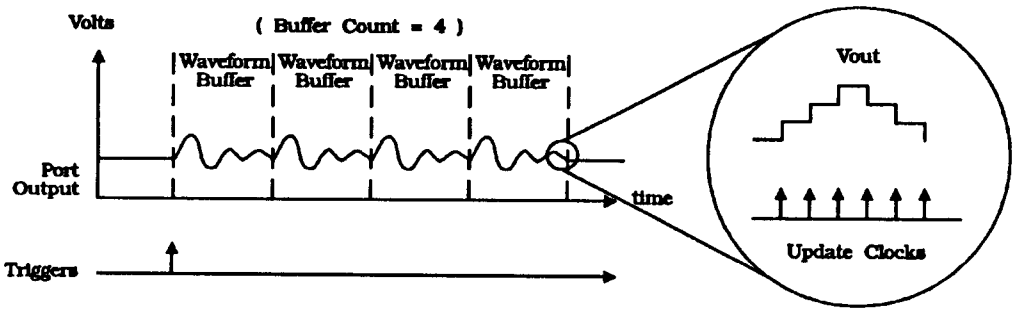


Figure 5.5: Waveform Mode Output

Immediate Mode

Immediate mode is identical to Waveform Mode except that no trigger is required to initiate data buffer output. Refer to Figure 5.6 for diagram of Immediate Mode Output. Data buffer output begins immediately upon receipt of the X command at the end of the command line setting this mode. Upon receipt of C4, the data buffer is output at the update rate. When the end of the defined data buffer is reached, it is repeated until the buffer count is exhausted. After the buffer count is exhausted, trigger detection is disabled, and the output is held at the last output value. If buffer count is set to 0, the buffer is cycled through forever.

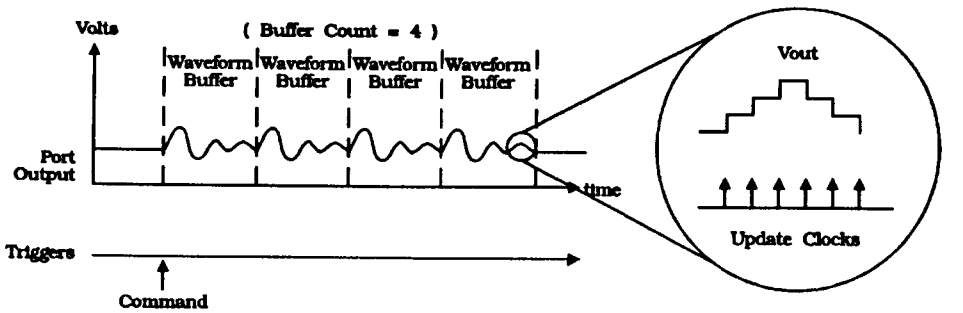


Figure 5.6: Immediate Mode Output

Continuous Modes

Three modes are supported for continuous data throughput from the IEEE 488 data bus to the analog output ports.

Continuous mode - one analog output port is for continuous output to a single port at up to 100 k samples per second. Only one analog output port at a time can be set to this mode. Attempting to set two analog output ports to this mode, or attempting to set one analog output port to this mode when another is set to this mode causes a Device Dependent (conflict) error.

Continuous mode - two analog output port odd configures an analog output port to capture the 1st, 3rd, 5th, etc. samples from a stream of interleaved data. Only one analog output port at a time may be set to this mode. Attempting to set two analog output ports to this mode, or attempting to set one analog output port to this mode when another is set to this mode causes a Device Dependent (conflict) error.

Continuous mode - two analog output port even configures an analog output port to capture the 2nd, 4th, 6th, etc. samples from a stream of interleaved data. Only one analog output port at a time may be set to this mode. Attempting to set two analog output ports to this mode, or attempting to set one analog output port to this mode when another is set to this mode causes a Device Dependent (conflict) error.

Interleaved data consist of two bytes of binary data for one analog output port, then two bytes of binary data for the other, and so on. Analog output for both of the interleaved ports will not begin until both ports' buffers have been filled.

Each port configured in continuous mode requires 4k byte sample size to trigger the output. If interleaved mode is specified, the output will not be triggered until the last port has filled to 4k byte of its sample buffer.

Binary transfer loading of the sample buffers in continuous mode is performed by issuing a # command. This command has two forms: fixed length and arbitrary length transfers. Fixed length transfers are implemented by issuing the following:

#d$n_1 \ldots n_d$<data>

Where d is the number of digits in the transfer size, $n \ldots n_d$ reprsents the transfer size (in bytes), and data represents the binary data.

For example, the following would specify an 8k byte transfer of fixed length in continuous mode:

```
#48000
```

Arbitrary length transfers have the following form:

```
#0<data>LF EOI
```

Where data represents arbitrary length binary data. Note that LF EOI is necessary to terminate the transfer.

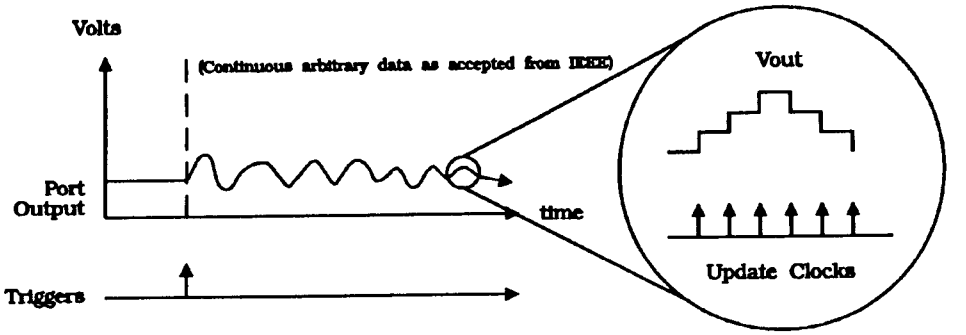


Figure 5.7: Continuous Mode Output

Example:

`PRINT#1,"OUTPUT10;P1 C0 X"`	Disable trigger response for analog output port 1.
`PRINT#1,"OUTPUT10;P1 C? X"`	Query trigger control mode for analog output port 1.
`PRINT#1,"ENTER10"`	Read DAC488HR for the query response.
`INPUT#2,A$`	Read the response from Driver488.
`PRINT A$`	Screen shows C0.

Digital Output Dn

Type: System Command, Deferred

`Dn`	Outputs the value n (0-255) to a digital output port. Default is `D0`.
`D?`	Return the present value of the digital output port in three-digit decimal format.

The Digital Output command outputs eight bits of data to the digital output port. The data appear at the digital output port as soon as the command is executed. There are no triggering or buffer options available when using the digital output port.

The default output value is 0 (all digital output lines low, 0 volts). All bits on the digital output port are affected when using this command. For example, if `D1 X` is sent, line 1 is set to a logic high level (+5 volts) and the other lines are set to a logic low level (0 volts), with the digital output configuration jumper set to TTL compatible outputs. If the digital output configuration jumper is set to high-voltage, high-current relay compatible outputs, output 1 is on (pulled to GND by the open collector output transistor) and the other outputs are off (open).

Example:

```
PRINT#1,"OUTPUT10;D6 X"      set lines 2 and 3 high on the digital output port
                             (6 = 0000 0110 binary)

PRINT#1,"OUTPUT10;D?"        read the programmed state of the digital output port

PRINT#1,"ENTER10"

INPUT#2,A$

PRINT A$                     display shows D006
```

Error Query E?

Type: System Command, Immediate

E? Returns and clears present error condition.

After execution of the Error Query command, the DAC488HR returns one of the following error codes:

E000 No error (default).

E001 Invalid device dependent command (IDDC).

E002 Invalid device dependent command option (IDDCO).

E004 Command conflict.

E008 Calibration enable.

E016 Trigger overrun.

E032 Non-volatile setup.

E064 Non-volatile calibration constants.

E128 DMA Underrun.

The Error Query command returns the present error condition of the DAC488HR. After execution of the Error Query command, error conditions are cleared. When an error occurs, it sets the appropriate bit in the error source register. This in turn sets a bit in the event status register as shown in Figure 5.8.

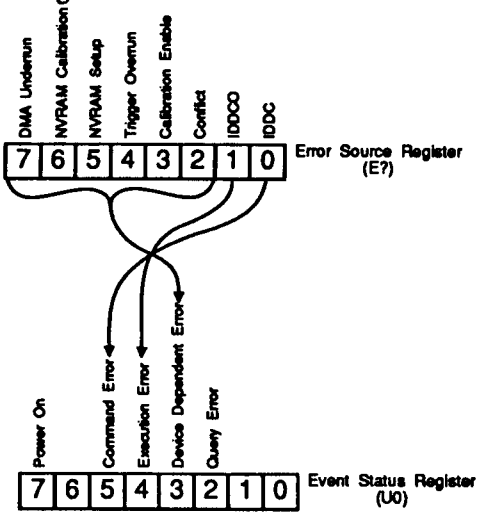


Figure 5.8: DAC488HR Status Reporting Registers

Return	Error	Description
001	IDDC	Set when an illegal command is sent to the DAC488HR (ex.: !0X).
002	IDDCO	Set when the option to a legal command is out of range (ex.: A2X).
004	Command Conflict	Set when a combination of commands is illegal, or some condition in the DAC488HR prevents the execution of a command (example: R6 V-2.0X – unipolar does not allow negative voltages).
008	Calibration Enable	Set when an attempt to write NVRAM setups (S1) or NVRAM calibration constants (S3) is made when the CAL ENABLE switch is in the disabled position.
016	Trigger Overrun	Set when triggers occur too fast for DAC488HR processing. If a trigger event occurs before the DAC488HR has finished processing a previous event, a Trigger Overrun is generated.
032	Non-volatile Setup	Set when the checksum on the setup stored in NVRAM is invalid. When this occurs, the DAC488HR reverts to factory defaults for all settings, but the NVRAM checksum is not updated until the execution of an S1 command.
064	Non-volatile Calibration Constants	Set when the checksum on the calibration constants stored in NVRAM is invalid. When this occurs, the DAC488HR reverts to nominal values for all calibration constants, allowing normal operation. However, the NVRAM checksum is not updated until the execution of an S3 command.
128	DMA Underrun	Set when the DAC488HR has at least one analog output port running in continuous output mode, and the rate of data transfer from the IEEE 488 bus is slower than the rate of output.

When an error has occurred, the ERROR indicator light on the front panel of the DAC488HR lights and the error bit in the serial poll status byte register is set to one. The ERROR indicator light remains lit and the error bit in the status byte register remains set to one until an Error Query (E?).

Example:

`PRINT#1,"OUTPUT10;A2 X"`	Set linear buffer mode to an illegal value (ERROR indicator light should turn on).
`PRINT#1,"OUTPUT10,E? X"`	Query the error condition.
`PRINT#1,"ENTER10"`	Read DAC488HR for the query response.
`INPUT#2,A$`	Read the response from Driver488.
`PRINT A$`	Screen shows E002 (Invalid device dependent command option). ERROR indicator light should go off.

Format Fn

Type: System Command, Deferred

F0	Sets format to signed volts in ±10.00000 (default).
F1	Signed volts in fixed format (+ implied) −10.00000 (leading space for positive).
F2	Sets format to decimal bits.
F3	Sets format to hexadecimal bits.
F4	Sets all daughterboard output order to low byte first.
F5	Sets all daughterboard output order to high byte first.
F?	Returns present format selection in the following order: Fn, Fm where n is format 0-3 and m is output order 4-5.

The Format command selects the input and output format the DAC488HR uses when sending a voltage value to the controller.

Voltages are programmed with decimal floating point numbers in the range of ±10.00000. As opposed to previous versions, the F0 and F1 formats will pad the two digits to the left of the decimal point with zero's if there are no significant digits there. Also the F? now returns Fn,Fm unlike previous versions which only returned Fn.

Voltages may also be programmed in decimal bits (range ±32,767) or 16-bit hexadecimal two's complement numbers (range = 0000 to FFFF).

RANGE	RESOLUTION	
	Unipolar	Bipolar
1 V	15.3µV	30.5µV
2 V	30.5µV	61µV
5 V	76.3µV	153µV
10 V	153µV	305µV

Example:

```
PRINT#1,"OUTPUT10;P1 X"   Select analog output port 1.

PRINT#1,"OUTPUT10;L0 F2 R2 X"
```

Set data buffer location pointer to 000000, set format to decimal bits, set range to ±2 volts.

```
PRINT#1,"OUTPUT10;B2000,32000,-1000 X"
```

Store 2000 in data buffer location 000000, store 32,000 in location 000001, store −1000 in location 000002. Note that the data buffer location pointer automatically increments.

```
PRINT#1,"OUTPUT10;1500,-300,1000 X"
```

Store 1500 in data buffer location 000003, store −300 in location 000004, store 1000 in location 000002.

```
PRINT#1,"OUTPUT10;F3 X"
```

Set format to hexadecimal.

```
PRINT#1,"OUTPUT10;BFF,4000,3E8 X"
```

Store 255 in data buffer location 000006, store 16,384 in location 000007 and store 1000 in location 000008.

```
PRINT#1,"OUTPUT10;F1 X"
```

Set format to signed volts.

```
PRINT#1,"OUTPUT10;B-2.000 X"
```

Store −2 volts in data buffer location 000009.

```
PRINT#1,"OUTPUT10;L2 X B? X"
```

Set data buffer location pointer back to 000002.

```
PRINT#1,"ENTER10"
```

Query data value at that data buffer location.

```
INPUT#2,A$
```

Read the response from Driver488.

```
PRINT A$
```

Screen shows `B-00.06100`.

Update Source, Mode Gn

Type: System Command, Deferred

G0	5 MHz, synchronous.
G1	2.8224 MHz (64 x 44.1 KHz), synchronous.
G2	3.072 MHz (64 x 48 KHz), synchronous.
G3	200 kHz, synchronous (default).
G4	External Clock, synchronous.
G5	5 MHz, asynchronous.
G6	2.8224 MHz, asynchronous.
G7	3.072 MHz, asynchronous.
G8	200 kHz, asynchronous.
G9	External Clock, asynchronous.
G?	Returns present update source.

This command specifies the Update Clock source and the Update Clock mode. For maximum flexibility, five different clock sources for the Update Clock may be specified: 5 MHz, 2.8224 MHz, 3.072 MHz, 200 kHz or external clock (TTL compatible). Any of these clock sources can be used as the raw clock input to the divider set by the Update Divider (In) command. The update rate is the time between update clocks. It is programmable by the user via system commands. A 16-bit divide counter provides update rates to meet application requirements.

The Update Rate is calculated using the formula:

$$\text{Update Rate} = \frac{\text{Update Source frequency}}{\text{Update Divider}}$$

For example, to generate an Update Rate of 100 kHz, a 200 kHz Update Source and an Update Divisor of 2 can be used. Alternatively, a 5 MHz source and an Update Divisor of 50 can be used.

NOTE: The DAC488HR will issue a conflict error, if an update rate of over 100 kHz is specified. Exceeding this maximum update rate is allowed but may cause port command failures. In order to prevent the inadvertent setting of the conflict error, the In and Gn commands should be used on the same command line without an "X" separating them

There are two different modes of Update Clock generation. Both of these modes specify how triggers are recognized and how they affect Update Clock generation. The update clock can be generated in synchronous or asynchronous mode. These two modes determine how the DAC488HR responds to triggers.

When synchronous update mode is used, triggers are recognized at the next regular occurrence of the update clock. Maximum trigger latency is equal to the update rate. Synchronous mode

guarantees that no discontinuities in the update rate occur. This ensures that no frequency components higher than the update rate show up in the frequency spectrum of generated waveform outputs.

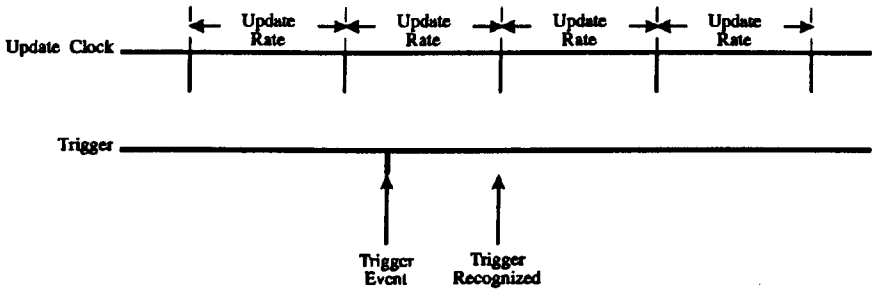


Figure 5.9: Synchronous Update Mode

When asynchronous update mode is used, triggers are recognized as quickly as possible. Update rate generation is resynchronized to the occurrence of the trigger event. This allows the minimum reaction time to trigger events, but allows higher frequency components than the update rate to appear in the frequency spectrum of waveform outputs from the DAC488HR. The minimum trigger latency in this case is two periods of the update source rate. Thus, with an update source of 5 MHz, asynchronous update mode, minimum trigger latency is 400 nanoseconds. The maximum trigger latency is equal to the time it takes to shift the next sample across the optoisolator barrier to the analog circuitry. Once an update clock has occurred, it takes approximately four µseconds to shift the next sample across the optoisolator barrier in preparation for the next update.

In addition, further update clocks are resynchronized to the occurrence of the trigger event. This mode minimizes trigger latency, but does cause discontinuity in the update rate of all analog output ports for every occurrence of a trigger event.

Update	Minimum Trigger Latency
5MHz	400 ns
2.8224 MHz	710 ns
3.072 MHz	650 ns
200 KHz	10 µsec

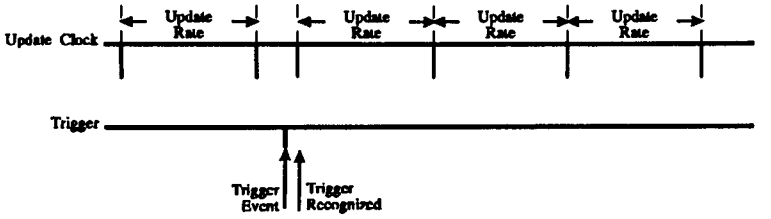


Figure 5.10: Asynchronous Update Mode

Example:

`PRINT#1,"OUTPUT10;G4 X"`	Set update source to external, update mode to synchronous.
`PRINT#1,"OUTPUT10,G? X"`	Query the update source, mode.
`PRINT#1,"ENTER10"`	Read DAC488HR for the query response.
`INPUT#2,A$`	Read the response from Driver488.
`PRINT A$`	Screen shows G4.

Offset Calibration Hn

Type: Port Command, Deferred

Hn	Set the offset calibration constant to value n (0 - 4095) for selected range and analog output port.
H?	Returns the offset calibration constant value for the selected range and analog output port.

Calibration of the analog output port involves two calibration constants. Offset Calibration sets the calibration constant that trims out offset errors at the output of the analog output port. This value should nominally be half of the adjustment range (2048). When calibrating the DAC488HR, this value should be used as a starting point.

Calibration constants must be programmed with trigger detection disabled (Trigger Control Mode set to C0). If calibration constants are programmed while trigger detection is enabled (C1 through C7), trigger detection is disabled (C0) before the calibration constants are changed.

The last programmed output voltage appears at the analog output upon execution of this command.

Example:

```
PRINT#1,"OUTPUT10;P1 C0 R2 H2048 X"
```

Select Port 1, select ±2 volt range, set offset for ±2 volt range to 2048.

```
PRINT#1,"OUTPUT10;P1 H? X"
```

Query Port 1 for present offset calibration constant.

```
PRINT#1,"ENTER10"
```

Read the response from the DAC488HR.

```
INPUT#2,A$
```

Read the response from Driver488.

```
PRINT A$
```

Display shows H2048.

Update Divider In

Type: System Command, Deferred

In	Set Update Source divisor to generate Update Clock (1 - 65,535) (default is 2).
I?	Returns the present Update Clock interval in Update Source clocks.

The Update Divider command specifies the time interval between the output of each data point. The frequency of any cyclical data stored in the data buffer can be controlled by selecting the proper interval based on the number of voltage points per cycle.

The Update Source, Mode command (G) chooses the clock source for update clock generation. This update source clock is fed to the input of a 16-bit down counter for generation of the Update Clock. The Update Divider determines the number of update source clocks between update clocks. Every time the counter underflows, an Update Clock pulse is generated.

The external clock source can be used directly by setting the update divider to one.

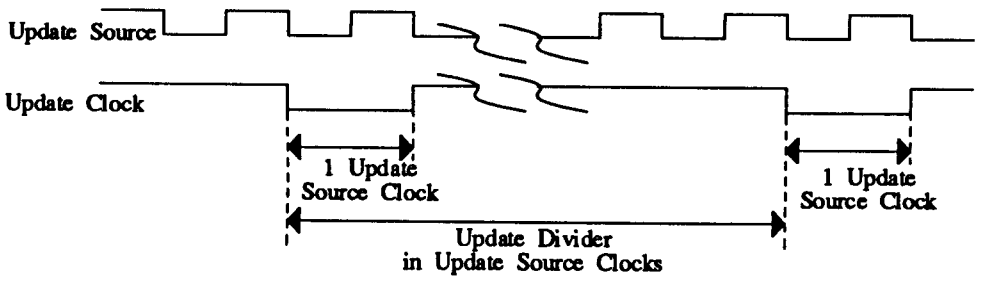


Figure 5.11: Update Clock Timing Diagram

The Update Rate is calculated using the formula:

$$\text{Update Rate} = \frac{\text{Update Source frequency}}{\text{Update Divider}}$$

For example, to generate an Update Rate of 100 kHz, a 200 kHz Update Source and an Update Divisor of 2 can be used. Alternatively, a 5 MHz source and an Update Divisor of 50 can be used.

NOTE: The DAC488HR will issue a conflict error, if an update rate of over 100 kHz is specified. Exceeding this maximum update rate is allowed but may cause port command failures. In order to prevent the inadvertent setting of the conflict error, the In and Gn commands should be used on the same command line without an "X" separating them.

When in the Burst (C2), Waveform (C3), Immediate (C4), or Continuous (C5, C6, C7) Trigger Control Modes, this value and the Update Source determine the time interval between voltage changes on the analog output ports.

When in Stepped Trigger Control Mode (C1), with the Update Mode defined as asynchronous (G5 through G9), this value determines the maximum trigger latency.

Example:

`PRINT#1,"OUTPUT10;G3I20 X"`	Set update divider to 20 and update source to 200kHz.
`PRINT#1,"OUTPUT10,I? X"`	Query the update divider value.
`PRINT#1,"ENTER10"`	Read DAC488HR for the query response.
`INPUT#2,A$`	Read the response from Driver488.
`PRINT A$`	Screen shows `I20`.

Gain Calibration Jn

Type: Port Command, Deferred

Jn	Set the gain correction calibration constant to value n (0 - 4095) for the selected range and analog output port.
J?	Returns the gain correction calibration constant for the selected range and analog output port.

Calibration of an analog output port involves two calibration constants. This command sets the calibration constant that trims gain errors at the output of the analog output port. This value should nominally be half of the adjustment range (2048). When calibrating the DAC488HR, this value should be used as a starting point.

Calibration constants must be programmed with trigger detection disabled (C0). If calibration constants are programmed while trigger detection is enabled (C1 through C7), trigger detection is disabled (C0) before calibration constants are changed.

The last programmed output voltage appears at the analog output upon receipt and execution of this command.

Example:

```
PRINT#1,"OUTPUT10;P1 C0 R2 J2048 X"
```

select Port 1, select ±2 volt range, set gain calibration constant for ±2 volt range to 2048.

```
PRINT#1,"OUTPUT10;P1 J? X"
```

Select Port 1, query present gain calibration constant.

```
PRINT#1,"ENTER10"
```

Read DAC488HR for query response.

```
INPUT#2,A$
```

Read Driver488 for response.

```
PRINT A$
```

Display shows J2048.

Buffer Count Kn

Type: Port Command, Deferred

Kn	Set the number of times the entire data buffer is repeated (0-65,535, 0 = continuous). Default is K1.
K?	Return the number of repetitions specified.

The Buffer Count command sets the number of times the entire data buffer is repeated for every Trigger Control Mode of operation except the continuous modes (C5, C6 and C7). If buffer count is set to 0, the entire data buffer is repeated forever.

In the simple linear buffer mode (A0), this sets the number of times that the entire buffer is output before recognition of triggers is disabled by the analog output port.

In the complex buffer mode (A1), this sets the number of times that the sequence control table is run through before recognition of triggers is disabled by the analog output port.

Example:

`PRINT#1,"OUTPUT10;P1 K1000 X"`	Select Port 1, set buffer repeat count to 1000.
`PRINT#1,"OUTPUT10;P1 K? X"`	Query buffer count for Port 1.
`PRINT#1,"ENTER10"`	Read DAC488HR for query response.
`INPUT#2,A$`	Read Driver488 for response.
`PRINT A$`	display shows K01000.

Data Buffer Location Pointer Ln

Type: Port Command, Deferred

Ln — Set the data buffer location pointer for the selected analog output port to location n. Default is L0.
n is between 0 and 8,191 standard.
n is between 0 and 131,071 with the MEMX3 option.
n is between 0 and 491,519 with the MEMX4 option.

L? — Returns the present buffer location. The L? command may be used when the DAC488HR is running in any trigger control mode. In this case, the value returned is the approximate location in the buffer that is being output at that time.

The Data Buffer Location Pointer command sets the location pointer to a specified location in the internal data buffer. This command is used in conjunction with the Buffer Data (Bval) command. After a data value is put in the buffer using the Bval command, the data buffer location pointer is automatically incremented.

Once an analog output port is triggered in any trigger control mode (Cn) except continuous, the data buffer location pointer always points to the next location containing the data to be output. Each analog output port has its own data buffer location pointer, so the location pointer can be set to a different location for each port.

If the user sets the data buffer location pointer to 0 and starts writing data to the data buffer, the buffer contents are overwritten.

Example:

`PRINT#1,"OUTPUT10;P1 F2 X"`	Select analog output port 1 for this example. Set format to decimal bits.
`PRINT#1,"OUTPUT10;L2 X"`	Set data buffer location pointer to location 2.
`PRINT#1,"OUTPUT10;L? X"`	Query data buffer location pointer.
`PRINT#1,"ENTER10"`	Read DAC488HR for query response.
`INPUT#2,A$`	Read Driver488 for response.
`PRINT A$`	Screen shows L000002.
`PRINT#1,"OUTPUT10;B1000 X"`	Store value to data buffer.
`PRINT#1,"OUTPUT10;L? X"`	Query data buffer location pointer.

`PRINT#1,"ENTER10"`	Read DAC488HR for query response.
`INPUT#2,A$`	Read Driver488 for response.
`PRINT A$`	Screen shows `L000003`. The data buffer location pointer is incremented after the `Bval` command.

SRQ Mask Mn

Type: System Command, Deferred

M000	Clear SRQ mask (default).
M001	SRQ on trigger.
M002	SRQ on end of trigger sequence.
M004	SRQ on ready.
M008	SRQ on error.
M016	SRQ on Message AVailable (MAV).
M032	SRQ on enabled event (ESB).
M128	SRQ on DMA underrun
M?	Read Service Request Enable Register. Return present service request mask.

The SRQ Mask command enables SRQs on one or more of the conditions listed above. Multiple SRQ Mask conditions can be enabled simultaneously by issuing them separately or by combining them in one command string. The programmed SRQ modes remain enabled until the M000 (clear SRQ mask) command is sent, or the controller sends a *R command. This command acts directly on the Service Request Enable Register (see the DAC488HR Serial Poll Model, Section 5.7).

Like all mask commands, the SRQ Mask bits are logically ORed together as they are received.

Example:

`PRINT#1,"OUTPUT10;M001 X M002 X"`	Enable SRQ on Trigger, enable SRQ on end of trigger sequence.
`PRINT#1,"OUTPUT10;M? X"`	Read present Mn setting.
`PRINT#1,"ENTER10"`	Read DAC488HR for the query response.
`INPUT#2,A$`	Read the response from Driver488.
`PRINT A$`	Computer screen shows M003, the logical "OR" of both of the above settings.

Event Mask Nn

Type: System Command, Deferred

N000	Clear event mask (default).
N001	End of trigger sequence.
N004	Enable query error.
N008	Enable device dependent error.
N016	Enable execution error.
N032	Enable command error.
N128	Power-on.
N?	Read standard event status enable register.

This command directly sets the Event Status Enable Register (ESE). ESE determines which conditions in the Event Status Register (ESR) are enabled to generate Event Status register Bit (ESB) in the Serial Poll Status Register. See the section on the DAC488HR Serial Poll Model (Section 5.7) for complete details.

Multiple ESR bits can be enabled simultaneously by issuing them separately or by combining them in one command string. The programmed event enables remain set until a Clear Event Mask (N000) command is sent or the controller sends a *R command.

A query error (N004) is set when an attempt is made to read data from the output queue when no data are present or data in the output queue were lost. Data may be lost when too many data are requested to be buffered in the queue (for example, issuing multiple commands to return data without ever reading them).

A device dependent error (N008) is set when one of the following six errors occur: DMA underrun, NVRAM Calibration, NVRAM Setup, Trigger Overrun, Calibration Enable and Conflict error. Refer to Figure 5.12.

An execution error (N016) is set when a parameter exceeds valid limits for a particular command. This is also referred to as Invalid Device Dependent Command Option (IDDCO) error.

A command error (N032) is set when an illegal command is sent to the DAC488HR. This is also referred to as Invalid Device Dependent Command (IDDC) error.

A power on (N128) is set whenever DAC488HR is first powered up.

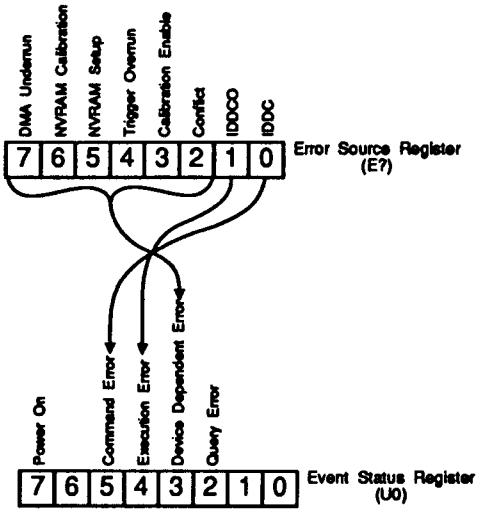


Figure 5.12: DAC488HR Status Reporting Registers

Example:

`PRINT#1,"OUTPUT10;N0 X"`	Clear ESE.
`PRINT#1,"OUTPUT10;N? X"`	Read ESE back, computer screen shows `N000`.
`PRINT#1,"OUTPUT10;N4 X N8 X"`	Set enable on query error, enable on Device Dependent (conflict) error.
`PRINT#1,"OUTPUT10;N? X"`	Query event mask setting.
`PRINT#1,"ENTER10"`	Read DAC488HR for the query response.
`INPUT#2,A$`	Read the response from Driver488.
`PRINT A$`	Computer screen shows `N012`, the logical OR of these two conditions.

Sequence Pointer On

Type: Port Command, Immediate

On	Set the sequence pointer to n. Default is 00000.
O?	Returns the present sequence pointer.

The sequence pointer determines where in the Sequence Control Table the next operation involving a Sequence Control Block occurs. This affects defining a Sequence Control Block (Q1,n,r) or querying a Sequence Control Block (Q?). The sequence pointer determines which sequence control block is referenced by the next Q command. Each time the Q command is used to define a sequence control block, this pointer automatically increments. Thus, successive sequence control blocks may be loaded without setting the sequence pointer each time.

The size of the Sequence Control Table determines the range of values (n) allowed for this command. The size of the Sequence Control Table depends on the data buffer memory. The limits are:

Memory Size	Sequence Pointer Range
8K	0-127
128K	0-2047
480K	0-2047

Example:

`PRINT#1,"OUTPUT10;P1 X"`	Select analog output port 1.
`PRINT#1,"OUTPUT10;O0 X"`	Set sequence pointer to 0000.
`PRINT#1,"OUTPUT10;Q0,0,0 X"`	Delete sequence control block 0.
`PRINT#1,"OUTPUT10;O? X"`	Query present sequence pointer.
`PRINT#1,"ENTER10"`	
`INPUT#2,A$`	Read Driver488 for response.
`PRINT A$`	Screen shows 00000. Note that the sequence pointer did not increment.

`PRINT#1,"OUTPUT10;Q0,100,2 X"`	Define a sequence control block starting at data buffer location 000000, 100 samples in length, to repeat two times.
`PRINT#1,"OUTPUT10;O? X"`	Query present sequence pointer.
`PRINT#1,"ENTER10"`	
`INPUT#2,A$`	Read Driver488 for response.
`PRINT A$`	Screen shows `00001`. Note that the sequence pointer incremented.

Port Select Pn

Type: System Command, Immediate

P1	Select analog output port 1 (default).
P2	Select analog output port 2.
P3	Select analog output port 3 (DAC488HR/4 only).
P4	Select analog output port 4 (DAC488HR/4 only).
P?	Return present analog output port selection.

The Port command determines which analog output port is selected for use with subsequent commands. Only one analog output port at a time may be selected.

Port Select, unlike most other commands in the DAC488HR, is not deferred until an Execute (X) is sent. It becomes active as soon as it is encountered in the command line. This allows multiple analog output ports to be selected and commands issued to them within the same command line terminated with only one execute (X) command. When the execute command is encountered, Pn is left at its last value.

Example:

```
PRINT #1,"OUTPUT10;P1 C0 P2 C1 P3 C5 X"
```

Set up analog output port 1 for disabled trigger response, port 2 for burst trigger control mode, and port 3 for continuous trigger control mode.

```
PRINT #1,"OUTPUT10;P? X"
```

Query DAC488HR for present port selection.

```
PRINT#1,"ENTER10"
```

Read DAC488HR for the query response.

```
INPUT#2,A$
```

Read the response from Driver488.

```
PRINT A$
```

Screen shows P3, the last port selection setting prior to X.

Define Sequence Control Block Ql,n,r

Type: Port Command, Immediate

`Ql,n,r` Define a segment starting at location `l` with a length of `n` (32-buffer limit) samples, with a repeat count of `r` (0-65,535) times. A length of `0` deletes the existing segment. The sequence pointer is advanced to the next location.

`Q?` Returns the Sequence Control Block pointed to by Sequence Pointer (`On`) and advances the sequence pointer, allowing multiple Sequence Control Blocks to be read with the `?` option.

Sequence control blocks control how the data buffer is output in response to triggers when an analog output port is set to Complex Buffer Mode (`A1`). They define arbitrary segments of the data buffer and control the order in which they are output.

The sequence control block is inserted at the present Sequence Pointer location (`On`) in the Sequence Control Table. The segment of the data buffer defined by `Ql,n,r` starts at sample location `l`, has a length of `n` samples, and repeats `r` times.

The limits on `l` (the data buffer starting point) depend on the size of data buffer memory, as shown below. The size of the data buffer for each analog output port may be determined by issuing the `U5` command.

Memory Option	Data Buffer Size	Limits
Standard	8K samples	0-8191
MEMX3	128K samples	0-131071
MEMX4	480K samples	0-491519

Parameter `n`, the length of the defined segment, must be at least 32 samples long, and cannot exceed the physical end of the buffer. No protection is provided against defining segments that use memory not written by the user. A length of `0` deletes the Sequence Control Block pointed to by `On`.

The repeat count may range from 0 (repeats indefinitely) to 65,535 times. (1 is effectively no repeat.)

Example:

`PRINT#1,"OUTPUT10;P1 X"`	Select analog output port 1.
`PRINT#1,"OUTPUT10;O0 X"`	Set sequence pointer to 0000.
`PRINT#1,"OUTPUT10;Q0,0,0 X"`	Delete sequence control block 0.
`PRINT#1,"OUTPUT10;O? X"`	Query present sequence pointer.
`PRINT#1,"ENTER10"`	
`INPUT#2,A$`	Read Driver488 for response.
`PRINT A$`	Screen shows 00000. Note that the sequence pointer did not increment.
`PRINT#1,"OUTPUT10;Q0,100,2 X"`	Define a sequence control block starting at data buffer location 000000, 100 samples in length, to repeat two times.
`PRINT#1,"OUTPUT10;O? X"`	Query present sequence pointer.
`PRINT#1,"ENTER10"`	
`INPUT#2,A$`	Read Driver488 for response.
`PRINT A$`	Screen shows 00001. Note that the sequence pointer incremented.

Range Rn

Type: Port Command, Deferred

R0	Select analog output port ground range (analog output pins are shorted together) (default)
R1	Select analog output port 1 volt range, bipolar
R2	Select analog output port 2 volt range, bipolar
R3	Select analog output port 5 volt range, bipolar
R4	Select analog output port 10 volt range, bipolar
R5	Select analog output port 1 volt range, unipolar
R6	Select analog output port 2 volt range, unipolar
R7	Select analog output port 5 volt range, unipolar
R8	Select analog output port 10 volt range, unipolar
R?	Returns voltage range selection for selected analog output port (selected with P command)

The range command specifies which voltage range is used on an analog output port, and whether the analog output is bipolar or unipolar.

When changing a range, the DAC488HR will momentarily short the analog high and low outputs. The output voltage will then be set to zero. After which, the short is removed. The DAC488HR ground range deliberately shorts the analog high and low connections on the analog output port connector together, guaranteeing 0 volts at the output. When initially powered up, each analog output port is forced to this state. Only after the power-up tests are completed without error are the DACs for each analog output port programmed for the default range and output voltage, and the short released from the analog output (short may be driven with no more than 0.5 amps, and has a maximum resistance of 200 milliohms).

The bit resolution and output voltage range for each setting is shown below.

RANGE	RESOLUTION	
	Unipolar	Bipolar
1 V	15.3µV	30.5µV
2 V	30.5µV	61µV
5 V	76.3µV	152.6µV
10 V	152.6µV	305.2µV

Example:

`PRINT#1,"OUTPUT10;P1 R3 X"`	Set analog output port 1 range to ±5 volt, bipolar.
`PRINT#1,"OUTPUT10;P1 R? X"`	Query range for Port 1.
`PRINT#1,"ENTER10"`	Read DAC488HR for query response.
`INPUT#2,A$`	Read Driver488 for response.
`PRINT A$`	display shows R3.

Save/Restore Sn

Type: System Command, Deferred

S0	Restore setup stored in NVRAM (default).
S1	Save the existing command settings as the power-on default setup in NVRAM.
S2	Restore calibration settings from NVRAM.
S3	Save the present calibration settings in NVRAM.
S4	Restore factory default power-on setups from NVRAM.
S?	Return the last Sn command executed

Note: Issuing a S3 command when the calibration enable switch is set to disable causes a Calibration Enable (E008) error.

The Save/Restore command saves or restores a default setting for each command. It is also used save or restore the calibration constants. The save/restore command may be used to configure the DAC488HR for an application automatically upon power-up. It is also used to save the calibration constants for all analog output ports during DAC488HR calibration procedure.

All calibration constants and a power-on default setting for each command are saved in internal non-volatile RAM (NVRAM). Once calibration constants are saved in NVRAM, they are used whenever the DAC488HR is commanded to output a voltage. The command options saved in NVRAM are the default settings when the unit is powered on.

After calibration, the S3 command saves the new calibration constants to NVRAM.

The S4 command restores power-on settings to the factory default values in NVRAM. These changes will not take place until either an X command is received or until power is cycled. The factory default values are:

A0	Linear buffer mode
D0	Digital outputs off
F0	Signed decimal format
F4	Low byte first
G3	200KHz, synchronous clock
I2	Divide-by-two
K0	Infinite repeat count
L0	Buffer location 0

M0	Clear SRQ mask
N0	Clear event mask
P1	Port 1
R0	Output grounded
T0	No triggering
V0	Output 0 volts
Y1	1 millisecond interval timing
Z1	1 update clock trigger out delay

Information saved in non-volatile RAM includes most of the DAC488HR states. Because the data buffer and sequence control table for each analog output port are volatile (i.e., their contents are erased when power is removed), states of analog output ports that depend on having these present and valid cannot be saved. Instead they are defaulted to known conditions: C0 K1 L000000 O0000 T0.

Information is saved for the commands An Dn Fn Gn Mn Nn Yn and Zn. Port information is saved for Rn and Vv for each analog output port separately.

The digital output port can be set up with an initial output bit pattern, and each analog output port can be configured for a specific range and voltage output value to be set at power-up. This must be handled with some care, however. Both the digital output bits and the analog output voltages are designed to power-up in a guaranteed known state. For the digital outputs, this known state is logic low for TTL compatible outputs, and off for relay compatible outputs. For the analog output voltages, this is a short at the output between high and low. After DAC488HR has run through its power-up self-tests, the digital output port is set to the value stored in NVRAM, and each analog output port, starting with port 1 and proceeding to port 4, is set to the analog output value stored in NVRAM.

Example:

`PRINT#1,"OUTPUT10;S1 X"`	Save existing settings as the power-on default.
`PRINT#1,"OUTPUT10,S? X"`	Query the last S command setting.
`PRINT#1,"ENTER10"`	Read DAC488HR for the query response.
`INPUT#2,A$`	Read the response from Driver488.
`PRINT A$`	Screen shows S1.
`PRINT#1,"OUTPUT10;S0 X"`	Restore saved settings.

`PRINT#1,"OUTPUT10,S? X"`	Query the last `S` command setting.
`PRINT#1,"ENTER10"`	Read DAC488HR for the query response.
`INPUT#2,A$`	Read the response from Driver488.
`PRINT A$`	Screen shows `S0`.

Trigger Source Tn

Type: System Command, Deferred

T0	Disable triggering (default).
T1	GET.
T2	Trigger on TTL rising edge.
T3	Trigger on TTL falling edge.
T4	Trigger on TTL either edge.
T5	Trigger on command trigger (@).
T6	Trigger on interval timer.
T7	Slave Trigger
T?	Return present trigger setting.

The Trigger Source command chooses which trigger source is used for all DAC488HR analog output ports. The power-up default is T0, trigger disabled.

When Trigger Source is set to GET, a Group Execute Trigger on the IEEE 488 bus constitutes a trigger. The DAC488HR handles GET differently than other triggers, however. If a GET is received while the DAC488HR is parsing and decoding a command line, the IEEE 488 bus is held off with an NDAC holdoff until the entire command line is processed up to the execute (X) command. The NDAC holdoff is then released, and a trigger generated. GET is typically used to synchronize DAC488HR with other devices on the IEEE 488 bus. If NDAC holdoff on GET is undesirable because of synchronization issues with other equipment on the bus, follow the commands to set up the DAC488HR with a loop waiting for the Ready bit in the SPoll register before issuing GET to the IEEE 488 bus.

When trigger source is set to interval timer, the Interval Timer command (Yn) controls the time between triggers.

When the trigger source is the @ command, the trigger is not executed until the execute command (X) is encountered in the command line, and a trigger is not generated until the entire command line is processed.

An external TTL signal may also be used to generate triggers, on rising edges, falling edges, or both rising and falling edges. The external TTL trigger signal must have a minimum width of 200 ns, high or low.

The slave trigger can be used only when using several DAC488HRs in a master and slave configuration. The purpose of this mode is to synchronize multiple channels on multiple DACHR's. This command sets the update source to external (G4) and the divisor to 1 (I1). Refer to Section 4.6 for information on synchronizing multiple DAC488HR's.

Example:

```
PRINT#1,"OUTPUT10;P1 C1 R2 T5 X"
```

Select analog output port 1, select stepped trigger control mode, select 2 volt bipolar range, select command trigger source.

```
PRINT#1,"OUTPUT10;@ X"
```

Trigger the DAC488HR.
The DAC488HR outputs the first value in analog output port 1's data buffer.

```
PRINT#1,"OUTPUT10;@ X"
```

Trigger the DAC488HR.
The DAC488HR outputs the second value in analog output port 1's data buffer.

Status Un

Type: System Command, Immediate

U0	Returns Event Status Register (ESR) (default).
U1	Returns status byte register (STB).
U2	Reads all system settings of instrument in the form DnFnFmGnInMnNnTnYnZn.
U3	Returns all installed analog output port settings in the form P1AnCnKnRnVv,P2AnCnKnRnVv...
U4	Returns a three-digit decimal number representing the present state of the eight digital input lines.
U5	Returns all analog output port data buffer sizes in the form xxxxxx,xxxxxx,000000,000000, where 000000 indicates port not installed.
U6	Returns all analog output port sequence control table sizes in the form xxxx,xxxx,0000,0000 where 0000 indicates port not installed.
U7	Returns entire data buffer for selected analog output port (in binary format).
U9	Returns IOtech DAC488HR/2,0,v.r or IOtech DAC488HR/4,0,v.r where v is the version and r is the revision of the firmware.
U?	Returns the last U command executed.

U0 returns the Event Status Register (see Section 5.7 for details on the serial poll model). The returns for U0 are:

Return	Error	Description
004	Query Error	Set when an attempt is made to read data from the output queue when no data are present or data in the output queue were lost. Data may be lost when the output queue buffer overflows (for example, issuing multiple commands to return data without ever reading them).
008	Device Dependent Error	Set when a conflict in programmed parameters is detected. This is also referred to as a conflict error.
016	Execution Error	Set when a parameter exceeds valid limits for a particular command. This is also referred to as Invalid Device Dependent Command Option (IDDCO) error.
032	Command Error	Set when an illegal command is sent to the DAC488HR. This is also referred to as Invalid Device Dependent Command (IDDC) error.

064	Unused	This bit always returns 0 (reserved for future use).
128	Power On	This bit is set on initial power-up of the DAC488HR.

U1 returns the Status Byte Register. This is a copy of the same byte returned in response to an SPoll from the IEEE 488 bus. U1 returns:

001	Triggered	Set when the DAC488HR has sensed a valid trigger. It is cleared whenever a new Trigger Source command (Tn) is executed or when the End of Trigger Sequence bit is set. This bit can only be used to catch the occurrence of the first valid trigger.
002	End of Trigger Sequence	Set when all analog output ports that have been set to act upon a trigger by the Trigger Control Mode command (C1, C2, C3 or C4) have been triggered and have completed their entire trigger sequence. This bit is cleared whenever any port has been reconfigured to the C1, C2, C3 or C4 trigger control modes or a a new Trigger Source command (Tn) is executed.
004	Ready	Set when the DAC488HR is ready to process another command. It is cleared when the DAC488HR is processing a command line. This bit should be examined with a serial poll prior to issuing a new command line. This allows any detected errors to be traced to the specific command line containing the error. If all the setup information for a specific DAC488HR operation is included in one line, this bit also indicates when all processing is done and the X command is completed. This ensures that the DAC488HR is done processing all state changes before initiating any further activity. NOTE: While processing the U1 command, the DAC488HR will always have the Ready bit unasserted. Use the SPOLL command to get a more accurate indication of the status of the Ready bit.
016	Message Available	Set when the output queue is not empty. It is cleared when the output queue is empty. This bit reflects whether any command responses are still in the output queue.
032	Event Status register Bit (ESB)	Reflects the logical OR of all the bits in the Event Status Register (ESR) ANDed with their equivalent enable bits in the Event Status Enable (ESE) register. If this bit is set, at least one bit in the ESR is set and has its corresponding enable bit in the ESE set. The status command U0 can be issued to read the ESR. See the following for more information on ESR and ESE.
064	Service Request Bit (SRQ)	Set when the DAC488HR is requesting service. It is cleared when an SPoll is performed.
128	DMA Underrun	Set when the DAC488HR has at least one analog output port running in continuous output mode, and the rate of data transfer from the IEEE 488 bus is slower than the rate of output.

U2 returns all the system command settings for the DAC488HR. This is all the information necessary to reconfigure the DAC488HR system commands to the same state as the existing state when this command was executed. The response to this command is returned in the format:

DnFnFmGnInMnNnTnYnZn

U3 returns all installed analog output port settings for the DAC488HR. No responses are generated for analog ports which are not installed. This is all the information necessary to reconfigure all DAC488HR analog output ports to the same state as the existing state when this command was executed. Care must be taken when using this and the U2 response in conjunction. To reconfigure DAC488HR to the same state, first issue the response to the U3 command to set all the analog output ports to the correct state. The U2 response should be issued to the DAC488HR after the U3 response. This order ensures that the analog output ports are set up for correct trigger response before any triggers are enabled.

U4 returns the digital input value. This is always a fixed length ASCII decimal string in the format nnn, where nnn ranges from 000 (all inputs low) to 255 (all inputs high).

U5 returns all analog output port data buffer sizes. This command can be used to determine exactly what memory configuration each analog output port has. The string returned is of the format nnnnnn,nnnnnn,nnnnnn,nnnnnn. The first number relates to analog output port 1, the second to analog output port 2, etc. The number returned is the highest data buffer location allowed (the maximum setting allowed for the Ln command), and is one less than the actual data buffer size because the data buffer starts at location 0. If an analog output port is not installed, 000000 is returned. Thus, DAC488HR/2 always returns nnnnnn,nnnnnn,000000,000000.

U6 returns the Sequence Control Table sizes in a format similar to that of U5. The string returned is in the format nnnn,nnnn,nnnn,nnnn. The first number relates to analog output port 1, the second to analog output port 2, etc. The number returned is the number of Sequence Control Blocks allowed (the maximum setting allowed for the On command). If an analog output port is not installed, 000000 is returned. Thus, DAC488HR/2 always returns nnnn,nnnn,0000,0000.

The values returned for U5 and U6 depend on the memory option installed for each analog output port, as shown in the table below.

Memory Option	U5 Return	U6 Return
Standard	008192	0128
MEMX3	131072	2048
MEMX4	491520	2048

U7 is provided to allow the backup of the entire data buffer in binary format. When this command is issued, the entire data buffer for the current port will be placed into the output queue. The first 9 bytes of this response has the form: B#6nnnnnn

Where n is the total size (in bytes) of the transfer. (Note that this number represents bytes not samples.) This number will be dependent upon the memory module of the daughterboard selected.

The response generated from U7 command may be pre-empted by the Device Clear command. This will clear the input buffer and output queues and will clear any binary data that has not yet been read.

U9 returns an ASCII string identifying the product and the revision and version of the firmware installed in the product.

Example:

`PRINT#1,"OUTPUT10;U9 X"`	Ask for identifier string from DAC488HR/2.
`PRINT#1,"ENTER10"`	Read DAC488HR/2 for the query response.
`INPUT#2,A$`	Read the response from Driver488.
`PRINT A$`	Screen shows `IOtech DAC488HR/2,0,1.0`

Voltage Output Vval

Type: Port Command, Deferred

`Vval`	Set output to `v` (format set by `Fn`)
`V?`	Returns present output value for the selected analog output port in the format specified by Format (`Fn`) command.

The Voltage Output command sets a voltage value on the analog output of the selected analog output port. A voltage may be specified in volts, decimal bits or hexadecimal bits, depending on the setting of the Format (`Fn`) command. Voltages cannot be specified for direct output in binary.

If the format command is set to `F0` or `F1`, there is no need to suppress or add the leading + sign for positive values. The DAC488HR is forgiving in both cases on input. When querying voltage values, however, `F0` always returns a + sign for positive values and `F1` always returns a leading space for positive values.

Voltages are specified as `V#.#`. Thus, all of the following command strings cause +5.6 volts to be output on the selected analog output port.

```
V 5.6 X
V+5.6 X
V 5.6 X
```

If the format command is set to `F2`, all voltages are specified in decimal bits. A leading + or space is optional. When querying voltage values, neither a + nor a leading space is returned for positive values.

If the format command is set to `F3`, voltages are specified in hexadecimal bits. Signed 16-bit integer format is assumed, so leading + or – signs are not accepted. 0 to 7FFF are interpreted as 0 to 32,767 decimal. FFFF to 8000 are interpreted as –1 to –32,768 decimal when in bipolar mode. In unipolar mode, 0 to FFFF are interpreted as 0 to 65,535 bits.

Care must be taken when specifying options in hexadecimal. Because hexadecimal numbers may contain valid command letters, the user must be sure that hexadecimal values are separated properly from commands immediately following them. In this case, white space characters, which consist of all ASCII values of 32 and below and include the space, tab, new-line and carriage-return characters, are significant. Thus, `B0FF F2 X` is valid, but `B0FFF2 X` will not result in the same action. In the second example, format is not set to decimal as intended, and the `B` value is set to `FFF2` hexadecimal.

When programming in decimal or hexadecimal bits, the voltage output depends on the range selected. To determine the actual output voltage, the number of bits must be multiplied by the resolution shown in the table below.

RANGE	RESOLUTION	
	Unipolar	Bipolar
1 V	15.3µV	30.5µV
2 V	30.5µV	61µV
5 V	76.3µV	152.6µV
10 V	152.6µV	305.2µV

Example:

```
PRINT#1,"OUTPUT10;P1"              Select analog output port 1.

PRINT#1,"OUTPUT10;F2 R2 X"         Set format to decimal bits, set range to ±2 volts.

PRINT#1,"OUTPUT10;V1000 X"

                                   Set output to 1000 counts (0.061 volts)

PRINT#1,"OUTPUT10;V1500 X"         Set output to 1500 counts (0.0915 volts).

PRINT#1,"OUTPUT10;F3 X"            Set format to hexadecimal.

PRINT#1,"OUTPUT10;VFF X"           Set output to 255 counts (0.0156 volts).

PRINT#1,"OUTPUT10;F1 X"            Set format to signed volts.

PRINT#1,"OUTPUT10;V-2.000 X"       Set output to -2 volts.

PRINT#1,"OUTPUT10;V? X"            Query the present voltage value.

PRINT#1,"ENTER10"                  Query response from DAC488HR.

INPUT#2,A$                         Read the response from Driver488.
```

```
PRINT A$
```

Screen shows `V-1.99994`
This represents the closest voltage that can be achieved with a 16-bit digital to to analog converter.

Waveform Load Ww,l,max,min,d

Type: Port Command, Deferred

`Ww,l,max,min,d`

Write pre-defined waveform `w` (0 - 2), of length `l` (32 - buffer limit), with maximum value `max` (±100%), minimum value `min` (±100%), and symmetry `d` (0 - 100%). Starting location is determined by `Ln`.

`W0` Sine wave.

`W1` Triangle wave.

`W2` Square wave.

`W?` Returns present waveform load configuration.

This command automatically writes one of three pre-defined waveforms into an analog output port's data buffer. Note that since the `max` and `min` values are percentages, this command is independent of the specified range.

When a pre-defined function is used, a value must be specified for the data buffer location pointer and the DAC488HR commanded to write one of the pre-defined functions. Using the pre-defined function command, one cycle of a sine wave, triangle wave, or square wave can be written automatically into an analog output port's data buffer.

Typical waveform generation rates are described in the chart below.

WAVEFORM	RATE
Sine	540 points/second
Triangle	1700 points/second
Square	5400 points/second

Note long waveforms may take several seconds to be generated.

The command for the pre-defined functions requires that parameters for the length (# of samples), maximum and minimum values (as a percentage of full scale), and the symmetry (as a percentage of length) of the waveform be specified. When data loading begins, DAC488HR automatically transfers one cycle into the buffer starting at the present location of the data buffer location pointer according to these parameters. A sequence control block can then be used to force this cycle to repeat as many times as desired by the user.

Refer to Figure 5.13 to view pre-defined waveforms.

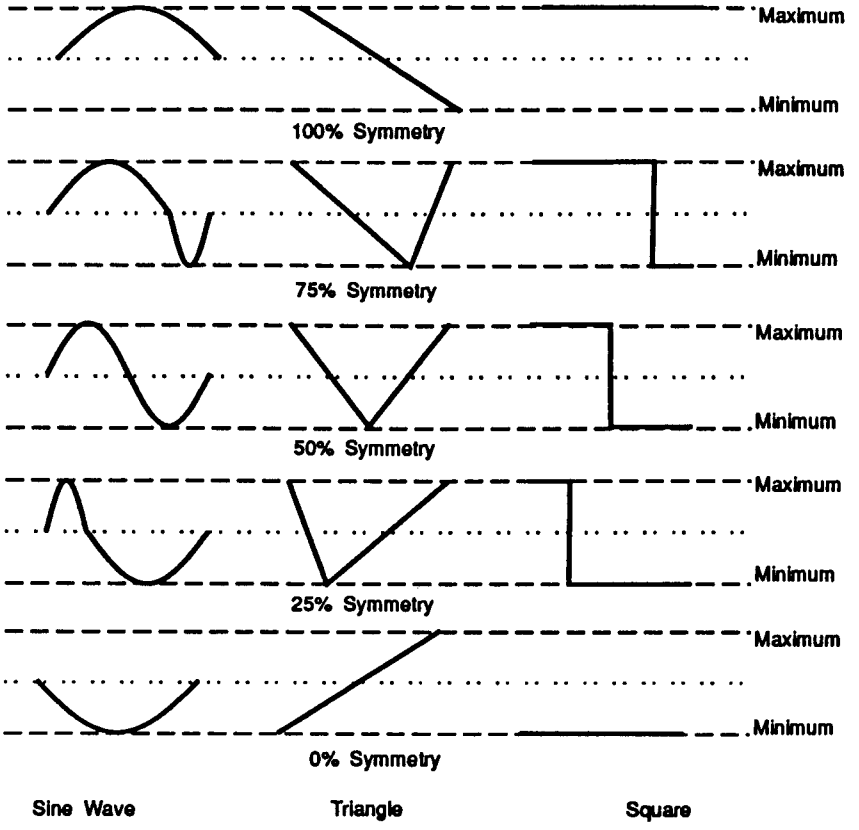


Figure 5.13: Pre-Defined Functions

Example:

`PRINT#1,"OUTPUT10;*R X"`	Return DAC488HR to power-on state (clears all data buffers and sequence control blocks in each analog output port; L set to 0000000, O set to 0000).
`PRINT#1,"OUTPUT10;P1 A1 X"`	Select analog output port 1, set buffer mode to complex.
`PRINT#1,"OUTPUT10;R4 X"`	Set range to 10 V bipolar.

```
PRINT#1,"OUTPUT10;W0,1000,100,-100,50 X"
```

Load a sine wave of 1000 samples, +10 V maximum, −10 V minimum, 50% symmetry starting at data buffer location 000000.

```
PRINT#1,"OUTPUT10;L? X"
```

Query present data buffer location pointer.

```
PRINT#1,"ENTER10"

INPUT#2,A$

PRINT A$
```

Screen shows L001000, the next data buffer location to be loaded.

```
PRINT#1,"OUTPUT10;W? X"
```

Query present waveform load configuration.

```
PRINT#1,"ENTER10"

INPUT#2,A$

PRINT A$
```

Screen shows W0,001000,0100,-100,050, the present waveform load configuration.

```
PRINT#1,"OUTPUT10;W1,1000,50,-50,50 X"
```

Load a triangle wave of 1000 samples +5 V maximum, −5 V minimum, 50% symmetry, starting at data buffer location 001000.

```
PRINT#1,"OUTPUT10;Q0,1000,5 X"
```

Define a sequence control block at sequence pointer 0 (reset default), 1000 samples long to be repeated five times. This causes the sine wave loaded above to be repeated for five cycles.

```
PRINT#1,"OUTPUT10;Q1000,1000,3 X"
```

Define a sequence control block at sequence pointer 1 (0 was automatically incremented after step above), data buffer location 1000, 1000 samples long, repeated three times. This causes the triangle wave loaded above to be repeated for three cycles.

```
PRINT#1,"OUTPUT10;C4X"
```

Start outputting the waveform on port 1.

Execute X

Type: System Command

X Execute preceding command string

Most commands are interpreted and processed as they are received but are not executed until the Execute (X) command is issued. Commands sent without an X are stored in an internal buffer and are not executed until an X is received.

While a command line is being interpreted, the front panel LEDs will not be updated. These LEDs will only be updated when the DAC488HR is in a ready state. In order to determine if the DAC488HR is in a ready state, perform a Serial Poll for the ready bit (4).

If multiple system commands are used in the same string, each use of the command must be followed by the Execute (X) command. The Port (P) command and the port commands do not have to be followed by X when used in the same string. Any number of Execute commands may be inserted into the same command string.

The port command (P) selects which analog output port all port-specific commands are directed to. This command changes the P register as soon as the instruction is found in a command string.

The Buffer Data command (B) can accept large numbers of comma-delimited data points for writing into an analog output port's data buffer, and the sheer size of the data buffer (up to 1 Megabyte), prohibits loading a copy of the analog output port's data memory and deferring all changes until an X command. All data points written via a B command are immediately acted upon as the data points are encountered in the command string.

Example:

`PRINT#1,"CLEAR10"`	Reset the DAC488HR.
`PRINT#1,"OUTPUT10;P1 V4"`	Send V4 to the DAC488HR command input buffer.
`PRINT#1,"OUTPUT10;X"`	Instruct the DAC488HR to execute its command input buffer. The DAC488HR outputs 4 volts on analog output port 1 upon receipt of the X.

To clear the SRQ mask and then set it for SRQ on trigger and SRQ on end of trigger sequence:

```
PRINT#1,"OUTPUT10;M000 X M003 X"
```

To program port 1 for 3 volts and port 2 for 5 volts using one command string:

```
PRINT#1,"OUTPUT10;P1 V3 P2 V5 X"
```

Interval Timer Yn

Type: System Command, Deferred

Yn	Set interval timer to n (1 – 65,535) milliseconds. Default is Y1.
Y?	Return present interval timer setting.

The interval timer is used as a trigger source when Trigger Source is set to T6.

Example:

`PRINT#1,"OUTPUT10;Y5 X"`	Set interval timer to 5 milliseconds.
`PRINT#1,"OUTPUT10,Y? X"`	Query the interval timer setting.
`PRINT#1,"ENTER10"`	Read DAC488HR for the query response.
`INPUT#2,A$`	Read the response from Driver488.
`PRINT A$`	Screen shows Y5.

Trigger Delay Zn

Type: System Command, Deferred

`Zn` Set trigger delay to n (1 – 65,535) update clocks. Default is `Z1`.

`Z?` Return present trigger delay setting

The delayed trigger output is used with applications that require some delay or setup time after the application of a signal from the DAC488HR before meaningful data can be measured. The delayed trigger output signal is common to all analog output ports. The time delay is specified in update clocks to any value from 1 to 65,535 update clocks. This sets the time delay between when any analog output port reacts to a trigger and when the delayed trigger output goes true. At a 100 kHz update rate, this yields 10µsec increments from 10µsec to 0.65 seconds.

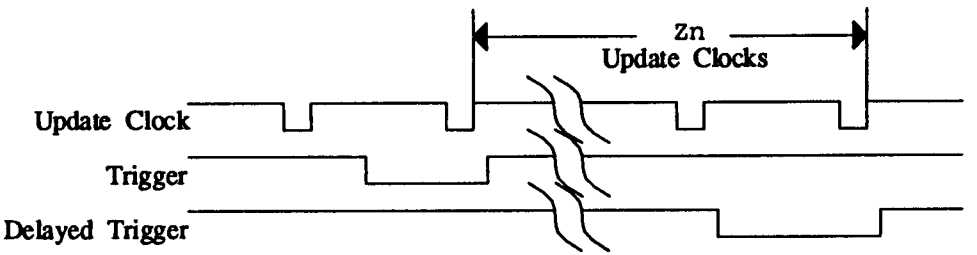


Figure 5.14: Delayed Trigger Timing Diagram

The delay is retriggerable, so if another valid trigger occurs before the trigger delay time has elapsed, the time delay is a full trigger delay after the second valid trigger.

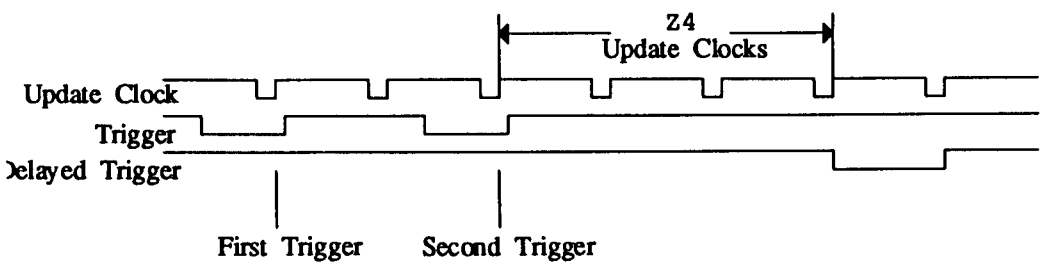


Figure 5.15: Retriggered Delayed Trigger Timing Diagram

Example:

`PRINT#1,"OUTPUT10;Z20 X"`	Set trigger delay to 20 update clocks.
`PRINT#1,"OUTPUT10,Z? X"`	Query the trigger delay setting.
`PRINT#1,"ENTER10"`	Read DAC488HR for the query response.
`INPUT#2,A$`	Read the response from Driver488.
`PRINT A$`	Screen shows `Z20`.

DAC488HR Calibration

6.1 Calibration Overview

Each analog output port of the DAC488HR should be calibrated every 12 months. Calibration of each DAC488HR analog output port involves computing the correct offset and gain calibration constants for each range (as selected by the Range command). These constants are then stored in the DAC488HR non-volatile RAM for subsequent use. When an analog output port is set to a specific range, the calibration constants are sent to internal correction digital to analog converters (DACs) that set the offset and gain for the given range.

6.2 Calibration Theory

An ideal digital to analog converter (DAC) translates digital input values to volts as shown in Figure 6.1.

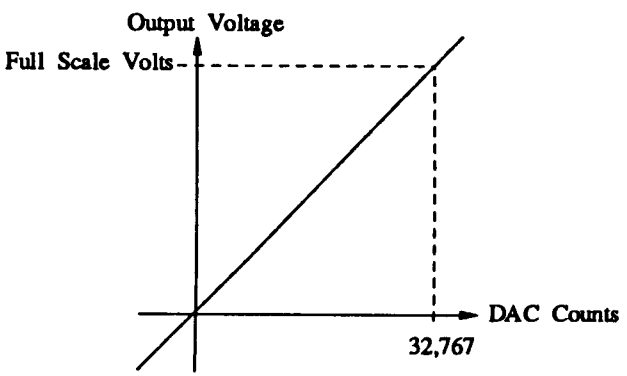


Figure 6.1: Ideal D/A Converter Response

This is a straight line that crosses the origin (0 volts output for 0 counts digital input) and has a slope of:

$$\frac{32{,}767}{\text{full-scale volts}}$$ (32,767 counts input for full-scale output for the bipolar case)

or:

$$\frac{65{,}535}{\text{full-scale volts}}$$ (65,535 counts input for full-scale output for the unipolar case).

A real DAC has two sources of error that cause it to deviate from this ideal curve as shown in Figure 6.2.

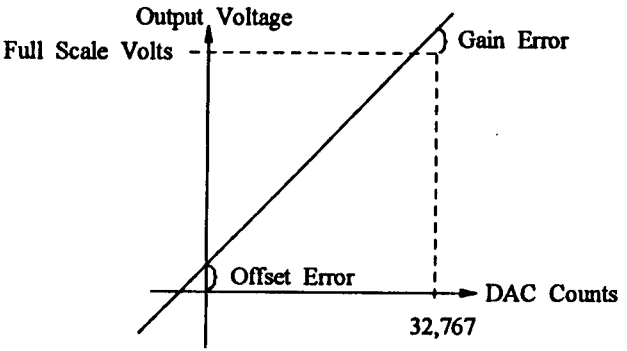


Figure 6.2: Deviation from Ideal Response

Offsets introduced by the analog circuitry following the DAC cause a small voltage to be added to or subtracted from the DAC's output. This is a constant that can be directly measured when the DAC488HR is commanded to output 0 volts. Instead of 0 volts, the output is some small plus or minus voltage. This is the offset.

Gain errors are introduced by small variations in the gain stages following the DACs and in the voltage reference. These differences actually cause a slight change in the slope of the line representing the response of the DAC488HR to digital inputs. This error has the most effect and is most noticeable at full scale.

Two 12-bit correction DACs are used on each analog output port to correct these errors in hardware, as shown in Figure 6.3. Each analog output port has correction values for offset and gain correction for each range. These correction values are stored in non-volatile RAM in the system unit. Whenever an analog output port is commanded to change range, these calibration constants are sent to the correction DACs.

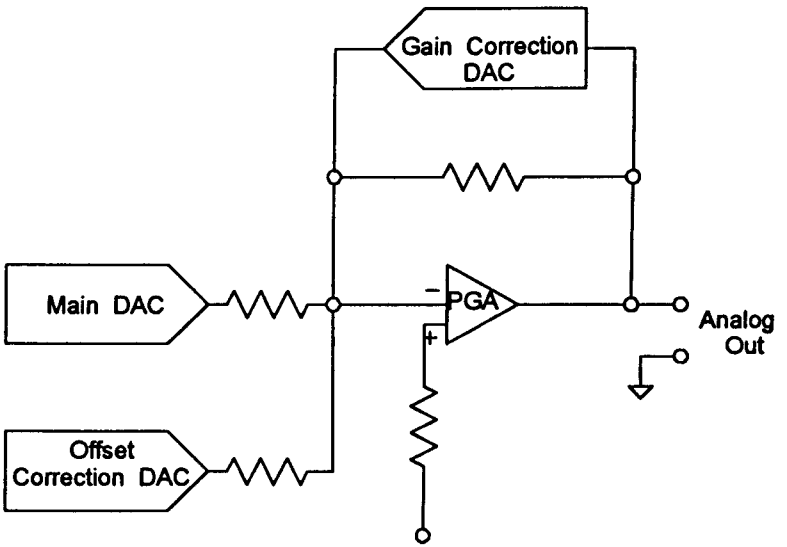


Figure 6.3: Analog Output DAC Circuitry

The DAC488HR stores calibration constants in non-volatile RAM for permanent storage. These calibration constants are read out of non-volatile RAM at power-on. Whenever a range command is issued to a selected analog output port, the offset and gain constants are selected for that port and range. These constants are then sent to the hardware correction circuitry along with the range information. The calibration constants for the selected port and range may be read by using the `H?` and `J?` commands. The calibration constants may also be manipulated by using the `Hn` and `Jn` commands. Any changes made to the offset and gain constants will be effective until any of the following conditions are met:

1. The unit is reset using the `*R` command.
2. The factory calibration constants are restored using the `S2` command.
3. The unit is powered down and back up.

WARNING

The calibration constants may be stored to non-volatile RAM using the `S3` command. However, this overwrites the factory calibration constants.

Any change to the factory calibration constants by means other than this calibration procedure may cause the DAC488HR to operate outside published specifications.

6.3 Calibration Procedure

The calibration procedure described below shows how to calibrate one analog output port on the DAC488HR. It is the same procedure used in the QuickBASIC program `Calibr8.bas`, included on the sample program disk (part #166-0600) provided with the DAC488HR. This program calibrates the DAC488HR using a Keithley 199 System DMM/Scanner and IOtech's Driver488. The source code for this calibration program is listed in Appendix 6. This program may be modified to use other IEEE 488 or RS-232C programmable multimeters to facilitate calibration.

This procedure should be repeated for each installed analog output port to be calibrated. All of the analog output ports should be calibrated at the same time.

6.3.1 Equipment Required

The equipment needed to calibrate DAC488HR with the QuickBASIC example program is:

- Keithley Instruments Model 199 System DMM/Scanner.
- An IBM PC or compatible.
- IEEE 488 cables to connect the 199 and the DAC488HR to the PC.
- Calibration cable as shown in Figure 6.4.

6.3.2 Calibration Setup

The DAC488HR and the Keithley 199 must be connected to the IEEE 488 controller with IEEE 488 cables. Wire the calibration cable to connect the DAC488HR and the Keithley 199 together. The wiring for this cable is shown in Figure 6.4. If calibrating a DAC488HR/2, the wiring for ports 3 and 4 can be omitted.

Calibration constants must be programmed with trigger detection disabled (Trigger Control Mode set to `C0`). If trigger detection is not disabled, any currently running operation will be aborted.

The switch labeled CAL ENABLE on the rear panel of the DAC488HR must be in the enabled position (1) to allow the new calibration constants to be saved in non-volatile RAM (see Section 2 for more information on setting switches). If calibration is not enabled, the calibration constants can be changed, but will not be saved to non-volatile RAM. In this case, if the DAC488HR is turned off, the calibration constants revert to their last saved values when power is returned or `*R` used. After calibration is complete, return the switch to the disabled position to prevent accidental corruption of the calibration constants. If this change is not made, a Calibration Enable error (`E008`) occurs when trying to save the new calibration constants. After completing calibration, change this switch back to disable (0).

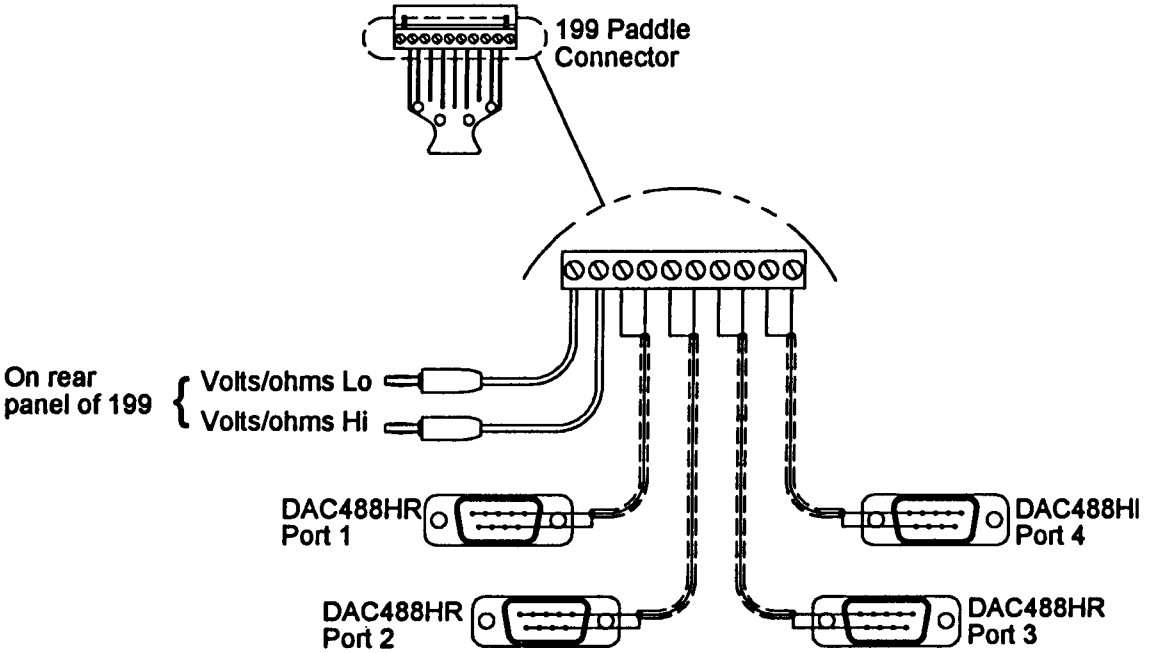


Figure 6.4: Calibration Cable Wiring Diagram

6.3.3 Calibration Method

The actual calibration is a simple operation, but must be completed in a specific order. At zero volts, gain error is small enough to be negligible and offset error dominates the reading. At full-scale volts, the reading error is the combination of both offset and gain errors. For these reasons, the calibration process first measures the offset error with zero volts produced from the analog output port and calculates the offset compensation. Then the gain error is measured, with full-scale voltage produced from the analog output port and the appropriate gain compensation calculated. This two-step calibration procedure is repeated for every range of every installed analog output port. When calibration is completed, the calibration constants are stored to non-volatile RAM for permanent storage.

Calibration of an analog output port involves two calibration constants. Offset calibration sets the calibration constant that adjusts offset errors at the output of the analog output port. Gain calibration sets the calibration constant that adjusts gain errors at the output of the analog output port.

Both of these values should nominally be half of the adjustment range (2048). This value should be used as a starting point when calibrating the DAC488HR. Both of these constants have an adjustment range of 0 to 4095. When either of these commands is executed, the existing programmed output voltage appears at the analog output.

To check the existing Offset Calibration and Gain Calibration settings, use the Offset or Gain query commands. They return the present offset calibration constant or gain correction calibration constant for the selected range and analog output port.

To understand the following procedure, the implementation and effects of the offset and gain calibration constants must be understood. The offset and gain calibration constants are 12-bit unsigned values used to adjust offset and gain errors out of the analog output circuitry.

The following four rules describe the effect of the offset and gain calibration constants:

1. With a voltage of 0 and an offset calibration constant of zero, the voltage at the analog output port is less than zero volts.
2. With a voltage of 0 and an offset calibration constant of 4095, the voltage at the analog output port is greater than zero volts.
3. With a voltage of + full scale, a calibrated offset calibration constant, and a gain calibration constant of zero, the voltage at the analog output port is greater than + full scale.
4. With a voltage of + full scale, a calibrated offset calibration constant, and a gain calibration constant of 4095, the voltage seen at the analog output port is less than + full scale.

These rules, combined with the bit values associated with the offset and calibration constants, provides the basis for the calibration procedure.

6.3.3.1 Offset Calibration

1. Select range 1 on the analog output port.
2. Set the voltage output to 0 volts.
3. Set offset correction to 0 by using the `H0` command.
4. Set gain correction to 0 by using the `J0` command.
5. Select the lowest range available on the multimeter. The reading should be less than zero volts at this time.
6. A running offset value must be maintained until offset calibration is complete. At this time, the running offset value is zero.
7. Starting with the most significant bit of the offset calibration constant, issue the `Hn` command to assert this bit. Also add the bit value to the running offset value. In this case, the value would 2048.
8. Read the voltage seen by the multimeter.

   If the voltage is greater than 0 volts +½ LSB, turn the selected bit off by subtracting the bit value from the running offset value.
9. Set the next most significant bit by adding its bit value to the running offset value.
10. Issue the `Hn` command using the running offset value.
11. Repeat steps 8 through 10 for each bit in the offset calibration constant (12 total).

12. Once the last bit (bit 1) has been checked, the running offset value is the offset calibration constant for the selected port and range.

Example of Offset Calibration

1. Apply 0 volts output to selected port. Running offset starts at 2048. Set the offset to the running offset value, using `H2048 X`.
2. The meter voltage indicates that the voltage is greater than 0. This offset value is too high. Subtract 2048 from the running offset value. Now select the next bit, which is 1024. Add this value to the running offset value. Set the offset to the running offset value, using `H1024 X`.
3. The meter voltage indicates that the voltage is less than 0. To maintain this value and add to it, do not subtract 1024 from the running offset value. Select the next bit, which is 512. Add this value to the running offset value (1024 + 512 = 1536) and set offset to the running offset value using `H1536 X`.
4. The meter voltage indicates the voltage is less than 0. Again, maintain this value and add to it, using the methods described above. Continue until the desired offset is obtained.

6.3.3.2 Gain Calibration

1. Select range on the analog output port.
2. Set the voltage output to + full-scale volts.
3. Use offset correction attained by the above procedure using `Hn` command.
4. Set gain correction to 0 by using the `J0` command.
5. Select the range needed to read + full-scale volts on the multimeter. The reading should be greater than + full-scale volts at this time.
6. A running gain value must be maintained until gain calibration is complete. At this time, the running gain value will be zero.
7. Starting with the most significant bit of the gain calibration constant, issue the `Jn` command to assert this bit. Also add the bit value to the running gain value. In this case, the value would 2048.
8. Read the voltage seen by the multimeter. If the voltage is less than + full-scale volts – ½ LSB, turn the selected bit off by subtracting the bit value from the running gain value.
9. Set the next most significant bit by adding its bit value to the running gain value.
10. Issue the `Jn` command using the running gain value.
11. Repeat steps 8 through 10 for each bit in the gain calibration constant (12 total).
12. Once the last bit (bit 1) has been checked, the running gain value will be the gain calibration constant for the selected port and range.

Example of Gain Calibration

1. Apply full scale volts to the selected port. Running gain starts at 2048. Set the gain to the running gain value, using `J2048 X`.
2. The meter voltage indicates that the voltage is less than the positive full scale value. This gain value is too low. Subtract 2048 from the running gain value. Now select the next bit, which is 1024. Add this value to the running gain value. Set the gain to the running gain value, using `J1024 X`.
3. The meter voltage indicates that the voltage is more than the positive full scale voltage. To maintain this value and add to it, do not subtract 1024 from the running gain value. Select the next bit, which is 512. Add this value to the running gain value (1024 + 512 = 1536) and set gain to the running gain value using `J1536 X`.
4. The meter voltage indicates the voltage is more than the positive full scale voltage. Again, maintain this value and add to it, using the methods described above. Continue until the desired gain is obtained.

At this point, the offset and gain calibration constants for the selected range and port have been set. Repeat the offset and calibration procedures outlined above for every range of every port. Once all ranges of all ports have been calibrated, the offset and gain calibration constants must be saved to non-volatile RAM using the `S3` command. The calibration switch must be in the enabled position at this time. If it is not, a Calibration Enable Error (`E008`) is generated.

6.4 Sample Calibration Program

```
'DAC488HR Calibration program
'REV 1.0

'First open DRIVER for communications
OPEN "\DEV\IEEEOUT" FOR OUTPUT AS #1
IOCTL #1, "BREAK"
PRINT #1, "RESET"
OPEN "\DEV\IEEEIN" FOR INPUT AS #2
PRINT #1, "FILL ERROR"

'Open text file on drive C for calibration report
OPEN "calrep.txt" FOR OUTPUT AS #3

'Next assign addresses to the devices to be used

dac$ = "05"' Default DAC488HR address
KI$ = "26"' Keithley 199 Address
PRINT #1, "CLEAR"; KI$
```

```
'Now initialize these devices

PRINT #1, "REMOTE 26"
PRINT #1, "OUTPUT"; KI$; ";F0 R0 Z0 P0 S1 T0 B0 I1 G1 M0 K0 Y0 X"
PRINT #1, "OUTPUT"; dac$; ";F0X"

'The following variable is calibration accuracy in percent
accuracy = .02

cal.tries = 3
ver.tries = 5

'Create arrays for cal constants (this is temporary and will be
'removed when calibration commands are implemented.
DIM cal.constants(4, 8, 2) AS INTEGER

'Create calibration accuracy array
DIM cal.acc(8) AS SINGLE

'Create test voltage array
DIM test.volts(8) AS INTEGER
DIM span(8) AS DOUBLE

'Now calculate calibration accuracy for each range using
'calibration accuracy variable and span data contained at the end of
'the program. Also read test voltages at this time

FOR n = 1 TO 8
    READ span#(n)
    cal.acc(n) = span(n) * accuracy * .01
    READ volts
    test.volts(n) = volts
NEXT n

start$ = TIME$

CLS
INPUT "Please enter number of ports to test "; num.ports

tries = cal.tries
```

```
GOSUB cal.loop

final = 1
GOSUB verify.loop
PRINT "Please turn unit off and press any key..."
WHILE INKEY$ = ""
WEND


PRINT "Please wait..."
SLEEP 5
PRINT "Now turn unit back on and press any key..."
WHILE INKEY$ = ""
WEND
PRINT "Waiting for DAC to finish power on..."
SLEEP 5



testpatch:
final = 0
PRINT #1, "OUTPUT"; dac$; ";F0X"
GOSUB verify.loop


CLOSE #2
CLOSE #1


END
cal.loop:
FOR port = 1 TO num.ports
   GOSUB set.channel
   FOR range = 1 TO 8
         PRINT "Calibrating DAC channel "; port; " on range "; range
         GOSUB offset.cal.loop
         GOSUB gain.cal.loop
         cal.constants(port, range, 1) = offset
         cal.constants(port, range, 2) = gain
   NEXT range
NEXT port


PRINT #1, "output"; dac$; ";s3x"
RETURN
```

```
verify.loop:
PRINT "Verifying calibration Data"
failure = 0
tries = ver.tries
FOR port = 1 TO num.ports
   GOSUB set.channel
   PRINT "Port "; port
   IF final THEN
         PRINT #3, "Port "; port
   END IF
   PRINT "Range OffsetV  Offset%  GainV   Gain%"
   IF final THEN
         PRINT #3, "Range OffsetV  Offset%  GainV   Gain%"
   END IF
   port.failure = 0
   FOR range = 1 TO 8
         volts = 0
         offset.fail = 0
         gain.fail = 0
         GOSUB write.data
         GOSUB get.readings

         offset.volts = rdg

         IF ABS(offset.volts) cal.acc(range) THEN
               failure = failure + 1
               port.failure = port.failure + 1
               offset.fail = 1
         END IF

         offset.percent = ABS(0 - offset.volts)
         offset.percent = offset.percent / span#(range)
         offset.percent = offset.percent * 100

         volts = test.volts(range)
         GOSUB write.data
         GOSUB get.readings
         gain.volts = rdg


         IF ABS(gain.volts - test.volts(range)) cal.acc(range) THEN
               failure = failure + 1
               port.failure = port.failure + 1
               gain.fail = 1
         END IF
         gain.percent = ABS(test.volts(range) - gain.volts)
         gain.percent = gain.percent / span#(range)
         gain.percent = gain.percent * 100
```

```
            PRINT USING "#   +##.###### #.###### +##.###### #.######"; ran
            offset.volts; offset.percent; gain.volts; gain.percent;
            IF final THEN
                  PRINT #3, USING "#   +##.###### #.###### +##.###### #.##
                  range; offset.volts; offset.percent; gain.volts; gain.p
            END IF

            IF (offset.fail OR gain.fail) THEN
                  PRINT " Failed!"
                  IF final THEN
                        PRINT #3, " Failed!"
                  END IF

                  ELSE
                  PRINT " Passed!"
                  IF final THEN
                        PRINT #3, " Passed!"
                  END IF
            END IF

            NEXT range
NEXT port
PRINT "Failures = "; failure
IF final THEN
   PRINT #3, "Failures = "; failure
END IF


PRINT "Calibration Complete"
IF final THEN
   PRINT #3, "Calibration Complete"
   PRINT #3, "Calibration constants"
   FOR port = 1 TO 4
         PRINT #3, "Range Offset  Gain   "
         FOR range = 1 TO 8
         PRINT #3, USING "#  ######  ######"; range; cal.constants(port,
         range, 1); cal.constants(port, range, 2)
         PRINT #3, "Port"; port
         NEXT range
   NEXT port

   stop$ = TIME$
```

```
        PRINT "Start = "; start$
        PRINT #3, "Start = "; start$
        PRINT "End = "; stop$
        PRINT #3, "End = "; stop$

        CLOSE #3
    END IF
    RETURN


    offset.cal.loop:
        gain = 0
        offset = 0
        volts = 0

        GOSUB write.data
        GOSUB write.cal
        GOSUB get.readings

        IF rdg -.01 THEN GOTO offset.fail

        offset = 4095
        GOSUB write.cal
        GOSUB get.readings

        IF rdg .01 THEN GOTO offset.fail

    'Now we know that the offset DAC and circuitry is functional, now
    'it's time to calibrate, using the bitwise algorithm.

        offset = 0
        FOR n = 11 TO 0 STEP -1
        offset = offset + (2 ^ n)
                GOSUB write.cal
                GOSUB get.readings
                IF rdg 0# THEN
                        offset = offset - (2 ^ n)
                END IF
        NEXT n
        GOSUB write.cal
        RETURN
    offset.fail:
        PRINT "Port "; port; " Range "; range; "offset failure! ";
```

```
    PRINT rdg; " "; offset; " "; gain
    RETURN

gain.cal.loop:
    volts = test.volts(range)
    GOSUB write.data
    gain = 0
    GOSUB write.cal
    GOSUB get.readings
    IF rdg volts THEN GOTO gain.fail
    gain = 4095
    GOSUB write.cal
    GOSUB get.readings
    IF rdg volts THEN GOTO gain.fail

'Now we know that the gain DAC and associated circuitry is
' functional, so we can attempt to calibrate

    gain = 0

FOR n = 11 TO 0 STEP -1
    gain = gain + (2 ^ n)
    GOSUB write.cal
    GOSUB get.readings
    IF rdg volts THEN
          gain = gain - (2 ^ n)
    END IF
NEXT n
    GOSUB write.cal
RETURN


gain.fail:
    PRINT "Port "; port; " Range "; range; " gain failure! ";
    PRINT rdg; " "; offset; " "; gain
RETURN


get.readings:
    avg = 0

    PRINT #1, "ENTER"; KI$
    INPUT #2, dummy
```

```
    FOR q = 1 TO tries
          PRINT #1, "ENTER"; KI$
          INPUT #2, rdg
          avg = avg + rdg
    NEXT q

    avg = avg / tries
    rdg = avg

RETURN

write.data:
    GOSUB wait.ready
    PRINT #1, "OUTPUT "; dac$; ";P"; port; " R"; range; " V"; volts; " X"
    GOSUB wait.ready
    RETURN

write.cal:
    GOSUB wait.ready
    PRINT #1, "OUTPUT "; dac$; ";P"; port; " r"; range; " H"; offset; _
    " J"; gain; "X"
    GOSUB wait.ready
RETURN
wait.ready:
    FOR o = 1 TO 20
          PRINT #1, "SPOLL"; dac$
          INPUT #2, spoll
          IF (spoll AND 4)  4 THEN EXIT FOR
    NEXT o

wait.ready.1:
    PRINT #1, "spoll"; dac$
          INPUT #2, spoll
          IF (spoll AND 4) = 0 THEN GOTO wait.ready.1
RETURN




RETURN
```

```
wait.mav:
   PRINT #1, "spoll"; dac$
   INPUT #2, spoll
   IF (spoll AND 16) = 0 THEN GOTO wait.mav
RETURN

set.channel:
   PRINT #1, "output"; KI$; ";N"; port; "X"
RETURN


'BI-POLAR spans and voltages
DATA 2#,1%,4#,2%,10#,5%,20#,10%

'UNI-POLAR spans and voltages
DATA 1#,1%,2#,2%,5#,5%,10#,10%
```

Appendix A: Glossary

Address Mode: See Primary or Secondary Addressing Mode

Analog Output Ports: Two on the DAC488HR/2; four on the DAC488HR/4. Each port has a Digital to Analog converter and a data buffer.

Asynchronous update mode: Also asynchronous update clock mode. Update mode in which triggers are recognized as quickly as possible. Forces the trigger to react by generating a new update clock within two cycles of the base frequency used for update clock generation. Update clock generation is resynchronized to the trigger event.

Attention (ATN): IEEE 488 bus management line. If attention is asserted, information contained on the data lines is interpreted as a multiline command. If it is not asserted, information is interpreted as data for the active listeners.

Bipolar analog output mode: The range is – full scale to + full scale and each bit has a value of 1/32,768 of the full scale range. Voltage is represented as a 16-bit two's complement binary number.

Buffer Count (`Kn`): Defines how many times the data buffer is to be repeated in response to a trigger event.

Buffer Data (`Bval`): Command that writes a voltage value to the internal data buffer. Buffer data are written to the location pointed to by the data buffer location pointer.

Buffer Mode (`An`): Command that sets the manner in which the data buffer and sequence control table for each analog output port handles data output in response to triggers. See also Linear Buffer Mode and Complex Buffer Mode.

Burst trigger control mode: A trigger control mode in which the data buffer is output at the update rate each time a trigger is detected. When the end of the defined data buffer is reached, the analog output is held at the last output value and triggers are rearmed. Triggers are rearmed and the data buffer output until the buffer count is exhausted. After the buffer count is exhausted, trigger detection is disarmed, and the output is held at the last output value.

Bus terminators: Specific character or signal that defines the end of an IEEE 488 bus communication. In the DAC488HR, bus terminators are fixed; they cannot be altered by the command set. The DAC488HR recognizes either linefeed, EOI, or both as an input terminator. The DAC488HR terminates data output with linefeed plus EOI.

Calibration Enable: Switch on rear of unit that protects calibration constants held in NVRAM. When enabled, existing calibration constants for each analog output port may be stored in NVRAM. When disabled, the calibration constants can be changed, but attempts to write them to NVRAM result in an error.

Command channel: In secondary addressing mode, the command channel is the address where commands are issued to the DAC488HR and responses are returned from the DAC488HR.

Command conflict error: See Device Dependent error.

Command Trigger (@) trigger source: Generates a trigger for all analog output ports when the Trigger Source is set for a command trigger (T5). The trigger command is not processed until an Execute (X) command is received.

Complex Buffer Mode (A1): Complex buffer mode allows flexible sequencing through the data buffer for data output. It is intended for applications with very complex or memory intensive waveforms or with multiple waveforms that fit in a large buffer with infrequent switching between them. See also: Sequence Control Table.

Continuous Mode: For continuous output to analog output ports from the IEEE 488 bus at up to 100 Ksamples per second.

CR: Carriage Return

DAC488HR: DAC488HR/4 or the DAC488HR/2.

DAC488HR/2: DAC488HR with two analog output ports.

DAC488HR/4: DAC488HR with four analog output ports.

Data buffer: One for each analog output port; stores analog output values. Has 8K sample points standard; can be 128K with the MEMX3 option or 480K with the MEMX4 option.

Data Buffer Location Pointer (Ln): Command that points to a specified location in the data buffer.

Data Buffer Output Control (Buffer Mode): See Linear Buffer Mode or Complex Buffer Mode.

Data channel: In secondary addressing mode, there are four data channels, each with a separate secondary address. These correspond to the four separate analog output ports. Commands may not be issued to these channels, nor may responses be read from these channels. Only data may be passed to and from each analog output port's data buffer through these channels.

Device Clear (DCL and SDC): Clears the command input buffer, the command response queue and any pending commands.

Device Dependent (conflict) error: Generated when two commands are issued that are mutually incompatible. The commands themselves and the options for them may be valid, but the combination of the two commands cannot be executed by the DAC488HR.

Direct Trigger Control mode: A voltage is output upon receipt of the voltage value command. Direct control is accomplished by selecting the analog output port and range and specifying the output voltage.

DMA underrun error: This error occurs when one or more analog output ports are set for continuous output mode, and data are not supplied fast enough to meet the update rate.

EOI: IEEE 488 bus management line. Used to signal the last byte of a multibyte data transfer. The device sending the data asserts EOI during the transfer of the last data byte. The Active Controller also uses EOI to perform a Parallel Poll by simultaneously asserting EOI and ATN.

ERROR Indicator Light: On when an error has occurred, off when no error condition exists.

Error Query (E?): Command that returns the present error condition of the DAC488HR. After execution of the Error Query command, most error conditions are cleared.

Event Mask (Nn): Command that directly sets the Event Status Enable Register (ESE).

Event Status Enable (ESE): Register that determines which conditions in the Event Status Register (ESR) are enabled to generate Event Status register Bit (ESB) in the Serial Poll Status Register.

Event Status Register (ESR): A status register in the DAC488HR. Always set to indicate the status of DAC488HR, but do not generate an Event Status Bit (ESB) in the Serial Poll Status Register unless the corresponding enable bit in the Event Status Enable (ESE) register has been set with the Event Mask (Nn) command.

Event Status register Bit (ESB): Bit in the Status Byte Register. It is the combined state of an 8-bit Event Status Register (ESR) and the Event Status Enable (ESE) register.

Execute (X): Command that executes preceding command string. Commands sent without an X are stored in an internal buffer and are not executed until an X is received.

External Clock: Allows synchronization of the analog output ports to an external frequency reference. External clock rates up to 10 MHz may be input and divided by a 16-bit divider.

External TTL trigger: A TTL-level signal that may be applied as a trigger source for DAC488HR operation. The rising, falling, or either edge of the external trigger may be selected as the active edge.

Format (Fn): Command that selects the input and output format the DAC488HR uses when sending a voltage value to the controller. Available formats are: Signed volts in ±10.0000
Signed volts in fixed format (+ implied)–10.0000 (leading space for positive)
Decimal bits
Hexadecimal bits

Gain Calibration (Jn): Command that sets the gain correction calibration constant to a value between 0 and 4095 for the selected range and analog output port.

Group Execute Trigger (GET): An IEEE 488 bus command that usually signals a group of devices to begin executing a triggered action. This allows actions of different devices to begin simultaneously.

Group Execute Trigger (GET) trigger source: Generated when GET is issued on the IEEE 488 bus when the DAC488HR is a listener. It is normally used to synchronize DAC488HR actions with other devices on the IEEE 488 bus.

High Voltage/High Current Digital Outputs: Special configuration of DAC488HR digital outputs. Can sink up to 100 mA at 50 V dc through the use of open collector drivers with integral diodes for inductive load transient suppression.

IEEE 488 bus: An instrumentation communication bus adopted by the Institute of Electrical and Electronic Engineers in 1975 and revised in 1978.

IEEE 488 Bus Address: Each device on the IEEE 488 bus must have a unique address to distinguish it from other devices on the bus. This address may range from 0 to 30.

Immediate trigger control mode: The data buffer outputs at the system update rate as soon as the command to set this mode is processed. When the end of the buffer is reached, it is repeated until the buffer count is exhausted. After the buffer count is exhausted, output is held at the last output value. If buffer count is set to 0, the data buffer repeats forever.

Interface Clear (IFC): IEEE 488 bus management line. Used by the system controller to place all bus devices in a known state. Although device configurations vary, the IFC command usually places the devices in the Talk and Listen Idle states.

Interleaved data: Consist of two bytes of binary data for one analog output port, then two bytes of binary data for the other, and so on. In order to set which analog output port captures which set of binary data, two different modes are provided (see trigger control mode).

Interval Timer (Yn): Sets interval time to 1 - 65,535 milliseconds. This time is used as a trigger source when Trigger Source is set to T6.

LA: Listener Active

LAG: Listen Address Group

LF: Line Feed

Linear buffer mode: Allows simple handling of an analog output port's data buffer at the expense of flexibility. It treats the entire data buffer for an analog output port as one unit. It is not necessary to keep track of beginning or end addresses in the buffer or to specify any control information governing data buffer output.

LISTEN Indicator light: On when the DAC488HR is in the Listener Active state, Off when the DAC488HR is in the Idle or Talker Active state.

Looping: Used with complex control mode to repeat the segments of the data buffer up to 65,535 times. Permits long periodic waveforms to be stored in the data buffer as one period of the waveform.

MEMX3: 128K sample (256K byte) single channel memory expansion module.

MEMX4: 480K sample (960K byte) single channel memory expansion module.

NDAC: IEEE 488 bus line that indicates to the talker that each device addressed to listen has accepted the information. Each device releases NDAC (high) at its own rate, but the NDAC does not go high until the slowest Listener has accepted the data byte.

Offset Calibration (Hn): Sets the offset calibration constant to a value between 0 and 4095 for selected range and analog output port. It trims out offset errors at the output of the analog output port.

Output Terminators: See bus terminators

Port commands: Commands that only affect the operation of the selected analog output port. They configure each analog output port and are active for each analog output port separately.

PORT 1, PORT 2, PORT 3, PORT 4 Indicator Lights: Indicate status of respective analog output port. Off when its analog output port is set to 0 volts. On when outputting a steady dc voltage. Flashing when outputting data.

Port Select (Pn): Command that determines which analog output port is selected for use with subsequent commands. Only one analog output port at a time may be selected.

POWER Indicator Light: On when power is applied to the DAC488HR and the power switch on the back panel is depressed. Off if power is not present.

Pre-defined waveform functions: The DAC488HR provides a command to load one cycle of a sine wave, triangle wave, or square wave into the data buffer of the selected analog output port. **Primary addressing mode:** In this mode, the DAC488HR occupies one primary address on the IEEE 488 bus.

Query (?): Option for each command that can be used to determine the present configuration or mode of a previously executed command. To use Query, follow the first letter of the command with a question mark (?). Any number of query commands can be combined into one string to create a specialized status command that returns only the information of interest for a given application. Query commands must be followed by the Execute (X) command.

Range (Rn): The range command specifies which voltage range is used on an analog output port, and whether the analog output mode is bipolar or unipolar.

Remote Enable (REN): IEEE 488 bus management line. It causes bus devices to respond to remote operation. Only the System Controller has control of the Remote Enable line.

Reset (*R): Command that has the same effect on the DAC488HR as removing and re-applying power.

Save/Restore (Sn): Command that saves or restores a default setting for each command. It is also used save or restore the calibration constants.

Secondary addressing mode: In this mode, one primary address and several secondary addresses are used on the IEEE 488 bus. One secondary address is dedicated to the command channel, and a single secondary address is dedicated to each analog output port for a data channel.

Selected Device Clear (SDC): See Device Clear.

Sequence Control Block: Structures in the sequence control table that define how data are output in complex buffer mode. They define the position and order of the data buffer segments to output and allow repeating, or looping, of segments of the data buffer.

Sequence Control Table: Defines how data are output in complex buffer mode. Table contains structures called sequence control blocks that define the position and order of the data buffer segments to output and allow repeating of segments of the data buffer.

Sequence Pointer (On): This command determines where in the Sequence Control Table the next operation involving a Sequence Control Block occurs.

Serial Poll Disable (SPD): IEEE 488 bus command that disables all devices from sending their Serial Poll status byte.

Serial Poll Enable (SPE): IEEE 488 bus command. After SPE is sent and ATN is unasserted, a device that is Addressed to Talk outputs its Serial Poll status byte.

Serial Poll Status Byte Register (Spoll) command: Sent when a serial poll (Spoll) command is received over the IEEE 488 bus from the active controller.

Service Request (SRQ): A device asserts SRQ when it needs the immediate attention of the active controller.

Service Request Enable (SRE): A register that determines which bit(s) in the Status Byte Register generate an SRQ on the IEEE 488 bus.

SPD: See Serial Poll Disable.

SPE: See Serial Poll Enable.

SRQ Indicator Light: On when the DAC488HR has generated a service request (SRQ), off when no SRQ is pending.

SRQ Mask (Mn): Sets the Service Request Enable Register. SRQs can be enabled on end of trigger sequence, Message Available, event status bit or IEEE 488 error.

SRQ: See Service Request.

Status byte register (STB): See Serial Poll Status Byte Register.

Status (Un): Command that returns the status of various DAC488HR functions.

Step trigger control mode: Causes an analog output port to output the next value from the data buffer each time a trigger is detected. After data are output, trigger detection is rearmed. When the end of the defined data buffer is reached, the data buffer is repeated. When the buffer count is exhausted, the output is held at the last output value and trigger detection is disarmed.

Synchronous update mode: Holds off the reaction to a trigger until the next occurrence of an update clock. Ensures that no frequency components higher than the update rate show up in the frequency spectrum of generated waveform outputs.

System commands: Affect operation of the entire DAC488HR unit and are not specific to a given analog output port.

TALK Indicator Light: On when the DAC488HR is in the Talker Active state, off when the DAC488HR is in the Idle or Listener Active state.

Terminators: See Bus Terminators.

TEST Indicator Light: On when the DAC488HR is performing a power-on self-test. Off when the unit is operating normally and all self-tests pass.

Trigger (@): The trigger command generates a trigger for all analog output ports when the Trigger Source is set for a command trigger (T5).

Trigger Control Mode (Cn): Command that defines how each analog output port responds to a trigger in the output of its data. Available trigger control modes are: step, burst, waveform, immediate and continuous.

Trigger Delay (Zn): Command that sets the trigger delay for the delayed trigger output. The time delay is specified in update clocks to any value from 1 to 65,535 update clocks. This sets the time delay between when any analog output port reacts to a trigger and when the delayed trigger output goes true.

Trigger latency: The time from the occurrence of a trigger event to DAC488HR response to the trigger event.

Trigger OUT: Pin on the DAC488HR. The trigger signal is buffered and routed to this digital output. Allows synchronization of external equipment to DAC488HR operation.

Trigger Source (Tn): Command that determines the trigger source used for all analog output ports. The power-up default is trigger disabled. Available trigger sources are GET, interval timer, command (@), and TTL input.

Unipolar analog output mode: The range is 0 to + full scale and each bit has a value of 1/65,536 of the full scale range. Voltage is represented as 16-bit unsigned binary integer.

Unlisten (UNL): UNL places the DAC488HR in the Listener Idle state.

Untalk (UNT): IEEE 488 bus command that tells bus devices to Untalk. UNT places the DAC488HR in the Talker Idle state.

Update Clock: Controls when the analog output voltages start to change. Is used when updating the analog output ports in any of the triggered modes of operation. A single update clock is shared by all analog output ports to insure synchronization of changes between them.

Update Clock generation mode: See Asynchronous Update Mode or Synchronous Update Mode.

Update Divider (In): Command that specifies the time interval between the output of each data point. The frequency of any cyclical data stored in the data buffer can be controlled by selecting the proper interval based on the number of voltage points per cycle.

Update mode: See Asynchronous Update Mode or Synchronous Update Mode.

Update Rate: The time between update clocks.

Update Source clocks: The Update Source, Mode command (Gn) chooses the clock source for update clock generation. This update source clock is fed to the input of a 16-bit down counter for generation of the Update Clock. The Update Divider determines the number of update source clocks between update clocks. Every time the counter underflows, an Update Clock pulse is generated.

Update Source, Mode (Gn): This command specifies the Update Clock source and the Update Clock mode. Available clock sources are: 5 MHz, 2.8224 MHz, 3.072 MHz, 200 kHz or external clock (TTL compatible). Update mode can asynchronous or synchronous.

Voltage Output (Vval): Command that sets a voltage value on the analog output of the selected analog output port. A voltage may be specified in volts, decimal bits or hexadecimal bits.

Waveform Load (Ww,l,max,min,d): This command automatically writes one of three pre-defined waveforms into an analog output port's data buffer. The available pre-defined waveforms are sine, triangle or square.

Waveform trigger control mode: Causes the data buffer to output at the system update rate when a trigger is detected. When the end of the buffer is reached, it is repeated until the buffer count is exhausted. After the buffer count is exhausted, output is held at the last output value and trigger detection is disarmed. If buffer count is set to 0, the data buffer repeats forever.

Character Codes and IEEE Multiline Messages

$00 0 NUL	$10 16 DLE	$20 32 SP 00	$30 48 0 16	$40 64 @ 00	$50 80 P 16	$60 96 ` SCG	$70 112 p SCG
$01 1 SOH GTL	$11 17 DC1 LLO	$21 33 ! 01	$31 49 1 17	$41 65 A 01	$51 81 Q 17	$61 97 a SCG	$71 113 q SCG
$02 2 STX	$12 18 DC2	$22 34 " 02	$32 50 2 18	$42 66 B 02	$52 82 R 18	$62 98 b SCG	$72 114 r SCG
$03 3 ETX	$13 19 DC3	$23 35 # 03	$33 51 3 19	$43 67 C 03	$53 83 S 19	$63 99 c SCG	$73 115 s SCG
$04 4 EOT SDC	$14 20 DC4 DCL	$24 36 $ 04	$34 52 4 20	$44 68 D 04	$54 84 T 20	$64 100 d SCG	$74 116 t SCG
$05 5 ENQ PPC	$15 21 NAK PPU	$25 37 % 05	$35 53 5 21	$45 69 E 05	$55 85 U 21	$65 101 e SCG	$75 117 u SCG
$06 6 ACK	$16 22 SYN	$26 38 & 06	$36 54 6 22	$46 70 F 06	$56 86 V 22	$66 102 f SCG	$76 118 v SCG
$07 7 BEL	$17 23 ETB	$27 39 ' 07	$37 55 7 23	$47 71 G 07	$57 87 W 23	$67 103 g SCG	$77 119 w SCG
$08 8 BS GET	$18 24 CAN SPE	$28 40 (08	$38 56 8 24	$48 72 H 08	$58 88 X 24	$68 104 h SCG	$78 120 x SCG
$09 9 HT TCT	$19 25 EM SPD	$29 41) 09	$39 57 9 25	$49 73 I 09	$59 89 Y 25	$69 105 i SCG	$79 121 y SCG
$0A 10 LF	$1A 26 SUB	$2A 42 * 10	$3A 58 : 26	$4A 74 J 10	$5A 90 Z 26	$6A 106 j SCG	$7A 122 z SCG
$0B 11 VT	$1B 27 ESC	$2B 43 + 11	$3B 59 ; 27	$4B 75 K 11	$5B 91 [27	$6B 107 k SCG	$7B 123 { SCG
$0C 12 FF	$1C 28 FS	$2C 44 , 12	$3C 60 < 28	$4C 76 L 12	$5C 92 \ 28	$6C 108 l SCG	$7C 124 | SCG
$0D 13 CR	$1D 29 GS	$2D 45 - 13	$3D 61 = 29	$4D 77 M 13	$5D 93] 29	$6D 109 m SCG	$7D 125 } SCG
$0E 14 SO	$1E 30 RS	$2E 46 . 14	$3E 62 > 30	$4E 78 N 14	$5E 94 ^ 30	$6E 110 n SCG	$7E 126 ~ SCG
$0F 15 SI	$1F 31 US	$2F 47 / 15	$3F 63 ? UNL	$4F 79 O 15	$5F 95 UNT _	$6F 111 o SCG	$7F 127 DEL SCG
ACG	UCG	LAG		TAG		SCG	

ACG = Addressed Command Group
UCG = Universal Command Group
LAG = Listen Address Group
TAG = Talk Address Group
SCG = Secondary Command Group

Appendix C: Command Summary

Command	Code	Type	Description	Page #
Command Trigger	@	System	Generate a command trigger.	5.12
Reset	*R	System	Return DAC488HR to power up state.	5.15
Buffer Mode	A0	Port	Linear buffer control mode (default).	5.16
	A1		Complex buffer control mode.	
	A?		Return present buffer control mode setting.	
Buffer Data	Bval	Port	Writes data value `val` into the data buffer and advances the data buffer location pointer for selected analog output port.	5.17
	B?		Return the data value from the data buffer at the data buffer location pointer for the selected analog output port.	
Trigger Control Mode	C0	Port	Trigger response disabled (default).	5.20
	C1		Step mode.	
	C2		Burst mode.	
	C3		Waveform mode.	
	C4		Immediate mode.	
	C5		Continuous, one analog output port.	
	C6		Continuous, two analog output ports, odd.	
	C7		Continuous, two analog output ports, even.	
	C?		Returns existing control mode for selected analog output port.	

Command	Code	Type	Description	Page #
Digital Output	Dn	System	Outputs the value n (0-255) to a digital output port. Default is D0.	5.25
	D?		Return the present value of the digital output port in three-digit decimal format.	
Error Query	E?	System	Returns and clears present error condition. After execution of the Error Query command, the DAC488HR returns one of the following error codes.	5.26
	E000		No error (default).	
	E001		Invalid device dependent command (IDDC).	
	E002		Invalid device dependent command option (IDDCO).	
	E004		Command conflict.	
	E008		Calibration enable.	
	E016		Trigger overrun.	
	E032		Non-volatile setup.	
	E064		Non-volatile calibration constants.	
	E128		DMA underrun	
Format	F0	System	Sets format to signed volts in ±10.00000 (default).	5.28
	F1		Signed volts in fixed format (+ implied) –10.00000 (leading space for positive).	
	F2		Sets format to decimal bits.	
	F3		Sets format to hexadecimal bits.	
	F4		Sets all daughterboard output order to low byte first.	
	F5		Sets all daughterboard output order to high byte first.	
	F?	(Fn,Fm)	Returns present format selection	

Command	Code	Type	Description	Page #
Update Source, Mode	G0	System	5 MHz, synchronous (default).	5.30
	G1		2.8224 MHz (64 x 44.1 KHz), synchronous.	
	G2		3.072 MHz (64 x 48 KHz), synchronous.	
	G3		200 kHz, synchronous.	
	G4		External Clock, synchronous.	
	G5		5 MHz, asynchronous.	
	G6		2.8224 MHz, asynchronous.	
	G7		3.072 MHz, asynchronous.	
	G8		200 kHz, asynchronous.	
	G9		External Clock, asynchronous.	
	G?		Returns present update source.	
Offset Calibration	Hn	Port	Set the offset calibration constant to value n (0 – 4095) for selected range and analog output port.	5.33
	H?		Returns the offset calibration constant value for the selected range and analog output port.	
Update Divider	In	System	Set Update Source divisor to generate Update Clock (1 – 65,535) (default is 2).	5.34
	I?		Returns the present Update Clock interval in Update Source clocks.	
Gain Calibration	Jn	Port	Set the gain correction calibration constant to value n (0 – 4095) for the selected range and analog output port.	5.36
	J?		Returns the gain correction calibration constant for the selected range and analog output port.	
Buffer Count	Kn	Port	Set the number of times the entire data buffer is repeated (0 – 65,535, 0 = continuous). Default is K1.	5.37

Command	Code	Type	Description	Page #
	K?		Returns the number of repetitions specified.	
Data Buffer Location Pointer	Ln	Port	Set the data buffer location pointer for the selected analog output port to location n. Default is L0. n is between 0 and 8,191 standard n is between 0 and 131,071 with the MEMX3 option n is between 0 and 491,519 with the MEMX4 option	5.38
	L?		Returns the present buffer location. The value returned is the approximate location in the buffer that is being output at that time.	
SRQ Mask	M000	System	Clear SRQ mask (default).	5.40
	M001		SRQ on trigger.	
	M002		SRQ on end of trigger sequence.	
	M004		SRQ on ready.	
	M008		SRQ on error.	
	M016		SRQ on Message Available (MAV).	
	M032		SRQ on enabled event (ESB).	
	M128		SRQ on DMA Underrun.	
	M?		Reads Service Request Enable Register. Returns present service request mask.	
Event Mask	N000	System	Clear event mask (default).	5.41
	N001		End of trigger sequence.	
	N004		Enable query error.	
	N008		Enable device dependent error.	
	N016		Enable execution error.	
	N032		Enable command error.	
	N128		Power-on.	

Command	Code	Type	Description	Page #
	N?		Read standard event status enable register	
Sequence Pointer	On	Port	Set the sequence pointer to n. Default is 00000.	5.43
	O?		Returns the present sequence pointer.	
Port Select	P1	System	Select analog output port 1 (default).	5.45
	P2		Select analog output port 2.	
	P3		Select analog output port 3 (DAC488HR/4 only).	
	P4		Select analog output port 4 (DAC488HR/4 only).	
	P?		Return present analog output port selection.	
Define Sequence Control Block	Q1,n,r	Port	Define a segment starting at location 1 with a length of n (32 – 32,768) samples, with a repeat count of r (0 – 65,535) times. A length of 0 deletes the existing segment. The sequence pointer is advanced to the next location.	5.46
	Q?		Returns the Sequence Control Block pointed to by Sequence Pointer (On) and advances the sequence pointer, allowing multiple Sequence Control Blocks to be read with the ? option.	
Range	R0	Port	Select analog output port ground range (analog output pins are shorted together) (default).	5.48
	R1		Select analog output port 1 volt range, bipolar.	
	R2		Select analog output port 2 volt range, bipolar.	
	R3		Select analog output port 5 volt range, bipolar.	
	R4		Select analog output port 10 volt range, bipolar.	
	R5		Select analog output port 1 volt range, unipolar.	
	R6		Select analog output port 2 volt range, unipolar.	
	R7		Select analog output port 5 volt range, unipolar.	
	R8		Select analog output port 10 volt range, unipolar.	

Command	Code	Type	Description	Page #
	R?		Returns voltage range selection for selected analog output port (selected with Pn command).	
Save/Restore	S0	System	Restore setup stored in NVRAM (default).	5.50
	S1		Save the existing command settings as the power-on default setup in NVRAM.	
	S2		Restore calibration settings from NVRAM.	
	S3		Save the present calibration settings in NVRAM.	
	S4		Restore factory default power-on setups from NVRAM.	
	S?		Return the last Sn command executed	
Trigger Source	T0	System	Disable triggering (default).	5.53
	T1		GET.	
	T2		Trigger on TTL rising edge.	
	T3		Trigger on TTL falling edge.	
	T4		Trigger on TTL either edge.	
	T5		Trigger on command (@).	
	T6		Trigger on interval timer.	
	T7		Slave trigger.	
	T?		Return present trigger setting.	
Status	U0	System	Returns Event Status Register (ESR) (default).	5.55
	U1		Returns status byte register (STB).	
	U2		Reads all settings of instrument in the form DnFnGnInMnNnTnYnZn.	
	U3		Returns all installed analog output port settings in the form P1AnCnKnRnVv,P2AnCnKnRnVv...	
	U4		Returns a three-digit decimal number representing the present state of the eight digital input lines.	

Command	Code	Type	Description	Page #
	U5		Returns analog output port data buffer sizes in the form xxxxxx,xxxxxx,000000,000000 where 000000 indicates port not installed.	
	U6		Returns all analog output port sequence control table sizes in the form xxxx,xxxx,0000,0000 where 0000 indicates port not installed.	
	U7		Returns entire data buffer for selected analog output port.	
	U9		Returns IOtech DAC488HR/2,0,v.r or IOtech DAC488HR/4,0,v.r where v is the version and r is the revision of the firmware.	
	U?		Returns the last Un command executed.	
Voltage Output	Vval	Port	Set output to v (format set by Fn)	5.59
	V?		Returns present output value for the selected analog output port in the format specified by Format (Fn) command.	
Waveform Load	Ww,l,max,min,d			
		Port	Write pre-defined waveform w (0 - 2), of length l (32 - 32,768), with maximum value max (±100%), minimum value min (±100%), and symmetry d (0 - 100%). Starting location is determined by Ln.	5.62
	W0		Sine wave.	
	W1		Triangle wave.	
	W2		Square wave.	
Execute	X	System	Execute preceding command string.	5.66
Interval Timer	Yn	System	Set interval time to n (1 – 65,535) milliseconds. Default is Y1.	5.67
	Y?		Return present interval timer setting.	

Command	Code	Type	Description	Page #
Trigger Delay	Zn	System	Set trigger delay to n (1 – 65,535) update clocks. Default is Z1.	5.68
	Z?		Return present trigger delay setting.	

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