DBS902

14 Bit 80Ms/s Digitizer

User's Manual
Revision 01

Proprietary Statement

The information contained in this publication is derived in part from proprietary and patent data of the Analogic Corporation. This information has been prepared for the express purpose of assisting operating and maintenance personnel in the efficient use of the instrument described herein. Publication of this information does not convey any rights to reproduce it or to use it for any purpose other than in connection with the installation, operation, and maintenance of the equipment described herein.

P/N 82-5136
Revision 01

DBS9900, DBS901 and DBS902 are trademarks of Analogic Corporation.

Warranty

Analogic warrants only to the original purchaser that this product, as purchased from Analogic or an Analogic distributor or dealer, will conform to the written specifications for a period of one year from the date of purchase. If the product fails to conform to these warranties, Analogic, as its sole and exclusive liability hereunder, will repair or replace the product and/or its components within a reasonable period of time if the product is returned to an Analogic service center within the warranty period. These warranties are made upon the express condition that:

a) The purchaser promptly notifies Analogic in writing of any non-conformity with the above warranty including a detailed explanation of the alleged deficiencies.
b) The product is returned to an Analogic service center at the buyer's expense after making suitable arrangements for performance of service.
c) When the product is returned for repair, a copy of the original bill of sale or invoice is sent with the product.
d) Analogic will not be liable for any incidental or consequential damages.
e) In the opinion of Analogic upon inspection, the product has not been misused, altered, or damaged due to abnormal handling and/or operation.
f) Repairs to the product and/or its components have not been made by anyone other than Analogic or one of its authorized repair agents.
g) The product has not been modified, altered, or changed in any manner by anyone other than Analogic or one of its authorized repair agents.

THIS WARRANTY EXCLUDES ALL OTHER WARRANTIES, WHETHER EXPRESSED OR IMPLIED, ORAL OR WRITTEN, INCLUDING WITHOUT LIMITATION WARRANTIES OF MERCHANTABILITY AND/OR FITNESS FOR A PARTICULAR PURPOSE.

No term, condition, understanding or agreement purporting to modify the terms of this warranty shall have any legal effect unless made in writing and signed by an authorized officer of Analogic and the purchaser.

Definition of Terms

Term	Definition
I/0 1, I/0 2	Also called daughterboards, and/or modules. There can be one or two modules installed in the DBS9900. I/O 1 refers to the first installed module and I/O 2 refers to the second.
DBS901	Arbitrary Waveform Generator daughterboard (purchased separately).
DBS902	Digitizer daughter board (purchased separately).
Sample Clock	Clock which drives the sample rate of each module. An *Internal* or *External* Sample Clock may be used. It is possible (using the driver) for one module to use an Internal Sample Clock and the other to be using an External Sample Clock.
Reference Clock	Timebase used by the DBS9900 as a reference for both installed modules (I/O 1 and I/O 2). This PLL 10MHz reference signal is provided internally by default but an *External* signal may be used. Can only be applied to the CLK 2 input on the Front Panel.

About this Manual

This manual is organized in a top-down manner.

This means that the first chapters provide overview and installation information.

Next, configuration and setup information is presented.

Finally, functionality is described. Each of the following sections describe similar content – but from a high-level perspective (Soft Front Panel) to low-level (Registers).

- DBS902 Soft Front Panel
- DBS902 Driver (.DLL) Functions
- DBS902 Registers

Reference material is presented in Appendices: Block Diagrams, etc.

Safety Precautions

Warnings and Cautions

The terms **WARNING** and **CAUTION** have specific meanings in this manual:

WARNING

A WARNING advises against certain actions or situations that could result in personal injury or death.

CAUTION

A CAUTION advises against actions that could damage equipment, produce inaccurate data or invalidate a procedure.

Mesures de securite

Mises en garde

Les terms **AVERTISSEMENT** et **ATTENTION** ont des sens spécifiques dans cette notice.

AVERTISSEMENT

Un AVERTISSEMENT informe des actions ou situations qui peuvent présenter un risque de blessure ou de décès.

ATTENTION

Une mise en garde débutant par le terme ATTENTION informe des actions que peuvent endommager le metériel, produire des données incorrectes ou nuire au fonctionnement.

Sicherheitsvorkehrungen

Warnung und Vorsicht

Die Bezeichnungen **WARNUNG** und **VORSICHT** haben in diesen Sicherheitsvorschriften eine besondere Bedeutung.

WARNUNG

Eine WARNUNG rät gegen bestimmte Handlungen oder Situationen, die Verletzung der Person oder Tod zur Folge haben können.

VORSICHT

VORSICHT weist auf Handlungen hin, die das Gerät beschädigen könnten, unrichtige Daten verursachen oder einen Vorgang auslöschen können.

SAFETY SUMMARY

The following safety precautions apply to both operating and maintenance personnel and must be observed during all phases of installation, operation, and service of the unit. Before applying power, follow the installation instructions and become familiar with the operating instructions for all components.

CAUTION

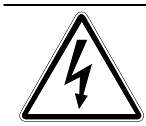
This product uses components which can be damaged by electrostatic discharge (ESD). To avoid damage, be sure to follow proper procedures for handling, storing, and transporting ESD-sensitive devices.

PRÉCAUTIONS

Ce produit utilise des composants pouvant être endommagés par décharge électrostatique (DES). Pour éviter les dégâts, suivre les procédures appropriées lors de la manipulation, du stockage, et du transport des dispositifs sensibles aux décharges électrostatiques.

VORSICHT

Dieses Produkt ist mit Komponenten ausgerüstet, die durch elektrostatische Entladung beschädigt werden können. Zur Vermeidung von Schäden muß dafür gesorgt werden, daß die maßgeblichen Vorschriften für die Behandlung, Lagerung und den Transport für elektrostatisch empfindliche Geräte beachtet werden.



WARNING

Do not operate the unit in the presence of flammable gases or fumes. Operation of the unit in any such environment constitutes a definite safety hazard.

AVERTISSEMENT

Ne pas faire fonctionner l'unité en présence de gaz ou de vapeurs inflammables. Le fonctionnement de l'unité dans un environnement de ce type constitue un danger évident pour la sécurité.

WARNUNG

Die Einheit darf nicht in Gegenwart von feuergefährlichen Gasen oder Dämpfen betrieben werden. Der Betrieb der Einheit in einer derartigen Umgebung stellt eine eindeutige Gefahrenquelle dar.

WARNING

Do not install substitute parts or perform any unauthorized modifications to this unit. Return the unit to Analogic for service and repair to ensure that the safety features are maintained.

AVERTISSEMENT - NE PAS SUBSTITUER LES PIÈCES OU MODIFIER L’UNITÉ

Ne pas installer de pièces de substitution ni effectuer des modifications non autorisées sur l’unité. Renvoyer l’unité à Analogic pour l’entretien et les réparations, de manière à assurer que les dispositifs de sécurité soient maintenus.

WARNUNG - NUR ORIGINALTEILE BENÜTZEN UND DIE EINHEIT NICHT ÄNDERN

Es sind nur Originalteile einzubauen, und ohne Erlaubnis darf keine Änderung an der Einheit vorgenommen werden. Zur Wartung und Reparatur ist die Einheit an Analogic zurückzusenden, damit die Beibehaltung der Sicherheitseigenschaften gewährleistet ist.

Contents

1 Overview

1.1 Introduction

The DBS 902 is a high-speed (80 MHz), 14-bit, Waveform Digitizer module that plugs into the Analogic DBS 9900 "C" size VXI carrier module. Up to two DBS 902s can be installed in a single DBS 9900. This mother/daughterboard concept allows a single VXI chassis slot to provide multiple functions, thereby maximizing VXI resources and decreasing the cost per slot. The DBS 902 provides a combination of speed and resolution that, until now, was not available and continues the Analogic tradition of providing the highest possible state-of-the-art performance at the lowest price available.

Acquisition may be triggered from the input signal itself or external inputs having programmable thresholds. Acquired data can be pre-trigger, post-trigger or anywhere in between. A programmable sample counter that controls the number of data points to be acquired is also provided. The ability to tag sample(s) on the fly for purposes of identifying external events such as time stamps, is available via a front panel digital input.

The DBS 902, when combined with the DBS 9900 carrier module, provides extremely versatile clock and trigger sources via internal circuitry, front panel connectors, and VXI TTLTRG lines. The internal clock can drive all plug-in function modules simultaneously so that clock skew and delays are reduced to an absolute minimum. This allows coherent operation with other plug-in modules in the DBS 9XX series of digitizers and waveform generators as well as multiple DBS 902. Trigger and clock thresholds are programmable for each plug-in function module.

The VXI Plug & Play compliant software driver supports all functions of the DBS 902 and provides automatic recognition and configuration for all plug-in modules that are installed in the DBS 9900 carrier unit. Source code is included as well as .DLL files to allow easy porting to most popular programming environments. These drivers go beyond VXI Plug & Play requirements to ensure system integration and software development time are reduced to an absolute minimum.

Features

- Over Range Data Detection
- Multiple Clock and Trigger Sources
- Programmable Trigger and Clock Thresholds
- Auto Calibration
- Pre- and Post-Trigger Data Acquisition
- Data Dependent Triggering
- Programmable Sample Counter
- Sample(s) Tagging via Front Panel Input

2 Preparation & Installation

2.1 Overview

This section provides instructions for setting up the unit and installing the necessary hardware and software. After unpacking, carefully inspect the hardware for any damage that may have occurred during shipment. If necessary, contact your shipping carrier to file a claim. Save the shipping carton and packing materials if for any reason you need to return the product for repair or replacement.

CAUTION

This product contains components which are sensitive to electrostatic discharge (ESD). Be sure to follow proper procedures for handling, storing and transporting ESD-sensitive assemblies.

2.2 Host Computer Requirements

Minimum system requirements for running the driver and soft front panel applications:

- Pentium processor
- 32 MB RAM
- Win 9x/Win NT 4.0, Service Pack 3 or higher
- 20 MB free hard drive space

2.3 Hardware Installation

The DBS902 Digitizer plugs into the DBS9900 High-Performance VXI Carrier.

The DBS902 is a plug-in module that is installed on the DBS9900 carrier. The DBS902 is shipped from the factory already installed onto the DBS9900 if both the DBS9900 and DBS902 are purchased together.

To install a DBS902 onto the DBS9900 Carrier:

- Line up the pin connectors on each board, paying close attention to orientation
- Press firmly to ensure the pin connectors are seated properly when plugged in.

Make sure the DBS9900 Carrier is removed from the VXI chassis before installing the DBS902 or any plug-in.

2.4 Software Installation

The DBS902 can be controlled:

- by using the DBS902 Soft Front Panel (SFP) in conjunction with the DBS9900 Soft Front Panel
- by programming applications using the .DLL shipped with each unit, or by
- directly manipulating the hardware registers on the DBS902.

Any necessary software is shipped on the CD/ROM.

Note: The DBS902 Soft Front Panel must always be opened by 'Connecting' to a 902 from the DBS9900 SFP Main Panel.

2.4.1 Installation Steps

Follow the steps below to install the software from the CD/ROM.

Step	Action
1.	Insert CD/ROM.
2.	If the setup procedure does not appear automatically, select Run from the Start Menu, then type **`d:setup`** and hit Return.
3.	Follow the instructions as they appear in the install script.

2.5 Calibration

2.5.1 AUTO CAL Procedure

Upon Initialization, an auto-calibration procedure automatically calibrates the DBS902.

This procedure calculates and loads the proper values into the EEPROM.

The DBS902 calibration routine is a software application running on the user's host workstation. It utilizes an onboard analog to digital converter to adjust the DC Offset. Gain is trimmed one time at the factory and should not require re-calibration. The calibration routine adjusts only the internal circuit offsets of the DBS902 such that the two analog to digital converters' outputs are centered at 0 input voltage. This sets the absolute DC accuracy of the DBS902.

<u>Recommendation:</u>
Temperature fluctuations over time may affect the calibration of the unit. If you are operating in a high-performance environment, it is recommended you re-calibrate every few hours using the AUTO CAL procedure.

<u>Caution:</u>
Any captured data will be lost upon re-calibration. Be sure to save acquired data prior to re-calibration.

2.6 Performance Specifications

2.6.1 Analog Input

Number of Channels	One differential
Full Scale Input Ranges	Programmable ±0.5V, ±1.0V, ±2.5V, ±5.0V, ±10V, and ±25V.
Impedance	Programmable 50, 75, or 1 MΩ, for input ranges ±0.5V, ±1.0V, and ±2.5V.
	Programmable 50, 75, or 100 kΩ for input ranges ±5.0V and ±10V. 100 kΩ for ±25V input range. All impedance values are ±1%.
Capacitance	50 pF max.
Common Mode Voltage CMRR	60 dB worst case. DC to 20 kHz on ±0.5V range.
Input Bias Current	±10 nanoamperes max.
Crosstalk Rejection Channel to Channel	Better than 100 dB
Overvoltage Protection	±3V max. for ±0.5V, ±1.0V and ±2.5V ranges ±30V max. for ±5.0V, ±10.0V, and ±25V ranges.
Offset	Calibrated to ±2 mV max.
Frequency Response[2]	±0.5V, ±1.0V, and ±2.5V[2] input ranges
	±0.1 dB DC to 5 MHz and
	±1 dB 5 MHz to 40 MHz
	±5.0V, ±10.0V and ±25V[2] input ranges
	±0.5 dB DC to 5 MHz and ±2 dB 5 MHz to 40 MHz
Slew Rate	Not slew rate limited.

2.6.2 Transfer Characteristics

Absolute DC Accuracy	0.03% of FS (@5 LSB)
	(±5°C from temp at autocal)
Integral Non-Linearity	±1.25LSB
Differential Non-Linearity	+1.5, -1.0 LSB

2.6.3 Dynamic Characteristics

SINAD	68 dB typ., 64 dB min., 20 Hz to 30 MHz with 80 MHz internal sample clock
Dynamic Amplitude Accuracy	0.02 dB
IMD	80 d: Two Tone 1 MHz @ 5 MHz @ -7 dBc Out
Harmonic Distortion	2nd Harmonic –70 dB @ 1 MHz
	3rd Harmonic –85 dB @ 1 MHz
SFDR	68 dB min., 20 Hz to 10 MHz with 80 MHz internal sample clock.
Noise	200 µV RMS max. @ 40 MHz bandwidth @ 50Ω source impedance.
ENOB	13 bits @ 5 MHz
	12 bits @ 5 MHz to 40 MHz

2.6.4 Signal Conditioning

Software selectable. 3rd order Bessel low-pass filters at 20 MHz and 40 MHz or high bandwidth with no filters.

2.6.5 Internal Sample Clock

Frequency	10 kHz to 80 MHz
	Also see DBS 9900 carrier module specs

2.6.6 Control Inputs

External Sample Clock	10 kHz to 80 MHz (Pipeline delay = 11 clock periods)
	Also see DBS 9900 carrier module specs.
Triggers	Internal via software write to register. External via front panel inputs or VXI TTLTRG lines. Data-dependent triggering derived from a sequence of two level comparisons having 8-bit resolution and programmable sign. Latency of 3 sample periods between levels.
	Also see DBS 9900 carrier module specs.
Phase Reference Clock	10 MHz via front panel input.
	Also see DBS 9900 carrier module specs.

2.6.7 Data Memory

Coding & Format	2's complement or offset binary Bit 15 is sign, bits 14-2 are magnitude, bit 1 is marker and bit 0 is sync.
Size	2 MS configurable as two 1 MS buffers.

2.6.8 Power

Total Power	30W max. including DBS 9900
	+5V: 2.5A max.
	±12V: ±0.5A max.
	-5.2V: 0.5A max.
	-2V: 0.3A max.
	±24V: 0A

2.6.9 Interrupts

Programmable Level	IRQ 1 through 7
Available For	Triggered Condition
	Done Condition
	Overrange Condition
	Diagnostic IRQ

2.6.10 Reliability

MTBF & Failure Rate	>40,000 hours[3]

2.6.11 Environmental & Mechanical

Operating Temperature Range	0°C to 40°C ambient for rated specifications
Storage Temperature Range	-25°C to +75°C non-condensing
Cooling Air Flow	4 litre/sec. For 10°C Rise at 10 mm H_2O back pressure
Relative Humidity	85% max. non-condensing
Vibration	2G sinusoidal @ 10-150 Hz frequency range
Shock	10G
Weight	5.0oz (142g)
Size	8 in. x 4 in., (20.32 cm x 10.16 cm.)

2.6.12 Stability

Warm Up Time	None for rated specifications
Offset Tempco	±50 μV/°C max.
Gain Tempco	±20 ppm/°C

2.6.13 EMC, RFI & Safety Regulatory Agency Compliance

Designed to meet all requirements for safety, RF emissions and immunity for marketing in the U.S. and Canada.

2.6.14 Calibration

Calibration is fully automatic via software driver provision. Calibration values are stored in on-board NVRAM

Notes:

1. All specifications are for modules installed in DBS 9900 carrier module unless otherwise stated.
2. Frequency response measured with no filter, 80 MS/s and –1 dBfs input @ 50Ω
3. Determined by the Generic Parts Count method of MIL-HDBK-217F for a Ground Benign environment at a temperature of 30°C.

Specifications subject to change without notice.

3 Operation

The DBS902 Digitizer is used to capture, digitize and display data.

3.1 Selecting High- or Low-Speed ADC

The DBS902 utilizes two analog to digital converters (ADCs) for optimum performance over the entire range of sample clock frequencies.

The low speed ADC is optimized for operation from 10KS/S to 10MS/S.

The high speed ADC is optimized for operation over the 10MS/S to 80MS/S range.

The user must select the appropriate ADC for the sample clock. This selection can be done directly by making a call to the an902_setADC driver function or by selecting the appropriate setting in the DBS902 Soft Front Panel.

Caution: Attempting to operate outside the specified sample clock ranges of the two ADCs can result in corrupted data or no data acquisition.

3.2 Memory Buffer Usage

The DBS902 has two 1 Meg-sample data buffers that can accept ADC data automatically. The reason for having two buffers is to allow data to be transferred out of one buffer by the VXI bus host computer while simultaneously acquiring ADC data into the other buffer. After all the data is transferred out, the VXI host can switch the buffers so that it can transfer the next block of data. This "double buffer" architecture helps improve throughput.

The DBS902 can also be programmed to combine both buffers into one large 2 Meg sample buffer for seemless acquisition of 2097152 samples.

The DBS902 memory space that is reserved on the VXI bus is large enough for the 2 Meg sample buffer. When using the double-buffered mode, the VXI host must use a different base address for the Upper data buffer and the Lower data buffer. It must also configure the DBS902 Input Control Register (0x20) so that the 902 memory-controller will acquire data into the correct buffer. Bits D6 (Buffer Select) and D7 (Buffer Mode) are used to control buffer usage. The following table summarizes this.

	Buf Select	Buf Mode	Upper	Lower	Base address (bytes)
2 Meg Sample buffer	X	1	Acquire		Acquire 902 Base +0
Lower buffer	0	0	Transfer		Acquire 902 Base+0
Upper buffer	1	0	Acquire	Transfer	902Base+0x200000

In order to read data out of the 2 Meg Sample buffer, data acquisition must be turned off by clearing the ARM bit to zero (D0 in register 0x24) in the Acquisition Control Register.

The Trigger Sample Pointer and the Last Sample pointer registers give memory addresses in units of WORDS relative to the DBS902 Base address. If the trigger sample happens to be at the first location of the Upper buffer, The trigger sample pointer will read 0x100000 and when the VXI host reads the data it should look for the trigger sample at 902 Base+ 200000.

3.3 Tagging Samples using Digital Inputs

The DBS902 has two digital inputs on the DB26 Control connector that can be recorded into memory at the same time as the ADC data. These can be used to mark data values so that they may be correlated to external events. The two inputs are called SYNC1 and SYNC2 and are located on pin 19 and 20 respectively, for module A and on pins 25 and 26 for module B. They can be enabled by setting the AUX_ENA bit (D14) in the Input Control register (0x20).

The digital inputs are sampled on the rising edge of the clock and have a pipeline delay matched to the AD converter. SYNC1 will be acquired into memory in bit position D0. The 14 bit ADC data is acquired into memory at bit positions D2 thru D15. Bit D1 is used to record one of 4 inputs that can be selected by a two-bit field (D12,D11) in the Level A Mode Register (0x2C). The four possible input selections are:

	Level A Mode Register	
Recorded into Bit D1	D12	D11
ADC Over/Under Range	0	0
Digital Input SYNC2	0	1
DBS9900 Trigger	1	0
Data Trigger	1	1

The ADC Over/Under Range bit is normally low if the ADC data is within the numerical range of the AD converter. If the analog signal is outside the range of the AD converter, this bit will go hi on a sample-by-sample basis. This feature restores the two end codes of the ADC to useful data points since they need not be assumed to be "in saturation" unless the Over Range bit is also set.

The 9900 trigger and Data trigger inputs can be used as another way to confirm the location of the external trigger input, or the data sample that generated the trigger condition.

The second digital input, SYNC2 works the same as SYNC1. An example of how this may be used could be in the area of engine testing. Consider an engine with a pressure sensor in the exhaust manifold of one of the pistons and a digital shaft encoder on the crankshaft set to output a single index pulse when the piston is at the top of its stroke. The pressure is digitized by the ADC and the encoder index pulse is wired to SYNC1. Data is acquired while the engine is run at various speeds. The pressure can be linked to the crankshaft position by examining the data for samples that have D0 set. If another encoder is mounted onto the opposite end of the crankshaft, and its index pulse is wired to SYNC2, then torsion in the crankshaft can be measured by measuring the time skew between the D0 and D1 bits in memory.

3.4 Interrupt Generation on the DBS902

The Interrupt Mask register (0x36) can be used to enable or disable any of four events to cause an interrupt to the VXI host processor. The Mask Register is also used to check the status of the DBS902 when an interrupt is detected. A typical use for this is for the 902 to generate an interrupt when it has recorded the required number of data points. This is the "DONE" interrupt. It saves the processor from checking the DONE status by reading the Status Register (0x22) in a loop all the time the data is being acquired. This frees the processor to do some other useful tasks. When the interrupt occurs, the processor can switch the data buffers and start a new acquisition with the interrupt enabled when the 902 is DONE again.

A typical interrupt cycle has four phases:

- Initialization
- Start
- Interrupt status/ID
- Device specific service

For the above example of a DONE interrupt from a DBS902 in module A position, the specific software operations would be as follows:

Initialization

1. Write the desired interrupt level (1 thru 7) to the Interrupt Level Register (0x10).
2. Write the desired 8 bit interrupt vector (0xAA) to the Vector Register (0x12).
3. Initialize host processor resources to map the interrupt so that it may be recognized.
4. Enable DONE interrupt by setting D0 in the Interrupt Mask register on the DBS902 in Module A position.
5. Set desired operating modes and parameters for data acquisition on the DBS902.

Start

1. ARM by setting D0 in the Acquisition Control Register (0x24).
2. Trigger the 902 by software trigger or with an external input

Interrupt Status/ID

1. The DBS902 will interrupt through the DBS9900 when it is DONE.
2. The VXI interrupt handler will respond by reading the Status/ID from the highest priority interrupter. Assuming that this 9900 has the highest priority, it will read the Vector and use it as a component of an address to jump to a service routine that is specific to this DBS9900 device. The Vector is a ten-bit value where D0 thru D7 are programmed during initialization. Bit D8 is used to indicate that the interrupt came from Module B and Bit D9 is used to indicate that the interrupt came from Module A. So a DBS9900 could interrupt with one of 4 unique vectors, each associated with a service routine. The table below indicates the function of each vector:

Vector	Function
0x0XX	Illegal condition
0x1XX	Module B is interrupting
0x2XX	Module A is interrupting
0x3XX	Both Module A and B are interrupting. This service routine may determine the priority between these two modules.

The vector for this interrupt would be 0x2AA, since it is Module A and the base vector written is 0xAA.

Device specific service

1. Once the VXI host processor knows which Module or modules are interrupting, it must determine the type of interrupt to perform the required operation and clear the interrupt for the next acquisition cycle. Clear D0 in the Module / Relay Select Register (0xA) to select Module A for I/O.
2. Read the Interrupt Mask Register (0x36) to determine the source of the interrupt. The register will read 0x11, indicating the DONE interrupt has occurred.
3. Clear D0 in the Interrupt Mask Register to clear the DONE interrupt source in the DBS902.
4. Set D0 again to enable the DONE interrupt for the next cycle.
5. Clear the ARM bit (D0) in the Acquisition Control register (0x24).
6. Toggle the Buffer Select bit (D6) in the Input Control Register (0x20).
7. Read the Trigger Sample Pointer and save.
8. Re - ARM by setting D0 in the Acquisition Control Register (0x24).
9. Re - Trigger the 902 by software trigger or with an external input.
10. Transfer the data just acquired using the saved value of the Trigger Sample Pointer and the Size of the data buffer.

3.5 Memory Buffer Options

The DBS902 has two 1 Meg-sample data buffers that can accept ADC data automatically. Two buffers allow data to be transferred out of one buffer by the VXI bus host computer while simultaneously acquiring ADC data into the other buffer. After all the data is transferred out, the VXI host can switch the buffers so it can transfer the next block of data. This "double buffer" architecture improves throughput.

The DBS902 can be programmed to combine both buffers into one large 2 Meg-sample buffer for seamless acquisition of 2,097,152 samples.

The DBS902 memory space reserved on the VXI bus is large enough for the entire 2 Meg-sample buffer. When using the double-buffered mode, the VXI host must use a different base address for the Upper data buffer and the Lower data buffer. DBS902 Input Control Register (0x20) must be configured so that the DBS902 memory controller will acquire data into the correct buffer. Bits D6 (Buffer Select) and D7 (Buffer Mode) of the Input Control Register are used to control buffer usage. Refer to the following table for a summary of this information.

	Buffer Select	**Buffer Mode**	**Upper**	**Lower**	**Base Address (bytes)**
2 MegSample buffer	X	1	Acquire	Acquire	902 Base+0
Lower buffer	0	0	Transfer	Acquire	902 Base+0
Upper buffer	1	0	Acquire	Transfer	902 Base+0x200000

To read data out of the 2 Meg Sample buffer, data acquisition must be turned off first. To turn off data acquisition, clear the ARM bit to zero in the Acquisition Control Register (D0 in register 0x24).

The Trigger Sample Pointer and the Last Sample pointer registers give memory addresses in units of WORDS relative to the DBS902 Base Address. If the trigger sample happens to be at the first location of the Upper buffer, The trigger sample pointer will read 0x100000 and when the VXI host reads the data it should look for the trigger sample at 902 Base+ 200000.

3.5.1 Tagging Samples using Digital Inputs

The DBS902 has two digital inputs on the DBS9900 DB26 Control connector that can be recorded into memory at the same time as the ADC data. These can be used to mark data values so that they may be correlated to external events. The two inputs are called SYNC1 and SYNC2 and are located on pin 19 and 20 respectively, for module A and on pins 25 and 26 for module B. They can be enabled by setting the AUX_ENA bit (D14) in the Input Control register (0x20).

The digital inputs are sampled on the rising edge of the clock and have a pipeline delay matched to the AD converter. SYNC1 will be acquired into memory in bit position D0. The 14 bit ADC data is acquired into memory at bit positions D2 thru D15. Bit D1 is used to record one of 4 inputs that can be selected by a two-bit field (D12, D11) in the Level A Mode Register (0x2C). The four possible input selections are:

	Level A Mode Register	
Recorded into Bit D1	**D12**	**D11**
ADC Over/Under	0	0

Range		
Digital Input SYNC2	0	1
DBS9900 Trigger	1	0
Data Trigger	1	1

The ADC Over/Under Range bit is normally low if the ADC data is within the numerical range of the AD converter. If the analog signal is outside the range of the AD converter, this bit will go hi on a sample-by-sample basis. This feature restores the two end codes of the ADC to useful data points since they need not be assumed to be "in saturation" unless the Over Range bit is also set.

The 9900 trigger and Data trigger inputs can be used as another way to confirm the location of the external trigger input, or the data sample that generated the trigger condition.

The second digital input, SYNC2 works the same as SYNC1. An example of how this may be used could be in the area of engine testing. Consider an engine with a pressure sensor in the exhaust manifold of one of the pistons and a digital shaft encoder on the crankshaft set to output a single index pulse when the piston is at the top of its stroke. The pressure is digitized by the ADC and the encoder index pulse is wired to SYNC1. Data is acquired while the engine is run at various speeds. The pressure can be linked to the crankshaft position by examining the data for samples that have D0 set. If another encoder is mounted onto the opposite end of the crankshaft, and its index pulse is wired to SYNC2, then torsion in the crankshaft can be measured by measuring the time skew between the D0 and D1 bits in memory.

3.6 Interrupt Generation

The Interrupt Mask register (0x36) can be used to enable or disable any of four events to cause an interrupt to the VXI host processor. The Mask Register is also used to check the status of the DBS902 when an interrupt is detected. A typical use for this is for the 902 to generate an interrupt when it has recorded the required number of data points. This is the "DONE" interrupt. It saves the processor from checking the DONE status by reading the Status Register (0x22) in a loop all the time the data is being acquired. This allows the processor to do some other useful task such as transfer data from another data buffer. When the interrupt occurs, the processor can switch the data buffers and start a new acquisition with the interrupt enabled when the 902 is DONE again.

A typical interrupt cycle has four phases, Initialization, Start, Interrupt status/ID, Device specific service. For the above example of a DONE interrupt from a DBS902 in module A position, the specific software operations would be as follows:

Initialization

1. Write the desired interrupt level (1 thru 7) to the Interrupt Level Register (0x10).

2. Write the desired 8 bit interrupt vector (0xAA) to the Vector Register (0x12).

3. Initialize host processor resources to map the interrupt so that it may be recognized.

4. Enable DONE interrupt by writing D0 hi in the Interrupt Mask register on the DBS902 in Module A position.

5. Set desired operating modes and parameters for data acquisition on the DBS902.

Start

1. ARM by setting D0 in the Acquisition Control Register (0x24).
2. Trigger the 902 by software trigger or with an external input

Interrupt Status/ID

1. The DBS902 will interrupt thru the DBS9900 when it is DONE

2. The VXI interrupt handler will respond by reading the Status/ID from the highest priority interrupter. Assuming that this 9900 has the highest priority, it will read the Vector and use it as a component of an address to jump to a service routine that is specific to this DBS9900 device. The Vector is a ten-bit value where D0 thru D7 are programmed during initialization. Bit D8 is used to indicate that the interrupt came from Module B and Bit D9 is used to indicate that the interrupt came from Module A. So a DBS9900 could interrupt with one of 4 unique vectors, each associated with a service routine. The table below indicates the function of each vector:

Vector	Function
0x0XX	Illegal condition
0x1XX	Module B is interrupting
0x2XX	Module A is interrupting
0x3XX	Both Module A and B are interrupting. This service routine may determine the priority between these two modules.

The vector for this interrupt would be 0x2AA, since it is Module A and the base vector written is 0xAA.

Device specific service

1. Once the VXI host processor knows which Module or modules are interrupting, it must determine the type of interrupt to perform the required operation and clear the interrupt for the next acquisition cycle. Clear D0 in the Module / Relay Select Register (0xA) to select Module A for I/O.
2. Read the Interrupt Mask Register (0x36) to determine the source of the interrupt. The register will read 0x11, indicating the DONE interrupt has occurred.
3. Clear D0 in the Interrupt Mask Register to clear the DONE interrupt source in the DBS902.
4. Set D0 again to enable the DONE interrupt for the next cycle.
5. Clear the ARM bit (D0) in the Acquisition Control register (0x24).
6. Toggle the Buffer Select bit (D6) in the Input Control Register (0x20).
7. Read the Trigger Sample Pointer and save.
8. Re - ARM by setting D0 in the Acquisition Control Register (0x24).
9. Re - Trigger the 902 by software trigger or with an external input.

Transfer the data just acquired using the saved value of the Trigger Sample Pointer and the Size of the data buffer.

4 Using the DBS902 Soft Front Panel

The DBS902 Soft Front Panel implements all of the functionality of the DBS902 Digitizer instrument.

Throughout this chapter the term SFP stands for Soft Front Panel. The DBS902 Soft Front Panel software contains 3 panels:

- DBS902 Main Panel
- DBS902 Display Panel
- DBS902 Trigger/Power Panel

These panels are accessed via the DBS9900 Main Panel as described in the next section.

4.1 Accessing the DBS902 Software

The DBS902 SFP is launched via the DBS9900 SFP Main Panel.

First, select an active DBS902 plug-in (indicated by a green light).

Then, press *Connect to an Active Instrument*. The DBS902 Main Panel will appear.

From this panel you may:

- Prepare to capture data,
- Return to the DBS9900 SFP Main Panel, or
- Access the other DBS902 panels.

4.2 DBS902 Main Panel

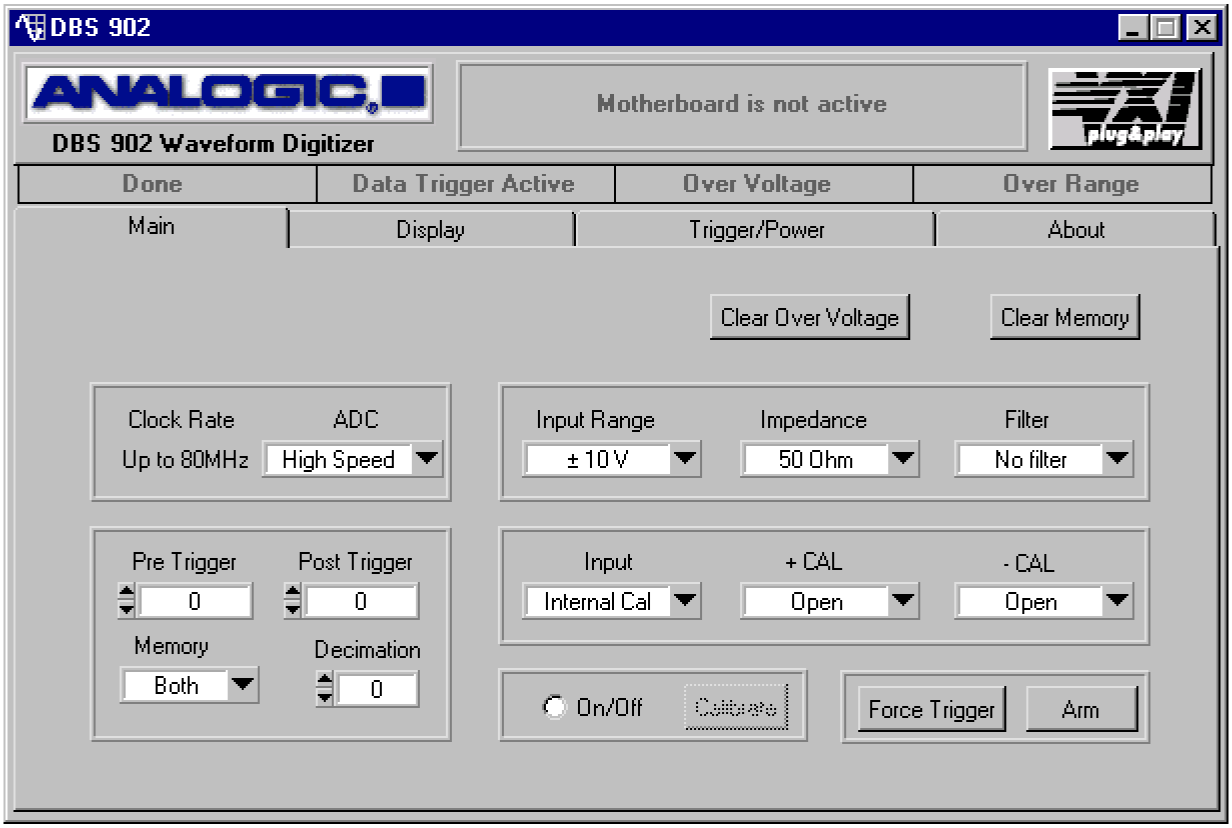


Figure 4-1 DBS902 SFP Main Panel

4.2.1 DBS902 SFP Main Panel Field Descriptions

Clear Over Voltage	Used to clear input voltage when it exceeds +/- 15V for 100mS and relays are disconnected.
Clear Memory	Fills selected memory buffer by represented value of "0" (0x0000 for signed and 0x8000 for unsigned format).
Clock Rate/ADC	Select high speed or low speed. High speed cannot be below 10MHz or above 80 MHz. Low speed cannot be below 10KHz or above 10 MHz.
Input Range	Select input range that is most compatible with input signal. Valid range from +/- 0.5V to +/- 25V.
Impedance	Dependent on input range. Software does not allow illegal impedance on a given input range. Three (3) selections: High (1Mohm), 50 Ohm and 75 Ohm.
Filter	Selects the filters to be used. Dependent on the application. Options are: No Filter, 20MHz, or 40MHz.
Pre Trigger	Controls the number of samples to be read before trigger point.
Post Trigger	Controls the number of samples to be recorded after a trigger occurs. When using a single 2M sample buffer, a value of 0x000 results in Pre-trigger Data (all data in the buffer occurs before the trigger). A value of 0x100000 results in Center trigger (trigger point is in the middle of the data set).
Memory Buffer	Select Buffer 1 (holds up to 1.024mS), Buffer 2 (holds up to 2.048mS), or both.
Decimation	Determines the number of samples to discard between stored points before recording in memory in Decimation mode. This is a 10-bit value allowing up to 1023 samples to be discarded.
Input	
+ Cal	Used to route internal reference to either or both inputs for calibration or diagnostic purposes. Three (3) selections: Open, Ground, + 2.5V or – 2.5V.
- Cal	Used to route internal reference to either or both inputs for calibration or diagnostic purposes. Three (3) selections: Open, Ground, + 2.5V or – 2.5V.
On/Off Calibrate	When on, this button stores calibration values in non-volatile memory.
Force Trigger	Starts trigger point immediately.
Arm	Starts digitizing.

4.3 DBS902 Display Panel

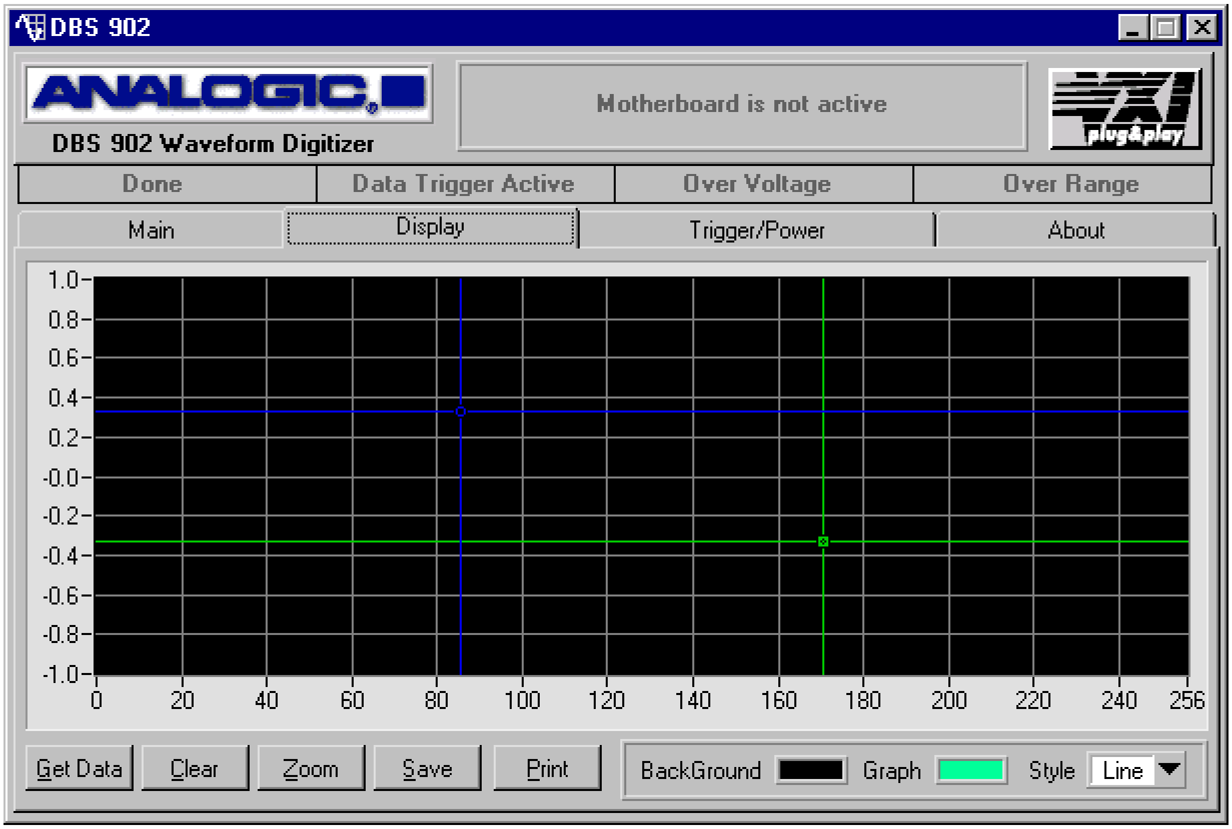


Figure 4-2 DBS902 SFP Display Panel

4.3.1 DBS902 SFP Display Panel Field Descriptions

Get Data	Reads data from the WD Memory buffer then displays the data as a graph.
Clear	Deletes graph from the screen, but graph remains in the memory.
Zoom	Allows the user to change graph scale. User can zoom in the following ways:
Horizontal/In	Gets part of the graph between two cursors (green and blue) and extends it to full screen in the Horizontal direction only.
Horizontal/Out	Changes the scale in the Horizontal direction only (Scale factor 0.5).
Vertical/In	Gets part of the graph between two cursors (green and blue) and extends it to full screen in the Vertical direction.
Vertical/Out	Changes the scale in the Vertical direction only (Scale factor 0.5).
Square/In	Gets part of the graph between two cursors (green and blue) and extends it to full screen in both directions.
Square/Out	Changes the scale in both directions (Scale factor 0.5)
Fit to Data	Returns all data on the screen to its original state.
Save	Save data as a file.
Print	Print graph in the current scale.
BackGround	Sets the background color of the graph.
Graph	Sets the LineColor (in the line mode) or PointColor (in the point mode).
Style	Plots the data in the form of a line or as a series of points.

4.4 DBS902 Trigger/Power Panel

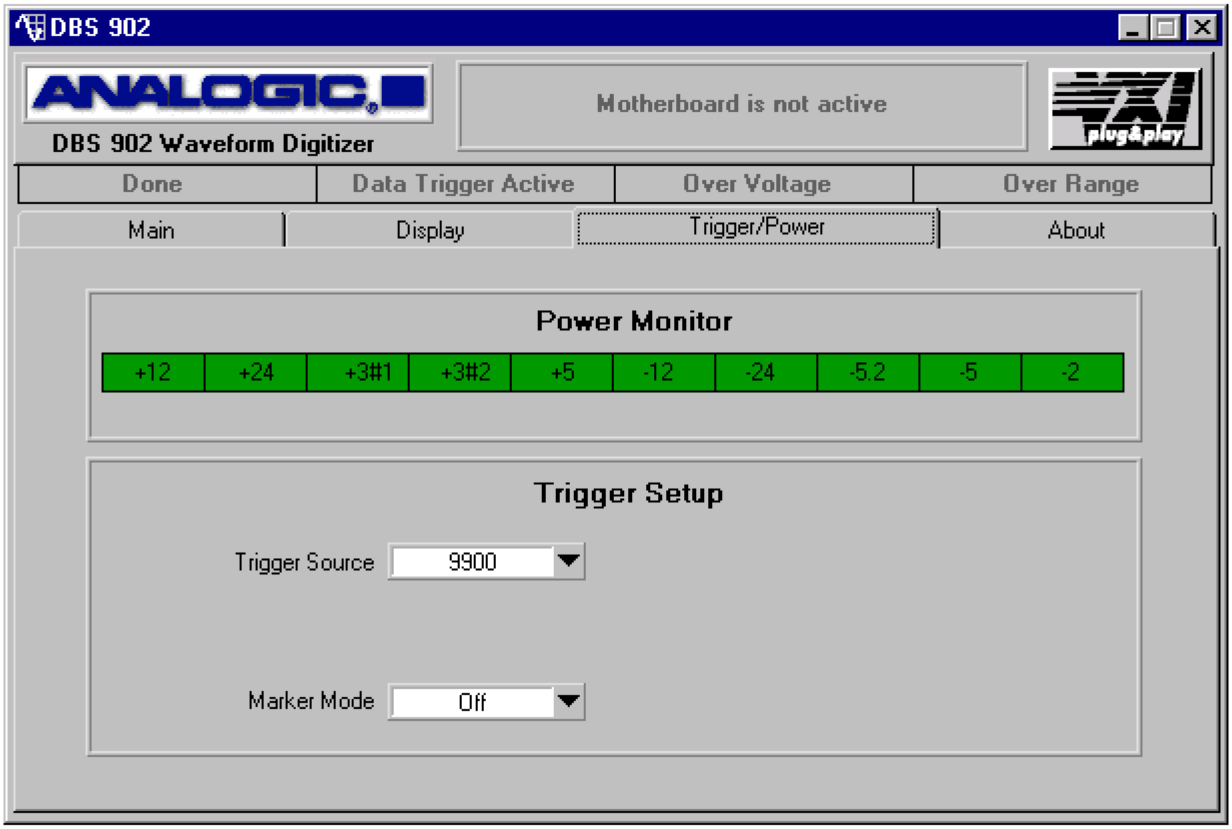
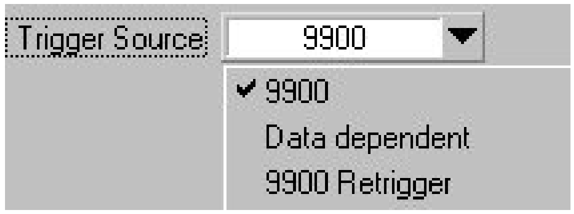
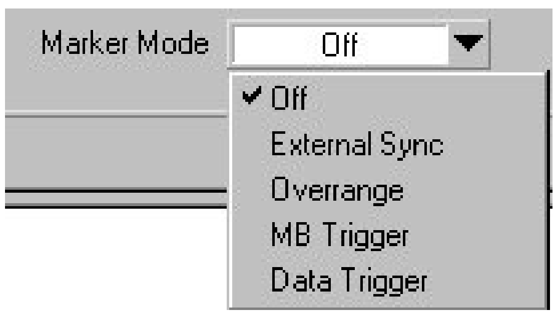
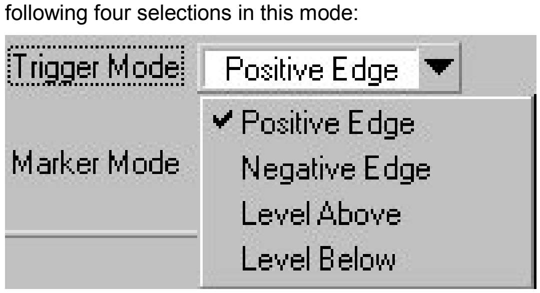


Figure 4-3 DBS902 SFP Trigger/Power Panel

The Power Monitor is contains a special circuit that is designed to monitor the power supply entering the module. Each volt has its own bit and register. The results of the power monitor circuits are stored in Status Register 22.

4.4.1 DBS902 SFP Trigger/Power Panel Field Descriptions

Trigger Source	There are three trigger sources: 
9900	WD triggered by Motherboard.
Data Dependent	WD triggered by the data that meets some conditions. If the user selects "Data Dependent" trigger, "Trigger Mode" and "Threshold" must also be inputted.
9900 Re-Trigger	Re-Trigger mode requires that the DBS9900 be set into GATE trigger mode. Acquires "bursts" of a specific number of data points at the leading edge of each GATE trigger pulse. When the Re-Trigger counter reaches zero, data stops being recorded into memory, but the trigger is still armed.
Marker Mode	Determines what data does into the two LSBs of data memory. The user may choose one of the five selections in this mode: 
Off	Selects no Markers: Bit 0,1 = 0
External Sync	Bit 0 = Sync A, Bit 1 = Sync B
Overrange	Bit 0 = Sync A, Bit 1 = Overrange Status
MB Trigger	Bit 0 = Sync A, Bit 1 = 9900 Trigger
Data Trigger	Bit 0 = Sync A, Bit 1 = Data Trigger
Trigger Mode	Option when the user selects the "Data Dependent" source. The user may choose one of the following four selections in this mode:  Hysteresis is not an option when Level Above or Below is selected.
Retrigger Size	12-bit value that allows up to 4096 samples in a burst. If the capture size register does not allow for all the samples set in the retrigger register, then acquisition will end and the last burst will be unsatisfied. The samples are acquired at the clock rate.
Input Range	Select input range that is most compatible with input signal. Valid range from +/- 0.5V to +/- 25V. User can either change the input range here or at the main panel. (*Important: user

	must change the input range before setting Threshold (V) and Hysteresis (V).
Threshold	Where trigger occurs. Range between 9 and –10.
Hysteresis	Difference between when user triggers and when user re-arms for next trigger.

5 DBS902 Driver Software - .DLL Function Definitions

5.1 Introduction

This section describes the DBS902 functions available in the shipped .dll.

WARNING

The application software that uses this driver should not try to access hardware registers directly. Using functions other than those provided by this driver to access hardware registers may end in unexpected results. However, *reading* register values or getting attribute values using VISA function calls with the Vi Session returned by 'an9900_init' is acceptable.

5.2 Initialize and Close Functions

5.2.1 Initialize

Initializes the instrument and sets it to a default configuration.

This function performs the following initialization actions:

- Opens a session to the Default Resource Manager resource and a session to the specified device using the interface and address specified in the Resource_Name control.
- Performs an identification query on the Instrument.
- Resets the instrument to a known state.
- Sends initialization commands to the instrument that sets any necessary program variables such as Headers Off, Short Command form, and Data Transfer Binary to the state necessary for the operation of the instrument driver.
- Returns an Instrument Handle which is used to differentiate between different sessions of this instrument driver.
- Each time this function is invoked a Unique Session is opened. It is possible to have more than one session open for the same resource.

5.2.2 Register Settings Upon Initialization

0x20 Input Control Register			
Bit	**Name**	**Default Setting**	**Description**
D0	Test Reg Enable	0	Disable Test Register
D1-D2	Input Source	00	Internal CAL input
D3-D5	Input Range	010	Gain 5 ± 10.0
D6	Buffer Select	0	Use Lower 1M Word buffer to acquire data
D7	Buffer Mode	1	One 2M buffer, BUF_SEL disable
D8	Filter Enable	0	Filter Disable
D9	Filter Select	0	20 MHZ LPF
D10	ADC_SEL	0	Hi Speed ADC
D11	Fill Buffer	0	Do not hold trigger
D12-D13	Term	10	50 Ω termination
D14	Aux_Ena	0	Acquire D1 and D0 into memory as zero
D15	/2s_Comp	0	(Waveform data format) ADC is coded as 2's Complement

0x24 Acquisition Trigger Control Register			
Bit	**Name**	**Default Setting**	**Description**
D0-D7		0	

0x2C Level A Mode Register			
Bit	**Name**	**Default Setting**	**Description**
D0-D15		0	

0x2E Level B Mode Register			
Bit	**Name**	**Default Setting**	**Description**
D0-D15		0	

Register Settings Upon Initialization, continue

Capture Size Register (indexed register) – should be 0	
Decimation Register	0
Retrigger Size Register	0
Interrupt Mask	0
EEProm	0
Offset Register 0x3C	0x0800
Test Data 0x3E	0

Warning: Don't do anything to the Address Generation Register!

0x30 INDEX Register			
Bit	**Name**	**Default Setting**	**Description**
D0-D2		0	

0x22 Status Register (read this last)
Status register should be read D6-D15 – each should be 1
If any are 0 – there should be a message issued to the user "-24V power supply out of tolerance" – whichever power supply has the problem.
But allow the user to Continue or Abort
FFDC hex is what it should read back.

CALIBRATION
After calibration the status register should be read to check out the power rails.

5.2.3 Initialize an902_init

Function Prototype

ViStatus **an902_init** (ViRsrc **resourceName**, ViBoolean **IDQuery**, ViBoolean **resetDevice**, ViSession ***instrumentHandle**);

an902_init		
Parameters	**Variable Type**	**Description**
<INPUT>		
resourceName	ViRsrc	This control specifies the interface and address of the device that is to be initialized (Instrument Descriptor). The exact grammar to be used in this control is shown below: Default Value = "GPIB-VXI::144" Note: Based on the Instrument Descriptor, this operation establishes a communication session with a device. The grammar for the Instrument Descriptor is shown below. Optional parameters are shown in square brackets ([]). <u>Interface</u> <u>Grammar</u> VXI VXI[board]::VXI logical address[::INSTR] GPIB-VXI GPIB-VXI[board][::GPIB-VXI primary address]::VXI logical address[::INSTR] The VXI keyword is used for VXI instruments via either embedded or MXIbus controllers. The GPIB-VXI keyword is used for a GPIB-VXI controller. <u>Optional Parameter</u> <u>Default Value</u> board 0 GPIB-VXI primary address 1
IDQuery	ViBoolean	This control specifies if an ID Query is sent to the instrument during the initialization procedure. 0 Skip Query 1 Do Query (Default Value)
resetDevice	ViBoolean	This control specifies if the instrument is to be reset to its power-on settings during the initialization procedure. 0 Don't Reset 1 Reset Device (Default Value)
*instrumentHandle	ViSession (passed by reference)	This control returns an Instrument Handle that is used in all subsequent function calls to differentiate between different sessions of this instrument driver. Note: Each time this function is invoked a Unique Session is opened. It is possible to have more than one session open for the same resource.
<OUTPUT>		
NONE		
<RETURN>		

= 0 > 0 < 0		The meaning of the VISA status returned by the function is as follows: “VI_SUCCESS” Warning: The function completed, but an exception condition occurred which may require attention. Error: The function did not complete successfully. NOTE: Use of the an902_error_message ( ) function to get a textual description of errors and warnings.

5.2.4 Close an902_close

This function accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

The instrument must be re-initialized before it can be used again.

Function Prototype

ViStatus **an902_close** (ViSession **instrumentHandle**);

an902_close		
Parameters	**Variable Type**	**Description**
<INPUT>		
instrumentHandle	ViSession	This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.
<OUTPUT>		
NONE	NONE	NONE
<RETURN>		
= 0 > 0 < 0		"VI_SUCCESS" Warning: The function completed, but an exception condition occurred which may require attention. Error: The function did not complete successfully. NOTE: Use of the an902_error_message ( ) function to get a textual description of errors and warnings.

5.2.5 Calibration **an902_Calibrate**

This function looks for offset register value to produce zero offset ADC output for each particular range (0.5 – 25V) and ADC (High/Low Speed) and stores them into EEPROM.

Function Prototype

ViStatus **an902_Calibrate** (ViSession **instrumentHandle**, char ***cData** [ ]);

an902_Calibrate		
Parameters	**Variable Type**	**Description**
<INPUT>		
instrumentHandle	ViSession	This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.
*cData [ ]	char	Current Data – ten-character string in the form MM-DD-YYYY, where MM is the month, DD is the day, and YYYY is the year.
<OUTPUT>		
None		None
<RETURN>		
= 0 > 0 < 0		"VI_SUCCESS" Warning: The function completed, but an exception condition occurred which may require attention. Error: The function did not complete successfully. NOTE: Use of the an902_error_message ( ) function to get a textual description of errors and warnings.

5.3 Signal Path Functions

5.3.1 Set Input Source an902_SetInput
Get Input Source an902_GetInput

The set function selects one of three Digitizer inputs. The get function specifies one of three inputs for the Digitizer.

Function Prototype

ViStatus **an902_SetInput** (ViSession **instrumentHandle**, AN902_INPUT_SOURCE **inputSource**);

ViStatus **an902_GetInput** (ViSession **instrumentHandle**, AN902_INPUT_SOURCE ***inputSource**);

an902_SetInput		
Parameters	**Variable Type**	**Description**
<INPUT>		
instrumentHandle	ViSession	This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.
inputSource	AN902_INPUT_SOURCE	Input Source: AN_902_INT_CAL Internal Calibration Input AN902_EXT_CAL External Calibration Input AN902_IO IO Input
<OUTPUT>		
None	None	None
<RETURN>		
= 0 >0 < 0		"VI_SUCCESS" Warning: The function completed, but an exception condition occurred which may require attention. Error: The function did not complete successfully. NOTE: Use of the an902_error_message ( ) function to get a textual description of errors and warnings.

an902_GetInput		
Parameters	**Variable Type**	**Description**
<INPUT>		
instrumentHandle	ViSession	This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.
<OUTPUT>		
*inputSource	AN902_INPUT_SOURCE (passed by reference)	Specifies Input Source: AN902_INT_CAL Internal Calibration Input AN902_EXT_CAL External Calibration Input AN902_IO IO Input
<RETURN>		
= 0		“VI_SUCCESS”
> 0		Warning: The function completed, but an exception condition occurred which may require attention.
< 0		Error: The function did not complete successfully. NOTE: Use of the an902_error_message ( ) function to get a textual description of errors and warnings.

Input Control Register (0x20), bits D1 and D2 control the input source.

	D2	D1
Internal	0	0
External	0	1
IO	1	X

Note: If the External Cal is selected – the function “an9900_moduleControl()” must be called to connect it to the proper module.

5.3.2 Set Calibration Input Status an902_SetCalMUX
Get Calibration Input Status an902_GetCalMUX

This function controls Calibration Input Relays

AN902_CAL_OPEN	Calibration Input Open
AN902_CAL_GND	Calibration Input Connected to Ground
AN902_CAL_PLUS_REF	Cal Input Connected to +2.5V Reference
AN902_CAL_MINUS_REF	Cal Input Connected to –2.5V Reference

Function Prototype

ViStatus **an902_SetCalMUX** (ViSession **instrumentHandle**, AN902_CAL_PORT_STATUS **plusCalMUX**, AN902_CAL_PORT_STATUS **minusCalMUX**);

ViStatus **an902_GetCalMUX** (ViSession **instrumentHandle**, AN902_CAL_PORT_STATUS ***plusCalMUX**, AN902_CAL_PORT_STATUS ***minusCalMUX**);

an902_SetCalMUX

Parameters	Variable Type	Description
<INPUT>		
instrumentHandle	ViSession	This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.
plusCalMUX	AN902_CAL_PORT_STATUS	AN902_CAL_OPEN — Calibration Input Open AN902_CAL_GND — Calibration Input Connected to Ground AN902_CAL_PLUS_REF — Cal Input Connected to +2.5V Reference AN902_CAL_MINUS_REF — Cal Input Connected to –2.5V Reference
minusCalMUX	AN902_CAL_PORT_STATUS	AN902_CAL_OPEN — Calibration Input Open AN902_CAL_GND — Calibration Input Connected to Ground AN902_CAL_PLUS_REF — Cal Input Connected to +2.5V Reference AN902_CAL_MINUS_REF — Cal Input Connected to –2.5V Reference
<OUTPUT>		
None	None	None
<RETURN>		
= 0		"VI_SUCCESS"
> 0		Warning: The function completed, but an exception condition occurred which may require attention.
< 0		Error: The function did not complete successfully. NOTE: Use of the an902_error_message ( ) function to get a textual description of errors and warnings.

an902_GetCalMUX

Parameters	Variable Type	Description
<INPUT>		
instrumentHandle	ViSession	This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.
<OUTPUT>		
*plusCalMUX	AN902_CAL_PORT_STATUS	AN902_CAL_OPEN — Calibration Input Open AN902_CAL_GND — Calibration Input Connected to Ground AN902_CAL_PLUS_REF — Cal Input Connected to +2.5V Reference AN902_CAL_MINUS_REF — Cal Input Connected to –2.5V Reference
*minusCalMUX	AN902_CAL_PORT_STATUS	AN902_CAL_OPEN — Calibration Input Open AN902_CAL_GND — Calibration Input Connected to Ground AN902_CAL_PLUS_REF — Cal Input Connected to +2.5V Reference AN902_CAL_MINUS_REF — Cal Input Connected to –2.5V Reference
<RETURN>		
= 0 > 0 < 0		"VI_SUCCESS" Warning: The function completed, but an exception condition occurred which may require attention. Error: The function did not complete successfully. NOTE: Use of the an902_error_message ( ) function to get a textual description of errors and warnings.

0x24 Register bits D4, D5, D6, D7 control the internal calibration input relays.

	- CAL_MUX		+ CAL_MUX	
	D7	D6	D5	D4
Open	0	0	0	0
Ground	0	1	0	1
+2.5V	1	0	1	0
-2.5 V	1	1	1	1

5.3.3 Set Lowpass Filter Frequency — an902_setFilter
Get Lowpass Filter Frequency — an902_getFilter

The set function sets the internal lowpass filter. The get function gets the applied filter.

Function Prototype

ViStatus **an902_setFilter** (ViSession **instrumentHandle**, AN902_FILTER **filter**);

ViStatus **an902_getFilter** (ViSession **instrumentHandle**, AN902_FILTER ***filter**);

an902_setFilter		
Parameters	**Variable Type**	**Description**
<INPUT>		
instrumentHandle	ViSession	This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.
filter	AN902_FILTER	Specifies the filter to be applied. Predefined values: AN902_FILTER_OFF / / No filter (maximum bandwidth) AN902_FILTER_20MHZ / / 20 MHz filter AN902_FILTER_40MHZ / / 40 MHz filter
<OUTPUT>		
None	None	None
<RETURN>		
= 0 > 0 < 0		"VI_SUCCESS" Warning: The function completed, but an exception condition occurred which may require attention. Error: The function did not complete successfully. NOTE: Use of the an902_error_message ( ) function to get a textual description of errors and warnings.

an902_getFilter		
Parameters	**Variable Type**	**Description**
<INPUT>		
instrumentHandle	ViSession	This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.
<OUTPUT>		
*filter	AN902_FILTER	Returns the applied filter. Predefined Values: AN902_FILTER_OFF / / No filter (maximum bandwidth) AN902_FILTER_20MHZ / / 20 MHz filter AN902_FILTER_40MHZ / / 40 MHz filter
<RETURN>		
= 0 > 0 < 0		"VI_SUCCESS" Warning: The function completed, but an exception condition occurred which may require attention. Error: The function did not complete successfully. NOTE: Use of the an902_error_message ( ) function to get a textual description of errors and warnings.

Input Control Register (0x20), bits D8 and D9 control the filter.

D8 set to 1, filter enable
D9 then specifies which filter

For no filter:
D8 0
D9 Don't care

5.3.4 Set Input Resistance an902_setZin
Get Input Resistance an902_getZin

The set function sets the input resistance. The get function gets the input resistance. An error should be reported if invalid Range+Impedance combination.

Function Prototype

ViStatus **an902_setZin** (ViSession **instrumentHandle**, AN902_ZIN **zin**);

ViStatus **an902_getZin** (ViSession **instrumentHandle**, AN902_ZIN ***zin**);

an902_setZin		
Parameters	**Variable Type**	**Description**
<INPUT>		
instrumentHandle	ViSession	This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.
zin	AN902_ZIN	Input resistance (ZIN) AN902_ZIN_HIZ, // Input impedance is high impedance AN902_ZIN_500HM // Input impedance is 50 ohms AN902_ZIN_750HM // Input impedance is 75 ohms
<OUTPUT>		
None	None	None
<RETURN>		
= 0 > 0 < 0		"VI_SUCCESS" Warning: The function completed, but an exception condition occurred which may require attention. Error: The function did not complete successfully. NOTE: Use of the an902_error_message ( ) function to get a textual description of errors and warnings.

an902_getZin		
Parameters	**Variable Type**	**Description**
<INPUT>		
instrumentHandle	ViSession	This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.
<OUTPUT>		
*zin	AN902_ZIN	Input resistance (ZIN) AN902_ZIN_HIZ, // Input impedance is high impedance AN902_ZIN_500HM // Input impedance is 50 ohms AN902_ZIN_750HM // Input impedance is 75 ohms
<RETURN>		
= 0 > 0 < 0		"VI_SUCCESS" Warning: The function completed, but an exception condition occurred which may require attention. Error: The function did not complete successfully. NOTE: Use of the an902_error_message ( ) function to get a textual description of errors and warnings.

Input Control Register (0x20), bits D12 and D13 controls this function. Default setting is 10 – 50 ohm termination.

5.3.5 Set Digitizer Input Range Amplitude — an902_setRange
Get Digitizer Input Range Amplitude — an902_getRange

The set function selects one of six the full-scale input voltage ranges. It applies the correct calibration corrections. The get function returns the full-scale input voltage range.
Error – should be reported if invalid Range+Impedance combination – refer to notes on previous function prohibiting low impedance settings for the +/- 25V range.

Function Prototype

ViStatus **an902_setRange** (ViSession **instrumentHandle**, AN902_RANGE **range**);

ViStatus **an902_getRange** (ViSession **instrumentHandle**, AN902_RANGE ***range**);

an902_setRange		
Parameters	**Variable Type**	**Description**
<INPUT>		
instrumentHandle	ViSession	This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.
range	AN902_RANGE	AN902_RANGE_HALF_V // Full Scale Input Range AN902_RANGE_ONE_V // Full Scale Input Range AN902_RANGE_TWO_HALF_V // Full Scale Input Range AN902_RANGE_FIVE_V // Full Scale Input Range AN902_RANGE_TEN_V // Full Scale Input Range AN902_RANGE_ TWENTY_FIVE_V // Full Scale Input Range
<OUTPUT>		
None	None	None
<RETURN>		
= 0 > 0 < 0		"VI_SUCCESS" Warning: The function completed, but an exception condition occurred which may require attention. Error: The function did not complete successfully. NOTE: Use of the an902_error_message ( ) function to get a textual description of errors and warnings.

an902_getRange		
Parameters	**Variable Type**	**Description**
<INPUT>		
instrumentHandle	ViSession	This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.
<OUTPUT>		
*range	AN902_RANGE	AN902_RANGE_HALF_V // Full Scale Input Range AN902_RANGE_ONE_V // Full Scale Input Range AN902_RANGE_TWO_HALF_V // Full Scale Input Range AN902_RANGE_FIVE_V // Full Scale Input Range AN902_RANGE_TEN_V // Full Scale Input Range AN902_RANGE_ TWENTY_FIVE_V // Full Scale Input Range
<RETURN>		
= 0 > 0 < 0		"VI_SUCCESS" Warning: The function completed, but an exception condition occurred which may require attention. Error: The function did not complete successfully. NOTE: Use of the an902_error_message ( ) function to get a textual description of errors and warnings.

Input Control Register (0x20), bits D3-D5

101	Gain 1	+/- 0.5
110	Gain 2	+/-1.0
111	Gain 3	+/- 2.5
000	Gain 4	+/- 5.0
010	Gain 5	+/- 10.0
011	Gain 6	+/- 25

5.3.6 Set ADC Select and Data Format — an902_setADC
Get ADC Select and Data Format — an902_getADC

This function selects which ADC to use – either hi or low speed and the data format desired. The HIGH speed ADC must be used when the clock (internal or external) is > 10 MHz. When the clock is <=10 MHz, the LO speed ADC must be used.

Function Prototype

ViStatus **an902_setADC** (ViSession **instrumentHandle**, AN902_ADCSELECT **adc**);

ViStatus **an902_getADC** (ViSession **instrumentHandle**, AN902_ADCSELECT ***adc**);

an902_setADC

Parameters	Variable Type	Description
<INPUT>		
instrumentHandle	ViSession	This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.
adc	AN902_ADCSELECT	Select ADC – High/Low Speed Select Data representation – 2's Complement/Offset Binary Predefined values: AN902_ADCSELECT_LOWSPEED_2SCOMP // Selects low speed ADC, 2s complement AN902_ADCSELECT_LOWSPEED_OFFSET // Selects low speed ADC, offset binary AN902_ADCSELECT_HIGHSPEED_2SCOMP // Selects high speed ADC, 2s complement AN902_ADCSELECT_HIGHSPEED_OFFSET // Selects high speed ADC, offset binary
<OUTPUT>		
None	None	None
<RETURN>		
= 0 > 0 < 0		"VI_SUCCESS" Warning: The function completed, but an exception condition occurred which may require attention. Error: The function did not complete successfully. NOTE: Use of the an902_error_message ( ) function to get a textual description of errors and warnings.

an902_getADC		
Parameters	**Variable Type**	**Description**
<INPUT>		
instrumentHandle	ViSession	This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.
<OUTPUT>		
*adc	AN902_ADCSELECT	Returns selected ADC and Data representation functions Predefined values: AN902_ADCSELECT_LOWSPEED_2SCOMP // Selects low speed ADC, 2s complement AN902_ADCSELECT_LOWSPEED_OFFSET // Selects low speed ADC, offset binary AN902_ADCSELECT_HIGHSPEED_2SCOMP // Selects high speed ADC, 2s complement AN902_ADCSELECT_HIGHSPEED_OFFSET // Selects high speed ADC, offset binary
<RETURN>		
= 0		"VI_SUCCESS"
> 0		Warning: The function completed, but an exception condition occurred which may require attention.
< 0		Error: The function did not complete successfully. NOTE: Use of the an902_error_message ( ) function to get a textual description of errors and warnings.

	2's Complement		**Offset Binary**	
	D10	**D15**	**D10**	**D15**
High Speed ADC	0	0	0	1
Low Speed ADC	1	1	1	0

5.3.7 Set Decimation Value an902_setDecimation
Get Decimation Value an902_getDecimation

This function sets/gets the decimation value. Example: If set 2, alternate data samples are discarded. Register is decvalue – 1.

Function Prototype

ViStatus **an902_setDecimation** (ViSession **instrumentHandle**, short **decimation**);

ViStatus **an902_getDecimation** (ViSession **instrumentHandle**, short ***decimation**);

an902_setDecimation		
Parameters	**Variable Type**	**Description**
<INPUT>		
instrumentHandle	ViSession	This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.
decimation	short	Specifies decimation counter value.
<OUTPUT>		
None	None	None
<RETURN>		
= 0 > 0 < 0		"VI_SUCCESS" Warning: The function completed, but an exception condition occurred which may require attention. Error: The function did not complete successfully. NOTE: Use of the an902_error_message ( ) function to get a textual description of errors and warnings.

an902_getDecimation		
Parameters	**Variable Type**	**Description**
<INPUT>		
instrumentHandle	ViSession	This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.
<OUTPUT>		
*decimation	short	Returns decimation counter value
<RETURN>		
= 0 > 0 < 0		"VI_SUCCESS" Warning: The function completed, but an exception condition occurred which may require attention. Error: The function did not complete successfully. NOTE: Use of the an902_error_message ( ) function to get a textual description of errors and warnings.

Data Registers INDEX 0x5, then 0x34 D10-D15 don't care. See page 18 in AN902 Engineering Functional Specification.

5.3.8 Set Markers / Get Markers — an902_setMarker / an902_getMarker

This function gets/sets the various marker modes. The marker modes determine what data goes into the two LSBs of data memory.

<u>Function Prototype</u>

ViStatus **an902_setMarker** (ViSession **instrumentHandle**, AN902_MARKER **marker**);

ViStatus **an902_getMarker** (ViSession **instrumentHandle**, AN902_MARKER ***marker**);

an902_setMarker		
Parameters	**Variable Type**	**Description**
<INPUT>		
instrumentHandle	ViSession	This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.
marker	AN902_MARKER	Predefined values: AN902_MARKER_OFF // 1: Selects no Markers: Bit 0,1 = 0 AN902_MARKER_SYNCB // 2: Bit 0 = Sync A, Bit 1 = Sync B AN902_MARKER_OVER // 3: Bit 0 = Sync A, Bit 1 = Overrange status AN902_MARKER_MBTRIG // 4: Bit 0 = Sync A, Bit 1 = 9900 Trigger AN902_MARKER_DATATRIG // 5: Bit 0 = Sync A, Bit 1 =Data Trigger
<OUTPUT>		
None	None	None
<RETURN>		
= 0 > 0 < 0		"VI_SUCCESS" Warning: The function completed, but an exception condition occurred which may require attention. Error: The function did not complete successfully. NOTE: Use of the an902_error_message ( ) function to get a textual description of errors and warnings.

an902_getMarker		
Parameters	**Variable Type**	**Description**
<INPUT>		
instrumentHandle	ViSession	This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.
<OUTPUT>		
*marker	An902_MARKER (passed by reference)	Predefined values: AN902_MARKER_OFF // 1: Selects no Markers: Bit 0,1 = 0 AN902_MARKER_SYNCB // 2: Bit 0 = Sync A, Bit 1 = Sync B AN902_MARKER_OVER // 3: Bit 0 = Sync A, Bit 1 = Overrange status AN902_MARKER_MBTRIG // 4: Bit 0 = Sync A, Bit 1 = 9900 Trigger AN902_MARKER_DATATRIG // 5: Bit 0 = Sync A, Bit 1 =Data Trigger
<RETURN>		
= 0 > 0 < 0		"VI_SUCCESS" Warning: The function completed, but an exception condition occurred which may require attention. Error: The function did not complete successfully. NOTE: Use of the an902_error_message ( ) function to get a textual description of errors and warnings.

Part of Level A Mode Register 0x2C D11-D12

	0x20 Input Control Register	0x2C Level A Mode Register		
	D14	**D11**	**D12**	
1	**0**			
2	**1**	**1**	**0**	**SyncB**
3	**1**	**0**	**0**	**Over**
4	**1**	**0**	**1**	**MBTrig**
5	**1** DataTrig	**1**	**1**	

5.3.9 Get Range Scale — an902_GetRangeScale

Gets data range:

AN902_RANGE_HALF_V

AN902_RANGE_ONE_V

AN902_RANGE_TWO_HALF_V

AN902_RANGE_FIVE_V

AN902_RANGE_TEN_V

AN902_RANGE_TWENTY_FIVE_V

Calculate Data scale and Max Amplitude

Int Data/float Scale = float Voltage

Returns VI_SUCCESS or INCORRECT_INPUT_RANGE

Function Prototype

ViStatus **an902_GetRangeScale** (ViSession **instrumentHandle**, AN902_RANGE **range**, float ***scale**, float ***amplitude**);

an902_GetRangeScale		
Parameters	**Variable Type**	**Description**
<INPUT>		
instrumentHandle	ViSession	This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.
range	AN902_RANGE	AN902_RANGE_HALF_V AN902_RANGE_ONE_V AN902_RANGE_TWO_HALF_V AN902_RANGE_FIVE_V AN902_RANGE_TEN_V AN902_RANGE_TWENTY_FIVE_V
<OUTPUT>		
***scale**	float	Allows convert data to Voltage format Int Data/ float Scale = float Voltage
***amplitude**	float	Max Amplitude for selected range (float representation)
<RETURN>		
= 0 > 0 < 0		"VI_SUCCESS" Warning: The function completed, but an exception condition occurred which may require attention. Error: The function did not complete successfully. NOTE: Use of the an902_error_message ( ) function to get a textual description of errors and warnings.

5.4 TRIGGER FUNCTIONS

5.4.1 SW Trigger an902_SW_Trigger

This function is called to set/clear software interrupt.

Function Prototype

ViStatus **an902_SW_Trigger** (ViSession **instrumentHandle**, int **module**, int **state**);

an902_SW_Trigger		
Parameters	**Variable Type**	**Description**
<INPUT>		
instrumentHandle	ViSession	This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.
module	Int	module A (I/O 1) – 0 module B (I/O 2) – 1
state	int	0 – Clear Software Trigger 1 – Set Software Trigger
<OUTPUT>		
None	None	None
<RETURN>		
= 0 > 0 < 0		"VI_SUCCESS" Warning: The function completed, but an exception condition occurred which may require attention. Error: The function did not complete successfully. NOTE: Use of the an902_error_message ( ) function to get a textual description of errors and warnings.

0x2E D5=1

5.4.2 Set Trigger Source an902_setTrigSrc
Get Trigger Source an902_getTrigSrc

This function sets/gets the trigger source – data dependent, 9900 trigger, or 9900 with the re-trigger option.

Function Prototype

ViStatus **an902_setTrigSrc** (ViSession **instrumentHandle**, AN902_TRIGSRC **source**);

ViStatus **an902_getTrigSrc** (ViSession **instrumentHandle**, AN902_TRIGSRC ***source**);

an902_setTrigSrc		
Parameters	**Variable Type**	**Description**
<INPUT>		
instrumentHandle	ViSession	This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.
source	AN902_TRIGSRC	Predefined values: AN902_TRIGSRC_9900 AN902_TRIGSRC_DATADEP AN902_TRIGSRC_9900RETRIG
<OUTPUT>		
None	None	None
<RETURN>		
= 0 > 0 < 0		"VI_SUCCESS" Warning: The function completed, but an exception condition occurred which may require attention. Error: The function did not complete successfully. NOTE: Use of the an902_error_message ( ) function to get a textual description of errors and warnings.

an902_getTrigSrc		
Parameters	**Variable Type**	**Description**
<INPUT>		
instrumentHandle	ViSession	This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.
<OUTPUT>		
*source	AN902_TRIGSRC (passed by reference)	AN902_TRIGSRC_9900 AN902_TRIGSRC_DATADEP AN902_TRIGSRC_9900RETRIG
<RETURN>		
= 0 > 0 < 0		"VI_SUCCESS" Warning: The function completed, but an exception condition occurred which may require attention. Error: The function did not complete successfully. NOTE: Use of the an902_error_message ( ) function to get a textual description of errors and warnings.

Notes:

Data dependent triggering,	0x24. If	D1=1 then	D2 MUST be 0.
9900 triggering	0x24	D1=0	D2=0
9900 re-triggering	0x24	D1=0	D2=1

5.4.3 Set Retrigger Size an902_setRetriggerSize

This function sets re-trigger size. Determines the number of samples that are acquired in a burst in Retrigger mode.

Function Prototype
ViStatus **an902_setRetriggerSize** (ViSession **instrumentHandle**, short **retrigSize**);

an902_setRetriggerSize		
Parameters	**Variable Type**	**Description**
<INPUT>		
instrumentHandle	ViSession	This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.
retrigSize	short	Re-trigger size value is a 12-bit value allowing up to 4096 samples in a burst.
<OUTPUT>		
None	None	None
<RETURN>		
= 0 > 0 < 0		"VI_SUCCESS" Warning: The function completed, but an exception condition occurred which may require attention. Error: The function did not complete successfully. NOTE: Use of the an902_error_message ( ) function to get a textual description of errors and warnings.

Data Registers INDEX 0x3, then 0x34 D12-D15 don't care. See page 17 in AN902 Engineering Functional Specification.

5.4.4 Set Arm Trigger — an902_setArm

This function is called to arm the instrument in preparation for a trigger.

Function Prototype
ViStatus **an902_setArm** (ViSession **instrumentHandle**);

an902_setArm		
Parameters	**Variable Type**	**Description**
<INPUT>		
instrumentHandle	ViSession	This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.
<OUTPUT>		
None	None	None
<RETURN>		
= 0 > 0 < 0		“VI_SUCCESS” Warning: The function completed, but an exception condition occurred which may require attention. Error: The function did not complete successfully. NOTE: Use of the an902_error_message ( ) function to get a textual description of errors and warnings.

0x24 D0 = 1

5.4.5 Clear Arm an902_clearArm

This function is called to clear the arming of the instrument.

Function Prototype
ViStatus **an902_clearArm** (ViSession **instrumentHandle**);

an902_clearArm		
Parameters	**Variable Type**	**Description**
<INPUT>		
instrumentHandle	ViSession	This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.
<OUTPUT>		
None	None	None
<RETURN>		
= 0 > 0 < 0		"VI_SUCCESS" Warning: The function completed, but an exception condition occurred which may require attention. Error: The function did not complete successfully. NOTE: Use of the an902_error_message ( ) function to get a textual description of errors and warnings.

0x24 D0 = 0

5.4.6 Get Trigger Status an902_getTrigStatus

This function allows the user to determine whether the instrument is armed, triggered, and/or has completed a waveform cycle.

Function Prototype

ViStatus **an902_getTrigStatus** (ViSession **instrumentHandle**, ViBoolean ***armed**, ViBoolean ***triggered**, ViBoolean ***done**);

an902_getTrigStatus		
Parameters	**Variable Type**	**Description**
<INPUT>		
instrumentHandle	ViSession	This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.
<OUTPUT>		
*armed	ViBoolean (passed by reference)	Armed – 1 Not Armed - 0
*triggered	ViBoolean (passed by reference)	Triggered – 1 Not Triggered – 0
*done	ViBoolean (passed by reference)	Done – 1 Not Done – 0
<RETURN>		
= 0 > 0 < 0		"VI_SUCCESS" Warning: The function completed, but an exception condition occurred which may require attention. Error: The function did not complete successfully. NOTE: Use of the an902_error_message ( ) function to get a textual description of errors and warnings.

Here are two possible conditions for each parameter:

armed	VI_TRUE if 0x24 D0 = 1 VI_FALSE if 0x24 D0 = 0	
triggered	VI_TRUE if 0x22 D1 = 1 VI_FALSE if 0x22 D1 = 0	
done	VI_TRUE if 0x22 D0 = 1 VI_FALSE if 0x22 D0 = 1	// Instrument has completed acquire // Instrument busy

5.4.7 Get Module Status **an902_getModuleStatus**

This function reads the DONE status of both A/B. 9900 0x04 D8 (A) and D9 (B)

Function Prototype

ViStatus **an902_getModuleStatus** (ViSession **instrumentHandle**, ViBoolean ***Mod_A,**
ViBoolean ***Mod_B**);

an902_getModuleStatus		
Parameters	**Variable Type**	**Description**
<INPUT>		
instrumentHandle	ViSession	This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.
<OUTPUT>		
*Mod_A	ViBoolean (passed by reference)	Module A status: Done – 1 Not Done – 0
*Mod_B	ViBoolean (passed by reference)	Module B status: Done – 1 Not Done – 0
<RETURN>		
= 0 > 0 < 0		"VI_SUCCESS" Warning: The function completed, but an exception condition occurred which may require attention. Error: The function did not complete successfully. NOTE: Use of the an902_error_message ( ) function to get a textual description of errors and warnings.

Here are two possible conditions for each parameter:

Mod _A acquire	VI_TRUE if 0x4 D8 = 1	// Instrument has completed
	VI_FALSE if 0x4 D8 = 0	// Instrument busy
Mod_B acquire	VI_TRUE if 0x4 D9 = 1	// Instrument has completed
	VI_FALSE if 0x4 D9 = 0	// Instrument busy

Reduces polling of daughterboard – minimizes noise. However, a timeout mechanism should be made available so that if, for any reason, the Done bit on the motherboard is not set, then Done bit is 902 0x22 D0 is then polled. And if not done for a long time (function of clock frequency, decimation) then this probably indicates bad acquisition – user should be notified and given an option to restart acquisition.

5.5 DATA DEPENDENT TRIGGER FUNCTIONS

Note that these functions must be used in the following order because the range and mode both affect the threshold and hysteresis registers.

an902_setRange
an902_setTrigMode ( )
an902_setTrigThresh ( )
an902_setTrigHyst ( )

5.5.1 Set data-dependent trigger threshold voltage an902_setTrigThresh Get data-dependent trigger threshold voltage an902_getTrigThresh

This set function sets the trigger threshold to the nearest available value. The get function returns the actual value based on instrument registers. These functions take into account the range setting, so the range should be set first.

<u>Function Prototype</u>
ViStatus **an902_setTrigThresh** (ViSession **instrumentHandle**, float **threshold**);

ViStatus **an902_getTrigThresh** (ViSession **instrumentHandle**, float ***threshold**);

an902_setTrigThresh		
Parameters	**Variable Type**	**Description**
<INPUT>		
instrumentHandle	ViSession	This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.
threshold	float	Threshold volt
<OUTPUT>		
None	None	None
<RETURN>		
= 0 > 0 < 0		"VI_SUCCESS" Warning: The function completed, but an exception condition occurred which may require attention. Error: The function did not complete successfully. NOTE: Use of the an902_error_message ( ) function to get a textual description of errors and warnings.

an902_getTrigThresh		
Parameters	**Variable Type**	**Description**
<INPUT>		
instrumentHandle	ViSession	This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.
<OUTPUT>		
*threshold	Float (passed by reference)	Threshold volt
<RETURN>		
= 0 > 0 < 0		"VI_SUCCESS" Warning: The function completed, but an exception condition occurred which may require attention. Error: The function did not complete successfully. NOTE: Use of the an902_error_message ( ) function to get a textual description of errors and warnings.

5.5.2 Set data-dependent trigger mode — an902_setTrigMode
Get data-dependent trigger mode — an902_getTrigMode

The set function sets the data dependent trigger mode. The get function gets the data dependent trigger mode.

Function Prototype

ViStatus **an902_setTrigMode** (ViSession **instrumentHandle**, AN902_TRIGMODE **trigMode**);

ViStatus **an902_getTrigMode** (ViSession **instrumentHandle**, AN902_TRIGMODE ***trigMode**);

an902_setTrigMode		
Parameters	**Variable Type**	**Description**
<INPUT>		
instrumentHandle	ViSession	This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.
trigMode	AN902_TRIGMODE	Predefined values: AN902_TRIGMODE_EDGE_POS // Positive Slope AN902_TRIGMODE_EDGE_NEG // Negative Slope AN902_TRIGMODE_LEVEL_ABOVE // Above trigger level AN902_TRIGMODE_LEVEL_BELOW // Below trigger level
<OUTPUT>		
None	None	None
<RETURN>		
= 0 > 0 < 0		"VI_SUCCESS" Warning: The function completed, but an exception condition occurred which may require attention. Error: The function did not complete successfully. NOTE: Use of the an902_error_message ( ) function to get a textual description of errors and warnings.

an902_getTrigMode		
Parameters	**Variable Type**	**Description**
<INPUT>		
instrumentHandle	ViSession	This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.
<OUTPUT>		
*trigMode	AN902_TRIGMODE (passed by reference)	Returns trigger mode. Predefined values: AN902_TRIGMODE_EDGE_POS // Positive Slope AN902_TRIGMODE_EDGE_NEG // Negative Slope AN902_TRIGMODE_LEVEL_ ABOVE // Above trigger level AN902_TRIGMODE_LEVEL_ BELOW // Below trigger level
<RETURN>		
= 0 > 0 < 0		“VI_SUCCESS” Warning: The function completed, but an exception condition occurred which may require attention. Error: The function did not complete successfully. NOTE: Use of the an902_error_message ( ) function to get a textual description of errors and warnings.

Controlled by bits D14 and D13 in 0x2C and 0x2E

D15 in 0x2E to enable hysteresis

	Rise			Fall			Level Above			Level Below	
	D14	D13		D14	D13		D14	D13		D14	D13
Level A 0	1		1	0		1	1		0	0	
Level B 0	0		1	1		0	0		1	1	

5.5.3 Set data-dependent trigger hysteresis an902_setTrigHyst Get data-dependent trigger hysteresis an902_GetTrigHyst

The set function sets the data dependent trigger hysteresis in millivolts from 0.0 to 10.0. The get function gets the data dependent hysteresis in millivolts. Must take range into account. So the range should be set first.

Function Prototype

ViStatus **an902_setTrigHyst** (ViSession **instrumentHandle**, float **hyster**);

ViStatus **an902_GetTrigHyst** (ViSession **instrumentHandle**, float ***hyster**);

an902_setTrigHyst		
Parameters	**Variable Type**	**Description**
<INPUT>		
instrumentHandle	ViSession	This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.
hyster	float	Hysteresis in Volts
<OUTPUT>		
None	None	None
<RETURN>		
= 0 > 0 < 0		"VI_SUCCESS" Warning: The function completed, but an exception condition occurred which may require attention. Error: The function did not complete successfully. NOTE: Use of the an902_error_message ( ) function to get a textual description of errors and warnings.

an902_getTrigHyst

Parameters	Variable Type	Description
<INPUT>		
instrumentHandle	ViSession	This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.
<OUTPUT>		
*hyster	Float (passed by reference)	Hysteresis in Volts
<RETURN>		
= 0		"VI_SUCCESS"
> 0		Warning: The function completed, but an exception condition occurred which may require attention.
< 0		Error: The function did not complete successfully.
		NOTE: Use of the an902_error_message ( ) function to get a textual description of errors and warnings.

5.6 DIAGNOSTIC AND STATUS FUNCTIONS

5.6.1 Get Power Monitor an902_getPowerMonitor

This function checks whether all the power rails are OK and not. Reports back either OK (0) or a non-zero number representing bits D6-D15 in Register 0x22 – the Status Register. This will give the user the ability to check which power rails are bad.

Function Prototype

ViStatus **an902_getPowerMonitor** (ViSession **instrumentHandleB**, ViBoolean ***powerOK**, unsigned int ***psMap**);

an902_getPowerMonitor		
Parameters	**Variable Type**	**Description**
<INPUT>		
instrumentHandle	ViSession	This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.
<OUTPUT>		
*powerOK	ViBoolean (passed by reference)	power = 0 Power OK power = 1 Trouble
*psMap	unsigned int (passed by reference)	Represents bits D6-D15 in Register 0x22 – the Status Register 1 - OK 0 – Trouble D6 +12 volt supply D7 +24 volt supply D8 +3.3 volt supply # 1 D9 +3.3 volt supply # 2 D10 +5 volt analog supply D11 -12 volt supply D12 -24 volt supply D13 -5.2 volt supply D14 -5 volt analog supply D15 -2 volt supply
<RETURN>		
= 0 > 0 < 0		"VI_SUCCESS" Warning: The function completed, but an exception condition occurred which may require attention. Error: The function did not complete successfully. NOTE: Use of the an902_error_message ( ) function to get a textual description of errors and warnings.

Note: Put in CRC check if D4 OK (If DC OK is OK) but voltage bits not OK – D6-D15 must all be set. A notification should be sent if hardware problem but allow to Continue or Abort. Should tell the user which power supply is not OK.

5.6.2 Get Input Voltage Status — an902_getInpVoltStat

This function determines the status of the input voltage – either OK, an input trip has occurred, or voltage is over-range. If both conditions occur, the INPUTTRIP is reported since it is more important. Note that it would be a good idea to check for INPUTTRIP prior to each acquisition.

Function Prototype
ViStatus **an902_getInpVoltStat** (ViSession **instrumentHandle**; AN902_VOLTSTAT ***vStatus**);

an902_getInpVoltStat		
Parameters	**Variable Type**	**Description**
<INPUT>		
instrumentHandle	ViSession	This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.
<OUTPUT>		
*vStatus	AN902_VOLTSTAT (passed by reference)	Returns status. Predefined values: AN902_VOLTSTAT_OK // OK AN902_VOLTSTAT_INPUTTRIP // Input voltage trip detected AN902_VOLTSTAT_OVERRANGE // Over voltage range detected
<RETURN>		
= 0		"VI_SUCCESS"
> 0		Warning: The function completed, but an exception condition occurred which may require attention.
< 0		Error: The function did not complete successfully. NOTE: Use of the an902_error_message ( ) function to get a textual description of errors and warnings.

Note: Input trip – D3 of 0x22 Status register – if it's cleared – problem OverRange – D5 of 0x22 Status register – input voltage overrange – if it's set – problem.

5.6.3 Clear Input Voltage Trip — an902_clearVoltTrip

This function clears the input voltage trip. Clears D3 0x22. Can be done by reading 0x20 and writing it back out (rewrite the contents).

Function Prototype

ViStatus **an902_clearVoltTrip** (ViSession **instrumentHandle**);

an902_clearVoltTrip		
Parameters	**Variable Type**	**Description**
<INPUT>		
instrumentHandle	ViSession	This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.
<OUTPUT>		
None	None	None
<RETURN>		
= 0 > 0 < 0		"VI_SUCCESS" Warning: The function completed, but an exception condition occurred which may require attention. Error: The function did not complete successfully. NOTE: Use of the an902_error_message ( ) function to get a textual description of errors and warnings.

5.7 MEMORY FUNCTIONS

5.7.1 Clear Memory to mid-Scale an902_clearMemory

This function clears the memory buffers, as specified. It takes into account the data format and clears to 0x0000 for signed, and 0x8000 for unsigned format (offset binary).

Function Prototype
ViStatus **an902_clearMemory** (ViSession **instrumentHandle**; AN902_CLEAR _MEM **buffer**);

an902_clearMemory		
Parameters	**Variable Type**	**Description**
<INPUT>		
instrumentHandle	ViSession	This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.
buffer	AN902_CLEAR_MEM	Buffer to clear Predefined values: AN902_CLEAR_MEM_BUF1 // Clear memory Buffer 1 (Hi) AN902_CLEAR_MEM_BUF2 // Clear memory Buffer 2 (Lo) AN902_CLEAR_MEM_BOTH // Clear entire memory Buffer
<OUTPUT>		
None	None	None
<RETURN>		
= 0 > 0 < 0		"VI_SUCCESS" Warning: The function completed, but an exception condition occurred which may require attention. Error: The function did not complete successfully. NOTE: Use of the an902_error_message ( ) function to get a textual description of errors and warnings.

5.7.2 Set Number of Samples — an902_setNumSamples
Get Number of Samples — an902_getNumSamples

The set function specifies the number of pre-trigger or post-trigger samples that will be acquired. The get function returns an error if PreTrig+PostTrig samples exceed 2M or 1M in 1M mode.

Function Prototype

ViStatus **an902_setNumSamples** (ViSession **instrumentHandle,** unsigned int **numPreSamp**, unsigned int **numPostSamp**);

ViStatus an902_**getNumSamples** (ViSession **instrumentHandle**, unsigned int ***numPreSamp**, unsigned int ***numPostSamp**);

an902_setNumSamples		
Parameters	**Variable Type**	**Description**
<INPUT>		
instrumentHandle	ViSession	This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.
numPreSamp	unsigned int	Number of samples before trigger point Just Global variable There is no hardware to keep this data
numPostSamp	unsigned int	Number of samples after trigger point
<OUTPUT>		
None	None	None
<RETURN>		
= 0 > 0 < 0		"VI_SUCCESS" Warning: The function completed, but an exception condition occurred which may require attention. Error: The function did not complete successfully. NOTE: Use of the an902_error_message ( ) function to get a textual description of errors and warnings.

an902_getNumSamples		
Parameters	**Variable Type**	**Description**
<INPUT>		
instrumentHandle	ViSession	This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.
<OUTPUT>		
*numPreSamp	unsigned int (passed by reference)	Number of samples before trigger point Just Global variable There is no hardware to keep this data
*numPostSamp	unsigned int (passed by reference)	Number of samples after trigger point
<RETURN>		
= 0 > 0 < 0		"VI_SUCCESS" Warning: The function completed, but an exception condition occurred which may require attention. Error: The function did not complete successfully. NOTE: Use of the an902_error_message ( ) function to get a textual description of errors and warnings.

numPostSamp = number of post sample data points – 1
numPreSamp = must equal the number of pre-trigger samples directly

5.7.3 Get Trigger Sample an902_getTrigSample

Gets trigger sample position in 2MS space.

Function Prototype

ViStatus **an902_getTrigSample** (ViSession **instrumentHandle**, unsigned int ***trigSample**);

an902_getTrigSample		
Parameters	**Variable Type**	**Description**
<INPUT>		
instrumentHandle	ViSession	This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.
<OUTPUT>		
*trigSample	unsigned int	Trigger sample position in 2MS space.
<RETURN>		
= 0 > 0 < 0		"VI_SUCCESS" Warning: The function completed, but an exception condition occurred which may require attention. Error: The function did not complete successfully. NOTE: Use of the an902_error_message ( ) function to get a textual description of errors and warnings.

5.7.4 Get Last Sample an902_getLastSample

Gets last sample position in 2MS space.

Function Prototype

ViStatus **an902_getLastSample** (ViSession **instrumentHandle**, unsigned int ***lastSample**);

an902_getLastSample		
Parameters	**Variable Type**	**Description**
<INPUT>		
instrumentHandle	ViSession	This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.
<OUTPUT>		
*lastSample	unsigned int	Last sample position in 2MS space.
<RETURN>		
= 0 > 0 < 0		“VI_SUCCESS” Warning: The function completed, but an exception condition occurred which may require attention. Error: The function did not complete successfully. NOTE: Use of the an902_error_message ( ) function to get a textual description of errors and warnings.

5.7.5 Set Buffer Size an902_setBuffSize
Get Buffer Size an902_getBuffSize

The set function specifies either two 1MS buffers with switching or to use the entire 2MS buffer at once. The get function determines MAX Buffer size selected (1 or 2M)

Function Prototype

ViStatus **an902_set BuffSize** (ViSession **instrumentHandle**; AN902_BUFFSIZE **size**);

ViStatus **an902_getBuffSize** (ViSession **instrumentHandle**; AN902_BUFFSIZE ***size**);

an902_setBuffSize		
Parameters	**Variable Type**	**Description**
<INPUT>		
instrumentHandle	ViSession	This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.
size	AN902_BUFFSIZE	Predefined values: AN902_BUFFSIZE_1M // Use two 1M buffers, buffer switching AN902_BUFFSIZE_2M // Use one 2M buffer
<OUTPUT>		
None	None	None
<RETURN>		
= 0 > 0 < 0		"VI_SUCCESS" Warning: The function completed, but an exception condition occurred which may require attention. Error: The function did not complete successfully. NOTE: Use of the an902_error_message ( ) function to get a textual description of errors and warnings.

an902_getBuffSize		
Parameters	**Variable Type**	**Description**
<INPUT>		
instrumentHandle	ViSession	This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.
<OUTPUT>		
*size	AN902_BUFFSIZE (passed by reference)	Predefined values: AN902_BUFFSIZE_1M // Use two 1M buffers, buffer switching AN902_BUFFSIZE_2M // Use one 2M buffer
<RETURN>		
= 0 > 0 < 0		"VI_SUCCESS" Warning: The function completed, but an exception condition occurred which may require attention. Error: The function did not complete successfully. NOTE: Use of the an902_error_message ( ) function to get a textual description of errors and warnings.

5.7.6 Set Acquisition Buffer — an902_setAcqBuff
Get Acquisition Buffer — an902_getAcqBuff

The set function sets which one of the two buffers to acquire data to. It returns an error if 2M buffer size has been selected. The get function gets which one of the two buffers to acquire data to.

Function Prototype

ViStatus **an902_set AcqBuff** (ViSession **instrumentHandle**; AN902_ACQBUFF **bufferN**);

ViStatus **an902_getAcqBuff** (ViSession **instrumentHandle**; AN902_ACQBUFF ***bufferN**);

an902_setAcqBuff		
Parameters	**Variable Type**	**Description**
<INPUT>		
instrumentHandle	ViSession	This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.
bufferN	AN902_ACQBUFF	Predefined values: AN902_ACQBUFF_BUFF1 // Acquire to low buffer AN902_ACQBUFF_BUFF2 // Acquire to high buffer
<OUTPUT>		
None	None	None
<RETURN>		
= 0 > 0 < 0		"VI_SUCCESS" Warning: The function completed, but an exception condition occurred which may require attention. Error: The function did not complete successfully. NOTE: Use of the an902_error_message ( ) function to get a textual description of errors and warnings.

an902_getAcqBuff		
Parameters	**Variable Type**	**Description**
<INPUT>		
instrumentHandle	ViSession	This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.
<OUTPUT>		
*bufferN	AN902_ACQBUFF (passed by reference)	Predefined values: AN902_ACQBUFF_BUFF1 // Acquire to low buffer AN902_ACQBUFF_BUFF2 // Acquire to high buffer
<RETURN>		
= 0		"VI_SUCCESS"
> 0		Warning: The function completed, but an exception condition occurred which may require attention.
< 0		Error: The function did not complete successfully. NOTE: Use of the an902_error_message ( ) function to get a textual description of errors and warnings.

5.7.7 Get Buffer Data an902_getBuffData

This function reads data from the buffer. It outputs a pointer to the data, the data size (in samples), and the trigger point (in samples). All parameters are with respect to trigger point, not absolute address. If markers are enabled by SetMarker ( ) then the markers are in the two LSBs of the data.

Function Prototype

ViStatus **an902_getBuffData** (ViSession **instrumentHandle**, unsigned int **dataSize**, unsigned int **firstSample**, ViInt16 ***data**)

an902_getBuffData		
Parameters	**Variable Type**	**Description**
<INPUT>		
instrumentHandle	ViSession	This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.
dataSize	unsigned int	Data size in samples
firstSample	unsigned int	Trigger Point in samples
<OUTPUT>		
*data	ViInt16 (passed by reference)	Pointer to memory allocated to keeping data
<RETURN>		
= 0 > 0 < 0		"VI_SUCCESS" Warning: The function completed, but an exception condition occurred which may require attention. Error: The function did not complete successfully. NOTE: Use of the an902_error_message ( ) function to get a textual description of errors and warnings.

5.8 Utility Functions

5.8.1 Read Register an902_ReadRegister

This function get a register wanted to read and reads its contents

Input Register

Output Register value

Returns Status of operation

Function Prototype

ViStatus **an902_ReadRegister** (ViSession **instrumentHandle**, AN902_REG_NUM **register**, unsigned int ***value**);

an902_ReadRegister		
Parameters	**Variable Type**	**Description**
<INPUT>		
instrumentHandle	ViSession	This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.
register	AN902_REG_NUM	This input determines which register is written to. Valid Range: AN9900_STATUS_REG 0x04 AN902_INPUT_CONTROL_REG 0x20 AN902_STATUS_REG 0x22 AN902_ACQUISITION_CONTROL REG 0x24 AN902_LEVEL_A_MODE_REG 0x2C AN902_LEVEL_B_MODE_REG 0X2E AN902_INDEX_REG 0X30 AN902_DATA_MSB_REG 0X32 AN902_DATA_LSB_REG 0X34 AN902_INTERRUPT_MASK_REG 0X36 AN902_EEPROM_REG 0X38 AN902_DC_OFFSET_REG 0X3C AN902_TEST_DATA_REG 0X3E
<OUTPUT>		
*value	unsigned int	Displays the hex value of byte queried. Valid Range: 0x00 – 0xFFFF

<RETURN>		
= 0		"VI_SUCCESS"
> 0		Warning: The function completed, but an exception condition occurred which may require attention.
< 0		Error: The function did not complete successfully. NOTE: Use of the an902_error_message ( ) function to get a textual description of errors and warnings.

5.8.2 Write Register an902_WriteRegister

This function writes data to an instrument register to modify device settings.

Function Prototype
ViStatus **an902_WriteReigster** (ViSession **instrumentHandle**, AN902_REG_NUM **register**, unsigned int **value**);

an902_WriteRegister		
Parameters	**Variable Type**	**Description**
<INPUT>		
instrumentHandle	ViSession	This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.
register	AN902_REG_NUM	This input determines which register is written to. Valid Range: AN9900_STATUS_REG 0x04 AN902_INPUT_CONTROL_REG 0x20 AN902_ACQUISITION_CONTROL REG 0x24 AN902_LEVEL_A_MODE_REG 0x2C AN902_LEVEL_B_MODE_REG 0X2E AN902_INDEX_REG 0X30 AN902_DATA_MSB_REG 0X32 AN902_DATA_LSB_REG 0X34 AN902_INTERRUPT_MASK_REG 0X36 AN902_EEPROM_REG 0X38 AN902_DC_OFFSET_REG 0X3C AN902_TEST_DATA_REG 0X3E
value	unsigned int	This control writes data (in hex) to a register. Valid Range: 0x00 – 0xFFFF
<OUTPUT>		
None		None
<RETURN>		
= 0 > 0 < 0		"VI_SUCCESS" Warning: The function completed, but an exception condition occurred which may require attention. Error: The function did not complete successfully. NOTE: Use of the an902_error_message ( ) function to get a textual description of errors and warnings.

5.8.3 Set Register an902_SetRegister

This function writes data to an instrument register to modify device settings.

Function Prototype

ViStatus **an902_SetRegister** (ViSession **instrumentHandle**, AN902_REG_NUM **register**, unsigned int mask, unsigned int **value**);

an902_SetRegister		
Parameters	**Variable Type**	**Description**
<INPUT>		
instrumentHandle	ViSession	This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.
register	AN902_REG_NUM	This input determines which register is written to. Valid Range: AN9900_STATUS_REG 0x04 AN902_INPUT_CONTROL_REG 0x20 AN902_ACQUISITION_CONTROL REG 0x24 AN902_LEVEL_A_MODE_REG 0x2C AN902_LEVEL_B_MODE_REG 0X2E AN902_INDEX_REG 0X30 AN902_DATA_MSB_REG 0X32 AN902_DATA_LSB_REG 0X34 AN902_INTERRUPT_MASK_REG 0X36 AN902_EEPROM_REG 0X38 AN902_DC_OFFSET_REG 0X3C AN902_TEST_DATA_REG 0X3E
mask	unsigned int	This control masks the data (in hex) to write to a register. Valid Range: 0x00 – 0xFFFF
value	unsigned int	This control writes data (in hex) to a register. Valid Range: 0x00 – 0xFFFF
<OUTPUT>		
None		None
<RETURN>		

= 0 > 0 < 0		"VI_SUCCESS" Warning: The function completed, but an exception condition occurred which may require attention. Error: The function did not complete successfully. NOTE: Use of the an902_error_message ( ) function to get a textual description of errors and warnings.

5.8.4 Read EEPROM an902_readEEPROM

Communicate to specified EEPROM register.

Available commands:

READ 3 Reads data from EEPROM

WRITE 5 Writes data to EEPROM

EWEN 1 Enables EEPROM for writing by sending EWEN to register 0xFF

EWDS 1 Disables EEPROM for writing by sending EWDS to register 0x00

Function Prototype

ViStatus **an902_readEEPROM** (ViSession **instrumentHandle**, unsigned int **EEPROMAddress**, unsigned int ***EEPROMData**);

an902_readEEPROM		
Parameters	**Variable Type**	**Description**
<INPUT>		
instrumentHandle	ViSession	This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.
EEPROMAddress	unsigned int	EEPROM Address
<OUTPUT>		
*EEPROMData	unsigned int	Value stored into EEPROM
<RETURN>		
= 0 > 0 < 0		“VI_SUCCESS” Warning: The function completed, but an exception condition occurred which may require attention. Error: The function did not complete successfully. NOTE: Use of the an902_error_message ( ) function to get a textual description of errors and warnings.

5.8.5 Write EEPROM an902_writeEEPROM

Communicate to specified EEPROM register.

Available commands:

READ 3 Reads data from EEPROM

WRITE 5 Writes data to EEPROM

EWEN 1 Enables EEPROM for writing by sending EWEN to register 0xFF

EWDS 1 Disables EEPROM for writing by sending EWDS to register 0x00

Function Prototype

ViStatus **an902_writeEEPROM** (ViSession **instrumentHandle**, unsigned int **EEPROMAddress**, unsigned int **EEPROMData**);

an902_writeEEPROM		
Parameters	**Variable Type**	**Description**
<INPUT>		
instrumentHandle	ViSession	This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.
EEPROMAddress	unsigned int	EEPROM Address
EEPROMData	unsigned int	Specifies the data value to write into EEPROM
<OUTPUT>		
None		None
<RETURN>		
= 0 > 0 < 0		"VI_SUCCESS" Warning: The function completed, but an exception condition occurred which may require attention. Error: The function did not complete successfully. NOTE: Use of the an902_error_message ( ) function to get a textual description of errors and warnings.

5.8.6 Enable EEPROM an902_enableEEPROM

Communicate to specified EEPROM register.

Available commands:

READ 3 Reads data from EEPROM

WRITE 5 Writes data to EEPROM

EWEN 1 Enables EEPROM for writing by sending EWEN to register 0xFF

EWDS 1 Disables EEPROM for writing by sending EWDS to register 0x00

Function Prototype
ViStatus **an902_enableEEPROM** (ViSession **instrumentHandle**, unsigned int **command**);

an902_enableEEPROM		
Parameters	**Variable Type**	**Description**
<INPUT>		
instrumentHandle	ViSession	This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.
command	unsigned int	Available commands: Disables DISABLE_EEPROM 0 Disables EEPROM for writing ENABLE_EEPROM 1 Enables EEPROM for writing
<OUTPUT>		
None		None
<RETURN>		
= 0 > 0 < 0		"VI_SUCCESS" Warning: The function completed, but an exception condition occurred which may require attention. Error: The function did not complete successfully. NOTE: Use of the an902_error_message ( ) function to get a textual description of errors and warnings.

5.8.7 Error Message an902_errorMessage

This function takes the Status Code returned by the instrument driver functions, interprets it and returns it to a user readable string.

Function Prototype
ViStatus **an902_errorMessage** (ViSession **instrumentHandle**, ViStatus **errorcode**, ViChar **errorMessage [ ]**) ;

an902_errorMessage		
Parameters	**Variable Type**	**Description**
<INPUT>		
instrumentHandle	ViSession	This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.
errorcode	ViStatus	This control accepts the Status Code returned from the instrument driver.
<OUTPUT>		
errorMessage [ ]	ViChar	This control returns the interpreted Status Code as a user readable message
<RETURN>		
= 0 > 0 < 0		"VI_SUCCESS" Warning: The function completed, but an exception condition occurred which may require attention. Error: The function did not complete successfully. NOTE: Use of the an902_error_message ( ) function to get a textual description of errors and warnings.

5.8.8 Reset an902_reset

This function resets the instrument to a known state and sets any necessary programmatic variables necessary for the operation of the instrument driver. It will disable all output ports and set all other registers to default values. Reads and writes to other module registers and will not transfer data during this time.

Function Prototype
ViStatus **an902_reset** (ViSession **instrumentHandle**);

an902_errorMessage		
Parameters	**Variable Type**	**Description**
<INPUT>		
instrumentHandle	ViSession	This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.
errorcode	ViStatus	This control accepts the Status Code returned from the instrument driver.
<OUTPUT>		
None		None
<RETURN>		
= 0 > 0 < 0		"VI_SUCCESS" Warning: The function completed, but an exception condition occurred which may require attention. Error: The function did not complete successfully. NOTE: Use of the an902_error_message ( ) function to get a textual description of errors and warnings.

6 DBS902 VME / VXI Bus Interface

The VME / VXI Interface conforms to a subset of ANSI / VITA 1 -1994 VME64. In addition, the DBS902 fully conforms to Version 1.4 of the VXI specification for register-based devices. All VXI specific registers are present.

Per VXI definitions, the DBS902 is a *Register based Servant* device. *Dynamic Configuration* is supported. Base / logical address is selectable over the range established by the VXI specification. *MODID slot detection ECL* and *TTL backplane triggers* are provided. The standard Interface *Register Offset Mapping* is supported. MODID and TTL Trigger pinouts are per VXI specification.

In terms of the VME specifications, the DBS902 is an A16: A24: A32 (jumper selectable): D32: D16: Slave that will support the following:

- 16-bit word transfers
- 32-bit word transfers
- Standard 16 Mbyte and Short 64K Addressing
- Block Transfers (BLT)
- Maskable Interrupt generation
- The DBS902 *does not* support Bus Request
- The DBS902 *is never* a Bus Master Device

The DBS902 can generate Interrupts, but CANNOT request data bus operations as a bus master. Interrupts may be generated on one of 7 levels. Interrupt sources are maskable and readable. The Interrupt Level is register programmable, and the interrupt is cleared after the interrupt acknowledge cycle, in accordance with the VME-defined " ROAK " protocol.

6.1 Interface Register Offset Map

The VXI-defined registers for a *Register-Based Servant Device* are located at the first four locations following the base address. The base / logical address is set by on board DIP switches. These registers are compliant with the VXI Specification Revision 1.4. The remaining Device Dependent registers are defined below.

Base +	Description	R/W	# Bits
0x00 to 0x06	VXI Common Registers	R/W	16
0x08 to 0x1E	Motherboard Registers	R/W	16
0x20	Input Control Register	R/W	16
0x22	Status Register	R only	16
0x24	Acquisition Trigger Control Register	R/W	8
0x26	Unused		
0x28	Unused		
0x2A	Unused		
0x2C	Level A Mode Register	R/W	16
0x2E	Level B Mode Register	R/W	16
0x30	INDEX Register	R/W	3
0x32	DATA Register MSB	R/W	5
0x34	DATA Register LSB	R/W	16
0x36	Interrupt Mask Register	R/W	4
0x38	EEPROM Register	R/W	3
0x3A	Unused	R/W	12
0x3C	OFFSET REGISTER	R/W	12
0x3E	Test Data Register	R/W	16

6.1.1 VXI Common Registers (0x0 to 0x06)

These registers are used for VXI interface functions and are described in the DBS9900 Specification.

6.1.2 Motherboard Registers (0x8 to 0x1E)

These registers are used to control trigger and timebase functions. They are described in the DBS9900 Specification.

6.1.3 Input Control Register (0x20) Read/Write

This register controls the INPUT TERMINATION RELAY, which is used to connect the 50-ohm termination resistors to the HI and LOW differential inputs. If 50 ohm-input termination is not explicitly selected, the DBS902 defaults to a 1 Meg Ohm input impedance. This register is cleared on power up or during a Reset cycle. It is also used to select one of two input channels. The BUF_SEL bit selects which of the two 1Meg word buffers the ADC will write data into. BUF_SEL = 0 selects BUFFER0; BUF_SEL = 1 selects BUFFER1. When the ADC is writing data into BUFFER0, the VME Bus Host can access the data in BUFFER1, and vice-versa.

BIT	NAME	DESCRIPTION
D0	TEST REG ENABLE	0 = Disable Test Register 1 = Enable Test Register
D1-D2	INPUT SOURCE	Input Source Select 00 = INTERNAL CAL INPUT 01 = EXTERNAL CAL INPUT 1X = IO INPUT
D3-D5	INPUT RANGE	100 = same as 101 101 = Gain 1 +/- 0.5 110 = Gain 2 +/- 1.0 111 = Gain 3 +/- 2.5 000 = Gain 4 +/- 5.0 001 = same as 000 010 = Gain 5 +/- 10.0 011 = Gain 6 +/- 25.0
D6	BUFFER SELECT	0 = Use Lower 1M Word Buffer to Acquire data (BUFFER0) 1 = Use Upper 1M word Buffer to Acquire data (BUFFER1)
D7	BUFFER MODE	0 = Two 1 M buffers, BUF_SEL Enable 1 = One 2 M buffer, BUF_SEL Disable
D8	FILTER ENABLE	0 = FILTER DISABLE 1 = FILTER ENABLE
D9	FILTER SELECT	0 = 20 MHZ LPF 1 = 40 MHZ LPF

BIT	NAME	DESCRIPTION
D10	ADC_SEL	0 = Hi speed ADC 1 = Lo Speed ADC
D11	FILL BUFFER	0 = Do not hold trigger 1 = Hold trigger until buffer is filled with pre-trigger data
D12-D13	TERM	0X = HIGH IMPEDENACE input 10 = 50 Ω termination 11 = 75 Ω termination
D14	AUX_ENA	0 = Acquire D1 and D0 into memory as zero 1 = Acquire D1 as Marker and D0 as Sync
D15	/2S_COMP	0 = ADC is coded as 2's Complement 1 = ADC is coded as Offset Binary (In this case the logic will invert the MSB to convert to 2's Complement data)

6.1.4 Status Register (0x22) Read Only

BIT	NAME	DESCRIPTION
D0	DONE	Set data acquisition is complete. Cleared if ARM is cleared
D1	TRIGGERED	Set if Trigger Condition is met during Acquisition. Cleared if ARM is cleared
D2	READY	0 = Memory is in acquisition mode and cannot be accessed by VXI host. 1 = Memory can be accessed by VXI host
D3	INPUT_OK	1 = Input voltages are in normal range 0 = Input voltages are over-range and relays are disconnected. Cleared automatically when input voltage exceeds +/- 15 Volts for >100mS. Set by write to Input Control Register.
D4	DC_OK	Cleared if DC power is out of tolerance Set on power up, Hard RESET or Soft RESET
D5	OVERRANGE	Set if a data value greater than 0x7FFC or less than 0x8000 is sampled. Cleared if ARM bit is cleared or RESET
D6	+12_OK	+12 volt supply is in tolerance
D7	+24_OK	+24 volt supply is in tolerance
D8	+3V1_OK	+3.3 volt supply # 1 is in tolerance
D9	+3V2_OK	+3.3 volt supply # 2 is in tolerance
D10	+5VA_OK	+5 volt analog supply is in tolerance
D11	-12_OK	-12 volt supply is in tolerance

BIT	NAME	DESCRIPTION
D12	-24_OK	-24 volt supply is in tolerance
D13	-5.2_OK	-5.2 volt supply is in tolerance
D14	-5VA_OK	-5 volt analog supply is in tolerance
D15	-2_OK	-2 volt supply is in tolerance

6.1.5 Acquisition Trigger Control Register (0x24)

This register controls operational modes of the DBS902. It is cleared to zero on power up and after a VME Bus Reset. The ARM bit enables writing of data into the memory buffer, as selected by the BUF_SEL bit. Once started, acquisition may be terminated by clearing the ARM bit to '0'. Clearing the ARM bit to 0 in this register has the effect of clearing the DONE, OVERRANGE and TRIGGERED bits (located in Register 0x22).

BIT	NAME	DESCRIPTION
D0	ARM	1 = DBS902 Starts sampling data 0 = Stop Sampling
D1	Data Trigger	1 = Enable Data Dependant trigger 0 = Disable Data Dependant trigger
D2	Re-Trigger Mode	1 = Enable Re-trigger mode 0 = Disable Re-trigger mode
D3	Re-sync	1 = Reset decimation counter with trigger 0 = Do not reset decimation counter
D4 – D5	+CAL_MUX	00 = +CAL open 01 = +CAL input to GND 10 = +CAL input to +2.5V Reference 11 = +CAL input to –2.5V Reference
D6- D7	-CAL_MUX	00 = -CAL open 01 = -CAL input to GND 10 = -CAL input to +2.5V Reference 11= -CAL input to –2.5V Reference

The Data Trigger Mode cannot be used together with the Re-trigger Mode. If both D1 and D2 are set, D1 takes precedence and Data Trigger Mode is selected. The following table indicates all trigger modes and other compatible modes. All modes require ARM to be set before a trigger can occur.

Data Trigger only uses the data stream itself to activate the trigger, and does not operate in a gated mode in the sense that it cannot start and stop and restart data acquisition. Once triggered, the 902 digitizes data until the capture register is satisfied. Data trigger uses two registers to set threshold levels for triggering (0x2C and 0x2E). Any logical combination of greater than and less than conditions as well as rising and falling edges can be used to trigger.

Re-trigger mode is only compatible with gate mode because it needs to be able to start/ stop/restart data acquisiton mutiple times.

The Re-sync bit is used to enable or disable clearing of the decimation counter with the trigger input. If disabled, pre-trigger samples will be decimated, and post trigger data will be sampled at the same phase. The sample that is coincident with the trigger will not likely be recorded in memory. If enabled, the decimation counter is cleared by the trigger. The data sample that was coincident with the trigger is recorded in memory and all the post trigger data is decimated with a new phase determined by the trigger. The pre-trigger data, however will have been acquired at a different phase of the decimation clock.

MODE	**DECIMATION**	**EDGE**	**GATE**	**RE-SYNC**	**SOURCE***	**D2**	**D1**
Data Trigger	yes	yes	no	N.A.	Data	X	1
Normal Trigger	yes	yes	yes	yes	S,B,F	0	0
Retrigger	yes	no	yes	yes	S,B,F	1	0

*** Source: S=Software trigger, B=Backplane trigger, F=Front panel trigger**

6.1.6 Level A Mode Register (0x2C) Read/Write

Register 0x26 controls the LEVEL A condition for Data Trigger mode.

A comparator monitors the incoming data stream and compares the upper 11 bits of this data to the Level A threshold register. The comparator output is true if the data is greater than the Level A register. Both the current sample data (N) and the previous sample data (N-1) are tested against Level A and the output of the comparator is latched. All logical combinations of the comparator output can be selected by D13 and D14 as the trigger condition.

BIT	NAME	DESCRIPTION
D0 to D10	LEVEL A	BITS 3-13 of Level A Data Trigger
D11 to D12	MARKER MUX	00 = Over Range 01 = External Input B 10 = Motherboard TRIGGER 11 = DATA TRIGGER
D13	GTH(N)	1 = Current Sample (N) > LEVEL A 0 = Current Sample (N) <or= LEVEL A
D14	GTH(N-1)	1 = Previous Sample (N -1) > LEVEL A 0 = Previous Sample (N -1) <or= LEVEL A

D11 and D12 are used to select one of 4 signals that can be recorded in memory at the MARKER bit position (D1). The Overrange input can be used to mark individual samples if the ADC exceeded the full scale range. External input B can be selected from the front panel DB26 connector on the DBS9900. Motherboard trigger can be used to record the trigger signal from the 9900 as a sample by sample check of data / trigger coincidence. Data trigger can also be recorded as a check of trigger conditions and data.

6.1.7 Level B Mode Register (0x2E) Read/Write

Register 0x26 controls the mode of data-dependent triggering for LEVEL B.

As with the Level A register, A comparator monitors the incoming data stream and compares the upper 11 bits of this data to the Level B threshold register. All logical combinations of the comparator output can be selected by D13 and D14 as the trigger condition. The Level B condition can also be selected as a “don’t care” by setting D15.

BIT	NAME	DESCRIPTION
D0 to D10	LEVEL B	BITS 3-13 of Level A Data Trigger
D11 to D12	NOT USED	
D13	GTH(N)	1 = Current Sample (M) > LEVEL B 0 = Current Sample (M) <or= LEVEL B
D14	GTH(N-1)	1 = Previous Sample (M -1) > LEVEL B 0 = Previous Sample (M -1) <or= LEVEL B
D15	DISABLE	Disable Level B DATA trigger

6.1.8 Data Registers (MSB 0x32)(LSB 0x34)

These two register addresses are used as a window into a block of eight registers that are selected by the INDEX Register value. Some of these registers exceed 16 bits in length so an MSB and an LSB register are provided to accommodate these.

INDEX	R/W	Description
000	Read only	Trigger Sample Pointer Register
001	Read only	Last Sample Pointer Register
010	R/W	Capture Size Register
011	Write only	Retrigger Size Register
100	Read only	Address Generator Direct Access
101	R/W	Decimation Register
110		Reserved
111		Reserved

6.1.8.1 Trigger Sample Pointer Register (INDEX = 0x0)

This Read Only 21-bit register is a pointer into data memory indicating the first data sample that occurred just after the trigger was received. The 16 LSB's can be accessed in the Data Register at 0x34 when the Index Register is 0. The 5 MSB's are accessed at 0x32.

Since the data buffer is circular, and data is continually being recorded before the trigger condition, the first location in (physical) memory may not necessarily be the trigger sample. This register indicates the location of the *trigger data sample*. The register is written to automatically write by hardware, and read by software. It represents a sample offset from the base of the buffer memory. To convert to a VME address (byte), multiply by 2 and add to the base address set in the OFFSET REGISTER (0x06). The following tables indicate the assignment of the VME bus address bits in these registers. A0 does not appear in this register because it will always be zero during D16 word access.

MSB's – 0x32

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
											A21	A20	A19	A18	A17

LSB's – 0x34

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
A16	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1

6.1.8.2 Last Sample Pointer Register (INDEX = 0x1)

This Read only 21-bit address pointer indicates the address in data memory of the last recorded ADC Data sample. This is intended to mark the end of the data buffer. Since the data buffer is circular, and data is continually being recorded before the trigger condition, the last location (physical address) in memory will not be the newest data sample. This register indicates the location of the *last data sample*. This register is written to automatically write by hardware, and read by software following an acquisition. It represents a word offset from the base of the buffer memory. To convert to a VME address (byte), multiply by 2 and add to the base address set in the OFFSET REGISTER (0x06). The following tables indicate the assignment of the VME bus address bits in these registers. A0 does not appear in this register because it will always be zero during D16 word access.

MSB's – 0x32

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
											A21	A20	A19	A18	A17

LSB's – 0x34

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
A16	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1

6.1.8.3 Capture Size Register (INDEX = 0x2)

This Read/Write 21-bit register controls the number of samples to be recorded after a trigger occurs (the Trigger Offset Count). When using a single 2 Mega sample buffer, a value of 0x0000 results in Pre-trigger Data (all data in the buffer occurs before the trigger). A value of 0x1FFFFF results in Post-trigger data (all data in the buffer occurs after the trigger). A value of 0x100000 results in Center trigger (trigger point is in the middle of the data set). The value in this register must be written by the host software, prior to beginning of data acquisition. Following a trigger event, this register controls how many more data points will be stored in memory.

MSB's – 0x32

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
											Msb				

LSB's – 0x34

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
															Lsb

6.1.8.4 Retrigger Size Register (INDEX = 0x3)

The Write only Retrigger Register determines the number of samples that are acquired in a burst in Retrigger mode. This is a 12-bit value allowing up to 4096 samples in a burst. If the Capture Size register does not allow for all the samples set in the Retrigger Register, then acquisition will end with the last burst Retrigger Size unsatisfied. The samples are acquired at the clock rate.

Retrigger Size – 0x34

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
				Msb											Lsb

6.1.8.5 Address Generator Register (INDEX = 0x4)

The Address Generator Register (Read only) is used for diagnostic purposes. It is a 21 bit value that can be read back directly from the memory controller through the memory address bus path, thus allowing a direct bit by bit test of the address generator in the memory controller and the address bus path to memory. The address is automatically incremented in the memory controller once after each read cycle of the LSB counter value. Repeated reads can therfore cycle through all possible address codes.

Note that the complement of A20 and A21 are also readback. These are used as chip selects into the memory array.

Another way of using this register is after data is acquired into memory and the DONE status bit is set. Before the ARM bit is cleared, the LAST SAMPLE ADDRESS may be read from this register and should be one greater than the LAST SAMPLE POINTER register. Repeated reads will increment the address, so care must be taken to not read this register during data acquisition.

The upper 8 bits of register 0x32 should be masked by software because they are not driven to zero by hardware.

MSB's – 0x32

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
									/A21	A21	/A20	A20	A19	A18	A17

LSB's – 0x34

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
A16	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1

6.1.8.6 Decimation Register (INDEX = 0x5)

The Decimation Register (Read/Write) determines the number of samples to discard before recording in memory in Decimation mode. This is a 10-bit value allowing up to 1023 samples to be discarded.

For example, to reduce the sample rate to one half the sample clock frequency, write a value of 0x0001 here to discard every alternate sample.

Decimation – 0x34

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
						Msb									Lsb

6.1.9 Interrupt Mask (0x36) Read/Write

This register consists of two separate bit fields. Bits D3-D0 enable interrupt generation under the following conditions: Acquisition Done (a buffer is full); Over-range Data is detect; Trigger Detect; or Software interrupt. Bits D10-D8 indicate the status of the interrupt. The interrupt sources are edge triggered and remain latched on the 902 until the corresponding mask bit is cleared to 0.

A typical interrupt cycle would be to set the desired mask bit to enable the interrupt, for example D0 = 1 for the DONE interrupt. Set the interrupt Level and Vector on the 9900 Motherboard, perform a data acquisition, receive the interrupt, read the status / ID (vector), and read this register the determine the source (D10,D9,orD8). Then D0 should be cleared and set again to arm for the next DONE interrupt.

BIT	Name	DESCRIPTION
D0	EN_DONE	If set, interrupts when DONE is set. (R/W)
D1	EN_OVERRANGE	If set, interrupts when +OVERRANGE or -OVERRANGE is set. (R/W)
D2	EN_TRIGGERED	If set, interrupts when TRIGGERED is set. (R/W)
D3	EN_SOFT_INT	If set, enables software interrupt. (R/W)
D4-D7	UNUSED	UNUSED
D8	DONE INT	Set by DONE interrupt (R)
D9	OVERRANGE INT	Set by Overrange status from ADC (R)
D10	TRIGGER INT	Set by Triggered status from memory controller (R)
D11-D15	UNUSED	UNUSED

6.1.10 Serial EEPROM Register (0x38) Read/Write

The Serial EEPROM register (0x38) R/W contains information about the module such as serial number, model number, PCB revision, firmware revisions, calibration data, etc. This data is not volatile, but has a limited endurance of write or erase cycles. Software is required to read and write data to the EEPROM. This register allows one bit at a time to be written to (or read from) the device. See the 93C56 data sheet for details of the serial protocol. The serial data clock is generated automatically by hardware when each bit is transferred.

BIT	Name	DESCRIPTION
D0	Data out	Serial Data to the 93C56
D1	Chip select	This is high to select the device, low to prevent accidental erasure.
D2	Data in	Serial Data from the 93C56

6.1.11 OFFSET Register (0x3C) Read/Write

This register controls a DAC to add a DC offset to the DBS902 output. This data is 12 bit offset binary justified right. This register must be initialized by software to 0x0800. Later the calibration software will write a value to this DAC to null offset present in the input stage.

	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
				MSB											LSB
NOT USED				OFFSET DAC DATA											

6.1.12 Test Data Register (0x3E) Read/Write

This register is used for diagnostic functions and is split into two registers, one READ only and one WRITE only.

The WRITE function is used to write 16 bits of data to the ADC data bus in place of the ADC data to test this data bus and register pipeline. This data may be then acquired into memory and compared against the register . The upper 14 bits (D15 thru D2) are the 14 bits of simulated data, and are recorded in memory at the same bit locations.

Bit D1 is used as the /TRIGGER input, replacing the Motherboard trigger as long as the TEST REGISTER ENABLE bit is set. This bit is also recorded into memory if the AUX_EN bit is set, and the MARKER is set to Motherboard Trigger in the Level A MODE register (0x2C).

Bit D0 is recorded into memory in place of the OVERRANGE status indicator from the ADC converter if the AUX_EN bit is set.

For The test data register to work correctly, the TEST REGISTER ENABLE bit and the AUX_EN bit in the Input Control Register must be set, and the MARKER must be set to Motherboard Trigger in the Level A MODE register.

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
MSB													LSB	/TRG	OVR
DIAGNOSIC TEST DATA															

The READ function of the TEST DATA register is used to read the FAST or SLOW ADC converters directly, bypassing memory. D15 thru D2 are the ADC data.

D1 is used as the Marker, which may be switched from one of four sources including the SYNCB digital input. D0 is the SYNCA digital input.

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
MSB													LSB		
- - - - - - - - - - - - - - **A/D DATA BITS** - >>>														MK	SYN

6.2 Data Format

DBS902 data is in fractional 2's complement format, with D15 being the "sign bit", and D14-D2 representing the "magnitude". D1 is used as the Marker, which may be switched from one of four sources including the SYNCB digital input. D0 is the SYNCA digital input. These inputs are pipelined to match the analog data from the ADC. They may also be switched to ground so that the memory data LSB's need not be masked before being used in calculations.

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
MSB													LSB		
- - - - - - - - - - - - - -- A /D DATA BITS - >>>														MK	SYN

The DBS902 has 2M x 16 of SRAM, arranged as two separate banks, BUFFER0 and BUFFER1. Each bank is 1M x 16. BUFFER0 is the lower addressed block. The memory appears as one contiguous address space, starting at the DBS902 Base Address set in the OFFSET REGISTER. All memory access cycles are defined by Address Modifier Codes 3D or 39 for word transfers, or AM codes 3F or 3B for Block transfers.