

ARTISAN®
TECHNOLOGY GROUP

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

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

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

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

Custom engineering
so your equipment works exactly as you specify.

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

Expert team | Trust guarantee | 100% satisfaction

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

Visit our website - *Click HERE*

Keysight M93xxA PXIe Modules

This document supports the following products:

M9381A PXIe Vector Signal Generator
M9389A PXIe VNA Source
M9309A PXIe VNA Synthesizer
M9380A PXIe CW Source
M9391A PXIe Vector Signal Analyzer

Security Features and Certificate of Volatility

KEYSIGHT TECHNOLOGIES

Notices

Copyright Notice

© Keysight Technologies 2014-2022

No part of this manual may be reproduced in any form or by any means (including electronic storage and retrieval or translation into a foreign language) without prior agreement and written consent from Keysight Technologies, Inc. as governed by United States and international copyright laws.

Manual Part Number

M9300-90021

Edition

Edition 1, January 2022

Supersedes: August 2021

Published by:

Keysight Technologies Inc.
1400 Fountaingrove Parkway
Santa Rosa, CA 95403

Regulatory Compliance

This product has been designed and tested in accordance with accepted industry standards, and has been supplied in a safe condition. To review the Declaration of Conformity, go to http://www.keysight.com/go/conformity.

Technology Licenses

The hardware and/or software described in this document are furnished under a license and may be used or copied only in accordance with the terms of such license.

U.S Government Rights

The Software is “commercial computer software,” as defined by Federal Acquisition Regulation (“FAR”) 2.101. Pursuant to FAR 12.212 and 27.405-3 and Department of Defense FAR Supplement (“DFARS”) 227.7202, the U.S. government acquires commercial computer software under the same terms by which the software is customarily provided to the public. Accordingly, Keysight provides the Software to U.S. government customers under its standard commercial license, which is embodied in its End User License Agreement (EULA), a copy of which can be found at http://www.keysight.com/find/sweula The license set forth in the EULA represents the exclusive authority by which the U.S. government may use, modify, distribute, or disclose the Software. The EULA and the license set forth therein, does not require or permit, among other things, that Keysight: (1) Furnish technical information related to commercial computer software or commercial computer software documentation that is not customarily provided to the public; or (2) Relinquish to, or otherwise provide, the government rights in excess of these rights customarily provided to the public to use, modify, reproduce, release, perform, display, or disclose commercial computer software or commercial computer software documentation. No additional government requirements beyond those set forth in the EULA shall apply, except to the extent that those terms, rights, or licenses are explicitly required from all providers of commercial computer software pursuant to the FAR and the DFARS and are set forth specifically in writing elsewhere in the EULA. Keysight shall be under no obligation to update, revise or otherwise modify the Software. With respect to any technical data as defined by FAR 2.101, pursuant to FAR 12.211 and 27.404.2 and DFARS 227.7102, the U.S. government acquires no greater than Limited Rights as defined in FAR 27.401 or DFAR 227.7103-5 (c), as applicable in any technical data.

Warranty

THE MATERIAL CONTAINED IN THIS DOCUMENT IS PROVIDED “AS IS,” AND IS SUBJECT TO BEING CHANGED, WITHOUT NOTICE, IN FUTURE EDITIONS. FURTHER, TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, KEYSIGHT DISCLAIMS ALL WARRANTIES, EITHER EXPRESS OR IMPLIED, WITH REGARD TO THIS MANUAL AND ANY INFORMATION CONTAINED HEREIN, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. KEYSIGHT SHALL NOT BE LIABLE FOR ERRORS OR FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH THE FURNISHING, USE, OR PERFORMANCE OF THIS DOCUMENT OR OF ANY INFORMATION CONTAINED HEREIN. SHOULD KEYSIGHT AND THE USER HAVE A SEPARATE WRITTEN AGREEMENT WITH WARRANTY TERMS COVERING THE MATERIAL IN THIS DOCUMENT THAT CONFLICT WITH THESE TERMS, THE WARRANTY TERMS IN THE SEPARATE AGREEMENT SHALL CONTROL.

Safety Information

CAUTION

A **CAUTION** notice denotes a hazard. It calls attention to an operating procedure, practice, or the like that, if not correctly performed or adhered to, could result in damage to the product or loss of important data. Do not proceed beyond a CAUTION notice until the indicated conditions are fully understood and met.

WARNING

A **WARNING** notice denotes a hazard. It calls attention to an operating procedure, practice, or the like that, if not correctly performed or adhered to, could result in personal injury or death. Do not proceed beyond a WARNING notice until the indicated conditions are fully understood and met.

Table of Contents

Safety Information

Product Markings 8

Contacting Keysight 9

Where to Find the Latest Information 9

Is your product software up-to-date? 9

Contacting Keysight Sales and Service Offices 9

Memory Declassification Procedure

Security Terms and Definitions 13

Instrument Drivers 14

M9381A PXIe VSG, M9380A PXIe CW Source, and M9389A PXIe VNA Instrument Drivers 14

M9391A PXIe VSA Drivers 15

M9300A PXIe Frequency Reference 16

M9301A PXIe Synthesizer 17

M9309A PXIe Vector Network Synthesizer 19

M9310A PXIe Source 21

M9311A PXIe Digital Vector Modulator 23

M9350A PXIe RF Downconverter 24

M9214A PXIe IF Digitizer 26

VISA Address Tip for Provided Programming Examples 28

Memory Clear Code Programming Examples 29

M9381A Memory Clear Code 29

M9380A Memory Clear Code 33

M9389A Memory Clear Code 37

M9391A Memory Clear Code 40

M9300A Memory Clear Code 44

M9309A Memory Clear Code 47

Contents

Safety Information

The following safety precautions should be observed before using this product and any associated instrumentation. This product is intended for use by qualified personnel who recognize shock hazards and are familiar with the safety precautions required to avoid possible injury. Read and follow all installation, operation, and maintenance information carefully before using the product.

WARNING

If this product is not used as specified, the protection provided by the equipment could be impaired. This product must be used in a normal condition (in which all means for protection are intact) only.

The types of product users are:

- Responsible body is the individual or group responsible for the use and maintenance of equipment, for ensuring that the equipment is operated within its specifications and operating limits, and for ensuring operators are adequately trained.
- Operators use the product for its intended function. They must be trained in electrical safety procedures and proper use of the instrument.They must be protected from electric shock and contact with hazardous live circuits.
- Maintenance personnel perform routine procedures on the product to keep it operating properly (for example, setting the line voltage or replacing consumable materials). Maintenance procedures are described in the user documentation.The procedures explicitly state if the operator may perform them. Otherwise, they should be performed only by service personnel.
- Service personnel are trained to work on live circuits, perform safe installations, and repair products.Only properly trained service personnel may perform installation and service procedures.

WARNING

Operator is responsible to maintain safe operating conditions. To ensure safe operating conditions, modules should not be operated beyond the full temperature range specified in the Environmental and physical specification. Exceeding safe operating conditions can result in shorter lifespans, improper module performance and user safety issues. When the modules are in use and operation within the specified full temperature range is not maintained, module surface temperatures may exceed safe handling conditions which can cause discomfort or burns if touched. In the event of a module exceeding the full temperature range, always allow the module to cool before touching or removing modules from chassis.

Keysight products are designed for use with electrical signals that are rated Measurement Category I and Measurement Category II, as described in the International Electro-technical Commission (IEC) Standard IEC 60664. Most measurement, control, and data I/O signals are Measurement Category I and must not be directly connected to mains voltage or to voltage sources with high transient over-voltages. Measurement Category II connections require protection for high transient over-voltages often associated with local AC mains connections.Assume all measurement, control, and data I/O connections are for connection to Category I sources unless otherwise marked or described in the user documentation.

Exercise extreme caution when a shock hazard is present. Lethal voltage may be present on cable connector jacks or test fixtures. The American National Standards Institute (ANSI) states that a shock hazard exists when voltage levels greater than 30V RMS, 42.4V peak, or 60VDC are present. A good safety practice is to expect that hazardous voltage is present in any unknown circuit before measuring.

Operators of this product must be protected from electric shock at all times. The responsible body must ensure that operators are prevented access and/or insulated from every connection point. In some cases, connections must be exposed to potential human contact. Product operators in these circumstances must be trained to protect themselves from the risk of electric shock. If the circuit is capable of operating at or above 1000V, no conductive part of the circuit may be exposed.

Do not connect switching cards directly to unlimited power circuits. They are intended to be used with impedance limited sources. NEVER connect switching cards directly to AC mains. When connecting sources to switching cards, install protective devices to limit fault current and voltage to the card.

Before operating an instrument, ensure that the line cord is connected to a properly grounded power receptacle. Inspect the connecting cables, test leads, and jumpers for possible wear, cracks, or breaks before each use.

When installing equipment where access to the main power cord is restricted, such as rack mounting, a separate main input power disconnect device must be provided in close proximity to the equipment and within easy reach of the operator.

For maximum safety, do not touch the product, test cables, or any other instruments while power is applied to the circuit under test. ALWAYS remove power from the entire test system and discharge any capacitors before: connecting or disconnecting cables or jumpers, installing or removing switching cards, or making internal changes, such as installing or removing jumpers.

Do not touch any object that could provide a current path to the common side of the circuit under test or power line (earth) ground. Always make measurements with dry hands while standing on a dry, insulated surface capable of withstanding the voltage being measured.

The instrument and accessories must be used in accordance with its specifications and operating instructions, or the safety of the equipment may be impaired.

WARNING

Do not exceed the maximum signal levels of the instruments and accessories, as defined in the specifications and operating information, and as shown on the instrument or test fixture panels, or switching card.

When fuses are used in a product, replace with the same type and rating for continued protection against fire hazard.

Chassis connections must only be used as shield connections for measuring circuits, NOT as safety earth ground connections.

If you are using a test fixture, keep the lid closed while power is applied to the device under test. Safe operation requires the use of a lid interlock.

Instrumentation and accessories shall not be connected to humans.

Before performing any maintenance, disconnect the line cord and all test cables.

To maintain protection from electric shock and fire, replacement components in mains circuits - including the power transformer, test leads, and input jacks - must be purchased from Keysight. Standard fuses with applicable national safety approvals may be used if the rating and type are the same. Other components that are not safety related may be purchased from other suppliers as long as they are equivalent to the original component (note that selected parts should be purchased only through Keysight to maintain accuracy and functionality of the product). If you are unsure about the applicability of a replacement component, call an Keysight office for information.

WARNING

No operator serviceable parts inside. Refer servicing to qualified personnel. To prevent electrical shock do not remove covers. For continued protection against fire hazard, replace fuse with same type and rating.

Product Markings

Symbol	Definition
	The CE mark is a registered trademark of the European Community.
	Australian Communication and Media Authority mark to indicate regulatory compliance as a registered supplier.
	This symbol indicates product compliance with the Canadian Interference-Causing Equipment Standard (ICES-001). It also identifies the product is an Industrial Scientific and Medical Group 1 Class A product (CISPR 11, Clause 4).
	South Korean Class A EMC Declaration. This equipment is Class A suitable for professional use and is for use in electromagnetic environments outside of the home. A 급 기기 (업무용 방송통신기자재) 이 기기는 업무용 (A 급) 전자파적합기 기로서 판 매자 또는 사용자는 이 점을 주 의하시기 바라 며 , 가정외의 지역에서 사용하는 것을 목적으 로 합니다.
	This product complies with the WEEE Directive marketing requirement. The affixed product label (above) indicates that you must not discard this electrical/electronic product in domestic household waste. Product Category: With reference to the equipment types in the WEEE directive Annex 1, this product is classified as “Monitoring and Control instrumentation” product. Do not dispose in domestic household waste. To return unwanted products, contact your local Keysight office, or for more information see http://about.keysight.com/en/companyinfo/environment/takeback.shtml
	This symbol indicates the instrument is sensitive to electrostatic discharge (ESD). ESD can damage the highly sensitive components in your instrument. ESD damage is most likely to occur as the module is being installed or when cables are connected or disconnected. Protect the circuits from ESD damage by wearing a grounding strap that provides a high resistance path to ground. Alternatively, ground yourself to discharge any builtup static charge by touching the outer shell of any grounded instrument chassis before touching the port connectors.
	This symbol on an instrument means caution, risk of danger. You should refer to the operating instructions located in the user documentation in all cases where the symbol is marked on the instrument.
	This symbol indicates the time period during which no hazardous or toxic substance elements are expected to leak or deteriorate during normal use. Forty years is the expected useful life of the product.

Contacting Keysight

Where to Find the Latest Information

Documentation is updated periodically. For the latest information about these products, including instrument software upgrades, application information, and product information, see the following URLs:

http://www.keysight.com/find/m9300a

To receive the latest updates by email, subscribe to Keysight Email Updates:

http://www.keysight.com/find/emailupdates

Information on preventing instrument damage can be found at:

http://www.keysight.com/find/PreventingInstrumentRepair

Is your product software up-to-date?

Periodically, Keysight releases software updates to fix known defects and incorporate product enhancements. To search for software updates for your product, go to the Keysight Technical Support website at:

http://www.keysight.com/find/techsupport

Contacting Keysight Sales and Service Offices

Assistance with test and measurement needs, and information to help you find a local Keysight office, is available via the internet at, http://www.keysight.com/find/assist. If you do not have internet access, please contact your designated Keysight representative.

In any correspondence or telephone conversation, refer to the instrument by its model number and full serial number. With this information, the Keysight representative can determine whether your unit is still within its warranty period.

Memory Declassification Procedure

Some test equipment users have a need to “declassify” or “sanitize” their instruments for security purposes. This involves following a procedure to clear all user data from the instrument’s memory. The result is a sanitized instrument that can be removed from a secure area without any chance of classified data being recovered from it. This document details the internal memory locations of the M9300A Frequency Reference. It describes instrument security features and the steps necessary to declassify the products through memory sanitization or removal. For additional information on a particular product, the Keysight Instrument Security Database may be accessed here: www.keysight.com/find/security. For general information, the Keysight Aerospace and Defense web page may be found at www.keysight.com/find/ad.

What you will find in this section:

– “M9381A PXIe VSG, M9380A PXIe CW Source, and M9389A PXIe VNA Instrument Drivers” on page 14
– “M9391A PXIe VSA Drivers” on page 15
– “M9300A PXIe Frequency Reference” on page 16
– “M9301A PXIe Synthesizer” on page 17
– “M9309A PXIe Vector Network Synthesizer” on page 19
– “M9310A PXIe Source” on page 21
– “M9311A PXIe Digital Vector Modulator” on page 23
– “M9350A PXIe RF Downconverter” on page 24
– “M9214A PXIe IF Digitizer” on page 26
– “VISA Address Tip for Provided Programming Examples” on page 28
– “M9381A Memory Clear Code” on page 29
– “M9380A Memory Clear Code” on page 33
– “M9389A Memory Clear Code” on page 37
– “M9391A Memory Clear Code” on page 40
– “M9300A Memory Clear Code” on page 44

– “M9309A Memory Clear Code” on page 47

Security Terms and Definitions

Term	Definition
Clearing	As defined in Section 8-301a of DoD 5220.22-M, clearing is the process of eradicating the data on media before reusing the media so that the data can no longer be retrieved using the standard interfaces on the instrument. Clearing is typically used when the instrument is to remain in an environment with an acceptable level of protection.
Instrument Declassification	A term that refers to procedures that must be undertaken before an instrument can be removed from a secure environment, such as is the case when the instrument is returned for calibration. Declassification procedures include memory sanitization or memory removal, or both. Keysight declassification procedures are designed to meet the requirements specified in DoD 5220.22-M, Chapter 8.
Sanitization	As defined in Section 8-301b of DoD 5220.22-M, sanitization is the process of removing or eradicating stored data so that the data cannot be recovered using any known technology. Instrument sanitization is typically required when an instrument is moved from a secure to a non-secure environment, such as when it is returned to the factory for calibration. Keysight memory sanitization procedures are designed for customers who need to meet the requirements specified by the US Defense Security Service (DSS). These requirements are specified in the “Clearing and Sanitization Matrix” in Section 5.2.5.5.5 of the ISFO Process Manual.
Secure Erase	Secure Erase is a term that is used to refer to either the clearing or sanitization features of Keysight instruments.

Instrument Drivers

M9381A PXIe VSG, M9380A PXIe CW Source, and M9389A PXIe VNA Instrument Drivers

These products use the AgM938x driver. The driver installs the IVI-C, IVI-COM, LabView, and MATLAB driver components, as well as the soft front panel and kernel device driver on your controller.

Memory Type 1	
Memory Type: Controller Hard Drive	**Memory Size:** unknown
Memory Function: Stores device drivers, example programs, example waveforms, help system, and user documentation.	
User Modifiable? Yes	**Volatile?** No
Memory Erase Processes: To uninstall the AgM938x instrument driver from the controller, perform the relevant procedure below. Windows 7: 1. Select Start > Control Panel > Programs and Features 2. Select Keysight M938x 3. Select Uninstall Windows XP: 1. Select Start > Control Panel 2. Double-click Add or Remove Programs 3. Select Keysight M938x 4. Select Remove To clear all information from the controller used with the M9381A PXIe Vector Signal Generator, M9380A PXIe CW Source, or the M9389A PXIe VNA follow the memory erase procedure for the controller as recommended by the manufacturer.	

M9391A PXIe VSA Drivers

This product uses the same AgM9391 driver. The driver installs the IVI-C, IVI-COM, LabView, and MATLAB driver components, as well as the soft front panel and kernel device driver on your controller.

Memory Type 1	
Memory Type: Controller Hard Drive	**Memory Size:** unknown
Memory Function: Stores device drivers, example programs, example waveforms, help system, and user documentation.	
User Modifiable? Yes	**Volatile?** No
Memory Erase Processes: To uninstall the AgM9391 instrument driver from the controller, perform the relevant procedure below. Windows 7: 1. Select Start > Control Panel > Programs and Features 2. Select Keysight M9391 3. Select Uninstall Windows XP: 1. Select Start > Control Panel 2. Double-click Add or Remove Programs 3. Select Keysight M9391 4. Select Remove To clear all information from the controller used with the M9391A PXIe Vector Signal Analyzer, follow the memory erase procedure for the controller as recommended by the manufacturer.	

M9300A PXIe Frequency Reference

Memory Type 1	
Memory Type: Flash Memory	**Memory Size:** 128 M Bit
Memory Function: Stores module model number, serial number, manufacturing number, PCB part and version numbers, cal verify date, max module temperature, and calibration data.	
User Modifiable? No	**Volatile?** No
Memory Erase Processes: None, this is not user accessible.	

Memory Type 2	
Memory Type: Flash Memory	**Memory Size:** 128 M Bit
Memory Function: Stores Device firmware. Images can be changed using the Keysight Soft Front Panel firmware update utility.	
User Modifiable? No	**Volatile?** No
Memory Erase Processes: None, this is not user accessible.	

Memory Type 3	
Memory Type: Flash Memory	**Memory Size:** 128 M Bit
Memory Function: Stores calibration preferences: due date subject to periodic cal, module cal warnings, cal due reminder, and module cal reminder and passphrase.	
User Modifiable? Yes	**Volatile?** No
Memory Erase Processes: You can clear the passphrase by using the relevant IVI driver code. See "M9300A Memory Clear Code" on page 44.	

Memory Type 4	
Memory Type: Flash Memory	**Memory Size:** 128 M Bit
Memory Function: Passphrase, user customizable asset number and system identification.	
User Modifiable? Yes	**Volatile?** No
Memory Erase Processes: You can clear the asset number and system identification values using the relevant IVI driver code. See "M9300A Memory Clear Code" on page 44.	

Memory Type 5	
Memory Type: FPGA	**Memory Size:** N/A
Memory Function: Reference output selections, external reference and frequency selections, time shift and self test results.	
User Modifiable? Yes	**Volatile?** Yes
Memory Erase Processes: Cycle power	

M9301A PXIe Synthesizer

Memory Type 1	
Memory Type: Flash Memory	**Memory Size:** 128 M Bit
Memory Function: Stores module model number, serial number, manufacturing number, options, PCB part and version numbers, cal verify date, max module temperature, and calibration data.	
User Modifiable? No	**Volatile?** No
Memory Erase Processes: None, this is not user accessible.	

Memory Type 2	
Memory Type: Flash Memory	**Memory Size:** 128 M Bit
Memory Function: Stores Device firmware. Images can be changed using the Keysight Soft Front Panel firmware update utility.	
User Modifiable? No	**Volatile?** No
Memory Erase Processes: None, this is not user accessible.	

Memory Type 3	
Memory Type: Flash Memory	**Memory Size:** 128 M Bit
Memory Function: Stores calibration preferences: due date subject to periodic cal, module cal warnings, cal due reminder, and module cal reminder. and passphrase.	
User Modifiable? Yes	**Volatile?** No
Memory Erase Processes: You can clear the passphrase by using the relevant IVI driver code: – If the module is used in an M9381A instrument, see "M9381A Memory Clear Code" on page 29. – If the module is used in an M9380A instrument, see "M9380A Memory Clear Code" on page 33. – If the module is used in an M9391A instrument, see "M9391A Memory Clear Code" on page 40.	

Memory Type 4	
Memory Type: FPGA	**Memory Size:** N/A
Memory Function: Frequency start/stop/step, power, waveform and impairments.	
User Modifiable? Yes	**Volatile?** Yes
Memory Erase Processes: Cycle power	

Memory Type 5	
Memory Type: Flash Memory	**Memory Size:** 128 M Bit
Memory Function: Stores user customizable asset number and system identification.	
User Modifiable? Yes	**Volatile?** No
Memory Erase Processes: You can clear the asset number and system identification values using the relevant IVI driver code. – If the module is used in an M9381A instrument, see "M9381A Memory Clear Code" on page 29. – If the module is used in an M9380A instrument, see "M9380A Memory Clear Code" on page 33. – If the module is used in an M9391A instrument, see "M9391A Memory Clear Code" on page 40.	

M9309A PXIe Vector Network Synthesizer

Memory Type 1	
Memory Type: Flash Memory	**Memory Size:** 128 M Bit
Memory Function: Stores module model number, serial number, manufacturing number, options, PCB part and version numbers, cal verify date, max module temperature, alignment, and calibration data.	
User Modifiable? No	**Volatile?** No
Memory Erase Processes: None, this is not user accessible.	

Memory Type 2	
Memory Type: Flash Memory	**Memory Size:** 128 M Bit
Memory Function: Stores Device firmware. Images can be changed using the Keysight Soft Front Panel firmware update utility.	
User Modifiable? No	**Volatile?** No
Memory Erase Processes: None, this is not user accessible.	

Memory Type 3	
Memory Type: Flash Memory	**Memory Size:** 128 M Bit
Memory Function: Stores calibration preferences: due date subject to periodic cal, module cal warnings, cal due reminder, and module cal reminder. and passphrase.	
User Modifiable? Yes	**Volatile?** No
Memory Erase Processes: You can clear the passphrase by using the relevant IVI driver code: – If the module is used in an M9389A instrument, see "M9389A Memory Clear Code" on page 37. – If the module is used in a standalone M9309A instrument, see "M9309A Memory Clear Code" on page 47.	

Memory Type 4	
Memory Type: FPGA	**Memory Size:** N/A
Memory Function: Frequency start/stop/step, power, waveform and impairments.	
User Modifiable? Yes	**Volatile?** Yes
Memory Erase Processes: Cycle power	

Memory Type 5	
Memory Type: Flash Memory	**Memory Size:** 128 M Bit
Memory Function: Stores user customizable asset number and system identification.	
User Modifiable? Yes	**Volatile?** No
Memory Erase Processes: You can clear the asset number and system identification values using the relevant IVI driver code in. – If the module is used in an M9389A instrument, see "M9389A Memory Clear Code" on page 37. – If the module is used in a standalone instrument, see "M9309A Memory Clear Code" on page 47.	

M9310A PXIe Source

Memory Type 1	
Memory Type: Flash Memory	**Memory Size:** 128 M Bit
Memory Function: Stores module model number, serial number, manufacturing number, options, PCB part and version numbers, cal verify date, max module temperature, and calibration data.	
User Modifiable? No	**Volatile?** No
Memory Erase Processes: None, this is not user accessible.	

Memory Type 2	
Memory Type: Flash Memory	**Memory Size:** 128 M Bit
Memory Function: Stores Device firmware. Images can be changed using the Keysight Soft Front Panel firmware update utility.	
User Modifiable? No	**Volatile?** No
Memory Erase Processes: None, this is not user accessible.	

Memory Type 3	
Memory Type: Flash Memory	**Memory Size:** 128 M Bit
Memory Function: Stores calibration preferences: due date subject to periodic cal, module cal warnings, cal due reminder, and module cal reminder. and passphrase.	
User Modifiable? Yes	**Volatile?** No
Memory Erase Processes: You can clear the passphrase by using the relevant IVI driver code in: – If the module is used in an M9381A instrument, see "M9381A Memory Clear Code" on page 29. – If the module is used in an M9380A instrument, see "M9380A Memory Clear Code" on page 33. – If the module is used in an M9389A instrument, see "M9389A Memory Clear Code" on page 37.	

Memory Type 4	
Memory Type: FPGA	**Memory Size:** N/A
Memory Function: Frequency start/stop/step, power, waveform and impairments.	
User Modifiable? Yes	**Volatile?** Yes
Memory Erase Processes: Cycle power	

Memory Type 5	
Memory Type: Flash Memory	**Memory Size:** 128 M Bit
Memory Function: Stores user customizable asset number and system identification.	
User Modifiable? Yes	**Volatile?** No
Memory Erase Processes: You can clear the asset number and system identification values using the relevant IVI driver code in: – If the module is used in an M9381A instrument, see "M9381A Memory Clear Code" on page 29. – If the module is used in an M9380A instrument, see "M9380A Memory Clear Code" on page 33. – If the module is used in an M9389A instrument, see "M9389A Memory Clear Code" on page 37.	

M9311A PXIe Digital Vector Modulator

The M9311A is only used with the M9381A PXIe VSG.

Memory Type 1	
Memory Type: Flash Memory	**Memory Size:** 128 M Bit
Memory Function: Stores module model number, serial number, manufacturing number, options, PCB part and version numbers, cal verify date, max module temperature, and calibration data.	
User Modifiable? No	**Volatile?** No
Memory Erase Processes: None, this is not user accessible.	

Memory Type 2	
Memory Type: Flash Memory	**Memory Size:** 128 M Bit
Memory Function: Stores Device firmware. Images can be changed using the Keysight Soft Front Panel firmware update utility.	
User Modifiable? No	**Volatile?** No
Memory Erase Processes: None, this is not user accessible.	

Memory Type 3	
Memory Type: Flash Memory	**Memory Size:** 128 M Bit
Memory Function: Stores calibration preferences: due date subject to periodic cal, module cal warnings, cal due reminder, and module cal reminder. and passphrase.	
User Modifiable? Yes	**Volatile?** No
Memory Erase Processes: You can clear the passphrase by using the relevant IVI driver code in "M9381A Memory Clear Code" on page 29.	

Memory Type 4	
Memory Type: FPGA	**Memory Size:** N/A
Memory Function: Frequency start/stop/step, power, waveform and impairments.	
User Modifiable? Yes	**Volatile?** Yes
Memory Erase Processes: Cycle power	

Memory Type 5	
Memory Type: Flash Memory	**Memory Size:** 128 M Bit
Memory Function: Stores user customizable asset number and system identification.	
User Modifiable? Yes	**Volatile?** No
Memory Erase Processes: You can clear the asset number and system identification values using the relevant IVI driver code in "M9381A Memory Clear Code" on page 29.	

M9350A PXIe RF Downconverter

The M9350A is only used with the M9391A PXIe VSA.

Memory Type 1	
Memory Type: Flash Memory	**Memory Size:** 128 M Bit
Memory Function: Stores module model number, serial number, manufacturing number, options, PCB part and version numbers, cal verify date, max module temperature, and calibration data.	
User Modifiable? No	**Volatile?** No
Memory Erase Processes: None, this is not user accessible.	

Memory Type 2	
Memory Type: Flash Memory	**Memory Size:** 128 M Bit
Memory Function: Stores Device firmware. Images can be changed using the Keysight Soft Front Panel firmware update utility.	
User Modifiable? No	**Volatile?** No
Memory Erase Processes: None, this is not user accessible.	

Memory Type 3	
Memory Type: Flash Memory	**Memory Size:** 128 M Bit
Memory Function: Stores calibration preferences: due date subject to periodic cal, module cal warnings, cal due reminder, and module cal reminder. and passphrase.	
User Modifiable? Yes	**Volatile?** No
Memory Erase Processes: You can clear the passphrase by using the relevant IVI driver code in "M9391A Memory Clear Code" on page 40.	

Memory Type 4	
Memory Type: FPGA	**Memory Size:** N/A
Memory Function: IQ, spectrum and power settings, advanced options, dither, and alignments.	
User Modifiable? Yes	**Volatile?** Yes
Memory Erase Processes: Cycle power	

Memory Type 5	
Memory Type: Flash Memory	**Memory Size:** 128 M Bit
Memory Function: Stores user customizable asset number and system identification.	
User Modifiable? Yes	**Volatile?** No
Memory Erase Processes: You can clear the asset number and system identification values using the relevant IVI driver code in "M9391A Memory Clear Code" on page 40.	

M9214A PXIe IF Digitizer

The M9214A is only used with the M9391A PXIe VSA.

Memory Type 1	
Memory Type: Flash Memory	**Memory Size:** 128 M Bit
Memory Function: Stores module model number, serial number, manufacturing number, options, PCB part and version numbers, cal verify date, max module temperature, and calibration data.	
User Modifiable? No	**Volatile?** No
Memory Erase Processes: None, this is not user accessible.	

Memory Type 2	
Memory Type: Flash Memory	**Memory Size:** 128 M Bit
Memory Function: Stores Device firmware. Images can be changed using the Keysight Soft Front Panel firmware update utility.	
User Modifiable? No	**Volatile?** No
Memory Erase Processes: None, this is not user accessible.	

Memory Type 3	
Memory Type: Flash Memory	**Memory Size:** 128 M Bit
Memory Function: Stores calibration preferences: due date subject to periodic cal, module cal warnings, cal due reminder, and module cal reminder. and passphrase.	
User Modifiable? Yes	**Volatile?** No
Memory Erase Processes: You can clear the passphrase by using the relevant IVI driver code in "M9391A Memory Clear Code" on page 40.	

Memory Type 4	
Memory Type: FPGA	**Memory Size:** N/A
Memory Function: IQ, spectrum and power settings, advanced options, dither, and alignments.	
User Modifiable? Yes	**Volatile?** Yes
Memory Erase Processes: Cycle power	

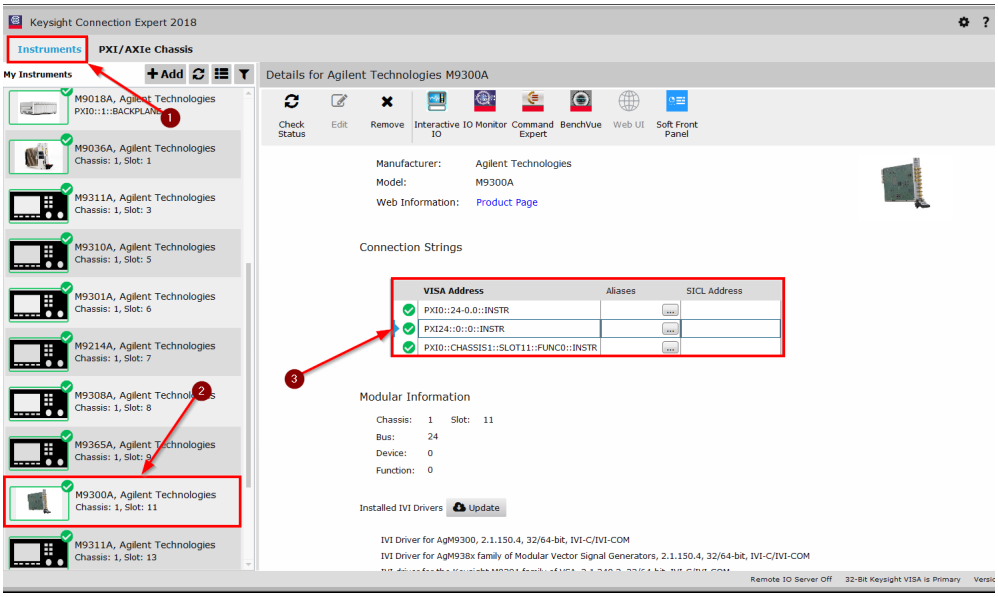
Memory Type 5	
Memory Type: Flash Memory	**Memory Size:** 128 M Bit
Memory Function: Stores user customizable asset number and system identification.	
User Modifiable? Yes	**Volatile?** No
Memory Erase Processes: You can clear the asset number and system identification values using the relevant IVI driver code in "M9391A Memory Clear Code" on page 40.	

VISA Address Tip for Provided Programming Examples

The programming examples below require that the "resource" string be populated with the VISA addresses of your PXI modules.

The VISA address of PXI hardware (including Modules, Controllers, and Chassis) can be found using Keysight Connection Expert.

After opening Connection Expert, view the Instruments tab (1), scroll down the instrument list to find the desired PXI component, and select it (2). The Details window of the Connection Expert page displays the current VISA address details (3).



Note that some of the examples require multiple PXI Hardware VISA addresses to be added. These VISA addresses should be separated by a semicolon ";". Below are examples that show how modules are added as groups.

- M9300A: "PXI0::24-0.0::INSTR"
- M9309A: "PXI0::12-0.0::INSTR"
- M9380A (M9300A, M9301A, & M9310A): "PXI0::25-0.0::INSTR;PXI0::26-0.0::INSTR;PXI0::8-0.0::INSTR"
- M9381A (M9300A, M9301A, M9310A, & M9311A): "PXI0::24-0.0::INSTR;PXI0::25-0.0::INSTR;PXI0::26-0.0::INSTR;PXI0::8-0.0::INSTR"
- M9389A (M9309A and M9310A): "PXI0::12-0.0::INSTR;PXI0::10-0.0::INSTR"
- M9391A (M9300A, M9301A, M9350A, & M9214A): "PXI0::24-0.0::INSTR;PXI0::25-0.0::INSTR;PXI0::26-0.0::INSTR;PXI0::8-0.0::INSTR"

Memory Clear Code Programming Examples

M9381A Memory Clear Code

Below is the IVI code to clear the memory from the M9381A Vector Signal Generator. The procedures in this code sample clear the Asset Number, System ID, and Cal passphrase from the flash memory.

The only step necessary is to copy and paste the code into a console application and include the correct driver references.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Ivi.Driver.Interop;
using Agilent.AgM938x.Interop;


namespace M9381A_Security_Erase
{
    class Program
    {
        static void Main(string[] args)
        {
            //running this program will clear the flash memory of the M9381A Vector
Signal Generator
            //The flash memory cleared is the Asset Number, System ID, and the
passphrase protecting the calibration preferences
           //ONLY run this program if you are sure you want to clear this information
            //initialize the driver
            IAgM938x m9381a = new AgM938x();
           string resource = ""; //enter in the VISA resource between the quotes for
the instrument getting cleared
            string options = "QueryInstrStatus=true, Simulate=false, DriverSetup =
Trace = false";
            bool idquery = true;
            bool reset = true;
            m9381a.Initialize(resource, idquery, reset, options);
            Console.WriteLine("Driver Initialized.\n Press enter to continue\n");
```

```
            Console.ReadLine();
            //test to write to modules. It is commented out because it does not need
to be run to clear the memory
            //m9381aWrite(m9381a.Modules.get_Item("M9300A"));
            //m9381aWrite(m9381a.Modules.get_Item("M9301A"));
            //m9381aWrite(m9381a.Modules.get_Item("M9310A"));
            //m9381aWrite(m9381a.Modules.get_Item("M9311A"));
            //Read back asset numbers and system ID from each module
            string refAsset =
m9381a.Modules.get_Item("M9300A").Nonvolatile.AssetNumber;
            string refID =
            m9381a.Modules.get_Item("M9300A").Nonvolatile.SystemIdentification;
            string synthAsset =
            m9381a.Modules.get_Item("M9301A").Nonvolatile.AssetNumber;
            string synthID =
            m9381a.Modules.get_Item("M9301A").Nonvolatile.SystemIdentification;
            string outputAsset =
            m9381a.Modules.get_Item("M9310A").Nonvolatile.AssetNumber;
            string outputID =
            m9381a.Modules.get_Item("M9310A").Nonvolatile.SystemIdentification;
            string dvmAsset =
m9381a.Modules.get_Item("M9311A").Nonvolatile.AssetNumber;
            string dvmID =
            m9381a.Modules.get_Item("M9311A").Nonvolatile.SystemIdentification;
            Console.WriteLine("Reference Asset is:" + refAsset + "\n");
            Console.WriteLine("Reference System ID is:" + refID + "\n");
            Console.WriteLine("Synthesizer Asset is:" + synthAsset + "\n");
            Console.WriteLine("Synthesizer System ID is:" + synthID + "\n");
            Console.WriteLine("Source Output Asset is:" + outputAsset + "\n");
            Console.WriteLine("Source Output ID is:" + outputID + "\n");
            Console.WriteLine("DVM Asset is:" + dvmAsset + "\n");
            Console.WriteLine("DVM System ID is:" + dvmID + "\n\n");
            //begin clear
            Console.WriteLine("Press Enter to Clear asset number and system ID");
            Console.ReadLine();
```

```
            //clear asset number and system ID and Calibration Preferences passphrase
             m9381aClear(m9381a.Modules.get_Item("M9300A"));
             m9381aClear(m9381a.Modules.get_Item("M9301A"));
             m9381aClear(m9381a.Modules.get_Item("M9310A"));
             m9381aClear(m9381a.Modules.get_Item("M9311A"));
             //read back module asset numbers and ID to verify memory clear
             Console.WriteLine("press enter to verify clear");
             Console.ReadLine();
             refAsset = m9381a.Modules.get_Item("M9300A").Nonvolatile.AssetNumber;
             refID =
m9381a.Modules.get_Item("M9300A").Nonvolatile.SystemIdentification;
             synthAsset = m9381a.Modules.get_Item("M9301A").Nonvolatile.AssetNumber;
             synthID =
m9381a.Modules.get_Item("M9301A").Nonvolatile.SystemIdentification;
             outputAsset = m9381a.Modules.get_Item("M9310A").Nonvolatile.AssetNumber;
             outputID =
             m9381a.Modules.get_Item("M9310A").Nonvolatile.SystemIdentification;
             dvmAsset = m9381a.Modules.get_Item("M9311A").Nonvolatile.AssetNumber;
             dvmID =
m9381a.Modules.get_Item("M9311A").Nonvolatile.SystemIdentification;
             Console.WriteLine("Reference Asset No is:" + refAsset + "\n");
             Console.WriteLine("Reference System ID is:" + refID + "\n");
             Console.WriteLine("Synthesizer Asset No is:" + synthAsset + "\n");
             Console.WriteLine("Synthesizer System ID is:" + synthID + "\n");
             Console.WriteLine("Source Output Asset No is:" + outputAsset + "\n");
             Console.WriteLine("Source Output System ID is:" + outputID + "\n");
             Console.WriteLine("DVM Asset is:" + dvmAsset + "\n");
             Console.WriteLine("DVM System ID is:" + dvmID + "\n\n");
            Console.WriteLine("\n Memory clear complete, press enter to exit program");
             Console.ReadLine();
             //close the driver session
             m9381a.Close();
         }
        //test method to write to the modules. It is commented out because it does not
need to be run to clear the memory
```

```
        //static void m9381aWrite(IAgM938xModule module)
        //{
        // module.Nonvolatile.Clear();
        // module.Nonvolatile.SystemIdentification = "system ID";
        // module.Nonvolatile.AssetNumber = "123456789";
        // string oldPassphrase = module.Nonvolatile.Passphrase;
        // module.Nonvolatile.Write(oldPassphrase);
        //}

        //method to clear the Passphrase and Asset Number/System ID of each module
        static void m9381aClear(IAgM938xModule module)
        {
            module.Nonvolatile.Clear();
            module.Nonvolatile.SystemIdentification = "";
            module.Nonvolatile.AssetNumber = "";
            string newPassphrase = "";
            string oldPassphrase = module.Nonvolatile.Passphrase;
            module.Nonvolatile.Passphrase = newPassphrase;
            module.Nonvolatile.Write(oldPassphrase);
        }
    }
}
```

M9380A Memory Clear Code

Below is the IVI code to clear the memory from the M9380A CW Source. The procedures in this code sample clear the Asset Number, System ID, and Cal passphrase from the flash memory.

The only step necessary is to copy and paste the code into a console application and include the correct driver references.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Ivi.Driver.Interop;
using Agilent.AgM938x.Interop;


namespace M9380A_Security_Erase
{
    class Program
    {
        static void Main(string[] args)
        {
            //running this program will clear the flash memory of the M9380A CW Source
            //The flash memory cleared is the Asset Number, System ID, and the
passphrase protecting the calibration preferences
            //ONLY run this program if you are sure you want to clear this information
            //initialize the driver
            IAgM938x m9380a = new AgM938x();
            string resource = ""; //enter in the VISA resource between the quotes for
the instrument getting cleared
            string options = "QueryInstrStatus=true, Simulate=false, DriverSetup =
Trace = false";
            bool idquery = true;
            bool reset = true;
            m9380a.Initialize(resource, idquery, reset, options);
            Console.WriteLine("Driver Initialized.\n Press enter to continue\n");
            Console.ReadLine();
            //test to write to modules. It is commented out because it does not need
to be run to clear the memory
```

```
            //m9380aWrite(m9380a.Modules.get_Item("M9300A"));
            //m9380aWrite(m9380a.Modules.get_Item("M9301A"));
            //m9380aWrite(m9380a.Modules.get_Item("M9310A"));
            //Read back asset numbers and system ID from each module
            string refAsset =
m9380a.Modules.get_Item("M9300A").Nonvolatile.AssetNumber;
            string refID =
            m9380a.Modules.get_Item("M9300A").Nonvolatile.SystemIdentification;
            string synthAsset =
            m9380a.Modules.get_Item("M9301A").Nonvolatile.AssetNumber;
            string synthID =
            m9380a.Modules.get_Item("M9301A").Nonvolatile.SystemIdentification;
            string outputAsset =
            m9380a.Modules.get_Item("M9310A").Nonvolatile.AssetNumber;
            string outputID =
            m9380a.Modules.get_Item("M9310A").Nonvolatile.SystemIdentification;
            Console.WriteLine("Reference Asset is:" + refAsset + "\n");
            Console.WriteLine("Reference System ID is:" + refID + "\n");
            Console.WriteLine("Synthesizer Asset is:" + synthAsset + "\n");
            Console.WriteLine("Synthesizer System ID is:" + synthID + "\n");
            Console.WriteLine("Source Output Asset is:" + outputAsset + "\n");
            Console.WriteLine("Source Output ID is:" + outputID + "\n");
            //begin clear
            Console.WriteLine("Press Enter to Clear asset number and system ID");
            Console.ReadLine();
           //clear asset number and system ID and Calibration Preferences passphrase
            m9380aClear(m9380a.Modules.get_Item("M9300A"));
            m9380aClear(m9380a.Modules.get_Item("M9301A"));
            m9380aClear(m9380a.Modules.get_Item("M9310A"));
            //read back module asset numbers and ID to verify memory clear
            Console.WriteLine("press enter to verify clear");
            Console.ReadLine();
            refAsset = m9380a.Modules.get_Item("M9300A").Nonvolatile.AssetNumber;
refID =
```

```
            m9380a.Modules.get_Item("M9300A").Nonvolatile.SystemIdentification;
            synthAsset = m9380a.Modules.get_Item("M9301A").Nonvolatile.AssetNumber;
            synthID =
m9380a.Modules.get_Item("M9301A").Nonvolatile.SystemIdentification;
            outputAsset = m9380a.Modules.get_Item("M9310A").Nonvolatile.AssetNumber;
            outputID =
            m9380a.Modules.get_Item("M9310A").Nonvolatile.SystemIdentification;
            Console.WriteLine("Reference Asset No is:" + refAsset + "\n");
            Console.WriteLine("Reference System ID is:" + refID + "\n");
            Console.WriteLine("Synthesizer Asset No is:" + synthAsset + "\n");
            Console.WriteLine("Synthesizer System ID is:" + synthID + "\n");
            Console.WriteLine("Source Output Asset No is:" + outputAsset + "\n");
            Console.WriteLine("Source Output System ID is:" + outputID + "\n");
           Console.WriteLine("\n Memory clear complete, press enter to exit program");
            Console.ReadLine();
            //close the driver session
            m9380a.Close();
        }
       //test method to write to the modules. It is commented out because it does not
need to be run to clear the memory.
        //static void m9380aWrite(IAgM938xModule module)
        //{
        // module.Nonvolatile.Clear();
        // module.Nonvolatile.SystemIdentification = "system ID";
        // module.Nonvolatile.AssetNumber = "123456789";
        // string oldPassphrase = module.Nonvolatile.Passphrase;
        // module.Nonvolatile.Write(oldPassphrase);
        //}
        //method to clear the Passphrase and Asset Number/System ID of each module
        static void m9380aClear(IAgM938xModule module)
        {
            module.Nonvolatile.Clear();
            module.Nonvolatile.SystemIdentification = "";
            module.Nonvolatile.AssetNumber = "";
```

```
            string newPassphrase = "";
            string oldPassphrase = module.Nonvolatile.Passphrase;
            module.Nonvolatile.Passphrase = newPassphrase;
            module.Nonvolatile.Write(oldPassphrase);
        }
    }
}
```

M9389A Memory Clear Code

Below is the IVI code to clear the memory from the M9389A Vector Network Analyzer Source. The procedures in this code sample clear the Asset Number, System ID, and Cal passphrase from the flash memory.

The only step necessary is to copy and paste the code into a console application and include the correct driver references.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Ivi.Driver.Interop;
using Agilent.AgM938x.Interop;


namespace M9389A_Security_Erase
{
    class Program
    {
        static void Main(string[] args)
        {
            //running this program will clear the flash memory of the m9389a Signal
Generator
            //The flash memory cleared is the Asset Number, System ID, and the
passphrase protecting the calibration preferences
           //ONLY run this program if you are sure you want to clear this information
            //initialize the driver
            IAgM938x m9389a = new AgM938x();
           string resource = ""; //enter in the VISA resource between the quotes for
the instrument getting cleared
            string options = "QueryInstrStatus=true, Simulate=false,
DriverSetup=Trace=false";
            bool idquery = true;
            bool reset = true;
            m9389a.Initialize(resource, idquery, reset, options);
            Console.WriteLine("Driver Initialized.\n Press enter to continue\n");
            Console.ReadLine();
```

```
            //test to write to modules. It is commented out because it does not need
to be run to clear the memory
            //m9389aWrite(m9389a.Modules.get_Item("M9309A"));
            //m9389aWrite(m9389a.Modules.get_Item("M9310A"));


            //Read back asset numbers and system ID from each module
            string synthAsset =
m9389a.Modules.get_Item("M9309A").Nonvolatile.AssetNumber;
            string synthID =
m9389a.Modules.get_Item("M9309A").Nonvolatile.SystemIdentification;
            string outputAsset =
m9389a.Modules.get_Item("M9310A").Nonvolatile.AssetNumber;
            string outputID =
m9389a.Modules.get_Item("M9310A").Nonvolatile.SystemIdentification;
            Console.WriteLine("Synthesizer Asset is:" + synthAsset + "\n");
            Console.WriteLine("Synthesizer System ID is:" + synthID + "\n");
            Console.WriteLine("Source Output Asset is:" + outputAsset + "\n");
            Console.WriteLine("Source Output ID is:" + outputID + "\n");
            //begin clear
            Console.WriteLine("Press Enter to Clear asset number and system ID");
            Console.ReadLine();
            //clear asset number and system ID and Calibration Preferences passphrase
            M9389A_Clear(m9389a.Modules.get_Item("M9309A"));
            M9389A_Clear(m9389a.Modules.get_Item("M9310A"));
            //read back module asset numbers and ID to verify memory clear
            Console.WriteLine("press enter to verify clear");
            Console.ReadLine();
            synthAsset = m9389a.Modules.get_Item("M9309A").Nonvolatile.AssetNumber;
            synthID =
m9389a.Modules.get_Item("M9309A").Nonvolatile.SystemIdentification;
            outputAsset = m9389a.Modules.get_Item("M9310A").Nonvolatile.AssetNumber;
            outputID =
m9389a.Modules.get_Item("M9310A").Nonvolatile.SystemIdentification;
            Console.WriteLine("Synthesizer Asset No is:" + synthAsset + "\n");
            Console.WriteLine("Synthesizer System ID is:" + synthID + "\n");
            Console.WriteLine("Source Output Asset No is:" + outputAsset + "\n");
```

```
            Console.WriteLine("Source Output System ID is:" + outputID + "\n");
           Console.WriteLine("\n Memory clear complete, press enter to exit program");
            Console.ReadLine();
            //close the driver session
            m9389a.Close();
        }
       //test method to write to the modules. It is commented out because it does not
need to be run to clear the memory
        //static void m9389aWrite(IAgM938xModule module)
        //{
        // module.Nonvolatile.Clear();
        // module.Nonvolatile.SystemIdentification = "system ID";
        // module.Nonvolatile.AssetNumber = "123456789";
        // string oldPassphrase = module.Nonvolatile.Passphrase;
        // module.Nonvolatile.Write(oldPassphrase);
        //}
        //method to clear the Passphrase and Asset Number/System ID of each module
        static void M9389A_Clear(IAgM938xModule module)
        {
            module.Nonvolatile.Clear();
            module.Nonvolatile.SystemIdentification = "";
            module.Nonvolatile.AssetNumber = "";
            string newPassphrase = "";
            string oldPassphrase = module.Nonvolatile.Passphrase;
            module.Nonvolatile.Passphrase = newPassphrase;
            module.Nonvolatile.Write(oldPassphrase);
        }

    }
}
```

M9391A Memory Clear Code

Below is the IVI code to clear the memory from the M9391A Vector Signal Analyzer. The procedures in this code sample clear the Asset Number, System ID, and Cal passphrase from the flash memory.

The only step necessary is to copy and paste the code into a console application and include the correct driver references.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Ivi.Driver.Interop;
using Agilent.AgM9391.Interop;


namespace M9391A_Security_Erase
{
    class Program
    {
        static void Main(string[] args)
        {
            //Running this program will clear the flash memory of the M9391A Vector
Signal Analyzer multi-module instrument.
            //The flash memory cleared is the Asset Number, System ID, and the
passphrase protecting the calibration preferences.
           //ONLY run this program if you are sure you want to clear this information.
            //initialize the driver
            IAgM9391 m9391a = new AgM9391();
            string resource = "";//enter in the VISA resource between the quotes for
the instrument getting cleared
            string options = "QueryInstrStatus=true, Simulate=false, DriverSetup =
Trace = false";
            bool idquery = true;
            bool reset = true;
            m9391a.Initialize(resource, idquery, reset, options);
            Console.WriteLine("Driver Initialized.\n Press enter to continue\n");
            Console.ReadLine();
```

```
            //Test to write to modules. It is commented out because it does not need
to be run to clear the memory.
            //m9391aWrite(m9391a.Modules.get_Item("M9300A"));
            //m9391aWrite(m9391a.Modules.get_Item("M9301A"));
            //m9391aWrite(m9391a.Modules.get_Item("M9350A"));
            //m9391aWrite(m9391a.Modules.get_Item("M9214A"));
            //Read back asset numbers and system ID from each module
            string refAsset =
m9391a.Modules.get_Item("M9300A").Nonvolatile.AssetNumber;
            string refID =
            m9391a.Modules.get_Item("M9300A").Nonvolatile.SystemIdentification;
            string synthAsset =
            m9391a.Modules.get_Item("M9301A").Nonvolatile.AssetNumber;
            string synthID =
            m9391a.Modules.get_Item("M9301A").Nonvolatile.SystemIdentification;
            string DCAsset =
m9391a.Modules.get_Item("M9350A").Nonvolatile.AssetNumber;
            string DCID =
            m9391a.Modules.get_Item("M9350A").Nonvolatile.SystemIdentification;
            string digAsset =
m9391a.Modules.get_Item("M9214A").Nonvolatile.AssetNumber;
            string digID =
            m9391a.Modules.get_Item("M9214A").Nonvolatile.SystemIdentification;
            Console.WriteLine("Reference Asset is:" + refAsset + "\n");
            Console.WriteLine("Reference System ID is:" + refID + "\n");
            Console.WriteLine("Synthesizer Asset is:" + synthAsset + "\n");
            Console.WriteLine("Synthesizer System ID is:" + synthID + "\n");
            Console.WriteLine("DownConverter Asset is:" + DCAsset + "\n");
            Console.WriteLine("DownConverter System ID is:" + DCID + "\n");
            Console.WriteLine("Digitizer Asset is:" + digAsset + "\n");
            Console.WriteLine("Digitzer System ID is:" + digID + "\n\n");
            //Begin clear
            Console.WriteLine("Press Enter to Clear asset number and system ID");
            Console.ReadLine();
           //Clear asset number and system ID and Calibration Preferences passphrase.
```

```
            m9391aClear(m9391a.Modules.get_Item("M9300A"));
            m9391aClear(m9391a.Modules.get_Item("M9301A"));
            m9391aClear(m9391a.Modules.get_Item("M9350A"));
            m9391aClear(m9391a.Modules.get_Item("M9214A"));
            //Read back module asset numbers and ID to verify memory clear.
            Console.WriteLine("press enter to verify clear");
            Console.ReadLine();
            refAsset = m9391a.Modules.get_Item("M9300A").Nonvolatile.AssetNumber;
            refID =
m9391a.Modules.get_Item("M9300A").Nonvolatile.SystemIdentification;
            synthAsset = m9391a.Modules.get_Item("M9301A").Nonvolatile.AssetNumber;
            synthID =
m9391a.Modules.get_Item("M9301A").Nonvolatile.SystemIdentification;
            DCAsset = m9391a.Modules.get_Item("M9350A").Nonvolatile.AssetNumber;
           DCID = m9391a.Modules.get_Item("M9350A").Nonvolatile.SystemIdentification;
            digAsset = m9391a.Modules.get_Item("M9214A").Nonvolatile.AssetNumber;
            digID =
m9391a.Modules.get_Item("M9214A").Nonvolatile.SystemIdentification;
            Console.WriteLine("Reference Asset No is:" + refAsset + "\n");
            Console.WriteLine("Reference System ID is:" + refID + "\n");
            Console.WriteLine("Synthesizer Asset No is:" + synthAsset + "\n");
            Console.WriteLine("Synthesizer System ID is:" + synthID + "\n");
            Console.WriteLine("DownConverter Asset No is:" + DCAsset + "\n");
            Console.WriteLine("DownConverter System ID is:" + DCID + "\n");
            Console.WriteLine("Digitizer Asset is:" + digAsset + "\n");
            Console.WriteLine("Digitzer System ID is:" + digID + "\n\n");
           Console.WriteLine("\n Memory clear complete, press enter to exit program");
            Console.ReadLine();
            //Close the driver session.
            m9391a.Close();
        }
       //Test method to write to the modules. It is commented out because it does not
need to be run to clear the memory.
        //static void m9391aWrite(IAgM9391Module module)
        //{
```

```
        //module.Nonvolatile.Clear();
        //module.Nonvolatile.SystemIdentification = "system ID";
        //module.Nonvolatile.AssetNumber = "123456789";
        //string oldPassphrase = module.Nonvolatile.Passphrase;
        //module.Nonvolatile.Write(oldPassphrase);
        //}
        //Method to clear the Passphrase and Asset Number/System ID of each module.
        static void m9391aClear(IAgM9391Module module)
        {
            module.Nonvolatile.Clear();
            module.Nonvolatile.SystemIdentification = "";
            module.Nonvolatile.AssetNumber = "";
            string newPassphrase = "";
            string oldPassphrase = module.Nonvolatile.Passphrase;
            module.Nonvolatile.Passphrase = newPassphrase;
            module.Nonvolatile.Write(oldPassphrase);
        }
    }
}
```

M9300A Memory Clear Code

Below is the IVI code to clear the memory from the M9300A Frequency Reference. The procedures in this code sample clear the Asset Number, System ID, and Cal passphrase from the flash memory.

The only step necessary is to copy and paste the code into a console application and include the correct driver references.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Ivi.Driver.Interop;
using Agilent.AgM9300.Interop;


namespace M9300A_Security_Erase
{
    class Program
    {
        static void Main(string[] args)
        {
           // Running this program will clear the flash memory of the M9300A Reference
            // The flash memory cleared is the Asset Number, System ID, and the
passphrase, protecting the calibration preferences
           // ONLY run this program if you are sure you want to clear this information
            // Initialize the driver
            IAgM9300 m9300a = new AgM9300();
          string resource = ""; // Enter the VISA resource between the quotes for the
instrument getting cleared
            string options = "QueryInstrStatus=true, Simulate=false,
DriverSetup=Trace=false";
            bool idquery = true;
            bool reset = true;
          m9300a.Initialize(resource, idquery, reset, options); // This will fail if
"resource" is not entered above
            Console.WriteLine("Driver Initialized.\n Press enter to continue\n");
            Console.ReadLine();
```

```
            // Test method to write to the module. It is commented out because it does
not need to be run to clear the memory
             //m9300aWrite(m9300a);
             // Read back asset number and system ID from the module
             string refAsset = m9300a.Nonvolatile.AssetNumber;
             string refID = m9300a.Nonvolatile.SystemIdentification;
             Console.WriteLine("Reference Asset is:" + refAsset + "\n");
             Console.WriteLine("Reference System ID is:" + refID + "\n");

             // Begin clear
             Console.WriteLine("Press Enter to Clear asset number and system ID");
             Console.ReadLine();
            // Clear asset number and system ID and Calibration Preferences passphrase
             m9300aClear(m9300a);
             // Read back module asset number and ID to verify memory clear
             Console.WriteLine("Press enter to verify clear");
             Console.ReadLine();
             refAsset = m9300a.Nonvolatile.AssetNumber;
             refID = m9300a.Nonvolatile.SystemIdentification;
             Console.WriteLine("Reference Asset No is:" + refAsset + "\n");
             Console.WriteLine("Reference System ID is:" + refID + "\n");
            Console.WriteLine("\n Memory clear complete, press enter to exit program");
             Console.ReadLine();
             // Close the driver session
             m9300a.Close();
        }
       // Test method to write to the module. It is commented out because it does not
need to be run to clear the memory
        //static void m9300aWrite(IAgM9300 module)
        //{
        // module.Nonvolatile.Clear();
        // module.Nonvolatile.SystemIdentification = "system ID";
        // module.Nonvolatile.AssetNumber = "123456789";
        // string oldPassphrase = module.Nonvolatile.Passphrase;
```

```
        // module.Nonvolatile.Write(oldPassphrase);
        //}
        // Method to clear the Passphrase and Asset Number/System ID
        static void m9300aClear(IAgM9300 module)
        {
            module.Nonvolatile.Clear();
            module.Nonvolatile.SystemIdentification = "";
            module.Nonvolatile.AssetNumber = "";
            string newPassphrase = "";
            string oldPassphrase = module.Nonvolatile.Passphrase;
            module.Nonvolatile.Passphrase = newPassphrase;
            module.Nonvolatile.Write(oldPassphrase);

        }
    }
}
```

M9309A Memory Clear Code

Below is the IVI code to clear the memory from the M9309A Vector Network Analyzer Synthesizer. The procedures in this code sample clear the Asset Number, System ID, and Cal passphrase from the flash memory.

The only step necessary is to copy and paste the code into a console application and include the correct driver references.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Ivi.Driver.Interop;
using Keysight.KtM9301.Interop;



namespace M9309A_Security_Erase
{
    class Program
    {
        static void Main(string[] args)
        {
            //running this program will clear the flash memory of the m9309a Signal
Generator
            //The flash memory cleared is the Asset Number, System ID, and the
passphrase protecting the calibration preferences
           //ONLY run this program if you are sure you want to clear this information
            //initialize the driver
            KtM9301 m9309a = new KtM9301();
           string resource = ""; //enter in the VISA resource between the quotes for
the instrument getting cleared
            string options = "QueryInstrStatus=true, Simulate=false,
DriverSetup=Trace=false";
            bool idquery = true;
            bool reset = true;
            m9309a.Initialize(resource, idquery, reset, options);
            Console.WriteLine("Driver Initialized.\n Press enter to continue\n");
            Console.ReadLine();
```

```
            //test to write to modules. It is commented out because it does not need
to be run to clear the memory
            //m9309aWrite(m9309a);
            //m9309aWrite(m9309a);

            //Read back asset numbers and system ID from each module
            string synthAsset = m9309a.Nonvolatile.AssetNumber;
            string synthID = m9309a.Nonvolatile.SystemIdentification;
            Console.WriteLine("Synthesizer Asset is:" + synthAsset + "\n");
            Console.WriteLine("Synthesizer System ID is:" + synthID + "\n");
            //begin clear
            Console.WriteLine("Press Enter to Clear asset number and system ID");
            Console.ReadLine();
            //clear asset number and system ID and Calibration Preferences passphrase
            m9309a_Clear(m9309a);
            //read back module asset numbers and ID to verify memory clear
            Console.WriteLine("press enter to verify clear");
            Console.ReadLine();
            synthAsset = m9309a.Nonvolatile.AssetNumber;
            synthID = m9309a.Nonvolatile.SystemIdentification;
            Console.WriteLine("Synthesizer Asset No is:" + synthAsset + "\n");
            Console.WriteLine("Synthesizer System ID is:" + synthID + "\n");
           Console.WriteLine("\n Memory clear complete, press enter to exit program");
            Console.ReadLine();
            //close the driver session
            m9309a.Close();
        }
       //test method to write to the modules. It is commented out because it does not
need to be run to clear the memory
        //static void m9309aWrite(IKtM9301 module)
        //{
        //    module.Nonvolatile.Clear();
        //    module.Nonvolatile.SystemIdentification = "system ID";
        //    module.Nonvolatile.AssetNumber = "123456789";
```

```
        //    string oldPassphrase = module.Nonvolatile.Passphrase;
        //    module.Nonvolatile.Write(oldPassphrase);
        //}
        //method to clear the Passphrase and Asset Number/System ID of each module
        static void m9309a_Clear(IKtM9301 module)
        {
            module.Nonvolatile.Clear();
            module.Nonvolatile.SystemIdentification = "";
            module.Nonvolatile.AssetNumber = "";
            string newPassphrase = "";
            string oldPassphrase = module.Nonvolatile.Passphrase;
            module.Nonvolatile.Passphrase = newPassphrase;
            module.Nonvolatile.Write(oldPassphrase);
        }
    }
}
```



This information is subject to change without notice.
© Keysight Technologies 2014-2022

Edition 1, January 2022

M9300-90021

www.keysight.com

Artisan Technology Group is an independent supplier of quality pre-owned equipment

Gold-standard solutions

Extend the life of your critical industrial, commercial, and military systems with our superior service and support.

We buy equipment

Planning to upgrade your current equipment? Have surplus equipment taking up shelf space? We'll give it a new home.

Learn more!

Visit us at **artisantg.com** for more info on price quotes, drivers, technical specifications, manuals, and documentation.

Artisan Scientific Corporation dba Artisan Technology Group is not an affiliate, representative, or authorized distributor for any manufacturer listed herein.

We're here to make your life easier. How can we help you today?

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

ARTISAN® TECHNOLOGY GROUP