O_{CC}

MICAS-X User Manual

version 3.3.0

1 Table of Contents

2 Introduction

MICAS-X, the Multi-Instrument Control and Acquisition System – eXtended, is a LabVIEW data acquisition and control program designed for acquiring data from and

controlling a wide range of instruments. MICAS-X is both configuration-based and custom-programmable, which is to say that many instruments and systems can be run in MICAS-X simply by configuring existing modules, but writing new LabVIEW components for more complex, custom situations can provide additional functionality. MICAS-X is designed to bridge the gap between using a pre-written, configuration-based program for data acquisition, and writing a complete, custom application. By providing a wide range of necessary functionality in its existing code base, MICAS-X can greatly accelerate the development of data acquisition software. By allowing modular expansion, it can also accommodate a vast range of capabilities specific to many different situations.

MICAS-X can be supplied as a compiled program, which does not need to have LabVIEW installed to run, or as LabVIEW source code run within the LabVIEW environment. Since MICAS-X source code is available to the end-user, and the MICAS-X API's and interfaces are documented, a LabVIEW programmer of moderate to advanced abilities can create custom modules for MICAS-X. Note, however, that MICAS-X does require a hardware security key for it to be properly licensed and functional. Without a hardware key, the Limited Demo version of MICAS-X will operate fully for 20 minutes. Note that the full version of MICAS-X will continue to operate even if the hardware key is missing, but will periodically remind the operator of the need to license the software. The low-cost Ltd version of MICAS-X will not start running if the license key is not present. In order to enforce the hardware key, a small amount of source code is password protected and unavailable to the end-user.

As of version 2.1, MICAS-X-RT allows MICAS-X to run on most National Instruments LabVIEW Real-Time embedded platforms. MICAS-X-RT provides much of the same functionality as MICAS-X, with the exception of the user interface, since RT platforms are designed to normally operate headless. A Scan Engine Driver provides easy access to most IO supported by the Scan Engine interface, and a MICAS-XRT Driver, which can be loaded into MICAS-X on Windows, allows seamless interfacing between the Windows program and the robust, real-time RT program.

A limited license of MICAS-X is designed to address situations for which the full license would be more expensive than a custom written LabVIEW program. These cases include fairly simple data logger and control systems for which most of the required functionality is already available in MICAS-X. For such relatively simple systems, MICAS-X can provide all the needed functionality and more, but a standard license may well cost more than writing a custom program. By using the Ltd (limited) license of MICAS-X, the application can take advantage of the robust, debugged code base of MICAS-X and while keeping the cost in line with the development of a much less functional custom program. The Ltd license is available only in the compiled version of MICAS-X. (Any application requiring source code access is inherently a better match to the full version

of MICAS-X, which provides more flexibility for the application.) The Ltd license allows only Ltd configurations to be run in MICAS-X. Any configuration file that has not been created specifically for the Ltd license will not allow MICAS-X Ltd to run. Similarly, any license key that has been created for an Ltd license will not allow any non-Ltd configuration file to run in MICAS-X. Furthermore, each Ltd license will specify the exact MICAS-X modules that are allowed for that license. The configuration for the Ltd license can be edited as needed, but only to the extent that it uses the modules that the specific Ltd license allows. Any configuration file that has been created for an Ltd license can only be edited by the MICAS-X Configuration Editor. (Although standard MICAS-X configuration files can be edited in a text editor, it is recommended that the MICAS-X Configuration Editor always be used to ensure valid configurations.)

Another licensing option for MICAS-X is the Student Edition (SE) license. This license functions the same as the Ltd license: e.g. it requires a SE license key to be present for MICAS-X to run, and it only allows SE configuration files to be run. Unlike the Ltd license, the SE license is available only for the LabVIEW source code version, not the executable version. This license is intended for undergraduate or graduate level students in the sciences, engineering, or similar disciplines who need a relatively easy way to control experiments and collect data. As such, it is free for student use, either for classroom work or for research. This license includes all the Drivers that come with the base MICAS-X license, as well as additional NIDaq Drivers. With MICAS-X-SE and an inexpensive NI Daq board, one can quickly configure analog input, analog output, digital input, digital output, and much more.

MICAS-X version 3 is written in LabVIEW 2018 and is supported in Windows 7 and Windows 10. Although no longer directly supported, MICAS-X version 1 has also been used successfully on Windows XP and Window 8. The MICAS-X Drivers for NIDaq Devices (NIDaqAD, NIDaqDA, NIDaqDI, NIDaqDO, and NIDaqCounters) support basic Daq functionality of any National Instruments data acquisition device that is supported by the National Instruments DAQmx hardware driver.

3 MICAS-X Overview

3.1 Program Structure

The overall design of MICAS-X is based on the assumption that many instruments and systems share a common set of infrastructure needs. Typically, these include some slow (usually 1 Hz) monitoring of “housekeeping” or instrument health data, logging that data to a file, sequencing through various states and modes of operation, setting outputs to the proper values for those states and modes, alarming off of conditions that are out of range, logging errors, and so on. In addition, many systems contain one or

more processes that are far outside the range of these features. Often, there will be a core functionality that involves high speed data acquisition, highly-customized control or timing, complex data structures, or other complex operations. MICAS-X is intended to accelerate the development of these types of systems by providing a re-useable set of infrastructure, while also allowing the creation of customized modules that can handle the complex operations. Although this pattern of accelerated development of advanced systems is the primary market for MICAS-X, the infrastructure that has been developed to support that goal results in a program that is applicable to many other experiments and systems as well. Many quick laboratory or prototype tests can be configured using just the components of MICAS-X that are already developed. Thus, due to its flexible nature, MICAS-X can be configured to accommodate many data acquisition, instrument monitoring, and experimental sequences with little or no custom programming needed.

The MICAS-X program itself serves as the coordinating shell for various modules. The Configuration Editor is used to define what modules are included and how each of them is configured to operate. Multiple configuration files can be created, allowing MICAS-X to take on many different personalities for different situations. Once MICAS-X is run, it reads the selected configuration file and starts each of the needed modules. The Acquisition module contains one or more loops which acquire data from the selected Driver modules. The Triggers module continuously monitors the data acquired by the Acquisition module and executes any commands that are configured to be triggered by user-defined conditions based on the data. The Sequences module executes any predefined sequences, changing output channels as needed to step the system through various states.

By only loading the modules specified in the configuration file, MICAS-X can support a huge amount of functionality, yet the program will only contain those resources needed for the current situation.

3.2 Components

MICAS-X includes numerous components, many of which are designed as plug-in modules. With this architecture, new features can be added to a MICAS-X system simply by copying folders of source code to the appropriate destinations and then editing the configuration file to use the new components. The primary MICAS-X Components are:

- MICAS-X.vi - the main control program. This VI controls the entire MICAS-X system, starts and stops modules, and provides the primary interface for the operator to interact with MICAS-X.

- Modules (7. 4) - a set of functions that is always included in MICAS-X. New modules can only be added by creating a new version of MICAS-X. This is

normally done only by OCC. Modules include Acquisition, Triggers, Sequences, File Writing, Commands, and others.

- Drivers (7. 5) - software interfaces to hardware devices and other functionality for reading data and controlling outputs. Drivers are intended for handling "slow" data, with rates on the order of once per minute to 20 Hz, with 1 Hz being typical.

- Instruments (7. 5) - components that incorporate specialized functions, often used for high-speed data acquisition or control, or other types of acquisition or control that do not fit into the Driver model. Each Instrument can also include its own Display.

- Displays (7. 8) - components that display data to the end user or allow access to MICAS-X functionality, but which may not be directly tied to a specific Instrument or Driver.

- Calculations (7. 5. 4) - VI's that can be written by the end-user, according to a well-defined API, which can incorporate more complex calculations than can be created using the Equations Driver (which allows text calculations). Calculations are evaluated by the Calculations Driver.

- Utilities (10) - top-level programs that can be run inside or outside of MICAS-X and which provide additional functionality of a type that is not required for MICAS-X to run. Examples are programs that review data or log files or that edit the configuration file.

- Configuration File (7) - a text-based file that defines what modules will be included when MICAS-X runs and how MICAS-X is set up to run.

- MICAS-X-RT (3. 11) – a version of MICAS-X that runs directly on a LabVIEW RT platform, providing an extremely robust, real-time program for control and acquisition, with direct integration with MICAS-X on a Windows platform. Many of the above modules function on MICAS-X-RT, including Sequences, Triggers and Alarms, File Writing, Commands, Event and Error Logging, many Drivers, and select Instruments. Displays and Utilities, which are inherently user-interface modules, are not supported under MICAS-X-RT, but the Configuration Editor (run on a Windows machine) is used to configure MICAS-X-RT for operation.

3.3 Channels

Much of the predefined functionality of MICAS-X is focused on the data Channels. Triggers compare Channel values to trigger conditions, and can set Channels to new values if the condition is met. Sequences can operate on Channel values, as can Commands. It is important, therefore, to understand where these Channels come from, and what their features and limitations are.

As described in section 3.5, all the MICAS-X Channel values are stored in a one-dimensional array of double-precision reals. This buffer contains the most recent value of every channel. The historical record of these Channels' values is recorded in various data files, described in section 3.7. The Acquisition module (section 3.4) is in charge of keeping the buffer up-to-date.

With a few exceptions, all Channels originate in MICAS-X from Drivers. Instruments, Displays, and Utilities cannot supply MICAS-X with Channels. Drivers, however, can operate on two types of Channels, Input Channels and Controllers (or Output Channels). (Note that the Controllers Driver refers to a way to implement PID and similar control loops, and does not refer to a Driver that handles all Controller Channels.) Input Channels are data that is read into MICAS-X, typically from a hardware device, but sometimes from another source (such as in the case of the Equations Driver and the Calculations Driver). Controller Channels are Channels that the operator or MICAS-X can write to. Typically, these would be output channels of a hardware device, such as an analog output voltage. But Controller Channels also include things like Manual Driver Channels and Globals, both of which are simply types of program variables for MICAS-X.

There are a few special Channels that do not come from Drivers. Globals (see section 9.6.13) are configured as a separate Module, not as a Driver. Program State Variables (section 9.6.14) are special Channels that MICAS-X always supplies, and are not configured in any Driver. Constants (section 9.6.16) can be used in a way similar to Input Channels, but are not logged to file, and are configured as a Module, not as a Driver.

3.4 Acquisition

Many of the functions of MICAS-X are designed around the data acquired and controlled by the Driver modules. Although custom modules can be written to handle much more complex data requirements, the Driver data is intended to provide the information and control MICAS-X needs to operate the system.

Any number of Drivers can be added to the MICAS-X configuration. Numerous Drivers are included in the base package, many more can be purchased separately, and custom Drivers can be created very cost-effectively. Many Drivers are written in a way that allows them to be instantiated more than once in a configuration. For example, the

NIDaqAD Driver acquires analog data from most National Instruments Daq devices. It can be instantiated multiple times, once for each Daq device attached to the system.

Within MICAS-X, the Drivers are controlled by the Acquisition module. The Acquisition module can dynamically create any number of Driver loops, and each loop can run at its own rate. Thus devices that each run at separate acquisition rates can all co-exist within MICAS-X. Data from a GPS may be acquired at 1 Hz, while an NI Daq device might acquire data at 10 Hz. Each loop can also control the writing of one or more data files. Thus the data from the different frequency loops can all be archived with no loss.

Drivers support three main forms of functionality: Aquire, Set, and Command. The Acquisition module calls the Acquire function of each Driver at the specified loop rate. This reads all the input channels that the Driver supports. The Set function can be called by numerous parts of the MICAS-X system, including Sequences, Triggers, and the user interface. The Set function controls the output channels that the Driver supports. The Command function allows the Driver to include additional functionality that doesn't fit in well to the paradigm of input and output channels. An example would be a motion control Driver. The Acquire function could read the current position of the motion device. The Set function would tell the device where to move to next. Then custom Commands would be used to start and stop the motion. Although many Drivers support only Acquire or only Set functions, Drivers can include any or all three of these functions.

MICAS-X includes an Acquire Command, which is often mapped to a switch at the top of the screen. When Acquire is Off, the Drivers are not queried for new data. When Acquire is On, the Acquisition Loops are running and the Driver data is updated at the defined loop rates. Drivers also have Enable and Disable Commands. Whereas Aquire turns all the Drivers on or off at once, Enable and Disable can be used on a single Driver. Disable can also be used if one device fails in hardware, so that MICAS-X ignores that device when it is disabled and prevents unnecessary redundant errors from being logged.

3.5 Data Types

For simplicity and flexibility, MICAS-X uses a single data type for the input and output data that the Acquisition module handles. This data is stored as a one-dimensional array of double-precision (8 byte) reals. When a device requires another numeric format (such as a 2 byte, unsigned integer), the data is coerced in or out of reals. For Boolean (true/false) data, MICAS-X uses the standard convention of False = 0, True = 1. In addition, any other non-zero value is also interpreted as True.

The IEEE definition of reals allows for the special values of +Infinity, -Infinity, and NaN (Not-a-Number). MICAS-X therefore supports these values as well. +Infinity and -

Infinity can be useful for moving the Thresholds of Triggers to a value that can never by attained, effectively turning the Trigger off until the Threshold is reset to another value. NaN is used in MICAS-X to indicate missing values. Some Drivers may return NaN within their data if communication to a device failed. In addition, all input channels for all Drivers are initialized to NaN when MICAS-X starts up, and all Channels return a value of NaN when Acquire is off or a Driver is Diabled.

The value of NaN causes complications when channels that represent Boolean data are used, since the normal interpretation of <>0 (not equal to 0) will return a True if the channel value is NaN. It is often the case that an uninitialized Boolean channel should not default to True. Within MICAS-X, the test (<>0 AND <>NaN) is used to avoid this situation. In addition, Triggers and Sequences support several special conditions to enable the proper handling of NaN.

Within MICAS-X, there is a buffer that includes all the most recent values of all the Driver channels. This buffer is updated by each Acquisition loop. In addition, any command to a Driver to Set an output channel updates that channel's value in the buffer when the channel is updated. (E.g. the output channels do not wait for the Acquisition loop (which updates the input channels) before updating the output channels in the buffer.) Whenever MICAS-X needs to know a value of any of the Driver channels, the buffer is queried. For optimal performance, a query to the buffer for one channel also returns the entire array of all channel values. Thus, when MICAS-X needs several channel values, one channel can be queried directly from the buffer, while the other channels are then pulled out of the full array of data that was returned. This allows the values numerous channels to be obtained while only making a single call to the buffer.

By limiting the Driver data to a single data type, and by defining a universal Driver software interface, MICAS-X can take optimal advantage of the data acquired and set by the Drivers. Any Driver written to the MICAS-X specification will have its data fully integrated into the MICAS-X system. That data can be logged, displayed, and used throughout MICAS, including within the Trigger and Sequence logic.

Although the Acquisition module is limited to this single data type, the modular, customizable nature of MICAS-X readily allows for the handling of more complex data types. Custom Instruments can be added to MICAS-X which can handle a nearly unlimited scope of data types.

3.6 Channel Lists

Since MICAS-X can include many Drivers, and each Driver can contain many channels, it is easy to create systems with hundreds of data channels. Having more than a few dozen channels can create user interface challenges. For instance, if there is a time-series graph for which the user can select a channel to display from a drop-down menu, it can be difficult to find the desired channel in a list of hundreds of channels.

MICAS-X resolves this issue by allowing the operator to configure any number of Channel Lists. Key elements of MICAS-X then operate on a selected List. A time-series graph may be configured so that it only presents channels from one List. A data logging file can be configured to write only the channels from another list. In this way, MICAS-X streamlines the user interface and provides additional flexibility for dealing with large numbers of channels.

3.7 File Storage

MICAS-X utilizes many different data files. As mentioned above, the configuration file (with an extension ".ini") defines how MICAS-X will operate. The "Config to Load.txt" file tells MICAS-X which configuration file to load. When MICAS-X starts running, it immediately archives the configuration file in a date-named data folder. This way the data acquired will always be in the context of how the system was configured when the data was acquired.

MICAS-X automatically creates a log file in the date-named data folder. The log file notes various software events, such as when the main program and each module starts and stops, the version of each module, and any errors that MICAS-X encounters. In addition, many other events can be configured so that they are logged, such as Sequence steps, Alarms, and Triggers. Operator notes and messages from other programs can also be sent to the log file. The log file therefore can provide a valuable record of how an experiment ran, and can also be an invaluable debugging tool when developing new modules or setting up new configurations.

Whenever MICAS-X is running, all the Driver data is automatically stored in a .tdms data file. (TDMS (Technical Data Management - Streaming) is an efficient, flexible, searchable file format created by National Instruments.) This data file serves as the ultimate archive of all the Driver data, and is used by MICAS-X for other purposes, including the History Data described below. In addition, a TDMS File Reader utility is included as a Utility for reviewing the data in this file. It is important to note that the TDMS file is written by the default Acquisition Loop (Loop 0). Thus if multiple loops are configured, it is important to consider which device(s) are configured to use Loop 0. Putting the fastest loop rate in loop 0 will ensure that all the data from the fastest drivers is logged to the .tdms file, which could be desirable, but which also has the effect of increasing the disk space used by the .tdms file. Putting the fastest devices in another loop will result in loss of fast data in the .tdms file, but other file logging can be enabled in the faster loop such that only the fast device data is logged to a separate file. This configuration ensures that no fast data is lost, yet also prevents the .tdms file from growing unnecessarily. Finally, note that the .tdms file is automatically restarted every three hours. This prevents any one .tmds file from growing too large.

Besides the .tmds file, it is possible to configure any number of additional text-based data files, or comma-separated-variable (.csv) files. Each .csv file can be

configured to save a specific Channel List, and can be configured to be written by any Acquisition Loop. This allows precise control of how and when data is logged.

MICAS-X includes a Record Command, which is often mapped to a switch at the top of the screen. Note that the Record Command only acts to turn file writing on and off for the .csv data files. The .tdms data file is always being written to when Acquire is On, regardless of the value of Record. Additional Commands can turn recording on and off for individual .csv files.

MICAS-X also makes uses of several other types of data files. A menu option allows one to save screen shots of the MICAS-X display in .png graphics files. Sequences can be saved to and read from .seq files. Custom MICAS-X modules can create any other types of files that are needed as well.

3.8 History Data

One of the primary uses of the .tdms data file is to provide history data for time-series graphs. Time-series graphs are present in many places in MICAS-X, including the Control Tab, various Displays, and Quick Graphs. Most time-series graphs allow the operator to select the Channel(s) to display from a List of Channels. Each time-series graph has a local data buffer associated with it. The program adds new data points for the selected channels to the data buffer every time the Acquisition module acquires new data. As long as the Channels displayed are not changed, the time-series graph utilizes the main data buffer for new data and its local buffer for history data. When the user changes the Channel(s) being displayed, the time-series graph accesses the .tmds file to read the history data for the new Channel(s). This history data is placed in the local buffer, and the graph proceeds as before.

Several aspects of this history data mechanism deserve note. First of all, there is a configuration option that tells MICAS-X whether to read a single .tdms file or to gather history data from the two most recent .tdms files. Enabling two-file history can improve time-series graphs significantly. Since .tdms files are automatically restarted every 3 hours, if only one .tdms file is being used for history data, it is possible that a time-series graph might only have 2 minutes of data shown in it even though MICAS-X has run for 3 hours and 2 minutes. Enabling two .tdms file reading ensures that time-series graphs have a reasonable amount of history data available when Channels are changed. Note that if the operator does not change the displayed channels, the local buffer for the time-series graph can grow much deeper than 3 hours. This depth is also controlled by another configuration parameter. Thus if very long time-series graphs are needed, it may be advisable to create a MICAS-X Display that predefines (in the configuration file) which Channel(s) are being graphed, so that the operator cannot change the Channel(s) and the history data can accumulate for a much longer period of time.

It should also be noted that as the number of Driver channels gets large in a MICAS-X configuration, reading history data from even just a three hour .tdms file can incur a performance hit. Thus switching channels on a time-series graph could potentially impact the operation of the program. Again, using time-series graphs with predefined channels can be a way to avoid this situation. Disabling two-file history can also reduce the potential performance hit.

3.9 Commands

Dozens of Commands are built in to MICAS-X, and any Driver can add more custom Commands to the program. The Commands are all handled by a Command module that runs as a top level program in parallel to all the other MICAS-X modules. All Commands are sent to this module, which then calls the required functions of various MICAS-X modules to execute the Commands.

The structure of a MICAS-X command is Command (string) : Parameter (string, optional) : Value (numeric, optional) Thus, some example Commands are:

Record : 1 (turns on recording)

Set : Flow : 22.4 (sets the Channel “Flow” to a value of 22.4)

Commands can be accessed in numerous ways in MICAS-X. There is a row of up to 12 buttons on the top of the main MICAS-X window that can be programmed to each execute a Command. Triggers and Alarms can execute one Command when turning from False to True, and another when turning from True to False. Sequences are made up of Commands, with additional parameters in each Sequence step that allow one to define conditional sequence steps. Finally, the Command Interface Instrument can be used to have MICAS-X receive Commands from another instrument or program.

The Commands allow the operator to define complex interactions, sequences, and conditions within MICAS-X. Often, the entire behavior of a complex system can be programmed in MICAS-X simply by configuring Triggers, Sequences, and other Commands. See Appendix B: Commands for a list of Commands.

3.10 Program Versions

MICAS-X for Windows is available in several versions. It can be supplied as source code and run within the LabVIEW environment, or it can be supplied as a compiled executable and be run without the need for a LabVIEW license. In addition, MICAS-X comes as the normal, full license version, a limited (Ltd) license, a Student Edition, and in a Demo version. MICAS-X-RT, a version that runs on LabVIEW Real-Time embedded platforms, is discussed in the next section, 3.11.

It is useful to purchase the LabVIEW source code version of MICAS-X if you plan to add your own modules to the system. With the source code version, you can see the

source code of approximately 90% of the MICAS-X system. (A small portion of the source code is password protected.) Thus you can use existing Drivers, Instruments, Displays, and Calculations as examples for how to write your own code. In addition, source code enables easier debugging of new MICAS-X modules. One major requirement for using the source code version is that you must also purchase a LabVIEW license from National Instruments.

The compiled, executable version of MICAS-X is useful if you wish to deploy a system on a computer that does not have LabVIEW installed on it. It is often advisable to debug the system first in source code before deploying it this way.

Both the source code and compiled, executable versions can be supplied as the normal, licensed version of MICAS-X. When MICAS-X is licensed, a small USB dongle is supplied by Original Code Consulting which must be attached to the computer running MICAS-X. This dongle verifies the licensing. If the dongle is not detected, the program will periodically switch to the Summary tab and display a message asking the operator to please ensure that the program is properly licensed. Other than switching tabs, no functionality of the program is impacted by the lack of a USB dongle, since it is imperative that experiments and data not be impacted by the possible accidental removal of the USB dongle or its possible failure.

The LTD (or Limited) version of MICAS-X is available in certain situations. It provides a lower-cost entry point than the full version of MICAS-X, but is tailored to a specific experiment or system. Modules that were not sold with a LTD license cannot be later added without assistance from OCC. The LTD version is available as an executable only, and the license key must be present for the LTD version to run.

The Student Edition (SE) is a special version of the LTD license. It is intended as a very low cost entry point for graduate or undergraduate students. It includes Drivers for National Instruments Daq, including analog input, analog output, digital input, and digital output, as well as the features normally included in the MICAS-X base package. Additional modules can be added at a significant discount. MICAS-X SE is available as either an executable or as source code.

The Demo version of MICAS-X is available for free. It does not come with a USB license dongle. It is available only as the compiled, executable version, not as source code. In addition, it is functional only for 20 minutes. After 20 minutes, acquisition and recording will stop and the program will no longer respond to most Commands. If you are using the Demo version, you accept all risk regarding what may happen to your system if it is running when MICAS-X becomes disabled. Note that the Demo version states that it is the Limited Demo version in the window title bar. Also note that if you are using the Demo version and you DO plug in a USB license dongle, the 20-minute time limit will not be imposed, and that version can then be used with full functionality. However, if you have a USB license dongle, it is advised that you obtain the full version

of MICAS-X so that if the dongle is removed or fails, the program will not become disabled.

3.11 MICAS-X-RT

MICAS-X-RT is a version of MICAS-X that has been edited to enable it to run on National Instruments' embedded platforms which run LabVIEW RT. It can be deployed in several ways. If it is delivered as source code, it can be launched directly from the LabVIEW environment and run in interactive mode. In this case, when the run arrow is pressed, it downloads the source code to the RT target and runs it. The front panel is open and available, and one can view and interact with the program directly. This is the only mode of operation in which there is a direct user interface to MICAS-X-RT. Although this mode does provide a limited user interface, from which one can issue commands and set channel values, MICAS-X-RT has been designed as an embedded, headless program. Thus the user interface available in this mode is fairly limited. For similar reasons, the Display and Utility modules of MICAS-X are not supported in MICAS-X-RT, since they are inherently user facing. This interactive mode of deploying MICAS-X-RT is useful for debugging, but is not normally the main mode of operation of MICAS-X-RT.

A second mode of deployment is for OCC to deploy all the relevant code to the RT target and then set the MICAS-X-RT program to run automatically at boot-up. This is the standard mode of operation of MICAS-X-RT. When properly configured, with a configuration file installed on the RT target, MICAS-X-RT can run unattended, keeping a system under continuous control and logging data as needed. MICAS-X on Windows can then connect to MICAS-X-RT whenever needed, to view status and data. In this mode, if new modules or source code needs to be added to the RT system, OCC can assist with the installation of new code, or directions can be given for the customer to use an FTP program to add the new code to the RT system.

A third mode of operation is very similar to the second, but allows for more flexibility. In this mode, MICAS-X-RT (and optionally MICAS-X) are present on a Windows computer, along with the LabVIEW environment. MICAS-X-RT is normally installed directly on the RT target as in the 2nd mode, but the MICAS-X-RT Project on the Windows computer allows for an easier mechanism for deploying new modules and code to the RT system when needed.

For all three options, it is possible and easy to use the MICAS-X program on a Window platform to provide a user interface to MICAS-X-RT. When the MICAS-XRT Driver is included in a MICAS-X configuration, all the channels and commands on the RT system become directly accessible in the MICAS-X Windows system. This allows for quick review of the RT system status and of channel values, and also allows for complete direct control of the RT system. The MICAS-X program on Windows can be stopped and restarted whenever desired without interrupting the MICAS-X-RT program (as long as the MICAS-X-RT configuration file defined in the MICAS-XRT Driver is not altered.) (Note

that MICAS-X-RT is the version of MICAS-X that runs under LabVIEW RT on a NI RIO system. The MICAS-XRT Driver is a module the runs under MICAS-X on Windows and communicates with MICAS-X-RT.)

In the various sections of this document, it is noted when a module is known to work only in MICAS-X-RT or MICAS-X for Windows. If not otherwise stated, a module may work in MICAS-X-RT, but may require additional testing and development.

3.12 Controlling one MICAS-X from another MICAS-X

Several mechaninism are available that allow multiple MICAS-X programs to communicate with each other. Selection of the appropriate communication mechanism depends on the requirements of the situation. Several options are described here that could help in many situations.

One simple way to share data between instances of MICAS-X is to use the Broadcast Instrument (see Broadcast.vit)7.7.1 and the MICM Driver (MICM.vit) (or the older MOSDS Driver (MOSDS.vit)). The Broadcast Instrument can send a selected Channel List to out a serial port or Ethernet port at a specific rate, such as once a second. On the second MICAS-X system, one can create an MICM configuration file that can parse that Channel List. In this way, current values of a selection of channels from the first MICAS-X system can be readily available on the second MICAS-X system. This mechanism works between Windows systems and RT systems. Note that whenever the Channel List being broadcast is edited, the corresponding MICM configuration must be updated to allow the receiving program to parse the data correctly.

The Command Interface Utility (see Command Interface) is often used in conjunction with the Broadcast Instrument described above. Where-as Broadcast shares data from the first MICAS-X with the second, the Command Interface Utility allows the second MICAS-X to send commands to the first. To use the Command Interface Utility, one must include the Command Instrument in the configuration of the MICAS-X program to which Commands are being sent, e.g. the server MICAS-X. In addition, the server MICAS-X configuration file must be available on the computer on which the Command Utility is being run, e.g. the client MICAS-X. By having a copy of the server's configuration file, the client Command Utility knows all the Channels, Sequences, Triggers, etc., that the server has available, and thus can format and send Commands that are specific to the server system. As with the Broadcast Instrument described above, it is necessary to keep the server's configuration file consistent between the server and client systems. E.g. if the server system configuration changes, one should copy that new configuration file to the client system. Also, note that since the Command Interface is a Utility, it cannot be configured to run automatically within MICAS-X. Instead, when you start the client MICAS-X, you must select the Command Interface from the Utility menu. Alternatively, one can run the Command Interface Utility as a separate program. The default Command Interface Utility has some configurable options in its user interface to

allow one to set up commonly-used Commands that can be easily executed. In addition, a more general, though more tedious, set of controls allows one to specify any available Command to issue to the server program. It should be noted that custom versions of the Command Interface Utility have been created for specific project, so that the user interface can support a variety of controls that are more relevant and familiar for that particular use of MICAS-X.

MICAS-X-RT (see MICAS-X-RT Operation), the version of MICAS-X that runs on NI's real-time platforms, is designed to seamlessly integrate with MICAS-X on Windows. The MICAS-X-RT program itself contains the communications links to support this functionality, so no additional modules are needed on the RT (server) program. On the Windows (client) MICAS-X, one must include the MICAS-XRT Driver (MICAS-X-RT.vit). As with the Command Interface Utility above, the MICAS-XRT Driver must be given the full configuration file of the MICAS-X-RT system, and if the MICAS-X-RT configuration is edited, one must remember to update the copy of that configuration file that the client (Windows) MICAS-X is using. With this interface, the full set of Channels on the MICAS-X-RT system are present in the Windows client system, with a "XRT_" prefix. In addition, all the Commands that the MICAS-X-RT system supports are available on the Windows client system as well, again with an "XRT_" prefix. This allows the operator to watch data on the RT system in real time from the Windows MICAS-X, and to issue Commands, start Sequences, etc., on the RT system directly from the Windows MICAS-X.

A mechanism similar to the MICAS-XRT Driver exists for linking two MICAS-X instances which are both running on Windows: M-Link. For the M-Link system, both the M-Link Server Instrument and the M-Link Driver must have the same prefix (even though these modules are on different computers). A blank prefix is a valid option. If more than one M-Link connection is being made, the modules must have unique prefixes, so that each M-Link Server and M-Link Driver have matching prefixes that are unique from all other M-Link Servers and Drivers. As with MICAS-XRT, the M-Link Driver must be configured to know the configuration file used by the instance of MICAS-X running the M-Link Server, as well as the IP address of the computer running the M-Link Server. With this information, the instance of MICAS-X running the M-Link Driver can access all the Channels of data, Commands, Sequences, etc., that are running in the instance with M-Link Server. See the sections on the M-Link Driver 7.6.9.2 and M-Link Server 7.7.4 for more information.

The Drivers (M-Link Driver 7.6.9.2 and MICAS-XRT Driver 7.6.14) for the M-Link and MICAS-X-RT communications links create a "connection" channel that indicates the health of the network connection between the two systems. For each communication link, there are three connections happening behind the scenes: one for Commands to the server from the client, one for Messages from the server to the client, and one to pass current data values from the server to the client. Each of these three connections has a value of 0 when it is not connected and a value of 1, 2, or 4, respectively, when

they are connected. Thus when the connection channel has a value of 7, all three connections have been made successfully and the link is working properly. A connection value of 5, for example, would mean that the Message connection has not been made successfully.

3.13 MICAS Metrics Program

MICAS-X includes an opt-in automatic feedback feature that sends limited metrics data to Original Code Consulting. This program, called MICAS Metrics, can be enabled or disabled in the full or LTD versions of MICAS-X from the Help menu. Note that for the Demo and SE versions, this program is turned on by default.

The purpose of the MICAS Metrics Program is to improve MICAS-X for all customers, by allowing OCC access to information on how MICAS-X is being used and what improvements are needed. If the Metrics option is turned on, information about the MICAS-X serial number, configuration, usage, and errors encountered with MICAS-X will be transmitted to OCC. No data that is acquired with MICAS-X will be sent, and all data transmitted by this mechanism will be held in strict confidence by OCC.

4 Notes on Installation

Hardware requirements for installation of MICAS-X are the same as those for installing LabVIEW. MICAS-X is currently supported for LabVIEW 2018, 32bit, running on Windows 7 and 10, 32 or 64 bit. It is generally compatible with Windows XP, Vista, and 8, but is no longer supported on those operating systems. As of Version 3.3, MICAS-X is compatible with 64 bit LabVIEW as well as 32bit LabVIEW.

Refer to the MICAS-X Installation Manual document for detailed instructions on installing MICAS-X either as LabVIEW source code to be used inside the LabVIEW environment, or as an executable stand-alone program. Also refer to Chapter 12MICAS-X Security Device for information about the USB license key.

In addition to installing MICAS-X and the USB security device, there are several other issues that should be considered when setting up your computer to run MICAS-X. If your computer is not connected to the internet, it may be worth considering disabling or uninstalling anti-virus software. Some anti-virus packages are known to interfere with data acquisition processes, though others have been used without any observed issues. You might also consider turning off Windows Auto-Updates, but do this at your own risk. If Auto-Updates are left on, it is important to configure your computer so that it does not automatically reboot after installing an update. A google search for terms such as “no reboot after automatic update” should return some useful links with instructions for disabling this annoying option, though Windows 10 has made it much more difficult to disable.

It is also very important to configure your computer's power options. It is recommended that you set the power scheme to never turn off your computer and never turn off the hard disk if the computer is plugged in (not on battery power). Having a data acquisition computer go to sleep when it is supposed to be taking data does no one any good. It is also recommended that you turn off screen savers, though no obvious issue arising from their use have been observed. In fact, MICAS-X includes code that automatically disables screen savers when it is running.

Numerous steps are required to configure MICAS-X-RT for a specific Real-Time target. This initial configuration will normally be done by Original Code Consulting. If appropriate, OCC will provide additional instructions for deploying future code modifications.

5 Locating Files and Directories on a Hard Drive

Various parts of the MICAS-X system are stored in different places on your hard disk. If you are using the executable version of MICAS-X, read the section immediately below. If not skip to the section on source code.

5.1 Directories Used with the Executable Version of MICAS-X

Under C:\, go to the folder labeled “OCC.” Inside are the following folders:

- MICAS-X Data – Where data collected by MICAS-X is stored by default. Folders within this folder will be titled in yyyymmdd format.
- MICAS-X Support – The configuration file is found in this folder. A text file named “Config to Load.txt” informs the program which file to use as the configuration file the next time the program is run. Both the text document and configuration file are in the first level of this folder. The configuration file has a .ini extension. If the file C:\OCC\MICAS-X Support\Config to Load.txt does not exist, then the MICAS-X program will automatically look for a configuration file named C:\OCC\MICAS-X Support\MICAS-X Configuration.ini

This MICAS-X Support folder also contains all or most of the following folders:

- Defaults
- Displays
- Drivers
- Documents
- Documentation (this folder is deprecated in favor of the Documents folder)
- Editors
- Resources
- Scripts

- Utilities
  These folders contain various resources and source code that the program uses.

Executable files for MICAS-X are located in C:\Program Files\MICAS-X, or on 64 bit operating systems, in C:\Program Files (x86)\MICAS-X. This fould should contain at least the following:

- MICAS-X Configuration Editor.exe
- MICAS-X Data Reader.exe
- MICAS-X Log Reader.exe
- MICAS-X mdoc Editor.exe
- MICAS-X TDMS File Reader.exe
- MICAS-X.exe

5.2 Directories Used with MICAS-X in Source Code

Under C:\, go to the folder labeled "OCC." Inside are the following folders:

- MICAS-X Data – Where data collected by MICAS-X is stored by default. Folders within this folder will be titled in yyyymmdd format.
- MICAS-X Support – The configuration file is found in this folder. A text file named "Config to Load.txt" informs the program which file to used as the configuration file the next time the program is run. Both the text document and configuration file are in the first level of this folder. The configuration file has a .ini extension. If the file C:\OCC\MICAS-X Support\Config to Load.txt does not exist, then the MICAS-X program will automatically look for a configuration file named C:\OCC\MICAS-X Support\MICAS-X Configuration.ini

Under C:\, go to the folder labeled "MICAS-X." Inside are most or all of the following folders:

- Acquisition
- Calculations
- Channel Lists
- Command Loop
- Common
- Configuration
- Defaults
- Development Documentation
- Displays
- Documents
- Drivers
- Editors
- File Writer

- FPGA Bitfiles (optional)
- Instruments
- KeyLok
- Log File Component
- Program
- Project Files
- Reporting
- Resources
- Sequencer
- Scripts
- Test
- Triggers
- Utilities

Other files, including .alias and .lvlps files and LabVIEW Project files may also be present in the aforementioned folders.

5.3 Directories Used with MICAS-X-RT

LabVIEW RT platforms use several different Operating Systems, depending on the particular model of hardware. Within MICAS-X, paths are built up with the "Build Path" function from individual folder names, thus allowing LabVIEW on each platform to create the correct path syntax. For this document, directories will be referred to using standard Windows path syntax, even though this will not be exactly correct on all RT platforms.

The main MICAS-X-RT program will normally be named startup.rtexe and stored in the RT platform's default startup location, which usually looks something like "home\lvuser\natinst\bin".

Data is stored in "C:\OCC\MICAS-X Data". Support files are stored in "C:\OCC\MICAS-X Support". The folder structure within the MICAS-X Support folder on RT systems is identical to that described above for executable versions of MICAS-X.

5.4 File Types Used with MICAS-X

The following file types are used with the MICAS-X program.

File Extention	**File Description**
.vi, .vit, .ctl, .lvproj, .lvlps, .aliases, etc .	The LabVIEW programming environment uses numerous file extensions. The list to the left contains the most common.
*.dig	Several Digitizers are supported as MICAS-X Instruments. Digitizer data can be stored to .dig files, and the dig File Reader Utility can be used to

	review them.
.exe	Executable versions (which do not run inside LabVIEW) of MICAS-X and its utilities have the .exe extension.
.ini	Configuration files for MICAS-X have the .ini extention. Executable (.exe) versions of MICAS-X and its utilities also have .ini files that determine how they run.
.txt	The Config to Load.txt file, which determines which configuration will be used when MICAS-X is run, has this extension.
.mhlp	Run-Time help files for MICAS-X use the .mhlp extension.
.mdoc	This extension is used by text documentation files used by the MICAS-X Document Driver.
.mseq	The Sequence Editor in MICAS-X allows for Sequences to be imported and exported to text documents using the .mseq extension.
.mscr	MICAS-X Scripts are saved in .mscr files. They are not contained directly in the MICAS-X configuration files.
.osds	OSDS format files, used by the MOSDS Driver, have this extension. OSDS data files, stored by numerous Drivers, also use this extension.

6 Configuration File Usage

MICAS-X can take on a wide range of personalities or functionality, as determined by the configuration file that is used. A single user or system can create multiple configuration files, and select which one is run depending on the experiment or process being carried out.

The Configuration Editor (run either as a Utility inside MICAS-X or as a stand-alone program in Windows or a VI in LabVIEW) can be used to define the start-up configuration file. Pressing the button "Mark as Start-up File" will write the name of the configuration file currently being edited to the “Config to Load.txt” file, causing it to be used the next time MICAS-X is started. In addition to using the “Config to Load.txt” file to determine which configuration MICAS-X uses when it starts, it is also possible to create shortcuts to MICAS-X which tell the program to load a specific configuration, and which thus over-ride the “Config to Load.txt” setting. The directions for creating shortcuts differ depending on whether MICAS-X is being run as an executable or in source code.

Note that the configuration file name “MICAS-X RunTime Configuration.ini” is a reserved name and should not be used to name a user-defined configuration file. This file is used by MICAS-X to track dynamic configuration parameters, which can be changed, saved, and used while MICAS-X is running. (Normal configuration information is only read when MICAS-X starts, so changing those configuration parameters does not alter how MICAS-X operates while it is running, but only when it next starts.) Examples of dynamic configuration parameters include the location of the MICAS-X window on your computer desktop. As the user moves MICAS-X around, the program remembers where it has been placed by writing that location to the “MICAS-X RunTime Configuration.ini” so that it will be placed in that same position the next time it is run. Another difference between normal configuration parameters and those in the “MICAS-X RunTime Configuration.ini” are that the latter are used with any and all MICAS-X configurations, and hence they are more global parameters that define how MICAS-X acts in general, but not how any one configuration acts.

As of version 1.5.0 of MICAS-X, a CRC check is placed in the configuration files by the configuration editor. This check can be used to identify configuration files that have become corrupt or were edited by hand outside of the Configuration Editor, as explained in section 7.2. It is also possible to configure a Backup Configuration file which will be automatically used if the Start-up configuration file is found to have an invalid CRC. The Backup Configuration file is described in section 7.3. This feature cannot be used when a shortcut is used to select a configuration file, as described above. It only works when the “Config to Load.txt” file is used to select the configuration file.

6.1 Shortcuts for MICAS-X.exe Executable

To create a shortcut for the MICAS-X.exe program that loads a specific configuration file, use Windows Explorer to navigate to the MICAS-X.exe file, which should be located in C:\Program Files\MICAS-X or C:\Program Files (x86)\MICAS-X. Right click on the file and select "Create Shortcut". Drag this shortcut to the desktop or to some other desired location. Right click on the shortcut and select "Properties". The Target parameter must be edited according to the following format:

"C:\Program Files\MICAS-X\MICAS-X.exe" "Configuration File.ini"

E.g. the first parameter must be the full path to the MICAS-X.exe program, in quotes. The second parameter must be the name of the start-up configuration file, in quotes. Note that for the second parameter, only the name of the configuration file should be used, not the full path. The configuration files must be located in C:\OCC\MICAS-X Support.

6.2 Shortcuts for MICAS-X.vi in Source Code

To create a shortcut for the MICAS-X.vi VI and have it run inside LabVIEW and load a specific configuration file, use Windows Explorer to navigate to the main MICAS-X VI. This should be located at C:\MICAS-X\MICAS-X.vi. Right-click on the MICAS-X.vi and select "Create Shortcut". Then drag the resulting shortcut to the desktop or some other desired location. Right click on the shortcut and select "Properties". The Target parameter must be edited according to the following format:

"C:\Program Files\National Instruments\LabVIEW 2015\LabVIEW.exe" "C:\MICAS-X\MICAS-X.vi" -- "Configuration File.ini"

E.g. the first parameter must be the full path to LabVIEW, in quotes. The second parameter must be the full path to the MICAS-X.vi in quotes. The third parameter must start with two dashes, followed by a space, followed by the name of the desired configuration file, in quotes. For this shortcut to work, LabVIEW must not be started or loaded before the shortcut is run. (E.g. if LabVIEW is running, it must be exited first.) In addition, the "Run When Opened" option must be selected for the MICAS-X.vi in the Execution options tab of the VI properties. Note that for the third parameter, only the name of the configuration file should be used, not the full path. The configuration files must be located in C:\OCC\MICAS-X Support.

6.3 Configuration Files for MICAS-X-RT

MICAS-X-RT uses three configuration files. All three are stored in the C:\OCC\MICAS-X Support directory on the RT machine. The file named MICAS-XRT.ini is the current, active configuration. This may be manually loaded onto the RT machine by file transfer, or it can be sent down to the RT machine when the MICAS-XRT Driver is run on a host machine. When it is sent down from the host, the MICAS-XRT Driver first reads the XRT configuration file on the host machine, then sends just the contents of that file as a string to the RT machine. Upon receipt, MICAS-X-RT saves the contents in the file MICAS-XRT.ini, along with an additional entry that stores the name of the original configuration file. (Note that if the configuration file is manually transferred to the RT machine, this additional entry will not be present. The only use for this entry is to display the original name of the configuration file on MICAS-X-RT when it is run in interactive mode.)

If the Save As Auto-Load parameter in the Program Parameters is true, the configuration file is saved as Auto-Load.ini on the RT machine. Whenever this configuration file is present as MICAS-X-RT starts running, it is loaded automatically. E.g. no interaction from the host is needed to configure and acquire data with MICAX-X-RT in this case.

The third configuration file is named Restart.ini. MICAS-X-RT uses this as a temporary configuration file whenever restarting itself to load a new configuration. As soon as this file is read in as the current configuration, it is deleted.

7 Configuration Editor

Before using MICAS-X, you must have or create a Configuration File that defines what features MICAS-X will have. This section explains how to use the Configuration Editor to create your configuration files. Proceed to the "Using MICAS-X" section of the manual if you already have configuration files and are interested in using MICAS-X.

To launch the configuration editor from within MICAS-X, go to the Utilities Menu and select "Configuration Editor". The Configuration Editor.vi VI can also be launched within LabVIEW, or the Configuration Editor.exe program can be run directly from Windows.

As with all Utilities, the Configuration Editor only runs on Windows, not in MICAS-X-RT. It is used to edit MICAS-X-RT configuration files, however.

Note that most MICAS-X Utilities do not have any related configuration parameters. However, as of version 3.2, some Utilities do have configurable parameters. If any Utility is installed on your system which supports configuration parameters, then a Utilities selection will appear at the bottom of the Components list. When Utilities is selected, the configurable Utilities will appear in the list below.

7.1 Using the Configuration Editor

In the top-left area of the window, select the "Read a File" button to select a file to open in the configuration editor. This will present a file dialog for the folder "MICAS-X Support", from which you can pick an existing configuration file to edit.

To the right of the "Read a File" button, two indicators display the file path of the file being edited and the start-up file (the configuration file that will run the next time MICAS-X is run). Next to each of these boxes are options to "Mark as Start-up File" (which saves the current configuration as the start-up configuration) and "Read Start-up File" (which reads the current start-up file given in the box to the left into the Configuration Editor). The "Save As" button allows the operator to save the currently edited configuration under a new file name. The "New" button will create a clean configuration with only default values. When this button is pressed, a file dialog will be presented so that the operator can name the new configuration file.

Note that the Start-up File refers to MICAS-X, and is not used for MICAS-X-RT. To define a start-up configuration file for MICAS-X-RT, one can use a file transfer program to put a configuration file named “Auto-Load.ini” in the “C:\OCC\MICAS-X Support” folder on the RT computer. Or one can use the “Save as Auto-Load” option in the MICAS-XRT Driver to have MICAS-X send a configuration file down to the RT system and have it saved to be used automatically each time the RT system is booted.

The “Version” box lists the version of the MICAS-X Configuration Editor currently being used. Finally, the “STOP” button will immediately stop and close the configuration editor window. Make sure to save all changes before pushing “STOP.”

On the left-hand side of the screen are three lists of options. The first, labeled “Target”, allows you to define which type of configuration file you are editing. When “Windows” is selected, the configuration file is intended to run in MICAS-X on Windows. When “Real-Time” is selected, the configuration file is intended for MICAS-X-RT on a LabVIEW Real-Time platform. Although either program can attempt to load a configuration file with either Target specified, an error message is generated if the Target does not match the platform being used. In addition, when the proper Target is selected, certain parameters are presented that are relevant to that Target and which may not work correctly on the other Target. E.g. if a Windows configuration file is loaded onto a Real-Time system, it is likely that the system will not operate correctly, as not all the parameter will have been set appropriately for the RT system.

If the Target parameter is changed, it is important to force a save in the Modules/MICAS windows to make sure that the Target selection is remembered. For example, change the Update Time (ms) parameter by a value of 1, then back to its former value. The Save button will then flash and become active, allowing you to save the Target selection.

The second list, labeled “Components,” allows you to select which part of the MICAS-X program you wish to edit. Your selection of either Modules, Instruments, Drivers, Displays, or Tools will change the contents of the second list. The third list will allow you to select the specific module you wish to independently configure. Be sure to save all your changes in one component editor before clicking to select another component.

When a component is selected in the list to the left, an editor window specific to that component is loaded on the right side of the window. A series of buttons near the upper right allows the operator to save changes to the configuration. The right most button is named “Save” and will flash red if any changes have been made that have not yet been saved. If another component is chosen without first pressing this save button,

any changes made will be lost. To the left of the Save button is a Revert button. Pressing this button will revert all the configuration parameters in the current window back to the values contained in the configuration file being edited. E.g. any changes that the operator has entered will be removed. Further to the left is a button labeled “Reset to Default”. If this button is pressed, all the displayed configuration parameters will be initialized to their default values. Those default values will not be saved to the configuration file until the Save button is pressed. Finally, if the configuration editor is being run inside MICAS-X, and if MICAS-X is in Debug mode, a fourth button appears even further to the left. This button is labeled “Save as Defaults”. When this button is pressed, the current values of the configuration parameters are saved to a special configuration file that defines what the default values for the component should be. This button is not intended for use in normal operation.

7.2 Configuration File CRC

A Cyclic Redundancy Check (CRC) was added to MICAS-X configuration files in version 1.5.0. This value serves as a way to ensure the integrity of the configuration file. The presence of the CRC allows the program to detect when a configuration file has become corrupted, or when a configuration file was edited by hand (rather than by using the MICAS-X Configuration Editor). Since manual editing of a configuration file is error prone, being able to detect when someone has edited a configuration file can be a valuable form or insurance.

Any configuration files created with version 1.5.0 or later of MICAX-X should have the “Require CRC” option (on the Security configuration page) set to True by default. Configuration files created with older versions of MICAS-X will not have this parameter, so when they are read in, it will default to False. The CRC will only be examined for configuration files for which this parameter is True. Thus, if you have a configuration file that was created in an older version of MICAS-X, and wish to apply this level of protection in version 1.5.0 or later, you should read the configuration file into the MICAS-X Configuration Editor and re-save it. This will apply a valid CRC to the configuration file.

If MICAS-X is run with the “Require CRC” parameter True, and the CRC in the configuration file is valid, no action will be taken. If the CRC does not match the contents of the configuration file, or if it is missing, error 7033 will be generated. Note that even when error 7033 is generated, MICAS-X will continue to run. If you are concerned about the integrity of your configuration file and want to know if this error is present, it is recommended that you set up an Alarm that will post an Alert for error 7033. This Alarm could also be configured to turn off acquisition or even exit MICAS-X if so desired. See section 7.5.13 for information on Triggers and Alarms.

The “Require CRC in Config File” parameter is only available in MICAS-X for Windows, not MICAS-X-RT.

7.3 Backup Configuration File

An additional safety mechanism that works in conjunction with the Configuration File CRC is the Backup Configuration File. This is defined in the Security page under the “Modules” section of the configuration editor. When a Backup Configuration File is defined, its name is stored in a file named “Backup Config.txt”. This file and the Backup Configuration File itself must both be located in the MICAS-X support directory.

If a Backup Configuration File is defined, and the CRC of the Startup Configuration File is found to be invalid, the file name in the “Backup Config.txt” will be automatically copied to the “Config to Load.txt” file, and that configuration file will then be used. Error 7034 will be reported, so that the operator is aware of the configuration file substitution. This leaves the original Startup Configuration File intact, so that it can later be examined to determine why the CRC check failed. Also note that if the “Config to Load.txt” and “Backup Config.txt” files contain the same file name, no configuration file substitution is performed.

Care should be taken when managing the Backup Configuration File. If the Backup Configuration File is not maintained and properly updated, then if the Startup Configuration File is unusable, MICAS-X will start with an out-of-dated and possibly incorrect Backup Configuration File.

The Backup Configuration File is only available in MICAS-X for Windows, not MICAS-X-RT.

7.4 Modules

7.4.1 MICAS

7.4.1.1 MICAS-X for Windows

The MICAS configuration page allows the configuration of much of the main functionality of the MICAX-X program. On this configuration page one selects which Displays and Instruments are included in the MICAS-X system, as well as how the options above the tab structure in MICAS-X are configured.

In the top left-hand corner of the window, enter the name you wish to call the system you are editing under the “System name” box. This name will be used to label the first tab of the MICAS-X display, which is otherwise named “Control”.

You can determine where the collected data is stored using the “Data Folder” parameter. This is set to “C:\OCC\MICAS-X Data” by default.

The “Acquire on Start Up” option allows you to turn on and off the ability to immediately acquire data when the program first starts.

The “Record on Start Up” option allows you to turn on and off the ability to immediately record data upon start-up of the program.

The “Max History Depth” option allows you to select the maximum size of the data buffer of data displayed on time-series graphs. The buffer size of data displayed in time series graphs depends on several parameters and conditions. See section 9.5 on the time-series graphs.

Below the “Max History Depth” option is a switch labeled “Read 2 files of History data” or “Only read 1 file of History data.” This parameter affects how much history data is available in time-series graphs, as described in section 9.5 on the time-series graphs.

The “Displays” section of the MICAS Module defines which Display modules are included in the current configuration. The “Available Displays” list to the far right shows all the Displays that can be added to the current configuration. Note that some Displays can be added more than once (those with the .vit extension), whereas others can only be added once (those with the .vi extension). Thus if a .vi Display is already added to the current configuration, it will no longer be listed in the Available Displays list.

To add a Display to the current configuration, double-click on the desired Display in the Available Displays list. The Displays present in the current configuration are listed in the “Displays” list further to the left. To remove a Display from the current configuration, press the red button on its row. Note that if more than 5 Displays are present in the current configuration, the scroll bar on the “Displays” list can be used to scroll up or down through the list of Displays.

Create a prefix for the Display by typing in the desired prefix into “Display Prefixes” in the appropriate row. Prefixes cannot contain spaces and should be kept short. Be sure to use prefixes if a Display module is being used more than once (e.g. a .vit Display). The prefix allows MICAS-X to distinguish between the multiple instances of the Display and keep track of which is which. The Display name is automatically used as the “Tab Name”, but this can be changed according to your preferences to give the Display tab in MICAS-X a specific label.

If the “Outside?” check box is marked, the Display will be instantiated outside of the main MICAS-X window, and will appear as its own window rather than in a MICAS-X tab.

The “Instruments” section of the MICAS Module defines which Instrument modules are included in the current configuration. The “Available Instruments” list to the far right shows all the Instruments that can be added to the current configuration. Note that some Instruments can be added more than once (those with the .vit extension), whereas others can only be added once (those with the .vi extension). Thus if a .vi Instruments is already added to the current configuration, it will no longer be listed in the Available Instruments list.

To add an Instrument to the current configuration, double-click on the desired Instrument in the Available Instruments list. The Instruments present in the current configuration are listed in the “Instruments” list further to the left. To remove an Instrument from the current configuration, press the red button on its row. Note that if more than 5 Instruments are present in the current configuration, the scroll bar on the “Instruments” list can be used to scroll up or down through the list of Instruments.

Create a prefix for the Instruments by typing in the desired prefix into “Instrument Prefixes” in the appropriate row. Prefixes cannot contain spaces and should be kept short. Be sure to use prefixes if an Instrument module is being used more than once (e.g. a .vit Instrument). The prefix allows MICAS-X to distinguish between the multiple instances of the Instrument and keep track of which is which. The Instrument name is automatically used as the “Tab Name”, but this can be changed according to your preferences to give the Instrument tab in MICAS-X a specific label.

Each Instrument can optionally have a Display of its own. The “Display?” check determines whether MICAS-X will open the Instrument VI's front panel as a Display. If the “Outside?” check box is marked, the Instrument Display will be instantiated outside of the main MICAS-X window, and will appear as its own window rather than in a MICAS-X tab.

The section in the lower left-hand part of the window allows you to configure up to twelve options that can appear along the top of MICAS-X. (Note that the 8th option will only be available if there are no Alarms configured, since the Alarm Status indicator appears in the same space as the 8th button. The 9th through 12th options appear on a second line, in line with the tab buttons. These options should only be used if few enough tabs are defined in MICAS-X to allow free space where these options appear.) To configure an option, select the number of the option you wish to edit using the arrows next to the “Option Number” control. Select the type of Option using the Option drop-down menu. Use “None” to have the program ignore that option. Other options that can be select are Command, Indicator, Controller, T/F Indicator, Switch (T/F Controller), Enum List Indicator, and Enum List Controller. Note that Indicators, Controllers, Enum

List Indicators, and Enum List Controllers take up the space of two options, since they require space for their channel label. Thus these options are available only for option numbers 2, 4, 6, 8, 10 and 12. This is important to keep in mind when editing options 1, 3, 5, 7, 9, and 11. For example, if option 2 is set to Controller, the option 1 will be forced to be “none”. In this case, if you change option 1 to another type of option, it will automatically revert to “none”.

The Command option creates a button which can issue any Command when it is Turned On and any other Command when it is turned Off. The left column of parameters determine how the button will act when it is turned from False to True, and the right column determines what will happen when it is turned from True to False.

Check the “Auto-Reset” button if the button should automatically switch back from True to False after the program reads it. As an example, a button that controls whether the program is in Acquire mode would not use auto-reset, as the button would stay True or False until the operator changes it. A button that starts a Sequence could use the Auto-Reset function, since once the Sequence is started, the button would automatically revert back to False so that the operator could press it again when needed.

“False Text” shows the text that will be shown when the button hasn’t been pushed, and “True Text” shows the text that will be shown when the button has been pushed. In these fields, add a short text string that indicates to the operator what the button will do. The remaining parameters are used to define the behavior of the button. Use these parameters to assign MICAS-X Commands to the False-to-True button push and the True-to-False button push. When a Command is selected, the string and value parameters below the Command are enabled or disabled depending on the parameters that the Command uses.

Indicator options display the current value of any one channel. Controller options present the value of one controller (output) channel and allow the operator to change it.

For both T/F Indicators and Switches, the associated channel name is used as the button label. T/F Indicator options are used to display the value of a channel when that channel naturally can have only two values. Normally a value of 0 is equated with False, and 1 (or any non-zero value) is equated with True. However, the “False Value” option can be used to explicitly specify the value that will be displayed as False. A Switch option (T/F Controller) is similar and is used when a controller (output) channel naturally can take on only two values. The “False Value” and “True Value” parameters specify the numeric value assigned to the channel whenever it is set to False or True. If another

value (besides the False or True Values) is assigned to the channel by another process within MICAS-X, the channel name displayed on the button is bracketed with asterisks to indicate that the channel currently has an out of range value.

Enumerated Lists are used to assign a set of text labels to a finite number of channel values. For example, a “System State” channel might use the value 0 for Off, 1 for Stand-by, 2 for Running, and -1 for Error. Enum List Indicators and Enum List Controllers allow one to define enumerated lists for displaying the value of a channel (Indicators) or setting the value of a channel (Controllers). See Section 7.4.12 for information on configuring Enumerated Lists.

The “Require CRC” parameter determines whether or not the program will generate an error if it finds that the configuration file CRC (Cyclic Redundancy Check) value is invalid or is missing. This feature allows MICAS-X to detect when the configuration file has been corrupted accidentally or has been edited by hand, rather than with the MICAS-X Configuration Editor. Configuration files created with versions of MICAS-X earlier than 1.5.0 do not contain this feature, so this parameter should be set to False for such files. It is recommended that this parameter be set to True for configuration files created or edited with versions of 1.5.0 or later. See section 7.2 for more information.

You can add notes about the configuration in the “Notes” section at the bottom of the window. These notes are for your own documentation purposes, to remind you of how you configured things and why. They are not used within MICAS-X and are only saved within the configuration file. Each configuration window for each module contains a Notes field that can be used to record notes about that module's configuration.

Additional configuration options that are general to the entire MICAS-X program can be found on the Advanced configuration. Some of the options on the Advanced window were previously on the MICAS configuration.

7.4.1.2 MICAS-X-RT

The MICAS configuration page for MICAS-X-RT has fewer options than that for Windows, since there are many fewer user-interface options on the RT platform. The “Save as Auto-Load?” option, if checked, will ensure that when MICAS-X-RT next uses the configuration file, it will save it with the name “Auto-Load.ini”, and that configuration file will thereafter be used automatically to start and configure MICAS-X-RT whenever the RT system is booted up.

The Acquire on Start-up and Record on Start-up parameters work as described above in the Windows section, as does the Notes parameter.

The Enable RT Watchdog parameter adds another level of robustness to the MICAS-X-RT program. On NI RT platforms, a hardware-based Watchdog is usually included. When enabled, this hardware circuit is ready to reboot the RT computer if the software fails to check in on a regular interval. MICAS-X-RT is designed to refresh the Watchdog once a second, with a two second expiration time. Thus if MICAS-X locks up and fails to refresh the Watchdog for over two seconds, then if this parameter is enabled, the RT system will be rebooted.

The Instruments parameters also function as described in the section above (7. 4. 1. 1), except that the Instruments available are those in the “Instruments RTX” folder. This folder only contains MICAS-X Instruments that have been designed and verified to be compatible with MICAS-X-RT.

7.4.2 Acquisition

The Acquisition Module allows the user to define which Drivers to use within MICAS-X. Driver are shown in the drop-down lists in the Drivers array. Use this list to add Drivers being used in the current configuration. Note that Drivers that end with “.vit”can be used more than once in a configuration. If a Driver ends with .vi and not .vit and you attempt to add it to the Drivers list more than once, the program will remove the new instance from the Drivers list.

If a specific Driver is only used once in the MICAS-X configuration, the Prefix is optional. If the same Driver is used more than once in a configuration, each instance must have a unique Prefix. (One such Driver can still have a blank Prefix, as long as no other instances of the same Driver also have a blank Prefix.) Prefixes should be short and cannot contain spaces. Create a prefix for the Driver by typing in the desired prefix into “Driver Prefixes” in the appropriate row. In addition to allowing MICAS-X to uniquely identify each instance of a Driver, the Prefix is prepended to each Channel of the Driver, so that the Channels from multiple instances of the same Driver are uniquely named.

The “Acq Loop” array to the right of the Prefixes allows you to change the Acquisition Loop number assigned to the Driver. Assigning drivers to different acquisition loops allows asynchronous acquisition based on multiple different acquisition rates. Utilizing multiple loops can be helpful in several situations. If one Driver is reading data from a streaming device (e.g. one that continuously sends data at a fixed rate, without being asked) at, for example 10 Hz, and another Driver reads data from another device at 1 Hz, putting the two Drivers in different Acquisition Loops allows for each device to be read at the appropriate rate. Another example is when one Driver is reading data from a streaming device at 1 Hz, and another Driver is controlling a data acquisition card at 1 Hz. The first Driver's 1 Hz operation will be tied to the external device's 1 Hz clock, whereas the second Driver will be tied to the computer clock for timing the data acquisition device. These two clocks will inevitably drift relative to each other. Although

they are both operating at close to 1 Hz, putting them in the same Acquisition Loop will eventually cause an error when the two devices become out of synchronization.

You can delete any driver by pressing the red delete button in the appropriate row.

Below “Drivers” are options to change cycle times. This control determines the timing of each Acquisition Loop. This array will have a number of elements determined by how many Acquisition Loops are defined in the Drivers configuration above it. If the Device Timed checkbox next to a Cycle Time is not checked, then the loop timing is determined by the MICAS-X software, and the Cycle Time is milliseconds per loop. If the Device Timed checkbox is checked, the Cycle Time is a millisecond delay, as discussed below.

If the Device Timed checkbox next to the Cycle Time is marked, the specified loop cycle time will time will be determined by the device assigned to it, rather than by software timing by the MICAS-X program. Software timed Acquisition Loops should be used for devices and Drivers which respond to a request for data. E.g. if one is using the NIDaqDI Driver to acquire digital input data from an NI device, this Driver should be assigned to an Acquisition Loop that is software timed. If the Cycle Time is set to 1000, MICAS-X will ask for data from this Driver every 1000 ms. However, if a Driver is being used such as GPS, which streams data once a second without needing to have MICAS-X ask for the data, then the Device Timed check box should be marked. In this case, the Acquisition Loop cycle time will be determined by the rate at which the device sends data. Note that when the Device Timed check box is marked, the Cycle Time parameter is actually used as a Delay Time in milliseconds. Thus, for the example of a GPS sending data once every 1000 milliseconds, a Delay Time of 500 milliseconds may be appropriate. In this case, MICAS-X will ask for data from the GPS, wait until data is available, then wait 500 milliseconds before requesting more data. The delay time thus acts to make the loop more efficient, by not having the Acquisition Loop ask for data until it is almost time for the data to be available.

Another way to view the difference between software-timed Acquisition Loops (Device Timed unchecked) and Device Timed Acquisition Loops is as follows. For a software-timed loop, MICAS-X uses its internal clock (the computer clock) to determine when it gathers data from the Driver(s) assigned to it. In the example of a 1000 ms Cycle Time, MICAS-X will note the time when the Acquisition Loop first starts, and will thereafter ask for data a 1000 ms intervals after that time. E.g. it will always try to request data at integer multiples of 1000 ms added to the time of its first acquisition. As an example, assume that the first data point was acquired at exactly 9:00 am and 0.0 seconds. MICAS-X will then try to acquire data at 9:00:01.0. If the Driver does not

respond quickly enough, and data is not acquired until 9:00:01.5, the Driver will still schedule the next data point for 9:00:02.0, not for 9:00:02.5. E.g. it will always try to acquire data based on when the first data point was acquired, plus an integer number of acquisition cycle times.

Device Timed Acquisition Loops defer instead to when the device or Driver makes data available. Take as an example a gas analyzer that streams data once every 2000 ms, without requiring MICAS-X to ask for the data. In this case, the Device Timed checkbox should be marked, and the Cycle Time (which is now used as a Delay Time) might be set to 1500. At 9:00:00.0, MICAS-X begins acquiring data. It waits until the gas analyzer presents data, which may happen at 9:00:00.2. It then waits for the Delay Time, or until 9:00:01.7. At that time, it again looks to the driver for data, which should arrive at 9:00:02.2. In this example, the Acquisition Loop timing is determined by the device, and the Delay Time is used to make MICAS-X more efficient by not looking for data during the entire cycle time.

Only one streaming device should be assigned to any one Acquisition Loop. The Acquisition Loops for each streaming device should be set to Device Timed. Device Timed Acquisition Loops can include other Drivers that are not for streaming devices. E.g. a Device Timed Acquisition Loop for a GPS, acquiring data from the GPS every 1 second, could also acquire data from a NIDaqDI Driver once a second.

Finally, it should be noted that some streaming devices will not respond well to having “Acquire” turned off in MICAS-X. E.g. MICAS-X has the ability to turn Acquire on and off, which determines whether or not the Drivers are queried for data. For most streaming devices, once the device starts sending data, turning Acquire off does not stop the device from sending further data. In this case, when Acquire is turned back on, there may be buffer overflows or other communication errors. If MICAS-X is configured to communicate with these types of devices, it may be advisable to configure it so that it is always in Acquire mode. To do this, the Acqurie at Start-up checkbox in the MICAS configuration should be checked, and any “Acquire” button that might be configured should be removed.

The Acquisition module works in both MICAS-X for Windows and MICAS-X-RT.

7.4.3 Channel Lists

Channel Lists are lists of Housekeeping Channels and can be used for several purposes. Often, for a complex MICAS-X configuration, the number of Housekeeping Channels can become fairly large. When selecting a channel to display on a time-series graph, for example, it can become difficult to quickly find the desired channel amongst

the list of all the channels. In this case, one can create a Channel List of just the important channels that one wants available on the time-series graph. Another use for Channel Lists is to define a set of channels that should be written to a .csv data file.

The “Channel Lists” array lists the channel lists that have been created for the current configuration. You can delete a channel list by pressing the red delete button to the right of the appropriate channel list. To create a list, press the green “New” button located at the top and to the right of the delete buttons. The new channel list will be given a default name of “New List”. To edit the list’s name, highlight the name in the “List Name” box in the center of the window and type in a new name.

Once a list has been created, you must define the Channels in that List. First, highlight the list in the Channels Lists box on the left. Any Channels already in this List will now be displayed in the Channels array in the center. Existing Channels can be removed from the List by pressing the red Delete button to the right of the Channel. The Insert button will add another channel at the desired location. Once a new Channel has been inserted, click on the name of the new channel to see a drop-down menu of all available channels, from which the desired channel may be selected.

Additional buttons to the right allow you to delete all the channels in the current list, set the current list to a list of all existing channels, or set to add all Controller (output) Channels to the current list.

Further to the right are three controls that allow you to add Channels to the current List which are related to any specific Driver. Use the Driver control to select which Driver you want to add Channels from. Then press either the “Add All Driver Channels” or “Add All Driver Controllers” to add the relevant channels to the currently-highlighted Channel List.

Directly under the Channels list in the center is the “Sort This List” button. This button sorts the channels in the above array alphabetically.

Note that, in addition to any lists you create, MICAS-X will always have a list named “All” which includes all MICAS-X channels.

Channel Lists work in both MICAS-X for Windows and MICAS-X-RT.

7.4.4 File Writer

The MICAS-X File Writer allows you to define any number of data files MICAS-X writes using the Housekeeping Channels. The Files cluster of parameters is an array of file definitions. A new File can be added by incrementing the number on the upper left

hand side of this cluster, then entering the parameters in the new, blank definition. Change the Prefix (characters prepended to the file name) and Suffix (characters appended to the file name) of the file in question by entering the desired names into the "Prefix" and "Suffix" boxes, respectively. Enter the desired file extension in the "Extension" box. (Note that .csv is recommended for Delimited Text files, and .itx is recommended for Igor .itx files.) To choose how a file is delimited, select the option you prefer using the "Delimiter" parameter (comma, space or tab separations). (The "Delimiter" parameter is used only for Delimited Text files and is ignored for Igor .itx files.)

The "Format" parameter acts as a format specifier, which defines how data will be written into the file. "%#g" is recommended, as this will use automatic formatting. See LabVIEW Help for more information on other types of specifiers. Note that when MICAS-X writes a value that it thinks is a LabVIEW time-stamp to a file, it over-rides the format specifier and uses "%.3f", which ensures that the time-stamp is written with sufficient resolution to preserve sub-second information. This is applied to data values in the range 3.4e9 to 4.2e9, since LabVIEW stores time in seconds since 1904.

The "Channel List" parameter determines which of your preset channel lists will be written to the file in question. Use the arrows to select which list will be used.

The "Start a New File Every" parameter allows you to choose how often a new data file is created. Select a time between every minute and two days (or never, if you never want MICAS-X to automatically restart your data file). The "Acq Loop" parameter determines which acquisition loop writes the file in question. Different loops can run at different rates, but whenever a loop writes to a file, it gathers all the most recent Driver data for all the channels in its Channel List.

Use the "File Type" parameter to define the file format used to write the data file. Currently-supported options are Delimited Text (e.g. .csv files) and Igor .itx. The .itx file type is used by the Igor data analysis program. Note that other File Writer parameters that are not relevant to the selected File Type will be greyed-out.

When the Igor .itx file type is selected, the Channel names are written to the file as Wave names. In order to provide compatibility with Igor, Channel names will be altered in certain ways. The characters "!@$^&(),.<>/?;':"\|", which are not allowed in Igor Wave names, are replaced with the underscore character "_". The character "%" is replaced with "pct", and the character "#" is replaced with "Num.". In addition, certain Igor reserved words are altered by added "_X" to them. These include "Time", "Debug", and "Sec". Note that not all Igor reserved words are currently filtered, so it is important

to test your channel list by creating an Igor .itx file and reading it into Igor. You may need to alter some MICAS-X channel names in order to get this file to work properly.

The “Wave Flags” and “Igor Commands” parameters are only used with the Igor .itx file type. “Wave Flags” can be used to set various Igor options. The most common flags to use are \O to have Igor overwrite any existing waves that are already loaded that have the same name as waves in the file, and \D to tell Igor to make the new waves as double-precision. See Igor documentation for additional flag information. Similarly, the “Igor Commands” is used to add Igor instructions at the end of the file. These can be used to automatically format and graph the data when it is loaded in Igor, for example.

The “Custom Path” parameter allows you to define a specific destination for the file being configured. If this is left blank, the default data directory is used. Otherwise, the data file is written to a date-named directory inside the path specified here.

Use the Write Frequency (Once Every n Loops) parameter to deteremine how often data is written to the file. Normally, this is set to 1, which causes data to be written on every iteration of the Acquisition Loop defined for the file. Setting this to 2 causes the data to be written every other time through the loop. As a further example, if the Acquisition Loop has a cycle time of 1000 ms (1 sec) and the Write Frequency is set to 60, the data will be written to the file once a minute. Note that these file writes are not synchronized to an even second or minute. E.g. in the above example, if the program starts at 5 seconds after a minute, data will be written to the file at every hr:mn:05 seconds, not at every hr:mn:00 seconds.

Note that the .tdms file, which is continuously being saved in the background, is written by Acquisition Loop 0. Thus the .tdms file will have a sampling rate determined by the Acquisition Loop 0 cycle rate. If Acquisition Loop 0 is running at 1 Hz and one Driver is running in Acquisition Loop 1 at 10 Hz, it is important to realize that the .tdms file will only record 1 out of every 10 data points from that Driver. If it is necessary or useful to save all the data points for that Driver, two options are possible: 1) Reconfiguration the acquisition so that the fast (10 Hz) Driver is in Acquisition Loop 0 and configure that loop to run at 10 Hz, and configure the other, slower Drivers to run in another (or multiple) acquisition loops at their own rates. Using this method, the .tdms file will be written at 10 Hz and all the fast Driver data will be recorded therein. Note that all the 1 Hz Driver data from the other loops will be written to the .tdms file at 10 Hz, creating 10 duplicate values for each value read in. 2) Leave the 10 Hz Driver in Acquisition Loop 1, and set up a .csv file to write a Channel List of just this Driver's data, and configure that File to be written by Acquisition Loop 1. In this case, the .tdms file will only have 1 Hz data, which is likely sufficient for data review and time-series graphs, but the higher frequency 10 Hz Driver data will all be saved to the .csv file.

The “Delete” Button to the right of the “Files” cluster will delete the File definition currently shown.

The File Writer module works in both MICAS-X for Windows and MICAS-X-RT.

7.4.5 Control Tab

The Control Tab module of MICAS-X Configuration Editor allows you to configure how the Control Tab looks and acts while the program is in use. The first box, “Control Tab Name,” allows you to rename how the Control Tab appears when MICAS-X is being run.

The “Controllers” cluster is an array that defines which Controllers are available to use on the control tab. Use the Insert and Delete buttons to add and remove Controllers from the list. Click on the name of a controller to access the list of all available Controllers and select the one desired.

To the right of the “Controllers” cluster, there is an option to “Set to All Controllers,” which will set the above list to all Controllers defined within MICAS-X. The “Delete All” button will remove everything you have added to the list.

The “Switches” cluster is an array that defines which Controllers are presented on the Control tab as switches. Controllers that logically have only two values (on/off, True/False, 1/0, etc.) are logical candidates for presentation as Switches on the Controllers tab. Add, remove, and select which controllers are used as Switches the same way as described above for Controllers.

In addition, the Off Value and On Value need to be defined for Switches. The Off Value defines what value the Controller channel will be set to when the switch is turned to the Off (down) position. This is normally 0. The On Value determines the value the Controller channel will be set to when the switch is turned On. This is often 1, but other values could be used as appropriate. For example, an analog output channel could be used as a digital control channel by making the On Value 5, so that the Switch would change the channel between 0V and 5V.

The “Sequences” cluster is an array that defines which sequences are presented on the Control tab. Add, remove, and select Sequences the same way as described above for Controllers.

The “Triggers” cluster is an array that defines which triggers are presented on the Control tab. Add, remove, and select Triggers the same way as described above for Controllers.

The “Graph” parameters allow you to choose whether or not a time-series graph is displayed in the Control tab and how it is configured. Toggle “Show Graph” to determine if the graph will be shown in the tab. When the Graph is selected for the Control Tab, it will appear at the bottom of the tab, and the Controllers, Switches, Sequences and Triggers lists will only occupy the top half of the tab. If the Graph is not selected, those lists can extend all the way to the bottom of the tab, depending on how many items they contain. In either case, if they contain more items than fit in the allotted space, a scroll bar appears so that all the items in the list(s) can be viewed as needed.

The “Left Axis” switch allows you to select Auto or Manual scaling on the left axis of the graph when MICAS-X first starts. The “Right Axis” switch functions like the “Left Axis switch.” “Graph Background” allows you to select the color of the background of the graph. “Left List” and “Right List” allow you to choose which lists of channels are are used to define the channels available to display on your graph’s left and right sides, respectively. “Left Channel” and “Right Channel” determine which channels initially appear on the left and right parts of the graph, respectively. “History Length” determines how far back the graph displays data. (Note that “Manual” and “From Pointer” cannot be selected as an initial value for this parameter. “All” will be used instead if “Manual” or “From Pointer” if either is the initial configuration value.) “Left Points” and “Right Points” determine whether points are displayed at each data point (as opposed to just plotting a line between the data points).

The “Channel List” control defines which channel list will determine the channels whose values are displayed in the table on the right of the control tab.

Below the “Channel List” parameter is a switch used to select “Show Message Logging” or “Don’t Show” Message Logging. If set to Show Message Logging, the Channel value table will be shortened by four rows, making room for a control at the bottom of the Control tab that can be used to type messages and send them to the log file.

The Control Tab configuration is available only for MICAS-X for Windows and is not supported in MICAS-X-RT.

7.4.6 Sequences

The Sequences Module allows the operator to define any number of Sequences which can automate much of the MICAS-X functionality. Multiple Sequences can run simultaneously. Sequences can use any of the MICAS-X commands listed in Appendix B, can reference any MICAS-X Channel as an input or condition channel, and can act on any MICAS-X Controller channel. Sequences can include branching (conditionals), looping, and timing. Note that sequence timing has sub-second resolution, but is intended for non-time-critical functions that typically operate on the time scale of seconds. Programmatic actions that require more precise timing should be implemented in MICAS-X in a custom-programmed Instrument module. As of version 1.5 of MICAS-X, Sequences can be paused, unpaused, and can be started at an arbitrary step. (Note that for the Start Sequence command, the step is specified by number. Specifying the step by its label is not currently supported.) Note that when a Sequence is paused, any Wait that may have been in process is considered finished when the Sequence is Unpaused. E.g. if a Sequence has a long wait that you wish to abort and have the Sequence continue without waiting the full time, you can Pause and then immediately Unpause the Sequence.

As of version 2.1.4 of MICAS-X, Sequences also include Scripts (see Section 7.4.8.) Scripts are a more powerful way of writing Sequences, but cannot be created with the Sequence Editor.

The Sequence Module is built into MICAS-X and is always present, and can be used with or without the Sequences Driver or the Sequences Display. The Sequences Driver is an optional MICAS-X Driver that provides additional functionality for Sequences, as described in the Drivers section of this document. The Sequences Display provides a display tab that allows the operator to see all the sequences, review what each step of each sequence does, start and stop sequences, and view the current step of any sequence that is running.

The Sequences editor allows the operator to define the sequences available in MICAS-X. In the “Sequences” list in the upper left, click on a sequence you wish to edit to highlight it. To add a new sequence, click “Insert.” Once a new sequence has been added, make sure it his highlighted, then the Sequence Name and other parameters can be edited. (Note that you cannot directly edit the Sequence Name in the Sequences list box.) To delete a sequence, click “Delete.”

In the “Steps” list near the top center, insert and delete steps in a manner similar to how Sequences are inserted and deleted. To edit the sequence name, type in the desired name in the “Sequence Name” box. “Off Label” allows you to define the text that is visible on the sequence switch when it is in the off position. If left blank, the

sequence name is used. “On Label” is used in a similar manner for the text on the switch when in the on position. Two switches below these buttons allow the user to include an Exit Step and to optionally Log each step of the sequence as it executes. An exit step is a single sequence instruction that is executed automatically whenever a Sequence stops. This step will execute regardless of why the Sequence is stopped – e.g. regardless of whether it ran its course and stopped normally, or the user clicked the Sequence button to stop it, another command stopped the Sequence, or the program stopped completely. If more than one command must be executed when a Sequences stops, it is possible to define another Sequence with all of those commands, and use the “Start Sequence” command as the Exit Step to call the other Sequence.

Once a Sequence is highlighted in the Sequences list box, you can program each step of the sequence. Give a step a label by typing in the appropriate “Label” box. Labels are semi-optional, but it is good practice to give each step of each sequence a unique label. Labels server two purposes. They provide documentation to help understand what the sequence step is intended to do, and they provide a target for the Goto Sequence Command.

Next to the Label, select one of the available commands to execute by clicking the white “Command” box, or by cycling using the arrows to the left of the box. “Condition” defines what condition is applied to define a sequence step. A Condition of “True” (the default value) means that a step will always be executed when the Sequence comes to it. A condition of “False” is a way to disable a step without removing it, as it means that the step will never be executed. Other conditions compare the value of the Condition Channel to the Threshold to determine if the step will be executed. For example, the “>” Condition means that a step will only be executed if the value of the Condition Channel (at the time that the step is executed) is greater than the Threshold. Note that three special conditions are included that deal specifically with the value “NaN”, or Not-a-Number. Refer to section 7.4.8 for more information on these conditions.

“Target” defines what the step will act on. The contents of the Target drop-down menu change depending on the Command selected. For example, if the Goto Command is selected, the Target drop-down list will contain all the Labels in the current Sequence. If the Set Command is selected (to Set a Controller Channel to a new value), the Target list will contain all the available Controller Channels. The “Value” parameter specifies the value that is used by the Command if the step is executed. The Target and Value parameters are greyed out for Commands that do not require a Target or Value.

At the bottom of the window, “Minimum Sequence Time” allows you to select the time in milliseconds that the program must wait between each step in a sequence.

The recommended time is 10 ms. Making it smaller can lead to more accurate sequence timing, but can potentially impact the overall performance of the program.

"Startup Sequence" and "Shutdown Sequence" options allow you to select sequences which automatically run each time MICAS-X launches and finishes, respectively. If you want more than one sequence to run at start-up or shutdown, one can create a new Sequence with a list of "Start Sequence" Commands, then call this new Sequence as the Startup or Shutdown Sequence.

Sequences can be exported and imported to .seq files for archiving, duplicating, or sharing. The Export button will write the currently selected sequence to a .seq file named by the operator. The Import button will read a .seq file and incorporate the new sequence before the currently selected sequence. To import a sequence at the end of the list of existing sequences, use the Insert button to add a new, blank sequence at the end of the list, then import the desired sequence while the new sequence is selected. Finally, delete the unwanted blank sequence.

The "Include Time?" parameter is used only for the Sequences Driver. It is ignored by the Sequencer module itself and by the Sequences Display. If this is set to True, the Sequences Driver channels are prepended with a timestamp channel so that the exact time that those channels were recorded is noted in the data file.

When the mouse is inside one of the Command selectors, help information for using that command will appear where the Notes normally are. Move the mouse over the Sequences or Steps lists to make the Notes reappear.

Additional information on Sequences and how to use them successfully in MICAS-X can be found in Section 9.6.4Appendix C: User Dialogs and Notifications. Information on how Sequence Sets can be defined and used can be found in Section 7.4.7.

The Sequences module works in both MICAS-X for Windows and MICAS-X-RT.

7.4.7 Sequence Sets

Sequence Sets allow Sequences to be grouped into logical sets for clarity and for increased control of the Sequences. The set "All" contains all defined Sequences and always exists. It should not be explicitly defined, since it is included automatically as a Sequence Set when the program is run. Use the Sequence Sets Configuration screen to define additional Sets. In this configuration window, name any Sequence Sets in the Sets list. Use the Delete and Insert buttons to its right to remove or add Sets from this list.

When one of the Sets in the list is highlighted, you can edit that Set. The Sequence Set Name can be edited in the control to the upper right. The Sequences that are members of that Set can be edited in the array to the right. Use the Delete and Insert buttons to remove and add Sequences to the selected Set. Note that all the Sequences you wish to add to a Set must have been defined in the Sequences Editor before using the Sequence Sets Editor. Also note that any Sequence can be included in as many Sets as desired.

Once Sequences have been assigned to Sets, one can select which Set to view in the Sequences Display and the XRT Sequences Display. By filtering which Sequences are displayed in this way, configurations with many Sequences defined can more easily be understood. The “Initial Set to Display” parameter in the Sequence Sets configuration editor determines which set is displayed when the program starts. The “Instructions” parameter allows one to include run-time instructions for the operator regarding Sequences. If this parameter is not blank, the text here will appear on the Sequences Display when the program is run.

In addition to selecting which Sequences to view, the Sequence Sets can be used with the Command StopSeqSet. This allows a single Command to stop a number of Sequences.

7.4.8 Scripts

Scripts were introduced in MICAS-X 2.1.4. Scripts are a more powerful type of Sequence. Scripts are stored outside of the MICAS-X configuration file in files with the .mscr extension, in the Scripts subdirectory of the support folder. The file name is the name of the Script in MICAS-X. The contents of a script file is simple text, with each line making up one Script line, as per the format described below. When a Script file is loaded into MICAS-X at run time, it is converted to a Sequence. The name of the Sequence is the Script name (e.g. the script file name), and each line of the Sequence is a Sequence Command “Script” with the string argument being the line from the script file. Once a Script has been loaded into MICAS-X, it is treated the same way as all other Sequences. It can be stopped, started, and paused with the same Commands. It can be displayed in the Sequence Display. Script files are usually created by custom tools in MICAS-X written for specific applications or customers. However, it is possible to create a Script file in a text editor, if care is taken to comply strictly with the syntax.

Use the Scripts Editor window to tell MICAS-X which Scripts are to be included in the current configuration file. Although simple Scripts can be written by hand in the Scripts Editor, Scripts are usually created through custom Utilities.

The Scripts list box on the right side of the editor show the Scripts that have been loaded into the current configuration file. Use the red “Delete” buttons to its right to remove a Script from the list. Click on a Script in the list to highlight it. When a Script is

highlighted, the contents of the Script file is displayed in the Script text box on the right. If this text is changed or edited while it is displayed, the changes are automatically saved to the selected Script file.

The Available Scripts list displays those Scripts present in the Script directory of the MICAS-X Support directory, but which have not yet been added to the list of Scripts included. Double click on a Script in this list to add it to the current MICAS-X configuration. To create a new Script, enter a name in the Script Name box, then press the New Script button. An empty file with the name and the .mscr extension will be created in the Scripts directory, and the Script will be added to the Scripts list. At this point, you can write the script manually if you wish, by typing in the Script box. The Commands, Channels, Triggers, and Drivers lists below the text box are intended to help with the writing of Scripts, since you can copy and paste these names into a Script as needed. Note that all these names are case sensitive.

The general syntax of a Script line is:

[label]command:string:expression;string:expression;string:expression{expression=expression}'comment

Label: The “label”, inside the square brackets, is optional. It acts as the step label, just as for a Sequence, and can be referenced by the Goto Command.

Command: The “command” is the Command to be executed. It must be spelled exactly, but is case insensitive. Refer to Section 14 for a list of Commands.

Parameters: The format of the parameters depends on the Command being used. The supported combinations of parameters are:

string
value
string:value
string:string

The string parameters can be a Channel name, a Controller Channel name, a Sequence name, a Trigger name, a Driver name, or a Sequence or Script Label. Any value can be a constant, a Channel name, or an Expression. (See the discussion of Expressions in the section below about Conditions.)

Multiple groups of parameters can be included, separated by semi-colons. When multiple sets of parameters are used, each set is sent with the same Command that the Script line specifies. These multiple instances of the same Command are sent in the order in which they appear in the Script line.

Condition: The text inside the curly brackets is an optional Condition. It consists of two expressions separated by a conditional symbol. Each equation can be a constant, a Channel name, or an expression made up of constants, Channels, and functions. The syntax used by expressions is the same as that supported by the Equations Driver, and is documented in Section 17.

The conditional symbols supported are:

False	the command is never executed
True	the command is always executed
<	executed when the first expression is less than the second expression
<=	executed when the first expression is less than or equal to the second expression
=	executed when the first expression is equal to the second expression
<>	executed when the first expression is not equal to the second expression
>=	executed when the first expression is greater than or equal to the second expression
<	executed when the first expression is greater than the second expression
=NaN	executed when the first expression is equal to Not a Number
<>NaN	executed when the first expression is not equal to Not a Number
<>NaN,0	executed when the first expression is equal to Not a Number

Comment: A comment starts with an apostrophe, and is optional. It is ignored by MICAS-X and is only used to provide clarity to anyone reviewing the Script. If a comment is on a line by itself, it is ignored and no line is included in the Sequence that is created from the Script when it is loaded into MICAS-X.

Below are some example Script lines:

[set valve1]Set:valve1:0{temp>25|'close valve0 if the temperature is greater than 25

This Script line has the label [set valve1], and a comment that describes what it is doing. The Set Command will only be executed at runtime if the Channel "temp" has a value greater than 25 when the Script line is run.

Set:valve1:1;valve2:1;valve3:0

This Script line sets three valve Channels, each to the value 0.

‘The lines below are the emergency shutdown procedure.

The Script line above is a comment only and is ignored when the Script is loaded into MICAS-X.

[add offset]Add:output1:5 + channel3

The Script line above adds the value of the Channel “channel3” and the constant 5 to the Channel “output1”. Note that the Add Command has arguments of a Channel and a Value. When used in a standard Sequence, the second parameter can only be a constant. When used in a Script, the second parameter can be an expression, including other Channel values.

7.4.9 Triggers

The “Triggers” Module allows you to define Triggers and Alarms which will be used in MICAS-X. The Triggers Module is always part of MICAS-X, and can be used with or without the “Triggers” Driver. The Triggers Driver adds additional functionality to the Triggers and is described in the Drivers section of this document.

A Trigger is a set of parameters that define a Command that will be executed whenever a Condition is met. MICAS-X continuously scans the defined Trigger Conditions and executes a Trigger whenever the Condition occurs. Alarms are a special type of Trigger that provide additional feedback to the operator. If any Alarms are defined, an Alarm indicator appears in the upper right corner of MICAS-X. The Alarm indicator is green (OK), yellow (Warning), or red (Alarm) depending on the highest state of all defined Alarms.

The “Triggers” list displays the defined triggers. Click on a trigger and highlight it to edit it. Insert a trigger using “Insert,” and delete one by pressing “Delete.”

Once a Trigger is highlighted, the various parameters for it can be edited. The “Trigger Name” allows you to change the name that describes the highlighted trigger by writing in the desired name in the box. The “Log” switch will allow you to choose whether or not the Trigger actions are logged when they occur. The “Alarm/Trigger” switch allows you to determine if the highlighted item will be a Trigger or an Alarm. “Min Time” specifies the minimum time in seconds that the trigger condition must be met before the trigger is set to true. Set this to 0 to make the trigger go off immediately upon recognizing a Condition. Setting higher can prevent noise from causing a trigger. If the Trigger Condition becomes True but the Min Time has not been met, the Trigger is set to the Warning condition. If the Condition remains True long enough for the Min Time to

be met, the Trigger will go into the True or Alarm state. If the Condition becomes false before the Min Time is met, the Trigger state will revert back to false or OK.

The “Condition Channel” box allows you to choose, from a drop-down menu, which channel is checked for the condition to determine the Trigger state. “Condition” determines which condition will cause the trigger to change. “Threshold” allows you to specify the value that the channel will be compared to, using the condition, to determine if the Trigger occurs. The “Hysteresis” parameter applies hysteresis to the trigger condition, to avoid noisy data from causing the trigger to oscillate.

Note that if condition = is used, a positive, non-zero Hysteresis creates an “In-Range” condition. E.g. for a Threshold of 10, a condition of =, and a Hysteresis of 2, any values between 8 and 12 will yield a true result. Similarly, for a < > (not equals) condition, non-zero Hysteresis creates an “Out of Range” condition. For both of these cases, the limits are inclusive of the values of the Threshold, plus or minus the Hysteresis.

Three conditions are included to deal with the value “NaN”, or Not-a-Number. NaN is used in MICAS-X to indicate missing data. For example, all Driver input channels are set to NaN when the program starts, and only have valid data after the first acquisition loop. Using normal conditions with NaN can yield surprising results. For instance, a value of NaN with the conditions > 0 or < 0 both give a result of False. If a condition must be sure to test for NaN correctly, use one of these conditions: =NaN, <>NaN (not equal to Nan), or <>NaN,0 (not equal to NaN and not equal to 0). The last condition is especially useful for MICAS-X channels that are used as Booleans. Booleans are typically encoded in MICAS-X as 0 for False, and 1 for True, though any non-zero value is also interpreted as True. The last of the NaN conditions, by checking for both <>0 and <>NaN, ensures that only a truly non-zero value is interpreted as a True value, and avoid interpreting a start-up value of NaN as True.

The Include Acknowledge Dialog parameter, if set True (On), will cause a dialog box to pop up whenever the Trigger/Alarm becomes True or Alarm. The dialog will remain until the operator closes it. Similar functionality can be created with the Alert command, but use of this option allows a different command other than Alert to be acted on while still presenting a dialog.

The Start Disabled? parameter determines whether or not the Trigger/Alarm starts normally, e.g. enabled (Start Disabled? = False), or if it is Disabled (Start Disabled? = True) when the program starts. Triggers can be enabled and disabled while the program is running using the DisableTrigger and EnableTrigger Commands.

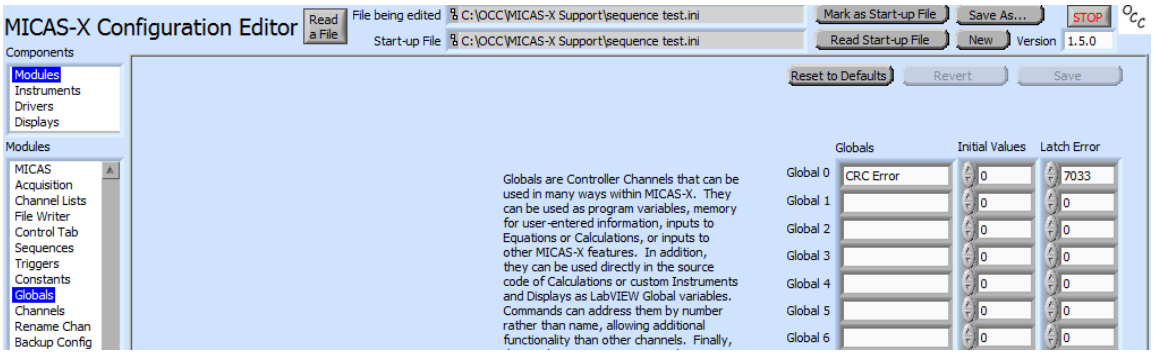
The State of a Trigger can have one of five values. (By including the Trigger Driver in the configuration, additional channels are created which hold the State and Theshold values for each Trigger.) The five possible values are -1: Disabled, 0: False, 1: Warning, 2: True, 3: Alarm. Note that Disabled is a type of “False” case. E.g. if a Trigger goes from True or Alarm to Disabled, the True-to-False action is executed. Warning means that the Condition for the Trigger has been met, but that there is a Minimum Time set which has not yet expired. Alarm is a type of “True” case, but is used when a Trigger has been specified as being an Alarm. All Alarms are evaluated together to determine the state of the Alarm indicator at the upper right of the MICAS-X screen.

When the mouse is inside one of the Command selectors, help information for using that command will appear where the Notes normally are. Move the mouse over the Triggers list to make the Notes reappear.

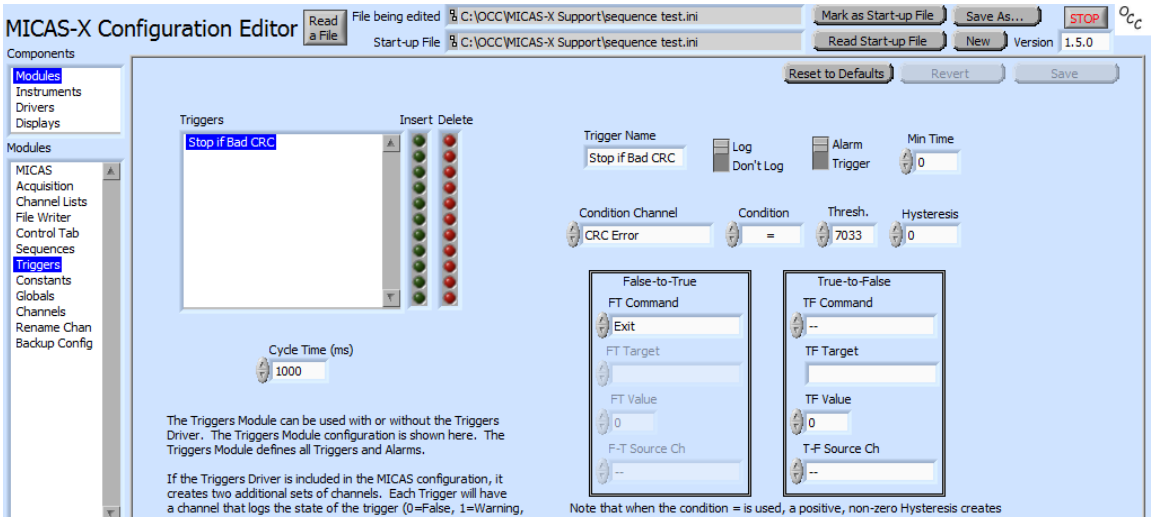
Example: Handling the CRC Error

For versions 1.5.0 and later of MICAS-X, the Configuration Editor places a CRC value in each configuration file. If MICAS-X is run and finds that the CRC value is not valid, indicating that the configuration file is corrupt, missing, or has been edited by hand, it is possible to use Triggers or Alarms to automatically handle this situation. In the first example here, an Alarm is configured which stops MICAS-X when the CRC is not valid. The second example causes an Alert to be shown, so that the operator is definitively notified of the issue. (Note that the CRC error will always be logged in the log file, but the use of the alert in the second example ensures that the operator does not overlook that subtler notification.)

For both examples, a Global must be used to latch the specific error. See section 7. 4. 11 for an explanation of why monitoring the Error channel directly is unreliable. The figure below shows the configuration for setting up Global 0 with the name “CRC Error” and the Latch Error value of 7033, which is the error code for when the CRC of the configuration file is invalid.

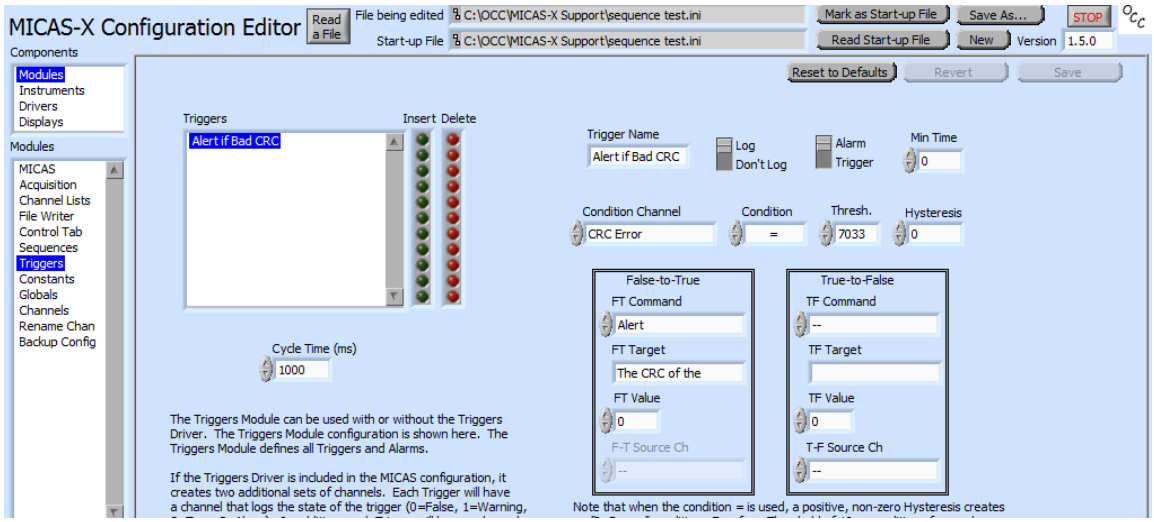


The second step is to create a Trigger that watches the Global and exits MICAS-X if the error is detected. The configuration for this is shown in the Figure below.

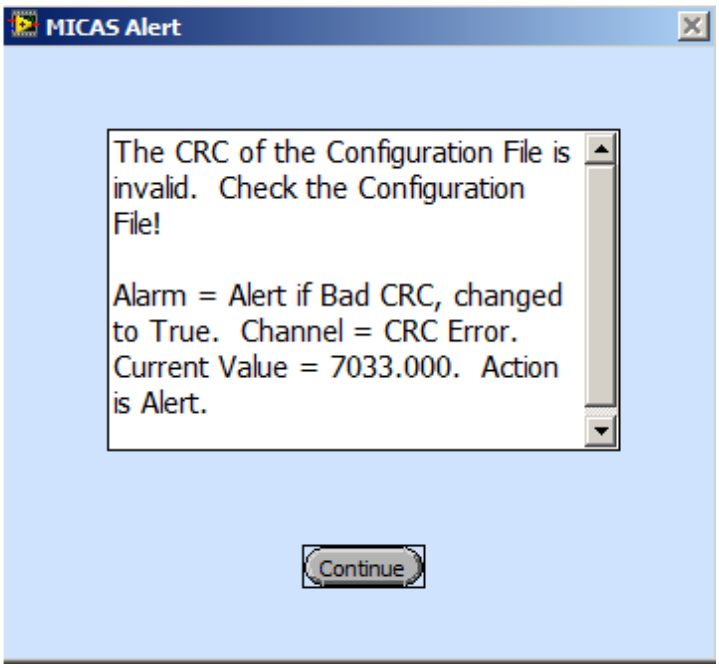


The combination of these to configuration steps will result in MICAS-X stopping automatically if the CRC error is present.

For the second example, the Global is configured the same as above, but the Trigger is configured as shown below. Note that the Trigger/Alarm parameter has been set to Alarm, to give make the alarm condition more visible in MICAS-X. Also note that the FT Target string is not completely visible in the screenshot below, but is shown in full in the following figure where it appears in the Alert. The FT Value of 0 tells MICAS-X to create the default Alert dialog box. See section 14 for other options of the Alert Command.



As a result of this configuration, when the CRC is detected as invalid, the user is presented with the following dialog box, which will not disappear until the user clicks on "OK". The MICAS-X program will continue even while the dialog is open, however.



The Triggers module works in both MICAS-X for Windows and MICAS-X-RT.

7.4.10 Constants

The final entry in the Modules list in the Configuration Editor is “Constant”. This configuration screen allows you to define any number of constants to use in the MICAS-X system. Constants are similar to channels. They can be used in equations, calculations, sequences, triggers, and commands. Unlike channels, however, they are not written to data files, do not show up in the lists of channels that can be shown on graphs, and cannot change value while the program is running. The value of each constant is determined in the configuration file, and the archive of the configuration file serves as the record of the constant values.

In the Constants configuration screen, enter names for constants in the “Constants” list, and enter the values of those constants in the Values list. The Delete and Insert buttons can be used to remove or add more constants to the list.

The Constants module works in both MICAS-X for Windows and MICAS-X-RT.

7.4.11 Globals

“Globals” are optional channels that can be accessed by any part of the program, e.g. they have global scope throughout the program. If you do not assign a name to a global variable, it will not be used within MICAS-X. Globals can be used as program variables, memory for user-entered information, or inputs to Calculations or Equations, among other things. There are a special set of commands that allow Globals to be referenced by index number, which adds additional functionality to the Commands. Finally, the Globals can be used directly in Calculations or custom Displays or Instruments just as a normal LabVIEW global variable. This allows Calculations, for example, to have configurable parameters by using the Globals configuration page.

The “Initial Values” list allows you to select the initial value of the Global when MICAS-X starts. The Latch Error list enables a special function of Globals. The Error Program State Variable automatically contains the error code number of the most recent error to occur. However, more than one error can happen during a data acquisition cycle, so the Error channel serves only as a snapshot to see if any error has happened. If you need logic that can identify a specific error without risk of missing it, use a Global with the Latch Error function. Set the Latch Error value to the numeric error code you wish to monitor, and assign a meaningful name to that Global. If that error ever happens, the associated Global will be set to that error code value and will not return to zero until it is explicitly set. Thus the Latch Error function can be used to enable Triggers or Sequences or other mechanisms to catch any particular error, even if many errors are rapidly occurring.

The Globals module works in both MICAS-X for Windows and MICAS-X-RT.

7.4.12 Enumerated Lists (Enum Lists)

An Enumerated List is a list of numeric values, each of which is associated with a text string. These can be useful for display Mode or Status channels. For example, a Channel might take on the values 0, 1, or 2, depending if the system being controlled is in Off, Standby, or Run mode. The enumerated list 0=Off, 1=Standby, 2=Run could then be used to display that Channel in a way that is much more intuitive for the operator than if the numbers 0, 1, or 2 is displayed. Note that Enumerated Lists in MICAS-X are a user-interface feature. The data for all Channels is stored inside MICAS-X and in all data files as numeric data, not as the text equivalents.

Prior to version 3.2, MICAS-X had enumerated lists available for the Options that appear above the main MICAS-X tabs, and within the Multi Display. In previous versions, the Enum Lists were defined in the MICAS-X Editor and the Multi Display Editor. As of version 3.2, the Enum Lists are defined in the Enum List Editor. The MICAS-X Options and the Multi-Display then can select an existing Enum List this single set of configured Enum Lists. In this way, the same Enum List can be used in multiple parts of MICAS-X without having to be defined separately in each location.

To define an Enum List in the Enum List Editor, use the Insert buttons to the right of the Enum Lists array to add a new Enum List. (The Delete buttons can be used to remove Enum Lists when needed.) When the new list is highlighted, edit its Name in the Name field to the right. Then, with it still highlighted, use the Enumerated List array to define pairs of numeric values and text values. The Delete and Insert buttons to the right can be used to remove and add pairs.

In many software systems, enum lists are limited to integer values. MICAS-X allows non-integer values in the numeric/text pair, but it is recommended that integer values normally be used.

Enumerated Lists can be used as Channel Indicators or as Channel Controls in MICAS-X. The value “[other]” is automatically added to the list of possible text values. If the Channel ever has a value that is not included in the defined Enum List, then the text “[other]” is displayed.

7.4.13 Commands Menu

As of version 3.3.7 of MICAS-X, the user can configure an additional menu item to appear at the top of the MICAS-X window. This menu will have the name “Commands”, and can contain any number of entries. The entries are descriptive names given to commonly-used Commands that the user wants easy access to. Use the Commands Menu Editor in the Modules section to configure these Commands.

The Command Names box to the left is a list of all the Commands that have been configured. Use the Delete and Insert buttons to remove or add Commands from this list. This list will show the descriptive Command Name, as it will appear in the menu, not the Command itself. For example, one might have the Command Names “Set Tank Temp to 45” and “Set Pressure to 120” for two different entries. Each entry would use the

same Command (e.g. “Set”), but the descriptive names allow them to be easily differentiated.

Once a Command Name is highlighted in the Command Names list, the parameters defining that Command are shown to the right. Use these parameters to specify the Command Name, the Command, the Channel, Sequence, Trigger, Driver, or other string used by the Command, any Value used by the Command, and the Source Channel, if applicable.

Make sure the “Command Menu Visible” box is checked if you wish the Command Menu to appear in MICAS-X. Unchecking this box ensures that no Command Menu is present, typically for the case when no Commands have been defined.

7.4.14 Security

This configuration panel allows you to define a backup configuration file and to set a password for MICAS-X. See section 7. 3 for more information on the Backup Configuration file.

The password feature of MICAS-X provides a mechanism for preventing unauthorized operation of selected features of MICAS-X by untrained individuals. It is not intended to provide security against malicious attack.

The password used for MICAS-X is encrypted before it is stored. The current password is never displayed. If the configuration file itself is not password protected (as determined by the “Secure Config File” parameter), a new password can be assigned at any time. If the configuration file is secured, the password must be entered before the configuration file can be accessed, after which a new password can be supplied if desired.

To enter a new password, type the new password into the “Password” and “Re-enter Password” parameters. If the two entries match, the “Save Password” button will become available. Press that button to save the new password. Press the Clear Password button to remove a password that has previously been assigned.

When a password is enforced, there will be a lock icon in the upper right of the MICAS-X window, and the Program menu will have a Password entry that is checked. Clicking on the lock or selecting the Password menu item will cause a password dialog to appear. If the proper password is supplied, the lock icon will become unlocked, and the Password menu item will become unchecked.

If the Time to Re-Secure parameter is non-zero, the password will be automatically put back in force after the designated number of seconds have expired. While this timer is counting down, the time remaining can be seen on the MICAS tab of the program in the Lock Time indicator.

The Secure Options check boxes allow one to select any or all of the eight possible options along the top of the MICAS-X window for protection by the password.

If an option is selected, it will be greyed-out and impossible to change until the password is entered. (This applies to options which control channels or commands only. Options which only indicate a channel value are not affected.)

The Secure Exit checkbox determines whether or not MICAS-X can be stopped without entering the password. Note that this parameter only prevents the use of the Exit menu item and the "x" in the upper right corner of the MICAS-X window which stops the program. Even when this option is selected, it is still possible for MICAS-X to be stopped via a command, if the commands are accessible to the operator.

The Tabs list allows each tab of the MICAS-X display to be secured. When a tab is selected, all controls and indicators on the tab are greyed out and cannot be changed until a password is entered. Note that Displays and Instruments that are configured to be Outside of MICAS-X cannot be secured using this mechanism. Also note that all Displays and Instruments should be configured into MICAS-X before selecting which tabs are secured, since the tabs are referenced by number, not name, and adding or removing Displays or Instruments could result in the wrong tabs being secured.

The Security module works only in MICAS-X for Windows and is not supported in MICAS-X-RT.

7.4.15 Advanced

This configuration window contains parameters the are less often edited. Previously to version 3.3.5, some of these parameters were in the MICAS configuration.

The Log Wait Time-Outs parameter determines whether certain time-out conditions internal to MICAS-X are logged as errors or not. Within parts of the UI of MICAS-X and within the Sequencer, when a channel is being set to a value, MICAS-X will often call a Wait For New Value or Wait For Different Value function. This is intended to help ensure that the channel is set to the desired value before the next Sequence Step or next action within the program takes place. However, these functions have a time-out, which allows the program to continue functioning even if the new value does not get set within the specified limit. This parameter determines whether or not such time-outs are logged as errors 7078 and 7079.

The "Delay on Startup" option is used to allow the program to initialize properly when it starts. If it is true, and the "Delay (sec)" parameter is non-zero, a modal dialog box will inform the user to wait while the program initializes, for the number of seconds specified by the "Delay (sec)" parameter. During this time, the operator will not be able to click on the MICAS-X window to affect any actions.

The Update Time is specified in milliseconds. The default value of this parameter is 500ms, or ½ second. This parameter determines the rate at which the Control tab is

updated with new data values. The proper value for this parameter depends on the rate at which Driver data is acquired.

7.5 Drivers

To add Drivers to a MICAS-X configuration, see the Acquisition section (8. 1). See that section also for information about using Prefixes with Drivers. Note that any time a single Driver is used more than once in a MICAS-X configuration, Prefixes are necessary to keep track of the instances of the Driver. Below are instructions on how to configure the drivers that come included with the purchase of every copy of MICAS-X software (e.g. drivers included with the base package).

7.5.1 Common Driver Parameters

There are several parameters that are found in all Drivers, and some that are common to a large subset of Drivers. For brevity, those parameters are described here.

Disabled on Startup? Is False by default. If it is True, the associated Driver will be disabled when MICAS-X first starts. Disabled Drivers to not attempt to communicate with their hardware and hence will not throw errors if the hardware is not functioning correctly. They return “NaN” values for their channels while disabled.

Include Time? Is set to True if you want the Driver to create a Time Stamp channel specifically for its own data. This can be useful when more than one acquisition loop is being used, especially if the loops have different acquisition rates. By making this parameter True for at least one Driver in each loop, you can later determine exactly when each data value was acquired and how the values from different Drivers relate to each other in time.

Include State? Is set to True if one wishes to create a Driver Channel that indicates the State of that Driver. Currently, the only State information that is returned is a 1 if the Driver is enabled, or a 0 if it is disabled. Further state information may be included in the future, encoded bit-wise in this channel.

Record Data Stream? Is a parameter which available in those Drivers which support recording their raw data stream to an ICM file. (Drivers that support ICM file writing are usually for instruments which send all the data for all of their channels in one packet, often without any querying necessary.) This raw data file can be useful in debugging the MICAS-X Driver for the device, or in helping configure the device properly. The ICM file recording can also be turned on and off while MICAS-X is running through the *Driver*FileOff and *Driver*FileON Commands.

Initial Mode is a parameter which is available in a subset of Drivers. This subset includes most of the Driver which have controller, or output Channels. This parameter

selects how the output Channels will be set when MICAS-X starts. If set to True, “Use Configuration Initial Value”, then when MICAS-X starts running, each output Channel will be set to a value that is specified in the configuration file. The False option may be labeled “Use Current SetPoint Value”, or “Use Previous SetPoint Value”, depending on how the device works. For devices which have the ability to read the current value of their output Channels, the “Use Current SetPoint Value” will cause the Driver to synchronize the MICAS-X Channel to the actual value that the hardware is using when MICAS-X starts running. For devices which cannot read back in their current output Channel values, the “Use Previous SetPoint Value” option reads a special configuration file that MICAS-X uses to store Driver output Channel values. These two mechanisms are used to allow MICAS-X to be stopped and restarted without interrupting the values that output Channels are set to.

Check for Serial Port is not a parameter, but is a feature found on many MICAS-X configuration editors for modules that use serial ports. Pressing this button allows one to determine which serial ports are currently configured on the computer that is running the MICAS-X configuration program, which can assist in configuring instruments and assigning them to the correct serial port.

7.5.2 Array.vit

The Array driver allows one to implement a one dimensional array of data that can be used programmatically throughout the MICAS-X program. Multiple arrays can be created by using this driver repeatedly. This type of array can be used, for example, in sequences, to step through a set of process variable values which cannot be easily specified by an equation. For example, a voltage could be stepped through a 1,2,5,10,20,50,100 sequence by creating a seven element array of those values and incrementing the array index in the sequence, then reading the array values.

Enter a name for the array in the “Array Name” parameter. This name will be used to read and write the array values. In addition, it will be used as the basis for several other channels that the driver creates. For an array with the name “ArrayName”, there will also be a channel called “ArrayName Index”, which will be used to specify the location within the array to read or write data. The Channel “Array Name Time” is similar to the Time channel every driver has, and records the time when the Array values were recorded, if the “Include Time” parameter is True. If the “Include Size” parameter is True, then another channel named “ArrayName Size” will be created that is a read-only channel and which returns the number of elements in the Array. (Note that the ArrayName Index channel can have values from 0 to ArrayName Size – 1.)

If “Allow Array to Grow” is not checked, the array size will be fixed by the number of entries in the Values list in the configuration. Trying to set the ArrayName Index

channel past the ArrayName Size – 1 will result in an error. If this parameter is checked, then setting the ArrayName Index to a value greater than ArrayName Size and then writing a value to the Array channel will cause the Array to be extended as needed, with zeros placed in any new elements between the end of the old array and the new value.

Use the Delete and Insert buttons to remove and add values in the array list. The Import from File button can be used to read an array of data from a file. This file must be either a single column of numbers separated by returns, or a single row of numbers separated by commas.

The Array Driver supports two Commands. ClearArray sets all the array elements to the value 0. It does not change the number of elements in the Array. ShowArray causes a window to be displayed which summarizes the current state of the Array. It shows the values in the Array, the Array Index, the Array Size, and several configuration parameters which affect how the Array works. Note that if the state or contents of the Array change, this display does not update to reflect those changes. It only contains a snapshot of the Array state when the display was opened.

This Driver supports the Disabled on Startup?, Include Time?, and Include State? Parameters, which are discussed in section 7.5.1.

The Array Driver works in both MICAS-X for Windows and MICAS-X-RT.

7.5.3 Averager.vit

This Driver allows one to average the value of existing Channels over a user-defined number of acquisition cycles. The Channels to Average array defines the existing Channels that will be averaged. Use the Insert and Delete buttons to the right of the array to add and remove Channels from this list. The # of Pts to Average parameter defines how many acquisition cycles will be averaged for each Channel.

The averages calculated by this Driver ignore any instances of “NaN” that may be in the data stream for any Channels. E.g. If # of Pts to Average is set to 5, and a channel’s values for the last 5 acquisition cycles are 3,3,NaN,4,4, then the average calculated for that Channel will be 3.5, with the NaN value being ignored. Similarly, when this Driver starts, all the Channels to be averaged will be initialized with a buffer of n NaN’s, where n = # of Pts to Average. In this way, as the Channels’ data first becomes available, the average is calculated over the initial points, not over a full set of n points. E.g. after three acquisition loops, the buffer for a Channel (with n = 5) may look like 3,3,4,NaN,NaN, since only the first three elements of the buffer have been populated. In this case, the calculated average for that channel will be 3.333, with the NaN’s being ignored.

This Driver should be used with care, since it can have several unintended side effects if it is not configured correctly. If too many channels are added to the list of channels to average, and/or the Number of Points to Average is too high, performance

of the entire MICAS-X system could be impacted. Note that every channel being averaged will have an internal buffer of size equal to # of Pts to Average, which will be used to hold the channel history for calculating the average.

If Overwrite? is False: Create New Channels, then this Driver will create a new channel for each of the Channels to Average, with the text " (avg)" appended to each of the original channel name. The averaged values will be written to these new channels. The original Channels will be unchanged.

If Overwrite? is True: Overwrite Original Channels, then this Driver will place the Channel's averaged values in the original Channels' location in the main program's Channel Values buffer. In this case, the source Channel's Driver itself will still have the original, unaveraged values in memory, but that unaveraged value will not be accessible to any other part of MICAS-X. The Channel's averaged value will be the value that is distributed to the rest of the MICAS-X system. In order to successfully use this option, it is important to ensure that this Driver is added to the Acquistion list of Drivers immediately after the Driver(s) that contain the Channels to be averaged, and that this Averager Driver must be in the same Acquisition Loop as the Driver(s) whose Channels are being averaged.

Note that if one needs to average channels from drivers that are in different Acquisition Loops, it is possible to instantiate more than one instance of this Driver and assign one instance to each of the Acquisition Loops that need averaging.This Driver supports the Disabled on Startup?, Include Time?, and Include State? Parameters, which are discussed in section 7.5.1.

7.5.4 Calculations.vit

The Calculations Driver allows the user to calculate outputs based on one or more channels' values. To configure a calculation, begin by double-clicking on an option in the "Available Calculations" box in the upper-left-hand part of the window. This list contains all the Calculations installed on your system. Any calculation labeled ".vit" can be used more than once. The selected Available Calculation will be moved to the box directly below, labeled "Calculations". Click on the Calculation in the "Calculations" box to highlight it and edit its parameters. When a Calculation is highlighted, a Description will appear on the box to the right. This information will tell how many input and output channels the Calculation must have in order to work correctly, and what their order must be.

The "Channel Averager" calculation takes a single input and averages it over N samples to create a single output channel (where N is acquired from Global 0, or first global variable found under Modules/Globals). Select the desired Input Channel from the drop-down list labeled "Input Chanel" below the description of the Averager. Name the Output Channels by typing the desired names in the Output Channels list. Make sure

to set Global 0 to a valid initial value in the Modules\Globals editor for this Calculation to work properly.

The “Previous Value” calculation provides a way to create memory of a channel’s previous value. This calculation takes one channel as input and creates one Output Channel. The Output Channel will contain the value of the Input Channel on the previous iteration. This calculation can be used to create logic that looks for changes in a Channel’s value, or which measures the rate of change of a Channel’s value. This calculation is a .vit, so it can be instantiated any number of times.

The “Demo Calculation” calculation is a simple demonstration calculation that can be copied to create more complex calculations for use in various systems. To configure it select two Input Channels from the drop-down list. This calculation will sum the two channels to arrive at a single Output Channel. Name the Output Channel it creates by typing a new channel name in the first line of the Output Channels list.

This Driver supports the Disabled on Startup?, Include Time?, and Include State? Which are discussed in section 7.5.1.

The Calculations Driver works in both MICAS-X for Windows and MICAS-X-RT.

7.5.5 Document

The Document driver supplies additional functionality to the Document Display. By including the Document Driver, the Document Number and Section Number channels are created, which allow programmatic control of the document being displayed in the Document Display. See section 7.8.2 for information on the Document Display.

This Driver supports the Disabled on Startup?, Include Time?, and Include State? Parameters, which are discussed in section 7.5.1

The Document Driver is supported only for MICAS-X for Windows.

7.5.6 Equations.vit

The Equations driver allows the user to create new channels based on previously available channels, or to set existing output (controller) channels to values based on equations made up of existing channels.

To insert a new equation, press the Insert button to the right of the “Equations” section on the left side of the screen. To delete an equation, press the delete button next to the corresponding equation. To edit an Equation, first highlight it in the Equations list by clicking on it. The Equation’s parameters will then be shown in the parameters below the Equations list.

The value calculated by an Equation can either be used as the value of a new channel defined by that Equation, or can be sent to an existing output (controller)

channel. Choose which functionality you wish by setting the Target parameter to False (New Channel) to create a new channel with the calculated value, or to True (Existing Channel) to send the calculated value to an existing controller channel.

When the Target parameter is set to New Channel, the Target Channel parameter is used to name the new channel. When the Target is set to Existing Channel, the Target Channel is a drop down list of all the existing controller channels. Make sure that a controller channel is only targeted by one Equation. E.g. the value of the Target Channel will be indeterminate if more than one location in MICAS-S is attempting to set its value.

Equations are made up of Channels, operations, functions, and constants. See section 17, Appendix D: Syntax for Equations for information on the operations and functions supported by the equation parser. When writing the equation, select the entirety of a channel name from the 'Channels' list in the center-right portion of the window, and press CTRL-C to copy this into the clipboard. Paste the name of the selected channel in the large "Equation" box by clicking inside the Equation box and pressing CTRL-V. You can use multiple functions and channels to create the desired equation.

If multiple Channel Lists have been created in the configuration that you are editing, a specific Channel List can be chosen with the Channel List control so that the Channels displayed are limited to those in that list. This sorting can make it easier to find specific channels when a configuration contains many channels.

To confirm that the syntax of the equation is correct, copy and paste your new equation into the "Test Equation" box located in the upper-right portion of the window. Select a channel value (the value selected in the Channel Value control will be the value for all channels in this equation, as it is only a test to confirm correct syntax). If functional, a result will appear in the "Result" indicator found to the right of the "Channel Value" input control.

If there is an error in your test of the equation, the "Test Equation Error" box in the lower-right-hand portion of the window will indicate the error status (present or not), give the error code, and describe the error.

Note that the LabVIEW functions on which Equations are built do not support the numbers Inf, -Inf, or NaN. Therefore these numbers should not be used as constants within the equation text, and if any input channels have these values, the output of the equation will be set to NaN.

Example: Using the Equations Driver

The Equations Driver can be used for many purposes. One of the simplest examples would be to create a new channel that displays an existing channel's data in new units. For example, the equation:

(9 * Chamber Temp © / 5) + 32

Could be used to create a new channel named Chamber Temp (F) which converts the channel Chamber Temp © to degrees Fahrenheit.

A second, more complex example is from an instrument that measures wind direction. The direction output was sent out as an angle from 0 to 540 degrees. When a measured quantity was plotted versus wind direction, some of the data that should have shown up in the 0 to 180 degree range instead appeared over in the 360 to 540 degree range. Although the Equations syntax does not support IF/THEN commands, the following equation was used to resolve the issue:

Angle -360 * (sign (Angle – 360) + 1) / 2 * abs (sign (Angle – 360))

The figure below shows how this Equation was configured in MICAS-X, including the use of the Test Equation to verify that the Angle of 365 was converted to 5.

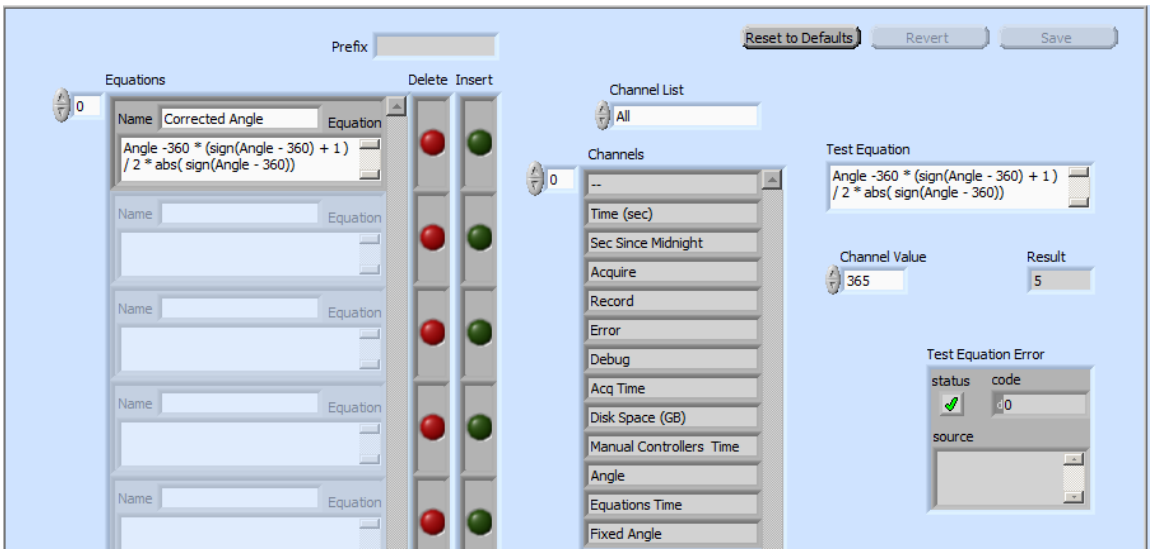


Figure 5.1 Configuring the Equations Driver to create a new channel that corrects an angle channel that overflows the 360 degree upper limit.

Since the “sign” function returns a -1, 0, or 1 depending on the sign of its argument, the (sign(Angle – 360) + 1) / 2 part of the equation calculates a value of 0, ½, or 1 depending on if the Angle channel has a value below 360, equal to 360, or above 360. The abs (sign (Angle – 360) section of the equation set the ½ value equal to 0 for the special case in which the Angle = 360.

An alternate resolution to this issue could have been to write a small Calculation VI that the Calculations Driver would use. Such a VI would have been trivial to write in LabVIEW, using a case structure to handle angles below and above 360 degrees. However, the Equation solution, though a bit convoluted, allows the resolution of the issue without having to program anything in LabVIEW.

A third example demonstrates how an Equation can be used to synchronize a MICAS-X Sequence to the computer clock. In this case, a Sequence needed to start on an even 5 minute time. E.g. it would be OK to start the Sequence at 1:00:00 or 15:25:00, but not at 15:26:00 or 15:25:01. The Equation used was

step(floor(Sec Since Midnight/300)*300 - floor(Sec Since Midnight))

This Equation created a new flag channel that has a value of 0 for all computer times except even 5 minute intervals. The first few steps of the relevant sequence were used to wait until this flag channel had a value of 1, then to proceed with the main part of the Sequence.

This Driver supports the Disabled on Startup?, Include Time?, and Include State? Parameters, which are discussed in section 7.5.1.

The Equation Driver works in both MICAS-X for Windows and MICAS-X-RT.

7.5.7 Manual.vi

The Manual Driver allows the user to create additional channels that are not directly associated with any hardware inputs or outputs. The values of these channels can be set manually by the user as the program runs, to indicate the values of experimental parameters, or can be used programmatically by Commands, Sequences, Triggers, or other mechanisms.

To create a manual channel, click the green “Insert” button. Delete a channel by pressing the corresponding red “Delete” button.

Name the channel by entering the desired text in the “Manual Controllers” field.

Use the “Initial Value” box to the right of the respective manual entry to define the value that the channel will have when the program first starts running.

This Driver supports the Initial Mode, Disabled on Startup?, Include Time?, and Include State? Parameters, which are discussed in section 7.5.1.

The Manual Driver works in both MICAS-X for Windows and MICAS-X-RT.

7.5.8 MICM.vit

The MICM Driver is an interface to the ICM (Instrument Communication Module). ICM is an open source project originally written by OCC and available on GitHub (https://github.com/NI-ALARM/ICM) It is a successor to OSDS, and allows one to communicate with a wide variety of instruments by creating a ICM configuration file rather than writing a new software driver. With ICM, one can read streaming data as well as query/response data, one can scale data through a variety of mechanisms, and one can write output data.

On the MICAS Parameters tab of the MICM Editor, use the ICM Config File control to specify a new or existing ICM configuration file. It is possible to use an existing ICM Configuration file that was created with the ICM Configuration Editor outside of MICAS-X, or create a completely new ICM configuration file.

Check the “Don’t Use Missing Data” box if you want the MICAS-X MICM Driver to ignore missing data. This can be useful for instruments that supply streaming data at irregular intervals. When this box is checked, the MICAS-X MICM Driver can be run in a “fast” acquisition loop (e.g. a loop faster than data is expected to arrive), and whenever no data is available, no error will be generated and the previous data will be retained. If this check box is not checked, any missing data will generate an error and will return NaN’s for the missing Channels.

Click on the ICM Parameters tab to access the ICM Configuration Editor, which will be loaded with the configuration file specified in the ICM Config File control. Refer to the ICM documentation for information on how to configure an ICM Configuration file.

The MICM Driver for MICAS-X is located in the Drivers folder, as are all MICAS-X Drivers. This MICM Driver calls into the ICM source code, which should be loaded separately onto the system. The ICM source code is normally loaded into C:\OCC\ICM. This directory has many sub-directories for the various plug-in modules of ICM, as well as a Documentation directory. The ICM user manual can be found in that Documentation directory

This Driver supports the Disabled on Startup?, Include Time?, and Include State? and Record Data Stream? parameters, which are discussed in section 7.5.1.

7.5.9 MOSDS.vit

The MOSDS Driver is an interface to the OSDS (OCC Streaming Data System) toolkit provided by Original Code Consulting. OSDS is a configuration-based system for

acquiring data from a wide variety of instruments. Please refer to the previously-issued OSDS document, called 'OSDS Description' for additional information about how to use OSDS. A copy of this document can be found under MICAS-X/Development Documentation (for Source Code users) or under OCC/MICAS-X Support/Documentation (for Executable Version users).

Note that OSDS Format Files are external to the MICAS-X configuration file. The Revert and Save buttons above relate only to the MICAS-X configuration file, which contains the file name of the OSDS format file and the other half dozen parameters on the left side of the configuration screen. The Revert, Save and Save As buttons on the right allow you to edit the OSDS Format parameters themselves and save them to an OSDS Format File.

To use an OSDS file in the MICAS-X MOSDS driver, click the browse button (a folder icon) above the box in the upper-left-hand corner of the control labeled "OSDS Format File". This box will show the file path of the selected OSDS format file. Below this, make sure to check or leave blank the OFF/ON box labeled "Don't Use Missing Data". When this option is selected, if the Driver encounters an error and does not get real data, it will not put NaN's into the data buffer, but will just leave the previous data in the buffer as the most recent. This can be useful for situations in which the device sends data at an unknown or variable rate. By running the Driver in a fast loop, this option will catch any data available but will not refresh the buffer with NaN's if no data is available.

The gray box in the center of the window, labeled "OSDS Format", will populate with information from the selected OSDS Format File. Many OSDS Format Files already exist, though they often need to be edited to work in any given situation. New files can also be created for talking to many different instruments. By default, OSDS Format Files are stored in the C:\MICAS-X\Resources\OSDS\OSDS Format Files director when using source code, or the C:\OCC\MICAS-X Support\ Resources\OSDS\OSDS Format Files directory when using the executable version of MICAS-X. Because OSDS Format Files are external to the MICAS-X configuration, care must be taken when moving a configuration that uses the MSODS Driver from one computer to another. In addition to moving the MICAS-X configuration file, be sure to also copy the required OSDS Format Files between computers.

This Driver supports the Disabled on Startup?, Include Time?, Include State? and Record Data Stream? parameters, as well as the "Check for Serial Ports" control, which are discussed in section 7.5.1.

7.5.10 NIDaqAD.vit

The NIDaqAD (National Instruments Data Acquisition, Analog to Digital) Driver acquires analog input voltages from a wide variety of National Instruments data acquisition devices. This Driver is a .vit, so it can be instantiated multiple times within a

MICAS-X configuration. One instance of this Driver is needed for each NI Device to be used. Any one NI Device can only have one NIDaqAD Driver associated with it.

To configure a device, enter the device name in the box in the upper-left-hand part of the window labeled “Device”. This parameter must match the name of the NI Daq Device as it appears in MAX (Measurement and Automation Explorer) and how it appears in the table in the lower right corner of the window. The table displays the NI Devices that are currently visible to your computer, as well as how many A/D channels each device has. You can copy and paste a device name from this table to the Device parameter.

The “Single-Ended/Differential” parameter directly under the “Device” parameter is applied to all the channels on the device, and determines the input configuration for the channels. Note that when this is changed, the chart in the lower right is updated to reflect the number of channels available in that mode on each detected device.

To name a given channel, type the desired label into the appropriate “Name” control, inside of the array. To the right of the “Name” parameter, select a type of scaling for the analog input channel. “None” is a 1 to 1 ratio that simply returns the input voltage. “Polynomial” allows the user to write simple equations using the subsequent polynomial coefficients (Offset, Linear, Quadratic and Cubic).

Various thermocouple types are supported using specific Scalings, including E, J, K and T (e.g. E Type TC). These work only with National Instruments Devices that directly support Thermocouple measurements, and are specifically designed for the NI 9213 CompactDAQ device. The “TC Mode” box in the lower-right-hand part of the window allows the user to select which Thermocouple mode should be used in the Analog to Digital conversion.

The “Lookup Table” options allow for complicated relationships between measured voltages and desired engineering units. The user can either fill these in manually (by inserting numbers in the ‘Raw Value’ and ‘Scaled Value’ fields in the controls on the left-hand side of the window), or import data from a text file. To do the latter, use a comma-delimited raw value followed by scaled value, with carriage returns after each pairing. To import, click the “Import Table X” button at the bottom of the window. Similarly, to store a user-created table, click the “Export Table X” button directly below the import button. Up to three Lookup Tables can be used in one instance of the NIDaqAD Driver.

The Channel parameter specifies the channel number on the NI Daq Device that will be used for the MICAS-X Channel. Note that the chart to the lower right shows how many channels are available for each of the NI Daq Devices visible to your computer. Channels are numbered beginning with 0. Hence if the device support 8 Channels, they will be numbered 0 to 7. The Range parameter specifies the voltage gain setting that will

be used for the Channel. Not all Daq Devices support all the input Ranges available in this parameter. Refer to the documentation for your NI Daq Device to determine which input ranges are supported.

Three “Acquisition Modes” are available for this Driver. Single Point is the default mode, and acquires a single measurement for each cycle of the Acquisition Loop that the Driver is in. Burst and Continuous are modes that allow for oversampling and averaging of the data for increased noise reduction. When either of these modes is used, the # of Samples to Avg and the Acq Rate (Hz) parameters must be configured.

In Burst mode, the Driver takes a number of samples each time the Acquisition Loop cycles. The number of samples times the sample rate should be smaller than the acquisition loop time. For example, with a 1000 ms (1 Hz) Acquisition Loop, one could set the # of Samples to Avg to 10 and the Acq Rate (Hz) to 20, so that the burst of acquisition would take 500 ms. This way the Acquisition Loop can keep up with its 1 Hz rate without any problem. Note that in Burst mode, all the acquisition happens when the Acquisition Loop polls the Driver. Thus in the above example, 500 ms of the loop time would be taken up by the single NIDaqAD Driver. If there are other Drivers in the same Acquisition Loop, it is possible that the loop may not keep up with its desired rate. For this reason, it may be a good idea to put the NIDaqAD Driver in its own Acquisition Loop.

In Continuous mode, the NI board is configured to acquire data continuously. The Acquisition Loop will then read and average a number of samples on each iteration. For this to work, the Acquisition Loop time (e.g. 1000 ms for a 1 Hz loop) must be configured to be the same as the acquisition time. To average 20 samples in a 1 Hz acquisition loop, you would then set the Acq Rate to 20 Hz. The advantage of Continuous mode is that the acquisition is clocked by the NI Device in the background, and the samples are merely read by the driver each time the acquisition loop cycles. E.g. the Acquisition Loop does not need to wait for the data to be ready. However, a possible disadvantage is that this mechanism ties the Acquisition Loop to the NI Device clock. It may therefore be advisable to put other Drivers in separate loops, particularly if they are tied to different hardware clocks. Also, with this configuration, it is important for the Acquisition Loop that contains this Driver to be configured as “Device Timed”.

The Voltage/Current parameters should be set according to the type of Daq Device being used. Most Daq Devices are Voltage Devices. Some, like the NI-9208, measure current from 0 to 20 mA. For these Devices, make this parameter True. Note that this Driver currently only supports Internal Shunt Resistors with Current Devices.

This Driver supports the Disabled on Startup?, Include Time?, and Include State? Parameters, which are discussed in section 7.5.1.

The NIDaqAD Driver works in only in MICAS-X for Windows and is not supported in MICAS-X-RT.

7.5.11 Sequences.vi

The Sequence Module allows the user to create sequences, or multi-step commands within MICAS-X. The Sequences Module can be used with or without the Sequences Driver or the Sequences Display. The Sequences Module Configuration is described in section 7.4.6.

If the Sequences Driver is included in the MICAS-X configuration, additional sequence functionality is provided. By including the Driver, the system has access to two new sets of Channels. One set of Channels is created from the Sequences names, with the text "SeqState" prepended. This is an input channel, and it will contain the number of the step that the sequence was on when the channel was recorded. If a Sequence is stopped (not running), the SeqState Channel for that Sequence will have a value of NaN. If a Sequence is Paused, the SeqState Channel for that Sequence will have value that is the negative of the Step it will resume at when the Sequence is Unpaused. The second set of Channels has the text "SeqStep" prepended to the Sequence name. These are output or Controller channels, and they can be used to turn Sequences on and off or Pause and Unpause the Sequence. By using these Channels, the Set "channel" command can be used instead of the Start Sequence command, which provides an additional level of flexibility to the control of sequences. Setting these Channels to NaN will stop a Sequence. Setting them to a positive number will start a Sequence at that Step. Setting them to a negative number will pause a Sequence so that it will resume at the absolute value of that number when an Unpause command is sent.

This Driver supports the Disabled on Startup?, Include Time?, and Include State? Parameters, which are discussed in section 7.5.1.

The Sequences Driver works in both MICAS-X for Windows MICAS-X-RT.

7.5.12 Timers.vi

The Timers Driver allows the user to create any number of channels that act as timers. In addition to the user-created timer channels, whenever the timer driver is included in MICAS-S, the timers "Run Time", "Acquire Time", and "Record Time" are always automatically included. (Do not add these channels additionally through the "Timers" section on the left-hand side of the window.)

To add a timer, click the green insert button. To delete a timer, click the respective red delete button.

Name a timer by entering a label into the "Timer" control. To determine the timer's starting value, select a number in the "Initial Value" control, located to the right of the "Timer" control. Timers can also be configured to count up or down using the "Up/Down" parameters, and to start either Paused or Running, using the "Paused" parameters.

The Timers Driver supports several Commands, which are listed and described in section 14, and which allow the Timers to be Paused, Unpaused, and change direction.

While a Timer is Paused, it retains its current value, and will resume counting at that value when it is Unpaused. In addition to pausing a Timer, it is also possible to Disable a Timer. To Disable a Timer, write the value “NaN” to it. While Disabled, a Timer will not count up or down.

The Rollover Value allows a Timer to be configured such that it automatically restarts when the specified value is reached. E.g. for a Count-Up Timer, a Rollover Value of 100 will cause the timer to be reset to 0 and restart counting up when a value of 100 has been reached. For a Count-Down Timer, the if the Rollover Value is non-zero and not NaN, the Counter will rollover when when it reaches 0, and will then reset to the Rollover Value. In both cases, there is no lost time or jitter when the Timer rolls over. If the Rollover Value is set to 0 or NaN, then the Timer will not have any Rollover behavior.

If the Include Rollover Channel? parameter is False: Do Not Include Rollover Channel, then no additional MICAS-X Channel will be created to contain the Rollover Value. The Rollover Values will still be present inside the Timers Driver, and if they are not zero and not NaN, they will be active for the Timers. However, it will not be possible to change these Rollover Values while MICAS-X is running. The Rollover Value defined in the Configuration File will be used as long as MICAS-X runs.

If the Include Rollover Channel? parameter is True: Include Rollover Channel in Data, then each Timer Channel will automatically have another Channel created in the Timers Driver as well. These Channels will have the same names as the Timers Channels, except with the text “ Rolllover” appended. In this case, the Rollover Value for each Timer can be changed while MICAS-X is running by writing to these Rollover Channels.

This Driver supports the Disabled on Startup?, Include Time?, and Include State? Parameters, which are discussed in section 7.5.1.

The Timers Driver works in only in both MICAS-X for Windows and MICAS-X-RT.

7.5.13 Triggers.vi

The functionality of Triggers in MICAS-X is described in section 7.4.8. The Triggers Module is always included in MICAS-X and can be used with or without the Triggers Driver. If the Triggers Driver is included, it provides additional functionality which enhances the utility of the Triggers. By including Triggers Driver in a MICAS-X configuration, two additional sets of channels are created.

One of the sets of channels created uses the text “TrigState” prepended to the each Trigger name. These channels log the state of the Trigger (0-False, 1-Warning, 2-True, 3-Alarm). The second set of channels uses the text “TrigThr” prepended to the Trigger name. These channels can be used to control the trigger threshold. These

threshold channels are output channels that can be changed while MICAS-X is running to alter how the triggers work. For example, a Trigger could be configured with an initial threshold of 10, such that if a channel exceeds that value, the Trigger will become True and execute a Command. Some time after the program has started, the Trigger's threshold could be set to Inf, which would effectively disable the Trigger, since the Channel's value can never exceed that threshold.

This Driver supports the Disabled on Startup?, Include Time?, and Include State?, parameters, which are discussed in section 7.5.1.

The Triggers Driver works in both MICAS-X for Windows and MICAS-X-RT.

7.5.14 Variables.vit

The Variables Driver allows the user to create additional channels that are not directly associated with any hardware inputs or outputs. The values of these channels can be set manually by the user as the program runs, to indicate the values of experimental parameters, or can be used programmatically by Commands, Sequences, Triggers, or other mechanisms. The Variables Driver is a newer version of the Manual Driver, with the main difference being that it is a .vit and can therefore be instantiated more than once in a given MICAS-X configuration.

To create a Variable, click the green "Insert" button. Delete a Variable by pressing the corresponding red "Delete" button.

Name the Variable by entering the desired text in the "Variables" field.

Use the "Initial Value" box to the right of the respective Variable entry to define the value that the channel will have when the program first starts running.

This Driver supports the Initial Mode, Disabled on Startup?, Include Time?, and Include State? Parameters, which are discussed in section 7.5.1.

The Variables Driver works in both MICAS-X for Windows and MICAS-X-RT.

7.6 Drivers Available for Additional Cost

7.6.1 2D Array.vit

The 2D Array Driver allows one to implement a two dimensional array of data that can be used programmatically throughout the MICAS-X program. Multiple 2D arrays can be created by using this driver repeatedly. This type of array can be used, for example, in sequences, to step through a set of process variables that are each an array of values and cannot be easily specified by an equation.

Enter a name for the array in the “Array Name” parameter. This name will be used to read and write the array values. In addition, it will be used as the basis for several other channels that the driver creates. For an array with the name “ArrayName”, there will also be a channels called “ArrayName Row Index” and “ArrayName Column Index”, which will be used to specify the location within the array to read or write data. The Channel “ArrayName Time” is similar to the Time channel every driver has, and records the time when the Array values were recorded, if the “Include Time” parameter is True. If the “Include Size” parameter is True, then another two channels named “ArrayName Row Size” and “Array Name Column Size” will be created that are read-only channels and which return the number of elements in the Array per row or column, respectively. (Note that the ArrayName Index channels can have values from 0 to ArrayName Size Of Dimension – 1.)

If “Allow Array to Grow” is not checked, the array size will be fixed by the number of entries in the Values list in the configuration. Trying to set either of the ArrayName Index channels past the ArrayName Size – 1 will result in an error. If this parameter is checked, then setting the ArrayName Index to a value greater than ArrayName Size and then writing a value to the Array channel will cause the Array to be extended as needed, with zeros placed in any new elements between the end of the old array and the new value.

Use the Delete and Insert buttons to remove and add rows of values in the array. The Import from File button can be used to read an array of data from a file. This file must be formatted with entries in the same row separated by commas and ones in separate rows separated by new lines. This is shown in the below format.

2,4,5

6,7,8

9,2,0

The Array Driver supports two Commands. ClearArray sets all the array elements to the value 0. It does not change the number of elements in the Array. ShowArray causes a window to be displayed which summarizes the current state of the Array. It shows the values in the Array, the Array Index, the Array Size, and several configuration parameters which affect how the Array works. Note that if the state or contents of the Array change, this display does not update to reflect those changes. It only contains a snapshot of the Array state when the display was opened.

This Driver supports the Disabled on Startup?, Include Time?, and Include State? Parameters, which are discussed in section 7.5.1.

7.6.2 Agilis.vit

The Agilis driver is used to control Agilis motorized mirror mounts. The Agilis controller must be in remote mode to accept commands from MICAS-X. It is put in this mode whenever the driver starts or is enabled, and is put in local mode when it ends of is disabled. The SetRemote and SetLocal commands can also be used to change mode. If include State? Is true, bit 8 will toggle so that in local mode the channel is 256 and in remote mode it is 0.

In Local mode, the manual controller can be used to move the axes, but MICAS-X will not be able to read values and it will not respond to MICAS.

The COM port must match the serial port the Agilis device is connected to. The serial ports in use can be found with the 'Check for Serial Ports" button. They can also be found with the windows device manager.

The delete and insert buttons to the right of the main panel allow the user to add new axes. Each axis must have a name entered in the name box. Channel is used only for the UC8 when it has a value of 1,2,3, or 4. For the UC2 use a value of 0 for the Channel. Axis # defines the axis to be controlled and can have a value of 1 or 2. Invert will invert which direction is considered positive for the axis. "# of Steps" defines the initial value of how many steps of the axis to move when given the "Move Steps" Command. Jog Speed determines how fast the axes will move and can have a value of -3, -2, -1, 1, 2, or 3.

If "Read Back Positions" is true, the position channel will be created for each axes. The Read position channels are of limited use since step movements are uncertain and vary in magnitude and direction.

This Driver supports the Disabled on Startup?, Include Time?, Include State?, and Initial Mode? Parameters, which are discussed in section 7.5.1.

7.6.3 Airmar.vit

The Airmar Driver allows acquisition of GPS, attitude, motion, wind, and meteorological data from the Airmar 200WX and related weather stations. The Airmar 200WX supports a large number of data sentences. Currently, the MICAS-X Driver for the Airmar supports the GGA, ZDA, VTG, ROT, MDA, MWV, XDRB, and XDRA sentences, which are the sentences that are enabled by default. Support for additional sentences could easily be added in the future if the need arises.

To configure the Airmar Driver, simply place an X in the box next to each sentence that you wish to acquire. Enter the number of the COM (serial) port that the Airmar is connected to on your computer, and enter the repetition rate in the Rate (sec) parameter. The repetition rate can have a resolution of 0.1 seconds. Note that this value is sent down to the Airmar during configuration, so that it streams data at this

rate. It is important to make sure that the MICAS-X Acquisition Loop that the Airmar Driver is assigned to is configured to accommodate this acquisition rate.

When the Airmar is powered on, it always begins communicating at 4800 baud. MICAS-X switches the Airmar automatically to 38400 baud, which reduces communication latency and allows for more data to be transmitted at a given loop rate. Note that it is possible to select data sentences that send more data than fits into a selected loop rate. Using 38400 baud helps alleviate this problem, but care must be taken when configuring the Airmar to avoid this communication limitation.

This Driver supports the Disabled on Startup?, Include Time?, Include State?, and Record Data Stream? Parameters, which are discussed in section 7.5.1.

7.6.4 Alicat.vit

The Alicat Driver is used to interface to Alicat Flow Controllers. One instance of this Driver can communicate to any number of Alicat Flow Controllers that are on the same serial port, using RS232 or RS485 serial communications. For Flow Controllers that are on separate, individual serial ports, multiple instances of this Driver can be used. Be sure to define a prefix for each instance when using multiple instances of any Driver or MICAS-X module.

Note that MICAS-X communicates to the Flow Controllers in polled mode. If a Flow Controller is set to Streaming mode, it must be reconfigured following the procedure in the Alicat manual.

Enter the serial port number to be used by the Flow Controller(s) in the COM Port field.

For each Flow Controller on that serial port, enter the eight parameters in the Flow Controllers array. The ID is a one letter code that acts as an address for the Flow Controllers. This comes set to “A” as a factory default. If multiple Flow Controllers are configured on a single serial port, each one must have a unique ID. This ID can be set via hyperterminal or another serial communication program, or in many cases, by using the interface panel on the front of the flow controller itself. Refer to the Flow Controller documentation for more information.

The Name field is used as a descriptive identifier within MICAS-X. This can be set to the same value as the ID, or, more usefully, to a short description of the function of the flow controller. This name is used as the root of all the channel names that the Driver creates for the Flow Controller.

Each Flow Controller can be configured to control in mass flow or volume flow. This configuration must be done externally from MICAS-X according to the procedure documented in the Alicat manual. Regardless of which mode the Flow Controller is using, it will report its flow both in mass flow and volume flow units. The Vol Flow Units

and Mass Flow Units fields allow the operator to enter these units as text, which is used in creating the relevant channel names.

The Full Scale Flow parameter is used to specify the nominal full scale flow of the Flow Controller. This must match the specification of the Flow Controller, so that MICAS-X can control the set points appropriately. The units of this parameter must match the units that the Flow Controller specifications show for its full scale flow.

The Mode parameter tells MICAS-X which flow mode the Flow Controller is configured to operate in. MICAS-X uses this information to create the name of the SetPoint channel and give it the correct units. This mode must match the mode that the Alicat is configured to run in, as described above.

The Initial SetPoint parameter tells MICAS-X what value to initialize the flow controller to when the program starts. The units of this value are those specified by the Mode control.

The Type parameter tells MICAS-X whether the Alicat is a Flow Controller, Pressure Controller, or Pressure Gauge. The data channels created for the Alicat depend on which type of device is selected. Also note that Alicat Flow Meters can be used with this Driver when they are defined as Flow Controllers, though a future upgrade to this Driver will clarify how Flow Meters are handled.

The Include Valve Position checkbox should only be marked for Alicat controllers that have the RVD option. This option allows the controller's internal valve position (0 to 100%) to be reported in the serial data stream as well as on the front panel. As of 20191022, only Pressure Controllers have been tested with this option.

The Include Gain Channel? parameter should be set True only if one needs to be able to set or read the PID Gains for the Alicat instruments. If this parameter is set to True, a new channel called "Alicat Gain Channel" is created. In addition, six new Commands are made available: ReadDGain, ReadIGain, ReadPGain, WriteDGain, WriteIGain, and WritePGain. These Commands make use of the Alicat Gain Channel to read and write the PID gains from and to the individual Alicat units. Refer to the Command documentation (Section 14) for more information on using these commands. This mechanism for reading and setting gains is less intuitive than if each Alicat had three additional channels for the three gains. However, due to the communication protocol used by the Alicats, continuously reading those channels in the Acquisition Loop could introduce significant latencies. The current solution, using Commands and the single Alicat Gain Channel, reduces the communication overhead significantly.

This Driver supports the Disabled on Startup?, Include Time?, Include State?, and Initial Mode parameters, which are discussed in section 7.5.1.

7.6.5 Aries Drive.vit

The Aries Drive Driver works with a Aries IPA Motion Controller from Parker Hannefin. It is designed to operate a single axis. It is assumed that the IPA controller is configured with the Parker Hannefin software before it is used with MICAS-X.

MICAS-X communicates with the motion controller via Ethernet. The IP Address field is used to specify the IP address of the IPA controller. The Initial Position (mm) and Initial Velocity (mm/min) parameters are used to set the values of the motor position and jog velocity. (Note that this is the velocity used when the motor is commanded to move. The motor is not commanded to move at initialization.) The Max Velocity (mm/min) is used to limit the commanded velocity. Any velocity set point sent to the Driver that is greater than this value will be limited to this value. The Counts/mm parameter specifies the calibration for the motor system from counts to mm. The Include Time? Parameter is used to determine if the Driver prepends its data channels with a time-stamp channel.

This Driver supports the Disabled on Startup?, Include Time?, and Include State? Parameters, which are discussed in section 7.5.1.

7.6.6 Controllers.vit

The Controllers Driver is used to create control loops, such that the value of a controller (output) Channel is determined by the value(s) of one or more other Channels. Several control algorithms are provided, as described below. Depending on which Controller Type is chosen, various parameters are required to fully define how the Controller will work. In the descriptions below, the Controller Type(s) that use each parameter are listed under each parameter's description.

The Controllers list on the left of the configuration screen shows the Controllers that are defined. To add a new controller, click on one of the Insert buttons to its right. Click on the associated Delete button to remove a Controller.

Once a Controller has been added, click on the Controller name in the Controllers list to highlight it and enable editing for that Controller. The name of the Controller can be edited in the SetPoint Channel parameter. This parameter also is used as the name of a new Channel created for the control loop, which is the SetPoint for the control loop. (Note that Followers are an exception and have no SetPoint Channel. This parameter is still used to define a name for the Follower definition, but that name is only used in the Controllers list to identify that Follower.)

Set the Controller Type parameter to the desired selection. The options are:
Mass Flow, Volume SP
Volume Flow, Mass SP
Thermostate

Simple Control
PID
Follower

The behaviors and associated parameters for each of these selections are described in the sections below.

Mass Flow, Volume SP

This Controller is designed to convert a Volume Flow SetPoint (from the operator or from another source) to a Mass Flow SetPoint that is then used to control a Mass Flow Controller. The conversion from specified Volume Flow to equivalent Mass Flow is done via the Ideal Gas Law, and hence requires knowledge of the temperature and pressure of the gas being controlled.

The Output Channel parameter must be set to the Mass Flow Controller SetPoint Channel. E.g. it is assumed that before this Controller is configured, a Mass Flow Controller (such as an Alicat) has already been added to MICAS-X. By assigning the SetPoint output Channel of the Mass Flow Controller as this Controller's Output Channel, this Controller will in turn set the Mass Flow Controller SetPoint according to the desired flow conversion.

Set the Pressure Channel to a MICAS-X Channel that contains the pressure of the gas being controlled, in mBar. If no pressure transducer is available, it may be possible to create a Manual Channel (section 7.5.7Error: Reference source not found) so that the operator can manually enter an appropriate pressure while the program is running. If a pressure transducer is being measured but has different units than mBar, an Equation Channel (section 7.5.6) can be used to convert the measured pressure into mBar.

Set the Temperature Channel to a MICAS-X Channel that contains the temperature of the gas being controlled, in degrees C. If no temperature transducer is available, it may be possible to create a Manual Channel (section 7.5.7)so that the operator can manually enter an appropriate temperature while the program is running. If a pressure transducer is being measured but has different units than degrees C, an Equation Channel (section7.5.6) can be used to convert the measured pressure into degrees C.

The Initial SetPoint parameter determines the value that the SetPoint Channel is initialized to when MICAS-X starts up. The Deadband parameter defines a minimum amount of change that a new calcaulation of the Output Channel must have relative to the current Output Channel value before the Output Channel will actually be written to. E.g. if the Deadband is non-zero, (for example, 100), then the Output Channel will only be updated if the calculated output is more than 100 different from its current value.

The Min Output and Max Output parameters allow one to specify minimum and maximum limits that the Output Channel can be set to. These parameters should normally be set to the limits that the associated Mass Flow Controller can accept. A common mistake is to leave the both of these parameters set to their default value of 0, which will cause the Controller to always output a value of 0.

The Slope and Offset parameters allow the Output Channel to be linearly scaled if needed. If a Mass Flow Controller is being used that accepts setpoints in units of mass flow, then a Slope of 1 and an Offset of 0 can be used. In this case, the Output Channel will have the same units as the Volume Flow SetPoint, except in mass instead of volume. The Slope and Offset can also be used to change the flow to other units. E.g. if the Volume SetPoint were in cc/sec, a Slope of 60 and Offset of 0 could be used to output a Mass Flow in cc/min. Another case would be when the Mass Flow Controller is controlled by a voltage, such that 0V is no flow and 10V is full scale flow. In this case, the Slope and Offset would be calculated to convert the Mass Flow from mass flow units to Volts. Another common mistake in configuring these Controllers is to leave the Slope and Offset parameters at their default values of 0, which will cause the Controller's Output Channel to always have a value of 0.

Volume Flow, Mass SP

This Controller is designed to convert a Mass Flow SetPoint (from the operator or from another source) to a Volume Flow SetPoint that is then used to control a Volume Flow Controller. The conversion from specified Mass Flow to equivalent Volume Flow is done via the Ideal Gas Law, and hence requires knowledge of the temperature and pressure of the gas being controlled.

The Output Channel parameter must be set to the Voume Flow Controller SetPoint Channel. E.g. it is assumed that before this Controller is configured, a Volume Flow Controller (such as an Alicat) has already been added to MICAS-X. By assigning the SetPoint output Channel of the Volume Flow Controller as this Controller's Output Channel, this Controller will in turn set the Volume Flow Controller SetPoint according to the desired flow conversion.

Set the Pressure Channel to a MICAS-X Channel that contains the pressure of the gas being controlled, in mBar. If no pressure transducer is available, it may be possible to create a Manual Channel (section$7.5.7$)so that the operator can manually enter an appropriate pressure while the program is running. If a pressure transducer is being measured but has different units than mBar, an Equation Channel (section $7.5.6$) can be used to convert the measured pressure into mBar.

Set the Temperature Channel to a MICAS-X Channel that contains the temperature of the gas being controlled, in degrees C. If no temperature transducer is available, it may be possible to create a Manual Channel (section$7.5.7$)so that the operator can manually enter an appropriate temperature while the program is running.

If a pressure transducer is being measured but has different units than degrees C, an Equation Channel (section7.5.6) can be used to convert the measured pressure into degrees C.

The Initial SetPoint parameter determines the value that the SetPoint Channel is initialized to when MICAS-X starts up. The Deadband parameter defines a minimum amount of change that a new calcaulation of the Output Channel must have relative to the current Output Channel value before the Output Channel will actually be written to. E.g. if the Deadband is non-zero, (for example, 100), then the Output Channel will only be updated if the calculated output is more than 100 different from its current value.

The Min Output and Max Output parameters allow one to specify minimum and maximum limits that the Output Channel can be set to. These parameters should normally be set to the limits that the associated Volume Flow Controller can accept. A common mistake is to leave the both of these parameters set to their default value of 0, which will cause the Controller to always output a value of 0.

The Slope and Offset parameters allow the Output Channel to be linearly scaled if needed. If a Volume Flow Controller is being used that accepts setpoints in units of volume flow, then a Slope of 1 and an Offset of 0 can be used. In this case, the Output Channel will have the same units as the Mass Flow SetPoint, except in volume instead of mass. The Slope and Offset can also be used to change the flow to other units. E.g. if the Mass SetPoint were in cc/sec, a Slope of 60 and Offset of 0 could be used to output a Volume Flow in cc/min. Another case would be when the Volume Flow Controller is controlled by a voltage, such that 0V is no flow and 10V is full scale flow. In this case, the Slope and Offset would be calculated to convert the Volume Flow from volume flow units to Volts. Another common mistake in configuring these Controllers is to leave the Slope and Offset parameters at their default values of 0, which will cause the Controller's Output Channel to always have a value of 0.

Thermostat

Thermostat control, also known as On/Off control, is the simples control loop algorithm. When the Process Variable is below the SetPoint, the control loop turns the output on. When the Process Variable is above the SetPoint, the control loop turns the output off. Hysteresis or Deadband adds a slight improvement to this simple algorithm.

The SetPoint Channel is the name of the control loop and of the new Channel created which allows the operator to set the desired SetPoint for this loop. The Output Channel parameter specifies the controller (output) Channel which is written to by this loop. Typically, this Channel will be a digital Channel, with just two states. Most often, those states will be 0 (off), and 1 (on). A digital output bit on an NI Daq Device would be a typical channel used for this purpose.

The Process Variable Channel parameter specifies the signal that is compared to the SetPoint Channel to determine if the Output Channel should be Off or On. The

Process Variable Channel and the SetPoint Channel should have the same physical units. If necessary, an Equation Channel (section7. 5. 6) can be used to convert the physical units of another Channel in order to get appropriate units for the Process Variable.

The Initial SP parameter specifies the value that the SetPoint Channel will be given when MICAS-X first starts.

The Off Value and On Value determine the values that the Output Channel will be set to when the control loop wants to turn the output off or on. Normally the Off value should be set to 0 and the On value should be set to 1. This will turn a digital bit True when the loop is On, and False when the loop is Off. The direction of the control loop can be reversed by exchanging these values. For example, if the Off Value is 1 and the On value is 0, then the Output Channel would be set to 1 whenever the Process Variable (such as a temperature) was below the SetPoint Channel value. This type of loop could be used as a hating loop as opposed to the cooling loop that the opposite settings would create.

Additionally, other values of the Off Value and On Value could be used to set an analog output channel to two distinct values depending on the state of the control loop. Using values of 0 and 5, for example, could allow one to use an analog output voltage channel as a surrogate for a digital output channel.

One common error in setting up these loops is to leave both the Off Value and the On Value at their defaults of 0. This will cause the Output Channel to never change, regardless of what the control loop is trying to do.

The final parameter for this type of controller is the Deadband. This parameter adds hysteresis to the thermostatic control. When this parameter is 0, the state of the Output Channel will change every time the Process Variable passes the SetPoint Channel value. When the Process Variable is noisy, this can cause rapid changes in the Output Channel, which may not be desirable. If the hysteresis is non-zero, noise immunity is provided. As an example, consider a temperature control loop with a SetPoint of 50 degrees and a hysteresis of 2 degrees. This loop will turn the Output Channel On when the temperature rises above 50 degrees. At that point, the threshold for the loop is internally changed from 50 degrees to 48 degrees, the SetPoint minus the Hysteresis (or Deadband). Thus the Output Channel will not turn Off until the temperature drops below 48 degrees.

Simple Controller

The Simple Controller option implements a semi-proportional control algorithm. The SetPoint Channel is used to name the setpoint channel for the Controller. The Output Channel is the controller (output) Channel that this Controller sets. The Process Variable Channel parameter specifies the Channel that is used to compare to the SetPoint Channel to determine how the Controller should react. The Initial SP parameter determines the value of the SetPoint Channel when MICAS-X starts.

The Simple Control algorithm compares the SetPoint Channel to the Process Variable Channel. If the Process Variable is below the SetPoint Channel, the algorithm adds the Step Size (V) to the Output Channel. If the Process Variable is above the SetPoint Channel, the algorithm subtracts the Step Size (V) from the Output Channel. Thus the Step Size defines fixed amount that the algorithm will move the output on each iteration. This value can be made negative to invert the direction of the control loop. Althought the name of the Step Size (V) parameter implies that the units of this parameter are in Volts (which often they are), they are actually in whatever units the Output Channel has.

The Deadband parameter is used to specify a “delta” or a minimum amount that the Process Variable must differ from the SetPoint Channel before the Output will be changed.

The Initial SetPoint (V) defines the initial value of the Output Channel when MICAS-X first starts up. The Min Output and Max Output parameters define the limits that the algorithm can drive the Output Channel to. These parameters keep the control loop from running away when it saturates.

The final parameter is Start-up, which has the options Disabled (-1), Manual (0) and Control Loop (1). As for PID Controllers, the Simple Controllers have three modes, Disabled, Manual and Control Loop. When set to Control Loop, the operation is described as above. When set to Manual, the Simple Controller’s SetPoint Channel has the same units as the Output Channel, rather than as the Process Variable Channel, and the SetPoint Channel value is sent directly to the Output Channel. When switching between Manual and Control Loop, the program automatically converts the SetPoint Channel’s value in such a way as to hold the loop at the current value. E.g. when switching from Control Loop to Manual, the current value of the Output Channel is automatically written to the SetPoint Channel. When switching from Manual to Control Loop, the current value of the Process Variable Channel is automatically written to the SetPoint Channel. When in Disabled Mode, the Controllers Driver does not write to the Output Channel at all. This mode allows for multiple control loops to be defined using the same Output Channel, as long as only one of the loops is enabled.

To change the Mode for a Simple Controller, two options are available. If the Simple Controller is configured to appear on the Control tab of MICAS-X, a Mode switch will automatically be placed to its right. A second way to change the mode is use the Set Command to set the MICAX-X Channel that has the same name as the SetPoint Channel, but with the word “Mode” preceding it.

PID

The PID Controller option implements a standard Proportional-Integral-Differential control algorithm. The SetPoint Channel is used to name the setpoint channel for the Controller. The Output Channel is the controller (output) Channel that

this Controller sets. The Process Variable Channel parameter specifies the Channel that is used to compare to the SetPoint Channel to determine how the Controller should react. The Initial SP parameter determines the value of the SetPoint Channel when MICAS-X starts.

The Kc, Ti(min) and Td(min) parameters specify the Proportional Gain, Integral Gain (in units of Kc * minutes) and the Derivative Gain (in units of Kc / minutes). The Min Output and Max Output parameters impose limits on how far the PID algorithm can set the Output Channel. These parameters keep the control loop from running away when it saturates.

The Slope and Offset parameters are used to scale the Output Channel in Manual mode, and the inverse of this linear relationship is used to calculate a SetPoint when the Controller switches from Control Loop to Manual mode. E.g. these parameters allow the Output Channel and the SetPoint Channel in Manual mode to have different units. As an example, assume that a pressure is being controlled by changing the value of a flow. The SetPoint Channel (in Control Loop Mode) would have units of pressure. The Process Variable Channel would have units of Flow. But assuming the flow controller required a voltage input, the Output Channel would have units of Volts. In Manual mode, the Slope and Offset would be used to calculate the appropriate Voltage for a Manual SetPoint in units of Flow. When the loop is changed from Control Loop mode to Manual mode, the inverse relationship would be used to calculate the Setpoint in Flow based on the current value of the Output Channel in Volts.

The final parameter is Start-up, which has the options Disabled (-1), Manual (0) and Control Loop (1). When set to Control Loop, the loop operates using the full PID logic. When set to Manual, the SetPoint Channel directly controls the Output Channel. When switching between Manual and Control Loop, the program automatically converts the SetPoint Channel's value in such a way as to hold the loop at the current value. E.g. when switching from Control Loop to Manual, the current value of the Output Channel, scaled by the Slope and Offset, is automatically written to the SetPoint Channel. When switching from Manual to Control Loop, the current value of the Process Variable Channel is automatically written to the SetPoint Channel. When in Disabled Mode, the Controllers Driver does not write to the Output Channel at all. This mode allows for multiple control loops to be defined using the same Output Channel, as long as only one of the loops is enabled.

To change the Mode for a PID Controller, two options are available. If the PID Controller is configured to appear on the Control tab of MICAS-X, a Mode switch will automatically be placed to its right. A second way to change the mode is use the Set Command to set the MICAX-X Channel that has the same name as the SetPoint Channel, but with the word "Mode" preceding it.

Follower

Followers are a unique type of Controller. Followers have no SetPoint Channel. The SetPoint Channel parameter is renamed Controller Name for Followers, and is used only to identify the Controller. A Follower is intended as a way to broadcast a Channel's value through an Analog Output or some other output Channel. E.g. the Output Channel for a Follower is linearly proportional to the current value of the Process Variable Channel. The Slope and Offset parameters specify the conversion to use from the Process Variable Channel to the Output Channel. To calculate the Output Channel, the Process Variable Channel's value is first multiplied by the Slope, then the Offset is added. Hence the Offset has the same units as the Output Channel, and the Slope has units of the Output Channel units divided by the Process Variable Channel units.

The Min Output and Max Output parameters are used to limit the range that the Output Channel can be set to. For an analog output Voltage channel, these parameters would normally be set to the lower and upper ranges of the analog output channel.

This Driver supports the Intial Mode, Disabled on Startup?, Include Time?, and Include State? Parameters, which are discussed in section 7.5.1.

The Controllers Driver works in both MICAS-X for Windows and MICAS-X-RT.

7.6.7 Expressions.vit

The Expressions driver is very similar to the Equations driver, but executes much faster and has a richer syntax. It allows the user to create new channels based on previously available channels, or to set existing output (controller) channels to values based on equations made up of existing channels.

To insert a new expression, press the green Insert button to the right of the "Equations" section on the left side of the screen. To delete an expression, press the red Delete button next to the corresponding equation. To edit an expression, first highlight it in the Equations list by clicking on it. The Equation's parameters will then be shown in the parameters box below the Equations list.

The value calculated by an Equation can either be used as the value of a new channel defined by that Equation, or can be sent to an existing output (controller) channel. Choose which functionality you wish by setting the Target parameter to False (New Channel) to create a new channel with the calculated value, or to True (Existing Channel) to send the calculated value to an existing controller channel.

When the Target parameter is set to New Channel, the Target Channel parameter is used to name the new channel. When the Target is set to Existing Channel, the Target Channel is a drop down list of all the existing controller channels. Make sure that a controller channel is only targeted by one Expression. E.g. the value of the Target

Channel will be indeterminate if more than one location in MICAS-X is attempting to set its value.

Expressions are made up of Channels, operations, functions, and constants. The Expressions Driver is based on the Expression Parser LabVIEW project from GPower. Refer to the document “Expression_Parser_User_Guide.pdf” in the Documents directory for information on the syntax used in Expressions.

When writing the expression, select the entirety of a channel name from the ‘Channels’ list in the center of the window, and press CTRL-C to copy this into the clipboard. Paste the name of the selected channel in the large “Expression” box by clicking inside the Expression box and pressing CTRL-V. You can use multiple functions and channels to create the desired Expression.

If multiple Channel Lists have been created in the configuration that you are editing, a specific Channel List can be chosen with the Channel List control so that the Channels displayed are limited to those in that list. This sorting can make it easier to find specific channels when a configuration contains many channels.

To confirm that the syntax of the Expression is correct, copy and paste your new Expression into the “Test Expression” box located in the upper-right portion of the window. The channels that are used in the Expression will populate the “Channels in Test” list. If they do not one or more is not spelled and capitalized exactly as in the “Channels” list. To the right of each channel, a test value can be selected. If functional, a result will appear in the “Result” indicator found to the right of the “Test Expression” input control.

If there is an error in your test of the equation, the “Test Expression Error” box in the lower-right-hand portion of the window will indicate the error status (present or not), give the error code, and describe the error.

This Driver supports the Disabled on Startup?, Include Time?, and Include State? Parameters, which are discussed in section 7.5.1.

7.6.8 File Reader.vit

The File Reader driver provides data to MICAS-X after reading in a file of previously recorded or created data. It can be used to simulate data or playback data that was acquired earlier. It reads in a .txt or .csv file, and sends out a row of data per acquisition loop. The file should contain one or more columns of numeric data, with one or more rows. Each column can be used as a channel, though extra columns at the end of a row can be ignored. Values should be separated by commas.

The file can have an optional header, which will be ignored when MICAS sends data during acquisition. The header will usually define the channels, and will be displayed in the “Data File Preview” to help with configuring the channels.

Use the “File to Read” controller to select which file you want to read into MICAS-X. If there is a header on the file defining channel names, use the “Import Channel Names”. The channels can also be added manually by adding a new channel or changing existing names. Each channel has three parameters. The first is the “Value to Use if No Data Available” this is the value that will be assigned to the channel if there is not data in that column. The “Noise to Add” parameter defines how much randomness to add to the values that are assigned to the channel. This helps a generated file simulate reality better. The “Use NaN or Value When Disabled” parameter defines whether or not the channel will be assigned the default value or Not a Number. The Repeat? Parameter defines whether or not the data in the file will be looped or if it will be used only once.

This driver supports the Disabled on Startup? Include Time? And Include State? Parameters.

7.6.9 GRIMM 1p109 OPC.vit

The GRIMM 1p109 OPC Driver acquires data from the model 1.109 Optical Particle Counter made by GRIMM. The “COM Port” parameter defines which serial port will be used by the scale. The “Baud Rate” parameter is used to specify the Baud Rate which the OPC is currently configured to use. The default for this value is 9600.

This Driver supports the Disabled on Startup?, Include Time?, Include State?, and Record Data Stream? Parameters, which are discussed in section 7.5.1.

7.6.9.1 Look Up Table.vit

The Look Up Table Driver is used to scale channel values via interpolation of a look-up table. This allows channels to be scaled in ways that are non-linear or cannot be approximated easily by an equation. The Look Up Table Driver uses the value of an existing channel, interpolates a new value from the associated look-up table for that channel, and then write the new value to a new channel.

There are two main components to the Look Up Table configuration, the Channels list and the Look Up Tables. The Channels list defines which Channels will be used as input values and will use the look-up tables. The Look Up Tables are one or more tables of data to use for interpolation. Thus different Channels can be configured to all use the same Look Up Table, or they can each have their own Table, as needed.

Start configuring this Driver by defining the Look Up Tables. Press the Insert button to the right of the Look Up Tables list to add a new table. (The Delete buttons can be used to remove an unwanted table from the list.) When the new entry is highlighted in the list, edit the name in the Look Up Table Name field. Keep that entry highlighted in the list, and add the table data. This can be done either by importing a text or .csv file that contains the data (by pressing the Import Table button), or by entering the values

directly in the Look Up Table arrays on the right. A table must have as many “raw” values as it has “scaled” values. The text or .csv file should contain two columns of data values, separated by commas, with no headers.

A Look Up Table can be exported by pressing the Export Table button. This will create a text file with two columns of data, separated by commas.

Once your table(s) are created, define the channels that will be scaled with this Driver. In the Channels array on the left, press the Insert button to its right to add a new entry at the desired location. (The Delte buttons can be used to remove an unwanted channel definition from the list.) Click on the Source Channel parameter to see a list of all the existing Channels in MICAS-X. Select the one which you want scaled. Enter a name for the resulting Scaled Channel in the Target Channel parameter. Finally, click on the Look Up Table parameter to select which Look Up Table you wish to use for interpolation of the Source Channel.

This Driver supports the Disabled on Startup?, Include Time?, and Include State?, Parameters, which are discussed in section 7. 5. 1.

7.6.9.2 M-Link.vit

This Driver is used in conjunction with the M-Link Server Instrument 7.7.4 to enable one instance of MICAS-X to communicate with and control another instance. The two instances must be on a network such that they are visible to each other. The “client” instance, running this Driver, must know the IP address of the “server” instance, as well as the name of the configuration file used by the “server” instance. A current copy of this configuration file must be present on the “client” computer in the support directory, so that when the “client” MICAS-X is run, it can read that configuration file to know the Channels, Commands, Sequences, etc., that are available on the “server” instance.

This Driver supports the Disabled on Startup?, Include Time?, and Include State?, Parameters, which are discussed in section 7. 5. 1.

7.6.9.3 MCCDaqAD.vit

The MCCDaqAD (Measurement Computing Data Acquisition, Analog to Digital) Driver acquires analog input voltages from a wide variety of Measurement Computing multifunction data acquisition devices. This Driver is a .vit, so it can be instantiated multiple times within a MICAS-X configuration. One instance of this Driver is needed for each MCC Device to be used. Any one MCC Device can only have one MCCDaqAD Driver associated with it.

To configure a device, enter the device name in the box in the upper-left-hand part of the window labeled “Device”. This parameter must match the name of the MCC Daq Device as it appears in the MCC Instacal program. Single-Ended vs. Differential input configurations must be done in the Instacal program, not in the MICAS-X configuration.

To name a given channel, type the desired label into the appropriate “Name” control, inside of the array. To the right of the “Name” parameter, select a type of scaling for the analog input channel. “None” is a 1 to 1 ratio that simply returns the input voltage. “Polynomial” allows the user to write simple equations using the subsequent polynomial coefficients (Offset, Linear, Quadratic and Cubic).

The “Lookup Table” options allow for complicated relationships between measured voltages and desired engineering units. The user can either fill these in manually (by inserting numbers in the ‘Raw Value’ and ‘Scaled Value’ fields in the controls on the left-hand side of the window), or import data from a text file. To do the latter, use a comma-delimited raw value followed by scaled value, with carriage returns after each pairing. To import, click the “Import Table X” button at the bottom of the window. Similarly, to store a user-created table, click the “Export Table X” button directly below the import button. Up to three Lookup Tables can be used in one instance of the MCCDaqAD Driver.

The Channel parameter specifies the channel number on the MCC Daq Device that will be used for the MICAS-X Channel. Channels are numbered beginning with 0. Hence if the device support 8 Channels, they will be numbered 0 to 7. The Range parameter specifies the voltage gain setting that will be used for the Channel. Not all MCC Daq Devices support all the input Ranges available in this parameter. Refer to the documentation for your MCC Daq Device to determine which input ranges are supported.

This Driver supports the Disabled on Startup?, Include Time?, and Include State? Parameters, which are discussed in section 7.5.1.

The MCCDaqAD Driver works in only in MICAS-X for Windows and is not supported in MICAS-X-RT.

7.6.10MCCDaqDA.vit

The MCCDaqDA (Measurement Computing Data Acquisition Digital to Analog) Driver is used to set output voltages using the DA channels on Measurement Computing multifunction Daq devices. The “Device” parameter must match the name of the MCC Daq Device as it appears in the Measurement Computing Instacal program.

To create a new channel, type the desired label into the “Name” control. The Slope and Offset parameters can be used to apply a linear scaling to the data being passed to the output channel. Normally, set the Slope to 1 and the Offset to 0, and the output voltage will be equal to the value of the channel in MICAS-X. However, the Slope and Offset can be used to map engineering units of a MICAS-X channel into voltage. For example, if the channel was named Flow (cc/s) and a setting of 20 cc/s required a setting of 5 V, then a Slope of .25 and an Offset of 0 would map the channel’s value of 20 to a voltage output of 5V.

The “Initial Value” is the value, in engineering units, used as a SetPoint when the program first starts. The “Channel” parameter specifies the number of the analog output channel to be used for the MICAS-X channel. Note that the chart to the lower left shows how many analog output channels are available for each device that is detected.

The “Range” parameter defines the output voltage range for an analog output channel in the MICAS-X system. Note that not all MCC A/D boards support all possible output ranges available in the selection.

The red delete and green insert buttons to the right of the window can be used to delete and create new channels, respectively.

This Driver supports the Disabled on Startup?, Include Time?, Include State?, and Initial Mode parameters, which are discussed in section 7.5.1.

The MCCDaqDA Driver works in only in MICAS-X for Windows and is not supported in MICAS-X-RT.

7.6.11MCCDaqDI.vit

The MCCDaqDI (Measurement Computing Data Acquisition Digital Input) is used to read digital inputs as channels using Measurement Computing multifunction Daq hardware. The “Device” parameter must match the name of the MCC Daq Device as it appears in the Instacal program.

To create a new digital output Channel used within the MICAS-X program, type the desired Channel name into the “Name” control. The “Channel” parameter defines the Channel number of a specific port on the MCC Digital IO Device which is to be used for the Digital Output bit. The “Port” parameter defines the Port on the MCC Digital IO Device. Refer to MCC documentation for information on the naming of the ports. The Invert parameter can be used to invert the logic level of the MICAS-X Channel relative to the logic level actually read by the MCC hardware.

The red delete and green insert buttons to the right of the window can be used to delete and create new channels, respectively.

This Driver supports the Disabled on Startup?, Include Time?, and Include State? Parameters, which are discussed in section 7.5.1.

The MCCDaqDI Driver works in only in MICAS-X for Windows and is not supported in MICAS-X-RT.

7.6.12MCCDaqDO.vit

The MCCDaqDO (Measurement Computing Data Acquisition Digital Output) is used to create digital output channels using Measurement Computing Daq hardware.

The “Device” parameter must match the name of the MCC Daq Device as it appears in the Instacal program.

To create a new digital output Channel used within the MICAS-X program, type the desired Channel name into the “Name” control. The “Channel” parameter defines the Channel or bit number on the MCC Digital IO Device which is used for the Digital Output channel. The “Port” parameter defines the Port number on the MCC Digital IO Device to which the bit belongs. Refer to MCC documentation for more information on the name conventions of the ports.

The “Initial Value” parameter defines which initial value that the digital output will be set to when the MICAS-X program starts. The Invert parameter can be used to invert the logic level of the MCC Digital Output bit relative to the MICAS-X Channel.

The red delete and green insert buttons to the right of the window can be used to delete and create new channels, respectively.

This Driver supports the Disabled on Startup?, Include Time?, Include State?, and Initial Mode parameters, which are discussed in section 7.5.1.

The MCCDaqDO Driver works in only in MICAS-X for Windows and is not supported in MICAS-X-RT.

7.6.13MCCDaqTC.vit

The MCCDaqTC (Measurement Computing Data Acquisition Thermocouple) Driver acquires temperatature data from Measurement Computing thermocouple devices. This Driver is a .vit, so it can be instantiated multiple times within a MICAS-X configuration. One instance of this Driver is needed for each MCC Device to be used. Any one MCC Device can only have one MCCDaqTC Driver associated with it.

To configure a device, enter the device name in the box in the upper-left-hand part of the window labeled “Device”. This parameter must match the name of the MCC Daq Device as it appears in the MCC Instacal program. The Instacal program must also be used to configure the type of thermocouple being used with each channel of the device.

To name a given channel, type the desired label into the appropriate “Name” control, inside of the array. To the right of the “Name” parameter, select the Units to use with the channel.

The Channel parameter specifies the channel number on the MCC Daq Device that will be used for the MICAS-X Channel. Channels are numbered beginning with 0. Hence if the device support 8 Channels, they will be numbered 0 to 7.

This Driver supports the Disabled on Startup?, Include Time?, and Include State? Parameters, which are discussed in section 7.5.1.

The MCCDaqTC Driver works in only in MICAS-X for Windows and is not supported in MICAS-X-RT.

7.6.14MICAS-X-RT.vit

The MICAS-X-RT Driver provides an interface for MICAS-X on Windows to MICAS-XRT on a Real-Time target. To configure this driver, enter the IP Address of the Real-Time target in the RT IP Address parameter. The Config File parameter must contain the configuration file that MICAS-XRT will run. Finally, the Sync Clock Every parameter can be used to determine how often the clock on the Real Time target is synchronized to the Windows computer clock whenever MICAS-X on Windows is running and is connected to the Real Time target. Whenever the time is synchronized between the two systems, a message is logged indicating the old and new RT times.

MICAS-X-RT has several data channels and commands that help interface to MICAS-XRT on a Real-Time platform. A channel named “MICAS-XRT Connected” is created which indicates the status of the three network communication methods used with the MICAS-X-RT Driver. A Network Stream connection from MICAS-X-RT to MICAS-XRT on the Real-Time platform allows commands to be sent from the Windows program to the RT program. If this stream is connected, a value of 1 is OR’ed into the value of this channel. Another Network Stream is used to send messages from the RT platform to the MICAS-X-RT Driver. If this stream is connected, a value of 2 is OR’ed into the value of this channel. Finally, a Current Value Table (CVT) is used on both platforms to hold the channel values. The CCC (CVT Client Communication) mechanism is used to transfer this CVT from the Real-Time platform to the MICAS-X-RT Driver. If the CCC is connected, a value of 4 is OR’ed into the value of this channel. Thus, when all three network connections are operating correctly, this channel should have a value of 7.

Two custom Commands are also supported by this Driver. The SaveConfig Command tells the RT system to save the current configuration as the “Auto-Load.ini” file, so that it will be used automatically whenever the RT system is booted. The ClearConfig Command does the opposite. It deletes the “Auto-Load.ini” file on the RT system so that MICAS-XRT will not automatically start running a configuration when the RT system boots up.

The MICAS-X-RT Driver works only in MICAS-X for Windows and is not supported in MICAS-X-RT.

7.6.15Modbus.vit

The Modbus Driver can be used to interface to a wide variety of instruments which support the Modbus communication protocol. Both serial and Ethernet Modbus are supported.

Enter the IP address or DNS name of the Modbus instrument in the “Modbus Server Name” parameter when using Ethernet Modbus. Also enter the Ethernet port used in the “Port” parameter. For Modbus over Ethernet, the port is often 502. Set the “Serial/Ethernet” parameter True (Ethernet) to enable Modbus over Ethernet.

With the Serial/Ethernet parameter False (Serial), you can communicate to Modbus devices which use a serial interface. In this case, set the “Port” parameter to the serial port connected to the Modbus instrument. Also set the “Baud Rate”, “Serial Type”, and “Unit ID” parameters as appropriate for the instrument. (Refer to the instrument manual for more information on the proper values of these parameters.)

On the Input Channels tab, enter the channels that are to be read from the Modbus instrument. These channels should correspond to Input Registers on the Modbus Instrument. Enter a descriptive name for the Channel in the “Name” parameter. Select the appropriate data type with the “Data Type” parameter. Enter the address of the Modbus channel in the Address parameter. The “Offset” and “Linear” parameters can be used to apply a linear scaling to the data so that the Channel in MICAS-X has can have different units than the Channel in the Modbus instrument. If the “Linear” parameter is 0, both the Offset and Linear parameters are ignored.

Use the Delete and Insert buttons to delete or add Channels to the list of Channels being edited.

On the Output Channels tab, enter the channels that are to be sent to the Modbus instrument. These channels should correspond to Holding Registers on the Modbus Instrument. Enter a descriptive name for the Channel in the “Name” parameter. Select the appropriate data type with the “Data Type” parameter. Enter the address of the Modbus channel in the Address parameter. The “Offset” and “Linear” parameters can be used to apply a linear scaling to the data so that the Channel in MICAS-X has can have different units than the Channel in the Modbus instrument. If the “Linear” parameter is 0, both the Offset and Linear parameters are ignored.

If the “Read Back” parameter is checked, then the associated Channel will also be read from the Modbus instrument on every acquisition cycle, as well as written to the Modbus instrument whenever its value in MICAS-X changes. This is useful for Channels that can be changed by either MICAS-X or by the Modbus instrument itself. By enabling “Read Back”, the Channel in MICAS-X is kept in sync with the Channel in the Modbus instrument regardless of which system changed the value of the Channel. If a Holding Register channel is not ever changed by the Modbus instrument and is only ever written to by MICAS-X, do not check the Read Back parameter, since doing so will only add more communication overhead to your system.

The Initial Value channel is used when the “Initial Mode” parameter is set to “Use Configuration Initial Value”, or True. In that case, the Output Channels will be set

to the values specified by these parameters when the Driver is Started. Use 1 for True and 0 for False for Boolean Output Channels.

Use the Delete and Insert buttons to delete or add Channels to the list of Channels being edited.

Modbus documentation from National Instruments states: “Per Modbus convention, a register address is always one less than the register name. This is similar to the first element of an array being element 0. The Modbus LabVIEW Library requires register addresses, not register names.” Dependingon the documentation that came with your Modbus instrument, you may need to subtract 1 from the documented address to communicate with it from MICAS-X. Also note that an older standard for Modbus documentation prepended the address with a Modbus function code. Code 3 was for read-only addresses and Code 4 was for read/write addresses. Hence an address of 31121 would indicate a request to read a read-only address at 1121, which would be the register 1120.

This Driver supports the Initial Mode, Disabled on Startup?, Include Time?, and Include State? Parameters, which are discussed in section 7.5.1.

7.6.16MyRIO Daq.vit

The MyRIO Daq driver interfaces to the Daq signals on the myRIO platform.

There are three types of channels in this driver: Analog Input, Analog Output and Digital IO. In order to include a channel, first find the one with the correct description in the correct tab then select the include check box. Give the channel a name in the Name field. The analog channels all have 4 scaling coefficients. The analog output channels also have an initial value parameter which defines the value they will be at start-up. The digital IO channels have 3 options. The direction option defines if this is an input or output channel. Initial values defines if the channel is initially low(false) or high(true). The invert parameter will invert the signal going to or from the channel.

This driver supports the Disabled on Startup?, Include Time?, Include State?, and Initial Mode? Parameters that are defined in 5.5.1.

The MyRIO Daq driver can only be used with MICAS-X-RT.

7.6.17NIDaqBridge.vit

The NIDaqBridge (National Instruments Data Acquisition Bridge Device) Driver is used to acquire data from a sub-set of NI Daq devices which are specifically designed for measuring bridge sensors, such as the NI-9237. These NI devices can measure a wide range of quantities, such as Force, Pressure, and Torque, and can scale the measurements by linear or polynomial equations or a lookup table. For the initial

release of NIDaqBridge, a subset of functionality is provided. This Driver measures force bridge sensors, using a linear transformation for scaling. Future versions of this Driver will add additional sensor types and scalings. Note that this Driver has only been tested with the NI-9237, though it should work well with other NI bridge devices as well.

In addition, this Driver is written to run in a 1000 ms acquisition loop, since it returns a single, averaged value once a second. This limitation is partly due to how the NI-9237 hardware works, since it has an internal clock that can only run at 31 discrete frequencies, which are defined by 50,000/n where n = 1 to 31. For this Driver, the device is configured to acquire 10,000 points at 10,000 Hz, which it then averages together to arrive at a single value once a second. If needed, future versions of this Driver could include additional acquisition frequencies and/or averaging, though constrained by the timing available on the device.

The “Device” parameter must match the name of the NI Daq Device as it appears in MAX (Measurement of Automation Explorer).

The “Channels’ array allows individual configuration of each of the channels of the device. The “Name” field defines the name given to the associated Channel in MICAS-X. “Channel #” defines which channel on the device is being configured. “Channel #” starts at 0.

“Minimum Force” and “Maximum Force” are used by the device along with the scaling information to determine the voltage range which the channel is configured to use. These values must be specified in the same units as the Physical Units in the scaling parameters.

To scale the measurement from measurement units to physical units, two pairs of scaling numbers are used. The First and Second Electrical Values are associated with the First and Second Physical Values, and a linear transformation is derived allowing any electrical value to be converted into the proper physical value. Also set the Electrical Units and Physical Units parameters to the appropriate values for your sensor.

The Bridge Information parameters allow you to specify how your device operates. The Bridge Configuration can be set to Full, Half, or Quarter Bridge. The Voltage Excitation Value can be specified, as can whether the excitation voltage is internal or is provided externally. (See the NI hardware manual for allowed values of internal excitation voltage.) Finally, the Nominal Bridge Resistance in Ohms must be specified.

With the above parameters, each channel can be configured differently, according to the sensor being attached. Refer to the sensor data sheet and the NI device manual for details about how to configure each channel and sensor and how to properly wire the sensor to the Daq device.

This Driver does not implement the Offset Null or Shunt Calibration functionality that may be available on the NI device.

This Driver supports the Disabled on Startup?, Include Time?, and Include State? Parameters, which are discussed in 7.5.1 Common Driver Parameters.

The NIDaqBridge Driver works in only in MICAS-X for Windows and is not supported in MICAS-X-RT.

7.6.18NIDaqCounter.vit

The NIDaqCounter (National Instruments Data Acquisition Counter) Driver is used to acquire data from the counter channels on a wide variety of NI Daq devices.

Each counter channel used by this Driver is configured separately. A single instance of this Driver can configure counters from more than one Daq device.

Label each channel by clicking in the “Name” box and entering the desired name. The “Physical Channel” parameter specifies the counter channel of an NI Daq Device. This name must exactly match the name found in MAX (Measurement and Automation Explorer). For convenience, the chart near the lower left shows the NI Daq Devices currently visible to your computer. Physical channel names can be copied and pasted from this chart to the configuration parameters.

The “Edge” parameter specifies when the count will be triggered (either when signal level is rising or falling). “Count Direction” specifies whether the counter will count up or down from its initial value. Insert the initial value of the counter in the “Initial Value” parameter. The units of measure (Hz, rpm or count) can be specified in the “Units” parameter. To specify counter output type, select either elapsed (absolute) or delta (relative) in the “Counter Type” parameter.

To delete or insert more counters, click the respective red and green buttons on the right-hand side of the window.

This Driver supports the Disabled on Startup?, Include Time?, and Include State?, parameters, which are discussed in section 7.5.1.

The NIDaqCounter Driver works in only in MICAS-X for Windows and is not supported in MICAS-X-RT.

7.6.19 NIDaqDA.vit

The NIDaqDA (National Instruments Data Acquisition Digital to Analog) Driver is used to set output voltages using the DA channels on NI Daq devices. The “Device” parameter must match the name of the NI Daq Device as it appears in MAX (Measurement of Automation Explorer) and it appears in the chart to the lower left. The chart shows the NI Daq Devices currently visible to your computer and how many

channels are available for each Device. Device names can be copied and pasted from this chart into the Device parameter.

To create a new channel, type the desired label into the “Name” control. The Slope and Offset parameters can be used to apply a linear scaling to the data being passed to the output channel. Normally, set the Slope to 1 and the Offset to 0, and the output voltage will be equal to the value of the channel in MICAS-X. However, the Slope and Offset can be used to map engineering units of a MICAS-X channel into voltage. For example, if the channel was named Flow (cc/s) and a setting of 20 cc/s required a setting of 5 V, then a Slope of .25 and an Offset of 0 would map the channel’s value of 20 to a voltage output of 5V.

The “Initial Value” is the value, in engineering units, used as a SetPoint when the program first starts. The “Channel” parameter specifies the number of the analog output channel to be used for the MICAS-X channel. Note that the chart to the lower left shows how many analog output channels are available for each device that is detected.

The “Range” parameter defines the output voltage range for an analog output channel in the MICAS-X system. Note that not all NI A/D boards support all possible output ranges available in the selection.

The red delete and green insert buttons to the right of the window can be used to delete and create new channels, respectively.

This Driver supports the Disabled on Startup?, Include Time?, Include State?, and Initial Mode parameters, which are discussed in section 7.5.1.

The NIDaqDA Driver works in only in MICAS-X for Windows and is not supported in MICAS-X-RT.

7.6.20 NIDaqDI.vit

The NIDaqDI (National Instruments Data Acquisition Digital Input) is used to read digital inputs as channels using NI Daq hardware. The “Device” parameter must match the name of the NI Daq Device as it appears in MAX (Measurement of Automation Explorer) and as it appears in the chart to the lower left of the window. The chart shows the NI Daq Devices that are currently present on your computer and the number of channels available on each one. A Device Name from that chart can be copied and pasted into the Device parameter if desired.

To create a new digital output Channel used within the MICAS-X program, type the desired Channel name into the “Name” control. The “Channel” parameter defines the Channel number of a specific port on the National Instruments Digital IO Device which is to be used for the Digital Output bit. The “Port” parameter defines the Port

number on the National Instruments Digital IO Device. The chart to the lower left shows how many Ports are available on each NI Daq Device found on your computer, and how many bits (Channels) are available in each Port. The Invert parameter can be used to invert the logic level of the MICAS-X Channel relative to the logic level actually read by the National Instruments hardware.

The red delete and green insert buttons to the right of the window can be used to delete and create new channels, respectively.

This Driver supports the Disabled on Startup?, Include Time?, and Include State? Parameters, which are discussed in section 7.5.1.

The NIDaqDI Driver works in only in MICAS-X for Windows and is not supported in MICAS-X-RT.

7.6.21 NIDaqDO.vit

The NIDaqDO (National Instruments Data Acquisition Digital Output) is used to create digital output channels using NI Daq hardware. The “Device” parameter must match the name of the NI Daq Device as it appears in MAX (Measurement of Automation Explorer) and as it appears in the chart to the lower left of the window. The chart shows the NI Daq Devices currently found on your computer and the number of channels available on each Device. A Device Name can be copied and pasted from the chart to the Device parameter if desired.

To create a new digital output Channel used within the MICAS-X program, type the desired Channel name into the “Name” control. The “Channel” parameter defines the Channel or bit number on the National Instruments Digital IO Device which is used for the Digital Output channel. The “Port” parameter defines the Port number on the National Instrument Digital IO Device to which the bit belongs. The “Initial Value” parameter defines which initial value that the digital output will be set to when the MICAS-X program starts. Note that the chart to the lower left shows how many bits (channels) and how many Ports are available for each of the NI Daq Devices currently found on your computer. The Invert parameter can be used to invert the logic level of the MCC Digital Output bit relative to the MICAS-X Channel.

The red delete and green insert buttons to the right of the window can be used to delete and create new channels, respectively.

This Driver supports the Disabled on Startup?, Include Time?, Include State?, and Initial Mode parameters, which are discussed in section 7.5.1.

The NIDaqDO Driver works in only in MICAS-X for Windows and is not supported in MICAS-X-RT.

7.6.22NiDaqRTD.vit

The NiDaqRTD (National Instruments Data Acquisition RTD) driver interfaces with NI Daq devices that can read RTD temperature sensors.

Each channel must have a channel name and unique channel number. The RTD parameter supports Pt3750, Pt3851, Pt3911, Pt3916, Pt3920, Pt3928 and custom. The 0deg resistance parameter specifies the nominal resistance of the RTD, and the min value and max value specify the range of temperature to be measured. Wiring configuration can be 2, 3, or 4 wire. The excitation source can be internal, external or none. The current excitation should be specified for External excitation.

The red delete and green insert buttons to the right of the window can be used to delete and create new channels, respectively.

This Driver supports the Disabled on Startup?, Include Time?, Include State?, and Initial Mode parameters, which are discussed in section 7.5.1.

7.6.23 NIMotion.vit

The NIMotion (National Instruments Motion) Driver allows the user to create one or more axis configurations compatible with the NI motion hardware. The “Axis Name” parameter allows the user to create a label for the axis. ‘Board ID’ lists the board identification number for the NI Motion board in use. This parameter must be consistent with the board’s configuration in MAX (Measurement and Automation Explorer). “Axis” lists the hardware axis number.

“Position Mode” sets the absolute or relative positioning mode for the axis. “Loop Mode” sets the Open or Closed-loop mode for the axis. ‘Primary Feedback’ associates the proper feedback resource for the axis.

“Unit Type” defines the type of information, Counts, or Steps that will be used when referring to axis motion. “Counts(Steps)/EngUnit” is a conversion scaling parameter which defines how many counts or steps equals one engineering unit. By scaling the motion with this parameter, MICAS-X can give commands in terms of mm or inches, for example, rather than steps or counts. “Accel/Decel (100000)” describes the acceleration and deceleration value in Engineering units per second squared. “Velocity (10000)” describes the velocity value in Engineering units per second.

The red delete and green insert buttons to the right of the window can be used to delete and create new axis configurations, respectively.

This Driver supports the Disabled on Startup?, Include Time?, and Include State? Parameters, which are discussed in section 7.5.1

The NIMotion Driver works in only in MICAS-X for Windows and is not supported in MICAS-X-RT.

7.6.24Obis.vit

The Obis Driver interfaces with the Obis lasers from Coherent. The “COM Port” parameter defines which serial port will be used by the interface. The available com ports can be found with the Check for Serial button or the Windows device manager. The “Initial Setpoint” parameter can be configured if Initial Mode? Is set to “Use configuration Initial Value”.

The Obis Driver also includes the commands “ObisStartLaser” and “ObisStopLaser”. These commands start and stop the laser corresponding with the driver.

This Driver supports the Initial Mode? Disabled on Startup?, Include Time?, Include State?, and Record Data Stream? Parameters, which are discussed in section 7.5.1.

7.6.25Omega.vit

The Omega Driver can be used to read a temperature measurement and set a temperature setpoint on an Omega Cni16 temperature controller. In addition, though not yet tested, this Driver is likely compatible with a wide range of Omega devices. At the very least, it can be altered to accommodate a wider range of Omega devices as well as a wider range of functionality within these devices.

Use the “COM Port” control to select the serial port used to communicate with the Omega device. Use the “Baud Rate” parameter to set the baud rate for the driver to be the same as the baud rate selected on the Omega controller. All other communication parameters are assumed to be the default values that the Cni16 comes preset to. In addition, the Omega Driver assumes that the Cni16 is configured to use the standard ASCII protocol, not ModBus.

Set the “Initial SetPoint” parameter to the value that the Omega temperature SetPoint should be set to when MICAS-X starts running. This parameter is only used when the “Initial Mode” parameter is set to “Use Configuration Initial Value”.

This Driver supports the Disabled on Startup?, Include Time?, Include State?, Record Data Stream?, and Initial Mode parameters, which are discussed in section 7.5.1.

7.6.26Picarro G2401.vit

The Picarro G2401 Driver acquires streaming data from a Picarro G2401 Cavity Ring Down Spectrometer. This driver was created since PicarrO was observed to send corrupt strings, making it difficult to use MOSDS without frequent errors. This driver identifies the errors and ignores them, reporting new data only when the string is valid. If this Driver is used in a system with many channels, it may be helpful to add a prefix like

“Picarro” to more easily identify channels. The COM port parameter defines which port the device is connected to.

This driver has the “G24010FileOff” and “ G24010FileOn” commands which turn on and off OSDS streaming file recording respectively.

This Driver supports the Disabled on Startup?, Include Time?, Include State? and Record Data Stream? parameters, as well as the “Check for Serial Ports” control, which are discussed in section 7.5.1.

This driver should be in its own hardware-timed acquisition loop, with a 500ms cycle time.

7.6.27PID.vit

The PID Driver is used to create PID control loops using an existing input channel as the process variable and an existing output channel as the control variable. Note that the Controllers Driver (7.6.6) can also create PID loops as well as other control loops. The PID Driver has additional advanced features that give the PID loops more flexibility, including multiple gains, gains that can be changed while the program is running, and an auto-tuning wizard.

Use the PID Loops list to select a specific PID loop to edit. The Insert and Delete buttons to its right allow the addition or removal of PID loops from this list.

Once a specific PID loop is highlighted in the list, one can edit the parameter for that PID loop. The SetPoint Channel is used to name the setpoint channel for the Controller. The Output Channel is the controller (output) Channel that this Controller sets. The Process Variable Channel parameter specifies the Channel that is used to compare to the SetPoint Channel to determine how the Controller should react. The Initial SP parameter determines the value of the SetPoint Channel when MICAS-X starts.

With the PID Driver, multiple gains can be defined for a single PID loop. At least one set of gains should be defined for each loop. When the program is running, the gain set to be used is selected with the Mode channel. A value of -1 will disable the PID loop, so that the PID Driver does not write to the Output Channel at all. A value of 0 places the loop in Manual Mode. A value of 1 or greater selects a Gain set to use for PID Mode.

For any one gain set, the Kc, Ti(min) and Td(min) parameters specify the Proportional Gain, Integral Gain (in units of Kc * minutes) and the Derivative Gain (in units of Kc / minutes). The Min Output and Max Output parameters impose limits on how far the PID algorithm can set the Output Channel. These parameters keep the control loop from running away when it saturates.

The Slope and Offset parameters are used to scale the Output Channel in Manual mode, and the inverse of this linear relationship is used to calculate a SetPoint when the Controller switches from Control Loop to Manual mode. E.g. these parameters allow the

Output Channel and the SetPoint Channel in Manual mode to have different units. As an example, assume that a pressure is being controlled by changing the value of a flow. The SetPoint Channel (in Control Loop Mode) would have units of pressure. The Process Variable Channel would have units of Flow. But assuming the flow controller required a voltage input, the Output Channel would have units of Volts. In Manual mode, the Slope and Offset would be used to calculate the appropriate Voltage for a Manual SetPoint in units of Flow. When the loop is changed from Control Loop mode to Manual mode, the inverse relationship would be used to calculate the Setpoint in Flow based on the current value of the Output Channel in Volts.

The final parameter is Start-up, which has the options Disabled (-1), Manual (0) and Gain (1, 2, 3…). When set to one of the gain selections, the loop operates using the full PID logic, using the gain set selected. When set to Manual, the SetPoint Channel directly controls the Output Channel. When switching between Manual and Control Loop, the program automatically converts the SetPoint Channel's value in such a way as to hold the loop at the current value. E.g. when switching from Control Loop to Manual, the current value of the Output Channel, scaled by the Slope and Offset, is automatically written to the SetPoint Channel. When switching from Manual to Control Loop, the current value of the Process Variable Channel is automatically written to the SetPoint Channel. When in Disabled Mode, the PID Driver does not write to the Output Channel at all. This mode allows for multiple control loops to be defined using the same Output Channel, as long as only one of the loops is enabled.

To change the Mode for a PID Controller, two options are available. If the PID Controller is configured to appear on the Control tab of MICAS-X, a Mode switch will automatically be placed to its right. A second way to change the mode is use the Set Command to set the MICAX-X Channel that has the same name as the SetPoint Channel, but with the word "Mode" preceding it.

The PID Driver also supports the AutoTune command. When this command is used, with a PID Setpoint channel as the argument, an autotuning wizard appears. From this wizard, one can select various autotuning parameters, start the autotuning process, and then choose whether or not to keep the resulting gains. Note that even if the gains are retained and used in the program from that time forward, you still need to write those gains into the configuration file for them to be used when the program is restarted.

This Driver supports the Intial Mode, Disabled on Startup?, Include Time?, and Include State? Parameters, which are discussed in section 7.5.1.

The PID Driver works in both MICAS-X for Windows and MICAS-X-RT.

7.6.28Prime Scales.vit

The PrimeScales Driver reads the weight from a PS-IN202 industrial scale made by Prime Scales, via a serial port. The “COM Port” parameter defines which serial port will be used by the scale. The “Tare Weight” parameter is used whenever the scale sends gross weight, as the Driver will then subtract the Tare Weight parameter to return net weight.

This Driver supports the Disabled on Startup?, Include Time?, Include State?, and Record Data Stream? Parameters, which are discussed in section 7.5.1.

7.6.29PWM

The PWM (Pulse Width Modulation) Driver is used to link a calculated Duty Cycle channel to a digital output channel, so that the power to a control point can be modulated to achieve stable control. The typical situation in which this Driver would be used is to have a Duty Cycle channel calculated by a PID loop, and then use the PWM Driver to cycle a digital output channel (such as on from a NIDaqDO Driver) to achieve the calculated duty cycle. This could be used to stabilize a temperature setpoint, among other things.

The PWM Driver allows one to create any number of PWM “sets”, or definitions. Each set has seven parameters. The Duty Cycle Channel defines which channel is the input for the PWM set. The selected channel can have any range or scaling, though 0 to 100% is typically most convenient. This channel is often the output channel from a PID loop, but could come from other sources instead.

The PWM Output Channel parameter defines the channel that should be switched on and off at the specified duty cycle. Typically, this channel would be a digital output channel, but it could be another type of output channel as well, if controlling a system can be achieved by setting that channel to two different values.

The Cycle Time (s) parameter defines the total time, in seconds for 100% duty cycle for the PWM set. Note that the PWM Driver uses software timing, which is only accurate to 10’s of ms, so cycle times of several seconds or more are most appropriate. Cycle times of less than one second may work in some cases, though the exact PWM pulse widths created for such short times will not be as accurate (relatively speaking) as for longer times.

The Duty Cycle Range parameter defines the scaling of the Duty Cycle Channel. If the Duty Cycle Channel is inherently in percent, then the Range parameter should be set to 100. If the Duty Cycle Channel is scaled to unity, so that it only varies from 0 to 1, then the Range parameter should be set to 1. Other ranges may be used as appropriate for the choice of Duty Cycle Channel. Note, however, that the Duty Cycle Channel is

assumed to range from 0 to some positive value. This Driver does not allow arbitrary scaling of Duty Cycle Channels that start at a value other than 0.

The Dead Band (%) parameter is used to prevent very small or very large Duty Cycles from causing frequent switching of the output channel. This parameter is always specified in percent, regardless of the Duty Cycle Range. It can be set to 0 to be ignored. As an example of a non-zero value, assume that the Cycle Time is set to 10 seconds, the Dead Band is set to 2%, and the Duty Cycle Range is set to 100 (e.g. the Duty Cycle Channel is already scaled to percent.) In this case, a duty cycle less than 2 will result in the output channel to be left in the Off state for the entire cycle, rather than cycling on for just 0.2 seconds. Similarly, a duty cycle of greater than 98 would result in the output channel be left in the On state for the entire cycle. This type of setting can save unnecessary wear on relays or similar components that occur for very short pulses, without significantly impacting the performance of the control loop.

The Off Value and On Value parameters define what numeric values the PWM Output Channel is set to during the Off and On portions of the cycle. Normally, these are set to 0 and 1, respectively. The logic of the PWM control can be inverted by setting these to 1 and 0 instead.

Each PWM Set will spawn an independent timing loop within MICAS-X, which will turn the PWM Output Channel on and off at the appropriate times. Note, however, that these loops must communicate to the output channel via the Command Loop and the Driver for the output channel, so some latency is to be expected. The more complex (and CPU intensive) that the MICAS-X configuration is, the higher the latency may become.

Use the Delete and Insert buttons to the right of the PWM Sets parameters to remove or add new PWM sets.

This Driver supports the Disabled on Startup? and Include Time parameters, which are discussed in section 7.5.1.

7.6.30RIO Scan Engine

The Rio Scan Engine Driver provides a quick way to acquire data on MICAS-X-RT for most channels that are supported by the Scan Engine.

The Notes parameter allows one to add documentation describing how the Driver has been configured.

On the Input Channels tab, enter the information for each Scan Engine input channel that the Driver should support. The Name parameter is used to specify the name of the channel within MICAS-X-RT. The four Scaling Coefficients allow one to apply a polynomial up to third order to the measured value to scale it into engineering units. The Channel Type parameter defines whether the channel is an analog channel (False) or

a digital channel (True). Finally, the IO Address parameter must contain the Scan Engine Variable address for the channel. The IO Address has the form "ni.var.io://RTTargetName/Mod1/AI0". Note that "RTTargetName can be either the name or the IP address of the target.

The configuration of the Controller (Output) Channels is similar to the Input Channels, with the addition of the Initial Value parameter. This parameter defines the value that the channel will be set to when Acquisition starts if the "Initial Mode" parameter is set to True, Use Configuration Initial Value.

Use the Delete buttons to remove a channel from the list of channels. Use the Insert button to add a channel before the channel next to the insert button being pressed.

Since adding all or most of the Scan Engine channels by hand can be tedious, two methods are provided that may help discover the available Scan Engine channels for a system. Both methods are available on both the Input Channels and Output Channels tabs.

The Find Channels from Project button attempts to populate the channels array based on information stored in the LabVIEW project. Because it requires the project, this button will not function properly in an executable version of MICAS-X. When pressed, the program will parse the project and attempt to find all the input channels that are currently configured for the Scan Engine. It will also attempt to set the Channel Type appropriately for each channel by looking at the names given to each channel in the project. If custom channel names have been entered in the project, this function will not be able to properly select the Channel Type. After using this button to populate the Channels array, one should carefully review all the channels and make corrections to all the parameters as necessary. Also note that this function has been tested on a limited number of projects and targets and may not work in all situations.

The Find Channels from Target button populates the channels array with available Scan Engine channels by querying the Scan Engine itself. It is therefore necessary to supply the IP Address of the target and ensure that the target is connected, running, and is in Scan Engine mode for this function to work. For Input Channels, there are additional options which can be checked if you wish to have the target return other input channels related to the target performance. Many targets have channels related to the Chassis, the Controller, and the CPU which can be found this way.

Additional tips and restrictions for using the Find Channels buttons are shown in the text boxes near the buttons. In addition, if an error occurs when using these functions, an error indicator is shown.

This Driver supports the Disabled on Startup?, Include Time?, Include State?, and Initial Mode parameters, which are discussed in section 7.5.1.

The RIO Scan Engine Driver works in only in MICAS-X-RT and is not supported in MICAS-X for Windows.

7.6.31 System

The System Driver allows one to include a wide variety of channels that are used to log the performance of the MICAS-X software and the computer on which it is running. Many of the channels provided by this Driver use the .NET interface. Thus .NET must be correctly installed on the target computer for this Driver to function properly. When MICAS-X is run inside LabVIEW, this is usually not an issue, since the LabVIEW installation ensures that .NET is installed. If MICAS-X is used as a compiled .exe program, it may be necessary to install the .NET framework before this Driver functions properly.

Use the Drives parameters to define any hard drives for which you wish to monitor the disk free space. Use the Delete buttons to the right to remove a Drive from this list.

The Channels to Include parameters allow one to select which of numerous additional predefined channels to include. These channels include:

- Sec Since Jan 1 – A time stamp formatted as seconds since midnight, January 1 of the current year, thus including month, day, and time in a single number.
- AC Power – This channel has a value of 1 if the computer is on AC power, a 0 if it is on battery power. This channel should work for laptops as well as systems with a properly-installed UPS.
- Battery Flag – This channel should work for laptops and systems with a properly-installed UPS. It returns the following codes for battery condition: 1 High, 2 Low, 4 Critical, 8 Charging, 128 No battery. These codes can be combined as needed. E.g. a code of 12 indicates a critical battery level and that the battery is charging.
- Battery Time Remaining (s) – This channel returns the estimated time in seconds that the computer will continue operating on battery power. This channel should work for laptops and systems with a properly-installed UPS.
- CPU Usage (%) – This channel returns the CPU Usage averaged over all cores.
- RAM Usage (GB) – This channel returns the RAM used in GB.
- Secured – This channel returns a 0 if the MICAS-X password is not currently enforced, a 1 when the password is in force.
- Acquire – This Channel has a value of 0 if the program is not acquiring data, 1 if it is.
- Record – This Channel has a value of 0 if the program is not recording data, 1 if it is.

- Debug – This Channel has a value of 0 if the program is not in Debug mode, 1 if it is.
- Last Acq Time – This channel is a time-stamp in seconds since Jan 1, 1904, indicating when the main acquisition loop (loop 0) last acquired data.
- # of Alarms – This channel indicates how many Alarms are currently in the Alarm state.

The Performance Channels allow one to add any of the .NET computer performance monitoring points to MICAS-X. To add a performance channel, select the proper Category, then the Instance, and if available, the Counter. Create a descriptive, unique Name for this channel, and the Add Channel button will become not-greyed-out, allowing you to press this button to add the defined channel to the list of Performance Channels. Use the Delete buttons to the right to remove a Performance Channel that is no longer desired.

This Driver supports the Disabled on Startup?, Include Time?, and Include State? Parameters, which are discussed in section 7.5.1.

The System Driver works in only in MICAS-X for Windows and is not supported in MICAS-X-RT. The System-XRT Driver provides similar support for MICAS-X-RT.

7.6.32 System-XRT

The System-XRT Driver allows one to include a wide variety of channels that are used to log the performance of the MICAS-XRT software and the real-time computer on which it is running. The Channels to Include parameters allow one to select which of numerous predefined channels to include. These channels include:

- Sec Since Jan 1 – A time stamp formatted as seconds since midnight, January 1 of the current year, thus including month, day, and time in a single number.
- CPU Usage (%) – This channel returns the CPU Usage averaged over all cores.
- Largest Avail Mem (kB) – This channel works on Pharlap and VXWorks RT platforms, but not on Linux. It returns the largest available memory segment in kB. An Alarm or Trigger on this channel could be used to reboot the system if the available memory becomes too low.
- Free Physical Mem (kB) – This channel works on Linux RT platforms, but not on Pharlap or VXWorks. It returns the largest available section of free RAM memory. An Alarm or Trigger on this channel could be used to reboot the system if the available memory becomes too low.
- Acquire – This Channel has a value of 0 if the program is not acquiring data, 1 if it is.

- Record – This Channel has a value of 0 if the program is not recording data, 1 if it is.
- Last Acq Time – This channel is a time-stamp in seconds since Jan 1, 1904, indicating when the main acquisition loop (loop 0) last acquired data.
- # of Alarms – This channel indicates how many Alarms are currently in the Alarm state.
- Connected-XRT – This channel is a bitwise representation of the connection status of MICAS-X-RT to a copy of MICAS-X on Windows that is running the MICAS-XRT Driver. Note that the MICAS-XRT Driver only connects to the MICAS-X-RT with the Windows MICAS-X program is acquiring data. This Channel as a bit value of 1 if the Command stream from Windows to RT is connected. A bit value of 2 means that the Message stream from RT to Windows is connected. Finally, a bit value of 4 means that the CCC connection for sending channel values from RT to Windows is working. Thus a value of 7 means that all three connections are good. Note that this channel is very similar to the Connected channel in the MICAS-XRT Driver, but the two channels operate on different sides of the connection. Also note that on the RT platform, there is no direct way for the program to tell if the CCC connection is valid or not. Thus the bit value of 4 is deduced based on a Watchdog channel. The MICAS-XRT Driver continuously updates the Watchdog channel whenever it is in Acquire mode. MICAS-X-RT on the RT platform monitors the Watchdog channel every 2.5 seconds. If a new value is not detected within that time, then the bit value of 4 is not set. Note that this mechanism works as intended whenever the MICAS-XRT Driver is in an acquisition loop that runs once every 2 seconds or faster. If it is in a slower loop, the bit value of 4 will never be set. In that case, the other two network connections (bit values 1 and 2) must be used to indicate that a connection has been made.

Use the Drives parameters to define any hard drives for which you wish to monitor the disk free space. Use the Delete buttons to the right to remove a Drive from this list.

This Driver supports the Disabled on Startup?, Include Time?, and Include State? Parameters, which are discussed in section 7.5.1.

The System-XRT Driver works in only in MICAS-X-RT for real-time platforms, and not in MICAS-X for Windows. The System Driver provides similar support for MICAS-X on Windows.

7.6.33 UPS

The UPS Driver is used to communicate with various brands of Uninterruptible Power Supplies to monitor the AC power state and the battery state. When a UPS is used with a MICAS-X system, this Driver can be used to safely shut down the program when a power outage occurs. (Depending on whether or not the instrumentation is also on a UPS, the safe shutdown could also set the hardware into a safe state.)

With its current release, this Driver is compatible with two models of UPS, though other models could easily be added in the future. When the "UPS Model" parameter is set to "Intellipower FA00339", the Driver uses the Intellipower communication API. Although this Driver has only been tested with the FA00339 model of Intellipower UPS, it will likely work with other Intellipower UPS's as well. When "CyberPower CP1500AVRLCD" is selected, the Driver monitors the CTS and DCD handshaking lines of the serial port to determine the AC power and battery states. This protocol is likely used on other CyberPower UPS's in addition to the 1500AVRLCD, and is also quite likely used by other brands of UPS's as well.

This Driver supports the Disabled on Startup?, Include Time?, and Include State? Parameters, which are discussed in section 7.5.1.

7.6.34 Vaisala HI70

The Vaisala HI70 Driver reads three channels of data over a serial port from the Vaisala HI70 humidity and temperature sensor. The "COM Port" parameter defines which serial port is used for communication. It is assumed that the baud rate of the HI70 will be set to the default value of 19200. In addition, this Driver configures the HI70 to send data automatically at 1 Hz. Thus this Driver should be assigned to a device-timed loop with a time of 1000 ms.

This Driver supports the Disabled on Startup?, Include Time?, Include State?, and Record Data Stream? Parameters, which are discussed in section 7.5.1.

7.6.35 Vaisala HMT310

The Vaisala HMT310 Driver reads 11 channels of data over a serial port from the Vaisala HMT310 high precision humidity and temperature sensor. The "COM Port" parameter defines which serial port is used for communication. The "Baud Rate" parameter must be set to match the baud rate for which the HMT310 is configured. The Channels parameters allow one to select which of the eleven available channels is returned to MICAS-X.

This Driver supports the Disabled on Startup?, Include Time?, and Include State? Parameters, which are discussed in section 7.5.1.

7.6.36 Watlow.vit

The Watlow Driver can be used to read a temperature measurement and set a temperature setpoint on a Watlow EZZone temperature controller.

Use the “COM Port” control to select the serial port used to communicate with the Watlow device. Use the “Baud Rate” parameter to set the baud used for communication. All other communication parameters are assumed to be the default values that the EZZone controllers come preset to. In addition, the Watlow Driver assumes that the EZZone is configured to use the standard ASCII protocol, not ModBus.

Set the “Initial SetPoint” to the value that the Watlow temperature setpoint should be set to when MICAS-X starts running. This parameter is used if the “Initial Mode” is set to “Use Configuration Initial Value”.

This Driver supports the Disabled on Startup?, Include Time?, Include State?, Record Data Stream?, and Initial Mode parameters, which are discussed in section 7.5.1.

7.6.37 Web Power Switch.vit

The Web Power Swith Driver is used to control a Web Power Switch 7 internet-enabled power strip. This device has eight outlets which MICAS-X can turn on and off. The IP Address must be set to the IP address which the Web Power Switch has been configured to use. The User Name and Password parameters must contain the username and password used with the Web Power Switch in its own web interface. Note that these parameters are not encrypted within MICAS-X. The Initial Mode parameter determines whether MICAS-X reads the current state of each outlet when MICAS-X is started, and uses those states as the initial values of the channels, or if MICAS-X sets the outlets to the Initial Value parameters defined in the configuration. The Include Time? Parameter determines whether or not MICAS-X prepends the Web Power Switch data with a timestamp.

The Outlets array contains the definitions for each of the eight channels. The channels must be entered in order. If a channel is unused, leave the Outlet Name blank. The Initial Value determines if the outlet set On or Off when MICAS-X 111xecut, but only if the Initial Mode is True (Use Configuration Initial Values).

When the Enable Watchdog parameter is set to True, MICAS-X will run scripts on the Web Power Switch 7 such that one or more outlets will be automatically turned off if MICAS-X or the computer on which it is running should crash. See Appendix F: Web Power Switch 7 Watchdog Function for information on configuring the Web Power Switch 7 to accommodate this functionality. Note that when MICAS-X stops, the Web

Power Switch 7 stops the watchdog functionality, so the guarded outlet(s) will then remain in their current states indefinitely.

The MICAS-X channels for the Web Power Switch are Controller, or Output, channels, since they are used to set the value of the outlet (On, 1 or Off, 0). However, since the outlets can be switched manually on the Web Power Switch itself, MICAS-X can also read the state of the outlets in during the Acquire command. (By comparison, most MICAS-X Drivers which set Controller channels only return their memory of the last values set when the Acquire command is used.) Because the outlets can be changed both by software and manually, there is a possibility of a race condition, wherein the value set by MICAS-X could be overwritten by a manually set value, or vice versa. To prevent this type of race condition, the Web Power Switch Driver imposes a wait of 400ms after any outlet value is set. (Note that this 400ms delay is only imposed when the Read Back parameter described below is set to True.)

In addition to the delay imposed after setting an outlet value, the Web Power Switch Driver interacts with the Web Power Switch via http calls, which can be rather slow in certain situations. For both these reasons, it is advised that the Web Power Switch Driver be assigned to its own Acquisition Loop in MICAS-X. It may also be worthwhile to set the Acquisition Loop for this Driver to a slower rate, such as 5 or 10 seconds, in order to limit the Ethernet traffic resulting from the communications to this device.

The Read Back parameter allows one more option for controlling the timing and functionality of the Web Power Switch. When set to True, Read Current Values from Device, every acquisition loop will read the switch states from the hardware itself, as described above. This read can take over one second, which is why it is recommended that this Driver be placed in its own acquisition loop with a cycle time of 2 seconds or more. When the Read Back parameter is False, however, the Web Power Switch Driver reports the state of the outlets based only on its memory of how it last set them. This option results in a very fast read, allowing the Web Power Switch Driver to be placed in a faster Acquisition Loop, but can potentially allow MICAS-X data to become out of sync with the states of the outlets if outlets are turned on or off manually at the device. To prevent this possibility, one should use the Web Power Switch configuration web page and lock out the hardware switches so that the outlets can only be changed by MICAS-X.

This Driver supports the Disabled on Startup?, Include Time?, Include State?, and Initial Mode parameters, which are discussed in section 7.5.1.

7.7 Instruments Available at Additional Cost

To read how to use instruments, see the Instruments section of this manual (9.6.8). In the Configuration Editor, first add the Instrument on the MICAS module editor. Then, to edit the configuration of these instruments, select the ‘Instruments’ tab

in the ‘Components’ menu located on the upper-left-hand side of the window. Next, select the appropriate instrument in the menu directly below ‘Components,’ now labeled ‘Instruments.’ Currently, no Instruments are included with the MICAS-X base package. Below are instructions on how to configure instruments that are available at additional charge.

7.7.1 Broadcast.vit

The Broadcast Instrument is a component of MICAS-X which allows the user to send data from MICAS-X to other systems by several methods. The Broadcast functionality allows MICAS-X to send a selected Channel list out a serial port or via UDP over Ethernet. The Send Command function allows the “Send” Command in MICAS-X in another instance of MICAS-X to send Commands that are then received by the Broadcast Instrument and executed by the receiving instance of MICAS-X. The Data Dashboard options allow data to be published to Shared Variables, which can be used by the Data Dashboard application to create remote panels for interacting with MICAS-X. Since this instrument is a .vit rather than a .vi, it can be instantiated multiple times within a MICAS-X configuration.

To Broadcast data, begin by selecting which channel list will be broadcast by using the drop-down menu labeled “Channel List” on the Channel List Broadcasting tab. (If no channel lists have been defined, the only list available will be the “All” list. Use the Channel Lists module editor to create additional channel lists.)

Define which strings will be prepended and added to the end of each line of data by inserting the desired entries into the “Prefix” and “Suffix” parameters.

The “Format” parameter should contain an appropriate format specifier, which defines how the data will be written in the broadcast data stream. *%#g is recommended (automatic formatting). See LabVIEW help for more information and options on format specifiers.

Select a delimiter (Comma, Space or Tab) using the ‘Delimiter’ parameter.

Check the “Include ISO Timestamp” box if the user desires to add an ISO timestamp to the output stream, after the prefix and before the first channel. The format of this time string is YYYYMMDDThhmmss.sss. Note that the letter “T” separates date and time, as per the ISO standard.

The “Bus” parameter defines which type of bus will be used to broadcast data. The options currently supported are Serial, UDP, and UDP Multicast.

When the Serial Bus is selected, choose which serial port to use by entering the desired number into the “Serial Port” parameter. The ‘Baud Rate’ parameter allows the user to determine the Baud rate that a serial port device will use.

When the UDP bus is selected, the user can set the “UDP Packet Size” parameter. The default of 0 means that the traditional packet size of 548 will be used. Note that non-default sizes may cause problems on some networks or OS’s. Use the “UDP Local Port” parameter to define the local port used for UDP transmission.

Either UDP Broadcast or UDP Unicast (one to all, or one to one) can be used when “UDP” is selected as the bus. To use UDP Broadcast, leave the “UDP Multicast or Unicast IP Address” blank, or set it to 255.255.255.255. To use UDP Unicast, set the “UDP Multicast or Unicast IP Address to the IP address of the receiving computer.

When “UDP Multicast” bus is selected, enter values for “UDP Target Port”, “Time-to-Live” and “UDP Mutlicase or Unicast IP Address” as well as “UDP Packet Size” and “UDP Local Port”. The “UDP Target Port” parameter determines the remote or target port to which UDP writes the broadcast data. The ”Time-to-Live” (TTL) parameter specifies the number of routers, minus 1, to forward a packet. The TTL value applies to all packets sent using this socket. Finally, the “UDP MC IP Address” control specifies the IP address that the UDP MultiCast stream will be sent to. If this control is left as default and empty, the address 234.5.6.7 will be used.

The “Cycle Time?” and “Cycle Time (ms)” parameters are used to determine when the Broadcast Instrument sends its data. If “Cycle Time?” Is False (Sync to Acq Loop 0), then the Broadcast Instrument sends data every time the default Acquisition Loop (Loop 0) gets new data. If “Cycle Time?” is True (Use Cycle Time), then the “Cycle Time (ms)” parameter’s value is used to determine how often data is Broadcast. Note that if “Cycle Time (ms)” is 0, a value of 1000 ms is used instead.

The “Send” Command in MICAS-X allows one to send arbitrary strings over UDP or a serial port to another system. If the receiving system is another instance of MICAS-X, and the strings are formatted as proper MICAS-X Commands, this allows one instance of MICAS-X to directly control another. For this to work, the Broadcast Instrument must be configured on the sending computer to define how the data is sent. On the Send Command Connection tab, use the Bus to select UDP, UDP Multicast, or Serial as the desired bus. Select None to disable this feature.

When the Serial Bus is selected, choose which serial port to use by entering the desired number into the “Serial Port” parameter. The ‘Baud Rate’ parameter allows the user to determine the Baud rate that a serial port device will use.

When using UDP or UDP Multicast, set the UDP parameters as described above for the Broadcast tab.

The “Send Prefix” parameter defines a string of text that will be prepended to every Command or string sent with the “Send” Command. This prefix is necessary, for example, with the NCAR RIC system, for which the prefix is usually the instrument name followed by a comma. Note that the comma must be included explicitly in this parameter. If a Send Prefix is used and the Send Command is being used to send MICAS-X Commands to an instance of the “Command” Instrument, then the Command Instrument must also be configured to know about that prefix so that it is removed from the Send string before the Command is processed.

The Broadcast Instrument can also configure how MICAS-X can communicate with a set of predefined Shared Variables. Other LabVIEW programs could be written which read and write to these Shared Variables, or they can be used with the Data Dashboard app for Android and iOS devices. The Data Dashboard is an easy, intuitive way to create graphical user interfaces for displaying and controlling data.

To configure the Shared Variables for the Data Dashboard, simply click on each of the Channel selection menus and choose the channel to assign to each variable. Up to twenty real Shared Variables and ten Boolean Shared Variables can be used. The Shared Variable names, which will be needed when you configure the Data Dashboard, are shown to the right of the Channels list. For each active Channel, there will be one Shared Variable with the name “Channel n Name” which is a string and will contain the selected Channel name, and another Shared Variable with the name “Channel n Value” which is a double precision real or a Boolean and will contain the updated value of the Channel.

The Data Dashboard Shared Variables functionality also allows communication from the Data Dashboard back to MICAS-X through the use of up to five predefined sets of Command Shared Variables. These are configured on the Data Dashboard Commands tab. These Commands appear on the Data Dashboard as buttons. When pressed, the associated Shared Variable transmits the command request to the Broadcast Instrument in MICAS-X, which then uses the configuration information to create the desired Command. To configure such a command, select the desired Command action using the “Command” drop down menu. Then select the command’s arguments using the Target parameter and the Value parameter. E.g. for the first line, Command 1, if the Set Command is selected, and the Target is the Channel “Temp Setpoint”, and the Value is “25”, then every time the Command 1 button is pressed on the Data Dashboard, the Command “Set:Temp Setpoint:25” will be issued in MICAS-X. Alternatively, the “SV Value?” parameter can be changed from “Local Value” (False) to “SV Value” (True). In this case, the value used with this Command will be taken from a Shared Variable with the name “Command 1 Value”. Similarly to the previous example, the Data Dashboard for this case would configure a button to be linked to the “Command 1” Shared Variable, and a numeric control to be linked to the “Command 1 Value” Shared Variable. Then when the button is pressed on the data Dashboard, the MICAS-X Command “Set:Temp

Setpoint:xx" will be executed, where xx is the value of the Command 1 Value Shared Variable which will be obtained from the numeric control on the Data Dashboard.

Since the Data Dashboard functionality utilizes a single set of predefined Shared Variables that are contained in a LabVIEW library in the MICAS-X project, only one instance of the Broadcast Instrument should be configured to use the Shared Variables.

The Broadcast Instrument is only available in MICAS-X for Windows, not MICAS-X-RT.

7.7.2 Command.vit

The Command Instrument provides an interface for receiving commands in MICAS-X from another instrument or software package. The Command Interface Utility is one example of a program that can send commands to MICAS-X via the Command Instrument. Since this instrument is a .vit rather than a .vi, it can be instantiated more than once in a single MICAS-X configuration.

Begin configuring the Command Instrument by selecting which bus will be used for receiving commands (Serial or UDP) by selecting an option in the "Bus" parameter.

If Serial is selected, enter the serial port number in the "Serial Port" parameter to define which port is used to receive commands. Next, select the "Baud Rate" parameter and insert the desired Baud rate that a serial port device will use.

If UDP is selected, define the port that will be used to listen to commands by entering the desired number into "UDP Local Port". For either option, choose whether or not to log all individual commands by toggling the "Log each Command received/Don't Log each Command" switch to the right of the other parameters.

The "Command Prefix" parameter defines a string of text that is prepended to every Command that is being sent to the Command Instrument. This prefix is necessary, for example, with the NCAR RIC system, for which the prefix is usually the instrument name followed by a comma. Note that the comma must be included explicitly in this parameter. If the Send Command is being used to send MICAS-X Commands to this instance of the "Command" Instrument, then the Broadcast Instrument on the sending computer must also be configured to know about that prefix so that it is prepended properly to each Send string before being transmitted. If the Command Interface is being used to send Commands to this instance of the Command Instrument, then it will know about this prefix because it will read the configuration file containing the Command Instrument parameters.

The Command Instrument is only available in MICAS-X for Windows, not MICAS-X-RT.

7.7.3 Email.vi

The Email Instrument allows one to configure Alerts that are sent automatically to one or more email addresses. In addition, it can be used to manually send short emails to the same list of recipients. (This second use case is typically for testing purposes.) Note that this Instrument has been tested to work with some email servers. However, the options for authentication that this modules supports are limited, and it may be the case that your email server will not work correctly with this Instrument.

Configure the Email Instrument by entering the information required to connect to an email server. This includes the Sender's Email Address, the Outgoing Email Server name, the Port (often 587), and authentication information, including Username, Password, and whether or not to use TLS authentication. Note that if the Hide Password control is True, the password will not be displayed directly on the configuration page.

When the Python version is selected, one must also enter the absolute path on the computer in use to the python program. This version also requires that a third-party installation of Python be present on the computer in use. The version 3.6.8 Python x86-64 executable installer, found at https://www.python.org/downloads/windows/ has been tested with MICAS-X. If the above link does not work, contact Original Code Consulting for an archived copy of this version. When this version of Python is installed, the path is usually C:\Users\YourUsername\AppData\Local\Programs\Python\Python36. Also note that when the Python option is used, MICAS must be able to find the Python script named "sendmailssl.py" in the MICAS-X Resources folder. This file is normally distributed with every copy of MICAS-X. The python version of e-mail will not work with any windows operating system using XP or earlier.

In the Subject field, enter text that will appear as the subject line for every email sent by this module. Text identifying the source as MICAS-X and/or your particular experiment is suggested, such as "Alert from MICAS-X". Finally, enter the addresses that you wish to have e-mails sent to in the Recipients' Addresses list. At least one address must be provided, but more than one can be used to enable a group of people to receive the email Alerts. Note that all emails sent by this Instrument will be sent to the entire list of recipients and will use the same Subject line.

Note that the password used with the server is saved in the configuration file. It is saved in a scrambled format, making it very unlikely that it could be extracted and used outside of MICAS-X, but the use of this Instrument is at your own risk. Note that the configuration file is saved with the data, so the scrambled password will be archived in numerous places.

The Version parameter determines which of several versions of code are used to send the emails. Each version has its own benefits and risks. It is also possible that some versions may work successfully with some email servers while other versions do not, so try using different versions if you have difficulty getting the email function to work. Currently, the available versions are:

- NI – This version uses routines written by National Instruments for sending emails. A serious drawback of this version is that it will hang the entire MICAS-X program when sending an email. The hang can be for as short as one second to as long as 5 seconds or more. If you use this version, it is advised that MICAS-X be configured to send e-mails only for very important notifications when the program or hardware is not working properly. You should avoid using the email function to send status messages when everything is working normally, as the action of sending the email will likely cause a loss of data.
- .NET – This version is based on an example found on the NI website. It makes use of a .NET interface to access email functions built into the OS. This version does not appear to cause MICAS-X to hang. It should be noted, however, that this version can only send to one address at a time, so when multiple addresses are specified, this version is called multiple times.
- Python – This version uses a call to the Python language to send emails. This version supports servers that use the "SSL\TLS" protocol.

The NI and .NET versions have been found to work with servers that use the STARTTLS protocol. In general, it is best to set the "TLS?" parameter True to work with these servers. The Python version works with servers that use the "SSL\TLS" protocol. Also set the "TLS?" parameter True to work with these servers as well. Set the "TLS?" parameter false to work with servers that do not use encryption.

It can be challenging to configure email server information. It is a good idea to verify the server address, username, and password first by using them in another email program. Always double-check the spelling of these items. Then try the various versions and turn the "TLS?" parameter on and off to determine which combinations work with your server. If more than one setup is found to work, choose the one that hangs MICAS-X the least when it is sending an email.

To have MICAS-X send an e-mail automatically, use the Alert Command, described in section 14. The Alert Command can be used with a Trigger, a Sequence, a Button, or sent from an external system. Use the bit value 16, 32, 64, or 128 in the numeric parameter of the Alert Command to tell MICAS-X to send an e-mail alert when the command is executed.

It is recommended that you create a dedicated email account for use with MICAS-X. Since MICAS-X will be sending e-mails only, and not receiving any, setting the incoming message space to 0B can prevent spam email from wasting disk space. If a

dedicated account is created, then, in the unlikely event that the password is compromised, no one's personal email will be jeopardized.

It is possible to create an email address for nearly any cell phone number. In this way, the Email module can be used to send sms text messages. Each cell phone carrier has a different format for such email addresses, so it is imperative that you know the carrier of the phone number you wish to text. Refer to the carrier's technical support for information on how to format the email address, or search the internet for information. Many websites list the common email-sms gateway addresses, including http://www.makeuseof.com/tag/email-to-sms/

As of January, 2016, the following email addresses should work:

AT&T	cellnumber@txt.att.net
Verizon	cellnumber@vtext.com
T-Mobile	cellnumber@tmomail.net
Sprint	cellnumber@messaging.sprintpcs.com
Virgin Mobile	cellnumber@vmobl.com
US Cellular	cellnumber@email.uscc.net
Nextel	cellnumber@messaging.nextel.com
Boost	cellnumber@myboostmobile.com
Alltel	cellnumber@mesage.alltel.com

The Email Instrument is only available in MICAS-X for Windows, not MICAS-X-RT.

7.7.4 M-Link Server.vit

This Instrument is used to allow another instance of MICAS-X (the "client" instance) to communicate with and control the instance of MICAS-X that is running this Instrument (the "server" instance). When configuring this Instrument, one specifies the Channel List that will be sent to the client instance. The default is "All" Channels, but one can specify a shorter, more specific list of Channels if desired to improve clarity. In the "client" instance, these Channels will have the prefix "ML" in addition to any prefix that the M-Link modules have, to indicate that they originated on the "server" instance of MICAS-X.

If the "Send all Log Messages" option is selected, then every message and error that is logged on the "server" MICAS-X will be transmitted to the "client" MICAS-X as well. Note that such messages will appear with a "ML" prefix (as well as any prefix that the M-Link Server instrument has) on the "client" instrument to indicate that they originated from the "server" instance.

The Update Time (ms) parameter determines how often the "server" instance of MICAS-X refreshes the Channel values that it is sending to the "client" instance. A value of 1000 is typical.

When using the M-Link system with a Channel List other than “All”, it is important to note a potential issue that can arise. Consider a “server” system that has two Channels, Temp SP, which is used to set the temperature setpoint of a heater, and Temp (C), which measures the temperature of the heated element. Further assume that a Channel List was used with the M-Link Server that only contained the Temp (C) Channel, but not the setpoint Channel. On the “client” system, it will be possible to select the “ML-Set” Command to set a Channel on the “server” system, and the “ML-Temp SP” Channel will be available for use with that Command, even though it is not in the Channel List of Channels being transmitted from the “server” to the “client” as data Channels. All of this will work as expected. However, when setting up this Command on the MICAS-X tab using the “Controllers” selection, MICAS-X attempts to display the current value of the Controller Channel that will be set. Since that Channel is not included in the Channel List, the “client” system has no knowledge of its value, and an error will occur. One can either just ignore such errors, or one can ensure that all Controller Channels on the “server” system are included in the Channel List being sent to the “client” system.

7.7.5 Ocean Optics.vit

The Ocean Optics Instrument is intended for acquiring spectra from various Ocean Optics spectrometers. Any Ocean Optics spectrometer that is compatible with the Ocean Optics instrument driver used in MICAS-X (currently OmniDriver 2.46) should work with this MICAS-X Instrument, though not all the available functionality for the OO spectrometers is included in this instrument. The Ocean Optics OmniDriver must be downloaded from Ocean Optic’s website and installed before using this instrument in MICAS-X.

The Spectrometer # parameter is used to specify which OO spectrometer attached to the computer is to be used by this instrument. Normally this is set to 0 when only one spectrometer is in use. Note that although the OO Instrument is a .vit and can therefore be instantiated more than once, the current version of this module (version 2.1.0) only supports one OO spectrometer in a MICAS-X configuration. Modifications will be made in the future to allow multiple instances of this Instrument to be used simultaneously with multiple spectrometers.

Exposure Time (ms) defines how long the spectrometer will accumulate light before transferring the data to the computer. Many OO spectometers have a lower limit of exposure time around 5ms, so values below that limit should be avoided. The value of this parameter is used for the Exposure Time when MICAS-X first starts running, but it can also be changed at any time while the program is running.

The # to Co-Add parameter defines how many spectra of the specified exposure time are added together before the result is written to the data file as a complete, co-

added spectra. As with Exposure Time, this parameter can be changed after the program is running.

The On/Off parameter defines whether the Ocean Optics instrument begins enabled (On) or disabled (Off) when MICAS-X starts. In its Off state, no calls are made to the hardware, so in this state the OO Instrument will not generate errors if the spectrometer is not currently hooked up to the computer. After MICAS-X is running, the On/Off state can also be changed at any time.

When the OO Instrument is On, it acquires the number of spectra specified by the # to Co-Add parameter, adds them together, displays them (as they are being co-added, and along with the most recent previous co-added spectra), and then writes them to file when a full spectra has been obtained. The file is ASCII text, with two lines of header. The top line has the label “Wavelengths”, as well as a row of comma-separated wavelength values. The Second row has column headers for the first three columns of data: Time, Exp Time (s), and # Co-Added. The time column is in the format hh:mm:ss. The data file will automatically restart at the top of each hour.

7.7.6 Ramp.vit

The Ramp Instrument is a module that can be used to create a linear ramp over time of the value of one output (Controller) channel. This Instrument can be added to a configuration multiple times, so that multiple ramps can be defined. The ramps created with this instrument can have a time resolution of 10’s of milliseconds, whereas Sequences can easily be used to create ramps with time resolution on the order of one second. Thus, if a very smooth ramp is required, the Ramp Instrument may be more useful than a Sequence. In addition, using the Ramp Instrument may be simpler than defining a Sequence for the same purpose.

The Ramp Output Channel is used to select which MICAS-X Controller Channel will be ramped over time.

The Ramp On Channel is optional. If it is left undefined (showing the value “–”), the Ramp can only be turned on and off manually using a switch on the Ramp display. (Note that the Ramp Instrument must be configured as Displayed in order to use this manual switch.) If this channel is defined, then the Ramp will turn on whenever the specified channel is non-zero. Changing that channel’s value to 0 will turn the Ramp Off. If the Ramp mode selected is not one of the Repeated modes, then when the Ramp finishes its normal run, the Ramp On Channel will be changed to a value of 0 by the Ramp Instrument. Since the Ramp On Channel’s value can be changed by the MICAS-X program, this Channel must be a Controller. An input Channel cannot be selected as the Ramp On Channel.

The Ramp Initial Channel is optional. If it is left undefined (showing the value "–"), the initial value of the Ramp will be selected from the Initial Value parameter, which can only be changed manually by the operator, not programmatically. If the Ramp Initial Channel is defined, then other functions within MICAS-X can alter the value of the Ramp's initial value by changing the value of that Channel. The Initial Value parameter is therefore only used if the Ramp Initial Channel is undefined.

The Ramp Final Channel is optional. If it is left undefined (showing the value "–"), the final value of the Ramp will be selected from the Final Value parameter, which can only be changed manually by the operator, not programmatically. If the Ramp Final Channel is defined, then other functions within MICAS-X can alter the value of the Ramp's final value by changing the value of that Channel. The Final Value parameter is therefore only used if the Ramp Final Channel is undefined.

The Ramp Time Channel is optional. If it is left undefined (showing the value "–"), the scan time of the Ramp will be selected from the Ramp Time (s) parameter, which can only be changed manually by the operator, not programmatically. If the Ramp Time Channel is defined, then other functions within MICAS-X can alter the value of the Ramp's scan time by changing the value of that Channel. The Ramp Time (s) parameter is therefore only used if the Ramp Time Channel is undefined.

The Ramp Mode is used to select what type of ramp will be created. The Instrument can ramp up from the Initial to Final Value, down from the Final to Initial Value, or Up and then Down, and each ramp type can be set for a single execution or continuously repeated execution.

The Ramp Scaling parameter defines whether the Ramp will be linear in time or logarithmic.

The Ramp Refresh Time (s) parameter defines how often the Ramp Output Channel will be updated. This parameter can be made as small as 10 ms, or can be multiple seconds or longer. If times less than 100 ms are used, it is advised that the program be tested carefully to ensure that it performs as expected. The minimum time that can be used depends on the performance of the computer on which the program is being run, the other items configured within MICAS-X, and the hardware associated with the Ramp Output Channel.

Note that the ramp value is calculated by taking the elapsed ramp time (current time minus the time that the ramp started) divided by the Ramp Time to arrive at a fraction of the ramp completed. The span of the ramp (Final minus Initial Values) is then multiplied by the fraction completed, and added to the Initial Value to arrive at the Ramp Value at any particular time.

Once the Ramp starts, the Initial and Final Values and the Ramp Time are saved in memory. If the Channels defining those values change after the Ramp has started, the

changes are therefore ignored, and the Ramp is calculated only based on the values that were present when the Ramp started.

The Ramp Instrument is only available in MICAS-X for Windows, not MICAS-X-RT.

7.7.7 RT File Transfer.vit

The RT File Transfer instrument automates the transfer and archiving of data files created on an embedded RT target by MICAS-X-RT to the Windows host. After transferring files, they can be deleted from the sender in order to maximize disk space.

The “FTP Server”, “User Name”, and “Password” parameters are meant for gaining accesss to the ability to transfer files to the computer. The Delete? Parameter will either delete or keep remote files after moving them. The Max Files parameter will prevent any one file transfer from occupying the system for too long. It is most useful when backing up a system that hasn't benn backed up in a while. 0 will mean there is no limit. The limit rolls over every 2 hours

7.7.8 SMPS Ramp.vit

The SMPS Ramp is an extension of the Ramp Instrument, hence many of the parameters have identical definitions and functionality. For explanations of the Ramp Output Channel, Ramp On Channel, Ramp Initial Channel, Ramp Final Channel, Ramp Time Channel, Scaling, Initial Value, Final Value, Ramp Time (s), Ramp Mode, and Ramp Refresh Time (s) parameters, see the section above on the Ramp Instrument.

The SMPS Ramp Instrument is intended for acquiring data from a scanning DMA (Differential Mobility Analyzer). Hence in addition to the above parameters, it has parameters that define how the particle count data is acquired, and what the flow conditions are. These parameters include the Particle Count Channel, which must be set to the data channel that contains particle counts. Note that the design of the SMPS Ramp Instrument is such that the Particle Count Channel should acquire data in its own loop at a 10 Hz loop rate. The Particle Count Time Channel must be set to the timestamp channel of the Driver that contains the Particle Count Channel. This channel is used by the Instrument to detect when new particle count data is available.

The CPC Aerosol Flow Channel must be set to a data channel that contains the flow rate of the CPC particle counter, in units of cm^3/s. The DMA Aerosol Flow Channel must be set to the data channel that contains the flow rate of the aerosol inlet of the DMA, and the DMA Sheath Flow Channel refers to the Sheath Flow of the DMA. Both DMA flows must be in units of LPM.

The Temperature and Pressure Channels must be set to data channels that contain the values of the DMA column temperature and pressure, in units of degrees C and kPa. Note that the flow, temperature, and pressure channels can be from real measurements, if the required sensors exist, from Equation channels that rescale measurements that are in other units, or from Manual channels that are used just so that the operator can enter the appropriate values. Also, note that Globals 4, 5, and 6 must be set to the values of the DMA column length in cm, and the inner and outer DMA diameters (R1 and R2) in cm.

The # of Bins parameter specifies how many bins the voltage scan is separated into and into which the particle counts are accumulated. The Plumbing Time (s) parameter specifies the delay due to plumbing length and flow rates between when a particle is transmitted by the DMA until it is detected by the CPC.

The SMPS Ramp Instrument is only available in MICAS-X for Windows, not MICAS-X-RT.

7.7.9 Speech Recognition.vit

The Voice Recognition Instrument Is useful for giving commands to MICAS across the room. For example, one could start and stop sequences while attending to the experiment and not present at the computer. This instrument uses the Windows Speech Recognition utility. It is recommended to train that utility before using this instrument.

The Voice Recognition configuration editor uses the “Phrases” box to select the phrase to be recognized by the computer. Clicking on a phrase selects it and makes it the active phrase. Once it is highlighted, you can change its text. The “Delete Selected Phrase” button will delete the active phrase. The “New Phrase” button will create a new phrase with a default name. The parameters to the right program the command for the active phrase to execute. The Command selection controls which MICAS command is to be used, and the parameters below it are the parameters for that command. The Phrase Prefix is a phrase that will be necessary to make the computer recognize a command. If the prefix is left blank, and the computer's speakers are turned on, the computer can get itself in a loop, issuing and acknowledging the same command, hence it is recommended to always use a prefix, such as “MICAS”. The Speech Recognition Starting Setting controls how the speech recognition instrument starts: Off, Test, or On. The Score Threshold is used to adjust how accurately the speech engine must match a phrase before acting on it. This parameter ranges from 0 (no match) to 1 (perfect match). A value of 0.85 to 0.9 usually works well.

The display for the speech recognition instrument displays each command recognized. The Phrase Prefix display shows the prefix which must be used before each phrase. The Speech Recognition Setting controls how the computer will react to the

commands. Off will have no reaction, Test will only say what command is recognized, while On will execute the command.

In order to issue commands, first make sure the Speech Recognition setting is set to Test or On. Then clearly say the prefix and then the phrase. For example, if the prefix is MICAS and the command is “say hi”, then say MICAS, Say hi”. If the computer has trouble recognizing the correct command you can try to make them more distinct or train the windows speech recognition engine further. The Score for matching a phrase is displayed, and the Score Threshold can be adjusted while the program is running to further optimize speech recognition.

7.7.10Text to Speech

The Text to Speech instrument is mostly used for voicing alarms. When included in a MICAS configuration, the text to speech instrument can be used to verbally announce the alarm. The configuration editor of the Text to Speech instrument has no parameters.

The display tab for the Text to Speech instrument allows the user to test messages. Either type a message to be spoken into the Message to Test space or copy and paste from the Pre-Programmed messages. Then click the Read button. The “Last Message Read” textbox will display the text of the message that was read most recently.

The Text to Speech functionality is normally used with via the Alert Command. Using bit 8 (value 256) of the Value parameter causes the text of the Alert to be sent to the Text to Speech Instrument.

7.7.11WebCam.vit

7.7.12Web Page.vit

The Web Page Instrument displays a web page in an embedded browser. Customization can allow for extra functionality like ingesting data from the website into MICAS.

The URL parameter defines what page the browser will load. This should be formatted as a regular URL. The Cycle Time parameter defines how often the web page will send an image and refresh. The “SV to Use” parameter defines whether or not to send an image of the site to a shared variable and which shared image variable. O will

not send the image, 1 will send it to Image1 shared variable, 2 will send it to the Image2 shared variable.

The “Region of Interest” parameter defines what part of the image will be shown on the display. The indicators on the chamber display are 480x360 pixels each.

This instrument may cause labview to crash when MICAS is stopped. Additionally, not all web page contents may be forwarded to the shared variables and some parts may be blank since the web page is controlling those parts, not MICAS.

7.8 Displays

To read about how to use Displays while MICAS-X is running, refer to the Displays section of this manual (9. 6. 9). For instructions on how to add Displays to the MICAS-X configuration, refer to section 7. 4. 1 To configure individual displays, select ‘Displays’ in the ‘Components’ menu in the upper-left-hand corner of the MICAS-X Configuration Editor. Then click the desired Display in the ‘Displays’ box that will appear below ‘Components.’

7.8.1 3Graphs.vit

The 3Graphs Display presents three time-series graphs, each with two traces, one on the left Y scale and one on the right. To configure the 3Graphs display, begin by selecting which channel list will supply the channels that will be viewable on the graphs. Although the channels being graphed on each graph can be changed while MICAS-X is running, only those channels that are part of the selected Channel List will be available. If it is important for the user to be able to access any and every Channel, the “All” List can be selected, though be aware that it can be difficult to find specific Channels in the list of All Channels when MICAS-X is configured with many Channels.

Each graph (1, 2, and 3) has options to select what is displayed on the left (‘GraphX Left’) and right axes (‘GraphX Right’). Use the drop down menus to select which channels are displayed on each graph when the program starts. Note that the user can change which Channels to display when the program is running.

Each left and right axis also has options to select whether the axis starts in an auto-scaling mode or a manual scaling mode. Note that for Manual Scaling, there is currently no way to define the minimum and maximum values of the scale. These can be edited by the user when the program is running, to set them to appropriate values for the data being displayed. Click the ‘Points?’ buttons (options for left and right can be used independently of one another) to display points where each data point is collected, in addition to being connected with a line. This can be useful in certain situations, such as determining which periods of time data is missing from.

On the right-hand portion of the screen, select how much time history should be displayed in the trend graphs by selecting an option on the drop-down menu labeled

'History Length.' The value of this history length can be changed when the program is running. This parameter merely sets its initial value. Note that "Manual" and "From Pointer" are not allowed as initial values.

Adjust the background color of the graph by clicking the shaded box labeled 'Graph Background' below the 'History Length' parameter.

The 'Show Message Log' parameter in the bottom-left-hand portion of the window allows the user to determine if Message Log controls on the 3 Graphs display will be shown. If these are made visible, the operator can use them to enter notes into the log file directly from this Display.

As with all Displays, the 3Graphs Display works only in MICAS-X for Windows and is not supported in MICAS-X-RT.

7.8.2 Big Display

The Big Display is designed to show a few key Channels in large fonts so as to allow them to be viewed at a distance. It allows for up to three Channels to be displayed as LEDs, and for up to three Channels to be displayed as numerics. The LED displays map a value of 0 to Off and any other value to On. Note that if any channels are left undefined (e.g. "–"), those indicators will be made invisible on the Big Display. It is often useful to configure this display to open up outside of the MICAS-X window so that it is always visible. (See section 7.4.1 for information on setting the "Outside?" parameter for a Display.)

As with all Displays, the Big Display works only in MICAS-X for Windows and is not supported in MICAS-X-RT.

7.8.3 Document Display.vit

The Document Display is intended for providing instructions and information to the operator of MICAS-X. This information can be synchronized to the operation of the program. In the Document Editor, add documents to the list by clicking on the folder browse button to the right of the document list. Remove a document by pressing the Delete button to its right.

Note that all documents used with the Document Display must be located in the "Documents" folder in the MICAS-X support folder. (When MICAS-X is used inside LabVIEW, the support folder is C:\MICAS-X. When used as an executable, the support folder is C:\OCC\MICAS-X Support.)

Use the Initial Document and Initial Section parameters to determine what documentation will be displayed when MICAS-X starts. The "Include Time?" parameter applies only to the Document Driver. The Document Driver can be used in conjunction with the Document Display. It creates two channels, Document Number and Section

Number, which can be used to programmatically control the display of the documentation.

The Document Display can display several types of documentation. These include .txt files (simple text files), .mdoc files (a simple text file with a small amount of additional formatting), and .rtf files (rich text format files, such as can be created with WordPad, Word, or Open Office Writer). Rich text format files provide the most options for formatting the displayed text. .mdoc files allow MICAS-X to access and display sub-sections of a document, whereas .txt and .rtf files are only displayed as one section. The mdoc Editor Utility (see section 11.5) provided with MICAS-X can be used to create .mdoc files (which can also be created manually in any text editor).

Since the Document Display uses a .NET accessor to display .rtf files, .NET must be properly installed on the target computer for this Display to function. When MICAS-X is run inside LabVIEW, this is usually not an issue, since the LabVIEW installation ensures that .NET is installed. If MICAS-X is used as a compiled .exe program, it may be necessary to install the .NET framework before this Display functions properly.

When the Document Display is running, the user can access any section of any document by using the Document and Section controls at the top of the display. In addition, new documentation can be read in by pressing the Load A Document button.

MICAS-X can also present documentation programmatically. To enable this, include the Document Driver in the MICAS-X configuration as well as the Document Display. Note that both the Document Driver and the Document Display must have the same prefix in order for them to work together.

As an example, consider a Sequence that has been created which steps the operator through several actions to complete an experiment. The Sequence could use the Alert command to present the user with small amounts of information. The Ask and Ask(Num) commands can be used to obtain information from the operator as the Sequence progresses. In addition, the Document Display could be used to present larger amounts of text, such as instructions and explanations. The Sequence could use the Set command with the Document and Section channels to ensure that the proper information is presented to the operator at the appropriate times during the Sequence.

Note that, as with any Display, the Document Display can be configured to be displayed inside MICAS-X in a tab of its own, or Outside of MICAS-X as its own window. When displayed Outside, the operator can position and size the Document Display window. MICAS-X will remember the most recent size and position of this window each time it is started. This allows the operator to place the instructional documentation in a convenient location on the computer screen, so that it is always accessible when needed, without having to move between tabs to see the documentation.

It is important to keep in mind that the Document Display is intended for presenting short instructions and information. This module does not do any

sophisticated memory management. When a document is loaded into the Display, the entire document is read from disk into memory. It is therefore unwise to read especially large documents into this module, as performance issues could result.

As with all Displays, the Document Display works in only in MICAS-X for Windows and is not supported in MICAS-X-RT.

7.8.4 Sequences Display.vi

The Sequences Display provides a tab in MICAS-X that shows all the Sequences as well as their current running state. Sequences can be used in MICAS-X without including the Sequences Display. The Sequences Display is included in MICAS-X by adding it to the Displays list on the MICAS editor, as described in section 7.4.1. There is no additional configuration specifically for the Sequences Display.

If Sequence Sets have been configured, a Sequence Set control appears on the right of this Display. This control can be used to select which Set of Sequences is currently displayed.

As with all Displays, the Sequences Display works only in MICAS-X for Windows and is not supported in MICAS-X-RT.

7.9 Displays Available at Additional Cost

7.9.1 Indicators Display.vit

The Indicators Display allows one to configure a Display with anywhere from 1 to 32 channel values displayed. Each channel can be configured to be a gauge, bar, numeric, or LED display.

Begin configuring the Indicators Display by selecting the number of rows and columns of indicators you wish to include. This Display was originally designed to fill a screen at 1920 x 1080 resolution (when set to be “Outside” of MICAS-X). In this case, all 4 rows and 8 columns can be displayed. When used inside of MICAS-X (as one of the tabs), 3 rows and 4 columns of indicators can be used. (A future version of this display will allow the size of the Indicators Display window to be adjusted to fit the number of rows and columns selected. With the initial release, the window size is fixed.)

After selecting the number of rows and columns, the grid of indicator definitions will adjust to the selected size. Click on each one of the grid locations for which you want an indictor to be displayed. When a location is selected, it turns grey, and its configuration information appears near the bottom of the screen. Use these parameters to select which kind of indicator will be used in that location, which channel’s value to display in that indicator, and the additional parameters that may be required for that indicator type, as described below.

If “none” is selected as the type, the grid location on the Indicators Display will remain blank.

If “Gauge” is selected as the type, a round gauge display will be used in the associated grid location. For this indictor type, the minimum and maximum displayed values must be defined. If the channel value exceeds these limits, the gauge display is pinned at the limit.

If “Bar” is selected as the type, a rectangular bar indicator will be used in the associated grid location. For this indictor type, the minimum and maximum displayed values must be defined. If the channel value exceeds these limits, the gauge display is pinned at the limit.

If “Numeric” is selected as the type, a text box “numeric” indicator will be used in the associated grid location.

If “LED” is selected as the type, a round LED Boolean indicator will be used in the associated grid location. In this case, the “False Value” parameter defines the numeric value associated with the LED False state. Normally, this is “0”. Any value other than the False Value will cause the LED to display its True state. If the “NaN = False?” box is checked, then the value “NaN” (Not a Number) will also be displayed as False, in addition to the False Value. This can often be useful since NaN is used in MICAS-X when a Driver is disabled or not ready, and often those conditions should not be interpreted as “True”.

7.9.2 Map.vit

The Map Display allows one to view any channel’s data color coded on a map, using GPS coordinates to mark the path over which the data was acquired. This display can be useful for mobile labs, airborne experiments, and similar applications. The maps for this Display must be loaded from an internet map tile archive ahead of time. Instructions for loading map tiles are contained in an additional document. Since the map tiles are pre-loaded, no internet connection is needed during acquisition to display data on the map.

The Map Folder Name parameter must contain the name of the map tiles folder. This folder must be located in the \Maps directory within the MICAS-X support directory: e.g. C:\MICAS-X\Maps for source code, or C:\OCC\MICAS-X Support\Maps for the executable version of MICAS-X.

The Vehicle Mark File parameter should contain the name of the .png file that will be used to display the vehicle location. This file must be in the \Objects subdirectory of the Maps directory mentioned above.

Use the Update Time (ms) to specify how often the Map Display is updated. If the GPS and/or displayed channel data is acquired at 1 Hz, a value of 1000 ms would be appropriate.

Center Latitude and Center Longitude define the center point of the map when MICAS-X starts. The Min Zoom and Max Zoom should be set to the minimum and maximum zoom levels of the map tiles that have been pre-loaded. Zoom levels range from 0 (the entire world) to 19 or more (roughly the size of a single building). Keep in mind when loading tiles that adding higher levels of zoom will dramatically increase the tile disk size and the download time. Set the Initial Zoom parameter to the zoom level you wish to have the map set to when MICAS-X starts.

The Latitude and Longitude parameters should be set to those channels which contain the vehicle's latitude and longitude data, typically from a GPS. The Data Channel parameter defines which channel is color-coded and displayed at the coordinates specified by the Latitude and Longitude channels.

The History Length parameter defines the starting value for the History Length control on the Map Display. This control can be changed while MICAS-X is running to select what time selection of data to display on the map.

The Max # Points parameter defines the maximum number of Lat/Long/Data points to display on the map at any one time. If more points than this are in the time selection, the data will be decimated before being displayed. With a large number of points, this decimation is usually undetectable to the human eye, but without a reasonable limit to the number of points displayed, the Map Display can become very sluggish and unresponsive.

Markers are an additional feature that can be useful for some mapping applications. Markers are small .png files which can be placed on the map at a specified location. Currently, markers must be defined in the configuration file, e.g. they cannot be added to the Map at when the program is running.

The Markers array allows one to specify the Latitude, Longitude, and .png image to use for each Marker. To delete a Marker, right click on the array near the unwanted marker until you are able to get a right-click menu showing with "Delete Element" as one of the options. Select Delete Element to delete the Marker you clicked on.

Note that there is an array of Marker Images to the lower left of the Markers array. This array displays all the .png files in the \Maps\Objects directory. These names can be copied and pasted (CTR-C, CTR-V) into the Markers array to ensure proper spelling of the file names.

The "Recenter Markers after each Zoom" option has a fairly subtle operation. The Map software locates a Marker at specific coordinates relative to the upper left corner of the Marker image. For the MICAS-X Map Display, the Marker coordinates are

mathematically adjusted so that the Marker is centered over the coordinates that were specified in the configuration file. However, this means that whenever the map is zoomed in or out, the Marker position on the map must be re-calculated to keep it centered on the correct coordinates. Unfortunately, when more than a few Markers are used on a map, repositioning all the Markers is very slow. Thus setting this parameter to "Don't Re-center Markers" greatly improves the map drawing rate in such a case. When this is done, the errors generated by not repositioning the Markers can be minimized by setting the Initial Zoom to a value close to the Max Zoom. E.g. if the Markers are first drawn at (or near) maximum zoom, their calculated position will be very close to correct as the user zooms the map in. On the other hand, if the map is initially drawn at a very low zoom and the Markers are not Re-centered as the map is zoomed in, the marker positions will be very noticeably out of position at higher zooms.

7.9.3 Multi Display.vit

The Multi Display is a single, flexible Display that can be configured to have up to six different Views. Each View defines a set of Channels to graph, a set of Buttons to use, and a set of Options.

The Multi Display contains two time-series graphs, and each graph supports two Y axes, one on the left and one on the right. The Top Graph can have up to 8 Channels displayed on it, with any channel on either axis. The Bottom Graph can have up to 4 Channels displayed, again with any channel on either axis. For the graphs, the channels are preselected in the configuration, rather than being user-selectable at run time. Each Channel graphed also displays its current value, and the data can be displayed with or without points.

Up to four Buttons can be defined for each View, which appear across the top of the Display. Each button can be configured to execute one Command when changing from False to True, and another when changing from True to False. The configuration of the Buttons is similar to those across the top of MICAS-X.

Up to five Options can be defined for each View. An option can be a single Channel numeric indicator, a Controller Channel that can be changed by the user, a T/F Indicator, a Switch (T/F Controller), a Sequence, an Enumerated List Indicator, or an Enumerated List Controller. Controllers, Switches, and Sequences are configured just as they are for the Control Tab of MICAS-X, except that Sequences are displayed only with the Sequence name, not with custom text for when the Sequence is Off or On. A T/F Indicator shows the value of one channel as either True or False, where the False value is defined and any other value is interpreted as True. (Generally, the value of 0 should be used for False, as that is how MICAS-X interprets any channel when it is used as a T/F in other contexts.) An Enumerated List allows a list of text values to be associated with a list of numeric values. These lists can be useful for displaying the state of a system, when the state is represented as a defined list of integer values. E.g. 0 = Stopped, 1 =

Stand-by, 2 = Running, 3 = Error. Enumerated List Indicators only display the value of a channel as a text value. Enumerated List Controllers allow the operator to set the value of a channel based on the text value, which is then converted to a numeric value within MICAS-X. Note that both types of Enumerated Lists automatically include a text value of "(other)", which is used to display the channel if that channel gets a value, from another part of MICAS-X, which is not included in the list. See Section 7.4.12 for information on configuraing Enumerated Lists.

As with all Displays, the Multi Display works only in MICAS-X for Windows and is not supported in MICAS-X-RT.

7.9.4 Recipes.vi

The Recipe Display allows the end-user of MICAS-X to create and run Reicipes, e.g. sequences of commands. The primary purpose of this Display is to allow an end-user who is not familiar with the subtle details of MICAS-X, including Sequences and Commands, to create Recipes at run time so that new sequences of operation can be executed with relative ease. Note that once a Recipe has been created, it appears in MICAS-X as a Sequence, and can be viewed and interacted with as any other Sequence, including through the Sequence Display.

Recipes are made up of Recipe Steps and Commands. Recipe Steps are pre-defined Sequences that have been designated specifically as Recipe Steps. In general, it is advised to have someone familiar with MICAS-X create these Recipe Steps as Sequences in the Sequences Editor. Once they are defined, they can be used by the end-user of MICAS-X as described below to create flexible variations on test sequences.

After having created the basic Recipe Steps as Sequences in the Sequences Editor, use the Recipes Editor to designate these Sequences as Recipe Steps. This is done in the Steps array on the right side of the Recipes Editor. Press the green Insert button next to the first unoccupied Steps entry to insert a new Step. Then click on the Sequence parameter to get a drop-down list of all the Sequences defined in the current configuration file. Select that Sequence that you wish to designate as a Recipe Step. (Note that any Sequence should only be added once as a Recipe Step, and not all Sequences need to be designated as Recipe Steps.) It is always a good idea to use the Description field to document the intended purpose of this Recipe Step.

Check the "Wait Until Done?" parameter if you want the Recipe to finish this Step before continuing on to the next Step. Checking this parameter is the most common situation. If a Step executes a single, very fast Command, such as setting a channel to a new value, then it may not be necessary to check this parameter.

The "Parameter Channel" is optional. It can be left as "--" to ignore it, or it can be set to any output (Controller) channel in MICAS-X. If a channel is selected, then when the step is run, that channel will be set to a specific value before the Recipe Step

executes. To understand the use of this parameter, consider a Recipe Step that sets the temperature setpoint of a heater to a value. This Recipe Step may have a single “Set” Command, or could include other Commands as well. The name of this example Recipe Step is “Set Temperature”. When this Recipe Step is used in a Recipe (as explained more later), there will be a value associated with each instance of it. Thus the same Recipe Step can be used first to set the heater setpoint to 30 C, and it can be used later in the same Recipe or a different one to set the heater setpoint to 50 C.

In the Configuration editor, Recipes can only be deleted, not created. (Recipes are created at run-time within the Recipes Display) To delete a Recipe, click the “Delete” button to the right of the Recipe to be deleted. The “Recipes” list displays all recipes and their descriptions. The “Test Recipe” parameter defines which recipe will be in the Test Recipe control at run-time. “Update Time” determines how often the Recipes display is updated. If it is left as 0, it will update once a second.

In the Recipes Display are two tabs, the Excute tab and Recipe Editor tab. Above the tabs is the Test Recipe control. Use this to select a Recipe to run, then press the Start button to the right to execute that Recipe. The Recipe Steps array on the Execute tab will show the Steps in that Recipe, and the blue arrow will indicate which step is being executed. The Execute tab also has a 2 channel graph for displaying time series of any two channels of data.

The Recipe Editor tab in the Recipes window of MICAS is used to create Recipes. In order to create a recipe, press the “New” to start a Recipe from scratch, or press the “Copy” button to use an existing Recipe as your starting point. Name the recipe and describe it with the “Description” box. Add steps with the insert buttons. The Label field is used as a description of the step as well as for the Target for any Goto commands. Step Type determines if the step is a Recipe or Sequence step. If it is a sequence step, the Command, Channel, Value, Condition Condition Channel, and Threshold parameters will be available. These are the same parameters used with the Sequence editor (7. 8. 4). When the step is a Recipe step the Recipe Step, Parameter Channel and Value parameters are available. The Recipe Step parameter allows for the selection of which Recipe Step to run, and the Parameter Channel displays the channel assigned in the configuration editor. The Value determines the value to assign to the channel in Parameter Channel when the Recipe Step is run.

7.9.5 XRT Sequences Display.vit

This Display is very similar to the Sequences Display. It runs on a Windows version of MICAS-X, but displays the Sequences that are running on a MICAS-X-RT embedded program. For this display to work properly, the Sequences Driver must be included in the MICAS-X-RT configuration. From this Display, the current status of each Sequence on the RT platform can be viewed, and Sequences can be started, stopped, and paused. Note that this display updates only as often as data is acquired on the

Windows host via the MICAS-XRT Driver, so the updates on this Display are usually less responsive than on the standard Sequences Display.

If Sequence Sets have been configured, a Sequence Set control appears on the right of this Display. This control can be used to select which Set of Sequences is currently displayed.

As with all Displays, the Sequences Display works only in MICAS-X for Windows and is not supported in MICAS-X-RT.

7.9.6 XYZ Graph.vit

The XYZ Graph Display is used to plot color-coded data vs both x and y axes. For example, one can use a Longitude channel for X, a Latitude channel for Y, and a temperature channel for Z to graph a geographic distribution of temperatures.

Select which channel list to pick observed channels from by using the drop-down menu of the 'Channel List' box at the top of the screen. Although the channels being graphed can be changed while MICAS-X is running, only those channels that are part of the selected Channel List will be available. If it is important for the user to be able to access any and every Channel, the "All" List can be selected, though be aware that it can be difficult to find specific Channels in the list of All Channels when MICAS-X is configured with many Channels.

Below, under the heading of 'Channels to Display', select which channel is to be displayed on each axis by selecting the desired channel in each of the drop-down menus of Channels. Select minimum and maximum values for this channel (when manually scaled) by entering the desired numbers in the '_ Min" and '_ Max' parameters. Complete this process for the X, Y and Z axes. Also set the Auto-Scale/Manual Scale parameters to the scaling behavior that you wish the axes to have when the program starts.

To define how much time history should be displayed in the XYZ graph, select the desired length of time from the 'History Length' drop-down menu in the upper-righthand portion of the screen. Note that "Manual" and "From Pointer" are not allowed as initial values for this parameter.

The 'Marker' switch below this will, if clicked, enable the Marker functionality on the XYZ Graph while the Program is running. The Marker allows the operator to place one reference marker, either at the most recent (current) value, or at an arbitrary location specified by the operator. The 'Mark Current Value' switch, located below the 'Marker' switch, if clicked true, will mark the current X,Y, Z value with a bright green X.

As with all Displays, the XYZ Graph Display works only in MICAS-X for Windows and is not supported in MICAS-X-RT.

8 Tools

The Tools entry in the Components menu does not contain editor windows that configure specific parts of the configuration file. Instead, it present a number of special tools that can be of use when editing a configuration.

8.1 Channels

The first entry in the Tools list in the Configuration Editor is “Channels”. The Channels screen a tool for reviewing and checking all the Channels currently defined in the configuration being edited. The left-most array lists all the Channels in the configuration. The second array lists the input Channels, and the third array lists the Controllers, or output, Channels.

To the far right on this screen are two additional arrays: Conflicting Channels and Partially Conflicting Channels. Conflicting Channels lists any Channels that were found to exist twice with the same spelling in the current configuration. Partially Conflicting Channels lists any Channels such that one Channel is a sub-string of another Channel.

In addition to simply creating confusion, Conflicting Channels can be problematic in MICAS-X, since, for example, it is indeterminate which Channel would be set if a “Set” Command were executed with the duplicate Channel name as the target.

Although Partially Conflicting Channels have unique names, they can cause problems if the Equations Driver is used and references a Channel with a Partially Conflicting name. Take for example the case where there is one channel named “Voltage” and another channel named “High Voltage”. Also assume that there is an Equation defined as:

Output Volts = 10 x High Voltage

When the Equations driver parses this equation, it could potentially find the word “Voltage” and replace that word with the value of the “Voltage” channel. It would be left with the word “High” which did not correspond to any operation or Channel, and would therefore cause an error in the parsing.

Use the Channels screen to review your configuration and ensure that each Channel in your configuration is unique. Furthermore, if you are using the Equations Driver, use the Partially Conflicting Channels list to ensure that no Channels that are used in Equations have Partial Conflicts.

Note that some Partially Conflicting Channels are unavoidable. For example, the Controllers Driver can be used to configure PID loops. In this case, a Mode Channel is automatically created for each SetPoint Channel, so that the loop can be changed between Manual and PID modes. The Mode Channel has the same name as the SetPoint Channel, but with the word “Mode” in front of it. Hence the SetPoint Channel and its Mode Channel will inevitably be Partially Conflicting Channels.

8.2 Rename

The Rename is used to rename Driver channels in more complex configurations. Sequences, triggers, and options reference channels by name. If any of these items have been configured, and then a channel name is changed within a Driver configuration, the new name will not be propagated to all the references throughout the configuration file. The Rename Tool is intended for handling this situation. Here, you first select a Driver. The Editable Names list will then contain the names of the channels associated with that Driver which can be edited. (Some Drivers have pre-determined, hard-coded names which cannot be edited.) Select a Channel with the Editable Names menu, then enter the new name for that channel in the “New Name” option and press Change Name to update the configuration file so that all references to the channel are properly changed. The # of Instances displays how many changes will be made to the configuration file, which can be useful in verifying the change.

Note that the Constants and Globals are not traditional Drivers, but are built in to MICAS-X at a lower level. Thus Constant and Global Channels cannot be renamed using this tool. If you rename a Constant or Global, you must manually find all references to it in the configuration file and change them as well.

The Rename Tool also has a full text display of the current configuration file, along with a “Search For:” field. It can often be useful to search for a Channel name, then use the Find First and Find Next buttons to display where that Channel name appears in the configuration file before renaming the Channel. In addition, any string can be searched for using this Tool, so although you cannot rename Sequences or Triggers with this Tool, one can search for all the references to those items to help decide how to deal with them.

8.3 Rename Prefix

The Rename Prefix Tool can be used to take an existing Driver that is already configured in MICAS-X and assign it a new Prefix. If you just change the Prefix of a Driver that is already configured by using the Acquisition Editor, all the parameters for that Driver will be reset to their defaults. Using this Tool allows the Driver to be assigned a new Prefix without losing its configuration

8.4 Alert Calc

This configuration panel allows you to determine the proper value to be used for the Alert Command, to achieve the desired functionality. See Section 15.1.1 for an explanation of the values used with the Alert Command. This panel also lets you test the .wav files used for Alert sounds. Note that any desired .wav file can be renamed appropriately and placed in the Resources directory to use that sound with an Alert.

8.5 Channel Scaling

This configuration panel allows one to calculate scaling coefficients to use with various Drivers. Currently, this panel includes a tool for converting two data pairs to a linear scale. E.g. high and low values are entered for the measured data and for the corresponding data in engineering units, and the appropriate slope and offset are calculated which can be used to scale the data from measured units to engineering units.

For example, assume a pressure gauge outputs data from 0 to 10V which, and has a full scale range of 15 psi. Then the values

Hi Measured Value =	10
Low Measured Value =	0
Hi Engineering Value =	15
Low Engineering Value =	0
will yield the results:	Slope = 1.5, Offset = 0

A tool for calculating a least squares polynomial fit to sample data is also provided. When calibration data is available as a table, this tool can convert the table into a set of polynomial coefficients which can be used to scale a channel

9 MICAS-X Operation

This chapter covers MICA-X for Windows. See Chapter 10 for information on the operation of MICAS-X-RT.

Most of the functionality of the MICAS-X program is contained in a single window named “MICAS-X”. A series of tabs allows numerous displays to be presented within this window. Menu items and a row of buttons and indicators above the tabs provides certain functionality that is always present regardless of which tab is being viewed.

9.1 Menus

9.1.1 Program Menu

The Program menu contains up to six items:

- Open Support Folder – Selecting this menu item will open the MICAS-X support folder, located under C:\OCC\MICAS-X Support.
- Open Data Folder – This menu item will open the current data folder, located under C:\OCC\MICAS-X Data.
- Quick Graph – This menu item opens a dialog that allows you to configure and launch a time-series graph window.
- Save Screen to JPEG – This menu item takes a screenshot of the current window and saves it as a .JPEG image on your computer in the current data folder.
- Save Screen to PNG – This menu item takes a screenshot of the current window and saves it as a .PNG image on your computer in the current data folder.
- Log a Message – This menu item is used to present a floating dialog that allows the operator to write any message or comment and send it to the log file. This allows documentation of the circumstances during a program run.
- Password – This menu item is only present when a password has been defined for the executing configuration file in use. It is checked when the password is being enforced, and is unchecked when the password is not being enforced.
- Exit – Closes MICAS-X and quits the program.

9.1.2 Drivers Menu

This menu has one entry for each Driver present in the current configuration, as well as the menu items “Enable All Drivers” and “Disable All Drivers”. This menu is used to enable or disable any or all Drivers. Any Driver that is enabled has a check mark next to it, and its name is followed by “ - Enabled”. And disabled Driver does not have a check mark next to it, and its name is followed by “ – Disabled”. The “Enable All Drivers” and “Disable All Drivers” menu items at the bottom of the menu can be used to affect all the Drivers at once.

9.1.3 Windows Menu

This menu allows quick access to any tab of MICAS-X or any Display or Instrument window that is configured to be "Outside" the main tab structure. Each tab and each "Outside" Window will have an item in this menu, which when selected will cause that tab or window to be displayed frontmost. Note that these menu items refer to the tabs and Windows by their "Tab Name" parameter. If the Tab Name parameter is blank for an Instrument or Display, the Instrument or Display name, with prefix, is used for the Tab Name. Note that if two or more tabs or Windows have the same Tab Name, none of those tabs or Windows will respond to their corresponding menu item.

9.1.4 Utilities Menu

This menu includes options to run each of the Utilities that have been installed for MICAS-X. These typically include the Configuration Editor, Data Reader, Log Reader, and the TDMS File Reader, though other utilities may also be present. All utilities may be run either inside MICAS-X (as when they are selected from this menu) or in the Utilities tab or independently, either in source code inside LabVIEW or as executables.

9.1.5 Help Menu

Within this menu, the Open User Manual item opens the pdf copy of the MICAS-X User Manual, if it is installed in the correct location. The Open Web Page item opens the MICAS-X web page in the computer's default browser. The Help item provides information on links on where to find help for the MICAS-X program.

Note that in addition to the resources mentioned on the Help window, MICAS-X provides context-sensitive help in many places. Context-sensitive help is a LabVIEW feature that is accessed by pressing CTRL-H to open the Context-sensitive help window. Once this floating window is open, position the mouse over a control or indicator in MICAS-X to view a short description of that item. Context-sensitive help is most useful in the Configuration Editor, since it provides brief explanations of how each parameter functions. Within MICAS-X itself, the context-sensitive help is frequently less useful, since the user interface of MICAS-X is, by its design, very general. A control or switch can be configured to work with any Controller channel, so the context-sensitive help for that control or switch cannot be specific to any particular channel or its function. The Context Help Window item in the Help menu functions similarly to pressing CTRL-H, and will open or close the context-sensitive help window.

Use the Send Feedback menu item to send information about your experience with MICAS-X to Original Code Consulting. Please enter your name, affiliation, and contact information, especially if you would like a reply from OCC. Note that this function requires that the computer running MICAS-X have a connection to the internet. After pressing the Send button, MICAS-X will attempt to send the feedback to OCC, and

will then tell you whether the information was sent successfully or not. If a reply is requested, OCC will try to respond to feedback within 48 hours. If you do not hear from OCC within that time, please send your feedback or a follow up message directly to OCC at support@originalcode.com.

The Send Bug Report menu item is similar to the Send Feedback described above, and like the Send Feedback function, this feature only works if the computer running MICAS-X has an active internet connection. In addition to including information submitted by the end-user, the Bug Report also allows the user to have the program automatically include a copy of the Configuration File and the last 20 entries of the Log file. Both these additional sources of information can be invaluable for determining the cause or fix for a bug or problem. OCC will try to respond to every bug report within 48 hours. If you do not hear back from OCC after successfully sending a bug report, pleace contact OCC directly at support@originalcode.com.

The Check for Updates item contacts the OCC servers to determine the version number of the most recent MICAS-X release. This feature also only works if the computer has an active internet connection. The About item provides information on the version of MICAS-X being run as well as on the security device that is currently detected.

9.2 Buttons

Below the file menu, there are up to twelve configurable buttons, eight in a row across the top, and four more immediately below them on the right half of the screen. The second row of four buttons is useful if only a limited number of tabs is configured, as these buttons appear horizontally in line with the tabs. If any Alarms are configured, the Alarm indicator appears in place of the right-most of these buttons, so the eighth button in the top row should not be configured for use. The labels and functions of these buttons is determined in the configuration file and can be customized for each specific application. Below are some common examples of how the buttons may be configured.

- Acquire – Begins data collection.

- Record – When Record is True, any .csv files defined in the configuration file will be recorded whenever data is being acquired. Note that the background .tdms master data file is being recorded whenever Acquire is True, regardless of the state of the Record button.

- Restart – Restarts the the program, loading any changes that have been made to the configuration file.

- Exit – Quits MICAS-X.

If any Alarms have been defined in the configuration file, the Alarm Status is displayed where the 8th button normally would appear. The Alarm Status is either green (OK), yellow (Warning), or red (Alarm), depending on the highest state of any of the defined Alarms. E.g. if any one (or more) Alarm is in the Warning state but all others are in the OK state, the Alarm Status will be yellow. If any one (or more) Alarm is in the Alarm state, the Alarm Status will be red. The Alarm Status indicator gives the operator one place to look that is always visible, regardless of what tab is being displayed in MICAS-X, to see if any Alarms have been triggered. If the Alarm Status is not green, the operator can clock on the Control tab to view more detailed Alarm information, or on the MICAS-X tab to review the log of Alarms.

To the right of up to eight buttons, there is an indicator that displays the current date and time.

9.3 Tabs

Below the file menu and buttons, there are a number of tabs. More tabs than the following may be present, depending on the configuration being used. Up to 18 tabs may be used at once. Often included are the following:

- Control
- Displays (Up to 14 Displays can be configured to load in these tabs.)
- Utility (The Utility tab will not be visible until the first Utility is started.)
- MICAS-X
- Debug (The Debug tab will not be visible until the Debug mode is enabled.)

Each tab and its functionality are described below.

9.3.1 Control Tab

The Control tab is the first (leftmost) tab. Note that this tab can be given a different name than “Control” by editing the “Control Tab Name” parameter in the Control Tab editing window.

This tab displays a list of Channels and their current values on the right. (The selection of channels that are shown here can be configured. See Channel Lists, in the Modules section of the Configuration Editor help.) The bottom of the tab can display an optional graph, which can display time series of any two Channels of data. If additional Channel Lists have been defined (besides the default “All” list), a two list selections will be visible above the channel table and the graph. These controls can be used to select which list of channels is shown on the data table and which list of channels are available for the graph.

Above the graph (and continuing down into where the graph is displayed if the graph is disabled) are displayed numerous controls. These include Controllers (user-settable output channels), Switches (Controllers that toggle between two discreet values, such as digital outputs), Sequences, and Triggers.

If enabled, a Log Message text box will appear in the lower right of the Control tab, under the Channel values list. This allows the operator to log notes specific to the circumstances while the program is running, similar to the “Log a Message” menu item in the Program menu.

9.3.2 Display Tab(s)

To the right of the Control tab are up to 14 Display tabs. Displays can be configured to populate any number of these tabs. Displays can also be configured to open up as additional windows outside of the main MICAS-X program.

9.3.3 Utility Tab

The Utility tab is to the right of the Display tabs. Only one Utility can be invoked at a time. Start Utilities by selecting them from the Utility menu. Selecting a new Utility will close any Utility that is already open.

9.3.4 MICAS-X Tab

The MICAS-X tab follows the Utility tab (though the Utilities tab is not always visible). This tab provides a summary/status view of the MICAS-X program. It displays the current configuration file, the names of any data files being written, a list of log file entries (errors and events), a list of all data channels, the status of the hardware security

device, and the MICAS-X serial number and version. It also provides controls that allow any Sequence to be started or stopped, any Controller to be set to a new value, any Command to be executed, or a custom message to be sent to the log file. Each of these last functions has a small check-box labeled “A-Z” next to them. When this box is checked, the drop-down list of Commands, Sequences, or Channels will be alphabetized rather than appearing in the order defined by the MICAS-X configuration file.

9.3.5 Debug Tab

The Debug tab appears on the far right when MICAS-X is in debug mode. This tab does not normally appear and is not needed for normal MICAS-X operation. This tab is useful for debugging MICAS-X when it is being run inside LabVIEW as source code. To enter debug mode and make the Debug tab appear, click on the small black square in the upper right corner of the MICAS-X tab.

9.4 A Note on MICAS-X Window Size

Note that the MICAS-X program has many controls and indicators that are very carefully placed on the front panel, so that they can be made visible and invisible as required by the configuration. Because of the dynamic nature of this layout, the MICAS-X.vi window cannot be resized or maximized. MICAS-X is set to not allow resizing or maximizing for this reason. Disabling these settings is not recommended. One exception to this rule is that Displays can be configured to open in their own window. This allows for Displays to be designed with any window size. However, if a Display is designed with a window size other than that defined for operation within MICAS-X, it should always be configured to open in an external window and never configured to open in a MICAS-X tab.

9.5 Time Series Graphs

Time series graphs of Housekeeping channels appear in several places in MICAS-X, including (optionally) at the bottom of the Control tab. All time-series graphs in MICAS-X have several common properties. In particular, MICAS-X uses a sophisticated buffering mechanism to provide a useful length of history data without the overhead of buffering all housekeeping channels in memory at once. Whenever MICAS-X is Acquiring data (regardless of whether or not it is set to Record data), the data acquired is stored in a special database-like file, using the .tdms file format. Furthermore, each graph has its own buffer of acquired data for the channels currently being displayed. If the operator changes any of the channels being displayed on the graph, MICAS-X reads the history for the newly-selected channel from the .tdms file(s). In this way, only the channels being

displayed are buffered continuously in memory. At the same time, access of the .tdms file is minimized to reduce performance degredation related to parsing large data files.

Every three hours, a new .tdms file will automatically be started. This three hour limit is imposed to keep each .tdms file from growing too large, which would impact performance when reading history data from it. In the MICAS section of the configuration, there is a parameter that determines whether 1 or 2 .tdms files are read when a new channel is selected to graph. If this is set to “Only read 1 file of History data”, then when a new channel is chosen for a time series graph, at most 3 hours of data will be available. If a new .tdms file has been started recently, the amount of history data could be much less. By setting this parameter to “Read 2 files of History data”, the time-series graphs will be guaranteed to have between 3 and 6 hours of history data (except for the first three hours of run time, when less than 3 hours of history data has been accumulated.)

If a large number of housekeeping channels are configured in MICAS-X, it may be advisable to set the above parameter to “Only read 1 file of History data”, since reading history data from two large files could affect system performance.

If the operator does not change the channels being graphed on a time series graph, the history can accumulate much longer than the 3 or 6 hour limit. In this case, the maximum history depth is determined by the “Max History Depth” parameter on the MICAS configuration page. This parameter is set in “number of points”, not seconds, so that actual history time depends on both this parameter value and the acquisition rate of the housekeeping data.

Another feature of time-series graphs in MICAS-X is a global per-channel memory of manual Y axis limits. Each time a graph channel is set to use a Manual Y scale, the selected Y axis limits are stored in a buffer. Whenever a new channel is selected for display with a manual Y axis limit, the buffer is scanned. If that channel is found in the buffer, the manual limits for that channel are automatically restored. Note that the buffer of channel limits is limited to between 90 and 100 channels. When more than 100 channel limits are stored in the buffer, the oldest 10 channels are removed from the buffer. This self-cleaning mechanism prevents the buffer from growing unbounded over long term use and thus degrading performance. Note that the manual channel limits buffer is used for the time-series graph on the Control tab and for the time-series graphs in the 3Graphs Display. It may be added to other graphs and Displays in the future, but it is generally not used for graphs that allow multiple channels on an axis.

9.6 Modules and Features

As described in the "MICAS-X Components" section of the manual, Modules are sets of functionality that are built into MICAS-X. Modules are not deployed as plug-ins, so new modules are only available when a new version of MICAS-X is released. Most modules have their own configuration editor. Modules currently available in MICAS-X include, but are not limited to:

9.6.1 Acquisition

The Acquisition module coordinates acquiring data from the Drivers. This module has one primary loop rate, but can include any number of additional acquisition loops. The various acquisition loops run in parallel and each acquires data from the Drivers assigned to it at its own rate, unsynchronized from the others. MICAS-X has a single buffer of all Housekeeping Channels (Input channels and Controller or Output channels) which each acquisition loop updates whenever new data is read. By using this single buffer, any part of MICAS-X can access the most current values of any channel. In addition, the primary Acquisition loop writes the contents of this buffer to the master .tdms database file on each iteration.

Typically, the primary Acquisition loop will run at 1 Hz, providing 1 second data from all the Drivers. There are several reasons why additional acquisition loops may be desired. One example of this is a situation where one Driver is acquiring 1 Hz data from a National Instruments data acquisition card, while another Driver is acquiring 1 Hz data that is being sent by an external computer or instrument at a rate determined by the external device. Although both Drivers are operating at a nominal rate of 1 Hz, they are using two different clocks. Eventually, one of these clocks will lag the other to the extent that one instrument will not have data available on within the time limit of the loop, and an error will occur. By assigning these Drivers to separate loops in MICAS-X, the error condition will be avoided. The primary Acquisition loop will record the most recent data values for each Driver. Eventually, the Drivers will become out of sync such that the primary Acquisition loop will record the same value for one of the Drivers on two consecutive iterations, or possibly the primary Acquisition loop will miss one value of one Driver if the second loop's clock is faster than the primary Acquisition loop's clock. This type of stuttering is unavoidable when using devices with different clocks, but by using separate acquisition loops, no error condition is generated. Also note that when a Driver reports its channels, it prepends all the channel data with a time stamp channel indicating when that data was taken. Stuttering of the type described above can therefore be detected by analyzing the Driver time stamps.

Another example of when multiple acquisition loops can be useful is when one Driver may be acquiring 1 Hz data from a digitizer or multi-function card, while another Driver needs to acquire data from an external instrument that sends data at a higher or lower rate. E.g.: if an external instrument sends data at 5 Hz, and cannot be configured

to run at 1 Hz, then two acquisition loops can be configured in MICAS-X so that one runs at 1 Hz, and the second runs at 5 Hz. The primary Acquisition loop will gather data from all the Drivers at its own rate, typically 1 Hz.

Finally, separate acquisition loops are often required when some devices impose their own timing on the data. For a simple A/D voltage read from an NI device once a second, MICAS-X can determine when to ask for the data. However, for a streaming device such as a GPS, which sends data once a second based on its own clock, it is important to have a separate loop that is "Device Timed", so that the acquisition rate of that loop is tied to the rate at which the device makes data available. For more information on Device Timed acquisition loops, refer to the Channels configuration section.

9.6.2 Channel Lists

Channel Lists are defined in the configuration. A list named "All" is always present and includes all the Housekeeping channels present in the configuration. As many additional Channel Lists can be created as the operator has use for.

Channel Lists are a feature of MICAS-X that have several purposes. Channel Lists can be used to define which Channels appear in conjunction with various front panel display objects. Channel Lists can also be used to define which channels are written to data files, or which channels are broadcast to another system. By creating custom lists of channels, it is possible to provide a much more intuitive and navigable user interface, especially when the total number of Channels defined in MICAS-X begins to get large. When dozens or even hundreds of channels are present, selecting which channel to graph on a time-series graph from a list of all channels can be tedious and frustrating. Channel Lists provide a way to filter the channels so that only those relevant to a particular use case are present where they are needed.

9.6.3 File Writer

Whenever MICAS-X is acquiring data, it writes the complete set of channels to a database-like .tdms file. This file is used as a fail-safe data archive as well as to provide history data for time-series graphs. This file is written by the main Acquisition Loop (loop 0). In addition to this data file, the File Writer module can be configured to create any number of text data files. Each data file can be configured to write a specific list of channels. In addition, other configuration parameters can be set that determine how the data file is formatted. Each file defined in the File Writer is assigned to an Acquisition Loop. Whenever the assigned loop acquires data from its drivers, it then also gathers the most current data from all the Drivers and writes any files that are assigned to it.

Note that the ability to assign each file to a different acquisition loop can be used to significant advantage when there are Drivers that acquire data at different rates. As an example, assume that most drivers acquire data at 1 Hz, and that Loop 0 of the Acquisition Task is writing not only the .tdms file at 1 Hz, but also writing one or more text files at 1 Hz. Also assume that one particular device sends data at 10 Hz, and is set to be acquired using Loop 1. Only one of every 10 samples of this driver's data will be recorded in the .tdms file and in any text files written by Loop 0. However, a Channel List can be defined that includes just this Driver's channels (or possibly other channels as well), and assigned to a file definition that is associated with Loop 1. In this case, a text file will be written that includes the selected channels, and it will write all 10 samples per second, so that no data from the fast device is lost.

The opposite case, creating data files with fewer entries than a loop time, can be handled with the Write Frequency (Once Every n Loops) parameter. E.g. a sparse data file with data written once a minute can be defined by associating it with an Acquisition Loop which runs once a second, and setting the Write Frequency to 60.

The "All" channel list is always present in MICAS-X. The first channel in this list is named "–" and always has a value of "NaN". This channel is used in several ways in MICAS-X. One example is that when selected as a graph channel, it prevents the graph from plotting any data for that channel, which can be an easy way to use a two-plot graph to only show one plot. Note that when the "All" channel is written to a user-defined file, the "–" channel is not included in the channels written to the file.

In order to preserve sub-second information in time-stamps, the File Writer automatically uses a specific formatting when writing data that appears to be time-stamp data to a data file. Refer to section 7.4.4 for more information.

9.6.4 Sequences

Sequences provide a powerful way to extend the functionality of the MICAS-X system. Sequences can be defined that can automate many processes. Sequences can loop and execute conditionally, and can act on any Controller (output) channel, based on the value of any channel. Sequences can use the many built-in MICAS-X Commands, as well as any Commands that are programmed into any of the Drivers. Sequences can be tied to various buttons in the MICAS-X interface, to Triggers or Alarms, and can be started via commands sent by another system.

Since Sequences have no “Hysteresis” parameter, the In-Range and Out-of-Range comparisons that are described in the Triggers section below are not available for Sequence conditions. For information on the “NaN” conditions, see section 7.4.8.

The Sequence Module can be used with or without the Sequences Driver. The Sequences Driver should only be used when the Sequences Module is used. The Sequences Driver adds additional functionality to the Sequences Module. When the Sequences Driver is included, new channels are created that are logged to the main data file. These channels are named “SeqStep” plus the name of each Sequence. These Controller (output) channels that can be used to start and stop Sequences. In addition, as a Sequence runs, the value of its SeqStep channel is updated with the step number that the Sequence is currently on.

The StartSeq command can be used to start a Sequence. Normally, the Value parameter is set to 0, so that the Sequence starts at the beginning (step 0), but it is possible to use a non-zero value to start the Sequence at any step. The StopSeq command is used to stop a Sequence that is running. You can also use the “Set” command with one of the SeqStep channels to start and stop Sequences. Using the value of 0 with the Set command and a SeqStep channel will start that Sequence at its beginning (step 0). Using a positive integer value will start the Sequence at a step part way through the Sequence. Using the value “NaN” will stop the Sequence. Using the negative value of the current Sequence step will Pause the Sequence at its current step. (Any negative number will pause the sequence, after which the SeqStep channel will automatically take on the value of the negative of the step at which is was paused. E.g. you cannot force the Sequence to pause at a different step then the one it was executing.) The PauseSeq and UnpauseSeq commands can also be used with a Sequence name to pause and unpause a Sequence.

Each Sequence can have an Exit Step defined. If this option is enabled, the Exit Step (which has the same form as any other Sequence Step) will be executed automatically whenever an action external to that Sequence causes the Sequence to stop. For example, if another Sequence, a Trigger, or a button issues a StopSeq command, the Sequence that is stopped will execute its Exit Step. If, however, the Sequence runs to completion and stops itself, the Exit Step is not automatically executed.

The Exit Step can be used to clean up the action of a complex Sequence which may be stopped at any point, and which would otherwise leave the system in an unknown state. Often, the clean-up procedure will involve many steps. In this case, the Exit Step will need to be a StartSeq command which will call another multi-step Sequence to complete the clean-up.

It may be desirable to execute the Exit Step even when the Sequence runs to completion. In this case, you can either duplicate the Exit Step at the end of the Sequence, or call the “ExitStep” Command, which will then execute the Exit Step as if it were a regular step of the Sequence.

The Sequence Module can be used with or without the Sequence Display. The Sequence Display provides a tab in MICAS-X that displays all the Sequences and their current state, and allows the user to turn them on and off.

9.6.5 Sequence Sets

Sequence Sets can be useful in configurations which contain a large number of Sequences. By defining various Sets, the Sequences viewed with the Sequences Display and the XRT Sequences Display can be filtered, providing a much clearer view of what the Sequenes are doing. In addition, the StopSeqSet Command can be used to stop all the Sequences in a specific Set at once. This functionality can be useful when an error, trigger, or alarm requires that the system be set to a known state, regardless of which of several different Sequences may be running at the time. See section 7.4.7 for additional information on Sequence Sets.

9.6.6 Triggers and Alarms

Triggers are actions that can be defined to occur based on the value of any Channel. Various programmable conditions can be used to determine when a Channel value causes a Trigger to be True. Once a Trigger becomes True, it can execute any of the MICAS-X Commands, including any custom commands defined in Drivers. An Alarm is a special case of a Trigger. Any Trigger can be configured to be an Alarm. Alarms are given special consideration on the MICAS-X user interface, so as to alert the operator whenever an Alarm becomes True, whereas Triggers can be used to provide functionality which might not be considered an Alarm condition.

The Triggers Module can be used with or without the Triggers Driver. However, the Triggers Driver should only be used when the Triggers Module is used. The Triggers Driver adds additional functionality to the Triggers Module. When the Triggers Driver is included, new channels are created that are logged to the main data file. These include a channel for each Trigger (using the Trigger Name as the channel name) which logs the state of the Trigger (0 = False, 1 = Warning, 2 = True, 3 = Alarm), and a Controller (output) channel for each Trigger that allows the Trigger Threshold to be set. These are named using “TrigThr “ prepended to the Trigger Name.

Changing Trigger Thresholds while MICAS-X is running can be used to disable Triggers (e.g. set the threshold to Inf, so that > is never attained), or to fine-tune the

Trigger's behavior. There are two ways to set Trigger Thresholds: using the SetTrigger command, or by using the "Set" command with the Triggers Driver channel. (The second is only available, of course, if the Triggers Driver is included in the MICAS-X configuration.) Note that the SetTrigger command can accept either the Trigger Name itself, or the Trigger Name prepended with "TrigThr ", e.g. the Trigger Threshold Controller channel.

Triggers are only active when data is being acquired. When the program is not Acquiring data, the Alarm Status and the Triggers display on the Control tab are grayed-out.

Note that for the conditions = and <> (not equals), the Hysteresis parameter can be used to allow for "In Range" (for =) and "Not In Range" (for <>) comparisons. For =, a Hysteresis of 0 will enforce a strict = condition, whereas a positive, non-zero Hysteresis will create a "In Range" condition. E.g. if the Threshold is 10 and the Hysteresis is 2, the = condition will cause the Trigger to be True whenever the channel value is between 8 and 12. For <>, a Hysteresis of 0 will enforce a strict not-equal condition, whereas a positive, non-zero Hysteresis will create an "Not In Range" condition. E.g. if the threshold is 10 and the Hysteresis is 2, the <> comparison will be True whenever the channel value is < 8 or > 12.

Three special conditions are included for dealing with NaN, or Not-a-Number. See section 7.4.8 for more information.

When viewing Triggers on the main MICAS-X Control tab, hover over one of the Triggers to see additional information on that Trigger's status, which will appear on top of the Channels list. The Channels list will reappear when you move the mouse off of the Triggers display.

9.6.7 Drivers

Drivers are at the heart of MICAS-X. A Driver obtains data from input channels and set the data values of output channels. Drivers are intended to operate on relatively slow data. 1 Hz "housekeeping" or instrument health data is typical, although Drivers can operate as fast as 20 or 25 Hz or much slower than 1 Hz. The channels from all the Drivers in MICAS-X are combined into an array of data this displayed and saved. Many of the functions of MICAS-X act on this data. Sequences can set output channels. Triggers and Alarms initiate action when a channel's data value matches some criteria. Although instruments provide a way to integrate complex hardware and data into MICAS-X, many systems can be implemented with just the data from Drivers.

If a Driver is written correctly, it can be multiply instantiated within MICAS-X. This means that the same Driver source code can be called multiple times within MICAS-X, with each instance working with a different instance of hardware and creating a separate set of channels. The Prefix parameter, which can be set for each Driver, is a requirement when multiple copies of the same Driver are used within a configuration. The Prefix assigned to each instance allows MICAS-X to differentiate the instances and manage them properly. In addition, the Prefix is prepended to the channel names, so that each Driver instance has uniquely named channels.

Drivers that are currently available for MICAS-X are listed below. Note that some of these Drivers are included with the MICAS-X base package, while others incur an additional charge.

- 2D Array – The 2D Array Driver allows one to create a two-dimensional array of numeric values. These values can be read and written to programmatically throughout the MICAS-X program, similarly to Array. This driver can be multiply instantiated. (additional charge)

- Agilis – This driver controls Agilis motorized mirror mounts. This Driver can be multiply-instantiated. (additional charge)

- Airmar – This Driver can interface to the Airmar 200WX weather station, to acquire GPS, magnetic heading, winds, and meteorological data. This Driver can be multiply-instantiated. (additional charge)

- Alicat – The Alicat Driver can interface to multiple Alicat Flow Controllers and can be used to set the flow setpoint and read back the flow (in volume and mass flow) and the pressure and temperature. This Driver can be multiply-instantiated. (additional charge)

- Aries Drive – This Driver controls one axis of motion using an Aries IPA controller from Parker Hannefin. This Driver includes several commands, which are documented in Appendix B: Commands. This Driver can be multiply-instantiated. (additional charge)

- Array – This Driver allows one to create a one dimensional array of values, which can then be read and written to programmatically throughout the MICAS-X program. Such an array could be used to direct a sequence to scan a process variable through a set of values that can not easily be written as an equation, such as a 1,2,5,10-type sequence. This Driver can be multiply-instantiated. (included in base package)

- Averager – This Driver averages the values of other, exisiting channels over a user-specified number of acquisitions. This Driver can be multiply-instantiated. (included in base package)

- Calculations – Each instance of this Driver can call a Calculation VI. A Calculation VI must adhere to the design specifications defined for MICAS-X in order to work in MICAS-X with the Calculations Driver. Such a VI can execute much more complicated algorithms than the scripted "Equations" Driver can handle. In general, a Calculations VI takes a number of channels as input and creates 1 or more channels of output data. A Calculations VI can be singly or multiply instantiable – e.g. if it is designed properly, a single Calculations VI can be instantiated multiple times within MICAS-X so that the same calculation can be performed on numerous sets of input channels. Likewise, the Calculations Driver can be multiply instantiated within MICAS-X. (included in base package)

- CANBus – This Driver allows communication to devices using the industrial CANBus protocol. This Driver can be multiply-instantiated. (additional charge)

- Controllers – This Driver creates control loops by taking an existing input channel as the process variable, an existing output channel as the control channel, and creating a new channel for the Set Point. Simple Controllers, PID loops, and Thermostatic Controllers are available options. In addition to those control loops, this Driver can also create new channels that scale an existing volume flow to a new mass flow channel, or an existing mass flow channel to a volume flow. Finally, this Driver allows the creation of "Follower" channels that output a voltage that is proportional to the value of any other existing channel. Follower channels can be used to send a channel value to another data system or program by encoding it as a voltage. (additional charge)

- Document – The Document Driver is only used in conjunction with the Document Display. It creates the channels "Document Number" and "Section Number" which allow MICAS-X commands to control the display of the information in the Document Display. (included in base package)

- Equations – This Driver executes a script-like equation on existing channel names, which result in a new channel of data. Multiple Equations can be configured within one Equations Driver, and the Equations Driver can be instantiated multiple times within the MICAS-X system. (included in base package)

- Expressions – This driver is very similar to the Equations driver, but is many times faster and includes many more functions. It can be multiply-instantiated. (additional charge).

- ESP7000 – This Driver controls the Newport ESP-7000 series motion control systems. (additional charge)

- Manual – The Manual Driver allows one to create manual data channels. These are Controllers (output channels, e.g. channels that can be set) that are generally used to allow the operator to enter values directly from the user interface. These values are then recorded, and hence can be used to annotate an experiment with manually entered parameters, or they can be used as inputs to calculations or sequences, allowing the operator to tailor the operation of the program by altering these values. (included in base package)

- MCCDaqAD – This Driver acquires analog input data from a wide range of Measurement Computing multifunction devices. (additional charge)

- MCCDaqDA – This Driver writes to analog output channels for a wide range of Measurement Computing multifunction devices. (additional charge)

- MCCDataDI – This Driver acquires digital input data from a wide range of Measurement Computing multifunction devices. (additional charge)

- MCCDataDO – This Driver writes to digital output channels for a wide range of Measurement Computing multifunction devices. (additional charge)

- MCCDataTC – This Driver acquires temperature data from Measurement Computing thermocouple devices. (additional charge)

- MICAS-XRT – This Driver acts as an interface between MICAS-X and MICAS-X-RT. All the channels and commands available on the Real-Time system can be accessed on the Windows MICAS-X when this Driver is included in the configuration. (additional charge)

- MICM – This driver implements the Instrument Communication Module(ICM) in MICAS-X, allowing a configuration-based mechanism for reading and writing data from a variety of instruments. This driver is similar to OSDS. This Drive can be multiply-instantiated. (Included in base package)

- Modbus – This Driver can be used to interface to a wide range of instruments which support the Modbus protocol, both over serial or Ethernet. (additional charge)

- MOSDS – The OCC Streaming Data System is a shareware library of LabVIEW routines that allow one to acquire data from a wide variety of devices by simply editing a configuration file. OSDS supports serial and UDP (Ethernet) data buses, and includes a wide range of parsing options. MOSDS is a MICAS-X Driver that incorporates the OSDS system into MICAS-X. The MOSDS Driver can be multiply instantiated. Note that the OSDS definition file is stored outside of the MICAS-X configuration file. (included in base package)

- myRIO Daq – This driver is only for MICAS-X-RT and only interfaces to the Daq signals on the myRIO platform. This driver can be multiply-instantiated. (additional charge).

- NIDaqAD – This Driver allows one to acquire analog input data from nearly any National Instruments multi-function device. It can be multiply-instantiated. (included in base package)

- NIDaqBridge – This Driver acquires signals from bridge-based sensors using National Instruments devices specifically designed for bridge sensors. Currently this driver supports force measurements. (additional charge)

- NIDaqCounter – This driver allows acquisition of counter data from National Instruments multi-function devices. It can be multiply-instantiated. (additional charge)

- NIDaqDA – This Driver allows one to set output voltages on the analog output channels of nearly any National Instruments multi-function device. Note that this MICAS-X Driver is intended for setting slowly changing setpoints, and does not support any hardware-timed function generation. It can be multiply-instantiated. (additional charge)

- NIDaqDI – This Driver acquires data from digital inputs on National Instruments multi-function devices. It can be multiply-instantiated. (additional charge)

- NIDaqDO – This Driver allows MICAS-X to use digital output channels on National Instruments multi-function devices as Controller channels. It can be multiply-instantiated. (additional charge)

- NIDaqRTD – This Driver interfaces with NI Daq devices that can read RTD temperature sensors. This driver can be multiply-instantiated. (additional charge)

- NIMotion – This Driver interfaces to National Instruments motion control systems. It can control an arbitrary number of motion axes. It provides input channels for reading position, velocity, and status. It provides output channels (Controllers) for setting the target position and velocity, and Commands for initiating movement, resets, etc. (additional charge)

- Obis Laser – This driver interfaces to Obis lasers from coherent. It can be multiply-instantiated. (additional charge)

- Omega – The Omega Driver reads one temperature value and sets one temperature setpoint on an Omega Cni16 temperature controller. Additional functionality can be added as needed, to handle more functions on the Cni16 or to work with other Omega products. This Driver can be multiply-instantiated. (additional charge)

- Picarro G2401 – This driver acquires data from a Picarro G2401 cavity ring down spectrometer. This driver can be multiply-instantiated. (additional charge)

- PID – The PID Driver creates PID control loops, similar to the controllers driver, but with more flexibility. It includes multiple gains for gain scheduling, ability to change gains while the program in running and an auto-tune wizard.

- PrimeScales – This Driver reads the weight from a PS-IN202 industrial scale made by Prime Scales. Its output is the net weight (tared), either by reading the net weight from the scale or by reading the gross weight from the scale and using a tare weight from its configuration. It can be multiply-instantiated. (additional charge)

- PWM – This Driver uses an existing channel to define the duty cycle to turn on and off an existing digital output channel. This driver can be multiply-instantiated. (additional charge)

- RIO Scan Engine – This Driver is only for use in MICAS-X-RT, and allows easy, program-free access to any channels that are supported by the Scan Engine. (additional charge)

- Sequences – This Driver adds additional support to the Sequences module of MICAS-X. Sequences can be used without the Sequences Driver. However, if the

Driver is included, two new channels are created for each Sequence that is defined. One of these channels records the step number that the Sequence is on. The second acts as a Controller channel that can be used to turn the Sequence on or off. (included in base package)

- System – The System Driver allows one to monitor the CPU and RAM usage, AC/Battery status, hard disk free space, and dozens of other computer resources. It also includes an optional timestamp channel that calculates time as seconds since January 1 of the current year. (additional charge)

- System-XRT: Similar to the System Driver for windows, but specific to RT platforms. (additional charge)

- Timers – This Driver can be used to create an arbitrary set of Timer channels. These channels use the computer clock as their time basis (software timing), and are useful for timing things from seconds to hours or more. They have sub-second resolution, but MICAS-X is not designed to respond to Channels with better than about 0.1 second accuracy. Timers can be reset, disabled, and paused. In addition to the user defined timers, this Driver also creates three program timers, which track the total time MICAS-X has been running, acquiring data, and recording data. (included in base package)

- Triggers – This Driver adds additional support to the Triggers module of MICAS-X. Triggers can be used without the Triggers Driver. However, if the Driver is included, two new channels are created for each Trigger that is defined. One of these channels records the state of the Trigger. The second acts as a Controller channel that can be used to set the Trigger threshold. (included in base package)

- Vaisala HI70 – This Driver reads three channels of data from the Vaisala HI70 humidity sensor. (additional charge)

- Vaisala HMT310 – This Driver reads 11 channels from the high-precision Vaisala HMT310 humidity sensor via the serial port. It can be multiply-instantiated. (additional charge)

- Watlow – This Driver writes a temperature setpoint to, and reads a process temperature from, a Watlow EZZone temperature controller. (additional charge)

- Web Power Switch – This Driver controls a Web Power Switch 7 internet-enabled AC power strip. By configuring a script on the Web Power Switch 7 (see Appendix F: Web Power Switch 7 Watchdog Function), MICAS-X can also

use this device as a Watchdog, such that if MICAS-X or the computer on which is running crashes, one or more outlets of the Web Power Switch 7 will be automatically turned off. This Driver can be multiply-instantiated. (additional charge)

9.6.8 Instruments

Instruments are MICAS-X modules that are intended for interfacing to more complex hardware, instruments, timing, and data types than Drivers allow for. These instruments may deal with data that does not fit the Drivers format (which is a 1-dimensional array of double-precision reals), or may involve data that has a much different time-scale, timing, triggering, or synchronization requirements than the Drivers can accommodate. A Display (see below) is an optional feature of an Instrument.

Currently, no Instruments ship with the base package of MICAS-X, and no Instruments have been created that are supported by MICAS-X-RT. Existing Instruments available for MICAS-X at an additional charge currently include:

- Broadcast – This MICAS-X component provides a way for the operator to configure an arbitrary number of broadcast, or telemetry, data streams that MICAS-X can use to send data to other computer systems. It supports serial ports and UDP Ethernet communication. This instrument can be multiply-instantiated. It also allows the user to configure up to 10 channels to send to Shared Variables that can be read by the Data Dashboard application for Android or iOS devices.

- Command – Enables MICAS-X to listen for valid commands over a serial or UDP port, so that other programs can control and interact with MICAS-X. Includes the "MICAS-X Command Interface" Utility program for remote program control.

- Email – Allows Alerts to be sent to any number of email addresses. Also allows the user to manually send short email messages to the addresses configured. Note that this Instrument has been tested to work with some email servers, but the configuration options for authentication are limited, so not all email servers are guaranteed to work. Also note that all emails sent by this Instrument have the same Subject line (as specified in the configuration file) and are all sent to the same list of recipients. In addition to configuring an Alert to send an email (see the notes in Section 14), if the Instrument is configured to 158xecu in MICAS-X as a Display, one can also send short emails directly from that display by typing text into the Message box and pressing the Send button. Be sure to read the section on configuring the Email Instrument (in Section 7. 5) for caveats that must be kept in mind when using this functionality.

- Ocean Optics – This Instrument allows the acquisition of optical spectra, with averaging and file writing, from an Ocean Optics USB spectrometer.

- Ramp – Can be used to create linear and logarithmic Ramps of the value of one channel over time. This module can create Ramps with a much finer time resolution than could be accomplished with a Sequence. It also provides a simpler interface than a Sequence would. This Instrument can be configured such that the Initial and Final Values of the Ramp, the Ramp Time, and the ability to turn on and off the Ramp can all be controlled by other Channels, or by manual controls on the Ramp Display.

- RT File Transfer – This Instrument automates the transfer and archiving of data files created on an embedded RT target by MICAS-X-RT to the Windows host. After transferring files from the RT target to the host, this instrument can optionally delete the original files on the RT target to ensure that the target does not run out of disk space.

- SMPS Ramp – This Instrument includes the same functionality as the Ramp Instrument, but extends it so as to acquire particle count data while the ramp is scanning. This allows the acquisition of data from a scanning DMA (Differential Mobility Analyzer). When the ramp is complete, the full particle spectrum is calculated and the relevant data is written to a text file. The path and name of the text file is displayed on the Instrument front panel.

- WebCam – This Instrument allows for one to display the image from a USB WebCam or a WebCam built into a laptop.

- Web Interface – This Instrument publishes data via the Skyline API, allowing it to be used on a custom web page written with LabVIEW NXG.

- Web Page – This Instrument displays a web page in an embedded browser. Customization can allow for additional functionality such as ingesting information from the web page into MICAS-X.

9.6.9 Displays

Displays are MICAS-X modules that can be instantiated inside a MICAS-X tab or as a separate window. A display can have nearly any functionality, but the primary purpose is to provide additional user interface elements for data or controls that are part of the MICAS-X system, e.g. that originate in a Driver, or that come from an Instrument. Each

Instrument can have a Display module that presents the Instrument to the operator, but the generic Display is not tied to a specific instrument. Displays currently included in MICAS-X are:

- 3Graphs – This display contains three time-series graphs, each with a data channel graphed on the left axis and another graphed on the right axis. (included in base package)

- Big Display – This display allows for the user to specify a handful of important channels that will be displayed in a large font size. It is intended to provide a view of select items that can be more easily seen from a distance, e.g. from across a room. This display is designed to be used outside of the MICAS-X tabs, as a separate window. (included in base package)

- Document Display – The Document Display is used to present instructions and documentation to the operator. It can be controlled by the MICAS-X program, so that instructions relevant to the current process are automatically presented. See section 7.8.3 for more information. (included in base package)

- Indicators Display – The Indicators Display can show up to 32 channel indicators. Each channel can be configured to be displayed as a gauge, bar, numeric, or an LED. (additional charge).

- Map – This Display uses map tiles downloaded ahead of time to display color-coded Channel data on a zoomable map. This functionality can be indispensable for a mobile lab that does not have internet access. (additional charge)

- Multi Display – Each instance of this Display allows the creation of up to six views that the operator can quickly select from. Each view contains two time-series graphs with up to 8 and 4 channels each, four buttons along the top which can be programmed with any command, and five options along the left side. Each option can be configured as a Command, control channel, indicator channel, or Sequence. (additional charge)

- Sequences Display – Sequences can be used with or without this Display, since Sequences can also be turned on and off via buttons on the Control tab and through other mechanisms. This Display presents a table that displays all the Sequences defined. This table provides a more detailed view of the Sequences than is available elsewhere, and includes the current step and the timer value for any Wait command that is being executed. Sequences can also be started and stopped from this Display. Finally, hovering the mouse over any step in a

Sequence displays the detailed parameters for that step. (included in base package)

- XRT Sequences Display – This Display is similar to the regular Sequences Display, except that it is used on a Windows version of MICAS-X to view and control Sequences that are running on an embedded RT target using MICAS-X-RT. (additional charge)

- XYZ Display – This Display creates a plot that uses two channels for the X and Y points, and a third channel is displayed by color at the coordinates of the first two channels. This Display is especially useful for geographic display of data, as Longitude and Latitude can be used for X and Y and any measured channel can be used as the color-coded Z channel. In addition, an optional Marker (black X) can be placed anywhere on the graph as a reference point. Note that for version 1.4.2 and later, the XYZ Display includes automatic data reduction. This algorithm ensures that a maximum of 2000 points will be displayed on the graph at any time. If a time-span is selected that includes more than 2000 points, the Display decimates the data, throwing out every other point, or every third point, or every n point, using whatever value of n is needed to ensure that only up to 2000 points are graphed. The amount of data recduction being used is displayed in the lower right-hand corner of the Display. This automatic data reduction ensures that the program performance does not suffer, as it could when displaying large amounts of data on this graph. (additional charge)

9.6.10Calculations

Calculations are VI's that are used with the Calculations Driver to provide calculated data channels based on other existing data channels. The Calculation VI's must adhere to the defined API exactly. It is also important to ensure that the resulting calculations can execute within the time allowed by the configuration of the acquisition loop rates. Calculation Vis can include sub-Vis. The user must have the correct version of LabVIEW installed to create new calculations, but the resulting VI's can be used when MICAS-X is run inside LabVIEW or as an .exe. MICAS –X currently includes the following Calculations:

- Demo Calculation serves as a template for the user to create their own calculations. It sums two input channels to result in a single new output channel. This Calculation can be multiply-instantiated.

- Channel Averager is a Calculation VI that takes one channel and creates a running average over the last N samples. N is defined by the value of Global 0. Since Calculation Vis don't have their own configuration editor, this example shows how the Globals can be used as configuration parameters for defining how a Calculation will work. Also note that this Calculation VI contains its own local memory, which it uses to hold the previous values used in the running average. This Calculation can be multiply-instantiated.

- Previous Value is a Calculation VI that returns the value that a specified channel had on the previous acquisition iteration. This memory allows for logic such as detecting when a channel has changed value, as well as for measuring the rate of change of a channel's value. This Calculation can be multiply-instantiated.

9.6.11 Timers

The Timers Driver creates three predefined Timers as well as any number of user-defined Timer Channels. These Timers can be used in Sequences or with Triggers to create conditions for timing various functions. The Run Time Timer is created automatically and contains the total elapsed time in seconds since the program started or restarted. The Acquire Time and Record Time Timers track how long MICAS-X has been Acquiring and Recording data. These timers cannot be set by the user. They are Disabled (set to NaN) when the program is not Acquiring or Recording.

User-defined Timers can be disabled (set to NaN) or paused. The Timer Driver includes the commands Disable, Pause, and Unpause for these purposes. To re-enable a Timer, write a new value to it with the Set command. To disable a Timer write the value "NaN" with the Set command. Timers always count up. Timers can be configured to start disabled by using the value "NaN" for their initial value. They cannot be configured to be paused when the program starts.

9.6.12 Constants

Constants are configurable, nameable channels that have a constant value while MICAS-X is running, but which can be assigned any value in the configuration file. This allows for convenience in creating calculations and equations, where Constants can be referenced just as a channel, but the constants do not appear in the data files or the lists of channels that can be graphed.

9.6.13Globals

Globals are a special kind of Controller, or output, channel available in MICAS-X. Twenty global channels are available. They are configured in the Globals module configuration window. If the name of the global is left blank, the global will not be used within MICAS-X.

Globals act very similarly to Manual channels, with some additional functionality. Like a Manual channel, a global can be shown on the Control tab so that the operator can change its value directly, or its value can be set by a command, sequence, trigger, and other programmatic methods. The global's value can be read and used as a condition in a sequence step or a trigger, and the global's value can be an input to an Equation or Calculation. In addition to all these properties, globals provide several additional functions.

Within LabVIEW, the global channels are implemented using traditional LabVIEW global variables. This means that they can be accessed, (read or written to) by any VI in the MICAS system. Thus custom Instruments, Displays, and Drivers can potentially use globals to communicate with other parts of MICAS-X. (Note that none of the standard MICAS-X Drivers, Displays, or Instruments use the globals in this way.) Another MICAS-X component that can take advantage of this property of Globals are the Calculations. Calculations are user-created LabVIEW VI's that can be directly incorporated in MICAS-X through the Calculations Driver. However, the Calculations Driver has limited configuration options, with no way to configure parameters specific to any particular Calculation. Globals provide a way to create configurable parameters for a Calculation. See the Calculations entry for more information, including how the "Channel Averager" Calculation uses a global to define the number of points to average.

The second function that Globals provide, beyond what Manual channels can do, relates to how Commands are used in MICAS-X. Globals allow for a special set of Commands that can operate on two channels by referencing one channel by name and the global by index. See the Commands section for more information.

A third special function of the Globals is to latch error codes so that they can be reliably detected. The Error Program State Variable automatically contains the error code of the most recent error to occur, and it is reset to 0 after each acquisition loop. If multiple errors occur during one acquisition loop, all but the last one are lost to the Error channel. The error channel therefore provides a snapshot of whether errors are occurring, but cannot guarantee that any specific error is identified. To allow specific error codes to be used in Trigger or Sequence conditions, the Latch Error function of the Globals can be used. If a specific error code is entered in a Globals Latch Error field, that Global will be set to that error code whenever the error occurs. The Global will never be

automatically cleared, so any action (Sequence) that acts on the error will need to provide its own clearing mechanism if that is needed.

9.6.14 Program State Variables

Several channels of data are always present in the MICAS-X system, regardless of which Drivers are included in the configuration. These Program State Variables are quantities which help indicate how the MICAS-X system is running. The Program State Variables are handled by Acquisition Loop 0, the default Acquisition Loop. The current Program State Variables are shown in the table below.

Variable	Definition and Purpose
--	This is the null channel. It is used to indicate that a specific channel has not be selected, or that the selected channel name cannot be found in the current list of channels. This channel always has a value of Not-a-Number (NaN).
Time (Sec)	This channel has the value of the time in seconds since midnight, Jan 1, 1904. This date and time value has a precision of tenths-of-a-second.
Sec Since Midnight	This channel has a value of the number of seconds since midnight of the current day.
Error	This channel has the value of the most recent error code. It is cleared on every loop of the Acquisition module. Thus if this channel has a constant value for more than one loop time, it is probable that the error is recurring at a regular interval.

In addition to the above listed Program State Variables, previous versions of MICAS-X had additional Program State Variables, which have now been moved to the System Driver. In this way, they are now optional, and if you do not wish them to be included in your data, you need not include them in the System Driver configuration. Some of the previous Program State Variables that are now System Variables are:

Variable	Definition and Purpose
Acquire	This channel has a value of 1 when the program is acquiring Driver data, 0 otherwise.
Record	This channel has a value of 1 when the program is recording data to the user-defined files, 0 otherwise. Previous to version 2.2, this variable directly reflected the result of the Record command. As of version 2.2, there are commands that can turn on and off writing for

	each file. This variable will have a value of 1 whenever at least on file is writing data.
Debug	This channel has a value of 1 whenever the program is in Debug mode, 0 otherwise.
Last Acq Time	This channel has the time in seconds since midnight, Jan 1, 1904, of when the Acquisition Loop 0 last acquired data.
# of Alarms	This channel indicates how many Alarms are currently True.
Current Tab	This channel identifies the tab being displayed in the main MICAS-X window. 0 = Control tab, 1-14 = Display tabs, 15 = Utility tab, 16 = Summary/MICAS tab, 17 = Debug tab.

9.6.15 Commands

Much of the MICAS-X functionality is designed around Commands. Sequences, Triggers, Buttons, and other mechanisms in MICAS-X all cause actions by sending Commands to the MICAS-X Command Loop. The list of standard Commands is shown in Appendix B. In addition, Drivers can implement custom commands. If a Driver is written correctly, any Commands that the Driver supports will be available throughout the MICAS-X system whenever that Driver is included in the current configuration.

Commands in MICAS-X have the following structure:

Command:String Argument:Numeric Argument

The functionality of the String Argument and the Numeric Argument vary depending on the Command being used, and some Commands may not require one or either of these arguments. When a Command is being configured in the MICAS-X Configuration Editor, the program knows which arguments are needed for each Command, and presents the user with the relevant parameters.

For example, the Set Command is used to set a Controller (output) channel to a new value. It therefore requires a valid Controller channel name as a string argument, and the new value of the channel as the numeric argument. Thus in the Configuration Editor, when the Set Command is chosen for a Sequence step or a Trigger, the string argument is automatically populated with a list of all available Controller channels, and the numeric argument is enabled. In contrast, the Restart Command requires no parameters, so when this Command is chosen in an editor, both the string and numeric arguments are disabled.

The fact that Commands can only take one string and one numeric argument is a significant limitation of the MICAS-X Command structure. In particular, this makes it difficult to structure a Command that will operate on the values of two channels, such as

adding them together. As a partial resolution to this limitation, a special set of Commands has been created that act on the Global Channels. All the Commands listed in Appendix B that contain the letters “Gbl” are part of this set of special commands. These commands act on a Channel and Global. The Channel is specified by the string parameter, and the Global is specified by its number.

A more recent addition to MICAS-X to address this command limitation are a new set of five two-channel commands. For these commands, the two channels required are put into the same string, with a comma between them. The channels are referred to as the Source Channel and the Target Channel. The Target Channel is where the result of the operation goes. So for the command AddChToCh, the sum of the values of the Source Channel and the Target Channel ends up in the Target Channel.

In the configuration editor, the Triggers and Buttons have direct entries for the Source and Target Channels. The Sequence Editor was not designed with room enough for an additional parameter, so a new Sequence Command (SourceCh) has been added. To use one of the two-channel commands, first use the SourceCh command to define a source channel, then use the two-channel command to define the target channel. (Note that the Source Channel for each Sequence in the current configuration is remembered, so if multiple two-channel commands are to be used that all have the same Source Channel, the Source Channel only needs to be set once.

E.g. Global 0, though it may have been given a text name in the configuration, such as “Temp Setpoint”, is referenced by using a value of “0” for the numeric parameter. As a further example, assume that there is a Manual Channel named “Temp Increment” in which the operator can enter a increment value in degrees C, and that Global 0, which has been named “Temp Setpoint” is used to control a temperature. The Command “AddToGlb” can be used with the string argument “Temp Increment” and the numeric argument “0” (the Global index) to add these two channels together, with the result being placed in the Global Channel.

The set of special global Commands includes commands to copy date from a Channel to a Global or from a Global to a Channel. It also includes commands to add, subtract, multiply, and divide two channels, where one is a standard channel and one is a global. Commands are available that place the result in either the normal channel or the global, as desired.

Although this mechanism is admittedly round-about, it can be used to operate on any two normal channels through the use of Globals. For example, if one needed to add two normal channels and put the result into the second of the two channels, one could create a Global for temporary use, copy the first channel to the

Global, then use the “AddGlbToCh” Command to add the value in the Global channel into the second normal channel.

9.6.16 Constants

Constants are much like Channels, but with several limitations. Constants are defined in the Configuration Editor and are given a value when configured. Their value cannot change while the program is running. Constants are not written to data files and do not show up in lists of channels to graph or select for other purposes. Thus constants can serve as experiment-specific parameters, but they help keep the data and user interface cleaner by not adding to the total number of Channels that the operator must deal with. The archived configuration file serves as the record of the value that the constants had when the data was taken.

Constants can be useful for experiment-dependent values such as the dimensions of a piece of apparatus that is used in a calculation, a physical constant used in an equation, or an experimental identification number. An alternative to using constants for these functions is to use a Manual Channel. However, Manual Channels will be written to the main .tdms data file, possibly written to other data files, and will appear in lists of channels used for graphing and display. Having channels with unchanging values appear in these places clutters the data unnecessarily.

9.6.17 OSDS

The OCC Streaming Data System (OSDS) is a LabVIEW software package produced by OCC and available to the LabVIEW community as open source software. A description of this package can be found on the OCC website (www.originalcode.com/OSDS.html). The purpose of OSDS is to abstract the reading and parsing of streaming data, so that streaming data sources (those which send data at regular intervals, without being queried) can easily be added to a LabVIEW program by simply creating a proper OSDS definition file. OSDS has been integrated into several data acquisition programs, including MICAS-X. OSDS supports reading data from serial ports and from Ethernet via UPD, UDP Multicast, TCP, and NTP. It also has several ways of reading the computer clock to yield timestamp data.

MICAS-X uses OSDS in two ways. First, there is an OSDS Driver in MICAS-X (see Section 7.5.9), which allows any OSDS format file, either new or pre-existing from another application, to be used to read data from streaming data sources, such as aircraft navigation data, GPS’s, laboratory scales, and other instruments. This MICAS-X OSDS Driver puts a MICAS-X compatible wrapper around all the existing OSDS functionality, and uses the existing OSDS Readers and Parsers. The OSDS Format Editor is included with the MICAS-X program so that OSDS Format files can be created and edited as necessary.

The second use of OSDS in MICAS-X, prior to version 3.3.2, is that several MICAS-X Drivers which operate with streaming data sources have commands with start and stop the recording of the raw data stream into OSDS Stream files. OSDS Stream files are an extension of OSDS Format files. They include the OSDS Format information, indicating how the data was read, as well as time-stamped records of the raw data that the program saw on the specified port. This can be useful in debugging communication with an instrument. By capturing the raw data stream in an OSDS Format file, one can then use the OSDS Simulator to play the data back again in the future and test new OSDS Formats against it until the data is properly parsed. Drivers which currently allow the recording of OSDS Stream files include the Airmar, MOSDS, Omega, and Prime Scales. Note that in version 3.3.2 and later of MICAS-X, the OSDS stream file writing was replaced by ICM stream file writing.

9.6.18 ICM

ICM (Instrument Communication Module) is a successor to OSDS (see Section 9.6.17.) Like OSDS, ICM is highly configurable, allowing one to add new data sources to MICAS-X (or other programs) by creating an appropriate configuration file rather than writing an entire Driver. ICM is more advanced than OSDS, however, as it can parse many more types of instrument data, can scale data through several mechanisms, and can write to output channels as well as read input channels (whereas OSDS could only read input channels). As of version 3.2.0, the ICM Driver (see Section 7.5.8) is included in the MICAS-X base package. As of version 3.3.2, ICM is used for writing streaming files in several of the MICAS-X Drivers. ICM is open source and can be found on GitHub at https://github.com/NI-ALARM/ICM.

9.7 Using Channel Values in Text

Text used in MICAS-X can be formatted in such a way that a Channel name is replaced with the Channel's value before the text is displayed or stored. This is done by putting a "#" character immediately before and after the Channel name. Thus, if a Sequence step is configured to use the Log Command to send the text "Chamber temperature is #Chamber Temp (C)# degrees C.", and if the Channel "Chamber Temp (C)" has a value of 25.3, then when the actual text is logged, it will appear as "Chamber temperature is 25.3 degrees C."

Previous to Version 3.2, only the Email Instrument and the Alert Command supported replacing Channel names with their values in this way. As of Version 3.2, however, this feature can be used in numerous places, including:

- Log a Message menu item in MICAS-X File menu
- Message text field on Control tab of MICAS-X

- Log This button MICAS-X tab of MICAS-X
- Log This button on 3Graphs Display
- Log Command
- Alert Command
- Ask Command
- AskOK Command
- Ask(Num) Command
- OpenNotifier Command
- Send Command

10 MICAS-X-RT Operation

This chapter discusses the operation of MICAS-X-RT on a Real-Time platform. See Chapter 9 for information on the operation of MICAS-X on Windows.

MICAS-X-RT can be configured to run automatically when the RT system boots, or in interactive mode from within the LabVIEW environment. The program has a limited user interface, which only is accessible when the program is run interactively. This UI will be discussed in Section 10. 1. Normal operation, however, is to have MICAS-X-RT installed on the Real-Time system as a startup program so that it runs automatically at boot-up. In this mode, it is possible to have a configuration file on the RT platform that defines how the program should configure itself and begin acquiring data. Without this "Auto-Load.ini" file, the program will run but will await a configuration from a Windows machine before starting any acquisition or control.

10.1 MICAS-X-RT User Interface in Interactive Mode

MICAS-X-RT is designed to be run headless and automatically on a Real-Time system. However, there may be times when it is useful to run it interactively. If it is launched from within the MICAS-X project in LabVIEW, one can see and interact with the program's front panel. This section describes the controls and indicators available under these circumstances.

The Config File indicator displays the name of the configuration file that MICAS-X-RT is currently using. See Section 6. 3 for information on MICAS-X-RT configuration files.

Under the Config File indicator are four status LED indicators. Command Link Connected and Message Link Connected will be lit whenever the relevant Network Streams connections are active between MICAS-X-RT and the MICAS-XRT Driver running in MICAS-X on Windows. These connections are discussed further in Section 7. 6. 16.

The Stop button in the upper right corner will stop the program. Underneath the Stop button is a list of all the channels in the current configuration and their current values.

Just to the left of the channel list is a scrollable array of all the Triggers defined in the current configuration. Each Trigger displays its Name, its State (both in text and by color) and the Time at which it most recently became active.

At the bottom of the screen is an array of messages. The messages displayed here are the most recent error and event messages generated by the program, but only those which have not been successfully sent up to MICAS-X on Windows. When MICAS-X is connected (via the MICAS-XRT Driver), any messages generated on the RT system are automatically sent to the MICAS-X log file. Note that all log messages generated on the RT system are also logged locally on the RT file system.

Above the Messages display is a section of controls which allow the operator to specify and execute any Command that the MICAS-X-RT system supports. Above the Command controls are a pair of controls that allow any Controller channel's value to be set. These Command and Controller controls operate in a similar manner to those on the MICAS-X program, described in Section 9. 3. 4.

10.2 Configuring MICAS-X-RT

MICAS-X-RT uses the same type of configuration file as MICAS-X. This configuration file should be created using the MICAS-X Configuration Editor on a Windows machine. When creating this configuration file, set the "Target" parameter to "Real-Time", as discussed in Section 7. 1. Section 7. 4. 1. 2 has additional information on editing a MICAS-X-RT configuration file.

In order for MICAS-X-RT to use the configuration file, it must either be sent to MICAS-X-RT by the MICAS-XRT Driver, or it must be renamed as "Auto-Load.ini" and transferred to the C:\OCC\MICAS-X Support" folder on the RT target. See Section 10. 3 for information on how to use the MICAS-XRT Driver to connect to MICAS-X-RT. Information on transferring files directly to the Real-Time target can be found in National Instruments' LabVIEW and hardware help, or can be requested from Original Code Consulting.

10.3 Connecting to MICAS-X-RT with MICAS-X

Although other mechanisms are available and/or can be developed, the primary method of connecting to MICAS-X-RT is through the MICAS-X program on Windows, using the MICAS-XRT Driver. Configure the MICAS-XRT Driver as described in Section 7. 6. 16. In particular, enter in the MICAS-XRT Driver configuration the name of the configuration file that you wish MICAS-X-RT to use. When MICAS-X is run on Windows with the MICAS-XRT Driver included, it attempts to connect to MICAS-X-RT whenever

Acquire is True. When a connection is made, the MICAS-XRT Driver sends the Real-Time configuration file to MICAS-X-RT on the Real-Time target. If the configuration file being sent down is the same (e.g. has the same CRC value) as the configuration file already in use on the Real-Time target, then MICAS-X-RT continues to run without interruption. If the configuration file does not match exactly, MICAS-X-RT will stop its current operation and load the new configuration. Thus, if uninterrupted operation of MICAS-X-RT is important, it is imperative that the current MICAS-X-RT configuration file be available on the host Windows computer and configured properly in the MICAS-XRT Driver.

MICAS-X-RT and the MICAS-XRT Driver are designed to allow repeated connection and disconnection. As long as the MICAS-XRT Driver does not send a new configuration file down to MICAS-X-RT, the connection can be severed and reacquired without interrupting any functionality on the Real-Time target. There are several caveats, however, with regards to how data is transferred to the Windows host MICAS-X program. Note that all error and event logs generated on the Real-Time target are saved to the local Real-Time log file. In addition, whenever MICAS-X on Windows connects to MICAS-X-RT, any pending log messages that have not been sent up to the host computer will then be sent. There is, however, a maximum buffer of 100 log messages. If more than 100 log messages are generated while the two systems are disconnected, only the most recent 100 messages will be sent to the host computer when the connection is re-established.

Also note that the channel values from MICAS-X-RT that are available on the host system are only transferred when the connection is present. When a connection is broken and then re-established, the history of missing values is not sent to the host computer. Thus time-series graphs and other history displays of MICAS-X-RT channels on the host system will only have data available from when the connection was maintained.

Because of the above limitation on history data, it is strongly suggested that the MICAS-X-RT system be configured to store data locally. Also note that MICAS-X-RT does not utilize the .tdms master data file that MICAS-X on Windows does. That is, if no File Writer files have been defined, MICAS-X-RT will not save any data. Use the File Writer configuration (Section 7.4.4) to create local log files on the Real-Time system. In addition, be sure to set Record ON. If Record is not turned on, data will not be saved. Record can be turned on when MICAS-X-RT starts via the “Record on Start-up” option (see Section 7.4.1.2), or by using the Record Command with a non-zero value. (The lack of a .tdms file and the default of not recording data automatically are due to the limited disk space available on many Real-Time platforms.)

Note that even if you do not record data on the Real-Time platform via the File Writer module, the Log File will continuously be updated. Thus it is important to manage the disk space on the Real-Time platform by reviewing, archiving, and deleting files as necessary.

10.4 Debugging MICAS-X-RT Issues

Due to the nature of the Real-Time platform, debugging code running within MICAS-X-RT can be significantly more challenging then debugging code running in MICAS-X on Windows. When errors occur in MICAS-X-RT, always review the log file carefully. (The same is true for MICAS-X.) Often, a situation will present itself as a repeating error that happens regularly the entire time the program is running. However, the root cause of this error can usually be traced back to the beginning of the program run. Look at the portion of the log file when the program was first started. Usually there will be a different error that occurred when the offending piece of code was first launch or first run. That error will usually give better clues to the problem than the later, recurring error.

One of the best approaches to resolving problems with MICAS-X-RT is to rebuild and redeploy all the RT code. This is most often done by OCC, but if you have a license for the LabVIEW environment and understand the process, you can execute this process as well. Rebuild each component of software that the MICAS-X-RT system is using. This process will guarantee that no components have compile-time errors present. Redeploying all the components will then assure that all the components are consistent and compatible with each other. It can also be useful to run MICAS-X-RT in interactive mode after deploying all the components, since the limited user interface in this mode may provide some additional clues to problems that are occurring.

If multiple errors are happening, it can be very useful to create new configuration files, each with just one Driver or Instrument included, so that issues can be pinned down to the specific component that has a bug. Sometimes, a single Driver will fail to load because one of its dependencies is not present in the LabVIEW software loaded on the RT target. In this case, MICAS-X will report a 1026 error code, indicating that a VI was broken and could not be run. This type of error might be resolved by rebuilding and redeploying, as above, or it may be necessary to install additional LabVIEW software on the RT target.

One common symptom is that one Driver may be broken, as described above, resulting in a 1026 error. When this happens, that Driver's channels are not known to MICAS-X-RT. Because the channel list is incomplete on the RT target, it will not match the channel list that the MICAS-XRT Driver on Windows expects. In this case, the MICAS-XRT "Connected" channel will not have bit 2 (value 4) set, and will therefore most often have a value of 3 (bits 0 and 1 may still be set, indicating command and message links have been made). With bit 2 not set, MICAS-XRT will not see any of the data from the RT target. This issue will only be resolved when both the RT and Windows targets have the same channel list, which means that MICAS-X-RT must be debugged until all its Drivers are working correctly.

11 Utilities

Utilities are helper programs that work with MICAS-X. By definition, they include functionality that is not needed continuously while MICAS-X is running, but may be needed occasionally. Utilities can be used for reviewing data, editing the configuration files, or performing calibrations. Utilities can be run inside MICAS-X, by selecting them from the Utilities menu (in which case they appear in the Utility tab), or they can be run inside LabVIEW, or they can be run as compiled executable programs separate from MICAS-X. Several Utilities are included with the base copy of MICAS-X, as described below.

Utilities are supported only in MICAS-X for Windows, and not in MICAS-X-RT.

11.1 Command Interface

The Command Interface is not included in the MICAS-X base package, but is included whenever the Command Instrument is purchased. The Command Instrument is a MICAS-X module that allows MICAS-X to receive commands from another system, either over UDP via Ethernet or over a serial line. The Command Interface is an example of how to send these types of commands to MICAS-X. It can be used to send commands to the same instance of MICAS-X that it is running in, as a test system, or it can send commands to another instance of MICAS-X.

11.2 Configuration Editor

The Configuration Editor is a key element of the MICAS-X system. It is described in its own section of this manual. The Configuration Editor is used to create and maintain the configuration file(s) which define how MICAS-X works and what functionality it includes.

11.3 Data Reader

The Data Reader Utility reads and presents the data saved in the .csv files that MICAS-X creates. With the Data Reader, you can graph any of the data channels vs time or vs any other channel. The format is simple enough that a .mdoc can be created with any text editor. However, the mdoc Editor Utility in MICAS-X can also be used as a quick and easy way to create .mdoc files.

11.4 FTP Utility

The FTP utility implements a simple FTP file transfer interface between the computer running MICAS-X and another computer. It can be useful for manually

transferring data files from an embedded MICAS-X program to the windows host, or for other file transfers.

First, on the Connect tab, provide the proper information for the “FTP Server”, “FTP Server Directory”, “User Name”, and “Password” parameters. Then press the “Press to Connect” button. Then move to the “Transfer” tab. Here you can select the directory you want to transfer files from. To transfer a file select it and click the move button nearest the file display you are transferring from. The file can also be deleted by selecting the corresponding delete buttons. Before deleting a directory, make sure to check that it is empty. The lower button of each pair corresponds to the right file display.

11.5 mdoc Editor

The .mdoc file format is a simple text file which includes section names that are created in such a way that MICAS-X can parse the sections for display with the Document Display. The formatting required for the Document Display is simply that each Section must be preceeded with a line of text that starts with two asterisks, contains the section name, and ends with a line feed. Thus a section format line might look like this:

**Introduction

The mdoc Editor Utility provides a quick and convenient way to create and edit mdoc files. To create a new mdoc file, run the program and cancel any file dialog asking for a file to load. Click on the uppermost Insert button to create the first section. Then click on that section in the Sections list. Once the section is highlighted, you can edit its name in the Section Name parameter below the Sections list. Enter the text for that section in the Text box to the right. You can continue to Insert and Delete sections, and edit any section’s name or text by first clicking on it in the Sections list. Use the Load A Document button to read in an existing mdoc file. Use the Save button to save the current document. If the document is new, the Save button will be greyed out. Use the Save As button to save the document to a new file.

11.6 .mdoc Reader

The .mdoc file format is a simple text file which includes section names that are created in such a way that MICAS-X can parse the sections for display with the Document Display.

The .mdoc Reader Utility provides a quick and convenient way to view .mdoc files. When you open the driver you will be asked to navigate to the desired .mdoc file. The Section menu allows the user to select a section from the current document. The

Document menu allows the user to select between loaded documents. The “Load a Document” button will load a new document to the documents menu.

11.7 Log Reader

The Log Reader Utility presents a log file for review. If it is launched within MICAS-X, the current log file is automatically loaded, and will be updated in the Log File Reader whenever it changes. Any other log file can be loaded whenever desired, to review previous logs.

11.8 TMDS File Reader

The TDMS File Reader allows you to review the data stored in the .tdms file that MICAS-X stores behind the scenes. It has numerous ways to view the data, including time-series charts and tables.

12 MICAS-X Security Device

The MICAS-X license is protected through a USB hardware security device. The device is a blue USB dongle about the size of a USB thumb drive and will have labels with “MICAS-X” on one side and “SN xx” on the other. The serial number for the MICAS-X license is embedded in memory on this device. Note that there is no user-accessible memory on this device.

If MICAS-X is run with no security device present, it will operate normally for 10 minutes in “demo” mode. After the 10 minutes are up, Recording and Acquiring will be turned off, and no new commands will be executed.

There is an indicator on the MICAS-X tab that displays the status of the security device. In addition, if no security device is present, an error message will be logged to the log file, both when the program is first run and it enters the demo mode, and 10 minutes later when most functionality is inhibited.

In certain licensing situations, an expiration date may be applied to the security device. If the expiration date has passed, MICAS-X will work for 10 minutes in demo mode before inhibiting most functionality.

The About menu item in the Help menu can be used to view the status of the security device. The About window displays the current security device status, the MICAS-X serial number, and the expiration date, if one is set.

13 Appendix A: Reserved Keywords

There are some keywords that are used internally within the MICAS-X system, and therefore should not be used as names for Channels, Sequences, Triggers, etc. Duplication of names should be avoided, including the use of names that contain other names within them. Note that the Channels window in the Configuration Editor under the Modules selection will help identify duplicate names and partially conflicting channel names.

In addition to the Keywords listed below, the MICAS-X Commands are documented in a separate section.

Reserved Keywords include:

Reserve Keyword	**Effect**
#DisplayTabs	This is a Constant Channel, the value of which is set equal to the number of Display tabs in use in the current configuration. This number does not include the Control tab (tab 0), the Utility tab (tab 15, which is often not shown), the MICAS tab (tab 16), or the Debug tab (tab 17). It starts at 1 and only includes tabs used for Displays and Instrument displays.
Acquire	This is a channel that is always created by MICAS-X (a Program State Variable) and which has a value of 1 if MICAS-X is acquiring data and a value of 0 if it is not.
Acquire Time	This is a Timer that is always included in MICAS-X whenever the Timers Driver is included. It contains the value of the time in seconds since MICAS-X started acquiring data. It has a value of -Inf if MICAS-X is not acquiring data.
All	This keyword is the name of a List of Channels that is always present. It contains all the Channels defined in MICAS-X.
(avg)	When the Averager Driver is configured to create new Channels, (e.g. not to overwrite the existing Channels), the new Channel created for each average is the name of the source Channel with the text " (avg)" appended to it.
Debug	This is a channel that is always created by MICAS-X (a Program State Variable) and which has a value of 1 if MICAS-X is in Debug Mode and a value of 0 if it is not.
DisableTimer	This is a command used by the Timers Driver.
Error	This is a channel that is always created by MICAS-X (a Program State Variable) and which has the value of the most recent error code reported by MICAS-X. Most of the error codes used in MICAS-X are defined within LabVIEW, though some custom

	error just for MICAS-X are defined and must be documented somewhere...
Median	This keyword is used in the Equations Driver to calculate the median value of a list of channels.
Min	This keyword is used in the Equations Driver to find the largest value in a list of channels.
Max	This keyword is used in the Equations Driver to find the smallest value in a list of channels.
ML	“ML” is a special prefix used by the M-Link Driver to uniquely identify Channels, Commands, and Sequences on a separate MICAS-X system. It should never be used as a prefix for any other MICAS-X module.
Mode	The word “Mode” is used as a prefix to names in the Controller Driver to create channels that are used to control the mode of PID and Simple Controller channels. The Mode channels switch between Manual and Auto (PID) modes.
Rate	This keyword is used in the Equations Driver to calculate the rate of change of a channel.
Record	This is a channel that is always created by MICAS-X (a Program State Variable) and which has a value of 1 if MICAS-X is recording data to a file and a value of 0 if it is not.
Record Time	This is a Timer that is always included in MICAS-X whenever the Timers Driver is included. It contains the value of the time in seconds since MICAS -Xstarted recording data. It has a value of -Inf if MICAS-X is not recording data.
Rollover	When Rollover Channels are included in the Timers Driver, each existing Timer Channel has a new Channel created that has the same name, with the text “ Rollover” appended.
Run Time	This is a Timer that is always included in MICAS-X whenever the Timers Driver is included. It contains the value of the time in seconds since MICAS-X started running.
Sec Since Midnight	This is a channel that is always created by MICAS-X (a Program State Variable) and which contains the time in seconds since midnight of the day that MICAS-X was started. (Note that if MICAS-X is configured to restart data files automatically at midnight, this channel’s value will also restart at 0 at midnight.)
SeqState	When the Sequences Driver is included in the MICAS-X configuration, “SeqState “ is prepended to each Sequence Name to create a channel for each Sequence State (0 = Off, 1 = Running), so that they can be turned on and off using the “Set command.

SeqStep	When the Sequences Driver is included in the MICAS-X configuration, “SeqStep “ is prepended to each Sequence Name to create a channel to record the step number each Sequence is currently in. These channels are set to a value of -1 for each Sequence that is not running.
Time (sec)	This is a channel that is always created by MICAS-X (a Program State Variable) and which contains the time and date in seconds in UTC Time since midnight, Jan 1, 1904.
TrigState	When the Triggers Driver is included in the MICAS-X configuration, “TrigState “ is prepended to each Trigger Name to create a channel to record each Trigger State (0 = False, 1 = Warning, 2 = True, 3 = Alarm).
TrigThr	When the Triggers Driver is included in the MICAS-X configuration, “TrigThr “ is prepended to each Trigger Name to create a channel for each Trigger Threshold, so that they can be set using the “Set” command.
Alarm, Buttons, Channel, Channels, Cluster, Command, Condition, Control, Controllers, Device, Disp, Display, Drivers, Enumerated, Exit, False, File, Files, Globals, History, Hysteresis, Include, Initial, Instr, Instrument, Label, Left, List, Log, Max, Min, Names, Notes, Option, Outside, Prefix, Prefixes, Right, Sequences, Set, State, Steps, Switches, Tab, Tabs, Text, Thresh., Time, Timed, Top, Trigger, Triggers, True, Update, Value, Values	These terms are used for various parameters in the configuration file. If any of these terms are used as channel names or prefixes, then the Rename Tools will corrupt these channel names. E.g. in most cases, these words can be used as channel names with no issues, but be aware that the Rename functions can cause problems when they are used. If these terms are used as part of a longer channel name, then the channel name can safely be renamed using the Rename Tool. Also note that this list is inherently incomplete, since every Driver, Instrument, and Display may have additional terms that are used in its configuration information.
XRT	“XRT” is a special prefix used by the MICAS-XRT Driver to uniquely identify Channels, Commands, and Sequences on a MICAS-X-RT RIO system. It should never be used as a prefix for any other MICAS-X module.

14 Appendix B: Commands

MICAS-X includes a set of Commands that can be used in Triggers, Sequences, Buttons, and Commands. In addition, Drivers can implement custom Commands. The custom Commands for Drivers are not documented here. They are listed in the documentation for each Driver and in the list of Driver Commands below.

The standard MICAS-X Commands are:

Command	Parameters	Description
Log	string to log	Sends text to the log file.
Alert[1]	string for alert text, numeric code	Creates various kinds of user alerts.
Acquire	numeric (0 off, 1 on)	Starts or stops the acquisition of Driver data.
Record	numeric (0 off, 1 on)	Starts or stops the recording of files.
RecordOnce	none	Starts recording (in Acquisition Loop 0) for just one point. Recording stops after Acquisition Loop 0 has acquired data once.
FileRecord	string, numeric (0 off, 1 on)	Turns recording on or off for a single file. The file string can be the Prefix of the file or the index number of the file.
File1ofN	string, numeric	Sets the file writing rate. The file string can be the Prefix of the file or the index number of the file. The numeric is the number “N” defining how often to write data to the file. Any value less than 1 is coerced to 1.
Set	channel, value to set	Sets the value of a controller channel.
Add	channel, value to add	Adds a constant value to a controller channel.
Subtract	channel, value to subtract	Subtracts a constant value from a controller channel.
Multiply	channel, value to multiply by	Multiplies a controller channel by a constant value.
Divide	channel, value to divide by	Divides a controller channel by a constant value.
Incr	channel	Increments a controller channel by 1.
Decr	channel	Decrements a controller channel by 1.
AddChToCh[2]	channel,channel	Adds a channel value to a controller

		channel.
SubChFromCh[2]	channel,channel	Subtracts a channel value from a controller channel.
MultChByCh[2]	channel,channel	Multiplies a controller channel by a channel value.
DivChByCh[2]	channel,channel	Divides a controller channel by a channel value.
CopyChToCh[2]	channel,channel	Copies a channel value into a controller channel.
AddToGlb[3]	channel, numeric address of global	Adds a channel value to a global channel.
AddGlbToCh[3]	channel, numeric address of global	Adds a global channel value to a controller channel.
CopyToGlb[3]	channel, numeric address of global	Copies a channel value into a global channel.
CopyFromGlb[3]	channel, numeric address of global	Copies a global channel value into a controller channel value.
DivGlb[3]	channel, numeric address of global	Divides a global channel by a channel value.
DivChByGlb[3]	channel, numeric address of global	Divides a controller channel by a global channel value.
MultGlb[3]	channel, numeric address of global	Multiplies a global channel by a channel value.
MultChByGlb[3]	channel, numeric address of global	Multiplies a controller channel by a global channel value.
SubFromGlb[3]	channel, numeric address of global	Subtracts a channel value from a global channel.
SubGlbFromCh[3]	channel, numeric address of global	Subtracts a global channel value from a controller channel.
Abs	channel	Makes a controller channel value equal to its absolute value.
Negate[4]	channel	Makes a controller channel value equal to the negative of its value.
Recip	channel	Makes a controller channel value equal to the reciprocal (1/x) of its value.
Not[4]	channel	Makes a controller channel value equal to its logical negation. (0 becomes 1, any non-zero value becomes 0.)
SetTrigger	channel (trigger), value	Sets a Trigger Threshold channel to a value. Channel can be the trigger name or the name of the Trigger Threshold

		Channel for that Trigger.
TriggerDisable	trigger	Disables a trigger. If it was True or Alarm before, it will execute the True-to-False action. While Disabled, it will ignore its Condition.
TriggerEnable	trigger	Enables a trigger. If its Condition is then met, it will execute its False-to-True action.
StartSeq	sequence, value	Starts a sequence. Value specifies the step number to start at. This defaults to 0.
StopSeq	sequence	Stops a sequence.
StopAllSeq		Stops all running sequences. Any exit steps of the sequences that are stopped will still execute. If this command is in a Sequence, it will stop its own sequence.
StopSeqSet	sequence set	Stops all sequences that are running and that are in the Sequence Set specified.
PauseSeq	sequence	Pause a sequence that is running.
UnpauseSeq	sequence	Resume a sequence that is paused.
SingleStepSeq	sequence	Causes a paused Sequence to execute a single step. Returns an error if the Sequence specified is not paused.
EnableDriver	driver name with prefix	Enables a Driver.
DisableDriver	driver name with prefix	Disables a Driver.
JPG	none	Stores a .jpg image of the current screen.
PNG	None	Stores a .png image of the current screen.
Tab	numeric of tab to go to or text of tab name	Displays the requested tab or window. This command uses the text string unless the text string is blank, in which case it uses the numeric.
Tab(Channel)	channel	Displays the tab specified by the value of the channel.
Launch	path and name of a program or VI	Starts a LabVIEW VI or Windows executable program.
Script	script command (string)	Embeds an entire Script Command in a Sequence Step.
HelpText	text (string), index n	Specifies the text for entry “n” for the

	(numeric)	Help Box, an optional text box on the Control tab.
ViewHelp	index n (numeric)	Specifies the index of the Help text that will be displayed in the Help Box.
Ask – only available in Sequences	text, source channel	Presents a dialog to the user asking for a Yes/No response, which is assigned to the channel.
Ask(OK) – only available in Sequences	timeout (numeric)	Presents a dialog to the user which waits for user acknowledgement. No information from the user is returned. If timeout > 0, the dialog will close after that many seconds even if no user input has been received.
Ask(Num) – only available in Sequences	text, source channel	Presents a dialog to the user asking for a value, which is assigned to the channel.
OpenNotifier	text(string), timeout (numeric)	Presents the Notifier window which displays the text string. If timeout is <> 0, the window will close automatically after that many seconds. If the timeout is <0, the absolute value of the time-out will be used, and a “Continue” button will be shown which can be pressed to close the window before the time-out is over.
CloseNotifier	none	Closes the Notifier window, if it is open.
CloseAllQG	None	Closes all Quick Graph windows.
Utility	Utility name (string)	Opens the requested Utility. The Utility name string can include the “.vi” extension or not. If the string is blank, the Command will cause any open Utility to close.
Send	string	Sends an arbitrary string out a serial or UDP port via the Broadcast Instrument.
Comment	documentation (string)	Used for documenting the sequence only, no operation.
CompReboot	none	Stops MICAS-X and reboots the computer.
CompShutdown	none	Stops MICAS-X and shuts down the computer.
Exit	none	Stops MICAS-X.

Restart	none	Stops MICAS-X and immediately restarts it.
Wait(Value) – only available in Sequences	numeric value to wait in seconds	Waits a constant number of seconds before executing the next Sequence step.
Wait(Channel) – only available in Sequences	channel	Waits the number of seconds specified by the channel value before 183executing the next Sequence step.
Wait(NewValue) – only available in Sequences	channel, value	Checks every “value” seconds to see if “channel” has changed its value. (The minimum “value” check time is 0.1 seconds.)
WaitUntilValue – only available in Sequences	channel, value	Waits until “channel” has the value specified by the “value” parameter
Wait(Sequence) – only available in Sequences	sequence, value	Checks every “value” seconds to see if “sequence” is done or not. Continues waiting until “sequence” is done. This allows Sequences to be used as sub-routines. (The minimum “value” check time is 0.1 seconds.)
Goto – only available in Sequences	label	Directs the Sequence to move execution to the specified step.
SourceCh – only available in Sequences	channel	Specifies the source channel for the following sequence step for commands that require both a source and a target channel[3]
CondSource – only available in Sequences	channel	Specifies a source channel for use with a Condition in a Sequence Step.
ExitStep – only available in Sequences	none	Executes the Exit Step for the Sequence (if one is defined)
Get – only available in Command Interface Utility	channel	Commands MICAS-X to reply to the Command Interface with a report of the value of the channel specified.
LastLog – only available in Command Interface Utility	none	Commands MICAS-X to reply too the Command Interface with a report of the last message set t the Log file.

Table B:1 – Standard Commands available in MICAS-X.

When any of the above Commands are configured in a Configuration Editor, the Editor knows what parameters the Command needs and presents the proper options. E.g. for the Set Command, it presents a list of Controller Channels and a Value field.

When new Commands are added to new Drivers, the VI “MICAS-X Select Config Display Options.vi” can be edited to inform MICAS-X of what parameters each Command needs. However, if custom Driver Commands are added that are not supported by “MICAS-X Select Config Display Options.vi”, the Editors do not know what parameters each Command requires. Hence the Editor will present a string parameter and a numeric parameter. The user must ensure that the string parameter (if the command requires it) is filled in with the proper item and that the item is spelled exactly correctly. In addition, in the Triggers and Sequences Editors, when the mouse is above a Command control, help information for that Command is displayed.

1) The Alert command can present an Alert in numerous ways. The numeric parameter is used to specify how the Alert is presented to the operator. The value of the numeric parameter is assembled in a bit-wise fashion.
If bit 0 (value = 1) is set, the dialog is suppressed.
If bit 1 (value = 2) is set, the Alert text will also be sent to the Log file.
If bit 2 (value = 4) is set, a Sequence using the Alert Dialog will not continue until the Dialog is closed.
If bit 3 (value = 8) is set, the program will beep when the Alert is issued.
If bits 4, 5, 6, or 7 (value = 16, 32, 64, or 128) is set, the program will send the Alert text out via Email. The four different bits correspond to four different email groups, so that different recipients can be defined for different email messages. These bits only work if the Email Instrument is currently included in the configuration.
If bit 12 is set, the program plays the Alert.wav file which must be located in the Resources directory.
Similarly, if bits 13, 14, or 15 are set, the program plays the Alert1.wav, Alert2.wav, or Alert3.wav file from the Resources directory. Any desired .wav file can be renamed appropriately and placed in the Resources directory to create the desired alert sound. Note that multiple bits can be set to enable multiple behaviors. E.g. a value of 17 will suppress the dialog box and send the Alert out over Email, whereas a value of 10 will post the Alert to the dialog box, write it to the log file, and beep. Note that a value of 1 is not useful, as the Alert would take no action. Bit 0 is used in conjunction with other bits when no dialog is desired. The Alert Calc panel of the Configuration Editor can be used to help calculate the proper value for an Alert to achieve the desired functionality.

Note that the Alert, Log, and Send text can include the current value of any channel(s) by referencing the channel name between #’s. For example, if a channel named Chamber Temp (C) has a current value of 101.0, then the Alert text “Chamber temperature too high. Current value is #Chamber Temp (C)#.” would result in the actual text presented to the end user of “Chamber temperature too high. Current value is 101.0.”

2) MICAS-X was original designed for commands with only a single channel parameter. As of version 1.4.0, several commands have been added that require two channel parameters, a source channel and a target channel. These new commands include

- AddChtoCh – adds the source channel to the target channel and puts the result in the target channel.
- SubChFromCh – subtracts the **source** channel from the **target** channel and puts the result in the target channel.
- MultChByCh – multiplies the source channel by the target channel and puts the result in the target channel.
- DivChByCh – Divides the **target** channel by the **source** channel and puts the result in the target channel.
- CopyChToCh – Copies the value of the source channel into the target channel.

Pay careful attention to the use of the source and target channels, especially for the SubChFromCh and DivChByCh commands.

When writing a command as text, the channels are separated by commas. E.g. "AddChToCh OffsetValue,NewValue" When configuring Buttons or Triggers, there will be parameters visible for the source and target channels. For Sequences, however, the editor only allows for a target channel. To accommodate these two-channel commands, a new Sequence-only command has been added named "SourceCh". Use this command to set the source channel before issuing a two-channel command. For the "SourceCh" command, use the Target Channel parameter to specify the Source Channel. Note that each Sequence has a memory for one source channel. Thus it is OK to specify a source channel only once at the beginning of a Sequence, if that is the only channel that will ever be used as a source channel in that Sequence. It is more common, however, to include a SourceCh command immediately before every two-channel command in a Sequence.

3) The Command language of MICAS-X is limited to a Command, a string parameter, and a numeric value. This limitation of the syntax makes it difficult to create a command that can act on two channels, such as adding one channel's value to that of another channel. In addition to the "SourceCh" option described above, another way to overcome this limitation uses a set of mathematical commands that was created that use the numeric value as an address to a global channel. Refer to the section on Globals 9.6.13 for more information.

4) The Negate command performs a mathematical negation (multiply by negative one) to a channel's value. The Not command performs a logical negation to a channels value, for which a value of 0 becomes 1, and any non-zero value becomes 0.

In addition to the standard MICAS-X commands listed above, commands can be included in Drivers to extend their functionality. The Driver Commands below have been fully integrated into MICAS-X and are therefore reserved keywords.

Command	Driver	Parameters	Description
AirmarFileOff	Airmar	none	Turns off the recording of an ICM streaming data file.
AirmarFileOn	Airmar	none	Turns on the recording of an ICM streaming data file.
AriesDisable	Aries Drive	none	Disables the Aries Drive.
AriesEnable	Aries Drive	none	Enables the Aries Drive.
AriesHalt	Aries Drive	none	Sends a Halt command to the Aries Drive.
AriesMoveFwd	Aries Drive	none	Starts forward motion of the Aries Drive.
AriesMoveRev	Aries Drive	none	Starts reverse motion of the Aries Drive.
AriesSetPosition	Aries Drive	value	Sets the current axis position to the specified value (in mm).
AriesStart	Aries Drive	none	Starts the axis motion.
AriesStop	Aries Drive	none	Stops the axis motion.
AutoTune	PID	channel	Starts a PID autotuning wizard.
AxisNDefineHome[1]	NIMotion	none	Sets the current axis position as Home.
AxisNEStop[1]	Micronix	none	Stops all motion of the axis.
AxisNGoToHome[1]	NIMotion	none	Moves the axis to the defined Home position.
AxisNHome[1]	Micronix	none	Moves the axis to the built-in home position.
AxisNMoveAbsolute[1]	ESP7000, Micronix	none	Moves the axis to the absolution position stored in the Target Position channel.
AxisNMoveHomeToZero[1]	Miccronix	none	After issuing the AxisNHome Command, this Command can be used to send the axis to the calibrated zero position, after which the AxisNSetToZero Command should be issued.
AxisNMoveRelative[1]	ESP7000, Micronix	none	Moves the axis to the relative position stored in the Target Position channel.

Command	Driver	Parameters	Description
AxisNMoveToHole[1]	Micronix	none	After the axis is calibrated, this Command will move the axis so that the hole is centered on one of the laser beam positions.
AxisNMoveToNegLim[1]	Micronix	none	Moves the axis to the negative limit of travel.
AxisNMoveToPosLim[1]	Micronix	none	Moves the axis to the positive limit of travel.
AxisNMoveToXEdge[1]	Micronix	none	After the axis is calibrated, this Command will move the axis so that the X Edge is centered on one of the laser beam positions.
AxisNMoveToYEdge[1]	Micronix	none	After the axis is calibrated, this Command will move the axis so that the Y Edge is centered on one of the laser beam positions.
AxisNResetPosition[1]	NIMotion	value	Sets the axis's current position to the value.
AxisNSetPosition[1]	ESP7000	none	Sets the current axis position to the value stored in the Target Position channel.
AxisNSetToZero[1]	Micronix	none	Sets the current axis position to 0.
AxisNStartMotion[1]	NIMotion	none	Starts the motion of the axis.
AxisNStopMotion[1]	NIMotion	none	Stops any motion of the axis.
AxisNStopMoving[1]	ESP7000, Micronix	none	Stops any movement of the axis.
CancelProfile	PNG Laser Positions	none	This Command is used to send a message to the PNG Profiling Display to cancel any profile that is currently running.
CancelWarmup	GAM Laser	none	Cancels the warm-up timer for the GAM Laser Driver, allowing the laser to be turned on. Useful if MICAS-X has been restarted but the GAM Laser is already warmed-up.
ClearArray	Array	none	Sets all the elements of the array to 0's. Does not change the size of the array.
ClearConfig	MICAS-XRT	None	This command deletes any "AutoLoad" configuration so that

Command	Driver	Parameters	Description
			MICAS-XRT will not have a configuration file to load the next time it runs.
CorrectTrigY	PNG Laser Positions	none	Issues a command to the mirror driver to move the Trigger laser in the Y dimension by an amount specified by the Trig Y Pos (Profile) (um) Channel.
CorrectTrigX	PNG Laser Positions	none	Issues a command to the mirror driver to move the Trigger laser in the X dimension by an amount specified by the Trig X Pos (Profile) (um) Channel.
CorrectExcY	PNG Laser Positions	none	Issues a command to the mirror driver to move the Excimer laser in the Y dimension by an amount specified by the Exc Y Pos (Profile) (um) Channel.
CorrectExcX	PNG Laser Positions	none	Issues a command to the mirror driver to move the Excimer laser in the X dimension by an amount specified by the Exc X Pos (Profile) (um) Channel.
CorrectTrigPSDY	PNG Laser Positions	none	Issues a command to the mirror driver to move the Trigger laser in the Y dimension by an amount specified by the value of the Channel defined by the Trigger Y PSD Position parameter.
CorrectTrigPSDX	PNG Laser Positions	none	Issues a command to the mirror driver to move the Trigger laser in the X dimension by an amount specified by the value of the Channel defined by the Trigger X PSD Position parameter.
CorrectTimPSDY	PNG Laser Positions	none	Issues a command to the mirror driver to move the Timing laser in the Y dimension by an amount specified by the value of the Channel defined by the Timing Y PSD Position

Command	Driver	Parameters	Description
			parameter.
CorrectTimPSDX	PNG Laser Positions	none	Issues a command to the mirror driver to move the Timing laser in the X dimension by an amount specified by the value of the Channel defined by the Timing X PSD Position parameter.
DigAcquire	ADQ14, NI Scope	numeric (0 off, 1 on)	Turns off or on the acquire mode for the digitizer.
DigSave	ADQ14, NI Scope	numeric (0 off, 1 on)	Turns off or on the saving of the digitizer data to a .dig file.
DigSetTrigThresh	ADQ14, NI Scope	numeric	Sets the Internal Trigger Level.
DigSWTrigger	ADQ14, NI Scope	none	Sends a Software Trigger to the digitizer.
DigTrigChannels	ADQ14	none	Sets the digitizer triggering mode to Channel data. A Channel Flag parameter is used to determine which Channel(s) trigger the digitizer.
DigTrigChannel0	NI Scope	none	Sets the digitizer triggering mode to Channel0.
DigTrigChannel1	NI Scope	none	Sets the digitizer triggering mode to Channel1.
DigTrigExternal	ADQ14, NI Scope	none	Sets the digitizer triggering mode to External
DigTrigInternal	ADQ14	none	Sets the digitizer triggering mode to Internal
DigTrigSoftware	ADQ14, NI Scope	none	Sets the digitizer triggering mode to Software mode.
DisableTimer	Timers	channel (Timer Channel only)	Sets the timer channel to Nan to disable the timer.
DownTimer	Timers	channel (Timer Channel only)	Sets the timer channel to Count Down mode.
FileRestart	File Reader	None	Resets the file data so that the next data transmitted is the first row of the file.
GrimmFileOff	GRIMM 1p109 OPC	none	Turns off the recording of an ICM streaming data file.
GrimmFileOn	GRIMM 1p109 OPC	none	Turns on the recording of an ICM streaming data file.

Command	Driver	Parameters	Description
G2401FileOff	Picarro G2401	none	Turns off the recording of an ICM streaming data file.
G2401FileOn	Picarro G2401	none	Turns on the recording of an ICM streaming data file.
HI70FileOff	Vaisala HI70	none	Turns off the recording of an ICM streaming data file.
HI70FileOn	Vaisala HI70	none	Turns on the recording of an ICM streaming data file.
HVSetAllDefaults	PNG HVPS	None	Sets all the high voltage setpoints to their configuration file values.
MICMFileOff	MICM	none	Turns off the recording of an ICM streaming data file.
MICMFileOn	MICM	none	Turns on the recording of an ICM streaming data file.
OmegaFileOff	Omega	none	Turns off the recording of an ICM streaming data file.
OmegaFileOn	Omega	none	Turns on the recording of an ICM streaming data file.
PrimeFileOff	PrimeScales	none	Turns off the recording of an ICM streaming data file.
PrimeFileOn	PrimeScales	none	Turns on the recording of an ICM streaming data file.
PauseTimer	Timers	channel (Timer Channel only)	Pauses the specified timer channel.
RampTo%	Chamber Lighting	channel (Chamber Lighting Channel only),value	Sets the lighing channel to ramp to the value specified, using the ramp time previously stored in the RampTime channel.
ResetCounts	GAM Laser	none	Sets to accumulated trigger counts to 0.
ReadDGain[2]	Alicat	address string or Alicat index	Reads the D gain for the Alicat at the address and puts it into the Alicat Gain Channel.
ReadIGain[2]	Alicat	address string or Alicat index	Reads the I gain for the Alicat at the address and puts it into the Alicat Gain Channel. Note that only some Alicats have a I Gain.
ReadPGain[2]	Alicat	address string or Alicat index	Reads the P gain for the Alicat at the address and puts it into the Alicat

Command	Driver	Parameters	Description
			Gain Channel.
ReverseTimer	Timers	channel (Timer Channel only)	Changes the timer channel mode from its current value to the other value: e.g. count up to countdown or down to up.
SaveConfig	MICAS-XRT	None	This command causes the current configuration to be saved as the "AutoLoad" configuration, so that MICAS-XRT will load it automatically the next time it powers up.
ShowArray	Array	none	This command causes a pop-up window to appear on which one can see the current contents of the Array.
StartCWProfile	PNG Laser Positions	none	This command is used to send a message to the PNG Profiling Display to start a CW Laser profile.
StartExcProfile	PNG Laser Positions	none	This command is used to send a message to the PNG Profiling Display to start an Excmer Laser profile.
SyncTurbo	Turbo V	none	Sets the MICAS-X values of the Turbo V Driver to match those on the hardware.
SyncXGS	XGS	none	Sets the MICAS-X values of the XGS Driver to match those on the hardware.
TLPartDataFilter	TL Filter Wheel	None	Moves the ThorLabs Filter Wheel to the position specified by the value of the TL Part Data Filter Channel.
TLProfileFilter	TL Filter Wheel	None	Moves the ThorLabs Filer Wheel to the position specified by the Profiling Filter configuration parameter.
UnpauseTimer	Timers	channel (Timer Channel only)	Unpauses the specified timer channel.
UpTimer	Timers	channel (Timer Channel only)	Sets the timer channel to Count Up mode.
WriteDGain[2]	Alicat	address string or Alicat index, value	Writes the value to both the Alicat Gain Channel and to the D Gain for the Alicat at the specified address or index.

Command	Driver	Parameters	Description
WriteIGain[2]	Alicat	address string or Alicat index, value	Writes the value to both the Alicat Gain Channel and to the I Gain for the Alicat at the specified address or index.
WritePGain[2]	Alicat	address string or Alicat index, value	Writes the value to both the Alicat Gain Channel and to the P Gain for the Alicat at the specified address or index. Note that only some Alicats have a I Gain.
WriteTrigY	PNG Laser Positions	numeric (position in um)	Writes the specified position (in um) to the Trig Y Pos (Profile) (um) Channel.
WriteTrigX	PNG Laser Positions	numeric (position in um)	Writes the specified position (in um) to the Trig X Pos (Profile) (um) Channel.
WriteExcY	PNG Laser Positions	numeric (position in um)	Writes the specified position (in um) to the Exc Y Pos (Profile) (um) Channel.
WriteExcX	PNG Laser Positions	numeric (position in um)	Writes the specified position (in um) to the Exc X Pos (Profile) (um) Channel.

Table B:2 - Some of the Driver-specific Commands currently available in MICAS-X.

Note that some of the Commands documented above support customer-specific Drivers that are not otherwise documented in this manual.

1) For “AxisNCommand” commands, the “N” is replaced by the axis number of the desired axis use. For the NIMotion Driver, the axes are numbered beginning with 0. For the ESP7000 and Micronix Drivers, the axes are numbered beginning with 1.

2) When reading or writing Alicat gains, the string parameter can be the Alicat Address (A, B, C, etc., as defined in the configuration), or it can be the index of the Alicat within the configuration. Thus a string of “0” (without quotes) would be used to specify the first Alicat in the configuration, “1” would specify the second, etc.

15 Appendix C: User Dialogs and Notifications

Several MICAS-X Commands are available that allow direct interaction with the operator during sequences, triggers, and other cases. These are further described here to assist in choosing which one is most appropriate in any situation. These Commands include Ask, Ask(OK), Ask(Num), OpenNotifier, and CloseNotifier. In brief, these can be

used as follows: Ask presents a Yes/No dialog to the operator and returns the answer (1 or 0) to the program. Ask(OK) presents an acknowledge dialog to the operator that returns no information to the progam. Ask(Num) presents a dialog to the operator that allows the operator to specify a numeric value. OpenNotifier presents a notification to the operator that returns no data to the program and does not cause the calling sequence to wait on any response. CloseNotifier closes the notification if it is open. Note that the dialogs/windows that these Commands invoke are Floating but not Modal, e.g. while these dialogs are open, one can navigate throughout the MICAS-X system while viewing and interacting with it. In addition, with the exception of the Notification, these dialogs can also be instantiated multiple times, with each instance independent of all the others. Note that since these are user interface commands, they are not supported under MICAS-X-RT.

15.1.1 Alert

Parameters: Alert Code (Numeric), Text (string)

The Alert Command can be configured to alert the operator to conditions through a number of mechanisms. The value of the string parameter (Target) is the text that is presented to the operator. The numeric Alert Code parameter determines how the Alert is presented. This value is a bit-wise integer. To determine how the Alert is presented, add together the bit values of the options that you want and set the Value to the result. Note that if Value = 0, the default Alert mechanism is to present a dialog window to the user with the Alert text and an acknowledge button.

Alert Code Bit Values

Bit	Value	Behavior
0	1	Suppress Dialog
1	2	Send Alert to Log File
2	4	Pause Sequence Until Dialog is Acknowledged
3	8	Beep
4	16	Send Alert to first Email Group
5	32	Send Alert to second Email Group
6	64	Send Alert to third Email Group
7	128	Send Alert to fourth Email Group
8	256	Say the Alert using Text to Speech
9	512	Reserved
10	1024	Reserved

11 2048 Reserved

12 4096 Play Alert1.wav

13 8192 Play Alert2.wav

14 16384 Play Alert3.wav

15 32768 Play Alert4.wav

As an example, to cause the Alert to Beep and not have an Alert Dialog, use a value of 9 (8 (beep) + 1 (suppress dialog)).

Note that the Alert Calc tool in the Modules menu of the Configuration Editor can help determine the value you should use for a given behavior.

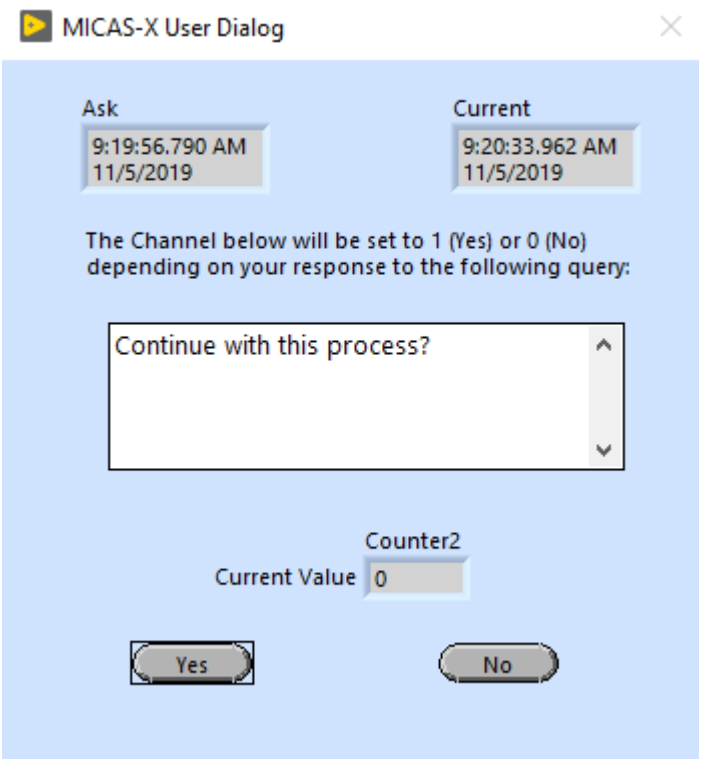
Bits 4, 5, 6, and 7 require that the optional Email Instrument be installed for them to work.

Note that the Alert text can contain current channel values by enclosing the desired channel names between #'s. For example, if the channel "Chamber Temp (C)" has a value of 101, then the Alert text of "Chamber Temperature too Hi. Current value = #Chamber Temp (C)#." will result in an alert that says "Chamber Temperature too Hi. Current value = 101."

15.1.2 Ask

Parameters: Text (string), Channel (string)

The Ask Command presents a Yes/No dialog to the operator. The string parameter (in the Target Channel field of the Editor) is used as the text of the dialog. The Source Channel, which must be set by a previous SourceCh Command, determines the Channel that the operator response is sent to. E.g. for this specific Command, the SourceCh parameter the data Channel that receives the result of the dialog. Typically the SourceCh parameter should refer to a Channel that is defined in the Manual Driver.



Example of Ask Command user dialog.

For example, using the text "Do you wish to continue?" and the SourceCh "Answer", after this Command executes, the Channel "Answer" will have a value of 1 if the user pressed the Yes button and it will have a value of 0 if the user pressed the No button.

This Command allows Sequences to branch according to user input, rather than just according to programmatic Channel values. For the example above, the Command immediately following the Ask Command could be a Goto Command, with a Condition set for the "Answer" Channel equal to 1. Note that the Sequence which calls this Command will not execute any further until the operator has answered the dialog. In order to help the operator determine how quickly they responded to a dialog, the dialog presents both the time that the dialog first opened as well as the current time.

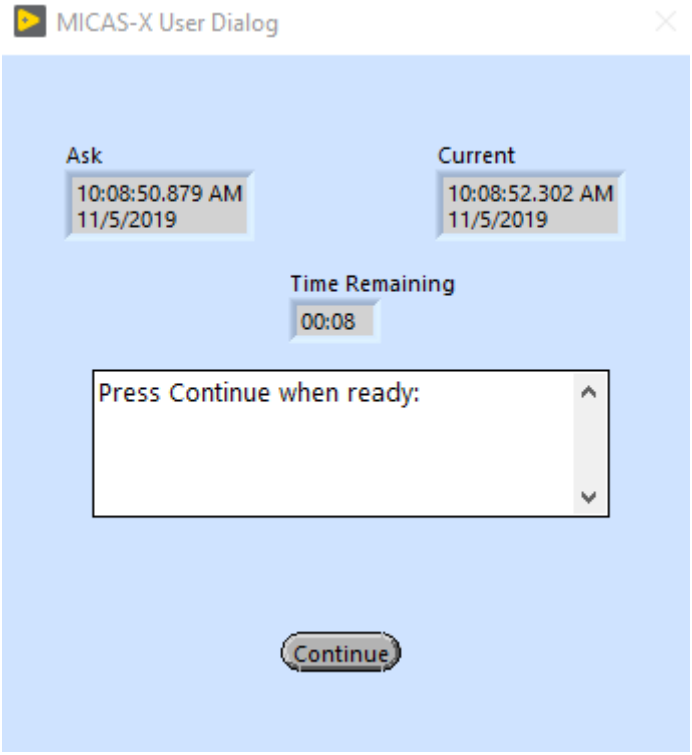
The Ask dialog can also be used programmatically in other MICAS-X modules as a convenient way to return data from the operator, as has been done for several Utilities. In this case, there are some additional options that are not available when the Ask Command is used. To call the Ask Dialog programmatically, use the MICAS Call Ask Dialog.vi located in the Common directory. In addition to inputs for the Channel and Text, this VI has optional inputs that allow one to specify the text on the “Yes” button and the “No” button, and an optional “Hide Channel?” input, which if True, makes the

“Current Value” Channel indicator invisible. Each instance of this VI creates its own instance of the dialog, so it can be used multiple times in parallel.

15.1.3 Ask(OK)

Parameters: Text (string), Timeout (numeric)

The Ask(OK) Command presents an OK dialog to the operator. The string parameter (in the Target Channel field of the Editor) is used as the text of the dialog. This Command is intended to supply the operator with information or instructions and wait for the operator's acknowledgement, but without receiving any information back from the operator. E.g. it can pause a Sequence until the operator has completed a manual adjustment to the equipment, then continue the Sequence when the operator is ready. Note that the Sequence which calls this Command will not execute any further until the operator has answered the dialog. In order to help the operator determine how quickly they responded to a dialog, the dialog presents both the time that the dialog first opened as well as the current time. The optional Timeout parameter is used to specify the maximum time that the dialog will be open. E.g. if this parameter is greater than 0, the dialog will close even without user input after the specified number of seconds.



Example of Ask(OK) Dialog with Timeout > 0.

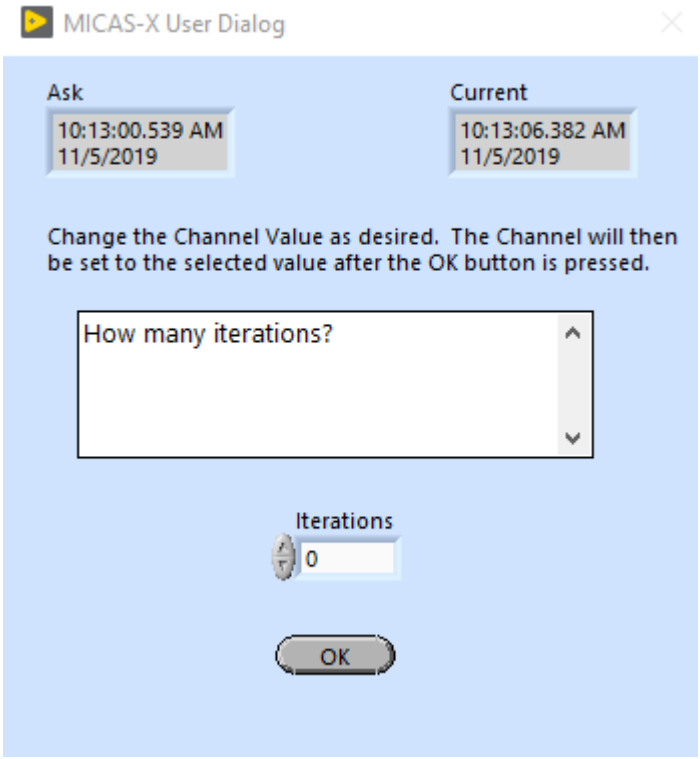
15.1.4 Ask(Num)

Parameters: Text (string), Channel (string)

The Ask(Num) Command presents a dialog to the operator which the operator responds to with a numeric value. The string parameter (in the Target Channel field of the Editor) is used as the text of the dialog. The Source Channel, which must be set by a previous SourceCh Command, determines the Channel that the operator response is sent to. E.g. for this specific Command, the SourceCh parameter really defines a Target Channel.

For example, using the text "How many iterations" and the SourceCh "Iterations", after this Command executes, the Channel "Iterations" will have specified by the operator before they pressed the "OK" button of the dialog.

This Command allows Sequences to execute based on operator input at the time of the test or Sequence, rather than just on Channel values and information the operator had when they created the Sequence in the Editor. Note that the Sequence which calls this Command will not execute any further until the operator has answered the dialog. In order to help the operator determine how quickly they responded to a dialog, the dialog presents both the time that the dialog first opened as well as the current time.



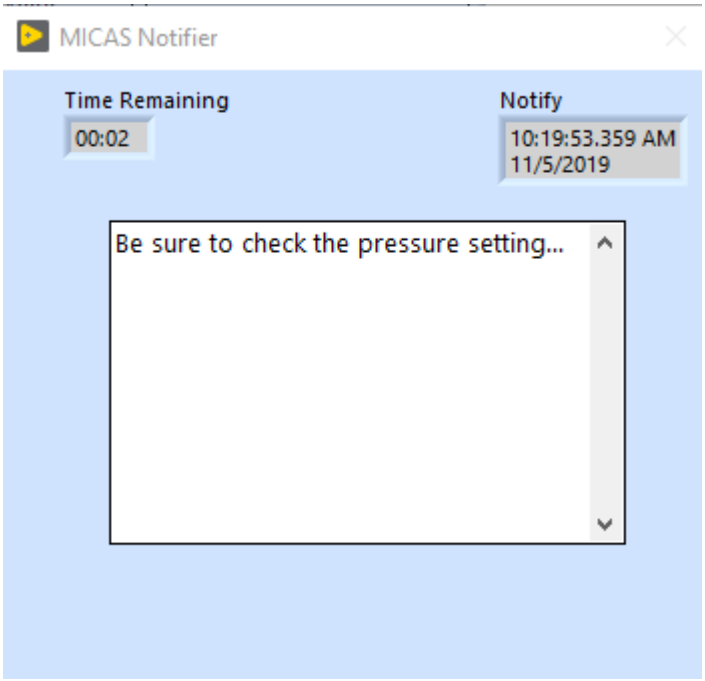
Example of Ask(Num) Dialog.

15.1.5 OpenNotifier

Parameters: Text (string), Timeout (numeric)

This Command opens the Notifier window and displays the specified text. If timeout is not equal to 0, the window will close automatically after the absolute value of the time-out seconds have passed. If the timeout is less than zero, a "Continue" button is pressed that allows the user to close the dialog window before the timeout has expired. E.g. use a positive timeout to enable a timeout with no "Continue" button, and a negative timeout to enable a timeout with a "Continue" button. With a zero timeout, the Notifier will only close when the Close Notifier Command is used.

Note that the Notifier window is unique and cannot be instantiated more than once.



Example Notifier window with positive timeout value.

15.1.6 CloseNotifier

Parameters: none

This command closes the Notifier window. The Notifier window can also be closed by pressing the “Continue” button (when configured to show the “Continue” button), when its timeout expires (if configured with a non-zero timeout), or when MICAS-X stops.

16 Appendix C: Using Sequences in MICAS-X

Although Sequences in MICAS-X are very powerful and flexible, they do have limitations and can get confusing. This section describes some of the uses and caveates for Sequences.

A Sequence in MICAS-X is made up of one or more Sequence Steps. Each Step contains five main components: the Label, the Command, the Condition, the Target, and the Value.

The Label of a Sequence Step serves two purposes. It is a descriptive text that helps the sequence author understand what each step does. It can also be a Target for the Goto Sequence Command. MICAS-X does not enforce the rule that all Labels should be unique, but it is good practice. Labels that are used as Targets for Goto Commands should definitely be unique within that Sequence, so that one can be sure where the Goto refers to. Note that Labels have a scope limited to the Sequence that they are in. E.g. duplicate Labels in different Sequences are not a problem.

The Command defines the action that a Sequence Step will take, though this action is only taken if the Step's Condition is evaluated as True. The Commands that are available within Sequences include all MICAS-X Commands. This includes all the standard Commands listed in Table B:1 as well as any Driver-specific Commands for Drivers that have been included in the active configuration, as shown in Table B:2. Also note that the last few Commands in Table B:1 are specific to Sequences and cannot be used in Triggers, Alarms, or other places where Commands are used.

The Condition for a Step is defined by the Condition parameter as well as the Condition Channel and the Threshold. The Condition is evaluated by comparing the value of the Condition Channel to the Threshold, using the Condition parameter. The Condition Parameter has the options True, < (less than), <= (less than or equal), = (equal), <> (not equal), >= (greater than or equal), > (greater than), =NaN (equal to Not-A-Number), <>NaN (not equal to Not-A-Number), <>Nan,0 (not equal to Not-A-Number or Zero), and False. As of version 2.0.3 of MICAS-X, additional conditions have been added which allow the Condition Channel to be compared to another channel (the Condition Source Channel), rather than just to a constant. These conditions include <Ch (less than the Condition Source Channel), <=Ch (less than or equal to the Condition Source Channel), =Ch (equal to the Condition Source Channel), <>Ch (not equal to the Condition Source Channel), >-=Ch (greater than or equal to the Condition Source Channel), and >Ch (greater than the Condition Source Channel). If one of these latter Conditions are used, it is imperative that the Condition Source Channel be previously defined using the CondSource Command.

The True Condition is the most commonly used. It means that the Sequence Step will always execute when a Sequence reaches it. The False Condition is much less frequently used, since it means the Step will never execute. The False Condition can be

useful for “commenting out” Sequence Steps in the Configuration Editor, but not removing them completely, so that they can easily be added back to the Sequence in the future.

The <, <=, >, >=, =, and <> Conditions use their standard definitions. Note that = and <> do not have a “delta” parameter in Sequences, in contrast to the “Hysteresis” parameter for Triggers. For Triggers, the “Hysteresis” parameter is used as a bandwidth, so that = and <> can be used for “Nearly Equal” and “Out Of Range”.

NaN is a IEEE-defined value meaning “Not-A-Number”. It is used in MICAS-X by many Drivers to indicate that a value could not be read from the related hardware for some reason. Within the IEEE definition, any normal comparison of any number (including NaN) with a value of NaN will result in False. Thus if the following condition were evaluated:
Condition: = Condition Channel: Counts Threshold: NaN

while the channel Counts had the value “NaN”, the answer would be (rather nonintuitively), False. In order to allow Conditions that can specifically identify NaN, the Conditions =NaN, <>NaN, and <>NaN,0 are available. These Conditions can be useful when writing Sequence Steps that need to be able to recognize when a Driver is working, e.g. when it's channel's value is not NaN.

Commands in MICAS-X were initially defined as having a Command (string), an optional Target (string), and an optional value (real). Later, some commands were defined that include a Source (string) as well, so that, for example, two channels can be added together. E.g. the AddChToCh Command adds the value of the Source Channel to the Target Channel and puts the result in the Target Channel. The Sequence Editor cannot directly accommodate the Source Channel, so the special Sequence Command “SourceCh” was added. Whenever a Command which requires a Source Channel is to be used, it should be immediately preceeded by the “SourceCh” Command, so that the Sequencer knows what Channel to use as the Source Channel. If the same Source Channel will be used repeatedly and exclusively within a Sequence, it is OK to set that Source Channel once at the beginning of the Sequence with the SourceCh Command and not set it again. Although this practice works, it makes the Sequence less readable, and is prone to bugs if another Source Channel is later added at some point in the Sequence.

Just as the lack of a Source Channel was problematic with Commands, previous to version 2.0.3, the Sequence Conditions also suffered from a limitation due to the fact that only one Condition Channel could be defined. Many Conditions can be defined in terms of one variable, such as terminating a Sequence loop when a Count channel exceeds a value. However, it is often the case that it is useful to have a Condition that compares the values of two channels. E.g. the Sequence might execute a Step if the ProcessTemp Channel is greater than the TempSetPoint Channel. Since each Sequence

Step only refers to a single Condition Channel, this is problematic. The previously suggested solution (which still works) was to create a new channel using the Equations Driver, such that the value of the new Channel is the difference of the two Channels in question. E.g. an Equation Channel named TempDifference could be defined as (ProcessTemp - TempSetPoint). Then the Condition could be applied to the TempDifference Channel with the Condition of “>” and a threshold of 0.

With the introduction of Condition Source Channels in MICAS-X 2.0.3, the above example can be addressed in an even easier and cleaner fashion. Before the Sequence Step that includes the condition in question, use the CondSource Command to set the Condition Source Channel to TempSetPoint. The Step in question would then use the Condition >Ch to compare the ProcesTemp Channel value to the TempSetPoint Channel value, and the TempDifference Channel would not have to be created at all.

When writing Sequences, it is important to consider how and when the various Channels are being acquired or calculated. In the example above, using the TempDifference Channel, it would be wise to ensure that the Equations Driver is in the same acquisition loop as the other two channels and that it is listed after the Drivers that contain the other two Channels. That way, the value of the TempDifference Channel will always be calculated using the current values of the other two Channels. If the Equations Driver were listed before the Drivers containing the other two Channels, it would use the previous iteration's values of those Channels to compute the value of TempDifference, with the result that the TempDifference Channel will always be one iteration out-of-date.

Another consideration when using Sequences is important when creating Sequences for stimulus-response type situations. For example, imagine a Sequence that moves a sensor over a raster pattern, and reads the value of the sensor at each position. At a high logic level, the Sequence would look something like:

1. Move to a new position
2. Read sensor
3. Check if all positions have not been done, loop back to Move (1)
4. Stop sequence

In implementing this Sequence, it is important to consider the timing of the system. Sequences and Driver are asynchronous in MICAS-X, so additional logic needs to be included in order to ensure that the above sequence operates correctly. For example, one might want to ensure that the sensor has finished moving and that the sensor channel value has updated before moving to the next location. This might be accomplished by a sequence such as this:

1. Move to a new position

2. Wait 1.1 seconds (to ensure that the move started)
3. Check if move is not done, loop back to Wait (2)
4. Check if sensor channel does not have a new value, loop back to 4
5. Check if all positions have not been done, loop back to Move (1)
6. Stop sequence

In this example, the time to wait in Step 2 depends on the hardware in use and the configuration of the Driver controlling the motion. Note that when MICAS-X sets a channel to a new channel value or executes a Driver Command, those actions act immediately and are not synchronized to the Acquisition Loop. Hence the Move Command (1) will execute very soon (milliseconds) after the Sequence issues it. However, to read from the motion Driver whether the motion is completed or not, one needs to be sure that the Sequence waits long enough that the Channel indicating whether or not the hardware is moving has been updated by the Driver. If the motion Driver is in a 1 Hz acquisition loop, a wait of 1.1 seconds should be sufficient to ensure that the move started AND the motion Driver has updated its channels.

Step for of the above Sequence could be implemented with the Wait(NewValue) command. This command takes a Channel and a wait time as arguments, and will check the channel for a new value, waiting by the wait time between each check. A relatively short time, such as 0.1 seconds, can be used for this command, reducing the latency of this wait. Note, however, that this Command assumes that the Channel being monitored WILL change. If the Channel being monitored has no appreciable noise or is not guaranteed to change, this Wait(NewValue) Command could hang indefinitely.

Finally, note that in the above example, the Wait(NewValue) was not suggested for Step 2. Instead, a longer Wait(Value) Command was used. This is due to the nature of motion control. For example, before the Command to move to a new position was entered, the motion system was likely at rest, and the motion Driver's “Done?” Channel was likely returning a value indicating that all motion was complete. (For this example, assume that Done? = 1 means that the motion hardware is stationary.) If the Wait(NewValue) Command had been issued immediately after the Move command, it may well have triggered on the Done? = 1 to Done? = 0 transition, when the motion hardware first starts to move, which would not be appropriate. A fixed Wait is therefore needed to ensure that the motion was started before the Check was made.

Note that Sequences are one of the few things in MICAS-X that can be changed while MICAS-X is running. For almost all other parameters, you must edit the configuration file and restart MICAS-X for changes to be made. Sequences, however, can be added to the MICAS-X configuration while the program is running. This is done by editing the current configuration file in the Configuration Editor, and pressing the Add to

Current Configuration button after highlighting the desired Sequence in the Sequences list on the left side of the Sequences Editor.

As of version 2.1.4 of MICAS-X, Scripts (Section 7. 4. 8) are also available. Scripts are a more powerful form of Sequences. They are executed in MICAS-X by the Sequence Module. Unlike Sequences, the Scripts themselves are not stored in the configuration file, but are stored in the Scripts sub-directory of the Support directory.

Scripts are stored as text and can be edited in a text editor or using Utilities created for editing them. Their format is similar to Sequences, and is documented in the above referenced section. The same Commands and parameters that are used in Sequences can be used in Scripts. In addition, the Script lines can have a Comment parameter, which is used only for documentation.

The main differences between Scripts and Sequences are that 1)Scripts can contain an Expression in any location that normally takes a Value, and 2)a single Script line can only contain one command, but it can contain multiple sets of parameters for that command to operate on. Because of these two differences, Scripts can be more compact and easier to understand than Sequences. With Sequences, if one wants to compare two Channel values in a Condition, one must first use the CondSource command to set the second (Source) Condition Channel, then use one of the Conditions that uses a second Channel. Alternatively, additional an additional Channel can be created with the Equations Driver which takes the difference of two channels, and that new Channel can be used in the Condition. With Scripts, neither of these extra steps is necessary.

17 Appendix D: Syntax for Equations

The Equations Driver allows users to create and calculate their own channels by writing text equations involving existing channels. Scripts also allow the use of Expressions as defined in this section. The table below describes the syntax for the available functions that can be used for these equations. In addition to the syntax supportted below by National Instruments, as of version 2.0.0, the MICAS-X Equations Driver also supports a number of customer functions. Those custom functions are described in the second table below.

Function Syntax	**Function Name**	**Description**
abs(x)	Absolute Value	Computes the absolute value of x.
acos(x)	Inverse Cosine	Computes the inverse cosine of x.

Function Syntax	Function Name	Description
acosh(x)	Inverse Hyperbolic Cosine	Computes the inverse hyperbolic cosine of x in radians.
asin(x)	Inverse Sine	Computes the inverse sine of x in radians.
asinh(x)	Inverse Hyperbolic Sine	Computes the inverse hyperbolic sine of x in radians.
atan(x)	Inverse Tangent	Computes the inverse tangent of x in radians.
atanh(x)	Inverse Hyperbolic Tangent	Computes the inverse hyperbolic tangent of x in radians.
ci(x)	Cosine Integral	Computes the cosine integral of x where x is any real number.
ceil(x)	Round to +Infinity	Rounds x to the next higher integer (smallest integer >= x.)
cos(x)	Cosine	Computes the cosine of x in radians.
cosh(x)	Hyperbolic Cosine	Computes the hyperbolic cosine of x in radians.
cot(x)	Cotangent	Computes the cotangent of x in radians (1/tan(x)).
csc(x)	Cosecant	Computes the cosecant of x in radians (1/sin(x)).
exp(x)	Exponential	Computes the value of e raised to the power x.
expm1(x)	Exponential(Arg)—1	Computes the value of e raised to the power of x— 1 ($e^x - 1$).

Function Syntax	Function Name	Description
floor(x)	Round to —Infinity	Truncates x to the next lower integer (Largest integer <= x)
gamma(x)	Gamma Function	Γ(n + 1) = n! for all natural numbers n.
getexp(x)	Mantissa and exponent	Returns the exponent of x.
getman(x)	Mantissa and exponent	Returns the mantissa of x.
int(x)	Round to nearest integer	Rounds its argument to the nearest even integer.
intrz	Round toward zero	Rounds x to the nearest integer between x and zero.
ln(x)	Natural Logarithm	Computes the natural logarithm of x (to the base e).
lnpl(x)	Natural Logarithm (Arg+1)	Computes the natural logarithm of (x+1).
log(x)	Logarithm Base 10	Computes the logarithm of x (to the base 10).
log2(x)	Logarithm Base 2	Computes the logarithm of x (to the base 2).
pi(x)	Represents the value π = 3.14159	pi(x) = x * π pi(1)= π pi(2.4) = 2.4* π
rand(0)	Random Number (0—1)	Produces a floating-point number between 0 and 1.
sec(x)	Secant	Computes the secant of x (1/cos(x)).
si(x)	Sine Integral	Computes the sine integral of x where x is any real number.

Function Syntax	Function Name	Description
sign(x)	Sign	Returns 1 if x is greater than 0. Returns 0 if x is equal to 0. Returns -1 if x is less than 0.
sin(x)	Sine	Computes the sine of x in radians.
sinc(x)	Sinc	Computes the sine of x divided by x in radians (sin(x)/x).
sinh(x)	Hyperbolic Sine	Computes the hyperbolic sine of x in radians.
spike(x)	Spike function	Returns: 1 if 0 <= x <= 1, 0 for any other value of x.
sqrt(x)	Square Root	Computes the square root of x.
square(x)	square(x)	square (x) returns: 1 if 2n <= x <= (2n +1), 0 if 2n + 1 <= x <= (2n +2) where x is any real number and n is any integer.
step(x)	Step function	step(x) returns: 0 if x < 0, 1 if any other condition applies.
tan(x)	Tangent	Computes the tangent of x in radians.
tanh(x)	Hyperbolic Tangent	Computes the hyperbolic tangent of x in radians.

The above table is excerpted from the G Math Toolkit Reference Manual produced by National Instruments. A complete version is available online.

The functions built in to the G Math Toolkit take a single value as an argument. Many of custom functions added to the Equation Driver by OCC may take an arbitrary number of channel names or constant values as arguments and act on all the values. Other custom functions operate on the channels as if they were Boolean values (True/False, where 0 or NaN = False and all other values (most typically 1) are True). The main advantage of using these Boolean operations is that they all specifically filter NaN out to be 0 or False, whereas simple math (such as using multiply for AND or add for OR) would generally treat NaN as True.

Note that the custom function added by OCC are not computed recursively! E.g. only simple expressions can be used with these functions, not nested expressions. For example:

And(channel1,channel2,channel3) is allowed.

Not(Or(And(Channel1,Channel2),Channel3) is not allowed.

Funtion Syntax	**Function Name**	**Description**
Max(x,y,z...)	Maximum Value	Returns the largest of the values that are passed in as arguments.
Median(x,y,z...)	Median Value	Returns the median of the values that are passed in as arguments.
Min(x,y,z...)	Minimum Value	Returns the smallest of the values that are passed in as arguments.
Rate(x,i,n)	Rate of change of a channel's value	Calculates a linear fit over the last n points of the channel x and returns the slope. Note that time is implicit and is whatever time interval is used to acquire new points and calculate the rate. (It is best to keep the Equation Driver in the same loop as the channel for which the Rate is calculated.) The index i must be a unique integer >= 0 for each equation that uses this function.
Bool(x)	Converts to Boolean (0 or Nan = False, 0; anything else = True,1)	This function takes a single channel as an argument. The main advantage of this function is that it converts NaN to 0, whereas a channel used as a Boolean otherwise would take NaN as True.
Not(x)	Converts the Negated Boolean value (0 or Nan = True, 1; anything else = False, 0)	Similar to Bool, but logically negated.
And(x,y,z…)	AND's all the channels	Takes any number of channels, converts them all to Boolean (including using NaN's as False) and computes a value of 0 or 1 based on AND'ing the channels. The result is 1 if all of the channels are True (e.g. not 0 and not NaN).
Or(x,y,z…)	OR's all the channels	Takes any number of channels, converts them all to Boolean (including using NaN's as False) and computes a value of 0 or 1 based on OR'ing the channels. The result is

		1 if any one of the channels is True (e.g. not 0 and not NaN).
Nand(x,y,z…)	NAND's all the channels	Same as And, but negated result.
Nor(x,yz…)	NOR's all the channels	Same as Or, but negated result.
Xor(x,y)	Exclusive Or of two channels	The output is 1 if either of the channels is True, but not both.
Nxor(x,y)	Negated Exclusive Or of two channels	The output is 1 if both channels are False or both channels are True.
ReplaceNaN(x)	Replaces channel values of NaN with x	See notes below.

Custom functions added to the Equations Driver in MICAS-X.

Note that the values Inf, -Inf, and NaN are not supported by the G Math Toolkit. If any input channels have these values, the output of the equation will be set to NaN.

Note, however, that the ReplaceNaN(x) custom function can be used in some situations to deal with NaN values. This function takes a numeric argument, not a channel name, and replaces the value of any channels in the equation that currently have a value of NaN with that numeric value. One use of this channel is when dealing with Boolean-like channels, which are intended to have a value of 0 or 1. These channels may be added or multiplied together to create OR- and AND-type functions. However, if a Driver is disabled, its channels return the value NaN. In this case, an equation adding or multiplying multiple channels will result in a NaN value, which may not be the desired behavior. By using the ReplaceNaN(x) function, NaN can be mapped directly into either a 0 or a 1, depending on the desired behavior.

18 Appendix E: Error Messages

Error codes reported in MICAS-X are generally those defined in LabVIEW. There are a few error codes, however, that were created specifically for the MICAS-X system. These error codes range from 7001 to 7999 and are documented here:

Error Code	**Meaning**
7001	Channel or name not found.
7002	Requested channel is not an output channel.
7003	Auto-Threshold algorithm returned a value of 0. Old threshold was used. (Specific to the SP2 Instrument.)
7004	Channel name is empty. Cannot set unknown channel.
7006	Auto Sampler error: Illegal or missing perameter. (Specific to the SP2

	Instrument with Autosampler.)
7007	Auto Sample error: (Specific to the SP2 Instrument with AutoSampler)
7008	Unknown command.
7009	Function not yet implemented.
7010	Security Device not found. Cannot execute command.
7011	Requested List not found.
7012	The requested Command cannot act on the Requested Channel.
7013	Checksum Invalid.
7014	Turbo V communication error.
7015	No Security Device Detected.
7016	No Security Device detected. Now in unlicensed mode.
7017	Resource was not registered correctly.
7018	An XGS gauge set to Auto-On cannot be turned off or on. That must be done on the XGS hardware control panel.
7019	Cannot turn the GAM Excimer on until the warm-up time has expired.
7020	Security Device expiration date has passed.
7021	Security Device expiration date has passed. Now in unlicensed mode.
7022	Lost TCP connection.
7023	Incorrect number of controllers sent to UWyo UF RT.vi. (This error is specific to the UWyo UF Driver.)
7024	Programming error. Please contact Original Code Consulting.
7025	The Array is not configured to allow itself to grow beyond its initial size. (This error occurs if “Allow Array to Grow” is not checked and the array is written to with an index greater than or equal to the initial array size.)
7026	Tab Command cannot be used for a Tab which is not visible.
7027	Email could not be sent. Check your email configuration.
7028	The requested Sequence cannot be started since it is already running.
7029	The requested Sequence cannot be stopped since it is not running.
7030	The requested Sequence cannot be paused since it is not running.
7031	The requested Sequence cannot be paused since it is already paused.
7032	The requested Sequence cannot be unpaused since it is not paused.
7033	Configuration File CRC is not valid or missing.
7034	Backup Configuration File has been defined as the Startup Configuration file due to CRC failure.
7035	Cannot Set a channel when Acquire is False.
7036	Cannot execute a Command when Acquire is False.
7037	Cannot find the Driver requested.
7038	Command buffer is growing. Commands are not getting processed properly. This can be caused by Acquisition Loops not having enough time to complete their tasks.
7039	The configuration file being loaded in MICAS-X for Windows is marked as

	being targeted to MICAS-X-RT for Real-Time. Functionality may be impaired.
7040	The configuration file being loaded in MICAS-X-RT for Real-Time is marked as being targeted to MICAS-X for Windows. Functionality may be impaired.
7041	The requested Command is not supported in MICAS-X-RT.
7042	Not all the Drivers were initialized before other modules started. Some functions may have failed. Report this error to OCC for support.
7043	Cannot set a PID Gain when the Mode is 0 (Manual) or -1 (Disabled).
7044	Cannot set the Thermo 49C PS SetPoint below 5 ppb.
7045	Error setting the Thermo 49C PS SetPoint.
7046	Could not parse the Script Condition.
7047	Command queue was overflowing. Cleared the Command queue of over 1000 unprocessed commands.
7048	Sequencer did not initialize in time. Sequences may not have loaded properly.
7049	Look Up Table not found.
7050	Adobe Acrobat must be properly installed on this computer in order to view .pdf files in the Document Display.
7051	Cannot set a Channel on the RT system. Connection may have been lost.
7052	Obis Laser Error.
7053	Device not found. The serial number searched for was not reported as available.
7054	Error arming the ADQ.
7055	ADQ error.
7056	Time-out expired before stage stopped moving.
7057	Micronix Error.
7058	Time-out expired before N laser measurements could be made.
7059	Time-out expired before a new Excimer Joulemeter measurement was made.
7060	Did not get both NI Scope signals before time-out.
7061	Laser cannot be turned on before the warm-up is finished.
7062	Invalid data read from HVPS. No Beacon character detected.
7063	Time-out expired before instrument reply was complete.
7064	Measured laser position offset was larger than the maximum adjustment allowed. Laser position will not be moved.
7065	Filter wheel was moved to a position which does not have a defined filter value.
7066	High Voltage could not be enabled due to software interlock condition.
7067	Invalid Filter Position was commanded. Coercing to a valid position.

7068	Cannot set the Obis laser diode temperature outside the specified range.
7069	Pololu Driver setpoing channel cannot be set higher then n-1, where n is the number of positions.
7070	K535 Driver cannot have cooling and regeneration turned on at the same time.
7071	Acqiris Instrument did not receive particle speed data from ATOFMS timing Instrument before the time-out expired.
7072	User selected O2 concentration is less than envirnomental O2. Cannot set flow controller less than 0.
7073	User selected CO2 concentration is less than envirnomental CO2. Cannot set flow controller less than 0.
7074	Error reading from MKS flow controller.
7075	Acquisition Loop took more than three times its configured Loop Time to execute.
7076	The ADQ14 cannot change its Acquire state when it is not Enabled. Use the ADQ14 Driver to Enable the ADQ14 Instrument before chaging its Acquire state.
7077	SingleStepSeq Command cannot operate on a Sequence that is not running and paused.
7078	Wait For Different Value timed-out. Actions may be taken before they were intended to.
7079	Wait For New Value timed-out. Actions may be taken before they were intended to.

19 Appendix F: Web Power Switch 7 Watchdog Function

MICAS-X can enable a watchdog function using the Web Power Switch 7. This creates a safety mechanism that can shut down the AC power to a device in the case that MICAS-X or the computer that it is on should crash or stop running unexpectedly. In order for this functionality to work, it is necessary to enable the watchdog function in the Web Power Switch 7 Driver configuration, as well as to configure several scripts on the Web Power Switch 7 itself. The required scripts are documented here. In addition to configuring the scripts below on the Web Power Switch 7, the Enable Watchdog parameter must also be set True in the Web Power Switch MICAS-X configuration.

To program the scripts, one must open the Web Power Switch 7 (WPS7) interface from a web browser. To do this, enter the IP address of the WPS7 in the web browser address bar. For example, if the IP address of the device is 192.168.1.100, enter “http://192.168.1.100” in the address bar of your web browser. Click on the Scripting link on the left of the resulting page.

On the Scripting page, you will need to enter two User Strings and several lines of Scripts, as shown below. Enter each item one at a time, then press the associated Edit button. Note that the Script items cannot be edited if the "Enable Scripting" check box is set. If the "Enable Scripting" check box is marked, first uncheck it and press "Submit". After editing all User Strings and Scripts, you must check the "Enable Scripting" button and press "Submit" again before the scripts are ready to run on the WPS7.

The directions below assume that Plug or Outlet 1 is being guarded by the Watchdog scripts. You can customize the scripts to act on any of the plugs as desired.

Under "User Strings", enter "\f\1REFRESH %O\2%d" without the quotes, and press "Edit". The \1 and \2 tokens instruct the WPS7 to write to the first and second lines of its display. %O causes it to display the current state of the eight outlets, and %d causes it to display the current date and time.

For the second "User String", enter "\f\1P1 OFF %O\2%d" without the quotes, and press "Edit". In this case, the text "P1 OFF" is displayed to indicate that plug 1 is being turned off. If you wish to guard a different plug(s) with the watchdog script, you can replace that text with other descriptive text as desired. However, it is best to limit this text to exactly seven characters, using spaces as necessary.

For Script Line 10, enter "KILL 0" (without quotes) and press "Edit". This script stops all other scripts. MICAS-X calls this script when it stops, so that the Watchdog functionality ends when MICAS-X stops. If this is not done, the guarded outlet(s) will be turned off shortly after MICAS-X stops running. If that is the desired behavior, leave line 10 with the default "End" entry.

For Script Line 20, enter "KILL 0".

For Script Line 21, enter "DISPLAY 1"

For Script Line 22, enter "GOTO 30"

These three script lines are called by MICAS-X once per acquisition loop. This refreshes the Watchdog function and keeps the guarded plug(s) turned on (if they are already on). Note that the Watchdog functionality does not turn ever turn the guarded plug(s) on. That must be done by the user manually at the WPS7, or from within MICAS-X.

For Script Line 30, enter "SLEEP 15".

For Script Line 31, enter "OFF 1"

For Script Line 32, enter "DISPLAY 2"

These three script lines are responsible for shutting the guarded plug(s) off if MICAS-X does not refresh the watchdog. Line 30 waits for 15 seconds. Note that this value can be adjusted as desired. This value should be 1.5 to 2 times longer than the

MICAS-X WPS7 Acquisition Loop time. If this value is shorter than the acquisition loop time, the scripts will turn off the plug(s) prematurely.

Line 31 turns off the guarded plug(s). Therefore, the value “1” should be changed to reflect which plugs you wish to have the watchdog guard. The plugs are numbered 1 to 8, and multiple plugs are specified by stringing their numbers together. Thus “OFF 135” will turn off plugs 1, 3, and 5, and “12345678” will turn off all the plugs.

Line 32 displays the second user string. This provides visual confirmation to the user that the watchdog functionality was activated.