

## YOMX Communication Driver

Driver for Ethernet Communication  
with YOKOGAWA MX100 Devices

### Contents

<b>INTRODUCTION .....</b>	<b>2</b>
<b>GENERAL INFORMATION.....</b>	<b>3</b>
DEVICE CHARACTERISTICS .....	3
LINK CHARACTERISTICS.....	3
DRIVER CHARACTERISTICS .....	3
CONFORMANCE TESTING .....	4
<b>INSTALLING THE DRIVER .....</b>	<b>5</b>
<b>CONFIGURING THE DRIVER .....</b>	<b>6</b>
SETTING THE COMMUNICATION PARAMETERS .....	6
CONFIGURING THE DRIVER WORKSHEETS .....	8
<b>EXECUTING THE DRIVER .....</b>	<b>11</b>
<b>TROUBLESHOOTING .....</b>	<b>12</b>
<b>SAMPLE APPLICATION .....</b>	<b>13</b>
<b>REVISION HISTORY.....</b>	<b>14</b>

## Introduction

The YOMX driver enables communication between the Studio system and YOKOGAWA MX100 devices communicating over Ethernet, according to the specifications discussed in this document.

This document was designed to help you install, configure, and execute the YOMX driver to enable communication with these devices. The information in this document is organized as follows:

- **Introduction:** Provides an overview of the YOMX driver documentation
- **General Information:** Provides information needed to identify all the required components (hardware and software) used to implement communication between Studio and the YOMX driver
- **Installing the Driver:** Explains how to install the YOMX driver
- **Configuring the Driver:** Explains how to configure the YOMX driver
- **Executing the Driver:** Explains how to execute the driver to verify that you installed and configured the driver correctly
- **Troubleshooting:** Lists the most common error codes for this protocol and explains how to fix these errors
- **Sample Application:** Explains how to use a sample application to test the YOMX driver configuration
- **Revision History:** Provides a log of all modifications made to the driver and the documentation

### **Notes:**

- This document assumes that you have read the “Development Environment” information in the Studio *Technical Reference Manual*.
- This document also assumes that you are familiar with the Windows XP environment. If you are unfamiliar with Windows XP, we suggest using the **Help** feature (available from the Windows desktop **Start** menu) as you work through this guide.

## General Information

This chapter explains how to identify all the hardware and software components used to implement communication between the Studio YOMX driver and the MX100 Devices.

The information is organized into the following sections:

- Device Characteristics
- Link Characteristics
- Driver Characteristics
- Conformance Testing

### Device Characteristics

To establish communication, you must use devices with the following specifications:

- **Manufacturer:** YOKOGAWA
- **Compatible Equipment:**
  - MX100
  - Any device that is fully compatible with the MX100 device
- **Device Runtime Software:** MX100 API (including the DAQMX100.dll)

For a list of the devices used for conformance testing, see “Conformance Testing.”

### Link Characteristics

To establish communication, you must use links with the following specifications:

- **Device Communication Port:** Ethernet Port
- **Physical Protocol:** Ethernet
- **Logic Protocol:** Proprietary – MX100 API
- **Third Party Library:** YOKOGAWA MX100 API (**DAQMX100.dll**)
- **Specific PC Board:** Any TCP/IP Adapter (Ethernet Board)

### Driver Characteristics

The YOMX driver is composed of the following files:

- **YOMX.INI:** Internal driver file. *You must not modify this file.*
- **YOMX.MSG:** Internal driver file containing error messages for each error code. *You must not modify this file.*
- **YOMX.PDF:** Document providing detailed information about the YOMX driver
- **YOMX.DLL:** Compiled driver

#### **Notes:**

- All of the preceding files are installed in the `/DRV` subdirectory of the Studio installation directory.

You can use the YOMX driver on the following operating systems:

- Windows XP

For a list of the operating systems used for conformance testing, see “Conformance Testing” on page 4.

**Notes:**

- The 3<sup>rd</sup> part library includes the following files: DAQMX100.dll, DAQMX.dll and DAQHandler.dll. All these files are provided by Yokogawa in the MXAPI installation. These files should be either on the \Windows\System32 folder or in the Studio\BIN folder
- If you need to use more than one instance of this driver (Duplicating the YOMX.ddl file), you also need to create copies of the DAQMX100.dll file as well, and properly configure these information in the driver's Communication Settings. Please check the *Configuring the Driver* section for more details

The YOMX driver supports the following Data Types:

Data Types	Length	Write	Read
PV	4 Bytes	–	•

### **Conformance Testing**

The following hardware/software was used for conformance testing:

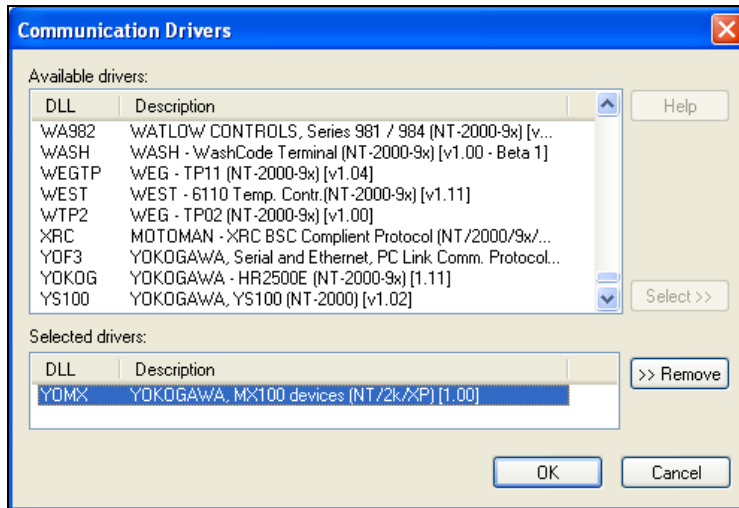
Driver Version	Studio Version	Operating System	Equipment
1.00	6.1	WinXP+SP2	YOKOGAWA MX100

## Installing the Driver

When you install Studio version 5.1 or higher, all of the communication drivers are installed automatically. You must select the driver that is appropriate for the application you are using.

Perform the following steps to select the driver from within the application:

1. Open Studio from the **Start** menu.
2. From the Studio main menu bar, select **File** → **Open Project** to open your application.
3. Select **Insert** → **Driver** from the main menu bar to open the *Communication Drivers* dialog.
4. Select the **YOMX** driver from the *Available Drivers* list, and then click the **Select** button:



*Communication Drivers Dialog*

5. When the **YOMX** driver displays in the **Selected Drivers** list, click the **OK** button to close the dialog.

 **Note:**

It is not necessary to install any other software on your computer to enable communication between the host and the device.

 **Attention:**

For safety reasons, you must use special precautions when installing the physical hardware. Consult the hardware manufacturer's documentation for specific instructions in this area.

## Configuring the Driver

After opening Studio and selecting the YOMX driver, you must configure the driver. Configuring the YOMX driver is done in two parts:

- Specifying communication parameters
- Defining tags and controls in the *STANDARD DRIVER SHEETS* (or Communication tables)

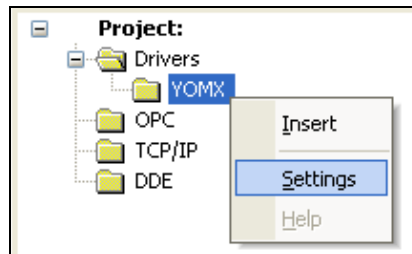
Worksheets are divided into two sections, a *Header* and a *Body*. The fields contained in these two sections are standard for all communications drivers — except the **Station**, **Header**, and **Address** fields, which are driver-specific. This document explains how to configure the **Station**, **Header**, and **Address** fields only.

**Note:**  
For a detailed description of the Studio *STANDARD DRIVER SHEETS*, and information about configuring the standard fields, review the product's *Technical Reference Manual*.

### Setting the Communication Parameters

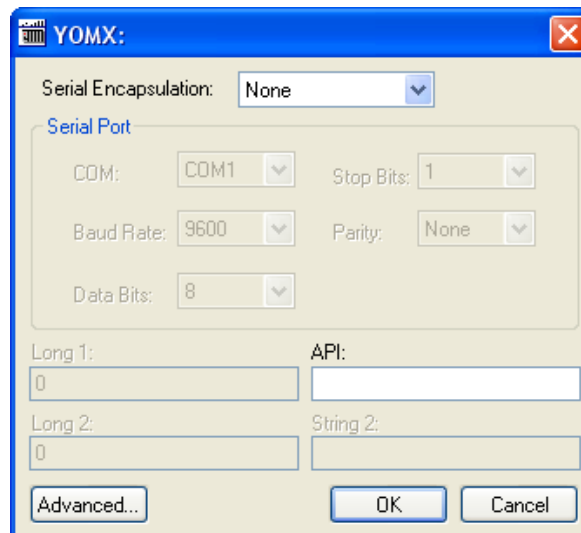
Use the following steps to configure the communication parameters, which are valid for all driver worksheets configured in the system:

1. From the Studio development environment, select the **Comm** tab located below the *Workspace* pane.
2. Click on the *Drivers* folder in the *Workspace* pane to expand the folder.
3. Right-click on the *YOMX* subfolder. When the pop-up menu displays, select the **Settings** option:



Select Settings from the Pop-Up Menu


The *YOMX: Communication Parameters* dialog displays:




Communication Parameters Dialog

4. Specify the custom parameters as noted in the following table:

Parameters	Default Values	Valid Values	Description
Station	0	0	Not used for this driver
API	Blank	Any	The filename of the DLL that is to be loaded by the driver to perform its external calls. If left blank, the default DAQMX100 is used. This parameter is particularly useful when you want to use more than one instance of the driver, duplicating the YOMX.dll file. Then, you need also to duplicate the DAQMX100.dll and configure this API field for each duplication of the driver with one name of the DAQMX100 copy Place the copies of the DAQMX100.dll in the same folder where the original one is (Windows\System32 or Studio\BIN folders)

 **Note:**  
 The device must be configured with *exactly the same* parameters that you configured in the *YOMX Communication Parameters* dialog.

5. Click the **Advanced** button on the *Communication Parameters* dialog to open the *Advanced Settings* dialog and configure the settings as necessary.

 **Notes:**

- Do not change any of the other *Advanced* parameters at this time. You can consult the *Studio Technical Reference Manual* for information about configuring these parameters for future reference.
- Generally, you must change the *Advanced* parameter settings if you are using a DCE (Data Communication Equipment) converter (232/485 for example), modem, and so forth between the PC, the driver, and the host. You must be familiar with the DCE specifications before adjusting these configuration parameters.

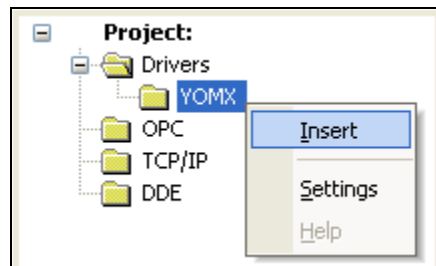
## **Configuring the Driver Worksheets**

This section explains how to configure the *STANDARD DRIVER SHEET*s (or Communication tables) to associate application tags with the device addresses. You can configure multiple *Driver* worksheets — each of which is divided into a *Header* section and a *Body* section.

### **Configuring the STANDARD DRIVER SHEET**

Use the following steps to create a new *STANDARD DRIVER SHEET*:

1. From the Studio development environment, select the **Comm** tab, located below the *Workspace* pane.
2. In the *Workspace* pane, expand the *Drivers* folder, and right-click the *YOMX* subfolder.
3. When the pop-up menu displays, select the **Insert** option:



***Inserting a New Worksheet***

 **Note:**

To optimize communication and ensure better system performance, you must tie the tags in different *Driver* worksheets to the events that trigger communication between each tag group and the period in which each tag group must be read or written. Also, we recommend configuring the communication addresses in sequential blocks to improve performance.



The *STANDARD DRIVER SHEET* displays (similar to the following figure):

The screenshot shows a configuration window for a driver. At the top, there is a 'Description' field containing 'Channels' and an unchecked 'Increase priority' checkbox. Below this are four columns of fields: 'Read Trigger' (RdTg), 'Enable Read when Idle', 'Read Completed', and 'Read Status'. A similar set of fields exists for 'Write Trigger', 'Enable Write on Tag Change', 'Write Completed', and 'Write Status'. The 'Station' field contains '192.168.1.101' and the 'Header' field contains 'PV:0'. To the right of the header field are 'Min:' and 'Max:' fields, both currently empty. At the bottom of the window is a table with the following data:

	Tag Name	Address	Div	Add
1	Channel[1]	1		
2	Channel[2]	2		
3	Channel[3]	3		
4	Channel[4]	4		
5	Channel[22]	22		

**STANDARD DRIVER SHEET**

In general, all parameters on the *Driver* worksheet (except the **Station**, **Header**, and **Address** fields) are standard for all communication drivers, but they will not be discussed in this document. For detailed information about configuring the standard parameters, consult the *Studio Technical Reference Manual*.

4. Use the following information to complete the **Station**, **Header**, and **Address** fields on this worksheet.

- **Station** field: Device IP
- **Header** field: Use the information in the following table to define a reference to the initial channel that will be read from the device. The default value is **PV:0**.

These variables must comply with the following syntax:

**PV:**<*ChannelReference*> (for example, **PV:11**)

Where:

<*ChannelReference*> is the initial Channel (reference).

After you edit the **Header** field, Studio checks the syntax to determine its validity. If the syntax is incorrect, Studio automatically inserts the default value in the **Header** field.

Also, you can type a tag string in brackets {**Tag**} into the **Header** field, but you must be certain that the tag's value is correct and that you are using the correct syntax; otherwise, you will get an **invalid Header** error.

The following table lists all of the data types and address ranges that are valid for the YOMX driver.

Header Field Information			
Data Types	Sample Syntax	Valid Range of Initial Addresses per Worksheet	Comments
PV	PV:1	Varies according to the equipment	Channel of device to be read

- **Address** field: Use this field to associate each tag to its respective device address. Type the tag from your application database into the **Tag Name** column. This tag will receive values from or send values to an address on the device. The address must comply with the following syntax:

**<Channel>**

Where:

**<Channel>** is the Channel Address to read on the MX100 devices.

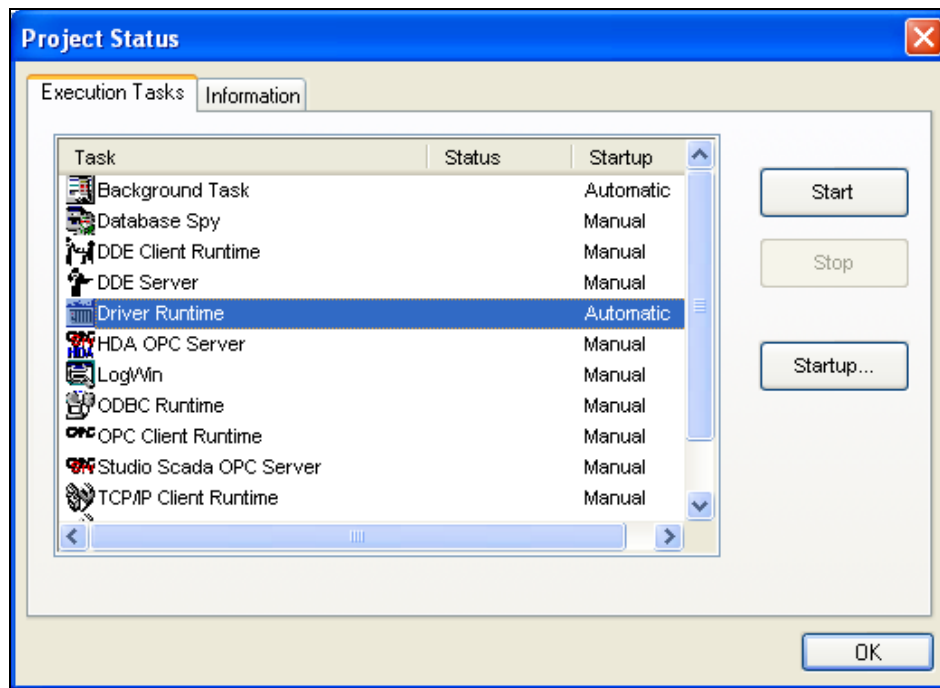
## Executing the Driver

After adding the YOMX driver to a project, Studio sets the project to execute the driver automatically when you start the run-time environment.

To verify that the driver run-time task is enabled and will start correctly, perform the following steps:

1. Select **Project** → **Status** from the main menu bar.

The *Project Status* dialog displays:



*Project Status Dialog*

2. Verify that the *Driver Runtime* task is set to **Automatic**.
  - If the setting is correct, click **OK** to close the dialog.
  - If the **Driver Runtime** task is set to **Manual**, select the **Driver Runtime** line. When the **Startup** button becomes active, click the button to toggle the *Startup* mode to **Automatic**.
3. Click **OK** to close the *Project Status* dialog.
4. Start the application to run the driver.

## Troubleshooting

If the YOMX driver fails to communicate with the device, the tag you configured for the **Read Status** or **Write Status** fields will receive an error code. Use this error code and the following table to identify what kind of failure occurred.

Error Code	Description	Possible Causes	Procedure to Solve
0	OK	Communication without problems	None required
1	Invalid Command	Write commands not supported	This driver only accepts Read operations.
2	Error Block Size	Too many channels configured in the same Driver Worksheet	Split the Driver Worksheet into two or more Worksheets.
3	API DLL not found	DAQMX100.dll not found	Verify that the DAQMX100.dll is installed in the <b>/BIN</b> subdirectory of the Studio installation directory.
4	Error Connect	<ul style="list-style-type: none"> <li>▪ Disconnected cables</li> <li>▪ PLC is turned off, in stop mode, or in error mode.</li> <li>▪ Wrong IP station number</li> </ul>	<ul style="list-style-type: none"> <li>▪ Check cable wiring.</li> <li>▪ Check the PLC state – it must be RUN.</li> <li>▪ Check the IP station number.</li> </ul>
5	Invalid Channel	<ul style="list-style-type: none"> <li>▪ Channel does not exist</li> <li>▪ Disconnected cables</li> <li>▪ PLC is turned off, in stop mode, or in error mode.</li> <li>▪ Wrong IP station number</li> </ul>	<ul style="list-style-type: none"> <li>▪ Verify that the Channel exists.</li> <li>▪ Check cable wiring.</li> <li>▪ Check the PLC state – it must be RUN.</li> <li>▪ Check the IP station number.</li> </ul>
-15	Timeout Start Message	<ul style="list-style-type: none"> <li>▪ Disconnected cables</li> <li>▪ PLC is turned off, in stop mode, or in error mode.</li> <li>▪ Wrong station number</li> <li>▪ Wrong RTS/CTS control settings</li> </ul>	<ul style="list-style-type: none"> <li>▪ Check cable wiring.</li> <li>▪ Check the PLC state – it must be RUN.</li> <li>▪ Check the station number.</li> <li>▪ Check the configuration. See the <i>Studio Technical Reference Manual</i> for information about valid RTS/CTS configurations.</li> </ul>

⇒ **Tip:**

You can verify communication status using the Studio development environment *Output* window (*LogWin* module). To establish an event log for **Field Read Commands**, **Field Write Commands** and **Serial Communication**, right-click in the *Output* window. When the pop-up menu displays, select the option to set the log events

If you are unable to establish communication with the PLC, try to establish communication between the PLC Programming Tool and the PLC. Quite frequently, communication is not possible because you have a hardware or cable problem, or a PLC configuration error. After successfully establishing communication between the device's Programming Tool and the PLC, you can retest the driver.

To test communication with Studio, we recommend using the sample application provided rather than your new application.

If you must contact us for technical support, please have the following information available:

- **Operating System** (type and version): To find this information, select **Tools** → **System Information**.
- **Studio Version**: To find this information, select **Help** → **About**.
- **Driver Version**: To find this information, read the full description of the driver on the *Communication Drivers* dialog.
- **Communication Log**: Displays in the Studio *Output* window (or *LogWin* window) when the driver is running. Be sure to enable the **Field Read Commands**, **Field Write Commands** and **Serial Communication** for the *LogWin* window.
- **Device Model and Boards**: Consult the hardware manufacturer's documentation for this information.

## Sample Application

This driver does not have a sample application

## Revision History

Doc. Revision	Driver Version	Author	Date	Description of changes
A	1.00	Eric Vigiani and Fabio H.Y. Komura	May/12/2005	Initial version
B	1.01	André Körbes	June/24/2010	Added an optional parameter on settings to indicate which library to load.
C	1.01	Andre Bastos	Aug/9/2011	Documentation changes only