

XL200 PLC Interface

Project Overview

Revised 4-07-05

Objective

To provide an industry-standard communications interface between an XL200 Series controller and a PLC. This interface would allow the XL200 to support several new features including:

- Mapping standard inputs & outputs to the PLC
- Mapping extended inputs & outputs to the PLC
- Mapping external gag outputs to the PLC
- Providing on-screen push-buttons (similar to a PLC HMI)
- Utilizing a PLC for Y-Axis positioning
- Publishing high-level data to the PLC
- Reporting E-Stop reasons and other PLC diagnostics messages
- Communicating Delay Codes and Scrap Codes between the XL200 and the PLC

Description

The XL200 Series controller will communicate with a PLC via a MODBUS network. MODBUS is a serial communications specification supported by many PLC manufactures including Allen-Bradley, GE-Fanuc, Automation Direct and others. The MODBUS spec includes two modes of communication: ASCII and Binary (RTU). The RTU mode was chosen for this project because it is much more commonly supported by various PLC devices. A typical MODBUS network will be configured in a master-slave arrangement with the XL200 acting as the MODBUS master and a PLC acting as a MODBUS slave.

Due to communication speed limitations (most PLCs do not support MODBUS baud rates above 38.4 kbps) the MODBUS interface is not well suited for activating time-critical I/O such as punch presses or boost cylinders. A user should have a thorough understanding of data transmission rates and PLC/XL200 scan times before using the MODBUS interface to activate inputs and/or outputs.

Many higher-end PLC devices provide a MODBUS/TCP interface which is basically the MODBUS serial protocol wrapped in Ethernet packets. The XL200 controller can communicate with MODBUS/TCP devices on an Ethernet network by using an external “gateway” to convert the serial data into Ethernet packets. Communication rates between

the XL200 and the “gateway” device can be much higher (115 kbps or higher). Communication rates between the “gateway” device and the PLC are comparable to typical Ethernet data communication rates.

Applications

The XL200 PLC Interface is intended for use (but not limited to) the following applications:

Standard Input/Output Mapping

The state of all of the XL200’s standard outputs will be transmitted to the PLC where they can be mapped to memory and monitored by the PLC’s program. The PLC can also instruct the XL200 when to activate its standard inputs via the MODBUS interface. This feature can be used to simplify and reduce the amount of wiring needed for a typical installation by eliminating several discrete I/O connections. *Note that any input or output values transmitted serially can **not** be activated in real-time and are subject to delays because of slow data transmission rates and scan times. This feature is intended only for inputs and outputs which are not time-critical.*

Extended Input/Output Mapping

Some applications require additional inputs and/or outputs that can not easily be provided by the XL200’s built in I/O structure. The XL200 PLC Interface could be used to define extra inputs and/or outputs and potentially eliminate custom controller software projects.

External Gag Output Mapping

The XL200 PLC Interface can be used to eliminate the need for the existing “gag board” hardware. The state of the external gag outputs will be transmitted to the PLC where they can be mapped to memory and monitored by the PLC’s program.

On-Screen Push-Buttons

The user will be able to create on-screen push buttons that can be accessed through the XL200’s keypad and display. These push-buttons can be defined to activate the XL200’s built-in inputs or the extended PLC inputs via the XL200 PLC Interface.

Y-Axis Motion Driver

The XL200 currently supports Y-Axis tool and machine setup positioning using communication interfaces with a SERCANS PC or an AMS MP325 controller. A new multi-axis driver utilizing the XL200 PLC Interface will allow the XL200 to manage one or more multi-axis devices by sending positioning commands and receiving status

information from the PLC. The PLC would then be responsible for interfacing directly with the servo drive or other positioning device.

High Level Data Publishing

The XL200 will be capable of publishing several pieces of information that may be useful to a PLC program such as Current Order Number, Current Part Length, Quantity Remaining, etc. The data will be transmitted through the XL200 PLC Interface at regular time intervals.

E-Stop Reasons and Other PLC Diagnostics Information

It is typical for a machine's emergency stop circuit to be wired through the PLC and for some PLCs to display status information regarding the reason for the e-stop condition. This and other status data can be reported to the XL200 controller through the MODBUS interface to provide a more-integrated, easier to use, centralized user interface.

Delay and Scrap Code Publishing

The PLC may be able to categorize certain machine downtime conditions and transmit this information to the XL200 controller. The XL200 may be able to use this data to simplify the operator's data entry when entering a delay or scrap reason.

Implementation

Hardware

The communication protocol specifies that a MODBUS serial system may use any of several different physical interfaces (RS232, RS485, RS422) with Two-Wire RS485 being the most common. The MODBUS implementation on the XL200 will utilize Communications Port D on the F connector and will require a two-wire RS485 cable. An approved, shielded, twisted-pair cable is recommended. The physical connections on the XL200 should be made as follows:

<u>Pin</u>	<u>Description</u>	<u>Details</u>
F11	Shield	Always connect the shield at only one end of the cable
F12	GND	Typically not connected
F13	RX+	Connect one signal wire from RS485 cable (B), also jumper this pin to F16
F14	RX-	Connect one signal wire from RS485 cable (A), also jumper this pin to F15
F15	TX-	Jumper this pin to F14
F16	TX+	Jumper this pin to F13

For wiring details on the PLC, refer to the manufacturer's documentation for the PLC being used. Applications with PLCs that do support an RS485 interface will require an external RS485 to RS232 converter. Applications with PLCs that support MODBUS/TCP will require an MODBUS Serial to MODBUS/TCP gateway device. Several commercial gateway devices are available. Refer to the manufacturer's documentation for wiring details.

XL200 Software Configuration

Model Option (Bit Code)

A new model option will be required to enable the XL200 PLC Interface. The new option will be designated in the controller's model name by the letter 'I'.

Setup Parameters

A new node will be visible in the Setup Menu tree whenever the MODBUS model option is enabled. The new node will be titled "PLC Communication" and will be located as a submenu under "Controller Settings" directly below the "Operator Preferences" node. ("Network Settings" will be renamed "Eclipse Settings.")

The following new setup parameters will be visible in the PLC Communication menu:

- **PLC Unit ID**
This parameter specifies the unit ID for the MODBUS slave device (PLC) that the XL200 communicates with. The valid range for unit IDs is from 1 to 247. The default value will be 0 which disables any and all MODBUS communication. Note: The PLC must be configured to use the same unit number that is specified by the XL200.
- **PLC Baud Rate**
This parameter specifies the baud rate used for communicating with the MODBUS slave device (PLC). The following list of baud rates can be selected from the drop-down list: 4800, 9600, 19200, 38400, 57600, 76800, 115200, 230400, and 460800. The default baud rate will be 19200. Note: The PLC must be configured to use the same baud rate as the XL200.
- **PLC Parity**
This parameter specifies the parity bit setting for communicating with the MODBUS slave device (PLC). The following list of parity settings can be selected from the drop-down list: None, Even, and Odd. According to the MODBUS RTU specification, when Even or Odd parity is selected, 1 stop bit will be used. When No parity is selected, 2 stop bits will be used. Note: The PLC must be configured to use the same parity as the XL200.
- **Configuration Register Address**

This parameter specifies the starting address for the PLC's holding register that contains the configuration data describing the structure of the data transfer between the XL200 and the PLC. The address must be entered in decimal and must match the address in the PLC where the configuration data structure begins. The configuration data structure will be described later in this document.

XL200 Controller Operation

In some applications using an AMS controller that is equipped with the PLC Interface feature, it may be critical that the PLC is properly installed and configured before the machine can operate correctly. We will do our best to ensure that machine safety is not sacrificed when using this feature by implementing some "safeguards" into the XL200 controller software.

Following a memory clear, if the PLC Interface has not been configured (i.e. the PLC Unit ID is equal to zero) and the operator tries to enter the run mode, the controller will prompt him with a pop-up window explaining that this controller has the PLC Interface feature but that no PLC is installed. If he chooses to accept this condition, he will not be warned again. If he does not accept, then the operator must enter a valid PLC Unit ID before he will be allowed to enter the run mode.

Once a valid PLC Unit ID has been entered, the controller will test that valid PLC configuration data has been read and that the PLC is currently online before allowing any change to the controller's run mode. This includes manual jog functions, manual punch and shear functions, as well as entering the run mode. If this test fails, the controller will display an appropriate error message.

Once the PLC has been installed and valid configuration data has been read, the controller will consider the PLC to be online. Any communication failure between the two devices will cause the controller to display a "PLC Communication Failure" error message and exit the run mode (if it was currently running). The PLC will then be considered offline until the operator goes to the Diagnostics screen, Printer Communication menu, and presses a function key to restore communications and bring the PLC back online.

PLC Configuration Data Structure

The following data structure must be provided by the PLC and located in holding registers at the address specified by the Configuration Register Address parameter in the XL200 controller. This data will be read by the XL200 during its power up sequence and will determine which pieces of data will be transferred between the XL200 and the PLC on a regular basis.

```
Struct
{
    CHAR cAMSIdentifier[4];
    WORD wVersionLevel;
```

```

WORD wConfigurationBits00;
WORD wXLStandardOutputsAddress;
WORD wXLStandardInputsAddress;
WORD wStartOfCoilsAddress;
WORD wPLCFunctionCoilsIndex;
WORD wPLCErrorMsgAddress;
WORD wPLCDiagnosticsMsgAddress;
WORD wUnusedAddress06;
WORD wUnusedAddress07;
WORD wXLExternalGagOutputsAddress;
}

```

The first member of the structure `cAMSIdentifier[4]` is an array of four characters and must contain the following four bytes of data in this order (all values in hex): 0x03, 0x41, 0x4D, and 0x53. These are the ASCII values for the text string “AMS” preceded by the number of characters in the string.

The next member of the structure `wVersionLevel` determines the remaining structure of the configuration data and the structures for all other user data. Once the XL200 has read this version level, it knows the structure of all the remaining user data which can be transferred back and forth to the PLC. Having a version level allows us to expand the configuration data structure and the other user data structures in the future so that as new features and capabilities are added, the version level will be increased. Therefore, newer releases of XL200 software will always be backwards compatible with older PLC programs because the XL200 can identify the version level in the PLC and use the older data structures. The data structure for the initial version level and subsequent version levels will be described later in this document.

The next member of the configuration structure `wConfigurationBits00` represents a configuration register where each individual bit enables or disables a particular structure of data to be transferred between the XL200 and the PLC. For example, one bit will be used to enable the transfer of the XL200’s standard 24 output values to the PLC. When this bit is set, the current values for each of the XL200’s 24 outputs will be transferred to the PLC during each “scan” of the communication loop. If this bit is not set, this particular data transfer will be disabled. Using individual configuration bits allows the PLC user to select only the data he wants to transfer during each scan, thus keeping the communication scan time to a minimum. The definition for all configuration bits is given in the section below.

The next few words in the configuration structure define the starting addresses in the PLC’s memory where the transferred data will be stored. The address for each particular structure of data must be specified separately. This gives the PLC user the flexibility of locating specific pieces of data at various places in the PLC’s memory. He can, for example, specify that the XL200’s standard outputs be mapped directly to coil memory, while PLC diagnostics messages are stored in holding register memory. *IMPORTANT*

NOTE: The user is responsible for ensuring that none of the data structures overlap one another, and that they do not conflict with any other memory locations being used by the PLC logic program. The definitions and sizes for all data structures are given in the section below.

WStartOfCoilsAddress is the address of where coils begin in the PLC's memory map.

PLC Configuration Bits and User Data Structures

Version Level 100

All configuration bits for wConfiguration0Bits are defined below:

```
#define CONFIG_0_XL_STANDARD_OUTPUTS          0x0001
#define CONFIG_0_XL_STANDARD_INPUTS          0x0002
#define CONFIG_0_STROBE_WATCHDOG_COIL        0x0004
#define CONFIG_0_PLC_ERROR_MSG                0x0008
#define CONFIG_0_PLC_DIAGNOSTICS_MSG         0x0010
// reserved for future use                    0x0020
// reserved for future use                    0x0040
// Reserved for future use                    0x0080

// The most significant byte of this word defines the number of external gag words used
#define CONFIG_0_EXTERNAL_GAG_1_16           0x01XX
#define CONFIG_0_EXTERNAL_GAG_1_32           0x02XX
#define CONFIG_0_EXTERNAL_GAG_1_48           0x03XX
#define CONFIG_0_EXTERNAL_GAG_1_64           0x04XX
```

Map XL200 Standard Outputs to PLC

Configuration Bit: CONFIG_0_XL_STANDARD_OUTPUTS (0x0001)
Starting address in PLC memory specified by: wXLStandardOutputsAddress
User Data Size: 2 words

Enabling this feature causes the XL200 to transfer the current state of all 24 standard outputs to the PLC during each communication scan. The data will be transferred as two data words with one bit representing the state of each output (logic 1 = ON, logic 0 = OFF). Output 1 corresponds to the least significant bit in the structure and only the lower 24 bits will be used. For example, a value of 0x0091 indicates that outputs 1, 5, and 8 are ON and all other outputs are OFF. *IMPORTANT NOTE: To help keep scan times to a minimum, the output values will only be transferred when one or more of the outputs has changed state since the previous scan.*

Map XL200 Standard Inputs to PLC

Configuration Bit: CONFIG_0_XL_STANDARD_INPUTS (0x0002)
Starting address in PLC memory specified by: wXLStandardInputsAddress
User Data Size: 2 words

Enabling this feature causes the XL200 to read a value from the PLC's memory during each communication scan. This value will be logically OR'ed with the XL200 standard inputs to effectively allow the PLC to activate a controller input without being hard wired to the terminal strip. The data will be transferred as two data words with one bit representing the state of each input (logic 1 = ON, logic 0 = OFF). Input 1 corresponds to the least significant bit in the structure and only the lower 24 bits will be used. For example, a value of 0x0008 indicates that the PLC wishes to activate input 4 on the XL200. *IMPORTANT NOTE: Input values are read during each communication scan when this feature is enabled. Each input will be active on the XL200 controller for the duration of time that the corresponding bit is set in the PLC's memory.*

PLC Function Request Coils Index

Configuration Bit:	See Below
Coil address in PLC memory specified by:	wPLCFunctionCoilsIndex
User Data Size:	3 Coils

wPLCFunctionCoilsIndex is the index of the first of three PLC function request coils. The XL will read or write to these bits to determine if the PLC is requesting a function, to indicate a function is complete or to request the PLC to perform a function. Grouping all of these bits together allows the XL to minimize the scan time of reading and writing Controller Standard IO.

The function bits currently defined are:

- CONFIG_0_STROBE_WATCHDOG_COIL
- CONFIG_0_PLC_ERROR_MSG
- CONFIG_0_PLC_DIAGNOSTICS_MSG

wStartOfCoilsAddress specifies where coils begin in the PLC's memory map. The three function coils are sequential in memory with the watch dog coil being first, the error message coil being second and the diagnostics message coil being third. If the PLC's program desires the watch dog coil to be the second coil (C1) in memory then the wPLCFunctionCoilsIndex should be set to 1 since the index of the first coil in memory has an index of zero(C0). Following this example, assuming both the PLC Error and Diagnostic are also enabled, C2 would be the coil for the PLC Error Msg indication and C3 would be the coil for PLC Diagnostic indication.

Strobe Watchdog Coil

Configuration Bit:	CONFIG_0_STROBE_WATCHDOG_COIL (0x0004)
Coil address in PLC memory specified by:	wPLCFunctionCoilsIndex
User Data Size:	1 Coil

Enabling this feature causes the XL200 to turn on the specified coil at least once every 5 seconds. This safety feature is intended to be used by the PLC to monitor whether or not the XL200 controller is actively communicating with it. If the PLC detects that the specified coil has not turned on for more than 5 seconds, it can assume that it has lost

communication with the XL200 and can take the appropriate action (i.e. turn off critical outputs, etc.). The PLC program is responsible for turning this coil off each time it detects that the coil has been turned on.

Report PLC Error Message

Configuration Bit: CONFIG_0_PLC_ERROR_MSG (0x0010)
Starting address in PLC memory specified by: wPLCErrorMsgAddress
User Data Size: 41 words

Enabling this feature allows the XL200 to display an error message reported by the PLC that indicates a machine stoppage condition. Approximately once every second, the XL200 checks to see if the PLC has an error message to display. If so, the XL200 exits the run mode, reads the error message from the PLC and displays the text as an error message on the XL200 display. The message text may be up to 80 characters long. The first word of this user data structure should contain the number of characters in the text message. The remaining 40 words should contain the ASCII values for the message (two characters per word). The XL200 reads the PLC Function coils to determine if there is a text message from the PLC. If a message is detected it reads the message and then immediately turns off the PLC Error Message function coil to inform the PLC that the message has been read. When creating a new message to display, the PLC user should ensure that the complete message text is filled in before turning on the PLC function coil to inform the XL of the message.

Report PLC Diagnostics Message

Configuration Bit: CONFIG_0_PLC_DIAGNOSTICS_MSG (0x0020)
Starting address in PLC memory specified by: wPLCDiagnosticsMsgAddress
User Data Size: 41 words

Enabling this feature allows the XL200 to display a diagnostics message reported by the PLC but does not necessarily indicate a machine stoppage condition. This feature works exactly the same as the PLC Error Message feature but does not cause the XL200 controller to exit the run mode when the message is displayed.

Map XL200 External Gag Outputs to PLC

Configuration Bits: CONFIG_0_EXTERNAL_GAG_1_16 (0x01XX)
CONFIG_0_EXTERNAL_GAG_1_32 (0x02XX)
CONFIG_0_EXTERNAL_GAG_1_48 (0x03XX)
CONFIG_0_EXTERNAL_GAG_1_64 (0x04XX)
Starting address in PLC memory specified by: wXLExternalGagOutputsAddress
User Data Size: 1-4 words

Enabling this feature causes the XL200 to transfer the current state of all external gag outputs to the PLC. This feature is only available on AMS controller models capable of controlling 12 presses and when no traditional external gag I/O board is present. The PLC can enable 16, 32, 48, or 64 external gag outputs to be transferred from the XL200, requiring 1, 2, 3, or 4 words of data to be transferred. The gag output words will be

transferred from the XL200 only when one or more of the outputs changes states. Each bit in the data represents one external gag output (logic 1 = ON, logic 0 = OFF). For example, a value of 0x00A5 in the first word indicates that external gag outputs 1, 3, 6, and 8 are ON and all other outputs are OFF.

MODBUS Parity and Stop Bit Settings

The MODBUS spec states that when communications is set for NO parity that 2 stop bits should be used. When EVEN or ODD parity are used that 1 stop bit should be used. We complied with the spec internally and did not allow the user to set the stop bits. It appears that none of the PLC manufacturers comply with this part of the spec and allow the user to select 1 or 2 stop bits no matter what parity is used. This has caused more than enough confusion so we are now providing the PLC Stop Bit setup parameter to allow the user to make sure all of the serial settings match between the PLC and the XL.

MODBUS High Level Data Fields

This SCN applies to v101 in the XL200 PLC Interface Project Overview document.

For the scope of this change notice the "current item" is by definition the item that will begin running if the controller enters the run mode.

For the scope of this change notice the "next item" is by definition the item that will run after the current item is completed. It should not be confused with the item with a status of next.

These are the changes:

Added the ability to send the Order Number, Material Code, PCode, Order User Fields, Item User fields, Bundle, Quantity, Quantity Remaining, Part Length and Pattern Number for the current Order and Item.

Added the ability to send Bundle, Quantity, Quantity Remaining, Part Length and Pattern Number for the next item.

In most cases there should be a current item. If there is no current item the fields associated with the current item will be set to zero or empty strings.

If there is no next item the fields associated with the next item will be set to zero

On shear only lines the queue is not maintained. When the line exits the run mode there will not be a next item so the associated fields will be set to zero

Added functions that allow the PLC to prepopulate the Scrap and Delay code fields when scrap is generated or delays are encountered.

Multi-Axis Jogging through MODBUS

In order to complete the Spirco project the following edits had to be made. Spirco has to use one of the Axis as a material guide while threading a coil. They are also required to be able to jog this axis using a pendant. The only method of jogging a SERCOS PC Axis was by using the Diagnostics screen.

This edit modifies the MODBUS Spec and code to allow a PLC to request that an axis be jogged forward or reverse. The MODBUS task will pass this request along to the specific Axis driver and the driver will jog the axis. The only driver that currently supports jogging is the SERCOS PC driver. Other drivers may be able to utilize this feature in the future.

PLC event, to Trigger PLC Messages, Scrap and Delay codes on XL

The MODBUS "I" option already allowed the PLC to display an error or diagnostic message on the XL Controller. The string to display was stored in the PLC. Dietrich did not want to store the strings in the PLC and requested that we provide the ability to store the string in the XL.

Similarly the "I" option allowed the PLC program to pre-populate Scrap and Delay codes in the XL. Having the PLC program use scrap and delay codes directly can present difficulty if the Scrap and Delay codes need to change.

A new feature of the "I" option has been added. The PLC will now report an event ID that the XL will look up in its PLC Message database. Each record in the database will have an enable field for Message, Scrap code and Delay Code. It will also have a data field for Message, Scrap code and Delay Code. If the data field is enabled by its corresponding enable field the controller will use the data to display a message or pre-populate a scrap code or delay code. The message enable field also determines if the message gets displayed as a diagnostics message or an error message.

The PLC Message database is not editable in the XL. The database can be viewed for troubleshooting purposes in the Diagnostics screen. Eclipse is responsible for editing and maintaining the PLC message database.

Details of how to enable this option are detailed in the AMS PLC MODBUS Spec.

Originally this change was implemented as a change to the error and diagnostics messages only. Before I could get it all checked in I found out about additional requirements for delay codes. We added scrap codes as an extension of the concept.

New Multi-Axis Driver for MODBUS

A new Multi-Axis Driver Type has been created to support Axes' controlled by a PLC using the MODBUS interface.

The Eclipse UART Spec has changed to support the new Driver Type. The UART Spec has been updated to reflect the changes.

Details about the interface between the controller and the PLC are outlined in the XL200 PLC Interface spec.

A new sub-menu of the Multi-Axis Devices menu in the Diagnostics screen has been added. It is entitled PLC. It will have a sub-menu for each Axis defined as a PLC Axis. This sub-menu will display a status window that shows the position and status of Axis. There will be and F2-Send Command function key. The function key displays a window that allows the user to Reference, Disable, Enable, Jog and position the axis. Positioning the Axis is always enabled as a command. Jogging, referencing, enabling and disabling are all functions that the PLC program enables or disables depending on what the PLC program supports. In addition the PLC supports two different styles of referencing. It can reference to a position if it has an absolute encoder or it can reference to a switch.