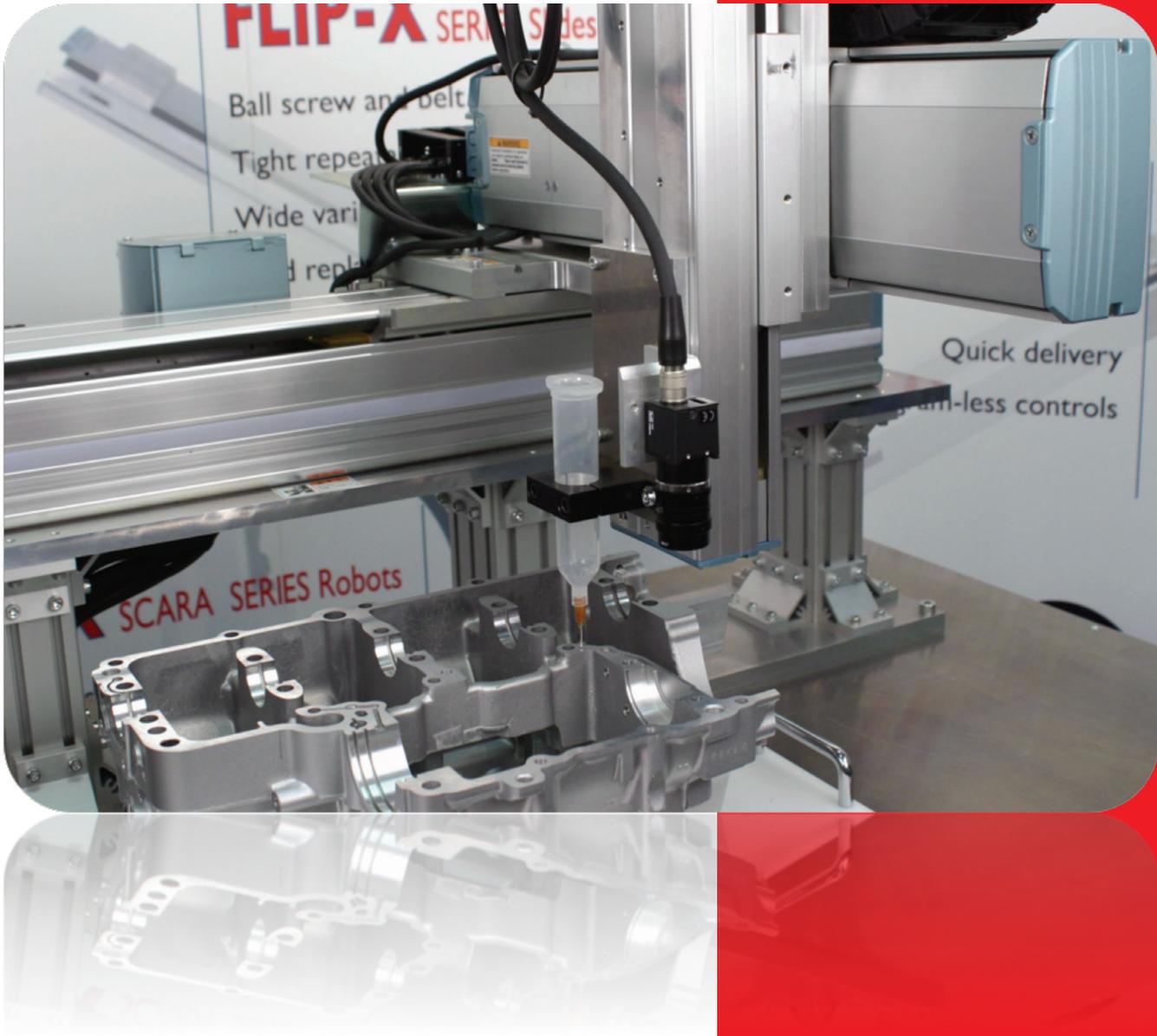


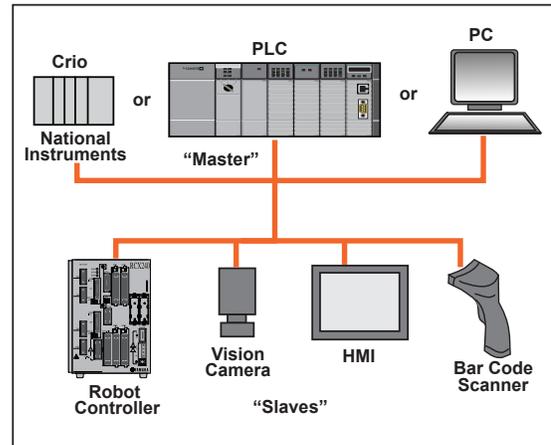
# Innovating Small Assembly Robot Integration Concepts within the PLC



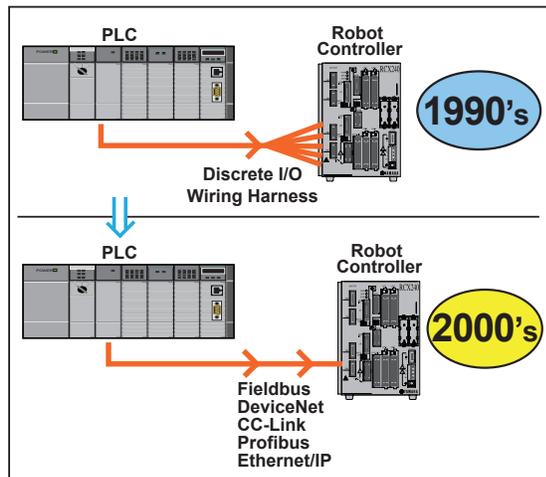
# Innovating Small Assembly Robot Integration Concepts within the PLC

## Introduction

For many years, Programmable Logic Controllers (PLCs) have remained the industrial controller of choice among control, automation, and design engineers. PLCs are often referred to as the “master” controller that sequences the main logic and manages network data on automated equipment. They are considered the hidden “workforce” that silently executes ladder logic instructions behind a closed electrical cabinet door. PLCs, however, are generally not installed or integrated alone on automated machines. Third party products are often used to complement a PLC, such as servomotors, vision systems, testing equipment, or robotics. These complementary devices are referred to as the “slave” components that work in harmony with the automated systems and are controlled by the PLC in a master/slave relationship.



Crios, PCs, or PLCs are considered the master controllers that must communicate to many different types of “slave” devices. Getting the slaves to talk to the master is the challenge of integration.



Over the years, integration styles have changed from discrete wiring to fieldbus hardware. Fieldbus has allowed for a streamlined connection between two different devices.

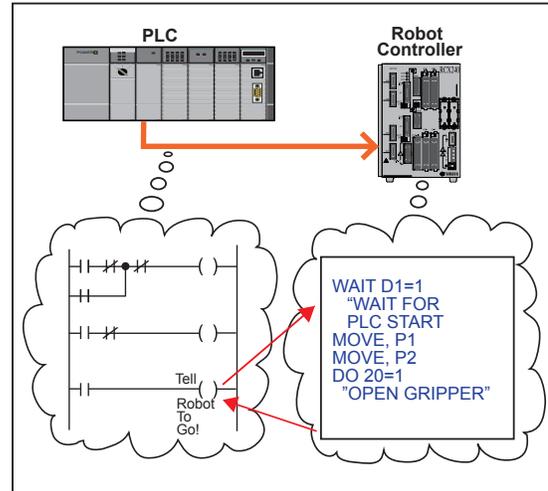
concept of data exchange happens in a manner where each device is treated individually or managed separately using each device’s configuration or programming software. PLCs have a programming software suite that must be learned by an engineer, along with other third party software to program and configure a slave device that will connect to the master PLC. The “old school” method of integrating the master and slave devices has been the de facto standard in industrial automation. This entails an engineer writing code in the third party device and writing ladder logic in the PLC manufacturing software. The software code then has to interact with each other, often referred to as “software handshaking.” Writing custom handshaking software between multiple devices has been cumbersome over the years. When an engineer wants to integrate another third party device, the whole process of writing custom code must be thought through and debugged. Most engineers have this thought process engrained in their minds on how to sequence and interlock separate controllers, becoming somewhat of a standard mindset.

## The Pain Level of Integration for Engineers

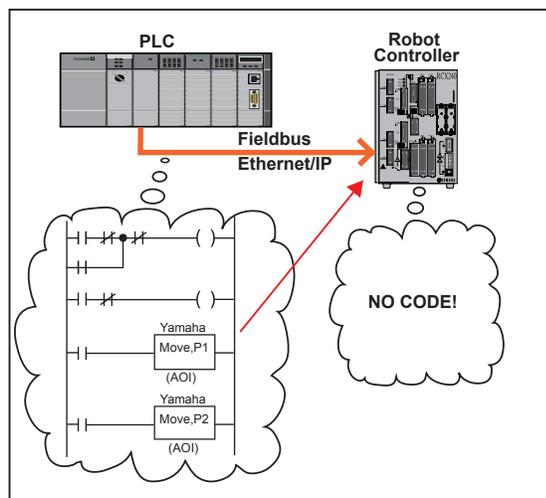
The problem with this old method of integration is that it is a point of pain for engineers as they must continually learn a new third party software package, or attend a vendor-training seminar to properly write software for the third party devices. In an industry that demands a great deal of time from a custom machine integrator, engineers simply do not have time to learn about a new third party device. This is a large deterrent for potential first-time users. They

are reluctant to try new vendors because of the learning curve involved in understanding how the slave device integrates and communicates with the existing PLC system that is designed into an automated machine as the main controller. This pain level creates additional costs in a project scope as engineers are forced to learn new software devices, directly impacting engineering time quoted on a project.

Over the last decade Fieldbus has advanced in sharing software memory mappings between the master PLC and third party slaves. Today, the logic concept still stands that the master PLC is written to sequence the main control on the automated machines as the third party slave devices have code written in a non-standard ladder logic format. This is, however, dependent on what kind of a device is handling the auxiliary process. In some cases, third party devices are written in structure text language sets that must be treated as a stand-alone process. These software sub-systems must be written to act independently of the main PLC architecture. Therefore, the pain level of learning comes into effect, as engineers must learn how to write code in these third party devices that are often not coded in the same way as the main PLC ladder logic standards. This exchange of information is generally a single bit level flag that is set in the main PLC from a ladder diagram rung, which instructs the slave devices to “Go” or “Start” their auxiliary process. At this time, the ladder sequence in the PLC must wait for a confirmation bit from the third party device indicating its process is complete. This is a low level of exchange of data, and extracted information is vague once its task is complete. As automation and technology have evolved over the years, the need to share detailed information about third party process controls has become more demanding in terms of preventing failures or logging statistical data about a part in production. Because of simple mapping concepts of the bit level exchanges, it can be difficult to extract the scientific data of actual results from third party processes to and from a PLC’s memory mapping exchange only.



This is an example showing software integration of two different styles of software development. The PLC on the left is written in ladder logic, while the robot controller on the right is written in structured text language.



By moving the sequence control into the PLC ladder logic, the robot software portion is eliminated.

### New Concepts of Robotic Control

How do you change the mindset of “This is how we always control a third party controller from a PLC?” For instance, with a robot controller, what if there was a way you could embed direct control of third party devices into PLC ladder logic? This type of control methodology would be a different concept or approach in a control scheme. By creating function blocks that communicate with the robot controller directly in ladder logic, it simplifies the integration handshaking between multiple devices such as a PLC and a third party controller. If you could simply make the robot move with a single ladder logic function block, this would solve all the issues of integration and eliminate the frustrations of learning another software package.

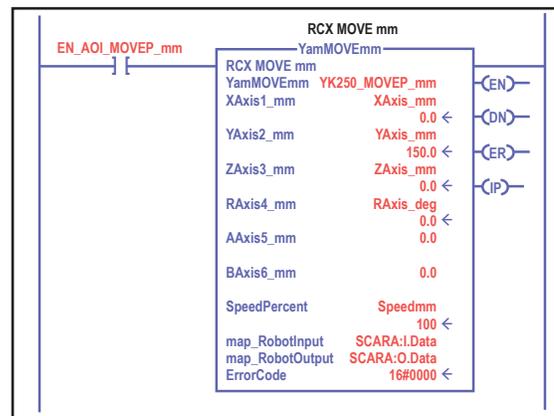
One of the major PLC manufacturers, Allen Bradley, with the introduction of tag-based PLCs, has provided the ability for vendors to create control function blocks called “Add-On Instructions,” or AOIs. AOIs allow third party vendors

to create control or function blocks that are imported into the PLC. However, not all AOIs are created equally. Some AOIs have been created that try to emulate the old school integration concept. These can make it easier to interact and integrate with a robot controller, but code still has to be written in the robot controller before the robot is instructed to move via the PLC. How can vendor-provided AOIs take the integration to the next level? This is done by changing the concept of that interaction to something that simplifies the control of a robot where the PLC commands the robot directly without the need to learn or write a single line of code in the robot controller. It would be a significant improvement to provide PLC programmers the ability to directly control their robot in an integrated manner within the PLC software and sequence direct robot motion from within a ladder logic instruction block. This is exactly the approach that Yamaha Robotics has taken with the RCX240 controller connected via Ethernet/IP to an Allen Bradley PLC.

## How the AOI Blocks Work

In the Yamaha RCX240 controller, special firmware accepts command codes from the PLC ladder logic AOI blocks. It commands a Yamaha robot to move in the same manner as if a robot programmer wrote Yamaha code using structured text scripting. Specially-designed AOIs can create and generate the command code automatically that the RCX240 controller understands and interprets. These commands are turned into instantaneous robot actions.

The AOI blocks are designed to send information that break down sequence control to an upper level code that behaves like an “API” application programming interface. AOIs receive programmer parameters that direct the robot to move, change parameter settings, or send information to the PLC when requested. For example, if a PLC ladder AOI block for MOVEmm was executed in ladder logic, this block receives direct Cartesian millimeter values in the form of X,Y, Z and R values along with a “speed” parameter. When the ladder logic block goes “true”, only these parameters are sent to the robot controller, which then processes this information to handle the motion command and moves the robot from where it is to the newly instructed position in 3D space. By controlling the sequencing in ladder logic, the PLC programmer can move the robot around in the work cell with AOI blocks.



This screenshot demonstrates a MOVEmm AOI block that is part of the ladder logic that can control an RCX240 Yamaha robot controller directly by passing in Cartesian millimeter parameters in the form of X,Y,Z,R dimensions in a 3D space. This will directly influence the robot to move the end-of-arm tooling from where the robot has traveled to the new destination position via its own trajectory calculations.

Typically, a fieldbus communication protocol is used to exchange information between the robot controller and the PLC. In the case of Allen Bradley, the fieldbus protocol of choice is Ethernet/IP. Other fieldbus options can be used such as Profibus with Siemens and GE PLCs, or CC-Link and Mitsubishi PLCs. The exchange of I/O memory mappings between the two devices is communicated via special block registers that the robot responds to when certain hex commands are received that are embedded within the Yamaha RCX240 firmware code. The robot controller listens to these registers and reacts instantly when certain codes are received in the proper sequence. This sequence is managed by the custom AOI blocks that the engineer imports into the PLC project.

For example, traditionally a robot or servomotor position is taught via a vendor software package using a method of jogging a robot and teaching its location to a point map memory area. These points are then recalled via a vendor software programming language to sequence a robot step-by-step in an automated work cell. With AOIs, an engineer can elect to teach points via the teach pendant and sequence the robot movement in PLC ladder logic

as one option. Another option is to create a user defined memory type (UDT) in a tag-based PLC that records and stores the cartesian dimensions of the robot or servo motors in the PLC memory area. With direct access to the point data in the PLC, the engineer can use any preexisting ladder logic to manipulate or command the robots to any point without having to store the robot's positional data in the robot controller. This provides the ability for special machine builders or engineers to design elaborate or control-savvy interfaces for their machines with the ability to manipulate data directly in the PLC for robot position and robot sequence control.

AOIs are developed natively in a PLC programming software environment. In the case of Allen Bradley, AOIs are created as add-on-instructions that appear as additional function blocks in RS Logix 5000 software. The engineer uses the built-in feature of the PLC programming environment to drag and drop instructions blocks from the toolbar directly into ladder logic as if the robot toolset was part of the PLC software suite. The concept of robot control via ladder logic can be used in any PLC software that provides the ability to create custom function blocks. Other examples include STEP 7 software for Siemens or Sysmac software for Omron NJ series PLC.

### Benefits

Benefits identified with new control philosophies:

- No additional training
- No additional robot software required to learn
- Development software cost savings
- Ease of integration between multiple third party devices

No additional training is required for maintenance or engineering staff. Instead of learning another vendor software package, the method of robot control is using ladder logic. A majority of maintenance staff and engineering personnel are usually already versed in PLC ladder logic. They are comfortable using a PLC or ladder logic in their own facilities. They feel confident using it in automated applications and competent enough to write or edit software using ladder logic over a structure text or robot programming environment.

Development cost for integrating a robot is reduced by as much as 20 to 40 percent. This number is an average of data collected by early adopters of programming the robot sequence all in ladder logic. Robots programmed in all ladder logic reduces software integration time as there is no additional learning curve or requirement to write traditional PLC ladder logic to interlock the robot sequence within the PLC sequence. Since the robot can be controlled in the PLC entirely, there is not any integration of a separate robot controller. The robot controller only needs to be supplied power and a common CAT5 cable that exchanges data via a fieldbus protocol such as Ethernet/IP.

### Expected Performance

Controlling a third party device over a fieldbus is limited via the fieldbus performance cyclic cycle and PLC scan times. With Ethernet/IP, typically an RPI (Request for Packet Interval) is 10 msec for each slave device attached to the master. If a PLC is commanding a motion control device with commands over fieldbus an expected performance can be derived from the following: (bus time + PLC scan time + slave device response) = total expected response.

MACHINE QUOTE		MACHINE QUOTE	
<i>Traditional</i>		<i>AOI's Used</i>	
	Hrs.		Hrs.
Elec. Eng.	60	Elec. Eng.	60
Assembly	80	Assembly	80
PLC Soft.	80	PLC Soft.	100
HMI Soft.	40	HMI Soft.	40
Robot Soft.	80	Robot Soft.	0
Debug Time	40	Debug Time	30
<b>Total</b>	<b>380</b>	<b>Total</b>	<b>310</b>
	x \$100 p/hr		x \$100 p/hr
	<b>\$38,000.00</b>		<b>\$31,000.00</b>

**\$7,000.00 Savings or 20%\***

\* Estimate from experience

A sample engineering and labor quote might look something like these two examples for a small assembly dial table machine with pick and place SCARA robots. Software labor is reduced by not needing to write code in the robot controller. Some time is added in the PLC software for robot sequencing. Debug time is reduced as well with only one software source code to debug.



These performance levels may vary depending on several variables; however, with Allen Bradley and Ethernet/IP, the command response between a standard Compactlogix PLC and a Yamaha RCX240 controller is about 20 to 30 msec. Controllers that are strictly designed for motion, such as robot controllers generally have a typical scan and execution time of 2 to 3 msec to execute a motion command from a written structure text program in comparison to the command of motion sent from a ladder logic PLC arriving in about 20 to 30 msec.

## Conclusion

Providing a new method to write sequence logic inside the PLC as ladder logic provides flexibility to the automation software engineers. Harnessing a powerful platform that controls factory automation and co-existing in the same development platform allows for ease of information exchanged between multiple controller platforms. Developing in the ladder logic software environment reduces training required for personnel to learn additional software languages. Most engineers and technicians that maintain automated equipment already have a basic understanding of sequencing automation software in a ladder logic environment. By reducing the amount of training required, this saves training costs to companies who embrace new control mythologies. Writing ladder logic sequence for motion control is not uncommon as Rockwell Automation has provided Kinetix Motion architecture that utilizes the same typical use motion function blocks to move servo motors on an automated machine. Bringing a SCARA robot or 360 degree zero blind spot reach robot into the mix of automation capabilities lends itself to easier integration options for smaller robotic actuators when designing a custom machine.

## About YRG Robotics

Yamaha Robotics has more than three decades of experience in manufacturing assembly robots to the industrial marketplace. Unsurpassed for payload, speed, and dependability, robotic solutions from Yamaha Robotics will help you save time and eliminate risk.

- Versatile, dependable, and proven robotic solutions have been used in Yamaha's own plants and thousands of others around the world.
- Diverse selection of single and multi-axis Cartesian robots for nearly any automated assembly requirement.
- Proven performance in sealing, dispensing, laboratory testing, bolt and nut fastening, conveyor tracking, palletizing, and material handling.
- Yamaha robots' extraordinary lifespan gives a return on investment unequalled by any other robot manufacturer.

All Yamaha products and services are designed to provide long-term savings needed to enhance your profitability.

For more information visit: <http://www.yrginc.com>

Email: [sales.support@yrginc.com](mailto:sales.support@yrginc.com)

Phone: 260-747-3482