PLCopen 4: Blurring the Lines between PLC, Robot, and Motion Control

Standard Combines Three Control Methods Into One Common Language

Jamie Solt, Senior Motion Product Engineer
Yaskawa America, Inc.

Overview

Traditionally, industrial robots have been programmed in complex proprietary languages that are difficult for anyone but robot programmers to understand. Motion controllers are wide and varied, and are usually programmed using a PC library or another proprietary language, while PLC's tend to be programmed in ladder logic. In today's automation environment PLC's, motion controllers, and robots are required to be tightly integrated. Many different elements are incorporated into the machine design with each requiring the programming strengths exhibited by their proprietary language. More and more end users are asking for robots, motion controllers, and PLC's to be programmed in familiar PLC languages. These languages are easier for machine builder programmers to understand, and for end users service personnel to maintain. To reduce the complexity and harmonize the look, feel, and function of these three separate platforms, the PLCopen working group for motion control has come up with a set of standardized tools to allow coordinated motion to be run directly from a PLC like programming environment.

PLC's

Since their inception in 1968 through a request by General Motors (to come up with a way to replace hardwired relays), PLC's have been programmed in ladder logic. They can easily control processes that require digital and analog devices, but more complex processes that are sequential in nature are more difficult than they would be in programming languages such as BASIC, C, or C#. Over the years PLC have evolved to include programming in BASIC or C, but a majority of them still rely on ladder logic. Many low end PLC's support motion control via step and direction outputs. Some higher level motion control can be achieved through expensive dedicated modules that must be added to the basic system. Even though most devices are programmed in ladder logic, most require an intimate knowledge of the programming environment which changes from manufacture to manufacture, and their higher level functions are usually achieved through specialized function blocks.

Motion controllers

Motion controllers for the general market typically include interpolated motion (linear and circular), coordinated motion, gearing, camming, and event triggered motion (where a sensor and position latch are used). Older controllers used dedicated inputs and outputs per axis. Motion inputs such as enable, over travel limits, and encoder inputs (one or two per axis) and motion outputs like servo command (normally +/- 10V analog) and/or stepper command (step and direction) were provided. Most controllers also have some general purpose I/O. New controllers rely on digital networks like EtherCat, or Yaskawa's Mechatrolink to pass control signals to the drives and receive and transmit the digital IO which is wired directly to the drive.

When dealing with the motion on linked axes, the typical motion controller cannot compete with robot controllers. With typical motion controllers, if you wanted to move the end effector to a specific point you had to figure out the correct positions for each of the axes. What you need for robots and other machines with mechanically linked mechanisms is inverse kinematics. The use of inverse kinematics requires formulas to translate the specific point in world space to the individual positions that each joint (or axis) needs to be at to move the mechanically connected mechanisms to the end point. Again, as these systems are wide and varied most require an intimate knowledge of their specific programming environment.

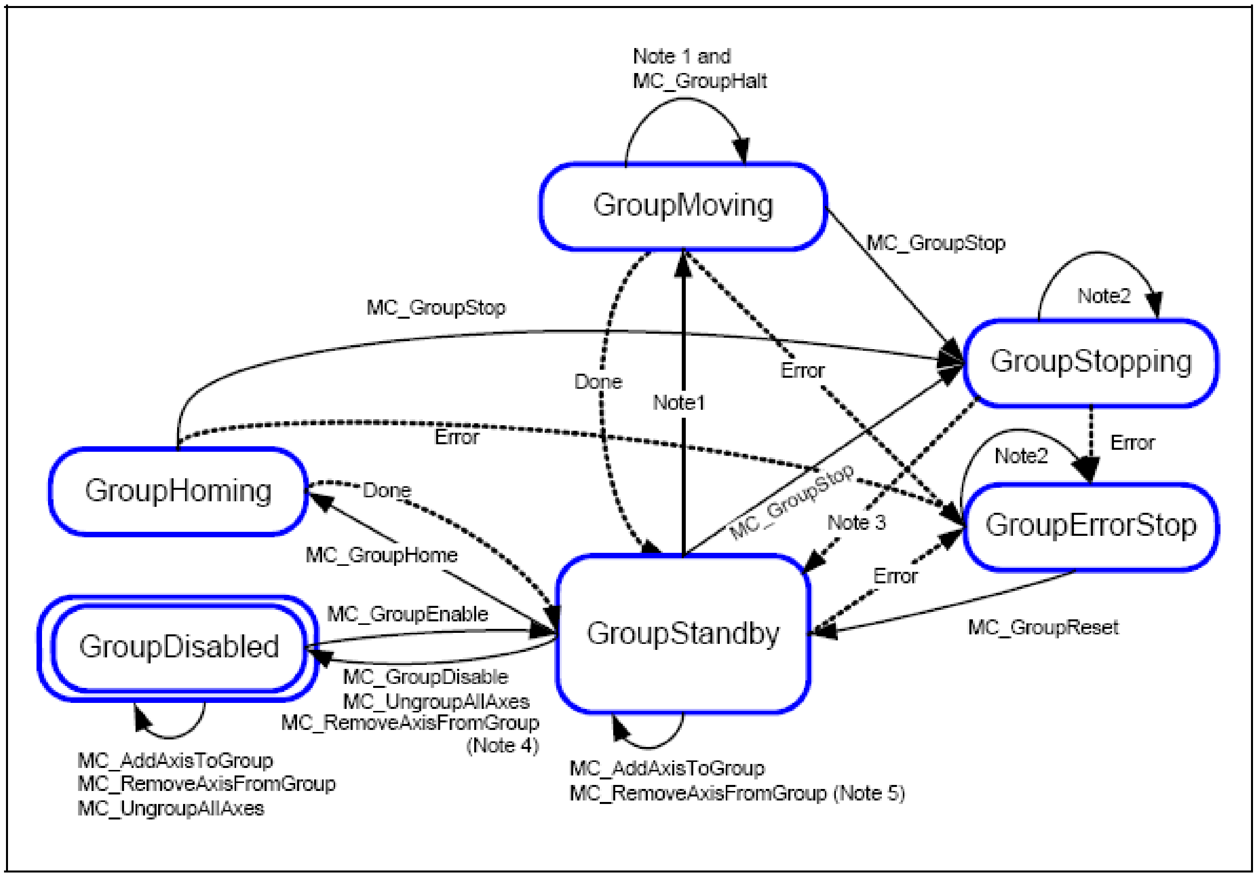
Robotic controllers

Robot controllers have been engineered to achieve the best control of specific complex mechanisms. Most controllers are manufactured for a specific device and are programmed in a specialized language created by the manufacturer that varies greatly from platform to platform. They are very efficient when controlling the library of devices for which they were designed; however, most are not the best in terms of communications, integration, or programming power. In the past, pretty much only the dedicated robot controllers supported kinematics and inverse kinematics. Now, it's a lot more common for many motion controllers to offers some subset of robot type commands, especially in controllers targeted for packaging automation. The lines are being blurred between robot controllers and motion controllers, but it is still up to the programmer to coordinate between these different systems with each programmed in a different language that is usually designed for their specific purpose.

Combining into one machine controller.

The PLCopen working group for motion control has standardized and logically defined all aspects of machine control programming. This is one of the best attempts of marrying PLC, robot, and motion control in one easy to understand language that is common among many different manufacturers. Many of the function blocks are basic, for example, a relative or absolute move are function blocks which are easily understood in any motion control system. The standardization and common look and feel of multiple control systems is really an advantage when the difficulty of the required motion increases. For example it is easy to string relative or absolute moves together when each individual move stops before the next move begins. But imagine a more complex set of movements where the axis is required to transition to the next move at some non zero velocity, blending the individual moves into one fluid motion throughout the entire path of the axis. PLCopen Motion Control defines standard blending operations to allow the programmer to achieve this fluid motion with common blending and transition modes that a manufacturer can implement.

One of the basic issues when moving multiple axes together with a mathematical model that controls mechanically linked axes, is that it is not always clear which axes are critical to move in synchronization. So when a fault occurs, the motion controller cannot always tell which other axes are affected. PLCopen addresses this by defining a motion group, so that the controller can generate a proper error response when one of the grouped axes has an error. This grouping concept allows the programmer freedom to concentrate on the specific task required of the machine and have the controller take care of the function of the group through implementation of the group state machine shown below.



The PLCopen motion standard includes part four which contains function blocks for coordinated motion. They define a standardized set of function blocks for the complex control of movement within 3D space that includes blocks for kinematic transformations. Typically, these transformations have to be supplied by the vendor, so for most manufactures, if the motion controller doesn't support it, it cannot be added. But Yaskawa has taken a different approach. There are the basic supported mechanisms like SCARA and delta, but in addition to these, any programmer is allowed to write his own kinematic transforms. Yaskawa provides specialized functions that are used to call these kinematic routines whenever a world position needs to be converted into joint space, or vice versa.

This standard is now creating a bridge between the once separate worlds of PLC's, CNC's, robotics and motion. It is now possible to program the complete control of the machine from one PLC like system. This standard has allowed robots, and motion controllers to become an integral part of a control system, rather than independent systems. They integrate motion control and logic control, the two primary requirements for modern machine control. There are definitive advantages to having both the motion control and logic control in the one package, including, but not limited to, practically unlimited exchange of data between the logic and motion engines, without the latency which can limit performance in traditional systems. In fact, it is now possible to perform perfect synchronization between a robot and additional servo axes using a machine controller, a feat which was previously only possible to achieve purely in the robot controller domain.

Conclusion

Ultimately, the goal of the PLCopen standard is to allow the program code to be completely independent of the hardware or specific manufacturer. When different hardware vendors support the same underlying code, and behave in the same manner, the programmer is free for learning proprietary languages associated with each manufacturer. This results in allowing complex complete machine control systems with improved accuracy and throughput to be developed in a shorter time to market. PLCopen has allowed this development by reducing engineering complexity and the specialized training required so that the overall system is more familiar to wide array of existing PLC programmers.