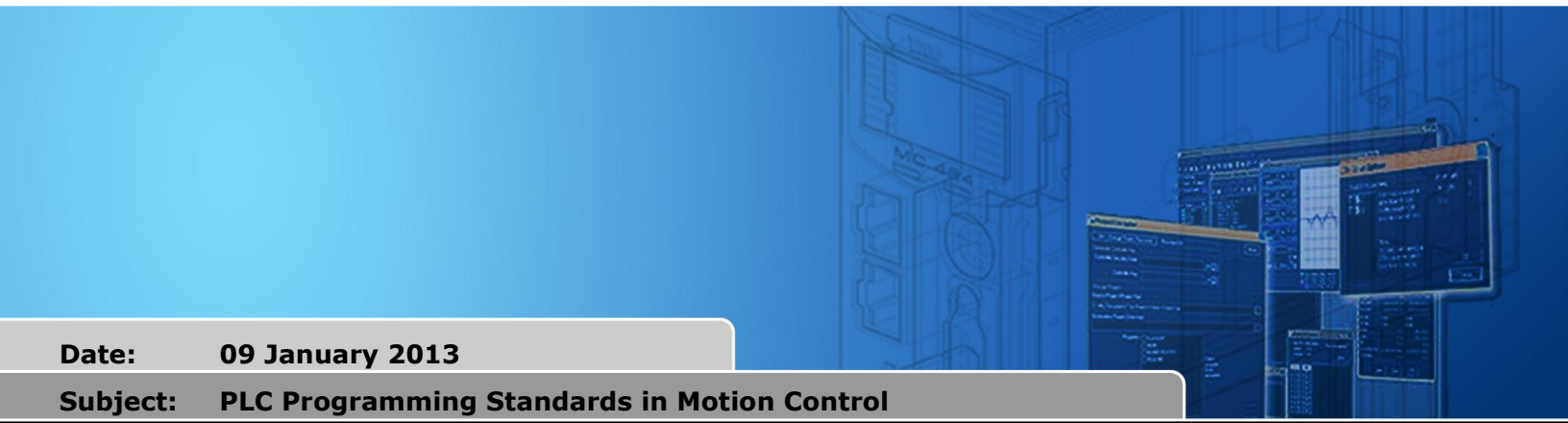
Trio Motion Technology, Ltd.
Shannon Way, Tewkesbury,
Gloucestershire. GL20 8ND
United Kingdom
Tel: +44 (0)1684.292333
Fax: +44 (0)1684.297929

187 Northpointe Blvd.
Suite 105
Freeport, PA 16229
United States of America
Tel: +1 724.540.5018
Fax: +1 724.540.5098

Tomson Centre
118 Zhang Yang Rd., B1701
Pudong New Area, Shanghai
Postal code: 200122
P.R. China
Tel/Fax: +86.21.58797659



Date: 09 January 2013

Subject: PLC Programming Standards in Motion Control

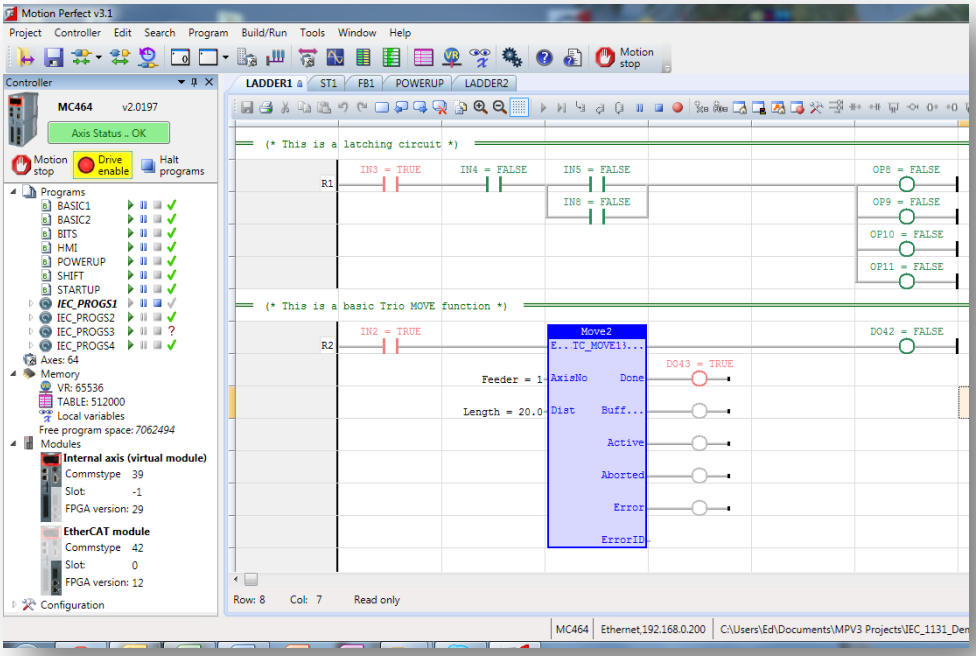
IEC 61131-3 Now in Motion

There is a natural tendency for industries to push toward a standard be it hardware or software, and programming motion controls is no exception. Dedicated motion controls have been around for over three decades and in the early years, nearly all had their own unique programming language. While generally not a problem, a more standardized language would make it easier to support and work between platforms in a factory environment. Time and technology moves on and the requirement for more standardization in programming motion have gained momentum. Enter European standard IEC 61131-3. With IEC 61131-3, programming of PLCs, distributed control systems, and motion controllers from different manufacturers is more manageable. The International Electrotechnical Commission (IEC) identifies five standard programming languages as the most common for both process and discrete programmable controllers: Ladder Diagram (LD), Function Block Diagram (FBD), Sequential Function Chart (SFC), Structured Text (ST), and Instruction List (IL). IEC 61131-3 is the third part of the open international standard IEC 61131 for programmable logic controllers, first published in 1993 by the IEC then later revised in 2003. Part 3 specifically deals with the five programming languages. The European standard has become more popular in the US in recent years, and more specifically, being adopted by dedicated motion control companies like Trio Motion Technology. By incorporating the standard, then adding advanced motion functions like cams, gearing, and interpolated motion, there can now be a uniform programming method for PLCs and controllers specializing in motion and machine control. In this article we want to look at the various IEC programming styles, and how they would apply specifically to motion control. When should you use one language style over the other?

Of course you could program an application using several IEC languages of the controller, and that may be ideal. However knowing the strengths and weaknesses of each style can simplify the task at hand while taking advantage of the controller's capabilities.

Ladder Diagram

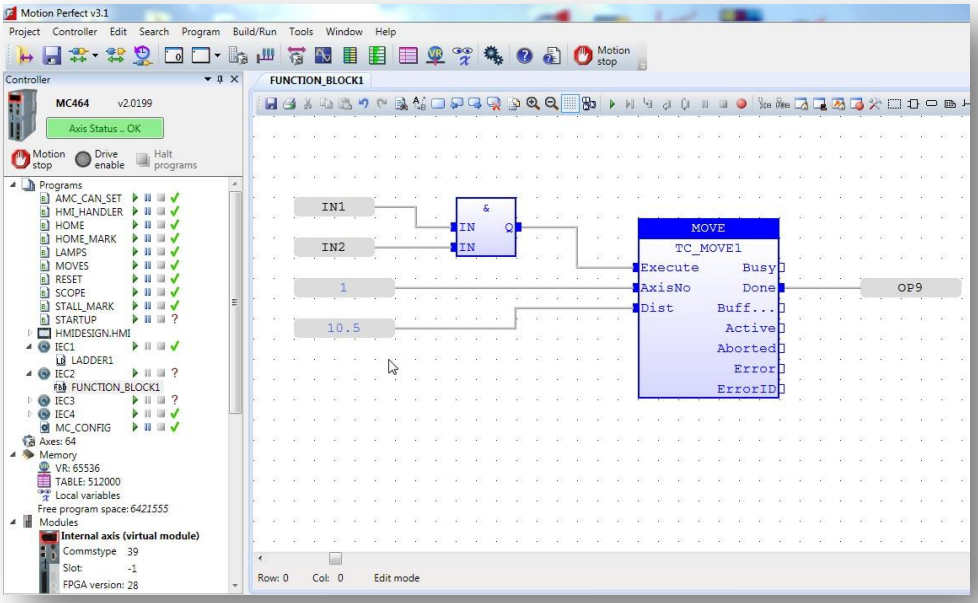
Probably the most widely recognized because of its use in PLCs and analogy to real world circuits is the Ladder Diagram. LD programming is a graphical language that is generally best suited to applications where only binary variables are required and where interlocking and sequencing of points is the primary control problem. An example would be material handling on a conveyor with sensors and discrete output actions. Execution of rungs is sequential within a program.



LD program with a basic interlock, and a triggered move with a Busy and Done output.

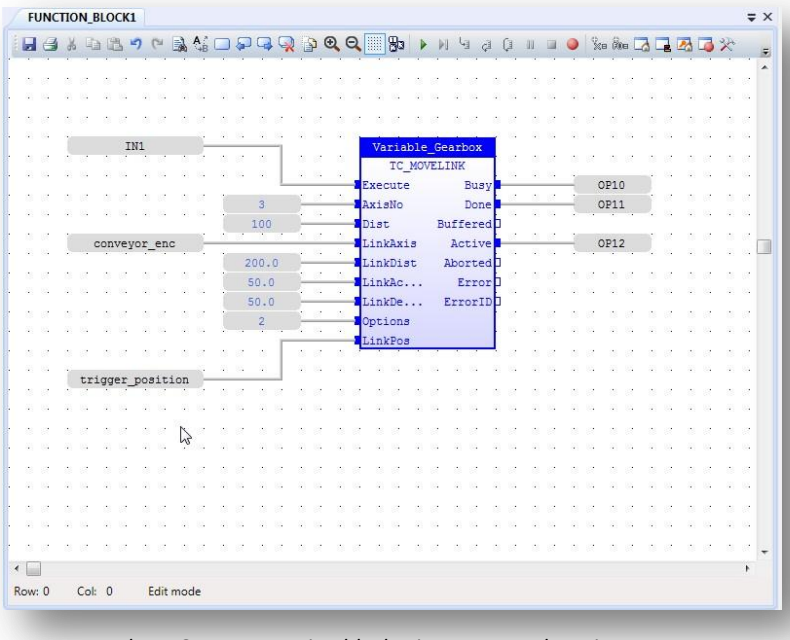
LD makes it easy to visualize machines functions and operations along a rung as it closely represents the physical connections of a system. This makes it easier for the non-programmer to troubleshoot, certainly one consideration for using LD in a motion application. Specific motion functions are not part of the IEC standard but are added by the manufacturer of the controller. The motion function block is setup the same as other IEC functions requiring specific inputs and having outputs. For a simple index move, an input value is specified for an axis number and a distance. The data type for each is required and for the function shown are USINT (small unsigned integer) and LREAL (long real floating point) respectively. A BOOL enable input triggers the motion, which could be from a box passing a photo-eye on a conveyor for example. This basic interlock and triggered move is ideal for LD. Easy to implement, and easy to follow.

Functional Block Diagram



FBD program using two inputs to trigger an incremental move, and an output when done

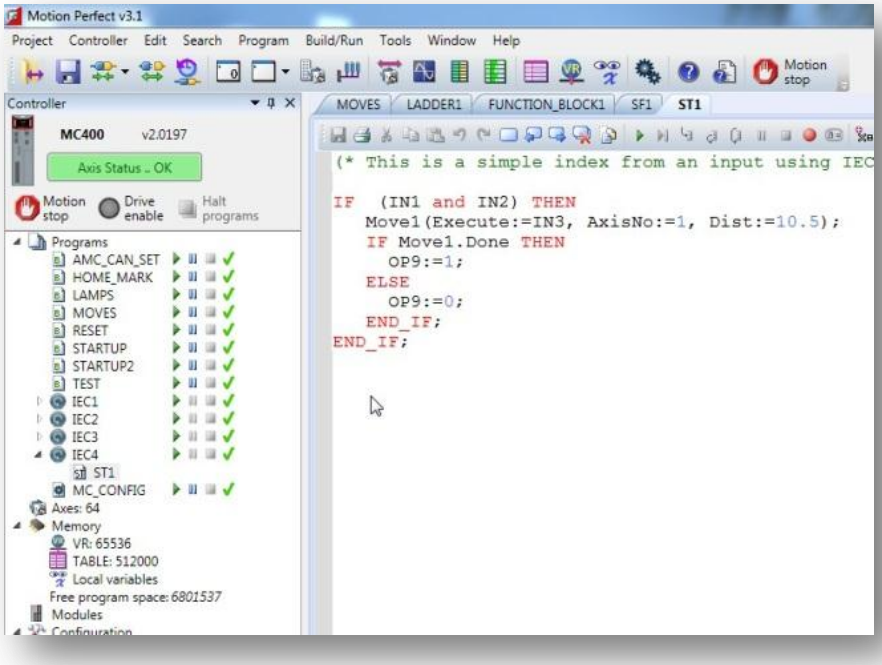
Another graphical language similar in functionality and style to LD is Functional Block Diagram. FBD allows a more freelance style of programming by connecting function blocks and tags together in a circuit fashion. Being graphical it is easy to follow and troubleshoot. Editing a FBD program is also easy using a drag and drop method then connecting the block with 'wires'. FBD is ideal for relatively simple motion processes; however it is possible to tackle motion such as a flying shears or traversing winders using advanced synchronizing motion blocks. A MOVELINK function, for example, performs geared movement set by the Dist input. The LinkAxis and LinkDist inputs are the master reference typically a conveyor or spindle encoder. The LinkPos input sets the trigger position that will start the movement. Such specialized function blocks can greatly reduce the size of a program. FBD is not ideal for large complex motion programs. The large sheet required can quickly become difficult to follow if not carefully planned. In a FBD program, there are no left-right bus bars as in a LD. The diagram is scanned left to right, top to bottom and does not stop or wait. Flags and variables are set to control the flow of operation. For example, the Busy output bit from the TC_MOVE1 motion block can be used to monitor the move in progress by another function block.



The MOVELINK motion block triggers geared motion at a finite distance for flying shear and traverse winding applications.

Structured Text

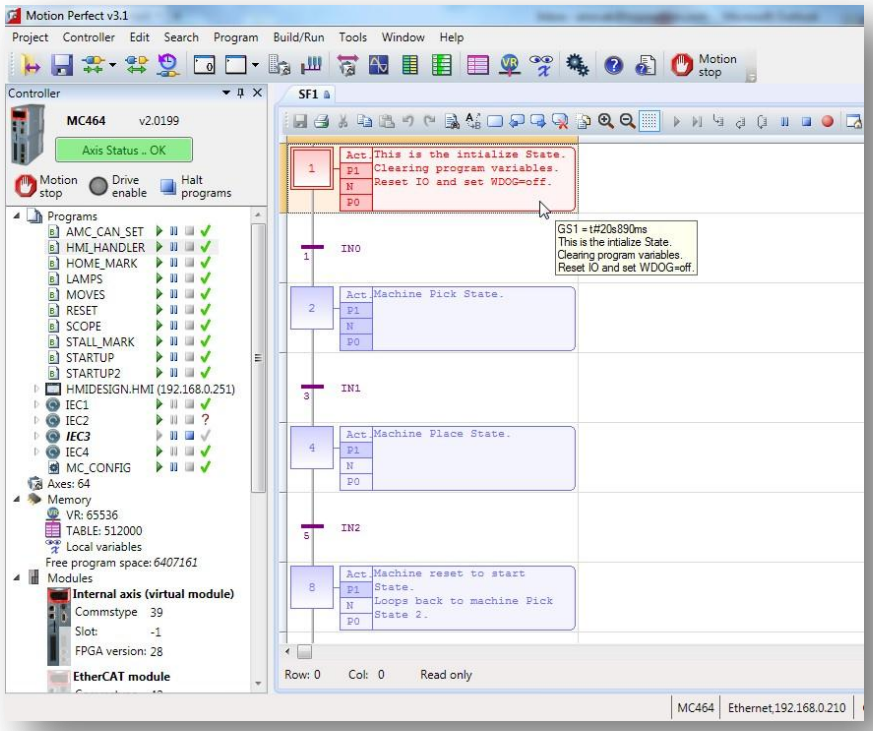
The third language in the IEC standard is Structured Text (ST). ST is a high-level text-based programming language resembling PASCAL or BASIC. As such it is popular among younger engineers having a stronger background in text programming as opposed to a circuit style. Rather than a graphical move block ST uses the same inputs and outputs only spelled out. Axis number and distance are listed as are Busy, Done and other outputs. A physical input is used to execute the motion. Note this code continuously loops [scans] regardless of no explicit looping structure. This is different than a traditional motion programming language such as Trio BASIC that will let the user program a WAIT UNTIL to hold for an event. ST has some very powerful advantages over the graphical styles. For one a very involved program will be much more compact and manageable. A complex high axis count machine such as printing with automatic registration correction and 2D or 3D CNC with file handling would be much better tackled in an ST program than LD or FBD. The simple process of listing and setting parameters is more efficiently handled in ST.



The ST program spells out inputs and outputs to a move function

Sequential Function Chart

The forth style in the IEC standard is Sequential Function Chart (SFC). SFC is analogous to flow charts and not really a language in



SFC program resembles flow chart programming and is ideal for motion defined by steps.

a manner of speaking, but actually a way to organize and control the flow of a process. The concept of a SFC program is pretty simple. An action [step] box with code written in any other IEC language is active until the transition immediately below activates. The next action box then becomes active. Steps in an SFC diagram can be active or inactive. Actions are only executed for active steps.

For motion and process control, SFC has some nice advantages. It's easy to organize and follow the flow of events. This would be helpful for an end user building and supporting an application. A Motion function for example, can be added inside an action box in either the P1 (set), N (continuous), or P0 (reset) sub-steps. The P1 sub-step executes once when entering [setting] the action step. This may be an initial move-to-start motion and firing an output, for example. Then the main program of the step executes continuously in the N sub-step as a typical IEC program. The N sub-step would generally contain the motion and logic control needed for that particular machine state. When the transition logic becomes true for the current running step, the program in the P0 sub-step executes once before exiting [resetting]. A transition could be triggered by a sensor input from a package or some other real world event. The SFC program flows to the next step, executing P1 once, then N, and so on in the new step. Applications best suited to SFC would be a pick and place operation or similar process that has clearly defined repeating states as it runs.

Instructional List

The last and least commonly used language in the IEC standard is Instructional List (IL). IL is a low level language that resembles assembly. It uses very simple instructions similar to the original mnemonic programming languages developed for PLCs.

```
(* This is a simple IL program *)
BEGIN_IL
   LD  x
   AND y
   ST  z
END_IL
```

IL programs use low-level methods similar to assembly language, making it impractical for most motion control.

Being the lowest level language of the IEC standard it does have some notable points. You could convert any IEC program to IL. It is more transferrable to other platforms than the other languages. However, for motion control IL does have some significant drawbacks. Any complex math, PID loop or other high-level functions would be very tedious to program in IL. What can be a simple geared move block in LD or FBD would be a list of low level commands making IL impractical for most motion control applications.

In Summary

The IEC 61131-3 standard defines 5 programming languages for PLCs and distributed control systems. More recently, dedicated motion companies such as Trio Motion Technology are incorporating the standard into their motion control hardware. By adding specific functions such as camming, electronic gearing, and interpolated motion the controllers have a widely supported usability on the factory floor. Additionally, such controllers can be very compact in size and low cost making them a good choice for a vast array of machines.

For simple and moderate motion applications including material handling, pick-and-place and flying shears, LD and FBD programming languages are well suited. SFC programming uses a flowchart approach to organize a process, and contains any other IEC language within each step. If a motion application can be represented as a state machine SFC is an ideal choice being easy to follow and troubleshoot. As applications get more complex as in printing with high-speed registration, 2D and 3D CNC machines, a combination of ST with other languages can be used to take advantage of the controller's programming flexibility.