PLC programming should be viewed as implementation activity of PLC software engineering output, unless you are using PLC as purely part of alternative components to mechanical or electrical solutions.

With this as basis, PLC programming environment is typically IEC61131 driven, gauranteed cycle time, "pre-emptive" realtime, no need to handle realtime OS related issues, continuous code scanning, non-program-pointer, different concept from typical computer task spawning kind of multi-tasking. Code execution is naturally atomic, no need to use monitors between tasks.

Each of the languages has its closeness to how conceivable is your code to the logic model you want to implement.

1. Ladder has its basic concept on electrical power flow interlocking style. Code resolution within single network is either horizontal or vertical scanning (your can find resource on this topic from manufacturer or other sites). If your code has single scan resolution nature and is within one network, some unconceivable behavior can be due to scanning type (important to remember that ladder is only emulation of electrical circuit, it is still sequential in execution).
2. FBD or function block diagram was electronic signal flow but today can be data flow depending on type of PLC. FBD shows clearer execution sequence quite similar to horizontal scanning ladder in scanning sequence. Today, FBD is typically used as container for object function blocks, although dependency implementation and visual similarity to process model is dependent on PLC type.
3. Literal is very similar to BASIC, but syntax only; execution is still scan-through. Literal language is good for mathematical calculation. For high level implementation, methods or derivation of attributes within object can be easier using Literal. State machine programming using English-like state representation or constants makes program very readable.
4. Statement list looks similar to assembly mnemonics but again execution is still scan-through and not program pointer. It is strong in bit operation and parenthesis-styled discrete logics. It can be a very efficient language to use with proper structuring and commenting.
5. SFC or sequential flow chart is a complementary language for sequence implementation. SFC has inherent rules on action block activation, state transitions, parellel sequence activation and merging. However, complex exception branching or concurrent action management can make implementation complicated and flow chart difficult to read.

PLC system management on IO handling, communication, hot-standby is hardware configuration effort, and is product dependent. Generally, can be treated separately from software engineering. However, data related to PLC system management are of "located" (independent data addressing area) type, good data modeling approach in software engineering can help in manageability of system data.

answered Sep 22, 2009 a