Industrial control languages: Forth vs. IEC61131

N.J. Nelson

Abstract

Programmable Logic Controllers (PLCs) have traditionally been programmed in ladder logic, or instruction step list. Each PLC manufacturer had is own programming language, incompatible with all others. Anyone able to program a PLC in a structured high-level language such as Forth, clearly had a huge advantage in terms of program development time, debugging and maintenance.
Now there is considerable industry pressure to adopt a common PLC programming language as defined by IEC61131. This paper will compare the relative merits of programming PLCs in Forth and IEC61131. It will also look at the possibility of using Forth to generate intermediate IEC61131 source code.

N.J. Nelson B.Sc., C.Eng., M.I.E.E.
Micross Electronics Ltd.,
Units 4-5, Great Western Court,
Ross-on-Wye, Herefordshire.
HR9 7XP U.K.
Tel. +44 1989 768080
Fax. +44 1989 768163
Email. njn@micross.co.uk

Introduction

The software community is constantly striving to improve the reliability and efficiency of its work. Programmers of industrial control systems have made relatively little progress in this direction, when compared with some areas of software study. Most programs for Programmable Logic Controllers (PLCs) are still written in “Ladder Logic”, a representation in software of how a control system would look if it were implemented using physical relays. In the early 1990s the International Electro-technical Commission (IEC) started an initiative aimed at improving the standardisation and efficiency of PLCs, including hardware, software and usage. The result of this is the IEC standard 61131.

Overview of the IEC61131 standard

There are currently eight sections in this standard, which were published in stages between 1992 and 1999. Revisions of several sections are in progress. Several sections are of interest to the software engineer (e.g. section 7 deals with the application of fuzzy techniques), but the principal software section is 3 – “Programmable languages: PLC software structure, languages and program execution”.

The IEC61131-3 PLC software standard

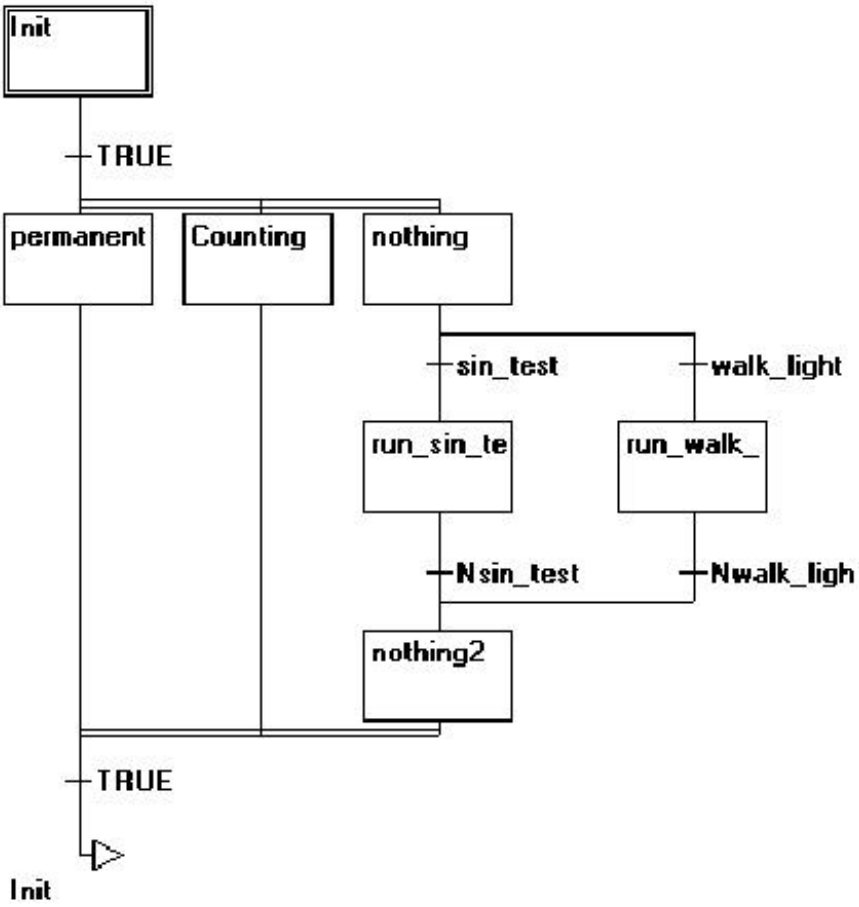
The standard contains five different languages, which can be mixed within an application.

There are three graphical languages:
- Sequential Function Chart (SFC)
- Function Block Diagram (FDB)
- Ladder Diagram (LD)

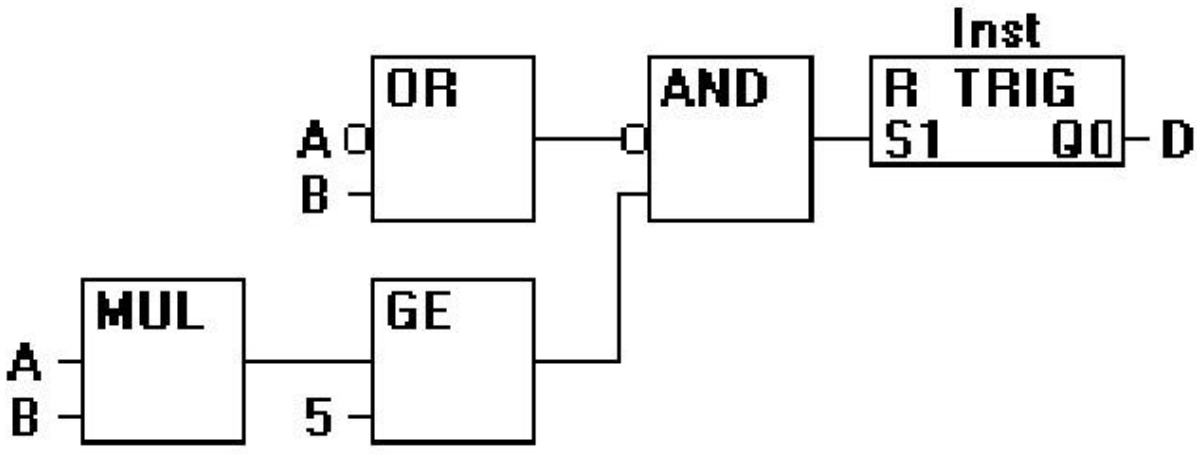
There are two text based languages
- Structured Text (ST)
- Instruction List (IL)

The standard contains specific encouragement for good programming practices, for example, it introduces facilities for structuring and code reuse. As a standard, it claims to be vendor-independent.

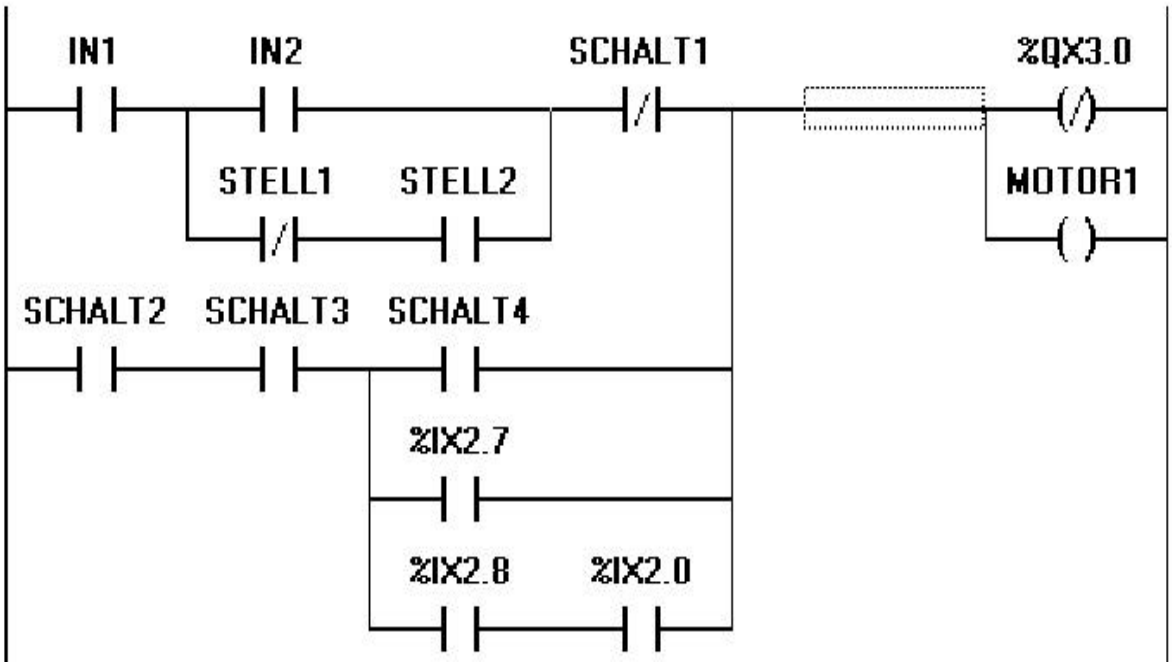
SFC is a method of representing the key elements of a sequential process, i.e. the conditions required for passing from one state to another, and the effects (physical outputs) present while in a particular state.



FDB looks a bit like a circuit diagram of logic elements, which may be quite primitive (e.g. AND gates) or more sophisticated (RS flip-flops).



LD looks like a wiring diagram of a relay control system.



ST looks like a Pascal program.

```
CASE INT1 OF
1, 5: BOOL1 := TRUE;
BOOL3 := FALSE;
2: BOOL2 := FALSE;
BOOL3 := TRUE;
ELSE
BOOL1 := NOT BOOL1;
BOOL2 := BOOL1 OR BOOL2;
END_CASE;
```

IL looks like an assembly language program.

```
LD TRUE (*load TRUE in the accumulator*)
ANDN BOOL1 (*execute AND with the negated value of the
BOOL1 variable*)
JMPC label (*if the result was TRUE, then jump to the label
"label"*)
LDN BOOL2 (*save the negated value of *)
ST ERG (*BOOL2 in ERG*)
label:
LD BOOL2 (*save the value of *)
ST ERG (*BOOL2 in ERG*)
```

The key question

Should anyone who has been accustomed to programming in Forth, consider moving to IEC61131-3? There are three issues to consider:

- Marketing considerations
- Technical considerations
- Efficiency considerations

Marketing considerations

We must consider whether our sales and marketing staff would benefit being able to claim "IEC61131-3 compliance" in the systems we sell.

At present, IEC61131 is "flavour of the month".
Forth, on the other hand, has a reputation of being more of an enthusiasts' toy rather than a professional language.

IEC61131 is a proper international standard.
ANSI Forth is a proper international standard.

The customer's own staff are familiar with IEC61131 and can make program modifications.
So it is often claimed. In fact, in all our installations, there is not a single site where there is a programmer capable of making code modifications to highly complex systems, whether in IEC61131-3 or in Forth. Also, should a programmable component of the control system need replacement, the backup code which we hold will not be up to date. Finally, if a parameter really needs changing from time to time, it should not be hard coded anyway, but should be enterable through a user interface.

In any case, we are very rarely asked nowadays about which language we use for programming.

Technical considerations

We must consider whether there are useful features in IEC61131-3 which are not present in Forth.

According to the standards, programs written in IEC61131-3 are vendor-independent and portable across platforms.
That is the theory. In practice, this cannot happen, because the user interfaces and methods of storing source code (particularly in the graphics languages) differ between implementations. Not all implementations support all the five languages. The languages themselves are very basic, and all but the most elementary programs will need to use additional function blocks provided by the vendor. Since these are outside the standard, they differ between implementations.
Much the same can be said of ANSI forth – but at least the source code is always (nowadays) in the same file format.

The IEC61131-3 standard encourages well structured program development.
So does Forth. But IEC61131 allows unstructured programs too, whereas Forth programs are at least constrained by the need to segment the application into Forth words.

The IEC61131 standard encourages strong data typing.
Whereas Forth does not have any data type checking at all. Our practical experience indicates that lack of type checking in Forth causes very few real bugs. All strongly typed languages must necessarily allow some form of type casting, so data typing errors can still occur. Data type errors in Forth are limited by having very few different types. In 32 bit Forths Windows for example, there are really only five types upon which functions operate – bytes, words, integers, floats and strings – and of these, only integers can be passed as parameters into and out of functions. By contrast, many PLC have dozens of different “types”, which makes it extremely hard for the programmer to remember all their details.

The IEC61131-3 standard allows different sections of an application to execute at different times, and at different rates.
In other words, the standard encourages you to write multi-threaded applications. Most large applications require segmentation between time-critical and non-time-critical sections. But synchronisation between threads is always a difficult issue for programmers, and encouraging the writing of programs containing lots of asynchronous threads is simply asking for trouble.

The IEC61131-3 standard has proper support for describing sequences.
In a paper to this conference in 1997, Jonathan Morrish introduced a technique for describing sequences in Forth in a most concise and clear manner. We have used this technique with complete success ever since.

The IEC61131-3 standard supports data structures.
Although data structures are not explicitly described in the ANSI Forth standard, they can be implemented with the greatest ease.

Efficiency considerations

We must consider whether we could produce results faster using IEC61131-3 compared with Forth.

The IEC61131-3 standard allows for code reuse.
This is true for applications using the same platform. ANSI Forth is easier to port across platforms.

The IEC61131-3 standard allows graphical programming.
There are two considerations here. The first is the quality of the graphical design environment. The ease with which graphical elements can be created, manipulated and connected is crucial to the speed of programming. It must be remembered that text is still required in the graphical languages, for defining the names of the input and output connections. So (in a simplistic example) we must ask whether it is quicker to place a graphical symbol for an AND gate, or to simply type the Forth word AND.

Availability of language implementations

Most PLC manufacturers now provide at least a partial implementation of IEC61131-3. Forth is available only on PLCs over which we have some form of hardware control, either because the PLC is our own in-house design, or because we have sufficient knowledge of its architecture to create a Forth implementation.

Providing a Forth interface to an IEC61131-3 system

A much wider choice of PLCs would be available to the Forth programmer, if it were possible to generate IEC61131-3 using Forth. There are three possible ways in which this might be achieved, and we have examined these methods in conjunction with the implementation of IEC61131-3 provided by Beckhoff (Beckhoff Industrie Elektronik, Eiserstrasse 5 D-33415 Verl). A free 30-day limited copy of this implementation, which is called TwinCAT, can be downloaded from www.beckhoff.com.

The first, and simplest method, would be to generate source code in IL form. Some implementations actually store IL source code as plain text files; others at least allow IL text files to be imported. Since IL is very similar to a generic assembly language, and methods of using Forth to generate assembly language source code have been demonstrated before, this implementation is quite straightforward. However, the standard defines a rather restricted list of IL "opcodes", therefore the functionality of applications written solely using this method would be quite restricted.

A similar method would be to generate ST source code. This is quite similar to Pascal, and although we have never seen a Forth-to-Pascal generator, it would seem to be possible.

Finally, the standard allows the code definitions for FDB to be expressed in ANSI "C", compiled using a standard external compiler. In the Beckhoff implementation, the Microsoft VC++ compiler is used to produce an object file that can be linked to the IEC61131-3 project. Forth-to-C generator programs have been described before, therefore, in a somewhat convoluted way, Forth could be used to generate IEC61131-3 function blocks. This method would only allow the internal descriptions of the FDB functions to be defined in Forth. A complete application would need a section in graphical FDB language to link the function blocks together.

So far, we have investigated only the first of these possible techniques.

Conclusion

To anyone who was accustomed to programming in ladder logic, moving to IEC61131-3 would represent a huge improvement. To anyone who was accustomed to programming in Forth, and who has access to Forth on the required platform, moving to IEC61131-3 would be a retrograde step.

References

1. The IEC61131-3 standard can be purchased online from the International Electro-technical Commission at www.iec.ch, price Swiss Francs 110.

2. “Programming industrial control systems using IEC 61131-3”, R.W. Lewis, IEE 1988, ISBN 0 85296 950 3

3. “Rapid development of real-time multi-sequence controls programmes”, J.H. Morrish, EuroForth 1997