PLC基础篇之编程语言的魅力

原创文章，转载请注明出处。

更多实用资料请登录方正智芯官网：www.founderchip.com

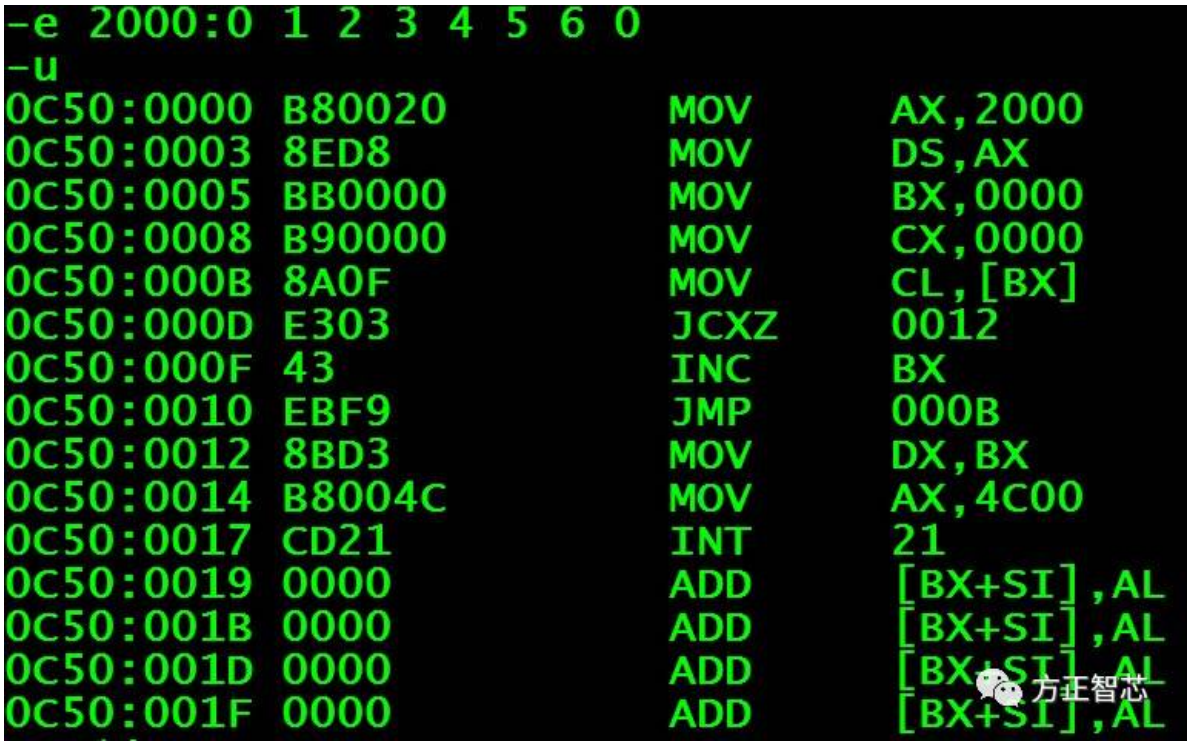
作者：北岛李工

微电子技术的发展使得集成芯片具有了一定程度的“智能”，微处理器（CPU）能够按照预先设定好的“程序”来运行，人们通过编写“程序”代码，达到与计算机（PC）沟通的目的。

在计算机技术发展的早期，“程序”代码的编写并不是一件简单的事情。由于计算机本身只能理解“0”和“1”组成的代码，于是编程人员只能把“0”和“1” 的数字编成的程序代码打在纸带（ “1”打孔，“0”不打孔）上，然后将纸带放入计算机的输入系统中，这样计算机才能明白程序员让它完成什么任务。程序员编程时要充分定义好每一个“位”的含义，然后按照某种逻辑将它们组合成代码。这种由“0”和“1”组成的代码十分地复杂，人类很难理解（不知道外星人看着好理解不？），给程序的编写、修改和维护都带来了很大的困难。下面这张图片是十六进制的机器代码（不管你能不能看懂，我反正是看不懂）：

为了提高程序编写的效率，提高代码的可读性及可维护性，计算机科学家们发明了“汇编语言”。汇编语言使用特定的英文符号代替机器指令，大大增强了程序的可读性，提高了编程的效率。

汇编语言的英文指令虽然便于人类理解，但微处理器（CPU）却无法理解。于是科学家们设计出一个专门的程序，用来把汇编语言的英文指令，“翻译”成微处理器（CPU）能够执行的机器代码，这个翻译程序，被称为“编译器”（编译器也有个发展的过程，不深究）。那些用汇编语言写成的代码，被称为“源代码”。下面这张图片，左边是机器代码，右边是汇编语言写的“源代码”：



汇编语言编程与早期的纸带打孔编程相比，程序的可读性大大增强。但随着微处理器技术的迅猛发展，用汇编语言来写程序也逐渐不能满足需求了，主要表现在两个方面：

第一：随着程序复杂性的提高，汇编语言编写的代码量显著增加，编写和维护的难度变大；

第二：汇编语言依赖于特定的微处理器，程序的跨平台移植性很差，很多时候需要重新编写；

于是，一种不需要记忆繁杂指令的、与硬件平台无关的编程语言就成了程序员们梦寐以求的东西。于是，计算机高级编程语言便顺天应地的诞生了，其典型代表是C语言。C语言用结构化的语句代替了汇编语言中的指令，提供了丰富的数据类型和运算符，支持指针功能，编写的代码简洁紧凑，自其诞生之初就受到了众多程序员的追捧，被视为编程语言的经典。时至今日C语言仍保持着强大的生命力，在单片机和嵌入式程序设计中有着不可替代的地位。

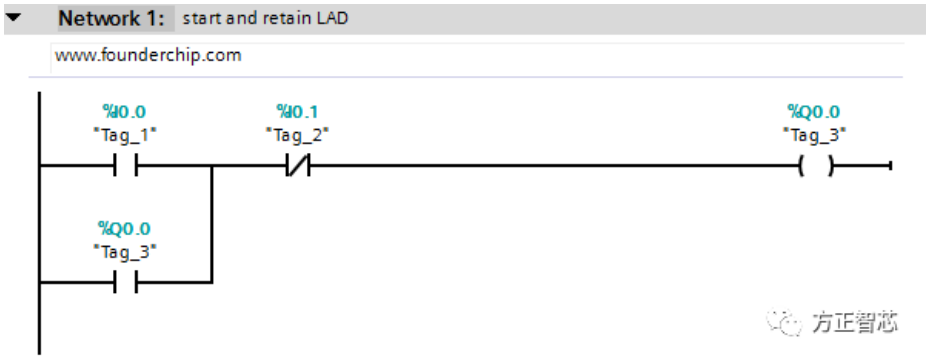
随着微电子技术在工业控制领域的使用，PLC（可编程逻辑控制器）取代了传统的继电器控制系统。作为“可编程”逻辑控制器，编程语言必不可少。但是由于各大厂家的PLC产品自成一派，相互不兼容，编程语言的语法也各有所好，形形色色，PLC应用和推广带来了不便。

为了规范PLC的编程语言，国际电工委员会(International Electrotechnical Commission)起草并颁布了工业自动化领域编程语言的标准（IEC 61131-3），制定了五种在工控领域使用的语言，包括图形式语言和文本式语言。图形式语言包括：梯形图（LD - Ladder Diagram）、功能块图（FBD - Function Block Diagram）和顺序功能图（SFC - Sequential Function Chart）。文本式语言包括：指令表（IL-Instruction List）和结构化文本（ST-Strutured Text）。

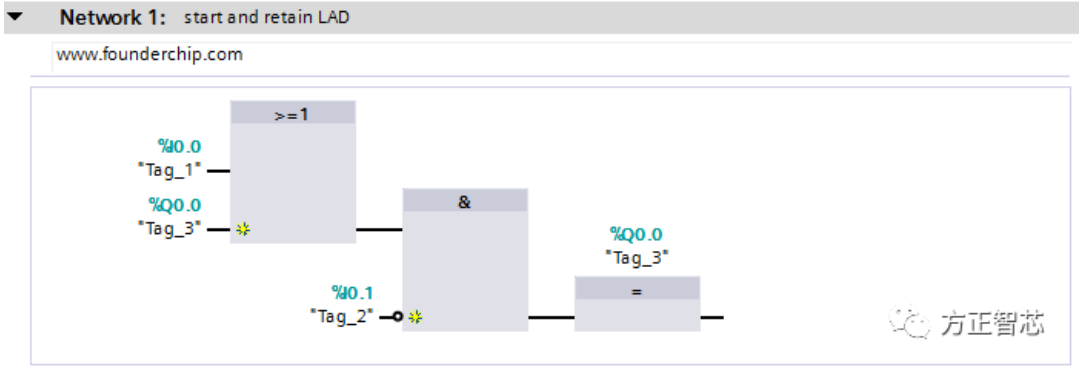
为了进一步推广IEC61131-3语言标准，1992年成立了PLCOpen国际组织，总部位于荷兰。PLCOpen国际组织是独立于生产商和产品的全球性机构，其宗旨是规范工业控制领域软件的编程方法，推广IEC61131-3语言标准。符合该标准的编程语言，给予颁发PLCOpen证书。
IEC61131-3语言标准颁布后，各大PLC厂家纷纷表示支持，对自己的产品进行了修正，下图是西门子公司S7产品编程语言与IEC61131-3语言标准的对比：

IEC 61131-3 语言	SIMATIC S7 产品	PLCopen 证书	
		基础水平	可用性水平
IL	STEP 7 (STL)	--	--
ST	S7-SCL	+	+
LD	STEP 7 (LAD)	--	--
FBD	STEP 7 (FBD)	--	--
SFC	S7-GRAPH	+	方正智芯

下面我们来聊聊这五种编程语言：
在各种PLC的编程语言中，使用最多的梯形图（LD - Ladder Diagram）语言。梯形图是从早期继电器控制系统原理图演变而来，与继电器电路图相似，直观易懂，保留了继电器电路图的风格和习惯，是熟悉继电器控制系统人员最容易接受和使用的语言。下图是一个简单的梯形图代码：



梯形图虽然容易上手，但是在编写大型系统程序的时候会很吃力。很多书都建议初学者学习梯形图，我不敢苟同，除非你接触的都是些简单逻辑控制。当你感到需要控制的逻辑系统变得复杂，想换一种语言的时候，基本还是要从头学起。所以我建议初学者学习功能块图（FBD - Function Block Diagram）。
功能块图（FBD）使用数字电路的逻辑符号（“与”、“或”、“非”）来表达控制逻辑，在编写大型复杂系统的时候也能保证逻辑清晰。下图是功能块图（FBD）的简单示例：



结构化文本（ST-Strutured Text）编程语言，在西门子PLC编程中被称为SCL（Structure Language），先给你看看我在某项目中使用SCL编写的程序代码的截图：

```
63  //Connect: connection built up
64 IF #UDP_Ctrl.Connect.Done   THEN
65      #UDP_Ctrl.Connect.Request := 0;//reset request if connection is completed
66      #UDP_Ctrl.Connect.Connected := TRUE;
67  END_IF;
68 IF #UDP_Ctrl.Connect.Error THEN
69      // if error happened during connection,reset request
70      #UDP_Ctrl.Connect.Request := 0;
71  END_IF;
72  //Give out bits
73  #CONNECT_DONE := #UDP_Ctrl.Connect.Done;
74  #CONNECT_BUSY := #UDP_Ctrl.Connect.Busy;
75  #CONNECT_ERROR := #UDP_Ctrl.Connect.Error;
76  #CONNECT_STATUS := #UDP_Ctrl.Connect.Status;
```


是不是和计算机编程很相似？SCL的语法类似VB（PASCAL）等高级语言，接近人类的思维方式，程序的可读性很强。在西门子Step7 5.x平台下可以使用插入源文件的方式进行编程，在博途平台下可以直接编写。SCL可以说是工控领域里“高大上”的编程语言，它的源代码编译后的效率也很高，如果你之前接触过计算机高级语言编程，强烈建议深入学习下SCL语言。

指令表（IL），在西门子PLC中称为语句表（STL）。它类似于汇编语言，对编程人员要求较高，需要熟悉PLC内部的各种寄存器、状态字等等，需要熟悉各种指令，并清楚某指令执行后会对哪些寄存器产生影响。语言表（STL）编写的程序可读性相对较低，但其执行效率在所有的语言中是最高的。有些特殊的功能使用其它语言（比如梯形图）表达很困难，或者根本无法表达，语言表（STL）可能几行代码就完成了。下图是一段简单的STL代码的截图：

Network 1: input handling

Comment

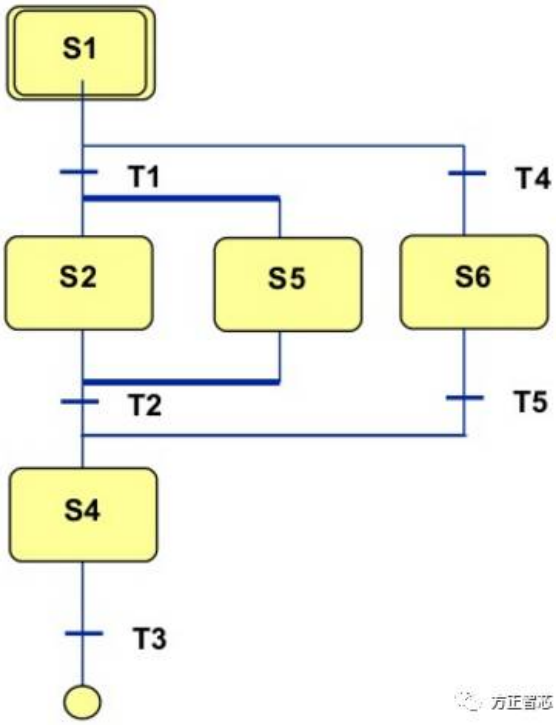
```
1      A     "T_ON"              %I0.0
2      AN    "T_OFF"             %I0.1
3      FP    "M_FP_Start_ON"     %M20.1
4      S     "Start_ONOFF"       %M1.0
5      A     "T_OFF"             %I0.1
6      R     "Start_ONOFF"       %M1.0
7
```

Network 2: conveyor go right

Comment

```
1      A     "T_Right"           %I0.2
2      AN    "T_Left"            %I0.3
3      =     "Conv_Right_Manual" %M1.1
4
```


顺序功能图（SFC- Sequential Function Chart）语言，在西门子PLC中称为S7-GRAPH。该语言非常适合生产过程的顺利控制，它把整个生产过程分成若干“步”，每一“步”都有某些动作要完成，当某个条件满足后，可以跳到下一“步”，也就是所谓的“顺序控制”。下图是S7-GRAPH代码的示意图：



IEC61131-3推荐的五种语言，在不同的工控场合下均有使用，作为初学者，建议学习功能块图（FBD）语言。如果你有计算机高级语言编程的基础，建议学习下SCL语言。当然，无论学哪种语言，都首先要理解PLC本身，要知道PLC内部资源是如何存储？程序是如何调用？知道了这些后，你才清楚程序应该怎样写，以及为什么这样写。

本来我只想写一篇介绍PLC语言的基础文章，想到这些语言编写的代码其实都需要编译成机器代码，后来又联想到计算机编程语言，于是就想从源头给大家介绍下语言的演变过程，于是乎就写了这么多，希望能帮你更好的理解计算机和PLC编程语言，在你以后的程序编写过程中，感受编程语言的魅力。就先写到这里吧，相关参考文章：

PLC基础篇之PLC的诞生及工作原理

S7-1200硬件篇之读懂CPU的内部存储区

S7-1200硬件篇之重新认识CPU

官网提供本文PDF版本的下载：