**一种多层架构嵌入式PLC扩展CNC的方法研究**

摘要：本文实现了一种在嵌入式PLC中扩展CNC的方法。嵌入式PLC采用彼此独立的多层架构，在控制程序层上增加一个CNC层，实现对工艺流程和参数的调整。在采用多线程的执行系统增加一个CNC驱动线程，用来解析和驱动CNC程序。

关键字：embedded PLC，CNC，motion control

## 引言

PLC以其稳定可靠和使用方便的特性广泛应用于工业控制领域，早期主要用于逻辑控制[0,1]。但是随着控制过程越来越复杂和要求越来越高，控制更趋分散化、个性化和智能化，传统PLC在这些方面都显得力不从心，制约了PLC的发展。因此不少学者开展了大量研究来增强PLC性能，如通过算法及相应的功能模块增强PLC[2]，通过设计方法提高PLC安全性和可靠性[3,4,5]以及通过预测和发现用IEC61131-3标准语言开发的控制程序中的错误提高程序质量[6]。除了增强PLC本身的性能外，随着集成电路技术的飞速发展，PLC的功能也得到增强，目前PLC普遍具备了运动控制能力，如定位控制、直线插补、圆弧插补等，并得到了广泛的应用【35,36,39, 48,49,50,51，52,】。

PLC运动控制功能的扩展大大简化了控制系统，使得一些简单的运动控制无需专用的运动控制器，通过PLC简单的开发就能完成常用运动控制功能。PLC扩展运动控制的方式多种多样，一种PLC扩展运动的方式是在PLC中集成简单的运动控制功能【37，38,40，47】，即将逻辑控制和运动控制结合在一起。由于运动控制和逻辑控制在同一个循环中完成，因此无法保证运动控制的精度和速度，一般用在一些简单的运动控制领域。另一种方式是通过独立的运动控制模块扩展，通过总线和PLC构成控制系统。运动控制模块有独立的执行结构和开发方式，只通过通信方式和PLC交互，能完成比较复杂的运动控制功能【41，42】，如松下FP2位置控制单元[43]。但是这种扩展方式需要用户购买额外的运动控制模块，需要较高的控制成本。

为了使PLC适应不断提高的控制需要，PLC的构造形式发生了重大变化。原有的PLC所有功能都由一块电路板实现，并封装在一个盒子内，仅通过IO端口和通信口与外界交互。在实现方式上根据功能不同选择使用1个或多个CPU，CPU间通过总线方式连接。当使用多个CPU时，将从CPU数据映射到主CPU实现统一编程，如图1所示。



图1 一体化PLC结构示意图

后来随着PLC功能的扩展，为了根据控制需要灵活配置控制系统，出现了各类扩展模块，如IO模块、模拟量控制模块、运动控制模块等，PLC可以只保留基本的逻辑控制功能，通过扩展模块实现功能的扩展，如图2所示。PLC和扩展模块间通过工业总线连接，实现数据和命令交互。



图2 模块化扩展PLC结构示意图

传统PLC难以构建开放的硬件体系结构，随着嵌入式处理器功能的增强，通过一个嵌入式处理器完成多种控制功能已成为可能，这就是目前新兴的嵌入式PLC。嵌入式PLC以嵌入式处理器为核心，按需添加外围电路，多个PLC间可以通过网络连接，如图3所示。嵌入式PLC在组成形式上更加灵活，不再局限于传统的盒式结构，是未来PLC的发展趋势。

图3 嵌入式PLC结构及扩展示意图

在PLC控制程序开发方式上，早期不同PLC厂商支持各自不同的编程方式，缺乏统一的规范。从上个世纪80年代开始，IEC国际标准化组织着手制定统一的可编程控制器的国际标准，以对PLC未来的发展制定一个方向或框架，并且相继颁布了一系列的PLC标准。其中IEC61131-3【24,25,45】是工业自动化控制系统的软件设计提供标准化编程语言的国际标准，该标准目前得到了大部分PLC厂商的支持。它极大地改进了工业控制系统的编程软件质量及提高了软件开发效率；它定义的一系列图形化语言和文本语言，不仅对系统集成商和系统工程师的编程带来很大的方便，而且对最终用户同样会带来很大的方便。

在具备运动控制功能的PLC系统中，开发方式各不相同。在PLC内集成运动控制的，采用PLC开发环境直接开发。对于独立的运动扩展模块，有采用G-Code开发，如OMRON公司CS1系列PLC的MC421/221；有通过专用设计语言开发，德国JETTER公司的PROCESS PLC的文字描述型语言SYMPAS【33】，或者在PLC梯形图编程中提供运动控制相关指令，如松下FP系列PLC【46】。为了满足不同的应用需要设计人员需要选择不同的控制器，并采用特定的开发环境和开发方式完成系统的设计和开发，为设计人员带来了极大的麻烦。为了解决这个问题PLCOpen TC2 推出了Motion Control标准【26,29】，该标准提供了可在不同硬件平台间复用的标准库，以降低复杂度，同时减少开发和维护的费用【27】。目前很多主流开发工具都采用了该开发模式，如3S公司的CoDeSys和K.W.的MULTIPROG。CoDeSys SoftMotion将PLC逻辑控制和运动控制(Motion Control)合二为一，将运动控制功能工具包集成在PLC编程系统中，采用IEC 61131-3标准编程语言作为开发手段，运动控制模遵循PLCopen motion control标准，以POU的方式插入PLC控制程序。对一些复杂的运动控制，还可扩展CNC POU库（CNC POU library），提供如插补等复杂的运动控制功能。

通过分析发现，目前PLC运动控制功能的实现基本都是在PLC编程系统中插入运动控制功能模块，这种开发方式具有以下几个缺点：

1. 功能开发和工艺开发混合在一起，参数优化调整困难。开发人员在开发某个设备控制程序时，不但要完成功能开发，还需要熟悉工艺流程和工艺参数，开发周期长。同时工艺参数固化在程序中，当需要调整参数时需要重新修改程序，调整过程复杂。
2. 复杂程序开发、维护和调试困难。IEC61131-3标准的几个开发语言适合完成逻辑控制程序的开发，对于具有运动控制的复杂程序开发是一个挑战，程序的编制、纠错和维护都很困难。
3. 最终用户无法按照实际控制需要调整控制程序。最终用户熟悉被控设备，为了达到最优控制往往需要优化控制流程和参数。但是现有的开发方式不支持最终用户对控制程序的调整，IEC61131-3开发语言不但需要一定的开发技巧，而且为了程序稳定可靠，控制程序一般都不开发给最终用户。

CNC是运动控制重要分支，CNC作为数控设备核心广泛应用于数控加工中心，通过CNC语言，如G-Code，完成运动过程定制，由执行系统完成设定的动作，实现产品加工。随着对数控系统开放性、交互性和适应性等要求越来越高[20]，STEP-NC作为一个面向对象的新型NC编程数据接口国际标准被提出并得到了广泛的认可[21,22],也有研究借助CAD图形化工具自动生成CNC代码，降低开发难度和编程错误.{ In recent years, the automatic programming technology is developed rapidly. The mistake caused by programmer is decreased obviously. }[23]。

由此可见开放性、可靠性和开发方便性是运动控制系统的追求目标，针对现有嵌入式PLC运动控制程序没有分离功能开发和工艺开发的问题，提出了一种新型的多层开发架构，将不同层次的开发人员彼此独立，每层需专注于各自任务的开发。并在最上层引入CNC层，在不影响程序稳定性的前提下为最终用户提供了工艺流程和参数调整的手段。

本文的组织结构如下，

## 基本思想和概念

### 2.1 多层设计架构

本文核心思想是在PLC控制程序开发中引入CNC层，并在执行系统中增加一个CNC线程实现CNC程序的解析和驱动。传统PLC内核执行系统中包含一个控制程序驱动模块，该模块通过循环扫描方式，重复执行输入刷新、控制程序、输出刷新的过程，同时通信、异常处理等其他功能也在循环中完成。随着PLC功能的增强和控制过程的复杂度增加，为了提高控制效率PLC内核逐渐开始采用多线程方式。在多线程内核中嵌入CNC功能就是要加入一个CNC处理线程，实现CNC程序的解析和执行。

为了实现功能设计和工艺设计分离的目的，在开发方式上采用了三层开发架构，如图1所示：

（1）Engine Layer。在Engine Layer中，专业开发人员通过C语言或汇编实现PLC内核及算法的编写。该层实现的程序称之为引擎程序(Engine Program，EP)，包括PLC的初始化、中断、定时器、通信和各类算法模块等。和运动控制相关的算法模块也在引擎程序中。

（2）Control Program Layer。在Control Program Layer中技术人员通过IEC61131-3标准支持的开发语言实现程序的编写，该层实现的程序称之为控制程序（Control Program,CP），包括逻辑控制和运动控制两大部分。和运动控制相关的功能通过调用底层控制算法模块实现，只完成运动控制功能，不涉及具体的工艺流程和参数。

（3）CNC Layer。在CNC Layer中，用户通过CNC 语言实现运动控制的序列和参数定制，CNC程序和实际被控对象的工艺相关，其功能通过顺序调用Control Program Layer中的运动控制模块实现,该层实现的程序称之为数控程序（CNC Program，CNCP）。通过CNC程序，用户能根据实际控制需要灵活定制运动的步骤和参数。

图1 嵌入CNC的PLC多层架构示意图

通过上述三层架构，能将一个复杂系统分割为几个简单系统，并且大大提高了系统的灵活性。一方面，不同开发人员间彼此独立，只要专注于各自任务的开发；另一方面，通过引入CNC层，控制程序中的运动控制部分不和实际应用相关，只需完成基本的功能，可以保持控制程序的稳定；而且用户能根据实际控制需要定制和调整控制程序，提高满足不同实际控制需要能力，用户不需要掌握专业的开发知识，降低了开发难度。中间控制程序层的存在，相比通过数控语言直接调用底层控制算法的实现方式，不但解决了逻辑控制的问题，而且使得运动控制具有更好的灵活性。

每一层开发的程序都存放在PLC的FLASH固定位置，上电后搬移到RAM中运行，以后都以在RAM中运行的情况进行讨论。

### 2.2线程关系分析

在我们的PLC系统中采用了基于优先级的抢占式多线程调度系统，系统共有四个优先级级别和六个线程，如图2所示。在同一级别上的线程具有相同的优先级，采用时间片方式分配运行时间。



图2 多线程结构示意图

Emergent Thread: Emergent Thread 具有最高优先级，用来处理紧急事务，如中断等。本系统中所有中断，包括IO中断、定时器中断和异常中断等，中断触发后不立即处理，而是压入队列，通过统一的Emergent Thread处理。

CNC Thread、Control Thread和Motion Thread: CNC Thread 、 Control Thread和Motion Thread 具有相同的优先级。CNC Thread用来处理CNC Program，完成CNC程序的读取、解析和执行；Control Thread处理Control Program，包括读入输入信号、执行控制程序、刷新输出、响应CNC thread请求和调用引擎算法执行；Motion Thread执行PLC引擎中运动控制算法，响应Control Thread请求，通过调用运动控制算法完成被控设备运行。用多线程协作的目的是在多种控制集成的环境中，避免因某个任务的延时而影响其他控制任务的执行，提高系统的响应，如运动控制线程在等待执行机构结束运动时，IO逻辑控制应正常工作。

Communication Thread: Communication Thread 用来完成PLC的通信任务，响应各类通信请求，如人机界面等。

Diagnose Thread: 在我们的系统中嵌入了一个诊断模块，在不影响正常控制功能的前提下，利用CPU的空闲时间完成系统故障诊断。

本文主要内容是嵌入CNC线程，并解决CNC thread 、 Control thread和Motion Thread间的交互和协作，共同实现设备控制。

### 2.3数据交互关系及定义

多线程系统中多个线程协作时，需要在线程间交互数据，解决线程的同步和互斥等问题。本系统中线程间数据交互及算法调用时的参数传递是通过共享内存实现的，和传统的PLC实现方式类似，在PLC内存中划分了一个专用公共数据区，用来保存各类软元件的值。

数据共享中的参数传递通过D区实现，线程同步互斥通过M区实现。在D区和M区中有若干个专用区域用来实现数据交换和运行控制，如图3所示，这些数据区定义如下：

定义4 引擎算法参数区（Engine Algorithm Parameter Data Area, EAPD）：在D区中的一个专用区域，用来存放引擎算法实现所需参数，所有算法参数都有固定的地址。

定义5 引擎算法标志数据区(Engine Algorithm Flag Data Area，EAFD): 在M区中的一个专用区域，用来表示引擎算法的启动、状态及完成标志，在梯形图中通过设置该标志区的标志启动相应算法执行，并通过读取状态标志获取算法执行情况。

定义6 CNC程序专用数据区（CNC [Dedicated](http://dict.bing.com.cn/?FORM=BNGCN#dedicated) Data Area, CNCDD）：在D区中的一个专用区域，用来存放从FLASH中搬移到RAM的CNC程序。

定义7 CNC程序参数交换区（CNC Parameter Swap Data Area，CNCPSD）：在D区中用来暂存解析CNC程序后得到的参数，只存放将要执行的数控指令参数，控制程序从该数据区获得执行算法所需要的参数，并将参数搬移到EAPD。

定义8 数控指令标志区(CNC Trigger Data Area，CNCTD)：在M区中的一个专用区域，每一位控制Control Program中和某条数控指令对应的程序模块执行。



图3 PLC专用公共数据区分配情况

## 引擎及算法实现

引擎程序（EP）实现PLC初始化、中断、时钟和通信等基本功能（BF），也包含若干增强系统功能的扩展算法（EA）。

设EP={BF,EA，EAD，EAS},EA={ea1,ea2,…eai…eam},表示引擎中共包含m个算法；EAD={ead1，ead2，…eadi…eadn}，引擎算法参数数据区包含n个参数；EAS={eas1,eas2,…easi…,eask},k个标志用来控制每个算法的执行和标记算法的运行状态；

和m个算法对应，EAD含有m个子集，EAD=EAD1∪EAD2∪…∪EADm,EADi 表示实现eai所需的所有参数集合，记作EADi»eai。而且∃EADi»eai·∃EADj»eaj:i≠j→EADi∩EADj=Φ。

EAS含有m个子集，EAS=EAS1∪EAS2∪…∪EASm,EASi 表示实现eai所需的所有标志数据区集合，记作EASi»eai。而且∃EASi»eai·∃EASj»eaj:i≠j→EASi∩EASj=Φ。

触发标志值集合TS={on,off},由上层调用层触发，在引擎中清除。运行状态集合RS={run，stop}，在引擎中设置和清除。当向执行机构发出运行指令后，将运行状态设置为run，一直到运行结束，运行状态被设置为stop，用来表示算法的运行状态。笛卡尔积（Cartesian product）TS×RS表示所有可能的状态组合，即对任意EASi元素的取值都是TS×RS的元素。

<off,stop>表示引擎算法处于Stop状态，在该状态下当触发标志从off被置为on时，<on，stop>标记进入Ready状态，在该状态下算法随时可以被线程调度执行，在执行前触发标志被清除，防止重复执行，同时将运行标志置为run，表示引擎算法处于运行状态，然后进入状态Run，当算法运行结束后运行状态标志被清除，重新进入Stop状态。



图 4 引擎算法状态转移图

## CNC设计和实现

在我们的多层设计架构中，CNC用来实现运动序列和参数的定制。包括程序设计、编译和解析执行三个部分。一个CNC程序由若干条CNC指令组成， CNC指令是组成CNC程序的最小单位，每条指令完成一个相对独立的功能，一个复杂运动过程由多条具有时间先后顺序执行的数控指令完成。在本系统中，CNC指令用来传递参数并触发控制程序中对应程序模块的执行。因此，首先需要设计满足使用需要的CNC指令。

### CNC指令的设计与实现

指令系统包含m条指令，IS={I1，I2，…,Im},Ii=（name，code，P）,任何指令都包含指令名name，指令码code和若干完成指令所需的参数集合P。指令名是指令功能助记符，指令码code用来唯一确定一条指令，即﹁∃Ii∃Ij: Ii.code==Ij.code∧i≠j。在我们系统中采用了自然数N作为指令码。

在PC端设计环境中，只使用指令名和参数，在格式上，采用指令名和参数列表结合的方式，即：

*Ii.name Ii.p1，Ii.p2，Ii.p3，…，Ii.pn*

任一CNC程序CNCP都是由有穷条具有先后执行顺序的指令组成。

### CNC程序的编译与执行

**1）CNC程序编译**

CNC程序的编译是将设计的数控程序CNCP转换成PLC中能识别的程序CNCPˊ，*f*:CNCP→ CNCPˊ，编译包含两步完成，f=g○h。

函数g完成指令名向指令码的转换，g: CNCP→T，g(Ii.name)=Ii.code，i=0,1,…,m，显然g是一对一的。

函数h完成参数的归一化处理，h:T→ CNCPˊ，将数据转换成标准格式。设计时为了增加程序的可读性，数据采用的都是易于理解的数据格式，如对距离长度的描述可以采用实际长度，也可以采用脉冲数，而在电机运行时长度都采用了脉冲数作为长度单位，因此需根据设定的转换信息进行参数转换。设在某系统中，伺服电机一圈分为10000个Pulse，1mm对应20个Pulse，以G00、G04、G90为例，设其指令码分别为i、j、k，对如下指令序列：

N100 G90

N101 G00 X12.00 Y33.500 Z24.000 S50

N102 G04 2

N103 G00 X11.00 Y36.500 Z16.000 S50

N100行表示采用绝对模式，N101表示以50mm/s的速度定位到坐标（12.00,33.50,24.00），N102表示暂停2秒钟，设使用1ms定时器，N103表示以50mm/s的速度定位到坐标（11.00,36.50,16.00，）处。则编译转换后的代码为：

k

i, 12000 ,33500 ,24000 ,50000

j, 2000

i, 11000 ,36500 ,16000 ,50000

所有编译后的CNC程序指令按照设计时顺序组合后，在最后加上END标志后构成下载到PLC中的数据帧，如图5所示。为方便解析执行，规定各参数均为4字节整数。该数据帧下载到PLC后被CNC驱动模块解释执行。



图5 编译后数控程序格式

**2）CNC程序解析执行**

CNC驱动模块完成从CNC程序专用数据区CNCDD中读取每条CNC指令，解析指令获取参数并将参数传递到指定地址，并启动指令执行。

解析指令所需的信息保存在一张CNC指令解析表中，一条解析表项包含了一条数控指令解析所需的信息。PT={pi1,pi2,…,pim}，解析表项数和指令数相等。对任一pii=(code,np,pea,ta)，code为指令码，用来和CNC指令关联；np为参数个数，参数个数表示执行指令所需参数的个数，由于约定每个参数占4个字节，因此CNC驱动模块通过参数个数可以确定一次数据搬移的大小，将数据搬移到参数交换区CNCPSD；pea为参数交换区地址，参数交换区地址是参数交换区的起始地址，为相对D区起始地址的偏移量，用于暂时存放CNC指令参数，供控制程序执行时使用；ta为启动标志起始地址，启动标志起始地址是启动控制程序中和当前指令对应的程序模块的标志位所在的地址，是相对M区起始地址的偏移量，以指令码作为字节内偏移可以确定指令标志位的位置。

解析和执行过程包含如下几个步骤：

步骤1：指令读取。从CNC程序区当前指针位置DP读入一个字节数据，作为CNC指令码，判断其是否为结束指令，若是，则结束，若不是，则进行步骤2；

步骤2：指令解析。根据得到的指令码，查询CNC指令解析表中对应的解析表项，获取解析指令所需的信息。

步骤3：数据传送。根据指令解析获取当前解析指令参数个数np和参数交换区地址pea，将参数搬移到参数交换区。

在搬移时从数控语言专用数据区中紧接指令码位置开始搬移4×np个字节数据到Dstart+pea地址，如图6所示。



图6 数控指令参数搬移过程示意图

步骤4：启动执行。根据启动标志地址ta和CNC指令码，将数控指令标志区中相应位置位，如图7所示。



图7 启动标志设置过程示意图

系统采用M区数据作为启动标志，启动标志地址ta表示相对于M区起始地址的字节偏移量。指令码code作为字节内偏移量得到需要置位的数据，然后将该为设为1（ON），启动对应控制程序执行。

步骤5：等待运行结束。当数控指令对应的运动执行时，线程进入等待状态，等待运动结束。运动结束后，重新进入步骤1循环执行。

上述步骤重复执行直到所有指令解析执行完成。

## PLC控制程序实现方法

控制程序（CP）是技术人员采用IEC61131-3标准语言，包括LD（梯形图），SFC(顺序功能图)，FBD(顺序功能块)，ST（结构化文本）或IL（指令表）开发的应用程序。梯形图作为图形化语言，具有直观易懂的特点，是使用最为广泛PLC编程语言，本文采用梯形图语言进行编程。CP处在系统的中间层次，响应上层数控语言层的请求，并完成对底层算法的调用执行。

CP={I,L,MC,MS},I为初始化程序，完成对数据区、端口和状态等的初始化操作；L为逻辑控制程序；MC为运动控制相关的程序，MC={mc1,mc2,…,mcm},mci用来实现第i条数控指令；MS用来控制MC执行的标志集合，MS={ms1,ms2,…,msm}，msi用来触发mci的执行，如图8所示。任意msi都包括ON和OFF两个状态，当状态从OFF跳变为ON时，对应的mci将执行。可以使用M区数据作为控制标志，MC中每个程序段都以M区中一位作为常开触点控制运行。

图 8 用标记控制程序模块执行示意图

mci实现参数搬移、条件判断和底层算法调用。mci最终将调用EA的子集EAj运行，运行前搬移参数到为EADj，并设置EASj的T为ON。

除了逻辑控制外，控制程序中的运动控制功能通过循环方式被线程调用执行，包含以下几个步骤：

步骤1：遍历CNCTD，判断是否有挂起需要执行的数控指令；若有转步骤2，否则本次循环结束；

步骤2：将CNCPSD数据按照约定格式搬移到EAPD；

步骤3：设置EAFD相应位，启动引擎算法执行，同时清除CNCTD中挂起的标志位；

步骤4：每次循环都读取EAFD中标记算法运行状态的位数据，等待引擎算法执行结束标志。当结束后，进入步骤1，开始下一次循环。

## 多线程协同运行

多个线程协同完成控制任务，线程在系统启动时创建并被调度执行。通过信号量T1互斥和同步CNC线程及Control线程，通过信号量T2互斥和同步Control线程和Algorithm线程。

CNC线程包含以下几个步骤：步骤1：指令读取（p1）,步骤2：指令解析(p2)，步骤3：获取执行令牌（ token） T1，步骤4：数据传送（p4）,步骤5：启动执行(p5)。

Control Thread包含以下几个步骤：步骤1：逻辑控制（q1），步骤2:遍历数控指令标志区，判断是否有需要执行的数控指令(q2)；步骤3：获取执行令牌T1，T2(q3)，步骤4：将参数交换区数据搬移到引擎算法参数区(q4),步骤5：启动引擎算法执行，交还T1(q5)。

Algorithm Thread包含以下几个步骤：步骤1：遍历算法执行标志区，判断是否有算法需要执行（s1）；步骤2：获取令牌T2，步骤3：启动算法执行（s3）；步骤4：等待执行机构执行结束，结束后交还令牌T2(s4)。

图9 三线程协同工作示意图

CNC Thread 首先进入指令读取（p1），当有需要处理的数控指令时，对指令进行解析（p2），然后尝试获取令牌T1（p3），一旦获取成功后，根据解析得到的信息完成数据搬移（p4）并将CNCTD中和指令对应的控制位置为ON（p5），并将令牌T1交给Control Thread,然后进入下一次循环。若无法获取信号量T1，则一直尝试获取，直到成功为止。

在Control Thread中除了完成逻辑控制（q1）外，不断遍历CNCTD区（q2），判断是否有需要执行的数控指令，若有则将尝试获取令牌T1，T2，当获取令牌T1，T2后，将参数从CNCPSD搬到EAPD（q4），然后将EAFD中和需调用算法对应的位设为ON,启动算法的执行。并将令牌T1交还给CNC Thread，令牌T2交给Algorithm Thread。当有挂起的需处理的数控指令而没有获取需要的令牌T1或T2时，Control Thread一直重复执行q1和q3，直到获取令牌T1和T2。

在Algorithm Thread中，首先一直遍历EAFD判断是否有算法执行（s1），当有需要执行的算法则尝试获取令牌T2（s2）。当获取令牌T2后，启动算法执行（s3），向执行机构发出运行信号，并一直等待执行机构运行结束（s4），当运行机构结束后，交还令牌T2。



图10 多线程间数据共享和互斥访问机制

线程间的数据共享是通过共享内存实现的，为了保护数据和保证一致性，在同一个时刻必须避免不同线程对同一个数据进行操作。由于在我们系统中采用了分层的设计结构，只有相邻两层的线程间会操作相同的数据区，而且在交互时参数都采用了上层写，下层读的方式。在我们系统中，通过令牌T1实现对CNCPSD数据区的互斥操作，CNC Thread只有在获取令牌T1后，才能写CNCPSD，同样Control Thread线程只有获取令牌T1后才能读CNCPSD。类似的，Control Thread 和Algorithm Thread通过令牌T2实现了对EAPD数据区的互斥操作。

## 实例分析

### 7.1实验设备 介绍

本文采用全自动绕线机作为实验设备，如图7.1，该绕线机采用5轴协同工作，外加12个气泵完成自动完成排线、缠脚、剪线、装卸骨架等功能。



图7.1绕线机外观

绕线机5个运动轴分别为X、Y、Z、U、Q轴。其中X、Y、Z轴负责排线进给和缠线操作，执行部件为伺服电机；U轴为主轴，负责缠绕，执行部件为伺服电机；Q轴负责控制绕线角度，执行部件为步进电机。12个气泵，每一个对应一个PLC的输出控制，并由气压驱动。

绕线机控制器采用我们自主研发的带运动控制的嵌入式PLC CASS-PLCA149B，如图7.2所示。控制器采用了双处理器结构，由两颗STM32 CortexM3芯片构成PLC处理系统，有32个输入点、32个输出点和6个电机接口。电机由主从处理器协同完成控制，CNC程序、控制程序和运动控制算法运行在主处理器，主处理器根据要完成的运动向从处理器发出运动参数，并启动从处理器。从处理器根据运行参数发出PWM脉冲信号、电机使能信号，驱动电机执行。

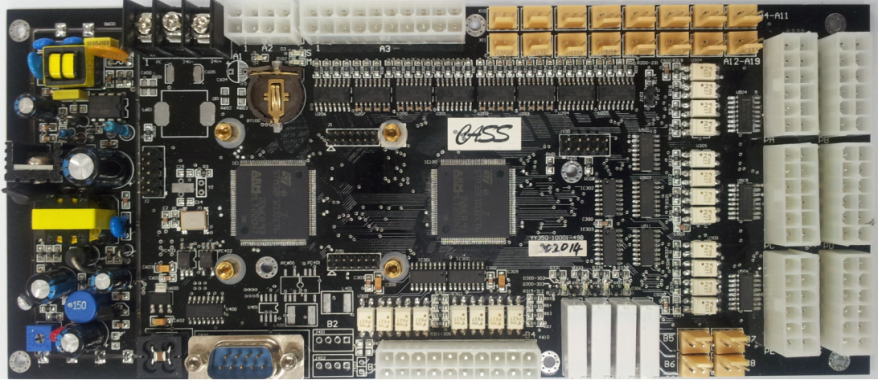


图7.2 CASS-PLCA149B

### 7.2绕线机动作分析

本文主要目的是运动控制，因此这里只考虑由电机完成的运动，不包括气泵完成的运动。绕线机在完成绕制线圈的过程中包括以下几个动作：

1. 复位：X轴，Y轴，Z轴，U轴，Q轴依次通过Z相定位的方式，找到电机Z点，然后各轴走指定距离作为各轴的原点，准备开始绕线；
2. 上料：绕线开始时，通过气泵的组合运动，将骨架从料台送到U轴上固定。
3. 退料：绕线结束时，通过气泵的组合运动，将骨架从U轴上卸到料台。
4. 直线进给：电机分别驱动X轴，Y轴，Z轴，使导针（引导铜线）到达指定位置。
5. 缠脚：电机驱动X轴，Y轴，Z轴三轴联动，X轴，Y轴以指定半径做画圆运动，同时Z轴按指定距离做提升或下降运动。用于将铜线固定在骨架的引脚上，以准备下一个动作，如断线等。
6. 断线：在铜线缠住引脚后，气泵驱动夹爪左移或前进或后退，将铜线扯断。
7. 退线：夹爪下旋，张开，退线气泵前推，将断线后的废线退到废料区，然后退线气泵复位。
8. 缠线：电机驱动X轴，U轴两轴做主从运动，其中U轴为主轴，X轴为从轴。U轴带动骨架高速旋转的同时，X轴跟随U轴做来回运动，从而将铜线按指定参数缠绕在骨架上。

绕线机工作过程如下：

复位=>料台送料=>缠端脚1=>夹爪断线1=>夹爪退废线=>主轴启动=>绕制(排线)=>主轴停止=>缠端脚2=>夹爪断线2=>料台退料。

### CNC指令设计和程序编写

本文在对绕线机动作分析的基础上，定义了如表7.1所示的指令，指令使用了类似G代码的形式。

表7.1 CNC指令定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 指令码 | 指令标志 | 指令格式 | 参数个数 | 参数介绍 | 指令功能 |
| 01 | M2000 | G00 X\_Y\_Z\_S\_ | 4 | X:X轴距离Y：Y轴距离Z：Z轴距离S：速度 | X,Y,Z轴快速定位 |
| 02 | M2001 | G00 U\_ S\_ | 2 | U：U轴距离 S：速度 | U轴快速定位 |
| 03 | M2002 | G00 Q\_ S\_ | 2 | Q：Q轴距离 S：速度 | Q轴快速定位 |
| 04 | M2003 | G00CW X\_N\_D\_S\_ | 4 | X：X轴距离N：圈数D：线径S：速度 | U轴顺时针绕线 |
| 05 | M2004 | G00CCW X\_N\_D\_S\_ | 4 | 同上 | U轴逆时针绕线 |
| 06 | M2005 | G02CW R\_N\_Z\_S\_ | 4 | R：画圆半径N：圈数Z：Z轴距离S：速度 | 顺时针绕端脚 |
| 07 | M2006 | G02CCW R\_N\_Z\_S\_ | 4 | 同上 | 逆时针绕端脚 |
| 08 | M2007 | G04 T\_ | 1 | T：延时时间 | 暂停/延时dwell |
| 09 | M2010 | G90 | 0 |  | 绝对坐标 |
| 10 | M2011 | G91 | 0 |  | 增量坐标 |
| 11 | M2012 | M00 \_ | 1 |  | 夹爪输出 |
| 12 | M2013 | M01 \_ | 1 |  | 送料退料 |
| 13 | M2014 | M02 \_ | 1 |  | 退线 |

然后在数控程序编辑环境ComEditorX中完成数控语言程序的编写、调试、编译和下载，如图7.3所示，图7.4为绕线机CNC程序段实例。



图 7.3 CNC指令编译器

|  |  |
| --- | --- |
| %绕线机数控程序代码  M1 N0 %送料  M1 N1 %送料返回  G00 X12.00 Y33.500 Z24.000 S50  G02CCW R1.8 N0.003 Z-0.600 S25  M0 N3  M0 N4  G00 X12.00 Y35.500 Z22.000 S50  G00 X11.00 Y36.500 Z22.000 S50  M0 N1  M2 N0  M2 N1  M0 N2  G00U U360 S0.8  G00 X11.00 Y36.500 Z16.000 S50 | G00 X11.20 Y28.000 Z16.000 S50  G00CW X18.500 N0.88 D0.076 S20  G00CW X11.200 N0.88 D0.079 S20  G00 X12.00 Y22.000 Z24.000 S50  G02CCW R1.8 N0.003 Z-0.600 S25  G00 X12.00 Y16.000 Z23.400 S50  G00 X5.00 Y-2.00 Z28.000 S50  M0 N1  G00 X8.00 Y-2.00 Z28.000 S50  G00 X8.00 Y-2.00 Z32.000 S50  M0 N2  G00 X0.00 Y0.00 Z0.000 S50  M0 N3  M0 N4  FINISH %程序结束 |

图7.4 绕线机CNC程序段实例

### 梯形图实现

在控制程序设计环境中用梯形图实现每条CNC指令的基本功能，如图7.5所示，编写后的程序编译后下载到PLC。

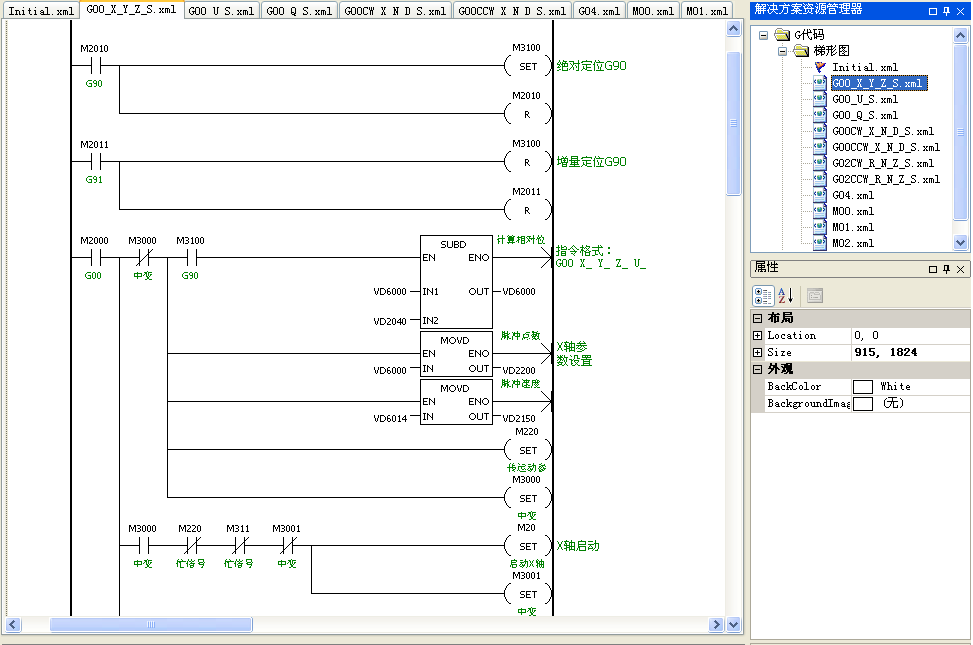


图7.5 控制程序梯形图设计环境

### 7.6引擎实现

引擎分为主引擎和从引擎，分别存储在CASS-PLCA149B的主控制器和从控制器中。主引擎用于运动控制算法实现以及CNC和控制程序解析执行，从引擎根据主引擎传递的参数以及控制命令实现PWM脉冲输出。如图7.6中代码为在引擎中扫描是否需要执行画圆算法。当需要执行算法时，调用对应的函数执行，如图7.6中的DrawCircleFirst()。

|  |
| --- |
| //程序是否有画圆命令M160  if((CassMem[MStart + DRAW\_CIRCLE\_M] & Circle\_Command) != 0){  if((PWMFlag[\*CircleX] & FLAG\_CIRCLE) == 0){  DrawCircleFist(); //画圆初始化  SendPending |= (1 << ID\_DrawCircle) ; //发送画圆参数到从MCU  }  } |

图7.6 引擎中扫描代码示例

### 7.7效果展示

最后通过联机调试，在给定骨架和线径的前提下，配合手操器取得每个运动的最优参数并优化CNC数控程序，下载到PLC执行，图7.7为绕制的电感效果图。若改变骨架尺寸和线径后，或改变固定脚位置，只需调整CNC程序参数就可实现，无需改变控制程序和引擎算法。

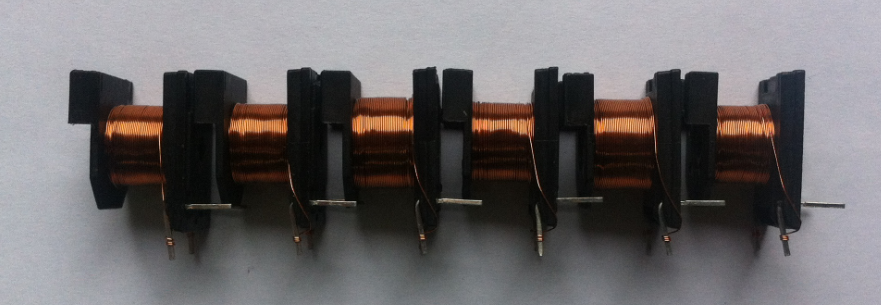


图7.7绕线效果图

## 结论

本文实现了一种在嵌入式PLC中嵌入CNC的方法，通过层次化开发结构和CNC程序，能将PLC程序开发分解为多个相对独立的部分，简化PLC程序设计。并通过CNC能将功能开发和工艺开发分离，为用户提供了一种在不改变PLC功能的前提下改变控制流程和参数的手段。

## 参考文献

[0] Salah Abdallah, Riyad Abu-Mallouh. Heating systems with PLC and frequency control. Energy Conversion and Management 49 (2008) 3356–3361.

[1] Ramazan Bayindir , Yucel Cetinceviz. A water pumping control system with a programmable logic controller (PLC) and industrial wireless modules for industrial plants—An experimental setup. ISA Transactions 50 (2011) 321–328.

[2] Advanced control algorithms embedded in a programmable logic controller，Samo Gerkšicˇ, Gregor Dolanc, Damir Vrancˇ ic´, etc. Control Engineering Practice 14 (2006) 935–948.

[3]Marek ´ Snie˙zek, Josef von Stackelberg. A fail safe programmable logic controller. Annual Reviews in Control 27 (2003) 63–72.

[4] Gregor Kandare, Stanislav Strmcˇnik, Giovanni Godena. Domain specific model-based development of software for programmable logic controllers. Computers in Industry 61 (2010) 419–431.

[5] Seo Ryong Koo, Poong Hyun Seong. Software design specification and analysis technique (SDSAT) for the development of safety-critical systems based on a programmable logic controller (PLC). Reliability Engineering and System Safety 91 (2006) 648–664.

[6] Inmaculada Plaza, Carlos Medrano. Exceptions in a Programmable Logic Controller implementation based on ADA. Computers in Industry 58 (2007) 347–354

[20] X.W. Xu, S.T. Newman.Making CNC machine tools more open, interoperable and intelligent—a review of the technologies.Computers in Industry 57 (2006) 141–152

[21] .Sanjeev Kumar, Aydin Nassehi, Stephen T. Newman etc. Process control in CNC manufacturing for discrete components: A STEP-NC compliant framework. Robotics and Computer-Integrated Manufacturing 23 (2007) 667–676.

[22]. M. Minhat , V.Vyatkin , X.Xua etc. A novel open CNC architecture based on STEP-NC data model and IEC 61499 function blocks. Robotics and Computer-Integrated Manufacturing 25 (2009) 560– 569.

[23] Pengfei Li , Tao Gao,Jianping Wang,etc. Open architecture of CNC system research based on CAD graph-driven technology. Robotics and Computer-Integrated Manufacturing 26 (2010) 720–724

[24] Karl-Heinz John, Michael Tiegelkamp. Programming industrial automation systems IEC 61131-3, Berlin: Springer-Verlag, 2001.

[25] John K. H, Tiegelkamp M. IEC 61131-3 International Standard, Programmable controllers. Part 3: Programming Languages, 2003.

[26] C. Sünder, A. Zoitl, F. Mehofer, et al. Advance use of PLCopen motion control library for autonomous servo drives in IEC61499 based automation and control systems. E&I Elektrotechnik und Informationstechnik, 2006, 123(5): 191-196.

[27] Nathan Massey, Sales Engineer, B&R Industrial Automation Corp. PLCopen Motion Control: Reducing development time and cost with standardized motion programming.2008.

[29] PLCopen Technical Committee 2: Function blocks for motion control version 1.1.PLCopen, Zaltbommel, The Netherlands,www.plcopen.org/. 2005.

[30] Johnson D. PLCs maximize machine, motion control [J], CONTROL ENGINEERING. 2005: 51-56.

[31] ISO 6983/1：Numerical control of machines – Program format and definition of address words –Part 1: Data format for positioning, line motion and contouring control systems [S]. First edition.1982.

[32] Programmable Controller, SYSMAC CJ-series, Motion Control Unit, CJ1W-MCH71, OPERATION MANUAL, OMRON, 2004.

[33] Programmable Controller, SYSMAC CS-series, Motion Control Unit, CS1W- MCH71, OPERATION MANUAL, OMRON, 2004.

[34] OMRON Co.Ltd CS1W-MC221(-V1)/MC421(-V1) Motion Control Units. Operation Mannual 2004.

[35] Mamlook, Rustom; Nijmeh, Salem; Abdallah, Salah M. A programmable logic controller to control two axis sun tracking system. Information Technology Journal, v5, n6,p:1083-1087.(2006).

[36] EI Din, Ashraf Salah EI Din Zien.PLC-based speed control of DC motor. Conference Proceedings – IPEMC 2006: CES/IEEE 5th Intenational Power Electronics and Motion Control Conference, v2, p848-852. (2007)

[37] Ioannides, Maria G. Design and implementation of PLC-based monitoring control system for induction motor. IEEE Transactions on Energy Conversion, v19, n3, p469-476. (2004).

[38] Syaichu-Rohman A, Sirius R. Model Predictive Control Implementation on a Programmable Logic Controller for DC Motor Speed Control. 2011 International Conference on Electrical Engineering and Informatics (ICEEI), Bandung, Indonesia. 2011.

[39] Nowlin, Brent C. A programmable system for motion control. Proceedings of the International Instrumentation Symposium, v49, p87-103, (2003).

[40] Aoki, Takeshi; Hahiyama, Tomonori; Zanma, Tadanao; Suzuki, Tatsuya; Okuma, Shigeru. Automatic generation of motion sequence in programmabel logic control. International Workshop on Advanced Motion Control, AMC, p448-452, (2000).

[41] Backman, Michael. Choosing motion control components. Medical Device and Diagnostic Industry, v21, n1, Jan 1999.

[42] Labs, Wayne. From process sensors to redundant PLC and motion. Control Solutions International, v76, n10, p52. (2003).

[43] Panasonic Co.Ltd, Programmable controller FP2 Positioning Unit Mannual. 2011.

[44] Panasonic Co.Ltd, Programmable controller FP2 Positioning Unit (Interpolation) Technical Mannual. 2011.

[45] Wagner R. Standard for programmable logic controllers. Elektronik. v55, n26, p44-49, 2006. （SCI收录）

[46] Panasonic Co.Ltd, Programmable controller FP Series Programming Mannual. 2011.

[47] Liu Ya-dong, Li Cong-xin, Wang De-cai, et al. Position control with programmable logic controller. Journal of Shanghai Jiaotong University. v36, n4, p491-3, 2002. （SCI收录）

[48] Li Qin, Zhang Xian-Min, Kuang Yong-cong, et al. Researching on motion control system of automatic screen printer. Microcomputer Information. n31, p1-3.2008. （SCI收录）

[49] Costea C, Silaghi H, Silaghi M, et al. Mill Speed Control Using Programmable Logic Controllers.12th WSEAS International Conference on Mathematical Methods and Computational Techniques in electrical Engineering(MMACCTEE ‘10),p26-30. Timisoara, Romania, 2010.

[50] Sun Cheng-zhi, Xiong Tian-zhong. Teaching and Playback in Two Axis Motion Control Based on PLC. Modular Machine Tool & Automatic Manufacturing Technique. n3, p65-7. 2010. （SCI收录）

[51] An Hongwei, Guan Fengnian. Application of PLC Position Control System in Scroll Cutting Machine. International Conference on Information Electronic and Computer Science. Vol1-3, p 468-471. 2010. （SCI收录）

[52] Jiang Wei-hua, Yan Dong. Design of Position System of Graphite Spraying Machine Based on PLC Controlling. Modular Machine Tool & Automatic Manufacturing Technique. n9, p52-55.2001. （SCI收录）

[53] Dylis W, Owczarski T. Positioning servo-drives with programmable logic controller. Prace Instytutu Elektrotechniki, vol(48),n(209),p41-49,2001. （SCI收录）