图形组态控制策略的识别方法研究*

李杰龙
(杭州电子科技大学 软件与智能技术研究所,浙江 杭州 310018)

摘要:针对常用几种图形组态控制策略识别方法在实际控制中存在逻辑执行顺序错乱的问题,提出了一种基于双堆栈的控制策略识别方法。从功能模块间的输入输出连接关系出发,分别分析了单输入输出、多输入输出、空输入输出、多条回路的处理方法,对于同一个周期被多次调用的功能模块,采取了中间变量分解的方法,解决了调用操作的重复性问题;在确定好控制策略的逻辑执行顺序后,利用栈存储分析了模块间数据的输入输出过程,很好地符合了组态数据流的走向;最后在 CASS 控制策略生成平台上对该识别方法进行了验证,编译成功后生成了相应的组态 IL 指令,并且通过代码自动生成过程生成了可用于下位设备的组态算法包。研究结果表明,控制算法生成的函数执行顺序与实际分析的顺序一致,证明了该方法的可行性。

关键词:图形组态软件;控制策略;堆栈;识别方法
中图分类号:TP311;TH39 **文献标志码**:A **文章编号**:1001 - 4551(2012)07 - 0850 - 06

Identification method for control strategy in graphics configuration software

LI Jie-long
(Institute of Software and Intelligent Technology, Hangzhou Dianzi University, Hangzhou 310018, China)

Abstract: In order to deal with the problems that the confusion of the logic execute order in practical control caused by some common configuration recognition methods, a new method for control configuration recognition based on double-stack was proposed. The manage approach for single input and output, multi inputs and outputs, empty input and output, multi control loop tanks were analyzed from the connection relationships between the function model's inputs or outputs. For the function models that would be called for many times in a cycle, intermediate variables were used to resolve the problem of the repeat calling operation. After determining the logic execute order of the configuration recognition, the characteristics of the stack were used to analyze the input and output process of the model data, which was accord with the running trend of the configuration data. At last, the method was proved by the control strategies in the computer assist special system (CASS), the corresponding configuration IL instructions was generated after the successful compiling, then the configuration algorithm packages used in the devices were generated through the auto-generation mechanism. The results show that the executable configuration instruction which is generated by the control strategy after the compilation of the platform is conform to the order analyzed by the identification method, which demonstrates the feasibility of the method.

Key words: graphics configuration software; control strategy; stack; identification method

0 引 言

随着计算机控制技术和嵌入式技术的快速发展,如何有效地将嵌入式组态软件运用到工业控制中已成为该领域最新研究动向。组态软件是完成数据采集与过程控制的专用软件。通过组态策略软件就可以实现预定的控制方案和策略。目前,实现嵌入式控制系统软件组态的方式主要有功能框填表式、组态语言式和图形式 3 种[1-3]。功能框填表组态法跟组态语言虽也能完成控制组态,但都存在不直观、使用不方便的缺点,用户使用起来非常不灵活。图形组态式由此应运而生,它借助形象易懂的功能块框图及方便、快捷的连接操作,能快速高效地生成一套控制算法,不仅具有接口直观、使用方便、组态灵活的优点,而且是一种可视

收稿日期:2012 - 02 - 06
基金项目:浙江省重点科技创新团队资助项目(2010R50008)
作者简介:李杰龙(1987 -),男,浙江宁海人,主要从事智能控制与嵌入式方面的研究. E-mail:lijielong1130@163.com

化的组态方式,目前这种图形组态的方式得到了广泛的应用。

文献[4]中,大连理工大学设计的组态软件 PLC_config 考虑到工程应用的惯例,采用了 PLC 的常规执行顺序,即按从左到右、从上到下的顺序依次完成;文献[5]中,研究者利用分布式智能系统策略组态软件 FIPCONFG 提出了一种类似于“电流”的“信号流”概念和基于“信号流”的功能块图连接定序规则;文献[6]中,王锦标教授提出了一种递归式回路识别方法,为了解决有的参数可能会在一个控制周期内计算多次,引入了“同步时钟”概念。该方法能够在回路识别的就可以同时完成回路中功能模块的执行。

本研究致力于探讨一种既可简化组态又可正确执行的控制策略识别方法。

1　图形组态控制策略常用识别方法

图形组态法采用图解连接的方法来描述控制策略,控制策略中的执行顺序对使用者而言是可视的,根据直观印象就能判断出不同的控制算法,如 PID 控制、串级控制、前馈控制等;但对计算机而言,必须人为设定有效的识别方法,通过读取形成的一个组态文件来识别。目前常用的控制策略识别方法有位置式和模块编号式两种[7-8]。

1.1　位置式识别方法

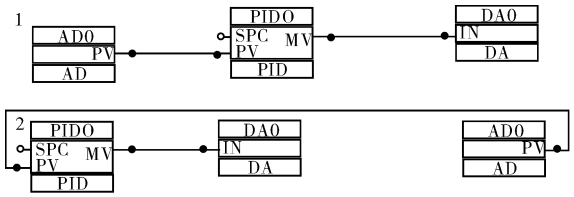
位置式识别方法是根据控制策略中的功能模块在组态画面上的摆放位置来识别,一般按照从左到右、从上到下的位置顺序执行,如图 1(a)所示。

使用该方法的组态软件一般把组态编辑区分成若干个区域,模块只能摆放在这些区域中,且每个区域只能摆放一个模块,用户用连线将各个模块连接形成控制策略,软件按照从上到下、从左到右的次序扫描各个区域,每扫描到一个非空区域(区域内有模块),就执行该区域内的模块。以图 1(a)中所示的控制组态为例,当用户按策略 1 的顺序摆放模块时,模块按 AD->PID->DA 的顺序执行,执行的次序合理;但是当用户按照策略 2 的顺序摆放模块时,虽然模块之间的连线关系与策略 1 相同,但是模块摆放位置却不同,其后果是按照 PID->DA->AD 的顺序执行,显然是不合理的。

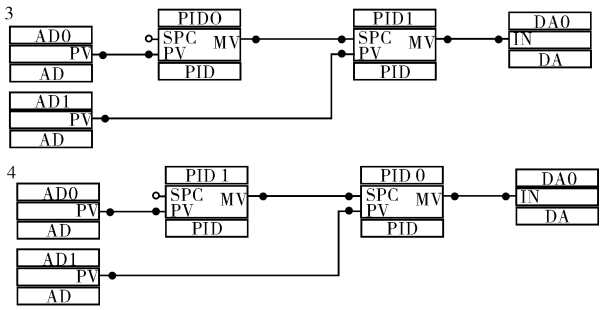
1.2　模块编号式识别方法

模块编号式识别方法是给各类基本模块设定一个优先级,一般按照“输入类模块->运算类模块->控制类模块->输出类模块”的从高到低的优先级顺序执行,如图 1(b)所示。对处于同一优先级的功能模块,由用户给这些模块编号,一般按照模块编号由小到大的顺序执行。最后整个控制策略结合模块的类型和编号的大小来顺序执行。

以图 1(b)中所示的串级控制组态为例,策略 3 和策略 4 的连线关系相同但模块编号略有不同。在策略 3 中,主调 PID 编号为 0,副调 PID 编号为 1,执行顺序为先执行主调 PID0,后执行副调 PID1,执行顺序合理。然而在策略 4 中,主调 PID 编号为 1,副调 PID 编号为 0,而执行顺序是按照编号由小到大的次序执行,因此执行顺序为先执行副调 PID0,后执行主调 PID1,执行顺序不合乎逻辑。



(a) 位置式识别法



(b) 模块编号式识别法

图 1　常用识别方法示例

由以上分析可知,排序规则的建立是连接定序操作的关键。位置式和编号式识别法都有各自很大的缺陷。虽然这些定序规则的原理简单易懂,实现方便,能降低组态编辑器在设计开发上的难度,缩短了开发周期,但无论在逻辑上还是在用户设计的意图上,都是不严谨、不科学的,不能正确地反映实际控制意图。为了解决上述两种控制策略识别法中模块执行顺序不合理甚至错乱的问题,本研究提出一种基于双堆栈的控制策略识别法。通过利用该方法,组态软件不需要通过模块序号或摆放位置来确定策略中的执行次序,能根据策略中模块的连接关系自动确定正确的执行顺序,克服上述两种识别方法的缺点。

2　基于双堆栈的识别方法

2.1　基于双堆栈的功能块逻辑顺序识别

控制策略图形组态编辑器以连线的方式连接各功能模块的数据输入输出端口。为了正确地表示模块间的数据流向,编辑器必须遵循以下规则:

(1)编辑器中的功能模块不能重名,保证每个设计器都有唯一的名称,这样在控制策略解析时可以通

过名称来唯一识别功能模块。

(2)一个功能模块的数据输出可以输出到多个功能模块的数据输入上,但一个功能模块的数据输入只能来自一个数据输出,即输入来源的唯一性。

(3)功能模块的数据输出不能直接来自另一个模块的数据输出,需要严格遵循输出到输入的连接关系。

本研究的功能块图排序采用倒序的方式,即以输出模块为起点,依次向前遍历。以下根据输入输出供给关系,分几种情况展开讨论:

(1)单输入输出。即一个输出变量有且仅输出给一个输入变量,其排序算法步骤如下(例子如图 2 所示):

① 一条完整的控制策略回路必须有输出模块,输出模块的特点是只有输入端口而无输出端口。根据这个特点在控件集中找到输出模块,然后建立两个堆栈,分别称之为主堆栈和从堆栈。本研究将找到的输出模块放入其中的从堆栈中。

②再从堆栈中弹出一个输出模块,并将该输出模块压入主堆栈中。主堆栈中的栈顶模块即为当前模块。

③分析主堆栈中当前模块的输入端口,如当前结点为输入模块,则进入第④步;如不是输入模块,则分析其输入线段,按输入端口从上到下的顺序将输入模块依次压入从堆栈中,直到取完该模块的所有输入模块为止。

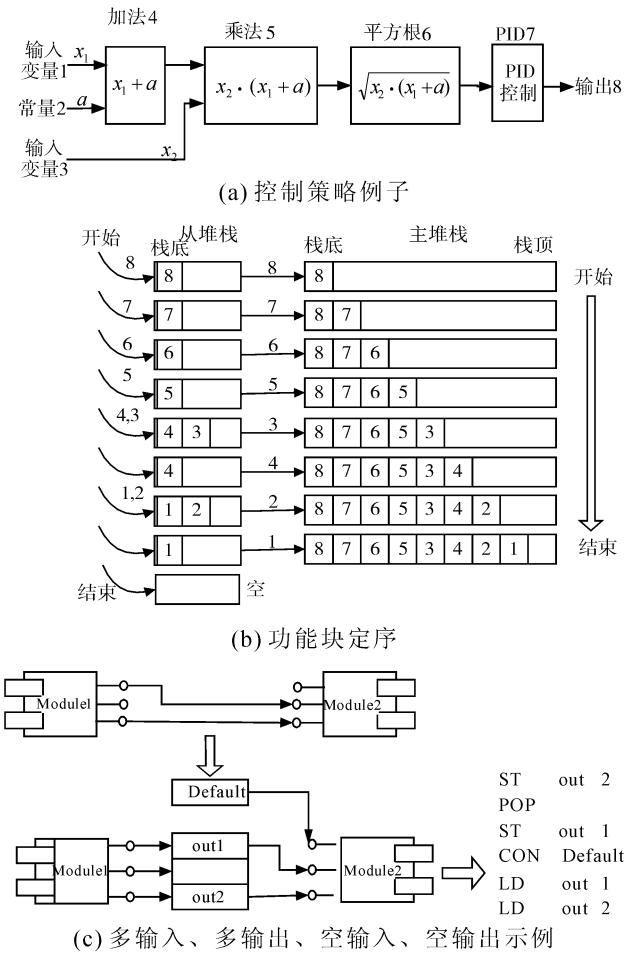
④检查从堆栈中是否还有模块,没有则搜索结束,最后得到的子堆栈就是组态算法序列;有则继续弹出模块,压入主堆栈中,并设为当前模块,跳转至第③步。

一个控制程序的组态图如图 2(a)所示,本研究以该图为例进行功能块定序,双堆栈定序的过程如图 2(b)所示,最后主堆栈中从栈顶到栈底的序号便是控制策略功能块的执行顺序。

(2)多输入与多输出。当一个模块的输出变量被多个输入同时使用时,则称该模块出现断层。在进行控件遍历时,本研究首先记录下出现断层的模块,放入临时变量中。这些模块的输出用中间变量的输入输出(LD,ST)来分解。一个模块本身有多个输出时,同样也是使用中间变量方式来分解。由于本研究使用的数据结构是栈结构,栈具有先进后出的特点,根据输出与输入的对应连接关系,多输出从下到上 ST,输入根据管脚从上至下 LD,例子如图 2(c)所示。

(3)空输入与空输出。默认值输入(CON n)或空语句(CON NULL),这里 CON 是常量输入的指令。空输出:输出给垃圾变量(ST dummy),例子如图 2(c)所示。

(4)平行关系。多回路控制策略逻辑顺序识别如



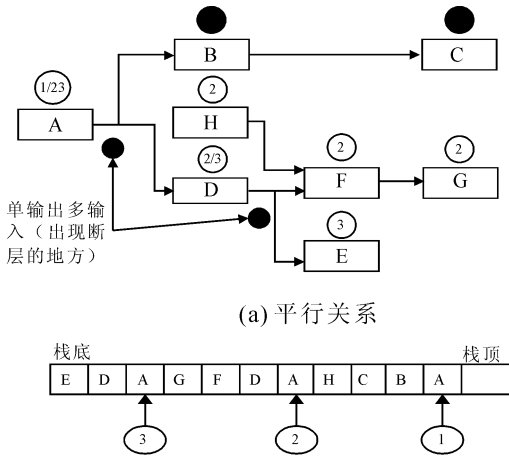
(a) 控制策略例子

(b) 功能块定序

(c) 多输入、多输出、空输入、空输出示例

图 2 单回路控制策略逻辑顺序识别

图 3 所示,当一个策略最后有多个输出变量时,即出现回路的并行关系时,将按照从上到下的顺序处理,如图 3(a)中的 3 条回路,按照 1、2、3 的处理顺序。单条回路的排序过程同上述单输入单输出的双堆栈排序步骤。最后得到的主堆栈里的排列顺序即为功能块逻辑执行顺序。



(a) 平行关系

(b) 主堆栈排序结果

图 3 多回路控制策略逻辑顺序识别

从图 3(b)的排序结果可以发现,出现断层的模块

均被多次调用。如 A 模块被调用了 3 次,D 模块被调用了 2 次。如直接调用重复的功能模块显然带来了很多重复工作。正如上述的出现断层的模块利用中间变量分解来处理,在第 1 次执行完 A 功能模块后,利用中间变量输出指令 ST A_0(A_0 代表 A 模块的输出变量名)输出 A 模块的最后输出值,在第 2 次、第 3 次使用 A 模块时,无需执行,直接使用前面临时存储的输出变量值即可。本研究利用中间变量输入指令 LD A_0,从而得到 A 功能模块的输出值。

2.2　控制策略数据执行分析

一般来说,控制策略编译生成下位机设备使用的中间码数据需包括:指令区数据、输入输出区数据和内含参数区数据。而目前大部分的组态系统都把这些数据存储在实时数据库中,通过对数据库的查找、修改等操作来存取数据[9-10]。然而,下位机设备在控制运行的过程中,只与功能块的输入输出区数据相关,与其他数据无关,因此,目前的这种使用数据库存储数据的方式不可避免地造成了存储空间的浪费,且对数据库的操作耗时,这种方式不能满足下位机设备对系统的资源、实时性方面的严格要求。

针对以上问题,本研究提出使用栈的数据结构来存储功能块图元数据,存取输入输出数据通过入栈出栈操作来实现。该系统进行功能块运算时,只存储相关的变量,避免了繁杂的查找、修改等操作,节省了系统资源,提高了运行效率。这里有如下 3 个规则:

(1)出栈操作表示输入,输入的点数决定弹出栈的数据个数;当上游无输入时,必须有默认值压栈。

(2)入栈操作表示输出,输出的点数决定压入栈的数据个数;当无输出到下游时,也要进行弹出操作,保证后面的取数正确。

(3)数据入栈时,按顺序入栈;数据出栈对变量进行赋值时,则要按反序进行赋值。因此,在控制算法指令实现中,必须按照该顺序进行出、入栈。

以图 2 所示的控制策略为例,本研究在得到功能块的逻辑执行顺序后,便根据该排列顺序进行数据流分析。输出输入存储结构如图 4 所示。由图 4 可知,首先执行 1,2 模块,即将输入变量跟常量压入到堆栈中。这里的输入变量 X_1 可以是中间变量的输入或者是 AD 采样的数据。由于加法功能块的输入点数有两个,本研究需从堆栈中提取出两个数据作为加法模块的输入变量,执行完该功能模块后,将输出变量重新压入此堆栈,以供下一个模块提取当做输入源来使用。依此类推,最后栈里得到的即为控制算法最后的输出结果。

输出输入存储结构如图 4 所示,通过栈的存取数据过程完全符合组态数据流的走向,可以得到正确的控制效果。

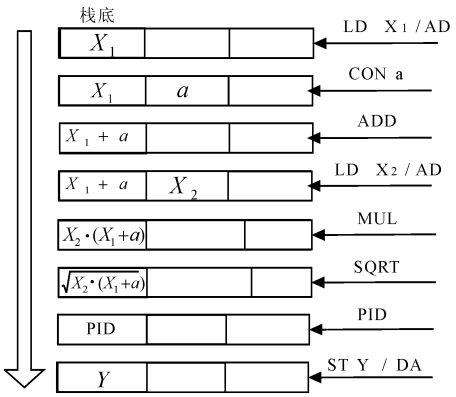


图 4　输出输入存储结构

以上就是对一个控制策略的识别和执行过程,每个控制周期中系统都会对回路完整地识别和执行一次,组态时用户只需要按控制策略原理连接各个模块,而不需要对模块编号或摆放相应位置来确定回路的执行次序。换言之,软件自身能够根据策略中模块的连接关系自动确定正确的执行顺序。

3　识别方法在图形组态软件上的实现

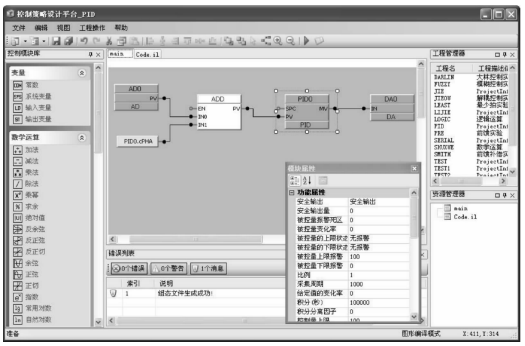
基于上述控制策略的识别方法,本研究借鉴了 .Net Framework的窗体设计器实现机制,本实验室设计了控制策略生成平台。该项目是省重大科技公关项目“基于嵌入式的计算机辅助专用控制系统开发平台的研究(CASS)[11]”的一部分,旨在将该平台生成的控制算法内嵌于 PLC 梯形图平台中,并以梯形图指令的形式来执行控制算法,从而克服 PLC 难以实现复杂过程控制算法的技术缺点。控制策略设计平台包括控件模块库、控制组态编辑区、工程管理器、资源管理器、编译信息区等几个部分,基本界面如图 5(a)所示。

当运行控制策略时,本研究首先分析组态库连线信息,确定各模块的运行次序;然后根据模块的运行次序处理程序,依次调用与其对应的控制算法,同时结合其参数配置信息实现控制策略的运行。因此,功能模块的参数配置是控制策略运行的关键。在本研究控制策略生成平台的实现中,笔者采用功能块组态跟属性设置界面分离的方法,属性设置界面如图 5(a)中小窗口所示,PID 功能模块利用属性设置对话框中配置的参数来实现数据的采集、处理、控制和输出等功能。

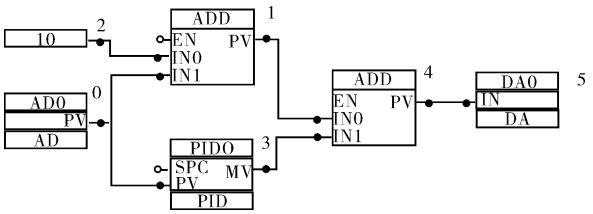
CASS 组态软件的一个控制策略示例如图 5(b)所示,正确完成功能块间的逻辑连线并设置好参数后,便可进行编译,生成相应的组态执行指令。

按照第 2 节的识别方法,本研究对图 5(b)中的控制策略进行分析,可得到主堆栈 2->0->1->0->3->4->5 的功能块逻辑执行顺序。图 5(b)中控制策略编

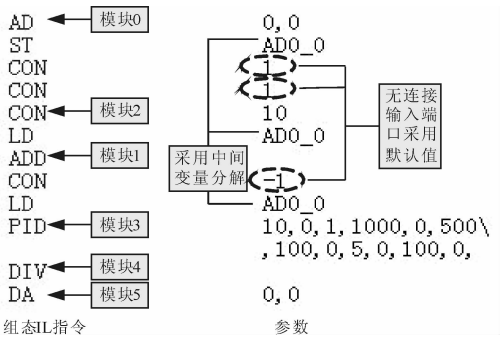
译后对应生成的组态 IL 指令程序如图 5(c)所示。该例中,A/D 模块输出给 ADD 和 PID 模块,出现断层,采用中间变量来分解。在首次执行完 A/D 功能后,本研究利用 ST AD0_0 输出其模块的输出量,暂存于中间变量中,当 ADD 和 PID 执行到 A/D 模块时,都通过 LD AD0_0 从暂存变量中获取。整个过程与识别方法所得到的主堆栈顺序一致,符合控制策略的逻辑走向。



(a) CASS 控制策略组态软件



(b) 组态软件中的控制策略示例



(c) 控制策略编译后生成的 IL 指令

图 5 在 CASS 控制策略组态软件上的实现

在本研究中,笔者采用“XML + XSLT”自动生成技术,即通过描述文件进行匹配模板从而自动生成控制策略的代码文件。组态软件包的自动生成包括 3 大部分:组态算法包、变量的初始化、控制程序块。变量的初始化包括直接初始化与从 Flash 中读出数据进行初始化。组态算法包是组态算法集合的子集,是通过裁剪得到的有用到的算法。控制程序块是用户使用的组态控制。组态算法包是通过控件与内部操作算法进行绑定机制生成的;变量的初始化和控制程序块是通过上位机根据用户设计的组态生成 XML,再与 XSLT 模

```
#include "Configuration_main.h"
// 多输出的中间变量
fp32 configuration13_AD0_0;
// 全局变量
ADStruct configuration13_AD0;
ADDStruct configuration13_ADD0;
DIVStruct configuration13_DIV0;
DAStruct configuration13_DA0;
PIDStruct configuration13_PID0;
// 子函数的向前声明
uint8 Configuration13_mainPage(void);
uint8 Configuration13_mainLoop(void);
// 初始化参数
void Configuration13_InitParam(void)
{
    configuration13_AD0.ucCN = 0;
    configuration13_ADD0.fLastVal = 0;
    configuration13_DIV0.fLastVal = 0;
    configuration13_DA0.ucCN = 0;
    configuration13_PID0.fHY = 0;
    .....//此处省略
    configuration13_PID0.Kp = 1;
    configuration13_PID0.Ts = 1 000;
    // 中间变量初始化
    configuration13_AD0_0 = 0;
}
// 子策略或子块函数
uint8 Configuration13_mainPage(void)
{
    if(! Configuration13_mainLoop()){return 0;}
    return 1;
}
// 子策略或子块函数
uint8 Configuration13_mainLoop(void)
{
    if(! ADControl(&configuration13_AD0)){return 0;}
    if(! STControl(&configuration13_AD0_0)){return 0;}
    if(! CONControl(1)){return 0;}
    if(! CONControl(1)){return 0;}
    if(! CONControl(10)){return 0;}
    if(! LDControl(&configuration13_AD0_0)){return 0;}
    if(! ADDControl(&configuration13_ADD0)){return 0;}
    if(! CONControl(-1)){return 0;}
    if(! LDControl(&configuration13_AD0_0)){return 0;}
    if(! PIDControl(&configuration13_PID0)){return 0;}
    if(! DIVControl(&configuration13_DIV0)){return 0;}
    if(! DAControl(&configuration13_DA0)){return 0;}
    return 1;
}
// 组态控制
uint8 Configuration13_Control()
{
    if(! Configuration13_mainPage()){return 0;}
    return 1;
}
```

图 6 自动生成的控制算法

版进行匹配生成代码。生成的目标代码主要包括头文件、多输出的中间变量声明、全局变量声明、初始化参数方法、子策略方法、主策略方法。

图 5(b)中控制策略对应生成的控制算法如图 6 所示,由图 6 可见:控制策略中的每个功能模块对用的是底层各自函数操作的封装。Configuration13_mainLoop 函数里的各个功能模块函数的执行顺序与实际分析的逻辑顺序一致,能够得到控制算法正确的逻辑执行顺序。上述功能模块函数的正确执行顺序,表明了该方法在实际组态软件中识别控制策略的可行性。

4　结束语

本研究采取了基于双堆栈(并结合中间变量)的控制策略识别方法,克服了传统识别方法的缺陷。同时采用了栈作为数据结构来处理输入输出数据,改善了传统组态软件采用实时数据库作为存储方式带来的耗时、资源浪费等问题。本研究最后简单介绍了上述识别方法在 CASS 控制策略生成平台上的实际实现方式,验证了其可行性,为以后新型组态软件的开发过程提供了一定的指导作用。至于 CASS 控制策略生成平台的详细开发过程,限于篇幅,本研究不再述及。

参考文献(References):

[1] LIN Yong-jun, ZHU Xiao-lin. Design and research of the intelligent front-end based on the DCS configuration software [C]//International Conference on Web Information Systems and Mining, 2010:425-428.

[2] 张　杰, 关　永, 孙继平. 工控组态软件的可视化[J]. 中国图象图形学报, 2002,7(2):201-204.

[3] 麻建华. 集散控制系统组态软件的设计[D]. 成都:电子科技大学计算机学院, 2007.

[4] 王慧丽. 功能块编程技术研究与组态软件设计[D]. 大连:大连理工大学控制科学与工程学院, 2008.

[5] 梁　庚. 基于 IEC6131-3 标准的分布式智能系统策略组态软件包开发研究[D]. 北京:华北电力大学自动化学院, 2004.

[6] 王锦标. 图形组态软件中的递归式回路识别方法[J]. 控制理论应用, 2004,25(11):168-171.

[7] 宋伯生. PLC 编程理论·算法及技巧[M]. 北京: 机械工业出版社,2005.

[8] LIANG Geng. Research on a control strategy configuration platform for field intelligent control network [C]//The Eighth International Conference on Electronic Measurement and Instruments. Xian:[s. n.], 2007:126-130.

[9] YANG Ya-luo, LI Ming, HUANG Ya-yu. The Use of Configuration Conceptionin Software Development [C]// IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Appliacation. Wuhan:[s. n.],2008:963-967.

[10] 罗　川. 基于 CASS 数控绕线机非线性算法的设计与实现[D]. 杭州:杭州电子科技大学计算机学院, 2010.

[11] 严　义, 朱旭燕. 基于 DSP 的 PLC 运动功能的研究[J]. 机电工程, 2011, 28(7): 818-822.

[编辑:李　辉]