A User-oriented Development Method with Multiprocessor Embeded PLCs applied in Complex Logic Control and Motion Control Mixed Scenario

摘要—PLC应用广泛，随着定制化需求的增加，大型装备程序朝着逻辑复杂化，运动控制功能增多方向发展，嵌入式PLC的应用遇到瓶颈而基于PC的PLC价格高，系统复杂，限制了大型装备的发展。为此，本文提出一种在多处理器嵌入式PLC上用户导向的开发方式。兼顾嵌入式PLC的低功耗，低价格特点同时通过多处理器方案来增强其性能；通过对基于图形化构件的多语言支持来提高开发者的适应性；通过用户导向式线程设计，LPM数据交换机制，以及处理器状态转换机制优化ePLC的系统结构。实验部分描述了用用户导向开发方式，实现多处理器ePLC在注塑机上的应用，对比了该结构控制系统和原有基于PC的控制系统的方案。分析得出，所提出的方案在机器性能关键指标和现有控制系统方案相当情况下，成本得到降低，启动速度提高了20倍，且支持分布式控制和人机界面分离。

***Index Terms*—Multiprocessor, motion control, Injection Molding Machine, embedded PLC, User-oriented**

I. Introduction

Some concepts, such as smart factory, intelligent manufacturing [1], [2], and some technologies, such as Internet of Things, 5G, augment reality [3], [4]，正驱动着第四次工业革命.在一个典型的制造业工厂中，包含了大量大型设备，包括起重机，数控加工中心，注塑机，气泵，冷水机，机械手等，而几乎所有大型设备都用PLC实现控制。PLC在工厂自动化中已经成为最主要的控制方式。很多学者对PLC进行了研究，来拓展PLC的应用领域。 [5]–[7]通过对PLC程序的验证来确保PLC程序的可靠性。 [8]–[10] 通过先进的控制算法提高PLC的性能, [11], [12] 通过改变PLC的软硬件的架构来提高PLC的性能和降低开发难度, [13], [14]基于代理的方式通过对整个生产环节传感信息的变化来实现PLC程序的动态更新。随着这些研究的深入，应用领域的扩展和定制化需求的不断增长，PLC的应用正朝着编程方式上以用户为中心，程序上逻辑复杂化，运动控制功能增多方向发展 [15]。数控加工中心，注塑机，压铸机等大型装备的PLC应用是这个方向的典型。

A. Motivations

目前PLC的硬件架构具有两个方向：基于嵌入式芯片的PLC，基于PC的PLC。基于PC的PLC因为功能性强，有更多用户导向的工具支持，所以在复杂逻辑和运动控制上应用开始增多 [15]。以注塑机行业为例，表I列出了注塑机系统的所有模块，数字量输入输出和模拟量输入输出，其中各模块可能存在多组情况。通常一个最小注塑机系统，具有10个模块，20组输入，30组输出，3组模拟量输出，7组模拟量输入，而系统不但对每个模块的控制算法提出了很高的要求，且每个模块、输入输出、模拟量输入输出之间都存在先后逻辑顺序关系，使得在逻辑上和运动控制算法上都变得极其复杂。列表II对比KEBA，BECKHOFF，Gafran，Teckmation 四家主要注塑机PLC控制器厂商的系统情况。可以看出所有控制器厂商均采用PC架构PLC，价格昂贵，而且由于控制系统和PLC程序的复杂，使得各厂商都绑定了HMI和PLC，客户不具有自主性。

嵌入式PLC，因为其低功耗，低价格一直广泛应用于逻辑控制领域。但是面对大型设备的复杂逻辑和多运动控制的情况，遇到了性能上的问题。另一方面无线通信，IOT等领域的发展，对低功耗提出了苛刻的要求 [16]，而无线控制让我们未来发展有了新的认识 [17]。如何结合嵌入式PLC低功耗，低价格的优势和PC-Based功能性强，有更好的用户导向的开发方式，引导了我们的研究。

B. Related Works

随着研究的深入，PLC已经在逻辑控制的基础上结合了运动控制,整合位置控制，直线插补，圆弧插补的PLC已经有了广泛应用 [18]–[20]。由于逻辑控制和运动控制开始变得密不可分，结合运动控制的PLC使得运动控制得到极大简化。目前有两种方式实现这个整合。一种是通过独立的运动控制模块通过和PLC通信实现逻辑控制和运动控制的整合。但是这种方式具有两种控制系统，两个不同的开发方式，开发和应用复杂。 [21], [22] 通过整合PMAC运动控制卡和PLC分别实现了PID的闭环控制和5轴机械手控制。Panasonic FP2 position control unit [23]也提供了这种功能。另一种方式是在PLC中直接整合运动控制功能 [18], [24]。然而由于逻辑控制和运动控制在一个扫描周期内，这种方式，难以保证在高可靠性逻辑控制的同时，能够实现高精度的运动控制。

为此我们提出多处理器*嵌入式PLC的概念，通过多个处理来提升PLC的性能和资源同时仍然保持PLC的开发方式。很多学者在多处理器上进行了研究， [28], [29]提出了多处理器内存之间相互交互的方法。 [30]中提出了一种平衡各处理器计算能力的方法。 [31]提出了一种自性式的线程调度方法。这些方法提供了多处理器之间内存，功能和线程调度的方法，但是并没在PLC中运用。 [11]提出了一种基于FPGA的多处理器PLC方案，然而该方案在开发方式上对于复杂逻辑和多运动控制应用中没有提出改进。

对于整合运动控制功能的PLC，通常采用PLC的开发方式，由于整合方式的不同以及单独运动控制模块厂家的不同存在更多的开发方式。为了规范运动控制的开发，PLCopen也制定了相关标准 [25]。3S等公司研发了整合的工具 [26]。我们早期的研究 [12]也提出了一种定制化语言的在线编译方法。然而有时候对于运动控制底层算法的研发，除了PLC标准 [34], [35]中的语言，用户习惯于更多的更广泛的开发语言例如C或者C++语言或者采用面向对象的设计，这在上诉研究中并没有涉及。对于运动控制模块的开发方式，各家采用的开发平台均不同，开发语言也不同。PMAC运动控制卡采用C++语言开发，the MC421/221 of OMRON CS1 series 采用G-Code开发 [27],the FP series PLC created by Panasonic 采用梯形图中的特殊运动控制指令开发 [23]。

*包含多个嵌入式处理器芯片和一个嵌入式芯片中包含多个核心的两种情况

表 I
MODULES, DI/DO, AI/AO OF INJECTION MOLDING MACHINE

No.	Module(M)	DI	DO	AI	AO
1	Mold[2]	安全阀控制开关	合模	合模位置尺	系统压力
2	Injection[4]	加热接触器检测	开模	注射位置尺	流量流量
3	Core[20]	私服系统报警	注射	座台位置尺	比例背压
4	Nozzle[4]	马达过载	储料	增压压力	
5	Heating[4]	急停按钮	绿灯	加热一段输入	
6	Ejector[2]	注射防护罩	红灯	加热二段输入	
7	AirValve[20]	检出电眼	黄灯	加热三段输入	
...	....	...	...	...	...
50		螺杆转速检测	可扩展		

这些复杂的开发方式和平台，让用户在完成需求时，遇到困难。现在用户导向性越来越重要 [32], [33]。因此，我们提出一种PLC中用户导向的开发方式，以用户为中心设计PLC系统。

C. Our Contributions

我们认为一种用户导向性的开发方式应该从PLC控制系统的组成（程序，处理器，内存，线程调度）各个方面进行改进，以用户为中心，提供一种优化的方案。为此，我们通过多处理器的嵌入式PLC方案来增强其性能基础上，通过对基于图形化构件的多语言支持来提高开发者的适应性；通过用户导向式线程设计，LPM数据交换机制，以及处理器状态转换机制优化ePLC的系统结构。最终，用所提出的用户导向式的开发方法将多处理器ePLC运用在复杂逻辑控制和多运动控制的应用场景。

接下来的部分，II中会介绍系统结构，多语言支持构件组成，内存设计，用户导向式线程设计。III部分介绍多语言构件的编译，LPM数据交互机制的实现，多线程的执行和多处理器状态转换机制。IV在实验部分将介绍本方案在注塑机中的实现，并且对比Teckmation方案,Keba方案和所提出的方案的市场情况，系统结构和运行性能情况。

II. 系统组成

A. 系统结构

多处理器的嵌入式PLC硬件架构如图1。该图展示了一个主处理器和两个从处理器的ePLC结构。主处理

表 II
APPLICATION OF PC-BASED PLC IN INJECTION MOLDING MACHINES

Brand	CPU	ROM	开发方式	System	HMI
Techmation	Intel	1G	汇编	不可扩展	集成
KEBA	Intel	1G	IEC61131-3	可扩展	集成
Beckhoff	Intel	1G	IEC61131-3	可扩展	集成
Gefran	Intel	1G	IEC61131-3	可扩展	集成

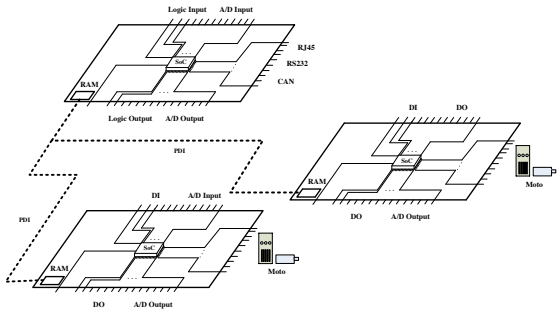

图 1. Hardware structure of the multi-processors $ePLC$.

器主要负责逻辑部分程序和通信。每个从处理器均具有数字量输入输出,模拟量输入输出,运动控制功能†。

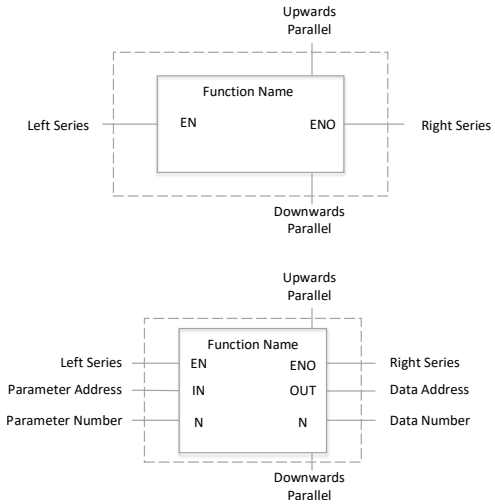

图 2. 多语言构件的两种设计，没有输入输出的构件和有输入输出的构件.

B. 多语言支持

我们的早期论文 [12]展示了如何用三层架构实现定制化程序开发。为增强嵌入式PLC中算法开发的灵活性，本文提出了融合多语言支持的构件和PLC标准图形化开发语言相结合的开发方法。为此，将多语言实现的算法封装成图形化构件。构件是用来描述多语言的语义、结构和非功能特性，一般构件描述如下：

$$LDC < Name, ID, PI, RI, PT, SF > \qquad (1)$$

其中：

Name 为构件的名称，用于描述构件实现的功能；

ID 为构件的标识符，在梯形图程序中作为唯一的标识；

PI 为构件能够提供服务的接口集合，包括梯形图构件的输出数据接口、右侧串联接、向下并联接口和部分辅助数据接口；

RI 为构件所需求服务的接口集合，包括梯形图构件的输出数据接口、左侧串联接口、向上并联接口和部分辅助数据接口；

PT 为构件内部所包含属性的集合，作为构件的基本构成元素，包括位置、大小、指令参数、注释、状态值等信息；

SF 为功能说明，以具体的文字、公式或框架模板说明构件实现的方法及功能。图形化基本构件分为触点构件、功能或功能块构件、线圈构件、横线竖线构件、换行符、注释等，多语言支持构件中包括开发语言，所支持的编译器和执行该算法的处理器等形式化描述的所有信息。

图2展示了多语言构件的两种设计。没有输入输出的构件和有输入输出的构件。

图3展示了加入多语言构件的开发方式后，从开发者角度，引擎层的开发和控制层的开发都在统一的PLC开发环境中完成。通过该开发方式，可以形成模块化的开发。每个模块包含逻辑程序和其调用的所有算法构件。算法构件可以支持不同的开发语言，包括IL指令，ST语言，C语言，C++语言等。算法构件主要指运动控制算法，还包括例如PID控制算法等。

C. 内存设计

内存主要由位数据区（M区）和字节数据区(D区)组成,在我们的系统中规定了集合的两个属性如下：

† 处理器每个功能都需增加外围电路

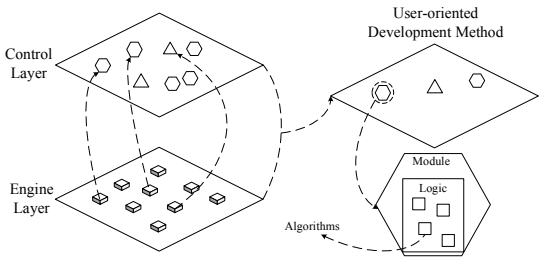

图 3. Software structure of the multi-processors $ePLC$.

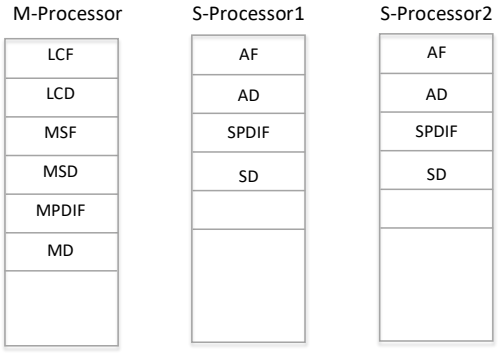

图 4. 主从处理器内存结构 $ePLC$.

Definition 1 集合 S 具有 $\mathcal{B}$ 属性：$\exists S \subset M$ and $(\exists s_i \in S) \in \{\mathcal{F}(s_i) = 0, \mathcal{O}(s_i) = 1\}$

Definition 2 集合 S 具有 $\mathcal{D}$ 属性：$\exists S \subset D$ and $\exists s_i \in S$ has 4 Byte。

其中$\mathcal{F}(s_i)$表示s_i中数据为0，$\mathcal{O}(s_i)$表示s_i中数据为1。

图4展示了主从处理器内存分区情况,从处理器的内存结构相同，各处理器之间通过共享标志位和共享数据区交互数据。所有区域介绍如下：

LCF: Logic control flag area, 存储逻辑程序的启动标志。

LCD: Logic control data area，存储逻辑程序的数据,用于将数据传递给算法。

AF: Algorithm flag area,包括算法启动标志区AFE，状态标志区AFS。

AD: Algorithm data area，存储算法运行需要的数据。

MSF: Message flag area，用户可以在该区域自定义消息标志，包括系统消息SMF和用户自定义消息UMF,均具有$\mathcal{B}$性。

MSD: Message data area,当消息发生时用于数据交换。

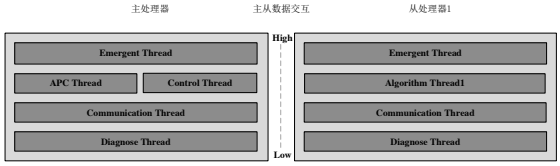

图 5. 定制化线程结构示意图.

MPDIF: Master processor data interaction flag area，主从处理器之间数据交互的标志，包括主到从的启动传递标志MS，清除标志MC和状态标志MF从到主的确认收到标志SA和状态标志SF,均具有$\mathcal{B}$性质。

MD: Master processor data interaction data area，用于存放主到从的传递数据，具有$\mathcal{D}$性。

SPDIF: Slave processor data interaction flag area，主从处理器之间数据交互的标志，包括主到从的启动传递标志SS，清除标志SC和状态标志SF从到主的确认收到标志MA和状态标志MF,均具有$\mathcal{B}$。

SD: Slave processor data interaction data area，用于存放从到主的数据传递，具有$\mathcal{D}$性。

D. 用户导向式线程设计

通常应用情况，逻辑程序的开发和算法开发可以独立设计完成 [12]并且算法的性能要求高,所以线程设计需要将逻辑线程和算法线程分开，并且算法线程需要单独运行在从处理器上。用户导向式线程结构如图5所示，每个处理器中均是四级抢占式调度线程。高级别线程可以中断低级别线程。**紧急线程**具有最高优先级，用来处理紧急事务，如中断等。**通信线程**用来完成PLC的通信任务，如人机界面等。**诊断线程**主要利用利用CPU的空闲时间进行系统故障诊断。三个特殊线程中：

控制线程:位于主处理器中，主要处理逻辑相关程序，包括读入输入信号、执行逻辑程序、刷新输出、响应用户线程和调用引擎算法执行；

定制化线程:定制化线程完成定制程序的读取、解析和执行；

算法线程:执行PLC引擎中运动控制算法，响应Control Thread请求，反馈运行过程中的数据，通过调用运动控制算法完成被控设备运行。

III. 运行机制

A. 多语言构件的图形化程序编译

通过将算法封装成构件，与图形化语言进行统一化编译 [36]。如图6含多语言构件的梯形图示例编译过程如下：

Step 1 首先根据梯形图语法库，通过拓扑抽象转化为有向图，分析梯形图拓扑结构错误；

Step 2 根据串联/并联规约，生成二叉分解树；

Step 3 根据IL语法库，执行语义翻译算法，形成IL指令。对于多语言构件，经宏处理后加载到应用程序工程中，作为调用多语言编程模块的人口点。

Step 4 读取梯形图内部地址映射表，根据内部地址映射表，将多语言算法中涉及的相关地址转化为相关处理器硬件系统中对应的逻辑存储地址，并将经过转化的多语言算法内容保存到对应处理器工程中，供编译器编译调用。

Step 5 调用处理器对应编译器编译完成。

B. LPM data interaction

我们定义了LPM数据交互方法：$\mathcal{L}$(layer data interaction),$\mathcal{P}$(processor data interaction) and $\mathcal{M}$(module data interaction)。$\mathcal{L}$参考 [12]中层于层之间的数据交互。$\mathcal{P}$定义如下：

$$\begin{cases} \mathcal{P}_{mts} = \mathcal{I}(ms_i, mc_i, mf_i, sa_i, sf_i, sd_i) \\ \mathcal{P}_{stm} = \mathcal{I}(ss_i, sd_i, sf_i, ma_i, mf_i, md_i) \end{cases} \tag{2}$$

$\mathcal{P}_{mts}$和$\mathcal{P}_{mts}$具有相同执行函数$\mathcal{I}$，且执行过程如下: $\mathcal{O}(ms_i) \rightarrow \mathcal{O}(mf_i) \rightarrow send(sd_i) \rightarrow \mathcal{O}(sf_i) \rightarrow check(sd_i) \rightarrow \mathcal{O}(sa_i) \rightarrow \mathcal{O}(mc_i) \rightarrow \mathcal{F}(ms_i) \rightarrow \mathcal{F}(mf_i) \rightarrow \mathcal{F}(mc_i) \rightarrow \mathcal{F}(sf_i) \rightarrow \mathcal{F}(sa_i)$

其中，$send(sd_i)$表示将数据传到从处理器的sd_i中。$check(sd_i)$表示对sd_i中数据进行合规性检查。

$\mathcal{M}$定义如下：

$$\begin{cases} \mathcal{M}_s = \mathcal{J}(smf_i, msd_i, \mathcal{C}) \\ \mathcal{M}_u = \mathcal{J}(umf_i, msd_i\, \mathcal{C}) \end{cases} \tag{3}$$

其中$\mathcal{C}$为所有需要相应的模块的回调函数集。$\mathcal{M}_s$和$\mathcal{M}_u$具有相同执行函数$\mathcal{J}$,且一个模块收取消息的执行过程如下： $\mathcal{O}(smf_i) \rightarrow GetMessage_j(smf_i) \rightarrow GetData(msd_i) \rightarrow \mathcal{C}_j$

其中，$GetMessage_j(smf_i)$表示第i个模块接收到消息。$GetData(msd_i)$表示第i个模块获取消息数据。

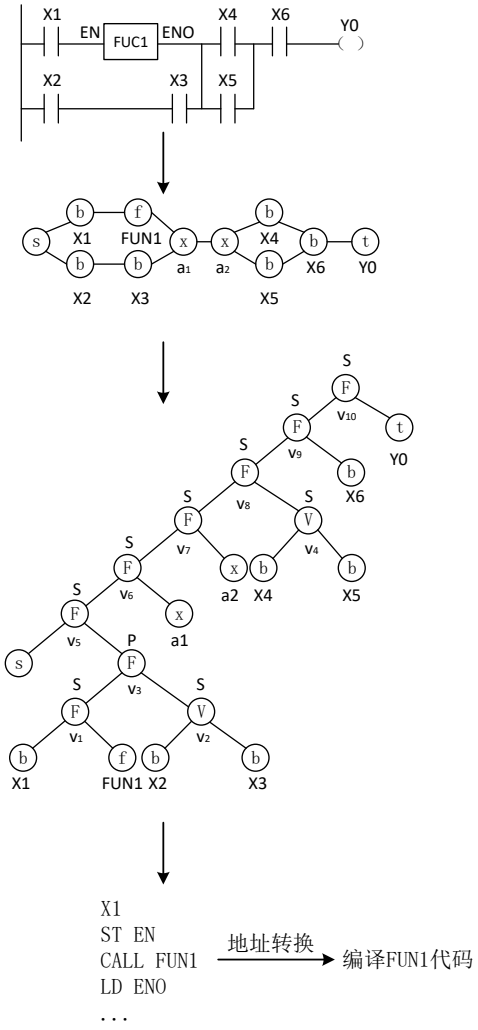

图 6. 含多语言构件的梯形图编译.

C. 多线程的执行

执行时主从处理器按各自线程优先级执行，线程间交互发生在控制线程和运动线程之间，两者间通过$\mathcal{P}_{mts}$和$\mathcal{P}_{mts}$实现。控制线程执行单元如下：

C_1 启动指定模块。可以通过控制模块通过消息机制完成或者通过输入输出或其他模块启动。

C_2 通过方法$\mathcal{P}_{mts}$,将数据传递给运动线程。

C_3 处理算法反馈数据。

C_4 消息广播。

C_5 消息处理。

运动线程执行单元如下：

M_1 通过对算法启动标志判断，是否启动算法并对算法状态和处理器状态处理，搬移数据到AD。

M_2 执行算法。

M_3 通过方法$\mathcal{P}_{mts}$,向主处理器反馈数据。

M_4 结束算法。对算法状态和处理器状态处理。

下面阐述了两种典型线程处理场景如图7：

Case one：三处理器三线程间数据交互和算法执行。控制线程遍历LCF,发现存在需执行模块,执行逻辑程序，发现需执行算法1，执行C_1单元,执行C_2单元，将数据从LCD传递给处理器1的SD，算法线程1执行M_1单元，数据从SD到AD后执行M_2，运控控制算法执行期间执行M_3单元，将运动过程数据范阔给主处理器，控制线程执行C_3单元，运动控制算法执行完毕后，算法线程1执行M_4单元，并反馈数据告知主处理器算法执行完毕。控制线程执行C_3单元完成结束算法流程，执行逻辑程序后发现需执行算法2，执行C_1单元，执行C_2单元，将数据从LCD传递给处理器2的SD，算法线程2执行M_1单元，数据从SD到AD后执行M_2单元，等待算法结束执行M_4单元，反馈数据告知控制线程结束算法，控制线程获得信息后执行C_3单元完成该算法。

Case two：三处理器三线程间带消息机制的数据交互和算法执行。控制线程遍历LCF,发现存在需执行模块1,执行逻辑程序，发现需执行算法1，执行C_1单元,执行C_2单元，将数据从LCD传递给处理器1的SD，算法线程1执行M_1单元，数据从SD到AD后执行M_2，运动控制算法1执行期间，模块1需要告知模块2开始执行，控制线程执行C_4单元，从模块1中获取数据放在MSD后广播消息,执行C_5单元解读消息，启动模块2，发现需要执行模块2中算法2，执行C_1单元，执行C_2单元，将数据从LCD传递给处理器2的SD，算法线程2执行M_1单元，数据从SD到AD后执行M_2单元，开始执行算法2，这期间算法线程1执行M_4单元，反馈数据告知控制线程结束算法1，控制线程获得信息后执行C_3单元完成算法1。算法线程2等到算法2结束执行M_4单元，反馈数据告知控制线程结束算法2，控制线程获得信息后执行C_3单元完成算法2。

D. 多处理器状态转换

主处理器状态集$MPS = \{mstop, mrun\}$,$mstop$表示主控制器处于空闲状态,$mrun$表示主控制器处于停止状态。从处理器状态集$SPS = \bigcap_{i=1}^{n} SPS_i$,其中$SPS_i$表示第$i$个处理器状态集,$SPS_i = \{sstop_i, srun_i, sready_i\}$,$sstop_i$表示第i个处理器处于停止状态， $srun_i$表示第i个处理器处于运行状态， $sready_i$表示第i个处理器处于等待状态。主从处理器状态转换过程如下：

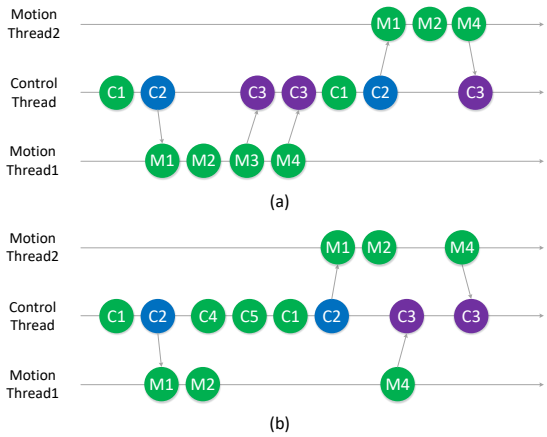

图 7. 线程执行的两个典型场景.

图 8. Injection Molding Machine

$$\begin{cases} mstop \rightarrow mrun \Leftarrow \exists lcf_i \in LCF = 1 \\ mrun \rightarrow mstop \Leftarrow \forall lcf_i \in LCF = 0 \\ sstop \rightarrow sready \Leftarrow \exists sa_i \in SA = 1 \\ sready \rightarrow srun \Leftarrow \exists afe_i \in AFE = 1 \\ srun \rightarrow sstop \Leftarrow \forall afe_i \in AFE = 0 \end{cases} \tag{4}$$

IV. 实验

A. 分布式扩展控制系统

如图8所示，本实验在一台200吨注塑机上验证。控制器芯片采用TI F28M35芯片，该芯片具有两个核心：

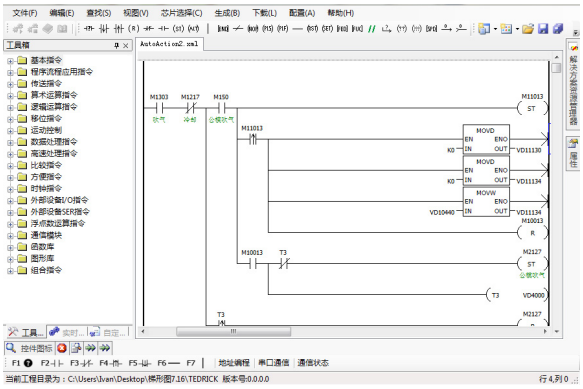

图 9. LD.

一个TI C28x 与一个ARM Cortex M3。Cortex M3处理逻辑控制，DSP处理运动控制。该ePLC具有一个RJ45接口，一个串口和一个CAN口。RJ45负责程序下载，串口和人机界面进行通信。CAN口用于扩展DI,DO,AI,AO模块。通过CAN总线，实现了整个系统的分布式控制。执行机构选用the Inovance IS580T080-R1-1 and ISMG2-48D17CD-R131F伺服系统。人机交互界面用户可以根据需要选择。

B. 程序架构

如图9所示，程序采用梯形图中嵌入C语言构件的方式完成开发。图X(a)中红框内显示了无参数的C语言构件。该构件表示点动构件。当执行该构件时，执行机构每次固定速度和压力执行。(b)为该构件的源码片段。当该构件开始执行时，先从交换数据区读取数据并检测数据，如果数据有误则报错，结束执行。如果数据正确则按给定压力流量执行。

完成基本构件设计后，整个注塑机程序根据表I,分模块设计。模块之间通过消息机制交互。通过一个特殊的控制模块对所有模块的调用实现整个注塑机程序。

当程序下载到ePLC后，用户可以通过HMI界面调整工艺和程序执行顺序。

用户导向的开发方式中几段典型的程序实现如下：

1) 逻辑程序和运动程序一体化设计。增加顶针模块且使用S曲线加减速。使用梯形图完成逻辑部分设计。S曲线加减速部分程序通过构件完成。
2) LPM数据传递实现模块化设计。合模高压开始前注射。通过消息机制，在合模模块中设立自定义标志位，当合模结束时把该标志置1.控制模块检测到该标志为1则通知所有模块。注射模块得到该消息立即开始注射。
3) 定制化在线编译方法。合模结束后增加顶针动作。在HMI界面中，合模后面插入一个顶针模块，ePLC在线编译，无需重启即可运行。

C. 系统对比

在相同机器中，对比了不同系统间的系统情况，系统架构，系统性能。，系统启动时间，开合模进度，注射进度，系统架构，相同周期时间下次品率。

1) 系统情况。Teckmation和KEBA占据了注塑机控制器很大市场,均是基于PC架构。由于本系统采用了嵌入式架构，成本上本系统更低。
2) 系统架构。对比表II,本系统采用用户导向的开发方式，包括多处理器的PLC，多语言图形化构件的支持，在多线程，数据交换等方面进行优化的系统结构。另外本系统采用了广泛使用的总线分布式结构，减少了电线的使用和增强了抗干扰能力。由于本系统的复杂度大大降低，使得HMI和PLC可以任意组合，给用户提供更多自主化选择。而其他系统均是HMI和PLC绑定，不具有多语言构件支持。而Teckmation更是使用汇编指令开发，不支持IEC61131-3开发语言。
3) 机器性能关键指标。注塑机性能关键指标为全自动生产的次品率，转保压位置重复精度,保压结束位置重复精度，储料结束位置重复精度，开模位置重复精度。在所有动作控制算法均采用斜坡加减速情况下，周期时间都控制在8秒，分别地合模时间控制在2秒，开模时间2秒，注射时间控制在1秒，储料时间控制在1秒，冷却时间0.5秒，射退时间0.5秒，顶出顶进时间1秒。对比了机器性能关键指标。如图10所示。(a)为100次转保压位置误差,(b)100次转保压结束位置误差，(c)100次储料结束位置误差(d)100次开模位置误差。从中可以看出本控制系统和KEBA控制系统接近，略高于弘讯控制系统。表III,可以看出系统启动时间提高了20多倍。其他次品率，100次转保压位置，转保压结束位置，储料结束位置和开模位置误差均值，各系统接近，Teckmation略差。

V. CONCLUSION

本文从PLC系统的组成：程序，处理器，内存和线程机制，阐述了一种支持多处理器ePLC的用户导向式开发方式。我们通过多处理器的嵌入式PLC方案在兼

表 III
注塑机控制系统性能对比

Brand	系统启动时间	次品率	转保压位置100次平均	保压结束位置100次平均	储料结束位置100次平均	开模位置100次平均
Techmation	120s	0.27%	0.32	0.13	0.13	0.14
KEBA	150s	0.24%	0.24	0.11	0.15	0.15
本系统	6s	0.25%	0.20	0.14	0.11	0.15

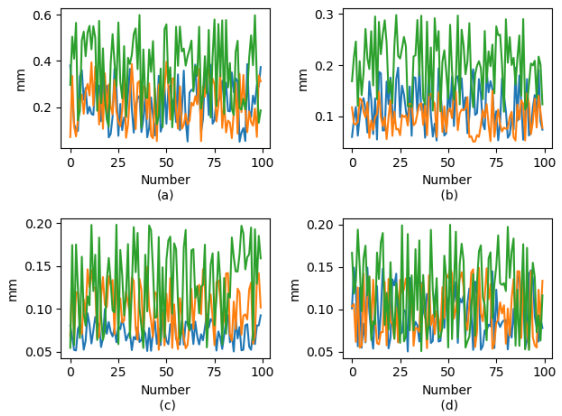

图 10. (a)为100次转保压位置误差,(b)100次转保压结束位置误差，(c)100次储料结束位置误差(d)100次开模位置误差。

顾嵌入式PLC的低功耗，低价格基础上增强其性能；通过对基于图形化构件的多语言支持来提高开发者的适应性；通过用户导向式线程设计，LPM数据交换机制，以及处理器状态转换机制优化ePLC的系统结构。最终，用所提出的用户导向式的开发方法将多处理器ePLC运用在注塑机控制上。分析得出，所提出的方案在机器性能关键指标和现有控制系统方案相当情况下，成本得到降低，启动速度提高了20倍，且支持分布式控制和人机界面分离。

参考文献

[1] A. G. C. Gonzalez, M. V. S. Alves, G. S. Viana, L. K. Carvalho, and J. C. Basilio, "Supervisory control-based navigation architecture: A new framework for autonomous robots in industry 4.0 environments," *IEEE Transactions on Industrial Informatics*, vol. PP, no. 99, pp. 1–1, 2017.

[2] D. A. Chekired, L. Khoukhi, and H. T. Mouftah, "Industrial iot data scheduling based on hierarchical fog computing: A key for enabling smart factory," *IEEE Transactions on Industrial Informatics*, vol. PP, no. 99, pp. 1–1, 2018.

[3] S. Li, Q. Ni, Y. Sun, G. Min, and S. Al-Rubaye, "Energy-efficient resource allocation for industrial cyber-physical iot systems in 5g era," *IEEE Transactions on Industrial Informatics*, vol. PP, no. 99, pp. 1–1, 2018.

[4] L. Ling, C. Chen, S. Zhu, and X. Guan, "5g enabled co-design of energy-efficient transmission and estimation for industrial iot systems," *IEEE Transactions on Industrial Informatics*, vol. PP, no. 99, pp. 1–1, 2018.

[5] H. Carlsson, S. Bo, F. Danielsson, and B. Lennartson, "Methods for reliable simulation-based plc code verification," *IEEE Transactions on Industrial Informatics*, vol. 8, no. 2, pp. 267–278, 2012.

[6] M. Obermeier, S. Braun, and B. Vogel-Heuser, "A model-driven approach on object-oriented plc programming for manufacturing systems with regard to usability," *IEEE Transactions on Industrial Informatics*, vol. 11, no. 3, pp. 790–800, 2015.

[7] B. F. Adiego, D. Darvas, E. B. Viñuela, J. C. Tournier, S. Bliudze, J. O. Blech, and V. M. G. Suárez, "Applying model checking to industrial-sized plc programs," *IEEE Transactions on Industrial Informatics*, vol. 11, no. 6, pp. 1400–1410, 2015.

[8] S. Gerkšič, G. Dolanc, D. Vrančić, J. Kocijan, S. Strmčnik, S. Blažič, I. Škrjanc, Z. Marinšek, M. Božiček, and A. Stathaki, "Advanced control algorithms embedded in a programmable logic controller," *Control Engineering Practice*, vol. 14, no. 8, pp. 935–948, 2006.

[9] C. Y. Chang, "Adaptive fuzzy controller of the overhead cranes with nonlinear disturbance," *IEEE Transactions on Industrial Informatics*, vol. 3, no. 2, pp. 164–172, 2007.

[10] S. Dominic, Y. Lohr, A. Schwung, and S. X. Ding, "Plc-based real-time realization of flatness-based feedforward control for industrial compression systems," *IEEE Transactions on Industrial Electronics*, vol. PP, no. 99, pp. 1–1, 2016.

[11] Z. Hajduk, B. Trybus, and J. Sadolewski, "Architecture of fpga embedded multiprocessor programmable controller," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 5, pp. 2952–2961, 2015.

[12] H. Wu, Y. Yan, D. Sun, and S. Rene, "A customized real-time compilation for motion control in embedded plcs," *IEEE Transactions on Industrial Informatics*, vol. PP, no. 99, pp. 1–1.

[13] D. Schütz, A. Wannagat, C. Legat, and B. Vogel-Heuser, "Development of plc-based software for increasing the dependability of production automation systems," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 4, pp. 2397–2406, 2013.

[14] J. D. L. Morenas, P. G. Ansola, and A. García, "Shop floor control: A physical agents approach for plc-controlled systems," *IEEE Transactions on Industrial Informatics*, vol. PP, no. 99, pp. 1–1, 2017.

[15] S. Hossain, M. A. Hussain, and R. B. Omar, "Advanced control software framework for process control applications," *International Journal of Computational Intelligence Systems*, vol. 7, no. 1, pp. 37–49, 2014.

[16] R. Arshad, S. Zahoor, M. A. Shah, A. Wahid, and H. Yu, "Green iot: An investigation on energy saving practices for 2020 and beyond," *IEEE Access*, vol. 5, no. 99, pp. 15 667–15 681, 2017.

[17] W. Zhang, M. Tomizuka, P. Wu, Y. H. Wei, Q. Leng, S. Han, and A. K. Mok, "A double disturbance observer design for compensation of unknown time delay in a wireless motion control system," *IEEE Transactions on Control Systems Technology*, vol. PP, no. 99, pp. 1–9, 2018.

[18] M. G. Ioannides, "Design and implementation of plc-based monitoring control system for induction motor," *IEEE Transactions on Energy Conversion*, vol. 19, no. 3, pp. 469–476, 2004.

[19] X. M. Shi, W. J. Fei, and S. P. Deng, "The research of circular interpolation motion control based on rectangular coordinate robot," *Key Engineering Materials*, vol. 693, pp. 1792–1798, 2016.

[20] N. Fang, "Design and research of multi axis motion control system based on plc," *Academic Journal of Manufacturing Engineering*, vol. 15, no. 1, pp. 17–23, 2017.

[21] W. F. Peng, G. H. Li, P. Wu, and G. Y. Tan, "Linear motor velocity and acceleration motion control study based on pid+velocity and acceleration feedforward parameters adjustment," *Materials Science Forum*, vol. 697-698, pp. 239–243, 2011.

[22] J. Qian, H. B. Zhu, S. W. Wang, and Y. S. Zeng, "A 5-dof combined robot platform for automatic 3d measurement," *Key Engineering Materials*, vol. 579-580, pp. 641–644, 2014.

[23] P. Co.Ltd, *Programmable controller FP2 Positioning Unit Mannual*, 2011.

[24] A. Syaichu-Rohman and R. Sirius, "Model predictive control implementation on a programmable logic controller for dc motor speed control," in *Electrical Engineering and Informatics (ICEEI), 2011 International Conference on*. IEEE, 2011, pp. 1–4.

[25] C. Sünder, A. Zoitl, F. Mehofer, and B. Favre-Bulle, "Advanced use of plcopen motion control library for autonomous servo drives in iec 61499 based automation and control systems," *E & I Elektrotechnik Und Informationstechnik*, vol. 123, no. 5, pp. 191–196, 2006.

[26] S. S. S. GmbH, "Logic and motion control integrated in one iec 61131-3 system:development kit for convenient engineering of motion, cnc and robot applications." 2017.

[27] O. Co.Ltd, "Cs1w-mc221(-v1)/mc421(-v1) motion control units." *Operation Mannual*, 2004.

[28] M. Dubois and C. Scheurich, "Memory access dependencies in shared-memory multiprocessors," *IEEE Transactions on Software Engineering*, vol. 16, no. 6, pp. 660–673, 2002.

[29] J. H. Patel, "Processor-memory interconnections for multiprocessors," *IEEE Transactions on Computers*, vol. C-30, no. 10, pp. 771–780, 2006.

[30] D. Zhu, L. Chen, S. Yue, T. Pinkston, and M. Pedram, "Providing balanced mapping for multiple applications in many-core chip multiprocessors," *IEEE Transactions on Computers*, vol. 65, no. 10, pp. 3122–3135, 2016.

[31] L. M. Albarakat, P. V. Gratz, and D. A. Jiménez, "Mtb-fetch: Multithreading aware hardware prefetching for chip multiprocessors," *IEEE Computer Architecture Letters*, vol. PP, no. 99, pp. 1–1, 2017.

[32] O. Verscheure, X. Garcia, G. Karlsson, and J. P. Hubaux, "User-oriented qos in packet video delivery," *IEEE Network*, vol. 12, no. 6, pp. 12–21, 2016.

[33] N. Choi, D. Kim, S. J. Lee, and Y. Yi, "A fog operating system for user-oriented iot services: Challenges and research directions," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 44–51, 2017.

[34] I. Tc65/Wg, "Programmable controllers – part 3: Programming languages," 1993.

[35] K. H. John and M. Tiegelkamp, *IEC 61131-3: Programming Industrial Automation Systems*. Springer Berlin Heidelberg, 2010.

[36] Y. Yan and H. Zhang, "Compiling ladder diagram into instruction list to comply with iec 61131-3," *Computers in Industry*, vol. 61, no. 5, pp. 448–462, 2010.