计算机集成制造系统 2012.11

基于 XSLT 的通用 STEP-NC 后置处理器开发

肖文磊[1]　邰　极[1]

（北京航空航天大学机械工程及自动化学院　北京　100191）

摘要：为了使 STEP-NC 兼容传统的数控系统，需要在传统数控系统上构建专用的 STEP-NC 后置处理器。本文建立了一个通用的后置处理器，以降低构建专用 STEP 后置处理器的开发工作量和开发难度。采用 XML 模式下的 STEP-NC 代码作为输入和基于 XSLT 的转换机制，实现面向不同数控加工设备统一化的后置处理过程。根据 XSLT 转换原理和要求，分别开发了 EXPRESS-X 和 P21-P28 的文件格式转换器。构建了以 P28 格式作为输入和以 XSLT 样式表语言定义机床设备接口格式的通用后置处理器。 最后，以一台切削加工机器人和一台三轴数控铣床为应用实例，分别以机器人语言和 G 代码作为后置处理结果，对后置处理器的功能性和可行性进行了验证。

关键词：STEP-NC；XML；XSLT；后置处理

中图分类号：**TP391**

Development of a Universal Post-processor for STEP-NC by using XSLT

XIAO Wenlei[1]　*HUAN Ji*[1]

(School of Mechanical Engineering and Automation, Beihang University, Beijing 100191)

Abstract：*In order to make STEP-NC compatible with the traditional CNC systems, it is necessary to develop for each traditional CNC system a customized STEP-NC post-processor. This paper developed a universal post-processor for STEP-NC, so that the workloads and difficulties in constructing the customized STEP-NC post-processor could be significantly reduced. The STEP-NC code in XML format was taken as the post-processor input and the principle of transformation was written by XSLT. To fulfil the working principle and requirements of XSLT, the data converters for EXPRESS-X and P21-P28 were individually developed. After that, a universal post-processor was developed, which takes P28 file as the data input and XSLT style sheet as the definition of CNC systems and machine tools. At last, some tests were implemented into a cutting robot and a three-axis milling machine, which*

accepted robot language and G-code respectively as the cutter location data. The results testified and verified the functionality and feasibility of the proposed post-processor.

Key words：STEP-NC; XML; XSLT; Post-processing

0 前言

当今，随着计算机技术的飞速发展，先进的数控系统逐渐向开放化、智能化、网络化和柔性化等方向发展，传统的数控编程语言 G 代码由于其固有的局限性，已经难以适应数控系统发展的新需求。STEP-NC 的提出即为解决这一问题，它将产品数据交换标准（STandard for the Exchange of Product model data，STEP）扩展至计算机数控（Computer Numerical Control，CNC）领域，旨在以 STEP 统一表征 CNC 加工过程中涉及的全部信息，为 CNC 系统提供完整的产品数据，从而取代传统的 G 代码接口格式，并为发展数控系统的开放性、智能化和网络化奠定基础。

能够支持 STEP-NC 编程语言接口的数控系统被称为 STEP-CNC。然而，由于 STEP-NC 技术本身还处于研究阶段，目前世界范围内还没有出现应用级别的 STEP-CNC 数控系统。基于 G 代码接口的数控系统在未来很长一段时间内还不会被完全淘汰。在可预见的将来，必然会出现一个 STEP-NC 代码与 G 代码并存的过渡时期。为了兼容现有的数控系统，以达到尽可能将发挥 STEP-NC 的优点，解决或部分解决 G 代码在传统数控系统之间的不兼容、不易调整以及不直观等局限性，一些研究学者提出了 STEP-NC 后置处理器的概念。STEP-NC 后置处理器的意义在于：如果每个 CNC 系统具有一个能够接收 STEP-NC 代码的专用后置处理器，那么输入的 STEP-NC 数据就不依赖于特定的控制系统，从而能够实现数据接口统一[1-3]。然而，由于 STEP-NC 代码解析本身的复杂性，而现有的 CNC 系统种类繁多且互不兼容，为每个 CNC 系统专门开发一套具有 STEP-NC 解析器的后置处理单元是非常庞大而难以实现的工作。

为了解决这个难题，尽可能减少开发成本和缩短开发周期，亟需一种通用的可针对不同控制系统和机床配置的 STEP-NC 后置处理器。在实现这个目标的过程中，首先需要解决的就是可配置的代码格式转换的问题。在解决这个问题之前，值得关注的是后置处理器的输入数据是 CAD/CAM 系统，输出数据是作为控制系统的输入，这是一个典型的应用程序之间的数据交换问题。变化无穷的输出格式常常使得难以兼容不同的平台。因此需要一种组织和呈现数据的统一方式，这是 STEP 格式本身无法提供的。幸运的是，计算机技术的发展为这一问题已经提供了理想的解决方案——可扩展标签语言（eXtensible Markup Language，XML）。XML 使用自描述的方式呈现数据，因此被广泛用来作为跨平台之间交互数据的形式，主要针对数据的内容，通过不同的格式化描述

手段（XSLT，CSS 等）可以完成最终的形式表达（生成对应的 HTML，PDF 或者其他的文件格式）。而为了适应于 XML 的迅猛发展，STEP 标准也在 2003 年推出了基于 XML 格式的 STEP 数据表达（ISO 10303-28），通常被称为 P28 格式或 STEP-XML。

目前，国内外针对 STEP-NC 的 XML 应用方面已有一些研究[4][5]，然而这些应用案例中都采用了各自自定义的 XML 描述格式，这为 STEP-XML 未来应用的通用性带来了很大的潜在威胁[4]。尽管 ISO 10303-28 给出了 STEP-XML 的标准规范[6]，然而到目前为止还没有真正被应用起来。本文应用的 STEP-XML 相关数据转换均遵从 ISO 10303-28 文档，是真正意义上符合标准定义和满足一致性检测要求的 STEP-XML 应用实例。此外，目前大部分的 STEP-XML 转换器的开发主要依靠 STEP Tools 公司提供的 ST-Developer 或针对某应用协议的特例开发的有限原型系统。本文提出的 STEP-XML 转换方法和原理不依赖于昂贵的专用开发包，并能够实现任意 EXPRESS 模式支配的 P21 文件向 P28 文件的转换过程。

以 STEP-XML 为基础架构，利用 XML 的在数据转换方面的强大功能，本文提出一种基于 XSLT 的通用 STEP-NC 后置处理器技术。采用 STEP-XML 封装的 STEP-NC 代码作为输入，通过 XSLT 语言配置指定控制系统和目标机床的自定义信息，可以实现 STEP-NC 后置处理器平台的统一，即使用一个 STEP-NC 后置处理器可以适用于不同的控制系统。

1 后置处理器转换原理

本问题提出的后置处理器主要是基于 XML 和 XSLT 的转换原理运行，因此后置处理器的输入输出都应遵循相应要求。

1.1 XSLT 工作原理

可扩展样式表转换语言（Extensible Stylesheet Language Transformations，XSLT）是一种 XML 转换语言。它用于将 XML 节点树作为源，通过一系列模板或规则将其转换为某种结果。XSLT 本身是一种 XML 语法，但 XSLT 转换的结果可以不是 XML，其规范允许的输出有 XML、HTML 或文本，甚至是二进制内容。其工作原理如图 1 所示[7]。

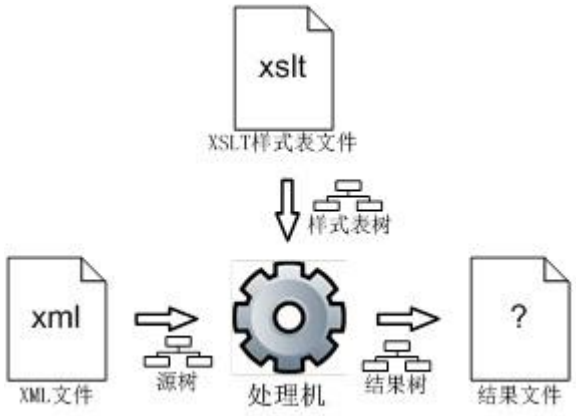

图 1 XSLT 样式表转换原理

根据 XSLT 这种特性，可以利用 XSLT 样式表定制相应数控系统的数据接口格式，把加工代码的不一致之处完全定义到 XSLT 样式表文件中。从而可以使用通用的 XSLT 处理机对输入的 STEP-NC 代码（基于 XML 格式）进行代码转换。而输出代码的格式完全由定义数控系统的 XSLT 样式表文件确定。

1.2 输入代码分析

STEP-NC 代码采用 STEP 格式进行数据定义。STEP 标准中主要提供了两种文件编码格式：纯文本表达[8]和 XML 表达[6]。通常，使用纯文本表达的 STEP 文件称为 P21 文件，而使用 XML 表达的 STEP 文件称为 P28 文件。尽管 STEP 标准定义了 XML 表达的 P28 文件格式，目前大部分 STEP-NC 的研究都是针对于 P21 文件，而市场现有的 CAD 系统的 STEP 文件输出几乎都采用 P21 文件格式。因此，目前最直接的后置处理器输入数据应为 P21 文件格式。然而，由于现有 XSLT 样式处理机的局限性，要求输入的 STEP-NC 文件格式必须为 XML 格式（尽管 XSLT2.0 可以支持一般文本格式的输入，对类似于 P21 等较为复杂的文件格式的支持方面尚存在一些技术障碍）。另外，随着网络化制造的进一步发展，基于 XML 的 P28 文件将逐渐体现出其优于 P21 文件的特性，因此，采用 P28 文件作为输入的 STEP-NC 后置处理器也更适应于未来的网络化制造。基于以上考虑，本文设计了一个 STEP-NC 的 P21 文件到 P28 文件的转换器。并规定后置处理器的输入为统一的 P28 文件。

1.3 输出代码分析

根据定义，STEP-NC 后置处理器的输出主要服务于各种加工设备的控制系统。使用 G 代码接口的数控系统是其典型的服务对象。然而，应用于制造系统中的自动化设备不仅仅有数控机床。例如，就加工而言，用于加工的设备不仅仅有数控机床，还有加工机

器人[9]。而数控机床控制器的输入有时也不一定是 G 代码。另外，在一些研究过程中的数控系统中，为求简化，往往采用一些自定义的输入格式。因此，一个通用的后置处理器应该能够满足上述不同的输出格式要求。

一般而言，后置处理器的兼容性主要体现在以下两个级别：

(1) G 代码兼容

由于大部分的数控机床都是支持 G 代码格式的。ISO 6983 定义了相当大部分的 G 指令，因此不同系统的 G 代码都比较类似，但又有一些不同之处。有些系统生产商往往还会加入一些自定义的 G 指令，例如表 1 给出的一些数控系统对 NURBS 样条插补的 G 指令[1]。

表 1　数控系统中 NUBRS 插补的 G 指令

数控系统	FANUC	SIEMENS	OKUMA	Mitsubishi
G 代码	G06.2	BSPLINE	G132	G70.0 G70.1

(2) 一般性兼容

对于更为广义的加工设备而言，不同系统之间的代码格式就往往毫无共同之处可言。例如，G 代码与工业机器人代码之间就存在非常大的差异。而用户自定义的代码更为千变万化，可以采用纯文本格式，也可以采用 INI、XML 等进行封装，甚至可以为二进制文件。要实现这一级别的兼容，需要后置处理器更大的灵活性和可配置性。

1.4 STEP-NC 后置处理过程

STEP-NC 代码是面向对象的，STEP-NC 代码一般会包含一个或多个加工工程（Workingproject）。一个加工工程一般包含若干个加工工步（Workingstep）。每个加工工步根据需要又会包含所加工对象的几何信息（加工特征，Manufacturing Feature）和加工过程的工艺方法（加工方法，Machining Operation）各方面的信息。

STEP-NC 代码通常分为不同的一致性分类（Conformance Class，CC）。根据 ISO 10303-238 的一致性要求，STEP-NC 定义了四种一致性分类[10]：

CC1：刀具路径编程

CC2：闭环编程

CC3：基于特征编程

CC4：可再生编程

其中，CC1 和 CC2 依赖于指定的刀具路径作为加工的指导信息，而 CC3 和 CC4 提供加工对象完整的几何和工艺信息，因此不依赖于指定的刀具路径。因此，对于 CC1 和 CC2 的 STEP-NC 代码，后置处理器不需要内置的刀轨生成器，而对于 CC3 和 CC4 的 STEP-NC 代码需要在后置处理器中包含一个独立于机床结构的刀轨生成器。

STEP-NC 后置处理器的基本处理过程如图 2 所示：

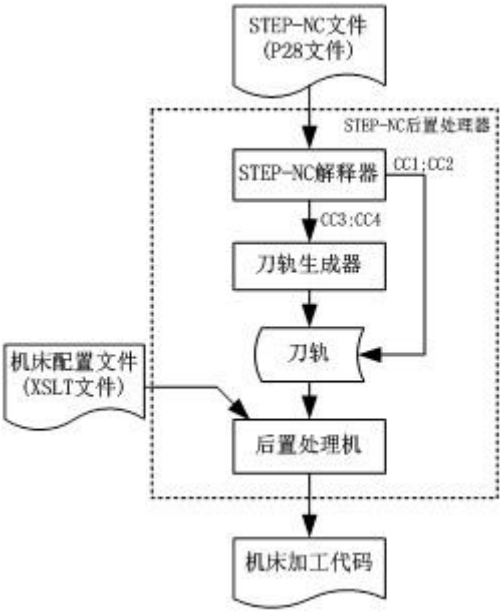

图 2 STEP-NC 后置处理器的基本处理过程

2 STEP-XML 转换器

根据 1.2 中的分析，需要开发一个 P21-P28 转换器。事实上，这项任务也可以通过强大的 XSLT 来完成。尽管 STEP-XML 的优越性十分显然，但是在其最初的应用中，却无法摆脱现有 P21 架构的影响。如前所述，每个应用协议、集成资源和通用资源等定义都对应有一个 EXPRESS 文本文件。在 STEP-NC 及其他应用协议的具体实施过程中，无法避免的需要使用到已有资源的信息模型，即已有的 EXPRESS 文件。而 STEP-XML 系统所需的信息定义为 XML 模式下的 EXPRESS（以下简称 EXPRESS-X）。在现有的 STEP 标准中，提供了大量的定义各种资源的 EXPRESS 文件。将这些 EXPRESS 文件都改写成 EXPRESS-X 文件将是一个非常庞大的工作量，且难以保证其正确性。对此，需要一个将 EXPRESS 语言翻译成 EXPRESS-X 的转换器。具备了 EXPRESS-X 文件之后，只是解决了 STEP-XML 中的数据模型定义问题。接下来需要考虑的就是如何将现存系

统输出的 P21 文件翻译成 P28 文件的问题。另外，由于 P28 文件自身定义了 3 种不同的表达方法，在使用过程中往往需要这 3 种不同格式之间的互相转换，因此还需要完成 P28 文件之间的转换。综上所述，STEP-XML 系统构建的主要研究问题以及相互的关系如图 3 所示。

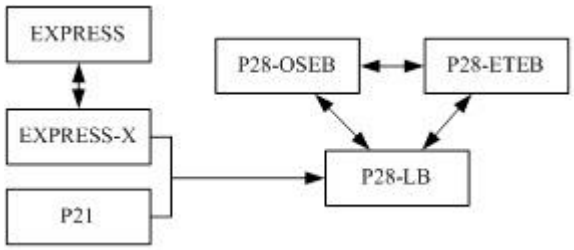

图 3 STEP-XML 系统的主要研究问题

其中，最重要的是 EXPRESS 到 EXPRESS-X 转换及 P21 到 P28 文件转换问题。限于篇幅，本节仅针对这两个问题进行论述。

2.1 XML 模式的 EXPRESS

ISO 10303-28 不仅定义了 STEP 数据部分的 XML 表达，也提供了 EXPRESS 模式的 XML 表达。EXPRESS 是一种规范化的信息模型语言，主要用来对 STEP 数据模型的信息建模。以下为一个 EXPRESS 描述实体定义的示例：

```
ENTITY cartesian_point
  SUBTYPE OF (point);
    coordinates : LIST [1:3] OF length_measure;
END_ENTITY; -- 10303-42: geometry_schema
```

对应的 EXPRESS-X 文件表达方式如图 4 所示。另外，为保证 EXPRESS-X 文件的一致性，ISO 10303-28 标准提供了用于 EXPRESS-X 一致性检测的文档类型定义（Document Type Definition，DTD）。只要生成的 EXPRESS-X 符合该 DTD 文档的规则，就表明 EXPRESS-X 满足标准的格式要求。例如，对 EXPRESS-X 实体定义（entity_decl）的 DTD 规则如下，给出了 entity_decl 可以具有的属性：

```
<!ELEMENT entity_decl ((embedded_remark | tail_remark)?, entity_id, (abstract_supertype | supertype_of)?, subtype_of?,
explicit_attr_block?, derive_clause?, inverse_clause?, unique_clause?, where_clause?)>
```

图 4 实体 cartesian_point 的 EXPRESS-X 表达

2.2 flex & bison

尽管 EXPRESS 语言并非一种可编译或解释后执行的语言，其复杂性决定了即使仅对其进行解码，用常规手工编程的方法也是极为困难的。对于此类词法和语法的解析常常需要使用专用的编译原理的方法和工具。其中最经典的工具是 lex 和 yacc。lex 和 yacc 由贝尔实验室在 20 世纪 70 年代开发，而 flex 和 bison 是它们的现代版本。具体来说，flex 与 bison 是为编译器、解释器设计者定制的工具，可以将 lex 文本（*.l 文件）和 yacc 文本（*.y 文件）转换为能够处理结构化输入的 C/C++语言，也常常用于非编译的语言解析[11]。因此，采用 flex 与 bison 成为了当前快速高效的开放 XML 模式的 EXPRESS 转换器的最可行选择。

本文采用美国国家标准和技术研究所（National Institute of Standards and Technology，NIST）开发的开源 SCL 库（STEP Class Library）所提供的用于 EXPRESS 解析的*.l 和*.y 文件作为基础，进行修改和扩展，实现 EXPRESS 解析以及翻译成为 XML 模式下的 EXPRESS 的功能。图 5 解释了 EXPRESS-X 转换器的工作原理。首先使用 flex 和 bison 将*.l 和*.y 编译成词法分析器（分析出文件中的每个单词，即记号）和语法分析器（分析记号之间的逻辑联系）的源代码 C 文件。之后，将自动生成的代码与其他部分代码（语义分析器、代码生成器等）联合编译成最终的 EXPRESS-X 转换器。而 EXPRESS-X 转换器将完成从 EXPRESS 文件到 EXPRESS-X 文件的转换任务。

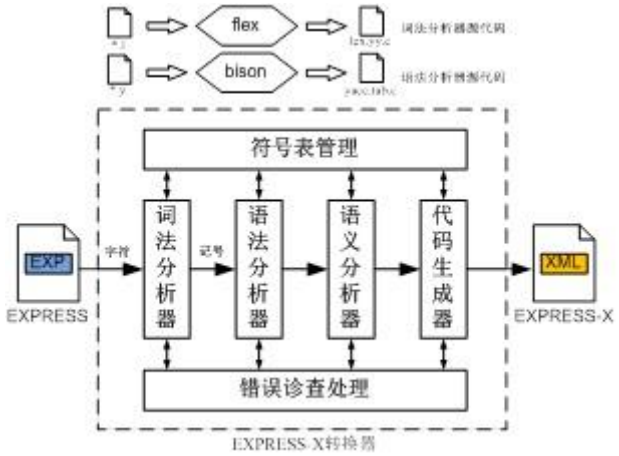

图 5 EXPRESS-X 转换器工作原理

2.3 P28 文件的生成

利用 EXPRESS-X 的信息模型定义和 XSLT 的统一过程处理，可以快速的将 P21 文件转换为 P28 文件，从而为通用的机床后置处理器生成统一的 XML 文件。在 ISO 10303-28 标准中定义了三种 P28 的文件格式，其中有两种是晚联编，一种为早联编。早联编的 P28 文件需要更小的文件存储空间和具有更快的处理速度，晚联编的 P28 文件则在通用性方面更胜

一筹。为追求更好的通用性，本文主要采用晚联编的 P28 文件格式。以下为分别用 P21 和 P28 文件格式定义一个笛卡尔坐标点的举例。

#66=CARTESIAN_POINT('CLAMPING_POSITION1',(0.000,20.000,25.000));

图 6 笛卡尔坐标点的 P21 和 P28 格式定义

在读取 P21 文件过程中，XSLT2.0 支持非 XML 的输入，并可以用正则表达式对非 XML 文档的内容进行解析。然而，到目前为止，由于 XSLT2.0 的技术还未发展成熟，各 XSLT 处理机提供商在支持上还存在很大的分歧，许多处理机内核还比较保守的只支持 XSLT1.0（如 Microsoft 的 MSXML）。近几年，尽管出现了一些对 XSLT2.0 支持的处理机（如 Saxon 9.3）等。而在这些 XSLT2.0 的处理机中，XSLT2.0 的正则表达式尚不能处理 P21 文件的嵌套结构时，即不支持递归的正则表达式。（所谓递归的正则表达式指的是对另一部分子表达式的引用，在处理嵌套的分隔记号时非常必要）

为解决上述问题，本文构建了一个辅助性的预处理程序，可以将 P21 文件转换为如下的 XML 中间格式。采用这种中间格式作为 XSLT 处理机的输入，以完成 P21 到 P28 文件的转换处理过程。

图 7 笛卡尔坐标点的 P21 和 P28 格式定义

3 面向机床的后置处理过程

3.1 机床处理机样式表

使用 XSLT 可以很轻松的定义各个不同机床控制器的代码格式，从而满足通用后置处理

过程。例如，G01 的后置处理过程可以采用如下语句完成。

```
- <sentence>
  - <index>
      <xsl:text></xsl:text>
      <xsl:value-of select="ex:GetIndexNumber()" />
      <xsl:text />
    </index>
    <key>G01</key>
  - <parameters>
      <xsl:value-of select="ex:CoordinatesFormat(../Cartesian_point[@id=$point]/@Coordinates)" />
    </parameters>
  </sentence>
```

图 8 G01 的后置处理过程语句

其中，样式表语言调用两个内嵌的脚本函数“ex:GetIndexNumber”和“ex:CoordinatesFormat”，分别获取 G 代码的语句号和坐标编码字符串。函数的定义如下：

```
int index = 0;
public String GetIndexNumber()
{
   index++;
   return "N"+index.ToString();
}
public String CoordinatesFormat(String coordinates)
{
   string[] temp = coordinates.Split(delim.ToCharArray());
   double x = Convert.ToDouble(temp[1]);
   double y = -Convert.ToDouble(temp[0]);
   double z = Convert.ToDouble(temp[2])-14.7;
   return "X" + x.ToString("0.0000") +
      " Y" + y.ToString("0.0000") +
      " Z" + z.ToString("0.0000");
}
```

3.2 利用 C#脚本语言的坐标变换

在 XSLT 中可以支持 C#脚本语言的处理。这就给后置处理过程提供了更为灵活的调节性。例如，零件的起刀原点、安装位置和安装角度等在不同的装夹过程一般都会不尽一致，使用脚本语言可以对 STEP-NC 刀具路径进行适当的坐标变换，以适应不同的装夹情况。由于零件编程是按照零件坐标系和特征坐标系进行的，而在机床加工过程中，主要是以机床坐标系作为参考系，所以后置处理时应分别将特征坐标系和零件坐标系的刀具路径数据转化为机床坐标系。图 9 为 STEP-NC 中定义的各个坐标系之间的关系[12]。

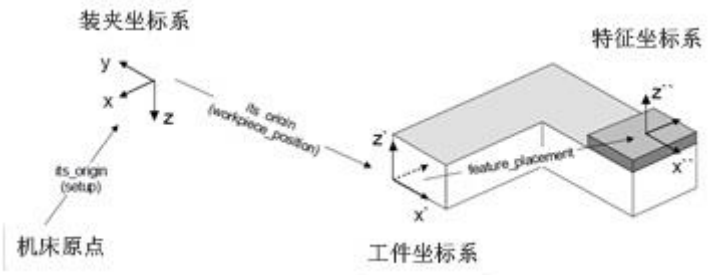

图 9 STEP-NC 坐标系之间的关系

采用 C#脚本语言编程可以方便对各坐标系的变换进行配置。例如，3.3.1 小节的 CoordinatesFormat 函数中，对刀具路径的坐标系进行了绕 Z 轴旋转 90° 和沿 Z 轴负方向平移 14.7mm 的坐标变换操作。

3.3 后置处理器实现方法

本文采用 C#编写开放式通用机床后置处理器程序。程序界面如图 10 所示，首先加载处理机样式表对后置处理器进行专用化配置，然后输入 XML 模式下的 STEP-NC 代码。经过转换处理后，程序能够自动生成目标机床代码（G 代码或其他）。

图 10 开放式 STEP-NC 通用机床后置处理器

4 测试与实验

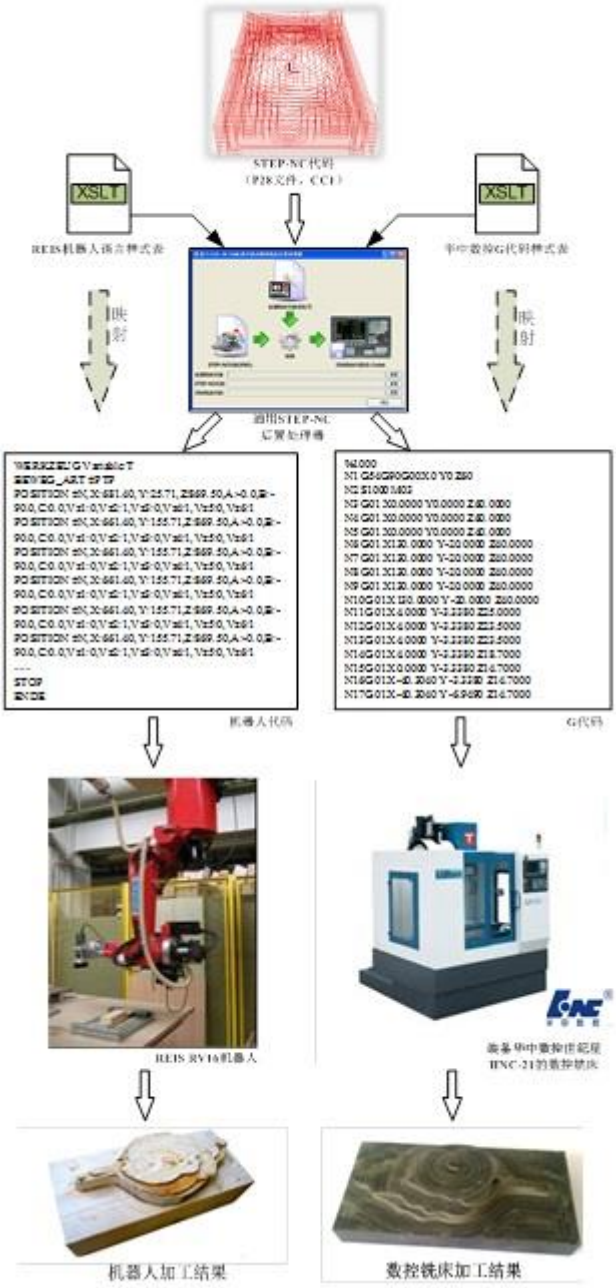

图 11 后置处理器测试实验

为验证本文提出的通用 STEP-NC 后置处理器，本文以一台加工机器人和一台三轴数控铣床。加工机器人见图 11，是将工业机器人在铣削加工领域的应用。三轴数控采用的是一台使用华中数控世纪星 HNC-21 的数控铣床。目前，绝大部分的工业机器人仅支持专用的机器人语言输入，而不支持 G 代码的输入。本文使用的是 REIS 机器人公司生产的 RV16 机器人，其机器人语言示例见图 11。经过 3.2 小节的后置处理程序对 STEP-NC 代码处理之后的相应代码经过加工机器人和数控铣床的执行之后，最终完成了同一零件的加工。后置处理过程和加工结果见图 11。

5 结论

本文提出了一种通用的 STEP-NC 后置处理器的处理模式和工作原理。采用 XML 模式下的 STEP-NC 代码作为系统输入，利用 XSLT 处理机样式表对目标机床的代码格式进行定义，从而对后置处理过程进行统一化。根据基于 XSLT 的后置处理过程的要求，本文开发了生成 XML 模式下的 EXPRESS 文件的 EXPRESS-X 转换器，P21 文件向 P28 文件的转换器以及通用的 STEP-NC 后置处理程序。本文提出的后置处理过程不仅适用于数控机床，也适用于非 G 代码输入格式的加工设备（如切削加工机器人）。以一台铣削加工机器人和一台三轴数控铣床为例，本文测试了通用后置处理器的功能性和可行性。本文提出的后置处理方法为使用 STEP-NC 对加工设备的接口统一工作具有重要意义。它验证了使用 XML 作为传统机床统一接口的可行性和证明了使用 XSLT 对后置处理过程配置的优势。本文提出的通用后置处理器构建方法大大的降低了 STEP-NC 后置处理的开发工作量和开发难度，对未来 STEP-NC 的推广和实施具有重要的现实意义。

参考文献

[1] SUH S H, KANG S K, CHUNG D H, et al. Theory and Design of CNC Systems[M]. Springer Verlag London Limited, 2008.

[2] 杜鹃，闫献国，田锡天，等. 面向铣削加工的 STEP-NC 文件到 G 代码转换技术[J]. 计算机集成制造系统, 2010, 16(1): 188-194.

[3] SEUNG J S, SUK H S, IAN S. Reincarnation of G-code based part programs into STEP-NC for turning applications [J].Computer-Aided Design, 2007, 39(1):11-16.

[4] LEE W, BANG Y B, RYOU M S, et al. Development of a PC-based milling machine operated by STEP-NC in XML format[J]. International Journal of Computer Integrated Manufacturing, 2006, 19(6):593-602.

[7] 蔡长涛. 基于 STEP/XML 的集成化工艺信息描述方法研究[J]. 计算机集成制造系统. 2008, 14(5):912-918.

[6] ISO. ISO 10303-28: Industrial automation systems and integration – Product data representation and exchange – Part 28: Implementation methods: XML representations of EXPRESS schemas and data, using XML schemas[S], 2003.

[7] WHITE C. XSLT 从入门到精通[M]. 电子工业出版社, 2003.

[8] ISO. ISO 10303-21: Industrial automation systems and integration – Product data representation and exchange – Part 21: Implementation methods: Clear text encoding of the exchange structure[S], 1994.

[9] 肖文磊，郇极. 切削加工机器人与 CAD/CAM 系统集成化[J]. 机械工程学报, 2011, 47(15):52-60.

[10] ISO. ISO/IS 10303-238 Industrial automation systems and integration – Product data representation and exchange – Part 238: Application protocol: Application interpreted model for computerized numerical controllers[S], 2006.

[11] LEVINE J. flex and bison[M]. O' Reilly Media, Inc., 2009.

[12] ISO. ISO 14649-10: Data Model for Computerized Numerical Controllers Part 10: General Process Data[S],

2003.