



# 机器人原理与应用

Principles and Applications of Robotics

**Instructor:** Dr. 阎华松(Huasong Min), professor

**Office Location:** 武钢楼1110(Room No. 1110, WUGANG Building)

**Class venue:** Room No. F4502

**Email:** [mhuasong@wust.edu.cn](mailto:mhuasong@wust.edu.cn)

**Mobile:** 13971365898

# Textbooks

- Textbook:
  - **Robot Modeling and Control**, by M.W. Spong, S. Hutchinson, M. Vidyasagar (2005) (required)  
机器人建模与控制 (中文译本)
  - **Springer handbook of robotics**. Siciliano, Bruno, and Oussama Khatib, eds. Springer, 2016.  
机器人手册 (中文译本)
  - **Modelling and Control of Robot Manipulators** (Second Edition), L. Sciavicco and B. Siciliano, Springer-Verlag, London, 2000.
  - **Robotics: Modelling Planning and Control**, B. Siciliano, L. Sciavicco, L. Villani, G. Oriolo, Springer-Verlag, London, 2008.
  - **Modern Robotics. Mechanics, Planning and Control**(现代机器人生学)



# CONTENTS

The course is divided into eight modules covering the following areas:

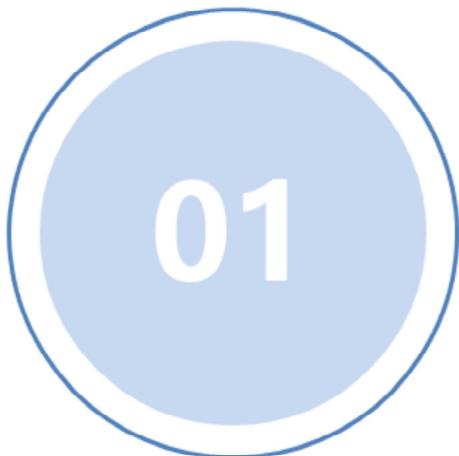
-  **绪论**  
Introduction and Conceptual Problems
-  **机器人系统分析基础**  
System Model of Robot
-  **运动学**  
Robot Kinematics
-  **动力学**  
Robot Dynamics
-  **机器人运动规划**  
Robot Motion Planning
-  **机器人控制**  
Robot Control
-  **机器人编程语言**  
Programming Language of Robot
-  **典型机器人系统的设计与实现**  
Design and Implementation of Robot System

This is an introductory of robotics course, containing both fundamental as well as some more advanced concepts. It presents a broad overview of robotics with focus on manipulators and mobile robots, and includes robot kinematics, dynamics, planning and control, programming language.

The course is divided between the following areas:

- Robotics Introduction
- System Model of Robot
- Robot kinematics
- Robot dynamics
- Robot control
- Robot motion planning

...



## 引言

机器人在发展初期，其功能单一，能完成的动作也较简单，其控制程序是固定的运动控制。随着机器人作业动作的多样化和作业环境的复杂化，依靠固定的程序或简单的示教方式已满足不了要求，机器人必须能够快速完成再编程，以适应作业和环境的变化。

随着机器人技术与应用的广泛发展，各家公司均推出了具有自己特色的机器人语言。

## 机器人语言发展历史

- George Devol 于 1954 年发明了第一台工业机器人Unimate，该机器人使用了较为低级的再编程概念，采用一种较为原始的示教方式；
- 1960年，麻省理工学院为其机械手MH-I开发的MHI (Mechanical Hand Interpreter，机械手解析器)可称为第一个机器人语言；
- 1973年，美国斯坦福大学研制出了世界上第一种通用机器人语言WAVE；
- 1974年，斯坦福大学人工智能实验室在WAVE语言的基础上开发出了AL语言；
- 1975年，IBM公司推出了用于机器人装配作业的AML(A Manufacturing Language)语言，随后该公司又研制出另一种语言-AUTOPASS语言；
- 1979年，美国Unimation公司推出了VAL语言；
- 20世纪80年代初，美国Automatix公司开发了RAIL语言；同时，麦道公司研制了MCL语言；
- .....

George Devol 于 1954 年发明了第一台工业机器人Unimate，该机器人使用了较为低级的再编程概念，采用一种较为原始的示教方式：用户命令机器人到一个位置，然后存储所有关节的位置。示教完成后，再现程序时，关节从初始位置移动到示教位置。1960年，麻省理工学院为其机械手MH-I开发的MHI (Mechanical Hand Interpreter，机械手解析器)可称为第一个机器人语言。MH-I 的整个手部配备了多个二进制触摸传感器、手指之间的压力传感器阵列以及手指底部的光电二极管，MHI程序在 TX-0 计算机上解析执行。MHI编程语言基本指令为移动指令“move”，即移动直到检测到传感器条件成立。MHI 基本语句如下：

- 1) “move”：表示方向和速度；
- 2) “until”：在某个指定条件下测试传感器；
- 3) “ifgoto”：如果检测到某个条件，则跳转到程序标签；
- 4) “ifcontinue”：如果某些条件成立，则分支以继续操作。

1973年，美国斯坦福大学研制出了世界上第一种通用机器人语言WAVE。WAVE是一种机器人动作语言，其语言功能以描述机器人的动作为主，包含力和接触的控制，且能配合视觉传感器进行机器人的手、眼协调控制。1974年，斯坦福大学人工智能实验室在WAVE语言的基础上开发出了AL语言。AL语言与高级计算机语言ALGOL结构相似，是一种编译形式的语言，带有一个指令编译器，能在实时机上控制，用户编写好的机器人语言源程序经编译器编译后对机器人进行任务分配和作业命令控制。AL语言不仅能描述手爪的动作，而且可以记忆作业环境和该环境内物体与物体之间的相对位置，实现对多台机器人的协调控制。

1975年，IBM公司推出了用于机器人装配作业的AML(A Manufacturing Language)语言，随后该公司又研制出另一种语言-AUTOPASS语言，这是一类用于装配的高级语言，它可以对几何模型类任务进行半自动编程。

1979年，美国Unimation公司推出了VAL语言。它是在BASIC语言基础上扩展的一种机器人语言，具有BASIC的内核与结构，编程简单，语句简练。VAL语言成功地应用于PUMA和UNIMATE型机器人。1984年，Unimation公司在VAL基础上，推出了改进的机器人语言-VAL II语言。VAL II语言除了含有VAL语言的全部功能外，还增加了对传感器信息的读取，使得机器人可以利用传感器信息进行运动控制。

20世纪80年代初，美国Automatix公司开发了RAIL语言，该语言可以利用传感器的信息进行零件作业的检测。同时，麦道公司研制了MCL语言，这是一种在数控自动编程语言-APT语言基础上发展起来的一种机器人语言。MCL特别适用于由数控机床、机器人等组成的柔性加工单元的编程。

## 机器人编程语言举例

目前在全球占据最大市场份额的机器人“四大家族”包括：

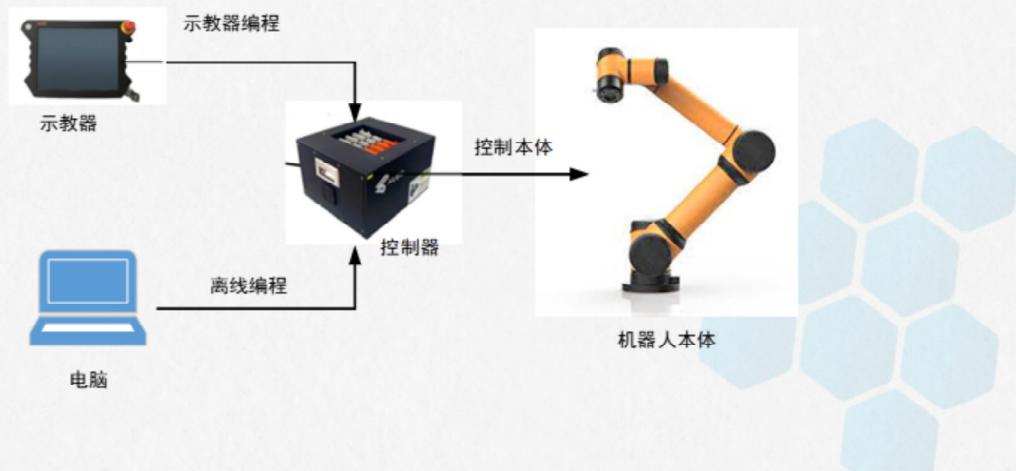
- ABB（瑞士）
- YASKAWA（安川、日本）
- KUKA（德国）
- FANUC（发那科、日本）

他们都有自己特定的机器人语言！

序号	语言名称	国家	研究单位	简介	13	LM	美	Artificial Intelligence Group of IMAG	类似PASCAL，数据类似AL。用于装配机器人（用LS11/3微型机）
1	AL	美	Stanford Artificial Intelligence Laboratory	机器人动作及对象语言研究的基础	14	ROBEX	美	Machine Tool Laboratory TH Archen Olivetti	具有与高级NC语言EXAPT相似结构的脱机编程语言
2	Autopass	美	IBM	组装机器人用语言	15	SIGLA	美	SIGMA机器人语言	
3	LAMA-S	美	MIT	高级机器人语言	16	MAL	美	Milan Polytechnic	两臂机器人装配语言，其特征是方便、易于编程
4	VAL	美	Unimation公司	用于PUMA机器人（用Z-11高级微型机）	17	SERF	美	三协精机	SKILAM装配机器人（用Z-80微型机）
5	RAIL	美	Automatix公司	用视觉传感器检查	18	PLAW	美	小松制作所	RW系列弧焊机器人
6	WAVE	美	Stanford Artificial Intelligence Laboratory	操作器控制符号语言	19	IML	美	九州大学	动作级机器人语言
7	DIAL	美	Charles Stark Draper Laboratory	具有RCGI顺应性手腕	20	KAREL	日本	FANUC	发那科研发的用于点焊、涂胶、搬用等工业用途的编程语言
8	RPL	美	Stanford Research Institute International	可与Unimate机器人/义子程序库	21	RAPID	瑞典	ABB	ABB公司用于ICRS控制器示教器的编程语言
9	REACH	美	Bendix Corporation	适于两臂协调动作广泛的语言	22	Robotics Studio	美国	Microsoft	微软公司开发的多语言、可视化编程与仿真语言
10	MCL	美	McDonnell Douglas Corporation	编程机器人、机床控制的计算机综合制造	23	INFORM	日本	YASKAWA	安川机器人专用编程语言
11	INDA	美	SRI International and Philips	相当于RTL/25编程语言的处理系统	24	KUKA	德国	KRL	库卡机器人专用编程语言
12	RAPT	美	University of Edinburgh	类似NC语言APT（用					

随着机器人技术与应用的广泛发展，各家公司均推出了具有自己特色的机器人语言，表7.1列出了大部分的工业机器人编程语言。目前在全球占据最大市场份额的机器人“四大家族”，包括ABB（瑞士）、YASKAWA（安川、日本）、KUKA（德国）、FANUC（发那科、日本），它们以及一些主流机器人公司都已经发展出了各自相对成熟的机器人编程语言、控制命令，其语法规则也已经相对完善。即使是一些新型机器人公司，例如丹麦的UR机器人和意大利的COMAU机器人等，也已经开发出了对应于自家机器人的独特机器人编程语言。

## 机器人编程模式



在图7.1所示的机器人编程控制结构中，机器人编程可以在专用的示教器 (teach pendant)上进行，也可能是在电脑上离线完成编程。由机器人语言编写的专用程序通过控制器里面特定的解析器，解析执行，并控制机器人本体运动。控制器和机器人本体之间的控制接口，按照机器人本体构成类型，可分成集中控制型与总线控制型。所谓集中控制型，即由控制器提供各关节伺服控制接口，传统的工业机器人大都是这种结构；而新型的协作机器人、模块化轻型机器人，则采用总线控制型，机器人的每个关节都有单独的控制与伺服接口，总控制器通过总线（RS485、CAN、Ethercat等）和关节控制器通信，控制机器人本体运动。

02

## 典型工业机器人语言简介

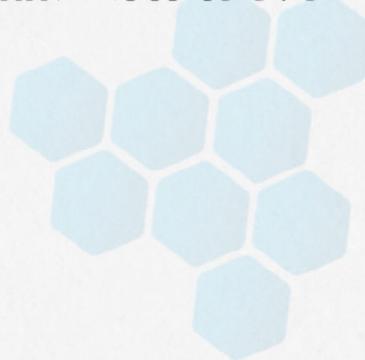
INFORM语言

RAPID语言

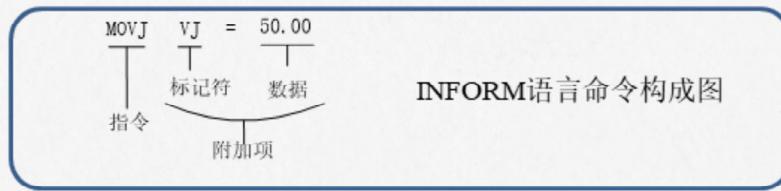
KRL语言

KAREL语言

URScript语言



## INFORM语言



INFORM语言命令构成图

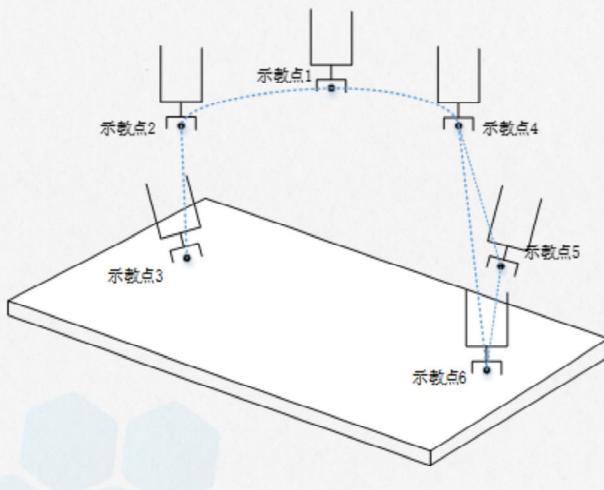
表7.2 INFORM编程语言指令表

命令类型	内容	举例
输入输出(I/O)指令	执行输入输出的指令	DOUT, WAIT
控制指令	执行处理和作业控制的指令	JUMP, TIMER
运算指令	使用变量等进行运算的指令	ADD, SET
移动指令	与移动和速度相关的指令	MOVJ, REPP
替换指令	表示当前示教点被替换的指令	SFTON,SFTOF
条件指令	条件执行指令	IF,UNTIL
操作指令	末端工具操作相关指令	TOOLON

**INFORM:** 日本安川机器人采用的机器人编程语言，该语言的机器人命令由指令和附加项构成，其中附加项包括标记符和数据。图7.2为INFORM语言的一个基本例子。

## INFORM语言

表7.3 INFORM语言示教程序及说明



行号	指令	解释
0000	NOP	开始行，空指令
0001	MOVJ VJ=25.00	以关节插补方式移动到示教点1(等待位置)，再现速度为25%
0002	MOVJ VJ=25.00	以关节插补方式移动到示教点2（接近位置），再现速度为25%
0003	MOVL V=100.0	以直线插补方式移动到示教点3（抓取位置），再现速度为100cm/min
0004	HAND 1 ON	抓取工件
0005	TIMER T=0.50	延时等待工件抓取完毕
0006	MOVL V=100.0	以直线插补方式移动到示教点2（接近位置），再现速度为100cm/min
0007	MOVJ VJ=25.00	以关节插补方式移动到示教点1(等待位置)，再现速度为25%
0008	MOVJ VJ=25.00	以关节插补方式移动到示教点4(接近释放位置)，再现速度为25%
0009	MOVL V=100.0	以直线插补方式移动到示教点5（辅助释放位置），再现速度为100cm/min
0010	MOVL V=50.0	以直线插补方式移动到示教点6(释放位置)，再现速度为50cm/min
0011	HAND 1 OFF	释放工件
0012	TIMER T=0.50	等待至工件释放完毕
0013	MOVL V=100.0	以直线插补方式移动到示教点4(接近释放位置)，再现速度为100cm/min
0014	MOVJ VJ=25.00	以关节插补方式移动到示教点1(等待位置)，再现速度为25%
0015	END	程序结束

## RAPID语言

### TPWrite String [\Num] | [\Bool] | [\Pos] | [\Orient] [\Dnum]

RAPID语言元素表

元素类型	内容说明	举例
标识符	对模块、程序、数据和标签命名	PROC, VAR
保留字	作为特殊意义的标识符	ALIAS, AND
空格和换行符	在标识符，保留字，数值中不能使用	
数值	数值有整数小数两种表示方式	
逻辑值	与C中的bool量类似	TRUE, FALSE
字符串	字符串的最长长度为80个字符	"This is a string"
注释	帮助理解程序，以“!”开始	! comment
单字符	包括一些冰岛字母和带重音字母	

RAPID是一种ABB公司机器人使用的高级编程语言，支持分层编程方案，可为特定机器人系统安装新程序、数据对象和数据类型，能对编程环境进行自定义（扩展编程环境的功能），并获得RAPID编程语言的充分支持。该语言可以很好的运行在Robot Studio开发环境中，就像C语言在Visual Studio中调试运行一样。基本的RAPID编程语言有自己的程序结构、程序数据和表达式。

RAPID编程语言内容较为繁多，下面只对该语言进行简单描述。

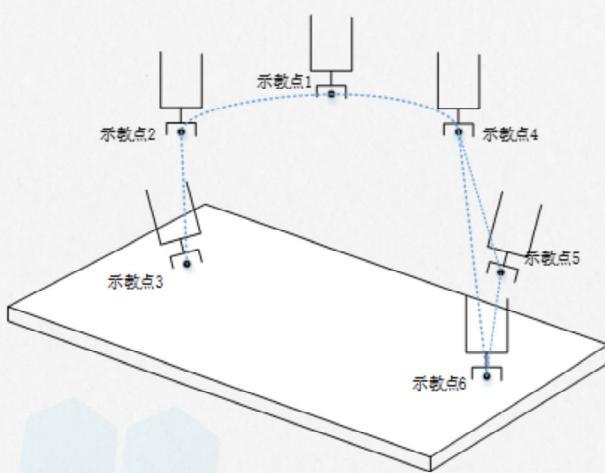
一个RAPID语言的简化语法示例如下：

### TPWrite String [\Num] | [\Bool] | [\Pos] | [\Orient] [\Dnum]

在上面的示例中，括号里不含强制性参数，用方括号将可选参数括起来，但可忽略这些参数。互相排斥的参数不能同时存在于同一指令中，在同一指令中就要用竖线隔开。用波形括号将可重复任意次的参数括起来。

上述的示例采用了如下参数：String为强制性参数；Num、Bool、Pos、Orient和Dnum为可选参数；Num、Bool、Pos、Orient和Dnum互相排斥。表7.4为RAPID编程语言部分基本元素列表。

## RAPID语言



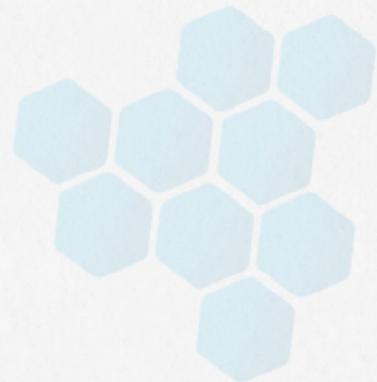
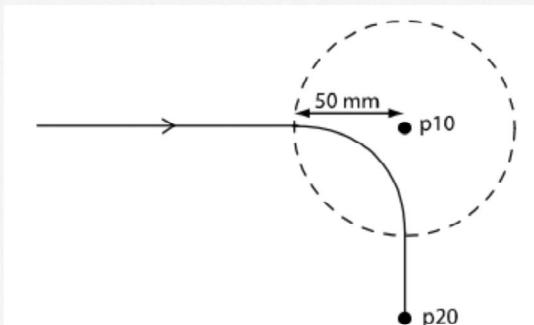
```
MODULE Module1
! The tool tool0 defines the wrist coordinate system, with the origin being the centre of
the mounting flange.
PERS tooldata tool0 := [ TRUE, [[0, 0, 0], [1, 0, 0, 0]], [0.001, [0, 0, 0.001], [1, 0, 0, 0], 0, 0,
0] ];
! Instructions for creating positions: P1 - P6
CONST robtarget P1:=[...];
CONST robtarget P2:=[...];
CONST robtarget P3:=[...];
CONST robtarget P4:=[...];
CONST robtarget P5:=[...];
CONST robtarget P6:=[...];
PROC main()
MOVEJ P1, v1000, fine, tool0;
MOVEJ P2, v1000, fine, tool0;
MOVEL P3, v1000, fine, tool0;
Set Gripper;
WaitTime 0.5;
MOVEL P2, v1000,fine,tool0;
MOVEJ P1, v1000,fine,tool0;
MOVEJ P4, v1000,fine,tool0;
MOVEL P5, v1000,fine,tool0;
MOVEL P6, v500,fine,tool0;
Reset gripper;
WaitTime 0.5;
MOVEL P4, v100,fine,tool0;
MOVEJ P1, v100,fine,tool0;
ENDPROC
ENDMODULE
```

RAPID语言用! 代表注释行。表7.5中的程序定义了一个程序块“Module1”，“Module1”中定义了一个主函数main()。程序首先定义了手腕坐标系tool0，其原点为机器人末端安装法兰的中心。接下来定义了6个作业点P1-P6，MOVEL为直线插补运动指令，MOVEJ为关节插补运动指令，运动指令中“v1000”代表TCP速度为1000mm/s,“fine”参数代表机器人直接运动到指定点，不进行转角的圆弧过渡（不指定交融半径）。

## RAPID语言

```
MoveL p10, v1000, z50, tool0;
```

```
MoveL p20, v1000, fine, tool0;
```



在移动到P10作业点的运动指令中，指明了转角切除半径为50mm

## KRL语言 类似Pascal

KRL主要数据类型表

数据类型	关键词	含义	值的范围
整数	INT	整数	-231-1到231-1
实数	REAL	浮点数	±1.1-38到±3.438
布尔值	BOOL	逻辑状态	真假
字符	CHAR	字符	ASCII字符

KRL语言主题类型表

主题类型	内容说明	举例
移动	库卡机器人可以有三种主要方式从一个点移动到另外一个点	PTP (点到点) LIN (线性) CIRC (圆形)
输入和输出	输入和输出与PLC中的I/O概念相同	\$ IN [33] \$ OUT [33]
执行控制	包括条件语句，切换语句，循环语句	IF, SWITCH, FOR
变量	包括整数，实数和E6POS变量（表示位置和姿态，INT, REAL, E6POS有6个值）	

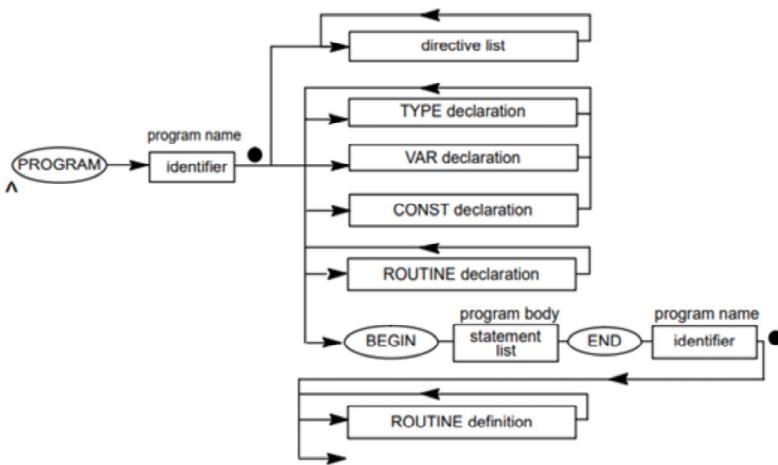
KRL为德国KUKA机器人的编程语言，为Kuka Robot Language三个英文单词的首字母缩写而成，这是一种与Pascal类似的专有编程语言。

任何的KRL代码由两个不同的文件组成，一个是永久数据文件，拓展名为.dat，另一个是移动命令文件，拓展名为.src，KRL有四种常见的数据类型，如表7.6所示。

KRL语言有很多种方法给机器人下达操作命令，其语言的编程方式是基于Topic主题的编程方式。KRL具有五个主题，每个主题里面都包括了相应的机器人指令集以及语法规则，表7.7只叙述了KRL编程语言的一部分主题类型和说明。

## KAREL语言

## FANUC



KAREL语言程序模块定义

```

PROGRAM prog_name
Translator Directives
CONST, TYPE, and/or VAR Declarations
ROUTINE Declarations
BEGIN
Executable Statements
END prog_name
ROUTINE Declarations
  
```

KAREL语言由Richard E. Pattis在他的一本书《Karel The Robot: A Gentle Introduction to the Art of Programming》中提出。Pattis在其斯坦福大学的课堂上使用这门语言向学习机器人学的学生讲解机器人编程。KAREL是以提出“机器人”一词的捷克作家卡雷尔-卡佩克的名字命名的一种机器人语言，后来发展成为用于FANUC机器人的一种主要编程语言。KAREL语言是一种离线编程语言，融合了类Pascal和PL/1的高级编程语言逻辑，采用了通常的程序结构以及机器人专用功能。KAREL编程语言的特征包括：

- 简单和结构化的数据类型
- 算术、关系以及布尔运算符
- 循环结构、选择结构
- 条件处理
- 过程和函数
- 输入输出
- 多编程支持

采用KAREL编程，程序申明和执行部分代码存放在源代码文件中，而变量数据存放在另外的变量文件(.vr)中。源代码必须通过编译成内部代码(.pc)，被称为*p-code*，才能被执行。

FANUC的KAREL程序，其执行体以关键词**BEGIN**开始，以**END**结束。

## URScript语言

URScript模块命令表

模块类型	模块内容	举例
运动模块	该模块包含了UR的运动命令	movec(pose_via,pose_to,a,v,r), 圆形移动，其中pose_via: 路径点；pose_to: 目标姿态；a: 工具加速度；v: 工具速度；r: 交融半径。
内部模块	该模块包含了UR的监控命令	get_actual_joint_positions(), 返回所有关节的实际角位置。
运算模块	该模块包含了UR的运算命令	acos(f), 返回f的反余弦主值。
界面模块	该模块包含了UR的I/O命令	get_analog_in(n), 获取模拟输入电平，其中n为输入的编号。

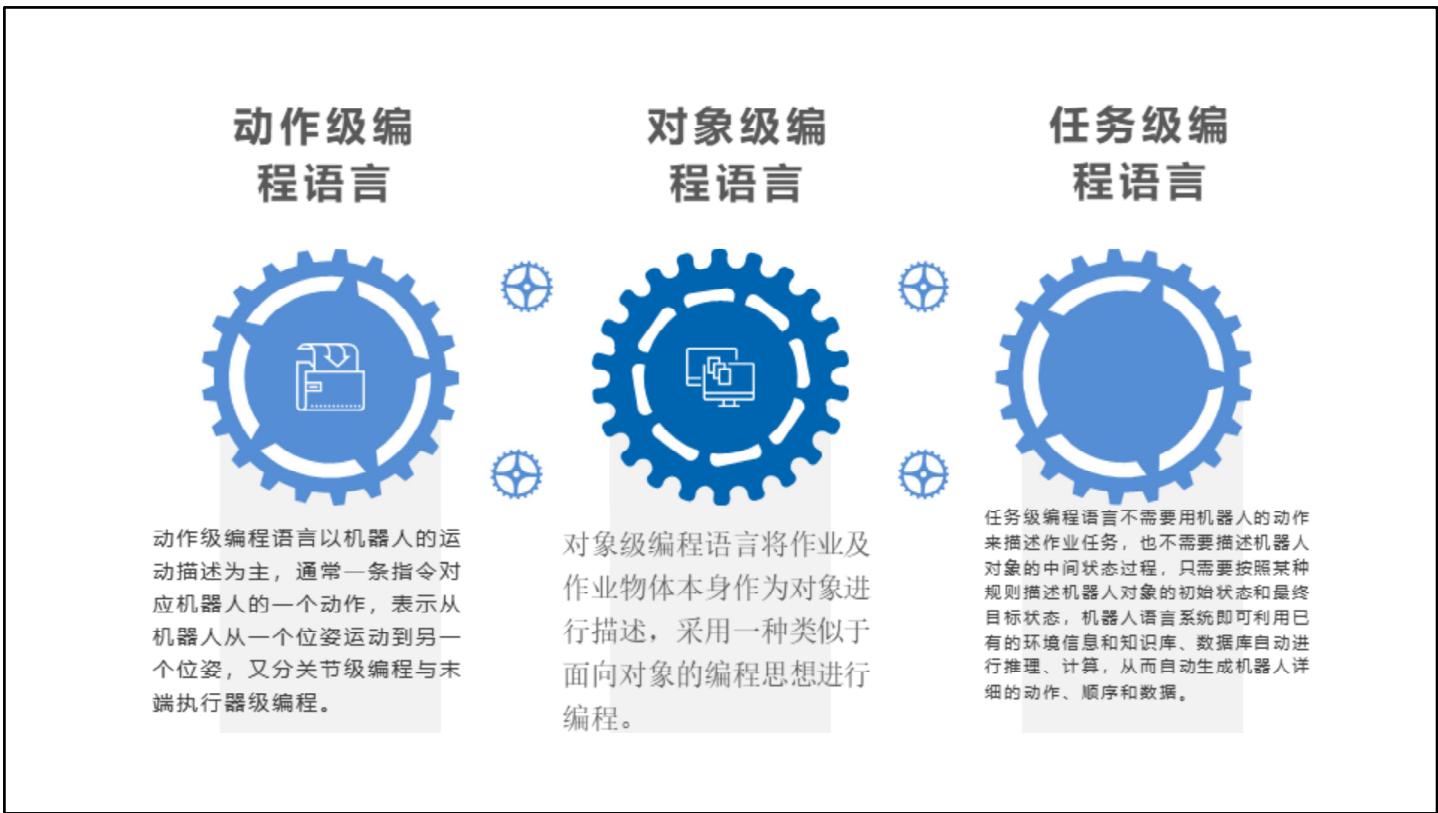
**URScript:** 丹麦的Universal Robot公司开发的新型协作机器人-UR系列机器人使用的编程语言，这是一个在脚本层上控制机器人的编程语言，该语言是UR公司在Python语言基础上独立研发的。URScript编程语言就像其他编程语言一样，它有变量类型，语法结构、方法等。另外，它还有一系列的专用方法来控制机器人运动以及IO状态。

URScript编程语言与大多数的计算机语言类似，也有数字、变量、类型、控制流、函数、进程等。该语言将机器人控制划分为运动模块、内部模块、数学模块和界面模块。该语言控制机器人采用模块化的命令，即该语言将机器人控制划分为运动模块、终端模块、运算模块、接口模块。

# 03

## 机器人语言的分类

按照机器人作业描述水平的程度，我们可以将现有的工业机器人语言分为动作级编程语言、对象级编程语言以及任务级编程语言三类。



### 7.3.1 动作级编程语言

最初的工业机器人编程语言基本都是面向作业动作进行描述的。动作级编程语言以机器人的运动描述为主，通常一条指令对应机器人的一一个动作，表示从机器人从一个位姿运动到另一个位姿（按照现代机器人学的描述习惯，可改为位形进行描述）。

早期的机器人语言基本为动作级编程语言，例如VAL语句中“MOVE TO (destination)”的含义为：将机器人从当前位姿运动到目的位姿。动作级编程语言比较直观且容易编程，缺点就是功能受限。一般无法进行复杂的数学运算，也不能处理浮点数和字符串，子程序不能含有自变量，除了开关信息之外，无法处理复杂的传感器信息，与计算机通信的能力较差。

采用动作级编程语言编程时，需要注意参考坐标系的设立。其编程可分为关节级(采用相对坐标系-关节坐标系)编程和末端执行器级(采用绝对坐标系-笛卡尔坐标系)编程两种。

**关节级编程**以机器人的关节为对象，编程时给出机器人一系列关节位置的时间序列，在关节坐标系中进行编程。对于直角坐标型机器人和圆柱坐标型机器人，由于直角关节和圆柱关节的表示比较简单，这种方法编程较为适用；而对于具有旋转关节的关节型机器人来说，由于关节位置的时间序列表示困难，即使一个简单的动作也要经过许多复杂的运算，故采用关节级编程，计算就会较为复杂。

关节级编程可通过简单的编程指令来实现，也可以通过示教器示教和键入示教实现。

**末端执行器级编程**指在绝对坐标系下进行的编程，即参考坐标系为机器人作

业空间的笛卡尔坐标系（直角坐标系）。在此直角坐标系中给出机器人末端执行器一系列位姿组成的位姿时间序列，连同其他一些辅助功能如力觉、触觉、视觉等时间序列，同时确定作业量、作业工具等，协调地进行机器人动作的控制。

这种编程方法允许有简单的条件分支，可以有感知功能，可以选择和设定工具，有时还可进行并行处理，数据实时处理能力比关节级编程模式强。

随着机器人语言解析技术的发展，在很多的工业机器人厂家推出的编程系统中，用户可以在关节坐标系和末端执行器坐标系等参考坐标系之间进行切换编程。

### 7.3.2 对象级编程语言

对象级编程语言将作业及作业物体本身作为对象进行描述，采用一种类似于面向对象的编程思想进行编程。对象级编程语言不需要描述机器人具体操作动作，只需由编程人员用程序语句给出作业过程顺序的描述以及环境模型描述，也就是机器人与操作对象之间的关系描述，通过编译程序，机器人就能知道如何动作。

AML、AUTOPASS等语言属于对象级编程语言，其特点有：

- (1) 具备动作级编程语言的全部功能；
- (2) 有较强的感知能力，能处理复杂的传感器信息，可以利用传感器信息来修改、更新作业状态，也可以利用传感器信息进行控制、测试与监督；
- (3) 开放性较好，系统一般提供二次开发平台，用户可以根据需要进行指令增加，扩展系统功能；
- (4) 数字计算和数据处理能力强，可以处理浮点数，能与计算机之间进行实时通信。

对象级编程语言采用接近自然语言的方法描述对象变化。对象级编程语言的运算功能、作业对象的位姿时序、作业量、作业对象承受的力和力矩等都可以表达式的形式出现。系统中机器人尺寸参数、作业对象以及工具等参数等，一般以知识库或数据库的形式存在。系统编译程序时获取这些信息，并可对机器人动作过程进行仿真，实现对作业对象的位姿控制、避障以及与其他设备通信等工作。

### 7.2.3 任务级编程语言

任务级编程语言不需要用机器人的动作来描述作业任务，也不需要描述机器人对象的中间状态过程，只需要按照某种规则描述机器人对象的初始状态和最终目标状态，机器人语言系统即可利用已有的环境信息和知识库、数据库自动进行推理、计算，从而自动生成机器人详细的动作、顺序和数据。

以机器人进行螺丝装配为例，螺丝的初始位置和装配后的目标位置已知，当发出装配螺丝的命令后，语言系统从初始位置到目标位置之间自动规划路径，在复杂的作业环境中找出一条不会与周围障碍物产生碰撞的合适路径，在初始位置处选择恰当的姿态抓取螺钉，并沿此路径运动到装配位置，然后进行力矩交互控制，最终将螺丝准确地拧紧到目标位置。在此过程中，作业中间状态、作业方案的设计、工序的选择、动作的前后安排等一系列问题都由计算机自动完成。

任务级编程语言的结构较为复杂，需要人工智能的理论基础和大型知识库、数据库的支持，目前还没有完善，是一种较为理想状态下的机器人编程语言，有待于进一步的研究。

随着人工智能技术的不断发展，人机交互技术(HMI)也在不断进步。等到基于自然语言的语义理解等理论与技术的发展成熟之后，机器人与人之间将来可能会出现无障碍的交流，机器人可能无需编程即可自动完成人类交付的作业任务。

# 04

## 机器人编程语言的要求

在编程和作业过程中，编程系统应便于人与机器人之间进行信息交换，方便机器人出现故障时及时处理，确保安全。而且，随着机器人动作和作业环境的复杂程度的增加，编程系统需要提供功能强大的人机接口。

随着计算机编程语言以及人工智能的发展，机器人编程语言也有了与人工智能编程语言相结合的趋势。例如在支持人工智能的通用编程语言，如Java、python、lua等脚本语言基础上，开发出机器人的编程语言，使得机器人编程系统更加强大、更加方便开发。

## 设计机器人编程语言的基本要求

先进的机器人编程语言需要达到如下要求：

- 1) 支持多种坐标系编程
- 2) 支持基本的程序设计常用结构
- 3) 支持机器人作业描述
- 4) 支持多种运动规划
- 5) 具有良好的编程环境
  - 1) 在线修改和重启功能
  - 2) 传感器输出和程序追踪功能
  - 3) 仿真功能
  - 4) 人机接口和综合传感信号

先进的机器人编程语言需要达到如下要求：

### 1) 支持多种坐标系编程

在进行机器人编程时，除了建立绝对坐标系（世界坐标系）之外，为了方便机器人工作，还应支持建立其他坐标系，如关节坐标系、末端执行器坐标系、作业对象坐标系等参考坐标系，同时建立这些坐标系之间的转换关系。

### 2) 支持基本的程序设计常用结构

机器人编程系统应该支持基本的程序控制结构，包括简单程序、等待程序、分支程序、循环程序、子程序以至中断程序等结构化编程方法。

### 3) 支持机器人作业描述

机器人作业的描述与其环境模型密切相关，编程语言水平决定了描述水平。现有的机器人语言需要给出作业顺序，由语法和词法定义输入语句，并由它描述整个作业过程。先进的机器人编程语言可以支持任务级的作业描述，只需要按照某种规则描述机器人对象的初始状态和最终目标状态，机器人语言系统即可利用已有的环境信息和知识库、数据库自动进行推理、计算，从而自动生成机器人详细的动作、顺序和数据。

### 4) 支持多种运动规划

描述机器人需要进行的运动是机器人编程语言的基本功能之一。用户能够运用语言中的运动语句，可选择与多种路径规划器连接，允许用户规定路径上的点及目标点，决定采用何种规划算法以及避障算法。

### 5) 具有良好的编程环境

如同任何计算机系统一样，一个好的编程环境有助于提高程序员的工作效率。好的编程系统应具有下列功能：

### 1) 在线修改和重启功能

机器人在作业时需要执行复杂的动作和花费较长的执行时间，当任务在某一阶段失败后，从头开始运行程序并不总是可行，因此需要编程软件或系统必须有在线修改程序和随时重新启动的功能。

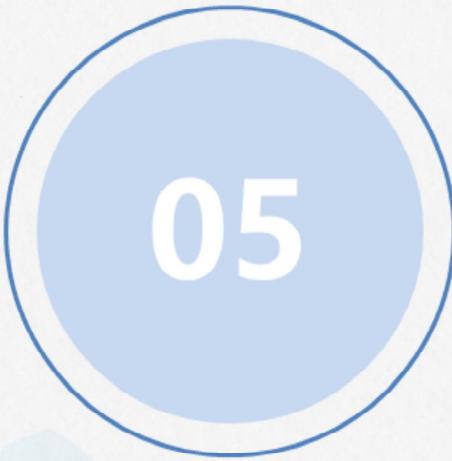
### 2) 传感器输出和程序追踪功能

因为机器人和环境之间的实时相互作用常常不能重复，因此编程系统应能随着程序追踪记录传感器的输入输出值。

### 3) 仿真功能

可以在没有机器人实体和工作环境的情况下进行不同任务程序的模拟调试。

### 4) 人机接口和综合传感信号

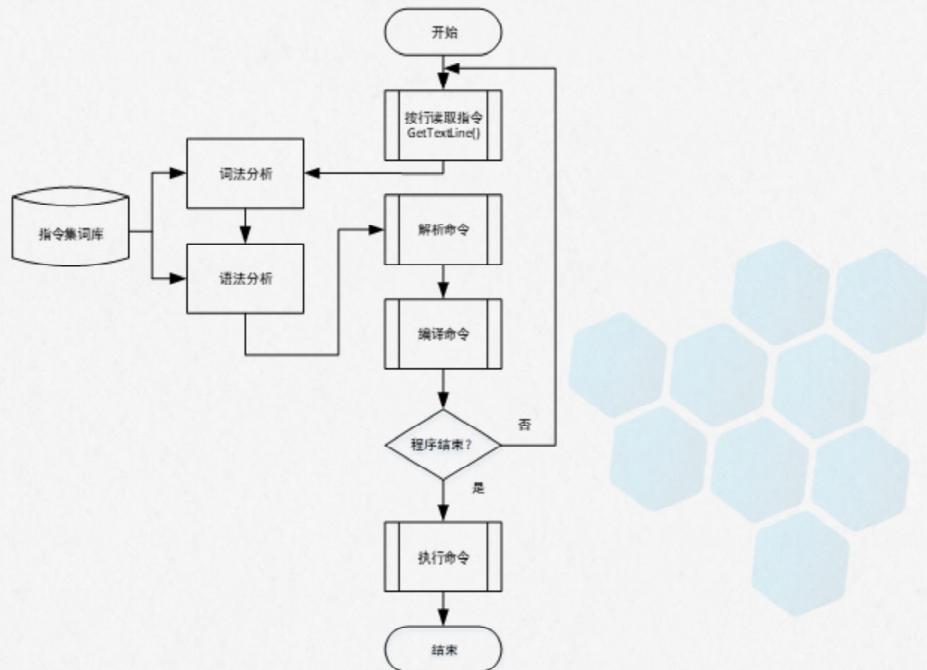


05

## 机器人语言解析器原理

机器人语言解析器的目的是为了把采用机器人语言编写  
的程序翻译成目标可执行代码，其处理过程按顺序可以  
划分为：分词处理、语法分析、语义分析、目标代码生  
成、目标代码执行等流程。

## 机器人语言解析器原理



在图7.6中，机器人解析器的实现被分为三个主要的步骤，包括机器人语言的解析过程、编译过程和执行过程，其中机器人语言解析过程中会利用到词法分析、语法分析等处理。

该解析器首先获取一行机器人指令（`getTextLine()`函数）。之后开始解析指令（`parserCmd()`函数，完整的解析过程包括词法分析`tokenizer()`、语法分析`parser()`，如果需要完成语义理解的功能，可以加入语义分析）。在指令解析完成后，可以生成一个编译矩阵。随后编译函数将获取的编译矩阵按行编译成可执行的指令矩阵(或模块化关节通信指令)。由此，程序控制主流程一直循环获取机器人指令文件直到结束。最终执行函数调用生成的指令矩阵，按行执行驱动机械臂，即可控制机械臂完成作业任务。

综合了解前面介绍过的典型工业机器人语言可以总结出：每一种机器人语言基本都包含了变量定义指令、算术指令、控制指令、状态指令以及移动指令等基本指令。

表 7.9 指令设计表

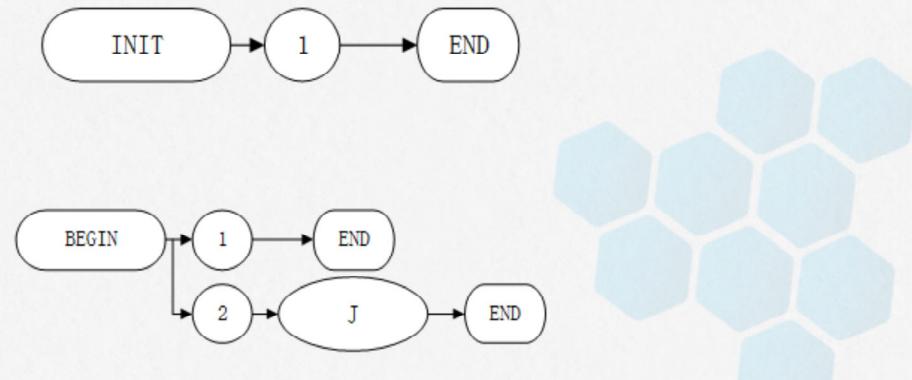
指令类型	指令内容	指令关键词
状态指令	初始化机器人状态、进入准备状态、程序结束	INIT、BEGIN、END
移动指令	从一点移动机器人末端到另一点、以直线插补从当前点移动到目标点、直接控制关节转动角度	MOVEP、MOVEL、MOVEJ
变量定义指令	字节型变量定义、整形变量定义、空间点定义、关节位置变量定义	DEFN、DEFX

综合了解前面介绍过的典型工业机器人语言可以总结出：每一种机器人语言基本都包含了变量定义指令、算术指令、控制指令、状态指令以及移动指令等基本指令。接下来，本节以一个实例讲解机器人语言解析器的实现。

为了方便起见，本教材仅介绍包含两条变量定义指令(DEFN、DEFX)、三条状态指令(INIT、BEGIN、END)、三条移动指令(MOVEP、MOVEL、MOVEJ)的最小指令集解析器的设计。

## 机器人语言指令集设计

### (1) 状态指令(INIT、 BEGIN、 END)



机器人状态指令包括初始化状态命令(**INIT**)、准备状态命令(**BEGIN**)以及程序结束指令(**END**)，初始化状态命令结构图如图7.7所示。

初始化状态命令名称：**INIT**。功能：初始化机械臂状态（变量，关节，位置）。举例如下：

**INIT**

说明：初始化机械臂状态。

准备状态命令(**BEGIN**)用法示意图如图7.8所示。

准备状态命令名称：**BEGIN**，功能：将机械臂移动至准备状态。举例如下：

**BEGIN**

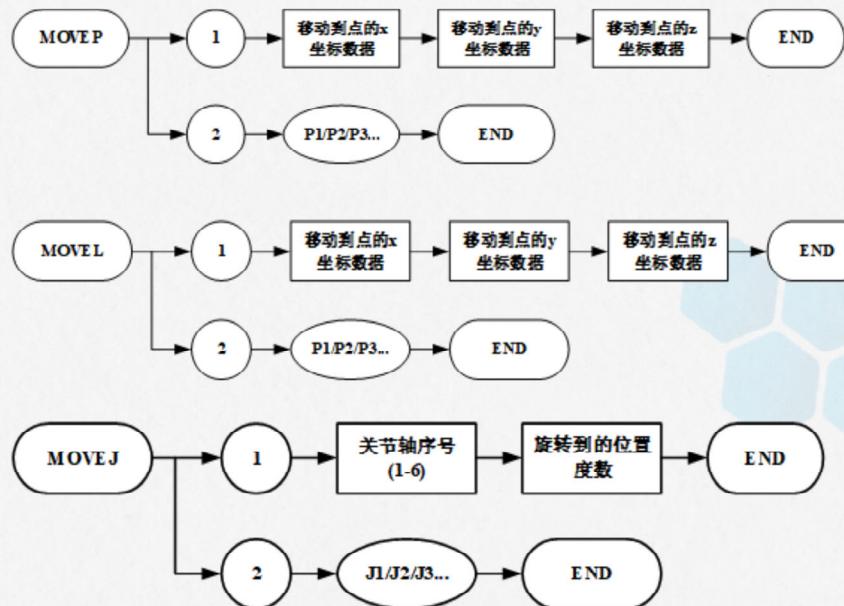
说明：将机械臂移动至准备状态。

**BEGIN J1**

说明：将J1(这里代指机械臂关节1)状态设为准备状态并移动至此。

## 机器人语言指令集设计

### (2) 移动指令(MOVEP、MOVEL、MOVEJ)



一般工业机器人移动指令包括MOVEP, MOVEL, MOVEJ三种移动指令，移动指令的总体结构图如图7.9所示。

移动指令名称：MOVEP，功能：在关节坐标系中进行点到点的轨迹规划（关节空间插补），然后控制机械臂按规划轨迹进行移动。举例如下：

**MOVEP 555 23 583**

说明：以关节空间插补的方式控制机械臂末端从当前点移动到坐标为(555, 23, 583)的点，坐标单位一般为毫米。

**MOVEP P1**

说明：以关节空间插补的方式移动到P1点。

移动指令名称：MOVEL，功能：在笛卡尔坐标系中进行直线插补，控制机械臂移动；用法示意图如图7.10所示。举例如下：

**MOVEL 555 23 583**

说明：以直线插补的方式移动到坐标为(555, 23, 583)的点，坐标单位一般为毫米。

**MOVEL P1**

说明：以直线插补的方式移动到P1点。

移动指令名称：MOVEJ，功能：直接控制机械臂的关节进行转动，用法示意图如图7.11所示。举例如下：

**MOVEJ 4 45**

说明：控制机械臂4号关节转至45度位置。

**MOVEJ J1**

说明：直接控制所有关节转动到J1状态。

## 机器人语言指令集设计

### (3) 变量定义指令 (DEFN、DEFX)

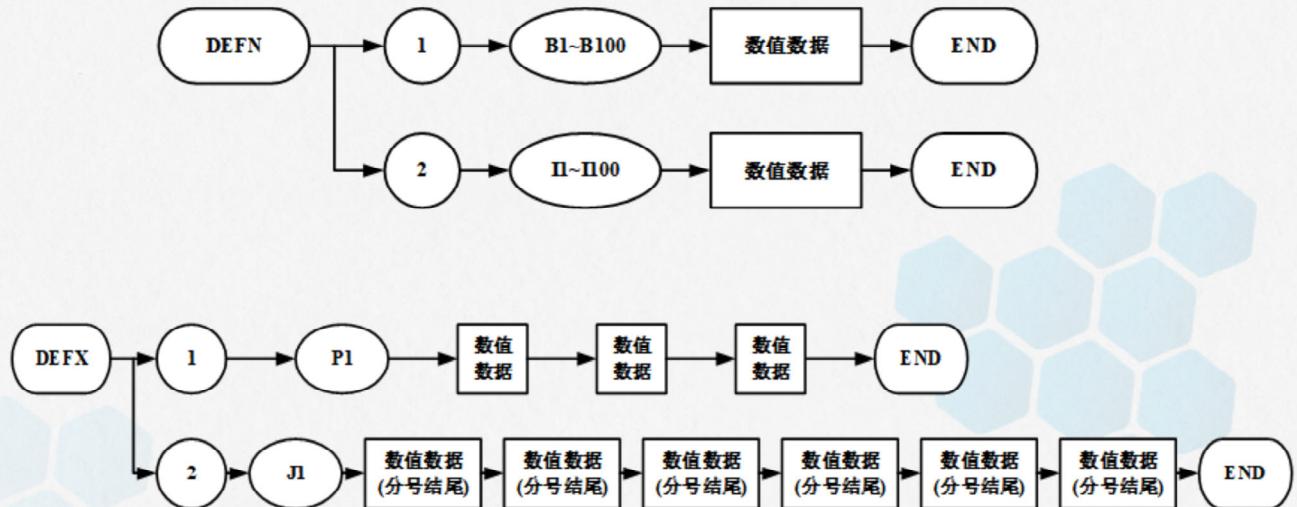
表7.10 机器人语言指令集变量说明表

名称	说明	单位	备注
B1~B100	字节型变量	字节型	运用 DEFN 命令定义
I1~I100	整数型变量	整型	运用 DEFN 命令定义
P	位置点坐标变量	毫米 mm	运用 DEXF 命令定义
J	关节位置变量	度。°	运用 DEXF 命令定义

由于前面介绍的机器人指令内容涉及到了变量，所以这里也简单设计了两种必要的变量指令，四种变量内容如表7.10所述。

## 机器人语言指令集设计

### (3) 变量定义指令 (DEFN、DEFX)



这里仅实现字节型变量与整形变量的定义命令(DEFN)以及空间点与关节位置变量的定义命令(DEFX)。

DEFN命令结构图如图7.12所示。

字节型变量与整形变量定义命令名称: **DEFN**; 功能: 设置B/I变量, 并对B/I变量赋值; 用法示意参见图7.12所示两种语法。

举例如下:

**DEFN I10 5**

**DEFN B0 c**

说明: 将I10值设为5, 将B0值设为'c'。

末端位置与关节变量定义命令名称: **DEFX**; 功能: 创建末端空间点P变量, 并对P变量赋值; 创建关节J变量, 并对J变量赋值, 用法示意图如图7.13所示。举例如下:

**DEFX P1 555 23 583**

说明: 创建P1点, 并为该点的坐标赋值, 坐标值单位一般为mm。

**DEFX J1 0; 0; 0; 45; 30; 90**

说明: 创建关节坐标变量J1, 并为每个关节角度进行赋值 (这里缺省为六个旋转关节)。

## 机器人语言词库设计

表 7.11 词库定义

```
typedef enum
{
    ENDFILE,ENDLINE,INVALID,
    INIT,BEGIN,END,MOVEP,MOVEL,MOVEJ,B,I,P,J,X,Y,Z,DEFN,DEFX,
    ID,NUM
}Tokentype;
```

表 7.12 词库结构体

```
static struct
{
    char *str;
    Tokentype tok;
} Words[WORDNUM] =
{{"BEGIN",BEGIN}, {"INIT",INIT}, {"END",END},
 {"MOVEP",MOVEP}, {"MOVEL",MOVEL}, {"MOVEJ",MOVEJ},
 {"DEFN",DEFN}, {"DEFX",DEFX}, {"B",B}, {"I",I},
 {"X",X}, {"Y",Y}, {"Z",Z}, {"P",P}, {"J",J},
 {"NUM",NUM}}
```

设计了指令集语法结构及用途之后，首先需要为解析器设计一个词库。前面设计的机器人语言指令集的指令单元为英文单词，按照英文语法，一个完整的机器人语句涵盖多个单词，这些单词被空格划分。因此首先定义一个必要的词库，该词库涵盖之前所有机器人命令关键词。该词库定义成一个枚举型数据结构，C++示例代码如表7.11。

其中，**ENDFILE**表示指令文件结束，**ENDLINE**表示一行命令结束，**INVALID**表示非法未定义指令，**ID**代表正在识别指令，**NUM**表示获取的分词为数字，其余**token**类型为前面定义的指令集中的关键词。

为了在词法分析中方便进行分词匹配，定义一个结构体，将**token**类型一一对应到代表文本指令的字符串。

其中，**WORDNUM**为所有分词库的总数，这里经统计，设为16。

## 词法分析

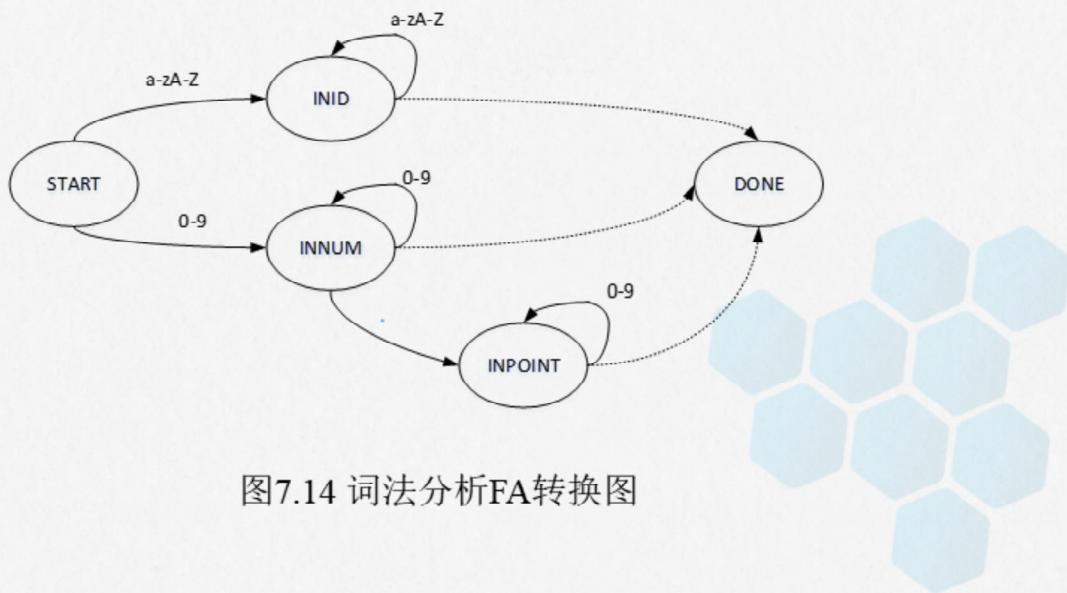


图7.14 词法分析FA转换图

读入一行程序代码后，需要进行词法分析。词法分析将读入的一行字符型源程序，将其组织成有意义的词素(lexeme)，并对于每个词素，产生词法单元(token)作为输出，其形式如：`<token-type, attribute-value>`，其中`token-type`表示该词法单元的类型名称，`attribute-value`是可选项，表示该词法单元的附加属性，词法单元下一步将被传送给语法分析器。当然，词法分析一般还将过滤掉代码中的空白和可忽略的字符，例如换行、注释等。

词法分析一般采用有限状态自动机(Finite Automata, FA)来实现。

在这个词法分析FA中，简单设计了5种转换状态，分别为：

`START`: 初始状态；

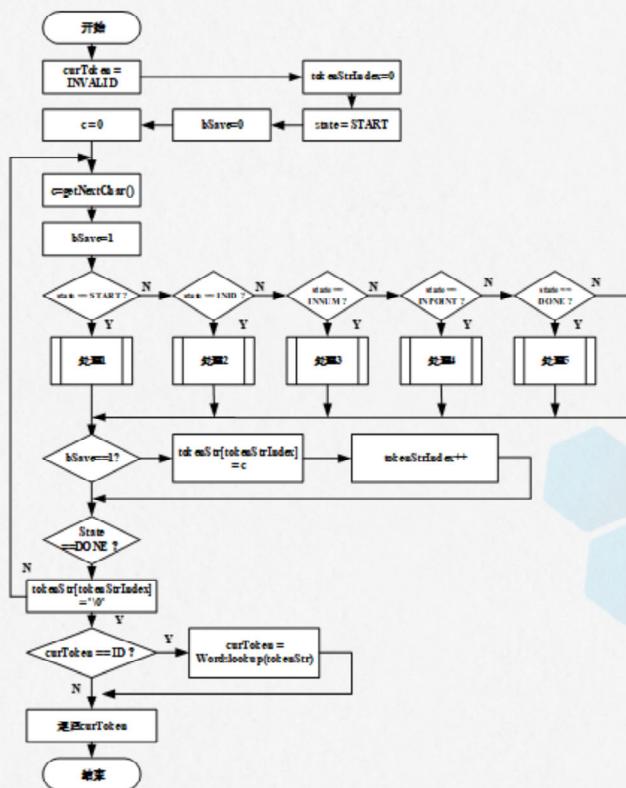
`DONE`: 终止状态；

`INID`: 当前读入字符为字母，正在识别命令字状态；

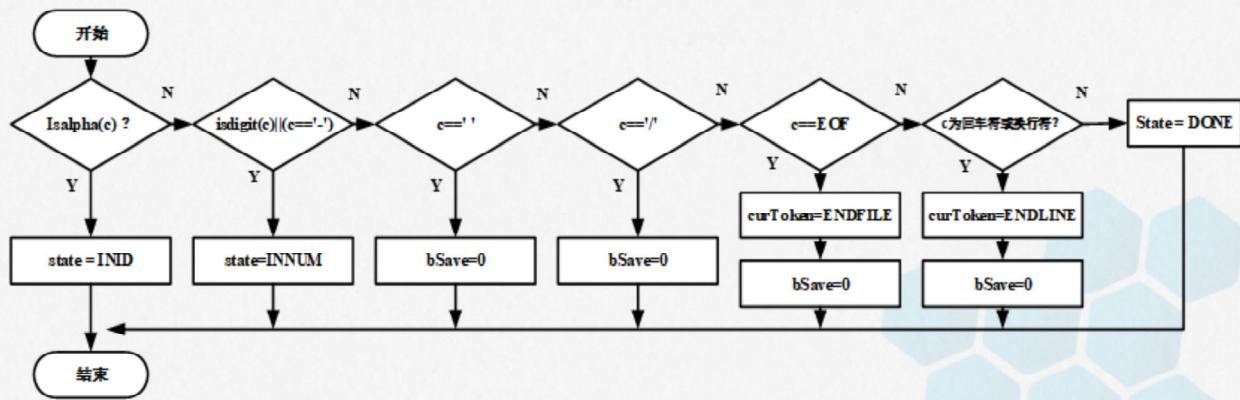
`INNUM`: 当前读入字符为数字；

`INPOINT`: 当前读入了小数点。

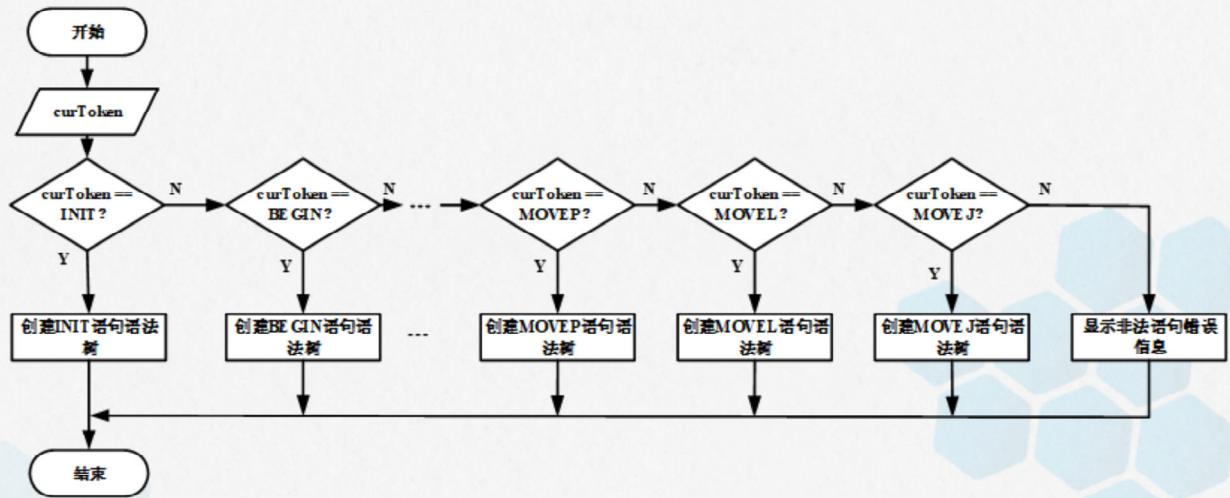
词法分析FA的转换图(Transition Graph)如图7.14所示。



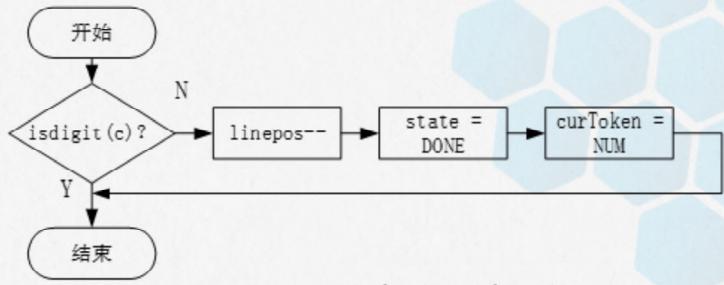
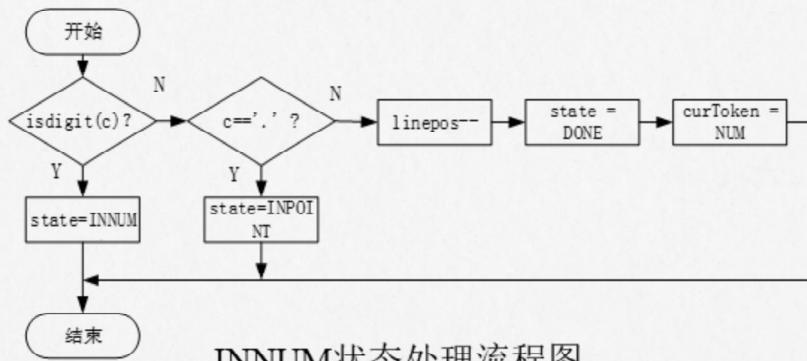
将转换图细化，可以设计出该词法分析器的代码，代码设计可以参考图7.15所示的流程图。



上述流程图中，state为START的处理流程如图7.16所示。该流程根据后续是否是字母或数字将状态机分别转入INID或INNUM状态，另外还根据布尔标记bSave设置，将空格、注释以及回车换行符等去除，将命令行有效内容分词之后按顺序存入tokenStr数组。



如图7.17所示，state为INID的流程处理较为简单，只需判别当前输入字符是否还是字母，如还是字母，则表明还需继续循环；如不是字母，说明命令关键词已读完，需要将命令行字符位置标识linepos回退一个字符位置，将状态设为终止状态(state=DONE)，并将需要返回输出的curToken设为ID。curToken设为ID表明该词素为一个命令符，在词法分析主流程中，设计了一个Wordslookup函数，根据存入tokenStr数组中的字符串，在前面定义的词库中查找对应的token类型。



同理，对于state为INNUM状态，说明当前读入字符为数字，需要将该数据位数全部读完（包括小数点后面数据，在本实例中小数点由INPOINT状态处理），INNUM状态处理流程如图7.18所示，INPOINT状态处理流程如图7.19所示。

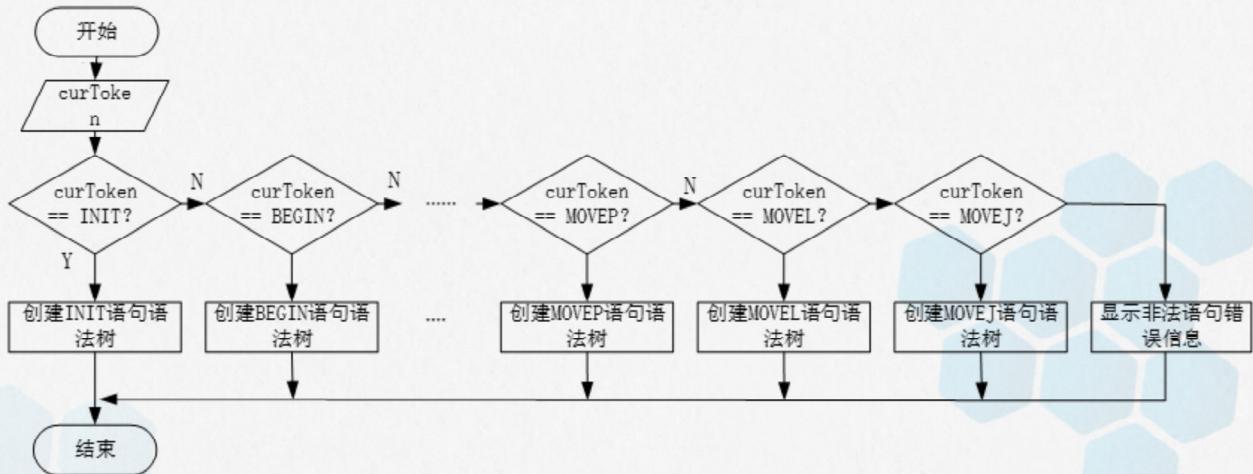
## 词法分析函数lex2token()

表 7.13 词法分析函数 lex2token()示例代码

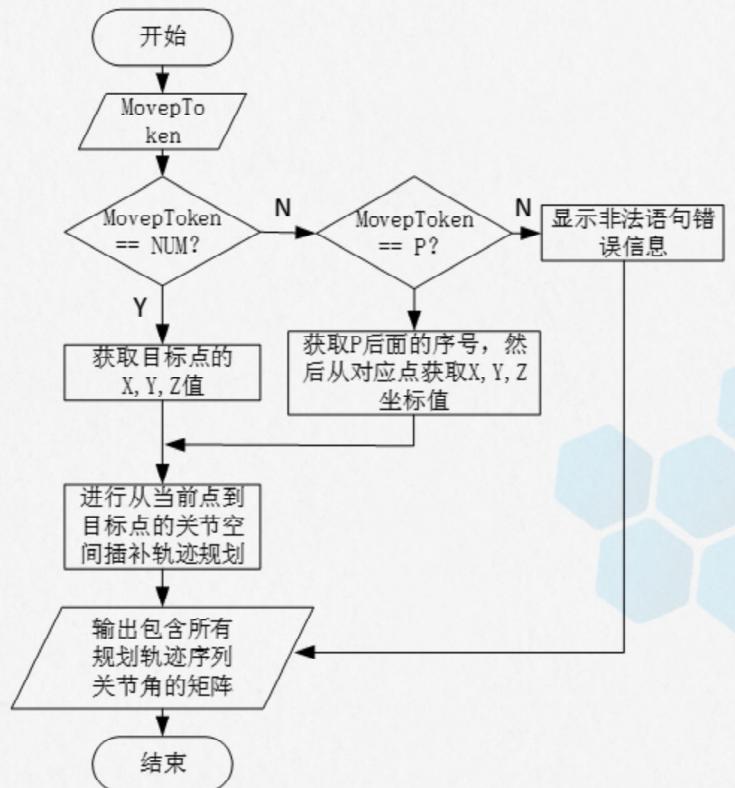
<pre> Tokentype lex2token(void) {     Tokentype curToken = INVALID;     int tokenStrIndex=0;     StateType state = START;     int save = 0;     char c = 0;     while(state!=DONE)     {         c=getNextChar();         save=1;         switch(state)         {             case START:                 if(isalpha())                     break;             case INID:                 if((isalpha())  (isDigit()))                     break;                 state=INNUM;             case INNUM:                 if((isalpha())  (isDigit())  (c=='.'))  c=='-'  c=='+'  c=='E'  c=='e')                     break;                 state=INPOINT;             case INPOINT:                 if((isalpha())  (isDigit()))                     break;                 state=DONE;                 curToken=NUM;                 break;             case INFILE:                 if(c=='\n'    c=='\r')                     break;                 save=0;                 state=ENDFILE;                 break;             case ENDFILE:                 if((c=='\n')  c=='\r')                     break;                 state=DONE;                 curToken=EOF;                 break;             case ENDLINE:                 if(c=='\n'    c=='\r')                     break;                 state=DONE;                 curToken=ENDLINE;                 break;             case ID:                 if((isalpha())  (isDigit()))                     break;                 state=INID;                 curToken=ID;                 break;             case EOF:                 if(save)                     break;                 state=DONE;                 curToken=EOF;                 break;             case DONE:                 break;         }     }     if((save) &amp;&amp; (tokenStrIndex &lt;= 20))         tokenStr[tokenStrIndex] = c;     else         tokenStr[0] = '\0';     if(curToken == ID)         curToken = WordIdentifier(tokenStr);     else         return curToken; }     </pre>	<pre> case INNUM:     if((isalpha())  (isDigit()))         state=INNUM;     else if(c=='.'  c=='-'  c=='+'  c=='E'  c=='e')         state=INPOINT;     else         saveNextChar();         save = 1;         state = DONE;         curToken = NUM;     break; case INPOINT:     if((isalpha())  (isDigit()))         saveNextChar();         save = 1;         state = DONE;         curToken = INUM;     break; case DONE:     break; if((save) &amp;&amp; (tokenStrIndex &lt;= 20))     tokenStr[tokenStrIndex] = c; else     tokenStr[0] = '\0'; if(curToken == ID)     curToken = WordIdentifier(tokenStr); else     return curToken; }     </pre>
--	---

这里采用C++编写一个简易的词法分析函数lex2token(), 其示例代码见表7.13。

## 语法分析



一般来说，语法分析 (Syntax Analysis) 是构建编译器的第二个步骤，语法分析器对上一步词法分析输出的词法单元(Token)进行判别，创建各类语句对应的语法树形结构 (语法树Syntax Tree)。本例中的语法分析处理流程图如图7.20所示。



以MOVEP语句为例，参照图7.9所示的语法结构图，MOVEP语句语法分析流程图如图7.21所示。

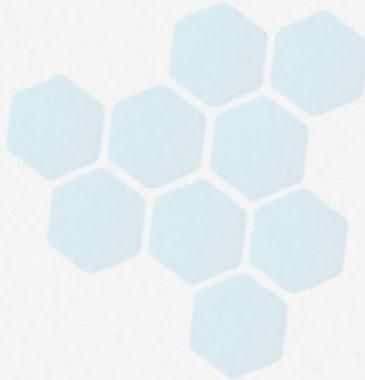
MOVEP语法分析流程对MOVEP后面的词素进行分析，判别是NUM还是P，然后进行相应的参数读入处理，并按照在第五章运动规划中介绍的关节空间插补轨迹规划算法，计算出所有轨迹点的关节角序列。MOVEL对应笛卡尔空间插补轨迹规划算法，而MOVEJ则直接给出机械臂所有轴的关节角。其余指令语法树的实现也较为简单，可参照实现。

一般编程语言其编译器的开发，在进行语法分析后，还会进行语义分析(Semantic Analyzer)。语义分析使用语法树和符号表中的信息来检查源程序和语言定义的语义是否一致，同时收集类型信息，并将这些信息存放在语法树或符号表中，供后续的中间代码生成使用。由于本章只介绍机器人语言解析器的基本原理，所以该过程在本章中被忽略，有兴趣的读者可以参阅编译器原理的相关课程章节内容。

## 编译/解析与执行

两种方式：

- 编译-目标代码-执行
- 解析-执行



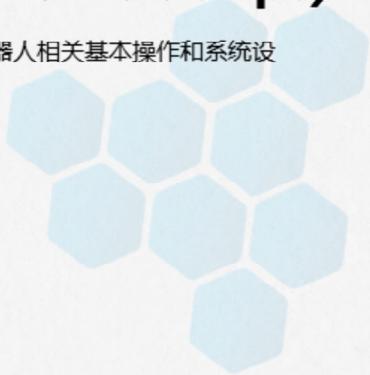
一般的计算机编译器，在对源程序完成语义分析之后，需要将源程序生成可执行的目标代码。有些编译器还会生成中间代码，将中间代码优化后，再翻译成目标机器代码。

机器人控制器一般采取解析执行的方式运行，控制程序读入经过语法分析后的合法语句，按照构建的语法树，生成可供控制的关节角序列，控制机器人完成相应运动。如果机器人采用总线控制，各关节为独立关节，则主控制器按照通信协议将相应的关节命令（编译器生成的关节角序列矩阵）发送给相应关节执行。

06

## 课程实验 (AUBOScript)

在实验室培训平台上掌握机器人相关基本操作和系统设计。



# 本门课程实验

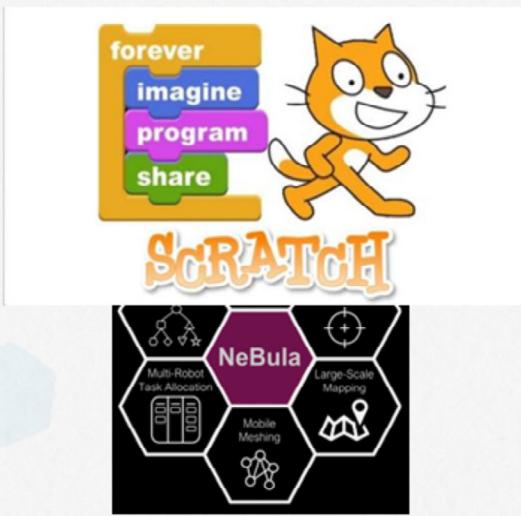
AUBOScript语言

VID\_20190918\_180002~1.mp4



## 扩展讨论

图形化编程语言



ChatGPT



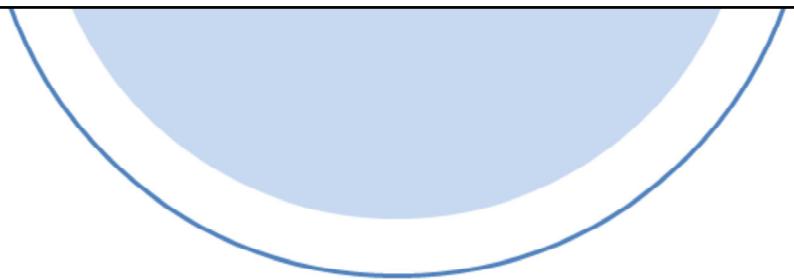
通过本章的学习，希望大家能对机器人编程语言一级编程语言解析器的开发有专业的了解，有兴趣的读者可以结合机器人编程语言解析器的特点，结合现有的高级脚本语言的解析器去设计机器人的高级编程脚本语言，比如在python或者lua等脚本语言上增加自己的机器人专用指令的支持。

面向非专业用户的可视化机器人编程将大大降低用户的学习事件和部署成本。如Rethink Robotics公司开发的可视化机器人软件平台Intera，将所有的任务都以图形的方式排列成行为树，从而更加容易想象、理解和调整机器人做的每一步工作。通过基于行为树开发的中间件，Intera能够协调其他装配线组件、传送带、分类零件、钻头等设备，从而快速的让机器人完成复杂费任务。通过不到一天的培训，一名工程师就能在短短一个多小时内部署及训练机器人；Paxton C等人推出了一种面向非专业用户的可视化机器人操作编程系统Costar。如同乐高机器人编程一样，在Costar中通过拖动模块控件和连接模块控件的方式即可完成机器人的任务编程。该系统中各类模块控件集成了不同的传感器、执行器或规划器及对应的智能算法，通过简单的学习，普通用户就可以编写程序让机器人完成自动化操作任务；Berenz V等人推出的Playful平台也是一种可视化的机器人编程系统，也是通过行为树模型来描述不同抽象层行为，再由底层引擎来实现机器人的操作。相比于Costar，Playful系统还能够自定义行为树模块的结构修改规则，使得机器人能够根据环境变化自主更新任务的规划方案，从而进一步提高了机器人编程的智能化程度。

随着智能机器人逐步进入到人类的生活中，机器人的发展趋势是通过语言识别、语义理解、语音合成等技术与人之间进行无障碍的交流，无需编程即可完成人类指定的任务。由OpenAI发布的对话式大型语言模型

**ChatGPT**在发布之处就在各大中外媒体平台掀起了一阵狂热之风。ChatGPT在机器人编程中就可以应用于多个方面。首先，ChatGPT作为一个强大的自然语言处理模型，可以被用于识别用户的意图、解析用户的语义、提取关键信息等。其次，ChatGPT可以被用于对话意图的识别、对话情境的理解、对话流程的设计等。在机器人编程中，对话管理可以帮助机器人管理用户和机器人之间的对话，使得机器人能够更好地理解用户的需求并作出恰当的回应。另外，在工业制造中可以使用ChatGPT实现机器人编程。在传统的工业制造中，机器人编程通常需要专业技能和经验。但是，现代的机器人编程工具可以使非专业人员也能够进行机器人编程。这些工具通常包括可视化编程界面、自然语言处理接口、云端存储和分享等功能，使得机器人编程变得更加容易和普及化。

### Chat Generative Pre-trained Transformer



# THANK YOU

标题、正文等都可以通过点击进行修改，可以对字体、字号、颜色、行距进行修改。

添加文字