Javascript 机器人编程指南

目录：

一、Javascript 简介

Javascript 诞生于 1995 年，当初主要目的是替代服务器端脚本在本地处理一些输入验证工作。目前 Javascript 已经不限于仅进行输入验证工作，通过 Javascript，可以与浏览器窗口及其内容、本地设备进行充分交互。它已经成为一种功能全面的编程语言，能处理功能复杂的计算和交互。应该说，它既是一门非常简单的语言，但它又是一门非常复杂的语言，说它简单是学会使用它只需要片刻功夫；说它复杂是完全掌握它则需要大量的时间。

Javascript1.0 最初由 Netscape 发布，随后微软在 IE 中含了 Jscript,同期还有 ScriptEase 推出的 Cenvi。3 个不同版本的存在使大家觉得制定统一标准的必要性，1997 年，以 Javascript1.1 为蓝本的建议被提交给了欧洲计算机制造商协会（ECMA），该协会指定 39 号技术委员会（T39）完成标准化工作，最终发布了 ECMA-262 一种名为 ECMAScript 的脚本语言标准。1998 年，国际标准化组织和国际电工委员会（ISO/IEC）也采用了该标准，命名为 ISO/IEC-16262.

自此以后，各种浏览器开始致力于以 ECMAScript 为基础实现交互，目前较为常用的开源 Javascript 引擎有两种：spidermonkey 引擎用于 Mozila 浏览器，V8 引擎用于 chrome 浏览器。

但是 ECMA-262 和浏览器并没有关系，它定义的只是一种脚本语言的基础，宿主环境可以通过提供 Javascript 扩展支持其它功能， 如对于浏览器通过扩展提供了对 DOM 的支持；而对于机器人控制领域，则提供对 Linuxcnc 和 Modbus 的扩展从而实现机器人或 CNC 数控功能。

二、Javascript 编程基础

1. Javascript 语法

- 分号结尾：Javascript 借鉴了 C 和类 C 的语法，但它比 C 简单很多，每一语句以分号结尾；
- 区分大小写：变量、函数名、操作符都区分大小写；
- 标识符：对于变量、函数、属性的名称或者参数名称，其命名按以下规则：
  - 第一个字符必须是字母、下划线（_）、或者美元符号($)；
  - 其它字符或以是字母、数字、下划线或美元符号；
  - 按照惯例，标识符第一个字母一般小写，剩余的每个有意义的第一个字母大写，如：

    myCar;

    firstSecond

    doSomethingImportant;
  -
- 注释：采用与 C 相同的注释方式
  - 单行注释采用”//”；
  - 多行注释采用如下形式：

    /*注释第一行；

    *注释第二行；

    *注释第三行；

    注释第四行；*/

- 使用花括弧：通过使用左右花括弧“{}”，可以包含多个语句，形成一个语句代码块；这些代码块可以用作条件或循环的控制体，或者仅用于更加清楚的表示程序结构。

2. 关键字和保留字
   - ECMA-262 定义了一组有特定用途的关键字，它们用于表述控制语句的开始和结束，或者用于执行某些操作，这些关键字属于系统保留，用户定义的标识符不能和它们相同。以下是 Javascript 的所有关键字：

break	do	instanceof	typeof
case	else	new	var
catch	finally	return	void
continue	for	switch	while
function	this	with	
default	if	throw	
delete	in	try	

   - ECMA-262 还描述了另外一组不能用作标识符的保留字，它们可能在将来被用作关键字，ECMA-262 第 3 版定义的全部保留字如下：

abstract	enum	int	short
boolean	export	interface	static
byte	extends	long	super
char	final	native	synchronized
cass	float	package	throws
const	goto	private	transient
debugger	implements	protected	volatile
double	import	public	

3. 变量
   - 变量的定义采用一种很松散简单的方式如下：
     - var message;
       上面定义了一个 message 变量，它可以保存任何类型的值。像这种没有初始化的变量，它保存了一个 undefined 值；
     - var message = “abc”;
       上面定义了一个变量 message 变量，并给它赋了一个字符串类型的初始值；
   - 变量的作用域，在函数内部使用 var 定义的变量，作用域仅限于函数内部使用，当函数退出来时，这些变量将销毁；对于全局变量作用于整代码文件，且不会自动销毁。
   - 同一行可以定义多个变量：
     Var message=”abc”,founded = false,age = 30;

4. 数据类型
   - undefined 类型：
     当变量定义后，如果没有赋初始值，则其初始值为 undefined,如下代码，则将

执行代码块 1。

```
var message;
if(message == undefined){
    代码块 1
}
代码 2；
```

- null 类型：
  如果定义的变量将用于保存对象,推荐在使用前给他赋初值为 null 这样只要检测该变量是否为 null 就可以知道该对象是否存在；

- Boolean 类型：
  该类型表示逻辑真和假，对应 true 与 false,另外请注意，true != 1; false != 0;

- String 类型:
  该类型用于表示零个或多个 UNICODE-16 字符表示的字符串，字符串可以由双引号(“”)或单引号(‘’)表示，如下两种表示方法均有效：

```
var firstName = “Eric”;
var firstName = ‘Eric’;
```

  ECMAScript 的字符串是不可变的，一旦创建其值就不会再改变，要改变其值，首先必须销毁原来的，重新定义变量的新值。如：

```
var lang = “java”;
lang = lang + “script”;//重新定义新值，并销毁原来的；
```

  字符串可以从其它类型变量转换得来，一般可以调用 toString()得到这些变量的字符串值，对于浮点型，可以调用 toFixed(n)得到字符串值；如下：

```
var age = 11;
var agestring = age.toString();//agestring == “11”;
var found = true;
var foundstring = found.toString();//foundstring == “true”;
var mask = 0x10;
var maskstring1 = mask.toString(10);//maskstring1 == “16”
var maskstring2 = mask.toString(2);//maskstring2 == “00010000”
var maskstring3 = mask.toString(16);//maskstring3 == “10”
var dvalue = 3.1415926;
var dvaluestring = dvalue.toFixed(3);//dvaluestring == 3.141
var dvaluestring1 = dvalue.toString();//dvaluestring == 3.1415926;
```

- Number 类型:
  ECMAScript 采用 IEEE754 格式来表示整型和浮点型，并支持 2 进制、10 进制、16 进制等多种表示方法；对于整型和浮点型，其内存表示方法不一样，浮点型占用内存是整型的两倍。如下：

```
var    intNum = 1990;//十进制整数；
var    intNum1 = 0x10FA;//16 进制整数；
var    intNum2 = 0b1010;//2 进制整数；
```

```
var    floatNum = 3.14;//浮点型赋值；
var    floatNum1 = 3.14e6;//floatNum1 == 3140000
var    floatNum2 = 3.14e-6;//floatnum2 == 0.00000314;
```

浮点数的最高精度为 17 位有效数值；最大数值为 1.7976931348623157e+308; 最小数值为 5e-324;

NaN 是一个特殊的值，表示非数值，即本来该返回数值时，由于非法操作或越界无法返回数值值，这时候通过返回 NaN 告诉出错。可以有效防止程序崩溃；NaN 与任何值运算都返回 NaN；同时 NaN 与任何值都不相等（包括 NaN 本身）。

有 3 个函数支持从其它非数值转换为数值：parseInt()、Number()、parseFloat();

```
var num1 = Number(“Hello!”);//num1 == NaN;
var num2 = Number(“ ”);//num2 == 0;
var num3 = Number(“123”);//num3 == 123;
var num4 = Number(true);//num4 == 1;
var num5 = parseInt(“1234blue”);//num5 = 1234;
var num6 = parseInt(“0xA”);//num6 == 10;
var num7 = parseInt(“22.5”);//num7 == 22;
var num8 = parseInt(“0xABCD”,16);//num8 == 0xABCD;16 进制数；
var num9 = parseInt(“123”,10);//十进制；
var floatNum1 = parseFloat(“1234blue”);//floatNum1 == 1234,为整型；
var floatNum2 = parseFloat(“3.14159”);//floatNum2 == 3.14159;
var floatNum3 = parseFloat(“3.125e7”);//floatNum3 = 31250000;
```

- Object 类型:

  ECMAScript 的对象类型对应的是一种功能和数据的集合，类似于 C++中的对象，他有方法与属性，可以通过 New 创建一个对象，也可以通过对对象变量赋 null 值使用销毁；

```
var obj = new Array();//创建了一个数组对象；
var objFile = new File(“/media/sdcard/sample.txt”);//创建了一个文件对象;
obj = null;//销毁；
objFile = null;//销毁；
```

5. 函数

Javascript 中的函数是用于根据一定的输入参数，按照指定的算法得到输出结果的功能块。函数采用 function 定义，后面跟上函数名、参数以及函数体，如下：

```
function set_g5x_pos(x0,y0,z0)
{
    var x,y,z;
    x    = 1.05;
    y    = 310.56;
    z = -65.7;
    …
    Linuxcnc.setG5x(0,1,x0);
```

```
    Linuxcnc.setG5x(0,2,y0);
    Linuxcnc.setG5x(0,3,z0);
    …
    return x+y+z;
}

var retSetG5x = set_g5x_pos(100.0,122.4,-43.0);
```

其中函数体用”{}”包括起来；

函数参数用”()”区分开；

函数体中可以定义局部变量；

函数有返回值为浮点型；

函数调用时，传入的参数不一定要和函数定义一样多，可以少于函数定义时的参数个数，也可以多余其定义的参数个数；

6. Javascript 运算

1) 基本运算，包含加、减、乘、除、赋值操作、求余数；

```
var n = 5 + 6 * 3 / 2;//结果为 14，赋值给变量 n；
var fn = 5.6 + 6.3 * 2.5 / 1.6;//浮点运算,结果赋值给变量 fn；
var rn = 7 % 3;//求余数，结果为 1，赋值给变量 rn;
rn   +=  5;//结果为 6，赋值给 rn;
rn   -= 3;//结果为 3，赋值给 rn;
rn++;//加 1 操作，结果为 4；
rn--;//减 1 操作，结果为 3；
```

2) 逻辑运算，包含逻辑与、或、非、异或；

```
var b1 = true;
var b2 = false;
var b3 = b1 & b2;//逻辑与；
var b4 = b1 | b2;//逻辑或；
var b5 = b1 ^ b2;//异或；
var b6 = !b1;//非；
```

3) 条件运算，包含大于、小于、等于、大于等于、小于等于、与、或、非；

```
Var na = 5,nb =3;
Var b1 = true,b2=false;
If(na > nb){//关系运算，比较大小；
}
If(na >＝ nb){//关系运算，比较大小；
}
If(na < nb){//关系运算，比较大小；
}
If(na <= nb){//关系运算，比较大小；
}
If( b1 && b2){//b1、b2 同时为真时，条件判断为真；
```

```
}
If(b1 || b2){//有一个为真，条件成立；
}
If(!b1){//b1 为 false，条件成立；
}
```

4) 位操作，包含按位与、按位或、控位异或、有符号左移位、有符号右移位、取反、无符号左移位、无符号右移位；

```
var bm1 = 0x7E;//bm1 初值为二进制 0b01111110;
var bm2 = 0x01;//bm2 初值为二进制 0b00000001;
var bm3 = -64;
var bm4 = bm1 & bm2;//bm3 按位与，结果为 0b00000000;
var bm5= bm1 | bm2;//按位或，结果为 0b01111111;
var bm6 = bm1 ^ bm2;//按位异或，结果为 0b01111111;
var bm7 = bm2 << 5;//左移位，结果为 0b00100000;
var bm8 = bm1 >> 5;//无符号右移位，结果为 0b00000011;
var bm9 = bm3 >>> 5;//有符号右移位，结果为 0x7FFFFFFE;
```

5) 数学运算库，包含常用数学运算库如三角函数(sin,cos,tan,acos,asin,atan,atan2)、取整(ceil,floor)、开方（sqrt）、平方(power)、对数(log)、随机数（random）、求绝对值(abs)、自然常数运算(exp);

7. Javascript 条件控制

1) if 语句：

```
if(condition) {
    statement1;
}else{
    Statement2;
}
```

2) switch 语句：

```
switch(expression){//表达式可以是任何数据类型，如整型、字符串等。
    case value1:
        statement1;
        break;
    case value2:
        statement2;
        break;
    default:
        statement3;
        break;
}
```

8. Javascript 循环语句

1) for 循环

```
for( i = 0; i < 100; i++){
    statement;
}
```

2) do-while 循环

```
do{
    statement;
}while(condition);
```

3) while 循环

```
while(condition){
    statement;
}
```

4) for-in 循环

```
for( property in object) {//枚举对象的属性；
    statement;
}
```

5) label 标签,用于循环体中跳出用；

```
start: for( var i=0; i< 100; i++){
var c;
statement;
if(c == 0)
    break    start;
else
    continue start;
}
```

6) break 语句；用在循环语句、switch 语句中；

```
for(var i= 0; I < 1000; i++){
    if( i%10)
        break;
    statement;
}
```

7) continue 语句,应用于循环语句中；

```
for(var I = 0; I < 1000; i++){
    if(i%10)
      continue;
    statement;
}
```

9. Javascript 正则表达式,用于查找匹配，是由一个字符序列形成的搜索模式；
   1) 语法为：/正则表达式主体/修饰符(可选)
      如：/\w.tab/i　\w 代表任意字母数字组合，常用于文件名；\w.tab 代表所有扩展名为.tab 的文件；i 代表忽略大小写；

   2) 其它正则表达式的模式如下：
      [abc]查找 abc 中任意字符；
      [0-9]查找数字；
      (x|y)查找任何以|分隔的选项；
      \d 查找数字；
      \s 查找空白字符；
      \w 字母数字文件名组合；
      \b 查找单词边界；
      \uxxxx 查找 16 进制数；
      n+ 至少一个；
      n* 包含零至多个；
      n? 零个或 1 个；

10. Javascript 类型转换
   1) 任何其它类型都可以转换为 Boolean 类型；在条件表达式中，他们会自动转换，也可以命名用 Boolean()函数转换；

   2) 数值转换，有三个函数可以把非数值类型转换为数值类型
      Number()
      parseFloat()
      parseInt();

   3) 字符串转换:
      几乎所有类型的变量都有 toString()功能，可以使用该函数把指定变量转成字符串；
      浮点值还能通过 toFixed()转换成字符串；

11. Javascript 字符串处理
   1) substring(start,end)返回字符串的子串；
   2) toLowerCase()转变为小写字符串；
   3) toUpperCase()转变为大写字符串；
   4) charAt(pos)获取字符串中的字符,返回包含单个字符的字符串；
   5) charCodeAt(pos)获取字符串中字符，返回该字符的数值；
   6) indexOf(searchstr,fromindex)从指定位置查找字符串，返回查找到的字符串在母字符串中的索引；
   7) lastIndexOf(searchstr,forindex);从母字符串结束开始匹配，返回查找到的字符串在母串中的索引；
   8) match(seachvalue)匹配母串与 seachvalue 字符串,其中 seachvalue 可以是正则表达式；

9) search(searchvalue)按正则表达式查找字符串；返回找到的子串；
10) replace(regexp/substr,replacement)字符串中，用一些字符替换另一些字符；
11) split(sep)把字符串分隔成字符串数组；
12) concat(string1,string2,…)连接两个或多个字符串；
13) slice(start,end)提取母串中的字串；
14) release();释放母串所在内存;

12. Javascript 时间日期处理

1、 定义

var mydate = new Date();//定义时间对象并赋初始值为当前时间；

2、 UTC 返回指定的时间距离 1970 年 1 月 1 日午夜的毫秒数；

```
var d = Date.UTC(2005,7,8)//2005 年 7 月 8 日午夜距 1970 年 1 月 1 日午夜的毫秒数；
```

3、 now 获取当前时间；
4、 parse 解析字符串并返回距 1970/1/1 00:00:00 的毫秒数；
5、 getDate//返回月份的某一天；
6、 getDay//返回星期几；
7、 getHours//返回本地时间一天中的小时数；
8、 getMinutes//返回本地时间一小时中的分钟数；
9、 getMonth//返回本地时间第几个月；
10、 getSeconds//返回本地时间一分钟中第几秒；
11、 getMilliseconds//返回本地时间一秒中第几毫秒；
12、 getTime 返回距 1970/01/01 00:00:00 的毫秒数；
13、 getTimezoneOffset 返回本地时间距格林威时间的分钟数；
14、 getYear 获取本地时间当前年份，对于 1900~1999 之间的年份返回 2 位数字，其它返回 4 位数字；
15、 getFullYear 返回 4 位数字年份；
16、 setDate//设置为某一天；
17、 setHours//设置为某一小时；
18、 setMinutes//设置为某一分钟；
19、 setMonth//设置为某一月；
20、 setSeconds//设置为某一秒；
21、 setMilliseconds//设置为某一毫秒；
22、 setTime//使用 1970/01/01 00:00:00 以来的毫秒数设置时间；
23、 setYear//设置年份；
24、 toGMTString//转换成本地时间字符串；
25、 toUTCString//转换成格林威时间字符串；

13. Javascript 文件操作

1) 定义：
   var fobj = new File(path);//定义了文件对象，文件名为 path;

2) open(strmode);//打开文件，strmode 指定文件打开方式，只读、只写、是否覆盖；

3) close()//关闭文件；
4) remove();//删除此文件；
5) copyTo(strdest);//拷贝此文件到目的文件；
6) seek(pos);//调整读写指针到指定位置；
7) read(want)//读取指定数量数据，返回字符串；
8) readln();//读取一行字符串；
9) readAll();//读取所有文件，返回字符串数组；
10) write(str1,str2…);//写入字符串到文件中，可传入多个字符串；
11) writeln(str1,str2…);//写入字符串到文件中，可传入多个字符串，所有字符串当做一行，结尾补回车符。
12) writeAll(strarray);//写入字符串数组,每个数组元素后面补回车符；
13) list(match/regexp);//列出目录中匹配条件的文件名，到文件数组中；其中匹配条件既可以是字符串，也可以是正则表达式；返回文件数组；
14) mkdir(strname);//如果当前文件是目录，合并当前目录，否则不合并，创建指定文件；
15) toString();//返回当前文件路径名字符串；
16) toURL();//返回当前文件路径 URL 路径字符串。
17) 文件属性
    01. length//获取文件长度；
    02. parent//获取文件目录对象；
    03. path//获取文件路径；
    04. name//获取文件名；
    05. isDirectory//判断该文件是否为目录；
    06. isFile//判断该文件是否为数据文件；
    07. exists//判断文件是否存在；
    08. canRead//判断文件是否可读；
    09. canWrite//判断文件是否可写；
    10. canAppend//判断文件是否可添加；
    11. canReplace//判断文件内容是否可替换；
    12. isOpen//文件是否打开；
    13. type//文件类型；
    14. mode//文件打开模式；
    15. creationTime//文件创建时间；
    16. lastModified//文件最后修改时间；
    17. size//文件大小；
    18. hasRandomAccess//文件是否随机可访问；
    19. hasAutoFlush//文件是否自动同步到磁盘；
    20. position//文件指针位置；

三、Linuxcnc 扩展

1、 Init(inifile);//初始化 Linuxcnc 系统，
Inifile:存放系统初如化参数的文件；

2、 unInit();//反初始化；
3、 setState(state);//设置机床状态；
state:
1:急停状态；
2：急停复位；
3：关机；
4：开机；

4、 setTeleop(enable);//设置是否进入交互操作模式；
enable:
1:进入交互模式，
0 退出交互模式；

5、 setMode(mode);//设置机床加工模式
mode:
1:手动模式；
2：自动加工模式；
3：MDI 模式；

6、 Abort();//退出，如果当前为自动，则退出自动模式，如为手动或 MDI 则中断当前命令；

7、 Stop();//停止，如果当前为自动，则停止加工；
8、 Restore();//断点继续，记住上次退出时加工的位置，并重新加工；
9、 Shutdown();//系统关机；
10、 jogCont(joint,speed);//手动连续移动指定轴，需在手动模式下才能操作；
joint:轴编号；
speed:轴移动速度，当速度为负时，反向移动；

11、 jogIncr(ja,speed,offset);//按指定偏移量移动指定轴，当轴移动到位后函数返回；
ja:轴编号；
speed:移动速度，为负表示沿负方向移动；
offset：移动的距离；

12、 jogStop(ja);//停止当前连续运行的轴,ja:为将停止的轴编号；
13、 spindleOn(type,speed);//启动主轴;
type:
0:正转；speed 为正转转速；

1：反转；speed 为反转转速；
2：转速增加；speed 为增加转速值；
3：转速减小；speed 为减小转速值；

14、 spindleOff();//关闭主轴；
15、 brakeEngage();//主轴抱闸；
16、 brakeRelease();//主轴松闸；
17、 setSpdRatio(override);//设置主轴倍率，0verride 为倍率值，一般为 0~1.2;
18、 setHome(joint,home,offset,home_final_vel,search_vel,latch_vel,seq,flags);
设置回机械零点参数，
joint:指定轴；
home:原点相对行程开关确定的零点的位置；
offset:零点回退距离，当找到行程开关点后，机械零点离此点的距离；
home_final_vel 找到行程开关后，定位到 HOME 原点的速度,如果为 0,采用该轴的默认速度；
search_vel 快速寻找速度
latch_vel:第二次精确定位速度;
flags:
几种 home 点定位方法：
1、存在硬件行程开关时，采用二次探测法；
此时，flags 设置为 0,home_final_vel,serch_vel,latch_vel 设置为大于零的值，offset 表示零点回退距离；
2、当前点设置为机床零点；
此时除 joint 外参数全为 0；
3、设置当前点机械坐标为指令值，一般用于断点保存恢复。
此时 home=0.0,offset=记忆的当前机械坐标值-实际当前机械坐标值，速度全为 0,flags=2.sequence=0.

19、 setG5x(index,axis,value);
设置['P1 G54', 'P2 G55', 'P3 G56', 'P4 G57', 'P5 G58',
'P6 G59', 'P7 G59.1', 'P8 G59.2', 'P9 G59.3',
_('T Tool Table')]
工件坐标偏移
index:1~9 对应 P1~P9;
axis:0~5 对应 x,y,z,u,v,w;
value:当前机械坐标（原点在机床零点）－新的工件坐标(原点在机床零点)
return:0 成功，小于 0 失败。
注意，此参数永久保存在配置文件中，掉电也不会消失。

20、 setG5xAngle(index,angle);
设置坐标系旋转
index:1~9 对应 P1~P9;
angle:旋转角度，0.0 ~360.0

return:0 成功，小于 0 失败。

注意，此参数和工件坐标系参数一起，永久保存在配置文件中，掉电也不会消失。

21、 setG5xMirror(index,mirror);

设置坐标系以 X/Y 轴为镜像

index:1~9 对应 P1~P9;

mirrorflag:镜像类型，位操作。

0x01:X 轴镜像控制；1 镜向，0 不镜像；

0x02:Y 轴镜像控制；1 镜向，0 不镜像；

return:0 成功，小于 0 失败。

注意，此参数和工件坐标系参数一起，永久保存在配置文件中，掉电也不会消失。

22、 goHome(joint);

回机械原点：

joint:指定轴，当 joint=-1 时，操作所有轴，joint >= 0 时，指定单轴回机械原点

返回：0 成功，负值失败

23、 clearHome(joint);

清除原点标识，便于下次重新设置机械原点

joint:指定轴，当 joint=-2 时，操作所有轴，joint >= 0 时，指定单轴

24、 goJobStart(axis[],seq[],safeh[],axisnum);

控制各轴回工件原点，

axis:需要控制的各轴编号所组成数组，数组元素值 0~5，对应 x,y,z,u,v,w;

seq:数组中各轴的回工件原点顺序，0 最先，1 次之。。。

safeh:各轴回工件原点时，应该离工件原点的安全高度，此安全高度应该使刀具更接近机械零点的方向；

如对于 Z 轴，有了此安全高度，回工件零后，刀具应该停在工件零点和机械零点之间，离工件 Z 轴方向安全高度的距离；

axisnum:总的轴数目；

return:0:成功；为负失败，此函数为阻塞操作函数，命令执行完后返回；

25、 goProbe(x,y,z,vel,maxvel,acc);

自动对刀可以采用此函数，应该先运动到安全点，然后再调用此函数慢速接近探头或对刀仪。

运动到探头所在位置，如果在运动过程中触发了探头信号，则停止运行，保存当前探头触发位置的坐标；

如果运运到指定位置，还没有触发探头信号，则有可能刀具磨损了，或其它错误，应该重新设置坐标；

x,y,z:探头所在机械坐标位置；

vel:平均速度；
maxvel:最大速度
acc:触发后停止的加速度；
返回：0 成功，负值失败

26、 clearProbe();
清除探头触发标志
返回：0 成功，负值失败

27、 initPlan();//初始化加工计划；
28、 Open(file);//打开加工文件，
1) 目前支持.nc,.ngc,.iso,.eng,.pip 文件，
2) 其中.pip 文件需要通过 postMDI 命令动态提供数据；

29、 Run(line);
运行已经打开的加工文件；
line:指定开始加工的行号；
返回：0 成功，负值失败

30、 Pause();
自动加工暂停
返回：0 成功，负值失败

31、 Resume();
自动加工恢复运行；
返回：0 成功，负值失败

32、 Step();
单步运行加工文件，每步运行一行程序
返回：0 成功，负值失败

33、 runRanks(x0,y0,xstep,ystep,rows,cols,line);
阵列加工：
x0:第一个阵列点的 x G54 座标；
y0:第一个阵列点的 y G54 座标；
xstep:x 方向阵列工件原点的间距，可以为负值；
ystep:y 方向阵列工件原点的间距，可以为负值；
rows:y 方向阵列的行数；
cols:x 方向阵列的列数；
line:从阵列的哪一行开始加工，单列阵列文件总行数为 n,当 line=6n+m 时，则代表从第 7 个阵列的第 m 行开始加工；
返回：大于等于 0:成功；其它不成功；

34、 runArray(x,y,z,nums,lines);

按原点数组加工：

xpos:原点数组 x-机床座标数组；

ypos:原点数组 y－机床座标数组；

zpos:原点数组 z 机床座标数组

nums:阵列点个数，代表 xpos,ypos 数组元素的个数；

line:从哪一行开始加工，单列数组元素文件总行数为 n,当 line=6n+m 时，则代表从第 7 个数组元素的第 m 行开始加工；

返回：大于等于 0:成功；其它不成功；

35、 preMilling(x0,y0,w,h,depth,zstep,fstep,millmode);

铣底平面，加工前预处理：

x0:x 开始 G54 座标；

y0:y 开始 G54 座标；

w:x 方向宽度，可以为负；

h:y 方向高度，可以为负;

depth:z 方向总加工深度；

zstep:z 方向单次进刀量,为正

fstep:x/y 方向进刀量，为正；

millmode:x/y 方向进刀方式，0:沿 y 轴进刀，1:沿 x 轴进刀；

返回：大于等于 0:成功；其它不成功；

36、 sendMDI(strcmd);

运行指定 MDI 指令，执行任务结束后函数返回

mdi:MDI 指令字符串；

返回：0 成功，负值失败

37、 postMDI(strcmd);

发送指定 PIP 文件的动态指令，发送成功后函数返回

strmdi:MDI 指令字符串；

返回：0 成功，负值失败

38、 setFeedRatio(override);

设置 G01 进给倍率：

override:倍率 0.0~配置文件指定最大值；

返回：0 成功，负值失败

39、 setRapidRatio(override);

设置 G00 快速移动倍率；

override:倍率 0.0~配置文件指定最大值；

返回：0 成功，负值失败

40、 loadToolTable(strfile);

装载刀具文件

file:刀具文件名

返回：0 成功，负值失败

41、 setToolOff(toolnum,zoff,diameter);

设置刀具偏移：

toolno:刀具在刀库中的编号；

zoffset:Z 方向刀具的偏移值；

diameter:刀尖的直径

返回：0 成功，负值失败

42、 setTooloffEx(toolno,zoff,xoff,diameter,frontang,backang,orient);

设置刀具详细参数；

toolno:刀具在刀库中的编号；

zoffset:Z 方向刀具的偏移值

xoffset:X 方向刀具的偏移值

diameter:刀尖的直径；

frontangle:刀具前倾角；

backangle:刀具后倾角

orientation:刀尖方向；

返回：0 成功，负值失败

43、 setToolTouch(lum,tooin,axis,value);

设置刀具偏移，

lum:保留；

axis:0~5 对应 x,y,z,u,v,w;

toolin:刀具编号；为 0 时，清除刀具偏移；

value:刀具偏移值,为正时，表示此刀比标准刀长；

return:0 成功，小于 0 失败。

44、 setbacklash(joint,backlash);

设置各轴的反向间隙补偿；

joint:指定轴

backlash:补偿值；

返回：0 成功，负值失败

45、 enJoint(joint,enable);

使能指定轴；

joint:0~8,对应 x,y,z,a,b,c,u,v,w;

benable:1 使能；0 禁止；

返回：0 成功，负值失败

46、 loadJointComp(joint,file,type);

装载丝杆补偿文件；

joint:丝杆对应轴；

file:丝杆补偿文件，文件中有各个行程点的丝杆正反转补偿参数，文

件内容如下：

名义位置　正转补偿　反转补偿

0.000000 0.000000 -0.001279

0.100000 0.098742　0.051632

0.200000 0.171529　0.194216

type:文件格式类型；

0:正转补偿和反转补偿值为绝对坐标值；

其它：补偿值为与名义位置的偏差值；

返回值：0 成功，负值失败

47、 axisLimits(override);//使能软限位；

48、 getParams(maintype,subtype);

获取参数数据；

maintype=0:常用参数，返回浮点数据；

subtype:

0：主轴转速(rpm F)

1: 当前进给合成速度(mm/s,F)

2：当前设定的合成速度(mm/s,F)

30：规划进给速度(mm/s,F)；

31：规划最大速度(mm/s,F)；

32：默认规划加速度(mm/s,F)；

33：最大规划加速度(mm/s,F)；

60：主轴允许的最大转速 (rpm,F)；

150~155：Joint0~5 编码器脉冲计数器寄存器（U32）

160~165: Joint0~5 编码器 Z 脉冲计数器寄存器(U32)

300~399：JS 实时参数，浮点值 F；

400~499:整型值；JS 实时参数(U31)

500~505:各实时 JS 中轴实际坐标, 浮点值 F；

maintype=1:获取机床状态；

返回值：

1：急停；

2：急停复位；

3：关机；

4：开机；

maintype=2:获取机床交互模式；返回 0 不在交互模式，1 在交互模式；

maintype=3:获取机床模式：

返回值：

1：手动；

2：自动；

3：MDI；

maintype=4：获取机床主轴倍率；返回：浮点型主轴倍率；

maintype=5：获取主轴转动方向，返回：1 正转，0 反转；

maintype=6：获取主轴转动步进值；返回：主轴转速步进值；

miantype=7：获取手刹状态，返回：0 没有抱闸，1 抱闸；

maintype=8：获取各轴回零状态；

subtype:各物理轴编号；

返回：0 没有回零，1 已回零；

maintype=9：对刀时检查探头是否触发；返回：0 没有触发，1 已触发；

maintype=10：获取探头位置；

subtype:

0：获取探头 X 轴的值；

1：获取探头 Y 轴的值；

2：获取探头 Z 轴的值；

返回：各轴浮点型坐标值；

maintype=11：获取探头传感器的值；返回：0 无信号，大于 0 有信号；

maintype=12：当前正在加工的行号；返回：整型行号；

maintype=13：获取当前代码状态，返回：字符串；

maintype=14：获取当前程序运行状态；返回：

1：空闲；

2：正在读文件；

3：加工暂停；

4：执行等待；

maintype=15：获取当前加工的文件名；返回文件名字符串；

maintype=16：获取当前 G01 进给倍率；返回浮点型倍率；

maintype=17：获取当前 G00 快进倍率；返回浮点型倍率；

maintype=18：获取当前各物理轴机械坐标值；

subtype:物理轴编号；

返回：浮点型坐标值；

maintype=19：获取当前指令机械坐标；

subtype:0~8 对应 x/y/z/a/b/c/u/v/w;

返回：浮点型坐标值；

maintype=20：获取当前实际机械坐标值；

subtype:0~8 对应 x/y/z/a/b/c/u/v/w;

返回：浮点型坐标值；

maintype=21：获取当前相对指令坐标值；

subtype:0~8 对应 x/y/z/a/b/c/u/v/w;

返回：浮点型坐标值;

maintype=22：获取当前相对实际坐标值；

subtype:0~8 对应 x/y/z/a/b/c/u/v/w;

返回：浮点型坐标值

maintype=23：获取当前各轴 G5X 偏移值；

subtype:按十进制数表示；

个位：轴编号，0~8 对应 x/y/z/a/b/c/u/v/w;

十位：G5X 索引，0 对应 G54，1 对应 G55，2 对应 G56，3 对应 G57，4 对应 G58，5 对应 G59，6 对应 G59.1，7 对应 G59.2，8 对应 G59.3；

返回：浮点型偏移值；

maintype=24：获取 G92 偏移值；

subtype: 轴编号，0~8 对应 x/y/z/a/b/c/u/v/w;

返回：浮点型偏移值；

maintype=25：获取 G5X 坐标类型；

返回：坐标系索引值；

0: G53

1: G54

2: G55

3: G56

4: G57

5: G58

6: G59

7: G59.1

8: G59.2

9: G59.3

maintype=26：获取单位类型；

subtype:

0:程序直线单位；

1：程序角度单位；

2：用户直线单位；

3：用户角度单位；

4：显示直线单位；

5：显示角度单位；

6：直线单位变换；

角度单位变换；

maintype=27：获取当前加工用的刀具编号；返回当前刀具编号；

maintype=28：获取当前刀具 Z 偏移；返回浮点型偏移值；

maintype=29：获取当前物理轴的状态；

subtype:物理轴编号；

返回：整型，按二进制表示，BIT0：1inpos;0running;

maintype=30：获取 IO 输入端口的状态；

subtype:IO 输入端口编号，400~423 通用输入端口，423~500 为扩展输入端口；600~699 为自定义逻辑 IO 输入；

返回：0：复位，1 置位；

maintype=31：获取当前模拟输入端口的值；

subtype:模拟输入端口编号；

10：PWM1

11：PWM2

100：ADV1

101：ADV2；

返回：模拟输入端口值；

maintype=32：获取当前等待类型；

返回：

1：未知；

2：接收到命令；

3：执行完命令；

maintype=33：获取加工时间；

未定义；

maintype=34：获取当前错误信息类型；

返回：

-1：无错

0：未知错误，一般为解析器错误。

1000,/*运动控制模块错误*/

3000,/*解析器错误*/

4000,/*NML 通讯错误*/

5000,/*未知其它错误*/

10000,/*急停或者没有开机*/

10001,/*因在手动模式不能运行*/

10002,/*因在自动模式且解析器空闲时不能运行*/

10003,/*因在自动模式且解析器读指令时不能运行*/

10004,/*因在自动模式且解析器暂停时不能运行*/

10005,/*因在自动模式且解析器等待时不能运行*/

10006,/*因在 MDI 模式不能运行*/

10007,/*切换模式时系统正忙*/

10008,/*关闭文件出错*/

10009,/*打开文件出错*/

10010,/*运行 MDI 命令时，却没有归零*/

10011,/*运行 MDI 命令时，却没有处于 MDI 模式*/

10012,/*运行程序时，没有归零*/

10013,/*定位超时*/

10014,/*定位失败*/

10015,/*未知等待错误*/

10016,/*软限位触发错误*/

10017,/*软限位触发错误*/

10018,/*因为 IO 的错误*/

Maintype=35：获取警告信息类型；

返回：

-1,/*无警告*/

0,/*未知警告*/

4000,/*其它警告*/

Maintype=36：获取当前液体喷雾冷却状态，0 关闭，1 打开；

Maintype=37：获取当前冷却液状态，0 关闭，1 打开；

Maintype=38：获取当前润滑油状态，0 关闭，1 打开；

Maintype=39：获取当前润滑油标尺水平；返回整型水平值；

Maintype=40：获取当前设备类型；

返回：identity=1,serial=2,parallel=3,custom=4

Maintype=41：获取当前配置文件；

返回：配置文件名字符串；

Maintype=50：获到当前文件加工百分比；返回已加工的整型百分比；
Maintype=51：获取当前 CNC 运行状态；
返回：
0：正常手动工作状态；
1：手动回调状态；
2：运行状态；

49、 setParams(type,value);
设置参数数据；
type:
0: 主轴转速(rpm F)
30: 进给速度
31: 规划时最大速度
32: 划时加速度，单轴加速度
33: 规划时最大加速度,空行加速度
34: 空行速度
60: 主轴允许的最大转速 (rpm,F)
61: 主轴达到指定转速的准备时间（s,F)
90: Z 轴抬刀安全高度(mm,F)
91: ENG 文件中插入抬刀指令的抬刀量(mm,F)
92: 暂停时 Z 轴抬刀量(mm,F)
97: Z 轴下刀速度(mm/s,F)
98: Z 轴离刀速度(mm/s,F)
88: 暂停时是否停止主轴(U32)
93: 暂停时的操作模式:0 不抬刀，不停主轴;1 抬刀到 Z 安全高度位置;2 指定抬刀量;(U32)
95: 使用默认主轴转速(U32)
96: 使用默认进给速度(U32)
113: 手轮脉冲计数器寄存器(U32)
99: 停止时不停止主轴选项,(U32)
94: 换刀时是否提示并暂停(U32,1 提示，0 不提示)
110: 上次关机时间(U32,time_t)
111: /*注册后持续时间(U32,秒)
120:X 轴手轮是否使能；value:0 不使能，1 使能；
121: Y 轴手轮是否使能；value:0 不使能，1 使能
122: Z 轴手轮是否使能；value:0 不使能，1 使能
123: A 轴手轮是否使能；value:0 不使能，1 使能
124: 保留
125: 保留
126: 保留
127: 保留
300~399:设置浮点参数，用于设置实时模块的参数；
400~499:设置整型参数，用于设置实时模块的参数；

50、 setDIO(ionum,bvalue);//设置 IO 端口状态；

ionum:

10~29：轴使能；

30~49：轴报警清除

50~69：轴报警有效电平；

70~89：轴默认方向；

99：主轴报警有效电平；

100~164：通用输入有效电平；

165~199：扩展输入有效电平；

200：输出有效电平；

300~323：通用输出控制；

324~399：扩展输出控制；

700~800: 逻辑 IO 输出，用户自定义通讯接口；

bvalue:0 置低，1 置高；

51、 setAIO(aionum,dvalue);//设置模拟端口数值；

aionum:

dvalue:模拟输出值；

52、 setMist(enable);//设置液体喷雾冷却功能，0 关闭，1 打开；

53、 setFlood(enable);//打开或关闭冷却液，0 关闭，1 打开；

54、 setLube(enbale);//打开或关闭润滑油，0 关闭，1 打开；

55、 setJointParam(type,joint,value);设置各物理轴的参数；

joint:物理轴编号；

value:参数值；

type:

0：最大物理位置值；

1：最小物理位置值；

2：最大速度；

3：最大加速度；

4：铁电记忆位置；

5：反向间隙补偿值；

6：电机驱动脉冲当量；

7：编码器逻辑脉冲记数；

8：编码器脉冲记数；

9：编码器 Z 脉冲记数；

56、 setAxisParam(type,axis,value);//设置各坐标轴的参数；

axis:坐标轴编号，0~8 对应 x/y/z/a/b/c/u/v/w;

value:参数值；

type:

0：坐标轴最大值；

1：坐标轴最小值；

2：最大速度；

3：最大加速度；

57、 getJointParam(type,joint);//返回浮点值；

joint:物理轴编号；

type:

0：最大物理位置值；

1：最小物理位置值；

2：最大速度；

3：最大加速度；

4：铁电记忆位置；

5：反向间隙补偿值；

6：电机驱动脉冲当量；

7：编码器逻辑脉冲记数；

8：编码器脉冲记数；

9：编码器 Z 脉冲记数

58、 getAxisParam(type,axis);//返回浮点值；

axis:坐标轴编号，0~8 对应 x/y/z/a/b/c/u/v/w;

type:

0：坐标轴最大值；

1：坐标轴最小值；

2：最大速度；

3：最大加速度；

59、 Update(type);//刷新数据，用于同步进程间数据与状态，否则 Modbus 设备可能不会更新坐标；type 参数保留；

四、Modbus 扩展

1、 Modbus 说明：

1) Modbus 通讯协议是一种主机与设备之间的通讯协议，主机一般做为控制发起者，设备处于受控地位，设备粗象为多个寄存器，其中包括位寄存器和数据寄存器；另外也包括只读寄存器和读取寄存器。其中 0X 指令用于读写位寄存器，1X 指令读写只读位寄存器，3X 指令用于读写数据寄存器，4X 指令用于访问只读数据寄存器；
2) 山龙通用控制器采用 Modbus 通讯协议，它作为 Modbus 设备，处于受控地位，可以由触摸屏或其它主机对其进行控制；
3) 用户编程对山龙通用控制器操作时，只能写入用户地址空间，其地址范围为 0~4095，可以读取所有地址空间数据，范围为 0~9999；
4) 4096~9999 地址为系统保留，其具体定义见《山龙通用运动控制器 Modbus 协议及地址定义 V1.3》
5) 对于山龙通用控制器，其数据寄存器（3X/4X 寄存器）最小地址单元为 2 个字节，存放数据时采用小端对齐方式，既低位数据存放在前面字节，高位数据存放在后面字节，也叫做低位在前。

2、 set0XIO(addr,bvalue);//设置 0X IO 读写寄存器；

addr：modbus 寄存器地址 0~9999；

bvalue：false 复位，true 置位；

3、 get0XIO(addr);//读 0X IO 寄存器；
addr：modbusIO 寄存器，地址 0~9999；
返回：Boolean 类型值；

4、 set1XIO(addr,bvalue);//本地设置 1X 只读寄存器；
addr：1X 寄存器地址；
bvalue: false 复位，true 置位；

5、 set4XFloat(addr,fvalue);
设置 4X 读写寄存器，按 4 字节连续地址 Float 数据类型存储；
addr：4X 寄存器地址；
fvalue：写入的浮点数；

6、 get4XFloat(addr);
读取 4X 读写寄存器，按 4 字节连续地址读取，返回浮点数据；
addr：4X 寄存器地址；
返回：浮点数据；

7、 set4XUInt(addr,uvalue);
设置 4X 读写寄存器，按 4 字节连续地址存储，写入无符号整型数据；
addr：4X 寄存器地址；
uvalue：写入的数据；

8、 get4XUInt(addr);
读取 4X 读写寄存器，按 4 字节连续地址读取，返回无符号整型数据；

9、 set4XInt(addr,ivalue);
设置 4X 读写寄存器，按 4 字节连续地址存储，写入整型数据；
addr：4X 寄存器地址；
ivalue：写入的整型数据；

10、 get4XInt(addr);
读取 4X 读写寄存器，按 4 字节连续地址读取，返回整型数据；
addr：4X 寄存器地址；
返回：整型数据；

11、 set4XUShort(addr,uvalue);
设置 4X 读写寄存器，按 2 字节存储，2 字节为 Modbus 寄存器的最小单位；写入无符号短整型；
addr：4X 寄存器地址；
uvalue：写入的整型数据（0~65535）；

12、 get4XUShort(addr);

读取 4X 读写寄存器，按 2 字节读取，返回无符号整型数据；

addr：4X 寄存器地址；

返回：整型数据（0~65535）；

13、 set4XShort(addr,ivalue);

设置 4X 读写寄存器，按 2 字节存储，写入短整型数据；

addr：4X 寄存器地址；

ivalue：整型数据（-32767~32768）；

14、 get4XShort(addr);

读取 4X 读写寄存器，按 2 字节读取，返回短整型数据；

addr：4X 寄存器地址；

返回：整型数据（-32767~32768）

15、 set4XString(addr,str);

设置字符串到 4X 读写寄存器，按连续地址存储，字符未写入'\0';

addr：4X 寄存器地址；

str：'\0'结尾的字符串；

16、 get4XString(addr);

读取 4X 读写寄存器，按字符串读取，'\0'结尾,返回字符串；

addr：4X 寄存器地址；

返回：'\0'结尾的字符串；

17、 set3XFloat(addr,fvalue);

设置本地 3X 只读寄存器，按 4 字节连续地址存储，写入浮点数据；

addr：3X 寄存器地址；

fvalue：浮点数据；

18、 set3XUInt(addr,uvalue);

设置本地 3X 只读寄存器，按 4 字节连续地址存储，写入无符号整型数据；

addr：3X 寄存器地址；

uvalue：写入的无符号整型数据（0~4294967295）；

19、 set3XInt(addr,ivalue);

设置本地 3X 只读寄存器，按 4 字节连续地址存储，写入整型数据；

addr：3X 寄存器地址；

ivalue：写入的整型数据（-2147483647~2147483648）；

20、 set3XUShort(addr,uvalue);

设置本地 3X 只读寄存器，按 2 字节存储，写入无符号短整型数据(0~65535);

addr：3X 寄存器地址；

uvalue:整型数据 0~65535；

21、 set3XShort(addr,ivalue);

设置本地 3X 只读寄存器，按 2 字节存储，写入短整型数据(-32767~32768);

addr：3X 寄存器地址；

ivalue：整型数据（-32767~32768）

22、 set3XString(addr,str);

设置本地 3X 只读寄存器，连续地址写入字符串，'\0'结尾；

addr：3X 寄存器地址；

str：'\0'结尾的字符串；

23、 getUserCMD();

获取用户交互命令，脚本中调用此函数可以主动获取用户发送的命令，或者用户改变了 modbus 寄存器的事件，因此脚本中不用一直查询寄存器是否有变化，只需获取此消息，就能得知哪个寄存器值有变化。

需要注意的是，此命令获取了寄存器值变化后，立即去读取该寄存器值可能存在延后。

返回：整型数据(ret)按 16 进制表示；

最低 0~7 位为命令类型(ret & 0x000000FF)；

1：0X 单个寄存器写命令；

2：4X 单个寄存器（2 字节）写命令；

3：0X 多个 IO 寄存器写命令；

4：4X 多个寄存器写命令；

5：系统停止，退出脚本运行状态；

6：开始运行脚本；

7：暂停脚本运行；

8：恢复脚本运行；

9：脚本单步运行；

10：调试运行；

11：脚本中止运行；

第 8 位~15 位为寄存器长度((ret&0x0000FF00) >> 8);

第 16~31 位为寄存器地址((ret & 0xFFFF0000) >> 16);

24、 saveRegData();//保存 Modbus 寄存器数据到磁盘永久存储；

五、系统函数扩展

1、 mDelay(ivalue);//延时指定毫秒；

ivalue:毫秒数；

2、 uSleep(ivalue);//延时指定微秒；

ivalue:微秒数；

3、 uartOpen(strname,nspeed,nbit,nevent,nstop,n485);

打开串口

strname:串口设备名，如/dev/ttyO3;

nspeed:波特率，如 115200;

nbit:数据位，如 8 指 8 位数据位；
nevent:校验类型，N：78，E：69；
nstop:停止位，1；
n485:是否设置为 485 接口，0：RS232，1：RS485；
返回：打开的文件句柄，整型，大于等于 0 有效；

4、 uartClose(nfd);//关闭串口文件；
nfd:文件句柄，由 uartOpen 返回；

5、 uartSend(nfd,strcmd);//发送串口命令；
nfd:文件句柄，由 uartOpen 返回；
strcmd:发送出去的字符串；
返回：发送出去的字节数，大于 0 有效；

6、 uartRecv(nfd,ntimeout);//接收串口消息；
nfd:文件句柄，由 uartOpen 返回；
ntimeout:设置的超时时间，超过此时间后命令返回；
返回值：接收到的消息字符串；为空接收失败；

7、

六、机器人编程实现

下面为完整 Javascript 实现的打磨机实现源代码，请参照该代码；

```
var cmdtype = 0;//接收到的命令类型；
var cmdaddr = 0;//modbus 寄存器地址；
var bstartwork = false;//是否已开始工作；
var runteachindex = 0;//自动运行时，当前示教点；
var teachpoint;//示教数据数组；
var teachindex = 0;//当前操作的示教数据点；
var pagestart = 0;//示教点位按页查询时的页面始地址；
var dirarray;//当前浏览目录的文件数组；
var dirstart = 0;//当前浏览目录页的始地址；
var workstation = 0;//工位编号；
var tmpcmd = 0;//临时命令；
var runstate = 1;//0:unkown,1:manual,2:auto;3:exit;
var makenum = 0;//加工次数，断电清零；
var idlenum = 0;
var lights = 0;
//var visfd = uartOpen("/dev/ttyO1",38400,8,78,1,0);//打开视觉器串口,78:N,69:E;
globalInit();
check_reg_time();
while(1)
{
    if(runstate == 1){//手动模式;
        on_cmd_manual();
        mDelay(50);
        test_io_out(0);
```

```
            idlenum++;
            if(idlenum >= 10){
                runstate = 2;
                Linuxcnc.setDIO(703,0);
                Linuxcnc.setDIO(702,1);
                Linuxcnc.setDIO(701,1);

                //uartSend(visfd,"trackp\r\n");
                //var cmd = uartRecv(visfd,200000);
            }
            else{
                //var cmd = uartRecv(visfd,100);
            }
        }
        else if(runstate == 2){//自动模式;
            run_teach_point();
            on_cmd_auto();
            mDelay(100);
            idlenum = 0;
            test_io_out(1);
        }
        else if(runstate == 3)
            break;
        else{
            mDelay(100);
        }
}

//uartClose(visfd);
stopwork();
function globalInit()
{
        Modbus.set4XShort(500,0);
        Modbus.set0XIO(28,false);
        Modbus.set0XIO(66,false);
        Linuxcnc.setState(2);//急停复位；
        Linuxcnc.setState(4);//开机
        Linuxcnc.setMode(1);//进入手动模式；
        Linuxcnc.setJointParam(3,0,Modbus.get4XFloat(1016));//X 轴最大加速度；
        Linuxcnc.setJointParam(3,1,Modbus.get4XFloat(1018));//Y 轴最大加速度
        Linuxcnc.setJointParam(3,5,Modbus.get4XFloat(1020));//ZR 轴最大加速度

        Linuxcnc.setDIO(200,1);
        Linuxcnc.setDIO(201,1);
```

```
Linuxcnc.setDIO(202,1);
Linuxcnc.setDIO(203,1);
Linuxcnc.setDIO(204,1);
Linuxcnc.setDIO(205,1);
Linuxcnc.setDIO(206,1);
Linuxcnc.setDIO(207,1);
Linuxcnc.setDIO(208,1);
Linuxcnc.setDIO(209,1);
Linuxcnc.setDIO(210,1);
Linuxcnc.setDIO(211,1);
Linuxcnc.setDIO(212,1);
Linuxcnc.setDIO(213,1);
Linuxcnc.setDIO(214,1);
Linuxcnc.setDIO(215,1);

Linuxcnc.setDIO(300,1);//通用 IO 输出；
Linuxcnc.setDIO(301,1);
Linuxcnc.setDIO(302,1);
Linuxcnc.setDIO(303,1);
Linuxcnc.setDIO(304,1);
Linuxcnc.setDIO(305,1);
Linuxcnc.setDIO(306,1);
Linuxcnc.setDIO(307,1);
Linuxcnc.setDIO(308,1);
Linuxcnc.setDIO(309,1);
Linuxcnc.setDIO(310,1);
Linuxcnc.setDIO(311,1);
Linuxcnc.setDIO(312,1);
Linuxcnc.setDIO(313,1);
Linuxcnc.setDIO(314,1);
Linuxcnc.setDIO(315,1);//通用 IO 输出；
Linuxcnc.setParams(300,-50.0);//设置实时脚本部份参数，对应 FP00；
Linuxcnc.setParams(301,-500.0);//设置实时脚本部份参数，对应 FP01；
Linuxcnc.setParams(302,-1000.0);//设置实时脚本部份参数，对应 FP02；
Linuxcnc.setParams(303,200.0);//设置实时脚本部份参数，对应 FP03；
Linuxcnc.setParams(304,-1000.0);//设置实时脚本部份参数，对应 FP04；
Linuxcnc.setParams(400,6);//设置实时脚本部份参数，对应 UP00；

Linuxcnc.setDIO(701,0);//设置逻辑 IO 输出，实际用于控制实时脚本部份输入信
号；
Linuxcnc.setDIO(702,0);//设置逻辑 IO 输出；
Linuxcnc.setDIO(703,0);//设置逻辑 IO 输出；

check_home_org();
```

```
        Linuxcnc.initPlan();//CNC 加工初始化；
        Linuxcnc.setParams(61,1.0);//主轴到达指定转速时间，1 秒；
        Linuxcnc.setParams(97,100.0);//Z 轴下刀速度 100mm/s;
        Linuxcnc.setParams(98,500.0);//Z 轴离刀速度 500mm/s;
        teach_init();
        Linuxcnc.Open("/tmp/linuxcnc.pip");//指定以管道方式打开加工文件，便于通过
POST 方式发送运动控制指令。
    }
    function check_home_org()
    {
        if(!Modbus.get0XIO(80))
            return;
        Modbus.set0XIO(60,true);//通知触摸屏准备编码器读数;
        Modbus.set4XShort(589,12);//通知触摸屏跳转到编码器页面；
        while(1){
            if(Modbus.get0XIO(66)){//触摸屏发送编码器读数给控制器用于回机械零；
                var xp = Modbus.get4XUInt(690) + Modbus.get4XUInt(672) * 131072;//
当前 X 编码器值；
                var yp = Modbus.get4XUInt(692) + Modbus.get4XUInt(674) * 131072;//
当前 Y 编码器值；
                var zp = Modbus.get4XUInt(694) + Modbus.get4XUInt(676) * 131072;//
当前 ZR 编码器值；
                var xs = Linuxcnc.getJointParam(6,0);//X 轴脉冲当量；
                var ys = Linuxcnc.getJointParam(6,1);//Y 轴脉冲当量；
                var zrs = Linuxcnc.getJointParam(6,5);//ZR 轴脉冲当量；
                var x0 = xs * ( xp - Modbus.get4XUInt(1820));//X 机械坐标；
                var y0 = ys * ( yp - Modbus.get4XUInt(1824));//Y 机械坐标；
                var z0 = zrs * ( zp - Modbus.get4XUInt(1828));//ZR 机械坐标；
                Linuxcnc.clearHome(-1);//清除各轴回零标志，可以重新回零；
                Linuxcnc.setHome(0,0.0,x0,0.0,0.0,0.0,0,2);//设置 X 轴回零参数，指定
当前点机械坐标为 x0;
                Linuxcnc.setHome(1,0.0,y0,0.0,0.0,0.0,0,2);//设置 y 轴回零参数，指定
当前点机械坐标为 y0;
                Linuxcnc.setHome(2,0.0,-60.0,0.0,0.0,0.0,0,2);//设置 Z 轴回零参数，指
定当前点机械坐标为-60;
                Linuxcnc.setHome(3,0.0,0.0,0.0,0.0,0.0,0,2);//设置 A 轴回零参数，指定
当前点机械坐标为 0;
                Linuxcnc.setHome(4,0.0,0.0,0.0,0.0,0.0,0,2);//设置 B 轴回零参数，指定
当前点机械坐标为 x0;
                Linuxcnc.setHome(5,0.0,z0,0.0,0.0,0.0,0,2);//设置 C 轴回零参数，指定
当前点机械坐标为 z0;
                Linuxcnc.goHome(-1);//所有轴执行回零动作；按上面参数设置，设置
当前点的机械坐标；
                Modbus.set0XIO(60,false);//清除标志；
```

```
                Modbus.set0XIO(66,false);//清除标志；
                break;
            }
            else
                mDelay(100);//延时 100ms;
        }
    }
    function set_curr_as_home()
    {
        var xp = Modbus.get4XUInt(690) + Modbus.get4XUInt(672) * 131072;//当前 X 编
码器值；
        var yp = Modbus.get4XUInt(692) + Modbus.get4XUInt(674) * 131072;//当前 Y 编
码器值；
        var zp = Modbus.get4XUInt(694) + Modbus.get4XUInt(676) * 131072;//当前 ZR 编
码器值；
        Modbus.set4XUInt(1820,xp);//保存当前点编码器值，用于确定机械坐标零点的
编码器值。
        Modbus.set4XUInt(1824,yp);
        Modbus.set4XUInt(1828,zp);
        Modbus.set0XIO(80,true);//设置标记，已设置机械零点；
        Modbus.saveRegData();
    }
    function on_cmd_manual()
    {
        tmpcmd = Modbus.getUserCMD();//获取用户输入；
        if(tmpcmd != 0)
        {
            cmdaddr = (tmpcmd >>> 16) & 65535;//高 16 位对应寄存器地址；
            cmdtype = tmpcmd & 255;//低 8 位对应用户输入的事件类型；
            if(cmdtype == 1){//MD_CMD_USER_SINGLE_WRITE_0X;
                var handspeed = Modbus.get4XFloat(1028);//手动移动速度
                var speedrate = Modbus.get4XFloat(1038);//手动移动倍率
                if(cmdaddr == 1){//正向移动 X 轴;
                    if(Modbus.get0XIO(cmdaddr))
                        Linuxcnc.jogCont(0,handspeed * speedrate);
                    else
                        Linuxcnc.jogStop(0);
                }
                else if(cmdaddr == 2){//负向移动 X 轴;
                    if(Modbus.get0XIO(cmdaddr))
                        Linuxcnc.jogCont(0,-handspeed * speedrate);
                    else
                        Linuxcnc.jogStop(0);
```

```
}
else if(cmdaddr == 3){//正向移动 Y 轴
    if(Modbus.get0XIO(cmdaddr))
        Linuxcnc.jogCont(1,handspeed * speedrate);
    else
        Linuxcnc.jogStop(1);
}
else if(cmdaddr == 4){//负向移动 Y 轴
    if(Modbus.get0XIO(cmdaddr))
        Linuxcnc.jogCont(1,-handspeed * speedrate);
    else
        Linuxcnc.jogStop(1);
}
else if(cmdaddr == 5){//正向移动 ZR 轴
    if(Modbus.get0XIO(cmdaddr))
        Linuxcnc.jogCont(5,handspeed * speedrate);
    else
        Linuxcnc.jogStop(5);
}
else if(cmdaddr == 6){//负向移动 ZR 轴
    if(Modbus.get0XIO(cmdaddr))
        Linuxcnc.jogCont(5,-handspeed * speedrate);
    else
        Linuxcnc.jogStop(5);
}
else if(cmdaddr == 8)//回到工位 1 的工件零点;
    Linuxcnc.sendMDI("G54 F3000 G00 X0 Y0 C0");
else if(cmdaddr == 9)//回到工位 2 的工件零点;
    Linuxcnc.sendMDI("G55 F3000 G00 X0 Y0 C0");
else if( cmdaddr == 17){//设置工位 1 的原点;
    Linuxcnc.setG5x(1,0,0.0);//设置此点 G54 工件原点;
    Linuxcnc.setG5x(1,1,0.0);
    Linuxcnc.setG5x(1,2,60.0);//Z 轴设置偏移植,防止处于极限位置;
    Linuxcnc.setG5x(1,5,0.0);
    Modbus.set4XFloat(1066,Linuxcnc.getParams(18,0));//另外保存工
件偏移，用于触摸屏显示;
    Modbus.set4XFloat(1070,Linuxcnc.getParams(18,1));
    Modbus.set4XFloat(1074,Linuxcnc.getParams(18,5));
    Modbus.saveRegData();
}
else if( cmdaddr == 18){//设置工位 2 的原点;
    Linuxcnc.setG5x(2,0,0.0);//设置此点 G55 工件原点;
    Linuxcnc.setG5x(2,1,0.0);
    Linuxcnc.setG5x(2,2,60.0);//Z 轴设置偏移植,防止处于极限位置;
```

```
            Linuxcnc.setG5x(2,5,0.0);
            Modbus.set4XFloat(1086,Linuxcnc.getParams(18,0));//另外保存工
件偏移，用于触摸屏显示；
            Modbus.set4XFloat(1090,Linuxcnc.getParams(18,1));
            Modbus.set4XFloat(1094,Linuxcnc.getParams(18,5));
            Modbus.saveRegData();
        }
        else if(cmdaddr == 20){//保存当前编码器值为机械零；
            //if(Modbus.get0XIO(20)){
                set_curr_as_home();
            //}
        }
        else if(cmdaddr == 23){//显示上一页示教数据；
            if(teachpoint != null && Modbus.get0XIO(23))
            {
                pagestart -= 5;//页面起始地址减小一页；
                if(pagestart < 0)
                    pagestart = 0;
                show_curr_page();//显示当前页到寄存器；
            }
        }
        else if(cmdaddr == 24){//显示下一页示教数据；
            if(teachpoint != null && Modbus.get0XIO(24) )
            {
                pagestart += 5;
                if(pagestart >= teachpoint.length)
                    pagestart = 0;
                show_curr_page();
            }
        }
        else if( cmdaddr    == 26){//显示下一页当前目录文件;
            if(dirarray != null && Modbus.get0XIO(26) )
            {
                dirstart += 10;
                if(dirstart >= dirarray.length)
                    dirstart = 0;
                show_curr_dir();//显示当前页文件；
            }
        }
        else if(cmdaddr == 25){//显示上一页当前目录文件；
            if(dirarray != null && Modbus.get0XIO(25))
            {
                dirstart -= 10;
                if(dirstart < 0)
```

```
                    dirstart = 0;
                    show_curr_dir();
                }
            }
            else if(cmdaddr == 33){//跳转到指令行的示教数据；
                if(Modbus.get0XIO(33))
                {
                    var id = Modbus.get4XInt(458) - 1;
                    if(id >= 0 && id < teachpoint.length)
                        teachindex = id;
                    show_curr_teach();//显示当前行示教数据；
                    Modbus.set4XShort(589,17);//跳转到示教数据显示画面；
                }
            }
        }
        else if(cmdtype == 4){//MD_CMD_USER_MUTI_WRITE_4X
            if(cmdaddr == 500)
                on_cmd_reg500();
            else if(cmdaddr == 330){//选择工位；
                workstation = (workstation == 0) ? 1 : 0;
            }
            else if(cmdaddr == 1016)//X 轴加速度
                Linuxcnc.setJointParam(3,0,Modbus.get4XFloat(cmdaddr));
            else if(cmdaddr == 1018)//Y 轴加速度
                Linuxcnc.setJointParam(3,1,Modbus.get4XFloat(cmdaddr));
            else if(cmdaddr === 1020)//ZR 轴加速度
                Linuxcnc.setJointParam(3,5,Modbus.get4XFloat(cmdaddr));
        }
        else if(cmdtype == 5){//退出加工，停止；
            Linuxcnc.Abort();
            runstate = 3;//改变状态，退出脚本执行；
        }
        else if(cmdtype == 11){//中止当前执行的操作。
            Linuxcnc.Abort();
            runstate = 3;//改变状态，退出脚本执行；
        }
    }
    else{//手动状态，更新机械坐标；
        Linuxcnc.Update();
    }
}
function copy_mfile()
{//根据 IO 状态，拷贝对应文件到 SD 卡指定目录或从 SD 卡拷出；
    var addr;
```

```
    for( addr = 100; addr < 110; addr++){
        if(Modbus.get0XIO(addr)){//拷贝此文件;
            var index = dirstart + addr - 100;
            if(index < dirarray.length && index >= 0){
                var dfile = dirarray[index];
                dfile.open("read");
                if(Modbus.get0XIO(30)){//copy from usb to sdcard;
                    var dname = "/media/sdcard/processfiles/" + dfile.name;
                    dfile.copyTo(dname);
                }
                else{//copy from sdcard to usb;
                    var dname = "/media/usb/" + dfile.name;
                    dfile.copyTo(dname);
                }
                dfile.close();
            }
        }
    }
}
function delete_mfile()
{
    if(Modbus.get0XIO(40)){
        var addr;
        var bpress = 0;
        Modbus.set4XShort(589,30);//跳转到删除确认对话框；
        while(1){//等待直到用户选择；
            if(Modbus.get0XIO(37)){//如果用户确认；
                bpress = 1;
                break;
            }
            else if(Modbus.get0XIO(38)){//如果用户取消；
                bpress = 2;
                break;
            }
            else
                mDelay(10);
        }
        if(bpress == 1){//确认删除
            Modbus.set0XIO(40,false);
            Modbus.set0XIO(37,false);
            Modbus.set4XShort(589,14);
            for( addr = 100; addr < 110; addr++){
                if(!Modbus.get0XIO(30)){//only delete the sdcard file;
                    if(Modbus.get0XIO(addr)){//delete the file;
```

```
                        var index = dirstart + addr - 100;
                        if(index < dirarray.length && index >= 0){
                            var dfile = dirarray[index];
                            dfile.remove();
                        }
                    }
                }
            }
        }
        if(bpress == 2){//取消
            Modbus.set0XIO(40,false);
            Modbus.set0XIO(37,false);
            Modbus.set4XShort(589,14);
            Modbus.set4XInt(500,0);
        }
    }
}
function select_work_file()
{
    var addr;
    for( addr = 100; addr < 110; addr++){//选择指定文件为当前加工文件;
        if(!Modbus.get0XIO(30)){//只能从 SD 卡目录下选择;
            if(Modbus.get0XIO(addr)){//select the file;
                var index = dirstart + addr - 100;
                if(index < dirarray.length && index >= 0){
                    var tstr = dirarray[index].name;
                    var ic = tstr.indexOf('.');
                    Modbus.set4XString(1439,"                ");
                    Modbus.set4XString(1439,tstr.substring(0,ic));
                    Modbus.saveRegData();
                    teach_init();
                    break;
                }
            }
        }
    }
}
function on_cmd_auto()
{
    tmpcmd = Modbus.getUserCMD();//自动模式下获取用户输入；
    if(tmpcmd != 0)
    {
        cmdaddr = (tmpcmd >>> 16) & 65535;
        cmdtype = tmpcmd & 255;
```

```
    if(cmdtype == 1){//MD_CMD_USER_SINGLE_WRITE_0X;
        if(cmdaddr == 140){//停止工作;
            stopwork();
            runstate = 1;
        }
    }
    else if(cmdtype == 4){//MD_CMD_USER_MUTI_WRITE_4X
        if(cmdaddr == 500){
            tmpcmd = Modbus.get4XInt(500);
            if(tmpcmd == 404){//停止工作；
                stopwork();
                runstate = 1;
            }
            else if(tmpcmd == 412){
                stopwork();
                runstate = 1;
            }
            else if(tmpcmd == 1020){//开关 R 轴打磨;
                if(Modbus.get0XIO(11))
                    Linuxcnc.setParams(125,1);
                else
                    Linuxcnc.setParams(125,0);
            }
        }
    }
    else if(cmdtype == 5){//上层发送来的停止工作；
        stopwork();
        runstate = 3;
    }
    else if(cmdtype == 7){//暂停
        Linuxcnc.Pause();
    }
    else if(cmdtype == 8){//暂停恢复；
        Linuxcnc.Resume();
    }
    else if(cmdtype == 11){//上层发送来的停止命令
        stopwork();
        runstate = 3;
    }
}
else{//自动状态，更新当前速度;
    Modbus.set4XFloat(1002,Linuxcnc.getParams(0,1));//更新显示当前速率；
    Linuxcnc.Update();//空闲时更新坐标显示；
}
```

```
}
function test_io_out(a)
{
	if(a == 0)
	{
		Linuxcnc.setDIO(300 + lights,0);
		lights++;
		var dioin = Linuxcnc.getParams(30,400);

		if(lights >= 16)
			lights = 0;
		Linuxcnc.setDIO(300 + lights,1);
	}

	var ivalue;
	var din = 400;
	var dout = 200;

	while(din < 424){
		ivalue = Linuxcnc.getParams(30,din);
		if(ivalue == 0)
			Modbus.set0XIO(dout,false);
		else
			Modbus.set0XIO(dout,true);
		dout++;
		din++;
	}
}

function on_cmd_reg500()
{
	tmpcmd = Modbus.get4XInt(500);
	if(tmpcmd >= 700 && tmpcmd <= 715)
	{// 下一行，上一行，跳转行，插入行，删除行，定位当前行，查看程序点位，
新建文件
		if(teachpoint != null)
			do_teach_edit();
	}
	else if(tmpcmd == 400){//启动
		runstate = 2;
	}
	else if(tmpcmd == 412){//回各工位原点;
		if(workstation == 0){
			var strcmd = "G54 G01 X0 Y0 C0";
```

```
                Linuxcnc.sendMDI(strcmd);
            }else{
                var strcmd = "G55 G01 X0 Y0 C0";
                Linuxcnc.sendMDI(strcmd);
            }
        }
        else if(tmpcmd == 900){//新建文件;
            teach_init();
        }
        else if(tmpcmd == 901){//显示目录文件;
            dir_list();
            show_curr_dir();
        }
        else if(tmpcmd == 902){//拷贝文件;
            if(dirarray != null)
                copy_mfile();
        }
        else if(tmpcmd == 903){//选择当前加工文件；;
            if(dirarray != null)
                select_work_file();
        }
        else if(tmpcmd == 904){//删除文件;
            if(dirarray != null)
                delete_mfile();
        }
        else if(tmpcmd == 1000){//保存寄存器数据;
            Modbus.saveRegData();
        }
        else if(tmpcmd == 1603){//移动到工件原点;
            if(workstation == 0){
                Linuxcnc.sendMDI("G54 F3000 G00 X0 Y0 C0");
            }else{
                Linuxcnc.sendMDI("G55 F3000 G00 X0 Y0 C0");
            }
        }
        else if(tmpcmd == 1020){//开关 R 轴打磨;
            if(Modbus.get0XIO(11))
                Linuxcnc.setParams(125,1);
            else
                Linuxcnc.setParams(125,0);
        }
    }

    function show_curr_dir()
```

```
{
    var i;
    var wi = 0;
    if(dirarray != null){
        for(;wi < 10;wi++){//先清除；
            Modbus.set4XString(1500 + wi * 6,"              ");
        }
        for (i = dirstart;i < dirarray.length; i++){
            wi = i - dirstart;
            var ic = dirarray[i].name.indexOf('.');
            var dname = dirarray[i].name.substring(0,ic);//去除扩展名；
            if(dname.length >= 8)
                Modbus.set4XString(1500 + wi * 6,dname.substring(0,7));//仅显示
前 8 个字符；
            else
                Modbus.set4XString(1500 + wi * 6,dname);
        }
    }
}
function dir_list()
{//获取当前目录下指定格式文件，并存放到数组中；
    if(Modbus.get0XIO(30)){//usb;
        var pdir = new File("/media/usb");
        if(pdir != null)
            dirarray = pdir.list(/\w.tab/i);//正则表达式显示扩展名为.tab 的所有文
件，忽略大小写；
    }
    else{//sdcard;
        var pdir = new File("/media/sdcard/processfiles");
        if(pdir != null)
            dirarray = pdir.list(/\w.tab/i);//正则表达式显示扩展名为.tab 的所有文
件，忽略大小写；
    }
}
function show_curr_page()
{//显示当前页示教点位数据；
    var i;
    var id = 0;
    var offd = 0;
    for(i = pagestart; i < pagestart + 5 && i < teachpoint.length; i++){
        var pstr = teachpoint[i];
        var j = pstr.indexOf("X");
        id = i - pagestart;
        offd = id * 6;
```

```
            if(j >= 0){
                var s = pstr.substring(j+1);
                var dvalue = parseFloat(s);
                Modbus.set4XFloat(1200 + offd,dvalue);
            }
            j = pstr.indexOf("Y");
            if(j >= 0){
                var s = pstr.substring(j+1);
                var dvalue = parseFloat(s);
                Modbus.set4XFloat(1202 + offd,dvalue);
            }
            j = pstr.indexOf("C");
            if(j >= 0){
                var s = pstr.substring(j+1);
                var dvalue = parseFloat(s);
                Modbus.set4XFloat(1204 + offd,dvalue);
            }
            Modbus.set4XUShort(430 + i- pagestart ,i+1);
            id++;
        }

        for( ; id < 5; id++){//当前页其余清零；
            offd = id * 6;
            Modbus.set4XFloat(1200 + offd,0.0);
            Modbus.set4XFloat(1202 + offd,0.0);
            Modbus.set4XFloat(1204 + offd,0.0);
            Modbus.set4XUShort(430 + id ,0);
        }
    }

    function parsefloatStr(fs,sf)
    {
        var i = fs.indexOf(sf);
        if( i < 0){
            var tolow = sf.toLowerCase();
            i = fs.indexOf(tolow);
        }
        if( i >= 0){
            s = fs.substring(i+1);
            return parseFloat(s);
        }
        return 0.0;
    }
```

```
function parseintStr(fs,sf)
{
	var i = fs.indexOf(sf);
	if( i < 0){
		var tolow = sf.toLowerCase();
		i = fs.indexOf(tolow);
	}
	if( i >= 0){
		s = fs.substring(i+1);
		return parseInt(s);
	}
	return -1;
}
/*
已知园上两点和半径，根据采用的是 G02 还是 G03 指令，找到园心位置；
园心坐标从数组中返回；
  [0]存放元素个数；
 [1]存放园心 X 坐标；
 [2]存放园心 Y 坐标；
*/
function calc_center(x1,y1,x2,y2,dr,g)
{
	var dx = x2 - x1;
	if(dx > 0.0001 || dx < -0.0001){//避免分母为 0；
		var c1 = (x2*x2 - x1*x1 + y2*y2 - y1*y1) / (2 *dx);
		var c2 = (y2 - y1) / dx;
		var a = (c2*c2 + 1);
		var b = (2 * x1*c2 - 2 * c1*c2 - 2 * y1);
		var c = x1*x1 - 2 * x1*c1 + c1*c1 + y1*y1 - dr*dr;
		var bc = b*b - 4 * a*c;
		if(bc > 0.0)
		{
			var yc0 = (-b + Math.sqrt(b*b - 4 * a*c)) / (2 * a);
			var xc0 = c1 - c2 * yc0;
			//根据面积的正负，决断是逆时针还是顺时针；当
P(xc0,yc0)--->P(x1,y1)--->P(x2,y2)方向求面积时，如果面积为负，则为顺时针；
			var s0 = xc0 * y1 + x1 * y2 + x2 * yc0 - xc0 * y2 - x1 * yc0 - x2 * y1;
			var yc1 = (-b - Math.sqrt(b*b - 4 * a*c)) / (2 * a);
			var xc1 = c1 - c2 * yc1;
			//var s1 = xc1 * y1 + x1 * y2 + x2 * yc1 - xc1 * y2 - x1 * yc1 - x2 * y1;
			if( dr > 0.0){//G02 R+ S-;G03 R+ S+;
				if( g == 3 ){
					if( s0 > 0.0)
						return Array(2,xc0,yc0);
```

```
                    else
                        return Array(2,xc1,yc1);
                }else{
                    if( s0 < 0.0)
                        return Array(2,xc0,yc0);
                    else
                        return Array(2,xc1,yc1);
                }
            }
            else{//G02 R- S+;G03 R- S-;
                if( g == 3){
                    if( s0 < 0.0)
                        return Array(2,xc0,yc0);
                    else
                        return Array(2,xc1,yc1);
                }else{
                    if(s0 > 0.0)
                        return Array(2,xc0,yc0);
                    else
                        return Array(2,xc1,yc1);
                }
            }
        }else if(bc == 0.0){
            var yc0 = (-b) / (2 * a);
            var xc0 = c1 - c2 * yc0;
            return new Array(2,xc0,yc0);
        }
    }else{
        var yc = (y2 + y1)/2;
        var dy = (y2 > y1) ? ((y2 - y1)/2.0) : ((y1 - y2)/2.0);
        if(dr > dy || dr < -dy ){
            var sqdy = Math.sqrt(dr * dr - dy * dy);
            var xc0 = x1 - sqdy;
            var xc1 = x1 + sqdy;
            var s0 = xc0 * y1 + x1 * y2 + x2 * yc - xc0 * y2 - x1 * yc - x2 * y1;
            //var s1 = xc1 * y1 + x1 * y2 + x2 * yc - xc1 * y2 - x1 * yc - x2 * y1;
            if( dr > 0.0){//G02 R+ S-;G03 R+ S+;
                if( g == 3){
                    if(s0 > 0.0)
                        return new Array(2,xc0,yc);
                    else
                        return new Array(2,xc1,yc);
                }else{
                    if( s0 < 0.0)
```

```
                    return new Array(2,xc0,yc);
                else
                    return new Array(2,xc1,yc);
            }
        }else{//G02 R- S+;G03 R- S-;
            if( g == 3){
                if( s0 < 0.0)
                    return new Array(2,xc0,yc);
                else
                    return new Array(2,xc1,yc);
            }else{
                if( s0 > 0.0)
                    return new Array(2,xc0,yc);
                else
                    return new Array(2,xc1,yc);
            }
        }
    }else if(dr == dy || dr == -dy ){
            return new Array(2,x1,yc);
    }
  }
  return null;
}

/*
在加工数据中插入 M63 P02 指令，使离终点未走的规迹长度为 d;同时不改变轨迹；
注意，仅支持带 R 参数的园弧指令；
*/
function insert_dist_before(ig,d)
{
    var x0 = 0.0;
    var y0 = 0.0;
    var r = 0.0;
    var x1,y1,g1,r1;
    var b1 = false;
    var ret = 0;

    var g = -1;
    var i = ig - 1;
    for( ; i >= 0; i--){
        var pstr = teachpoint[i];
        g = parseintStr(pstr,"G");
        x0 = parsefloatStr(pstr,"X");
        y0 = parsefloatStr(pstr,"Y");
```

```
if( g == 2 || g == 3){
    r = parsefloatStr(pstr,"R");
}

if( g < 0 || g > 3)
    continue;

if( !b1 ){
    g1 = g;
    x1 = x0;
    y1 = y0;
    r1 = r;
    b1 = true;
    continue;
}

if(g1 == 2 || g1 == 3 ){//园弧；
    //x0 = xc + rcosq0;
    //y0 = yc + rsinq0;
    //x1 = xc + rcosq1;
    //y1 = yc + rsinq1;
    var xx = x1 - x0;
    var yy = y1 - y0;
    var dd = Math.sqrt( xx * xx + yy * yy);
    var dq = Math.asin(dd/(2 * r1));
    dq = (r1 > 0) ? (2.0 * dq) : (-2.0 * dq);//计算园弧始点和终点夹角大小；
    var d0 = ( r1 > 0) ? (dq * r1) : ((6.2832 - dq) * r1);//计算园弧始点和终点
走过的弧长；
    if( d0 > d ){//分割成两条园弧指令，并在两指令间插入 M 反转指令；
        var axy = calc_center(x0,y0,x1,y1,r1,g1);
        if(axy != null){
            var q0 = Math.asin((x0 + y0 - axy[1] - axy[2])/(1.414 * r1)) -
3.14159/4;//求出 P(x0,y0)的极坐标角度值 q0;
            d0 = (r1 > 0.0) ? ((d0 - d)/r1) : (( d - d0)/r1);
            if( g1 == 2)
                q0 -= d0;//P（x0,y0)的角度减去分割的角度，为分割点
角度；
            else
                q0 += d0;

            var x2 = axy[1] + r1 * Math.cos(q0);//由园的极坐标方程求出
园上点的 XY 坐标；
            var y2 = axy[2] + r1 * Math.sin(q0);
            var strv = "G" + g1 + "X" + x2.toFixed(3) + "Y" + y2.toFixed(3) +
```

```
"R" + r1.toFixed(3);
                        teachpoint.splice(i+1,0,strv);//把园弧指令分解为二个；
                        teachpoint.splice(i+2,0,"M63 P02");//控制 2 号输出 IO 端口打开，置位，于是反转；
                        ret = 2;
                        break;
                    }
                }
                else if( d0 == d ){//插入 M 指令，控制反转；
                    teachpoint.splice(i+1,0,"M63 P02");//控制 2 号输出 IO 端口打开，置位，于是反转；
                    ret = 1;
                    break;
                }
                else{
                    d -= d0;
                }
            }
            else if(g1 == 0 || g1 == 1){//直线指令；
                var xx = x1 - x0;
                var yy = y1 - y0;
                var dd = Math.sqrt( xx * xx + yy * yy);
                if( dd > d){//把此指令分解为 2 条直线指令；
                    dd -= d;
                    if( xx > 0.0001 || xx < -0.0001){
                        var k = yy/xx;
                        var b = y0 - k * x0;
                        var a2 = k * k + 1;
                        var b2 = 2 * (b - y0) * k - 2 * x0;
                        var c2 = x0 * x0 + (b - y0) * ( b - y0) - dd * dd;
                        //a2 * x * x + b2 * x + c2 = 0;
                        var bc2 = b2 * b2 - 4 * a2 * c2;
                        if(bc2 > 0.0){
                            bc2 = Math.sqrt(bc2);
                            var x2 = (-b2 - bc2 )/(2 * a2);
                            if( ( x2 - x0) * ( x2 - x1) > 0.0 ){
                                x2 = (-b2 +bc2)/(2 * a2);
                            }
                            y2 = k * x0 + b;//插入此点，和反转指令；
                            var strv = "G" + g1 + "X" + x2.toFixed(3) + "Y" + y2.toFixed(3);
                            teachpoint.splice(i+1,0,strv);//把园弧指令分解为二个；
                            teachpoint.splice(i+2,0,"M63 P02");//控制 2 号输出 IO 端口打开，置位，于是反转；
```

```
                    ret = 2;
                }
            }else{
                var y2 = ( y0 < y1) ? (y0 + dd) : (y0 - dd);
                var strv = "G" + g1 + "Y" + y2.toFixed(3);
                teachpoint.splice(i+1,0,strv);//把园弧指令分解为二个；
                teachpoint.splice(i+2,0,"M63 P02");//控制 2 号输出 IO 端口打
开，置位，于是反转；
                ret = 2;
            }
            break;
        }else{//去掉此距离，寻找下一条指令；
            d -= dd;
        }
    }
    x1 = x0;
    y1 = y0;
    g1 = g;
    r1 = r;
  }
  return ret;
}

function ModifyCake()
{
  var i = 0;
  var bfirst = true;
  for( ; i < teachpoint.length ; i++ ){
    var pstr =    teachpoint[i];
    var pg0 = pstr.match(/[gG]0*[ xyzc]/i);//查找 G00 字符串，领头字符为 g/G,
后面紧跟 1 个或多个 0,后面再跟着空格、x,y,z,c,不区分大小写；
    if( pg0 != null ){
      if( bfirst){
        bfirst = false;
        teachpoint.splice(i+1,0,"M62 P01");//气缸下降；
        teachpoint.splice(i+2,0,"M62 P02");//出料电机正转出料；
        strv = "G4 P" + Modbus.get4XFloat(2416);
        teachpoint.splice(i+3,0,strv);//出料延时；
        i += 3;
        continue;
      }
      var irt = insert_dist_before(i,1270.45);//回退距离，提高反转出料电机；
Modbus.get4XFloat(2408)
      if(irt > 0)
```

```
                i += irt;
            teachpoint.splice(i,0,"M63 P01");//抬气缸；
            var strv = "G4 P" + Modbus.get4XFloat(2412);
            teachpoint.splice(i+1,0,strv);//回料延时；
            i += 2;
            teachpoint.splice(i+1,0,"M62 P01");//气缸下降；
            teachpoint.splice(i+2,0,"M62 P02");//出料电机正转出料；
            strv = "G4 P" + Modbus.get4XFloat(2416);
            teachpoint.splice(i+3,0,strv);//出料延时；
            i += 3;
        }
    }
}

function show_curr_teach()
{//显示当前行示教数据；
    if(teachindex >= 0 && teachindex < teachpoint.length){
        var pstr = teachpoint[teachindex];
        var i = pstr.indexOf("X");
        if(i >= 0){
            var s = pstr.substring(i+1);
            var dvalue = parseFloat(s);
            Modbus.set4XFloat(1330,dvalue);
        }
        else
            Modbus.set4XFloat(1330,0.0);

        i = pstr.indexOf("Y");
        if(i >= 0){
            var s = pstr.substring(i+1);
            var dvalue = parseFloat(s);
            Modbus.set4XFloat(1334,dvalue);
        }
        else
            Modbus.set4XFloat(1334,0.0);
        i = pstr.indexOf("C");
        if(i >= 0){
            var s = pstr.substring(i+1);
            var dvalue = parseFloat(s);
            Modbus.set4XFloat(1340,dvalue);
        }
        else
            Modbus.set4XFloat(1340,0.0);
    }
```

```
        else{
            Modbus.set4XFloat(1330,0.0);
            Modbus.set4XFloat(1334,0.0);
            Modbus.set4XFloat(1340,0.0);
        }
}
function do_teach_edit()
{//进行示教数据编辑与维护；
        if(tmpcmd == 700){//下一行；；
            teachindex += 1;
            if(teachindex >= teachpoint.length)
                teachindex = teachpoint.length - 1;
        }
        else if(tmpcmd == 701){//上一行;
            teachindex -= 1;
            if(teachindex < 0)
               teachindex = 0;
        }
        else if(tmpcmd == 702){//转到指定行;
            var id = Modbus.get4XInt(458) - 1;
            if(id >= 0 && id < teachpoint.length)
                teachindex = id;
            Modbus.set4XShort(589,17);
        }
        else if(tmpcmd == 703){//在指定位置前面插入;
            var id = Modbus.get4XInt(458) - 1;
            insert_teach(id);
            Modbus.set4XShort(589,17);
        }
        else if(tmpcmd == 710){//添加记录;
            insert_teach(-1);
            Modbus.set4XShort(589,17);
        }
        else if(tmpcmd == 704){//删除当前行;
            //teachpoint.splice(teachindex,1);
            //if(teachindex >= teachpoint.length)
            //    teachindex = 0;
            //Modbus.set4XShort(589,17);
            ModifyCake();
            teach_save();
        }
        else if(tmpcmd == 705){//移动到当前行示教数据指定坐标处;
            if(teachindex >= 0 && teachindex < teachpoint.length){
                Linuxcnc.sendMDI("G54 F3000\n");
```

```
                Linuxcnc.sendMDI(teachpoint[teachindex]);
            }
        }
        else if(tmpcmd == 706){//显示当前页;
            show_curr_page();
        }
        else if(tmpcmd == 707){//退出，并重新装载示教数据，前面修改取消;
            teach_init();
            return;
        }
        else if(tmpcmd == 708){//退出并保存修改后的示教数据;
            teach_save();
        }
        else if(tmpcmd == 709){//清除所有示教数据；;
            teachindex = 0;
            while(teachpoint.length){
                teachpoint.shift();
            }
            Modbus.set4XShort(589,17);
        }
        show_curr_teach();
        Modbus.set4XInt(346,teachindex + 1 );
        Modbus.set4XInt(344,teachpoint.length);
}
function teach_init()
{//从指定文件重新初始化数据；
        var savefile = Modbus.get4XString(1439);
        if(savefile.length > 0)
        {//获取指定的文件名；
            var str =    "/media/sdcard/processfiles/" + savefile + ".tab";
            var pfile = new File(str);
            if(pfile.exists){//如果文件存在，读取数据到数组；
                pfile.open("read");
                teachpoint = pfile.readAll();
                pfile.close();
            }
            else{//否则新建空的数组；
                teachpoint = new Array();
                teachindex = 0;
            }
        }
        else{//如果没有指定文件，则使用默认文件；
            var str =    "/media/sdcard/processfiles/point.tab";
            var pfile = new File(str);
```

```
            if(pfile.exists){
                pfile.open("read");
                teachpoint = pfile.readAll();
                pfile.close();
            }
        }
        if(teachpoint != null){
            teachindex = 0;
            show_curr_teach();
            Modbus.set4XInt(346,teachindex + 1 );//输出当前示教行号；
            Modbus.set4XInt(344,teachpoint.length);//输入当前示教总行号；
        }
}
function teach_save()
{//保存示教数据；
        if(teachpoint.length > 0)
        {
            var savefile = Modbus.get4XString(1439);
            if(savefile.length <= 0)
                savefile = "point";

            var strname = "/media/sdcard/processfiles/" + savefile + ".tab";
            var pfile = new File(strname);
            pfile.open("write,create,replace");
            pfile.writeAll(teachpoint);
            pfile.close();
        }
}
function insert_teach(a)
{//插入示教数据；
        var x = Linuxcnc.getParams(21,0);
        var y = Linuxcnc.getParams(21,1);
        var z = Linuxcnc.getParams(21,5);
        var strv = "G01 " + "X" + x.toFixed(3) + "Y" + y.toFixed(3) + "C" + z.toFixed(3);
        if(a < 0 || a >= teachpoint.length){
          teachpoint.push(strv);
          teachindex = teachpoint.length - 1;
        }
        else{
          teachpoint.splice(a,0,strv);
          teachindex = a + 1;
        }
        strv = null;
}
```

```
function modify_teach(a)
{//修改示教数据；
    var x = Linuxcnc.getParams(21,0);
    var y = Linuxcnc.getParams(21,1);
    var z = Linuxcnc.getParams(21,5);
    var strv = "G01 " + "X" + x.toFixed(3) + "Y" + y.toFixed(3) + "C" + z.toFixed(3);
    if(a < 0 || a >= teachpoint.length){
      teachpoint.push(strv);
    }
    else{
      teachpoint[a] = strv;
    }
    strv = null;
}
function run_teach_point()
{//运行示教数据；
    var ret = -1;
    if(!bstartwork){//如果是刚启动加工

        if(!Linuxcnc.getParams(30,600)){//获取逻辑 IO 输入，实际检测实时脚本部份传送带是否就位对应 IO 输出 P04；
            return;
        }

        var ads = Modbus.get4XInt(650);
        var fadj = Modbus.get4XFloat(1010);

        if(ads > 0){//指定了砂轮磨损补偿，则计算各轴补偿量，设置刀具偏移进行补偿；
            var ncnt = Math.floor(makenum / ads);//取整数部份
            fadj = ncnt * fadj;
            var adjx = fadj * 0.5;
            var adjy = fadj * 0.4;
            var adjz = fadj * 0.3;
            Linuxcnc.setToolTouch(0,1,0,adjx);//设置 X 轴 1 号刀具偏移；
            Linuxcnc.setToolTouch(0,1,1,adjy);
            Linuxcnc.setToolTouch(0,1,5,adjz);
        }

        startwork();
        if(workstation == 0){//设置工位 1 加工，指定加工速度
            var feed = Modbus.get4XFloat(1078);
            if(feed == 0.0)
                feed = 3000.0;
```

```
                var strstart = "G54 F" + feed.toFixed(2);
                Linuxcnc.postMDI(strstart);
            }
            else if(workstation == 1){//设置工位 2 加工，指定加工速度
                var feed = Modbus.get4XFloat(1098);
                if(feed == 0.0)
                    feed = 3000.0;
                var strstart = "G55 F" + feed.toFixed(2);
                Linuxcnc.postMDI(strstart);
            }
            runstate = 2;
        }

        for(;runteachindex < teachpoint.length; runteachindex++)
        {
            if(Modbus.get0XIO(28))//是否镜向加工，如果打开了镜向，则逆数组加工
                ret = Linuxcnc.postMDI(teachpoint[teachpoint.length - runteachindex -
1]);//反向运行
            else
                ret = Linuxcnc.postMDI(teachpoint[runteachindex]);
            if(ret <= 0)
                break;
        }
        if(runteachindex == teachpoint.length){
            ret = Linuxcnc.postMDI("%");//加工到最后一行后，通知本加工结束。添加
到命令缓冲区中;
            makenum++;
            if(ret > 0)
                runteachindex++;
        }
        else{
            ret = Linuxcnc.getParams(14,0);//检查是否加工结束，所有加工已经完成。
            if(ret == 1){//停止加工，返回手动模式；
                stopwork();
                Linuxcnc.setDIO(702,0);//停止传送带；
                Linuxcnc.setDIO(701,0);
                Linuxcnc.setDIO(703,1);
                runstate = 1;
            }
        }
    }

    function enter_passwd(pwdindex,pwd)
```

```
{//检查密码，如果输入的密码与保存的相当则返回真，否则为非；
    tmpcmd = Modbus.get4XInt(500);
    if(pwdindex == 1){//一期密码验证；
        if(tmpcmd == 31){
            var pwd1 = Modbus.get4XUInt(410);
            if(pwd1 != pwd){
                Modbus.set4XUShort(405,1);
                return false;
            }
            else{
                Modbus.set4XUShort(405,0);
                Modbus.set4XUShort(636,1);
                return true;
            }
        }
    }
    else if(pwdindex == 2){//二期密码验证；
        if(tmpcmd == 32){
            var pwd1 = Modbus.get4XUInt(416);
            if(pwd1 != pwd){
                Modbus.set4XUShort(405,1);
                return false;
            }
            else{
                Modbus.set4XUShort(405,0);
                Modbus.set4XUShort(636,2);
                return true;
            }
        }
    }
    else if(pwdindex == 3){//三期密码验证；
        if(tmpcmd == 33){
            var pwd1 = Modbus.get4XUInt(420);
            if(pwd1 != pwd){
                Modbus.set4XUShort(405,1);
                return false;
            }
            else{
                Modbus.set4XUShort(405,0);
                Modbus.set4XUShort(636,3);
                return true;
            }
        }
    }
```

```
        return false;
    }

    function check_reg_time()
    {//比较当前时间，如果超过了指定密码时间，则指示用户输入密码，如果密码正确，则清除本期密码标志，允许加工继续。
        var cdate = new Date();
        var curt = cdate.getFullYear() * 10000 + (cdate.getMonth() + 1) * 100 + cdate.getDate();
        var pdflag1 = Modbus.get4XUInt(380);
        var pdflag2 = Modbus.get4XUInt(386);
        var pdflag3 = Modbus.get4XUInt(390);
        var time1 =    Modbus.get4XUInt(350);
        var time2 = Modbus.get4XUInt(356);
        var time3 = Modbus.get4XUInt(360);
        if(pdflag1 == 0 && pdflag2 == 0 && pdflag3 == 0){
            Modbus.set4XUShort(636,0);
        }
        else if(pdflag1 != 0){
            Modbus.set4XUShort(636,1);
        }
        else if(pdflag2 != 0){
            Modbus.set4XUShort(636,2);
        }
        else if(pdflag3 != 0){
            Modbus.set4XUShort(636,3);
        }
        if(time1 != 0 && pdflag1 != 0 && curt > time1){
            Modbus.set4XUShort(405,2);
            Modbus.set4XUShort(589,23);
            while(1){
                if(enter_passwd(1,pdflag1))
                    break;//验证成功；
                else
                    mDelay(100);
            }
            Modbus.set4XUInt(380,0);//清除一期密码；
            Modbus.saveRegData();//保存数据；
        }
        else if(time2 != 0 && pdflag2 != 0 && curt > time2){
            Modbus.set4XUShort(405,3);
            Modbus.set4XUShort(589,23);
            while(1){
                if(enter_passwd(2,pdflag2))
```

```
                    break;//验证成功；
                else
                    mDelay(100);
            }
            Modbus.set4XUInt(386,0);//清除二期密码；
            Modbus.saveRegData();
        }
        else if(time3 != 0 && pdflag3 != 0 && curt > time3){
            Modbus.set4XUShort(405,4);
            Modbus.set4XUShort(589,23);
            while(1){
                if(enter_passwd(3,pdflag3))
                    break;//验证成功；
                else
                    mDelay(100);
            }
            Modbus.set4XUInt(390,0);//清除三期密码；
            Modbus.saveRegData();
        }
        else{
            Modbus.set4XUShort(405,11);
        }

        Modbus.set4XUShort(405,0);
        return 0;
    }
    function startwork()
    {//开始工件；
        Linuxcnc.setMode(2);//设置为自动模式；
        Linuxcnc.Run(0);//从第 0 行开始运行；
        runteachindex = 0;//指定从第一点加工，如果是逆向加工，则从数组最后一个
记录开始；
        bstartwork = true;
    }
    function stopwork()
    {//停止加工；
        Linuxcnc.Abort();//中止；
        Linuxcnc.setMode(1);//设置为手动模式；
        runteachindex = 0;
        bstartwork = false;
}
```

七、