实时 Javascript 编程指南

一、前言：

1. 此编程指南仅针对使用 Javascript 编写 Linuxcnc 实时模块部份，对于非实时 Javascript 编程不在此文档，如有需要了解非实时部份 Javascript 编程请参照其它文档。
2. 所谓实时 Javascript 程序，指运行与 Linuxcnc 的实时任务中的 Javascript 脚本程序，它在运行过程中经历了三个阶段，首先在实时任务启动时，会读取该脚本程序，并预编译成字节码；然后在实时时间片轮循过程中，将调用脚本中的时间片函数；最后在退出时，将释放资源。

二、编程说明

1. setPeriodFunc(<function>);

   设置实时时间片函数，用于轮循时调用。

   <function>:代表脚本中用户定义的函数名，此函数无输入输出参数；

2. setIOParams(bitInNum,bitOutNum,wordInNum,wordOutNum,floatInNum,floatOutNum, floatParamNum,wordParamNum);

   设置 Javascript 实时模块支持的输入输出端口数量：

   - bitInNum:IO 输入端口数量；
   - bitOutNum:IO 输出端口数量；
   - wordInNum:整型输入端口数量；
   - wordOutNum:整型输出端口数量；
   - floatInNum:浮点端口输入数量；
   - floatOutNum:浮点端口输出数量；
   - floatParamNum:浮点输入输出参数端口数量；
   - wordParamNum:整型输入输出参数端口数量；

3. setBitOut(index,bset);

   设置指定 IO 输出端口的状态:

   index：代表 IO 输出端口编号，最大值由 setIOParams 函数的 bitOutNum 参数指定，最小值为 0；

   bset：true 置位，false 复位；

4. setWordOut(index,nvalue);

   设置指定整型输出端口的数值：

   Index:代表整型输出端口编号，最大值由 setIOParams 函数的 wordOutNum 参数指定，最小值为 0；

   nvalue：为输出的整型值；

5. setFloatOut(index,fvalue);

   设置指定浮点型输出端口输出值；

   index：浮点型输出端口编号，最大值由 setIOParams 函数的 floatOutNum 参数指定，最小值为 0

   fvalue：浮点输出值；

6. bset = getBitIn(index);

获取指定 IO 输入端口的状态:
Index：指定的 IO 输入端口编号,最大值由 setIOParams 函数的 bitInNum 参数指定，最小值为 0；
return：IO 输入状态，true/false;

7. bset = getBitOut(index);
获取指定 IO 输出端口的状态;
Index：指定的 IO 输出端号号，最大值由 setIOParams 函数的 bitOutNum 参数指定，最小值为 0；
return：IO 输出端口状态，true/false;

8. nvalue = getWordIn(index);
获取指定整型输入端口的值，返回输入端口整型值；
index：指定整型输入端口编号，最大值由 setIOParams 函数的 wordInNum 参数指定，最小值为 0；
return：该整型输入端口的值；

9. fvalue = getFloatIn(index);
获取指定浮点输入端口的值，返回输入端口浮点值；
index：指定浮点输入端口编号，最大值由 setIOParams 函数的 floatInNum 参数指定，最小值为 0；
return：对应浮点输入端口的值；

10. setUParams(index,uvalue)
设置整型参数值；
Index: 指定整型输入输出参数的端口编号，最大值由 setIOParams 的 wordParamNum 参数指定；
Uvalue:设置的整型值，此值最小为 0，最大为 2147483647,第 0~30 位有效；

11. setFParams(index,fvalue)
设置浮点参数值；
index：指定浮点输入输出参数的端口编号，最大值由 setIOParams 的 floatParamNum 参数指定；
fvalue:浮点参数值；

12. nvalue = getUParams(index);
获取整型参数值；
index：指定整型输入输出参数的端口编号，最大值由 setIOParams 的 wordParamNum 参数指定；
return:对应整型参数值，此值最小为 0，最大为 2147483647,第 0~30 位有效；

13. fvalue = getFParams(index);
获取浮点参数值；
index：指定浮点输入输出参数的端口编号，最大值由 setIOParams 的 floatParamNum

参数指定；
return：对应浮点参数的值；

14. setJoint(index ,fvmax,famax,fperiod,nout,fscale);
    设置指定轴的运动参数以及输入输出端口数量；
    - index:轴编号，0~2，此轴编号并不等同于物理轴编号，需通过配置文件连接才能对应；
    - fvmax:最大速度，浮点值；
    - famax:最大加速度，浮点值；
    - fperiod:时间片函数的调用间隔；
    - nout:使用哪个输出端口做为本轴的脉冲输出，使用浮点输出端口，
    - fscale:脉冲当量，单位为 mm/脉冲,仅针对 1/2 运动类型有效；

15. clearHome(index);
    清除该轴的坐标为零，作为 home 原点,清除前先确保轴处于静止状态;
    index:指定轴；

16. getJointPos(index);
    获取指定轴的坐标值；
    index：指定轴的编号；
    return:该轴理论坐标值；

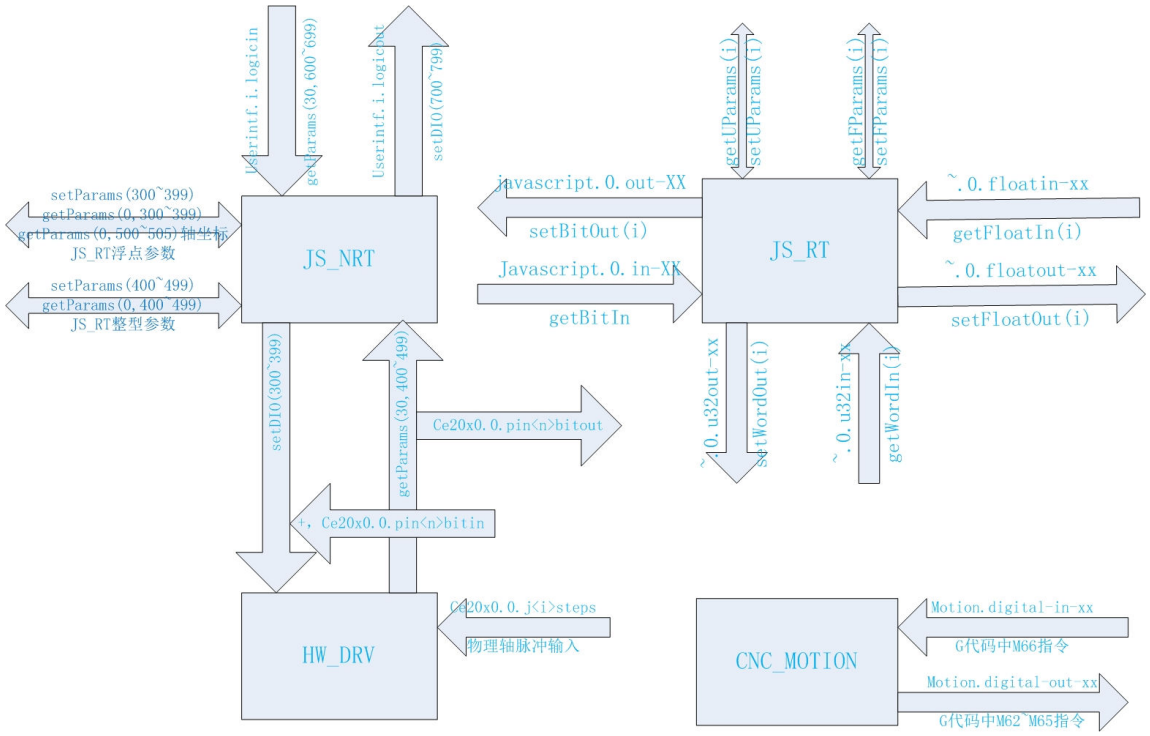
17. goJoint(index,fpos);//点位运动，指定轴运动到指定位置；将以当前轴的位置、速度、加速度为基础，规划至新的位置。
    index:轴编号；
    fpos:该轴运动到指定绝对位置；
    返回值：BOOL 型，true 运动到位，FALSE 没有到位；

18. jogCont(index,vel);
    连续运动指定轴，调用此函数不会影响轴的坐标输出；
    index:轴编号；
    vel:运动速度，为负表示反方向运动；

19. jogStop(index);
    停止指定轴，调用此函数不会影响轴的坐标输出；
    index:轴编号；
    返回值：BOOL 型，true 停止到位，FALSE 没有到位；

20. jogAlign(index);
    使指定轴坐标值与脉冲当量值匹配，长时间运行后，有可能坐标值太大，因此需要清零，但又不能产生累计误差,需要执行此操作。
    Index:指定轴编号，0~2；

三、实时任务与 Javascrpt 非实时任务的通讯与协调



如上图所示：

- 其中用函数表示的，由 Javascript 脚本中调用；
- 采用形于 xxx.x.xxx 表示的是信号名称，可能通过配置文件使之和其它信号相连接，如：net urep0 userintf.1.logout => javascript.0.in-01;
- 以上四个模块信号，可以任意连接，产生不通的通讯流程；
- 在 G 代码中，可以使用 M66 等待某信号发生，具体参考《LinuxCNC_Documentation.pdf》；
- 在 G 代码中，可以使用 M62~M65 触发某信号，具体参考《LinuxCNC_Documentation.pdf》；

四、示例程序

/****

IO 使用情况：

输入：

挤奶油轴(0)：

IO 输入端口：P01 轴 0 控制,奶油轴运动；

IO 输入端口：P03,奶油轴反转/正转;

传送带(1)：

IO 输入端口: P02 通知传送带开始运动；

MARK1:P04 高速检测光电传感器；

MARK2:P05 定位传感器；

IO 输出：

挤奶油轴(0)：

IO 输出端口：P01//到位；

浮点输出端口：F01;//坐标；

传送带(1)：

IO 输出端口：P02;//到位；

整型输出端口：I01;//脉冲；

传送带运动到位输出：P04;

参数输入：

浮点参数：

FP00：传送带低速运动速度值；

FP01:传送带高速运动速度值；

FP02：传送带步进长度值；

FP03：奶油轴正转速度；

FP04：奶油转反转运动距离；

整型值：

UP00：每盘加工次数；

传送带运行方式：初始化时，传送带处于未就位状态，收到 P02 有效信号后，此时首先应该查找 MARK1 点后降速接近 MARK2 点最终停止下来，进入就位状态；

然后开始奶油加工，加工完一排后，加工盘中下一排，此时传送带步进一定长；盘中加工完后，传送带处于未就位状态；

奶油轴的运动方式：初始时，此轴停止，收到 P01 有效信号且 P03 无效信号后，开始正转挤奶

油进入正常工作状态；收到 P03 有效信号后，停止正转，并反转一定距离；
反转到位后，处于封油状态；当 P03 变无效后，开始正转挤奶油，进入正常工作状态；当 P01 变无效后，停止正转，并反转一定距离，到位后进入初始状态；
****/

```
setIOParams(10,10,10,10,10,10,5,1);
setJoint(0,150.0,800.0,0.001,1,0.001);//脉冲输出；
setJoint(1,100.0,800.0,0.001,2,0.001);//脉冲输出；
var makecnt = 0;
var axisstat1 = 0;
var axisstat0 = 0;
var axis1destpos = 0.0;//用于增量运动时记忆目的坐标；
var axis0destpos = 0.0;//用于增量运动时记忆目的坐标；

function axisbelt()
{
    //传送带控制；
    if(axisstat1 == 0){//停止状态，未就位；
        if(getBitIn(2)){//收到通知，开始运动；
            setBitOut(4,false);
            if(getBitIn(5)){//处于定位点；
                makecnt = 0;
                axisstat1 = 4;
            }
            else if( getBitIn(4) ){//低速运动
                axisstat1 = 2;
                var dv = getFParams(0);
                jogCont(1,dv);
            }
            else{//高速运动；
                var dv = getFParams(1);
                jogCont(1,dv);
                axisstat1 = 1;
            }
        }
    }
    else if(axisstat1 == 1){//高速空行；
        if( getBitIn(4) ){//低速运动
            var dv = getFParams(0);
            jogCont(1,dv);
            axisstat1 = 2;
        }else{
            var dv = getFParams(1);
            jogCont(1,dv);
        }
```

```
}
else if(axisstat1 == 2){//低速定位；
    if(getBitIn(5)){//检测到停止信号；
        if(jogStop(1)){//停止到位；
            axisstat1 = 4;
            makecnt = 0;
        }
        else{
            axisstat1 = 6;
        }
    }else{
        var dv = getFParams(0);
        jogCont(1,dv);
    }
}
else if( axisstat1 == 6){//停止中；
    if(jogStop(1)){//停止到位；
        axisstat1 = 4;
        makecnt = 0;
    }
}
else if(axisstat1 == 3){//步进；
    if(goJoint(1,axis1destpos)){//到位；
        makecnt++;
        if(makecnt >= getUParams(0))//如果加工完一盘，则重新找盘；
            axisstat1 = 0;
        else
            axisstat1 = 4;
    }
}
else if(axisstat1 == 4){//就位状态；
    setBitOut(4,true);//通知运动控制部份，传送带已就位；
    if(!getBitIn(2)){//保证中间触发信号无效；
        axisstat1 = 5;
    }
}
else if(axisstat1 == 5){//等待加工完成；
    if(getBitIn(2)){//如果加工完成，则步进定长；
        axisstat1 = 3;
        axis1destpos = getFParams(1) + getJointPos(1);
        goJoint(1,axis1destpos);
        setBitOut(4,false);
    }
}
```

```
}

function axiscake()
{
    //挤奶油轴；
    if(axisstat0 == 0){//静止中；
        if(getBitIn(1) &&
            !getBitIn(3) ){//进入正转状态；
            var dv = getFParams(3);
            jogCont(0,dv);
            axisstat0 = 1;
        }
    }
    else if(axisstat0 == 1){//正转中；
        if( getBitIn(3) || !getBitIn(1) ){
            jogStop(0);//进入停止状态；
            axisstat0 = 2;
        }else{
            var dv = getFParams(3);
            jogCont(0,dv);
        }
    }
    else if( axisstat0 == 2){//停止中；
        if(jogStop(0)){//停止到位；
            if( getBitIn(3) ){//进入反转状态；
                axis0destpos = getFParams(4) + getJointPos(0);
                goJoint(0,axis0destpos);
                axisstat0 = 3;
            }else if(getBitIn(1)){
                var dv = getFParams(3);
                jogCont(0,dv);
                axisstat0 = 1;
            }else{
                axisstat0 = 0;
            }
        }
    }else if(axisstat0 == 3){//反转状态；
        goJoint(0,axis0destpos);
        if(!getBitIn(3))
            axisstat0 = 1;
    }
}

function axisswing()
```

```
{
	if(axisstat0 == 0){
		if(getBitIn(1)){
			clearHome(0);
			axisstat0 = 1;
		}
	}
	else if(axisstat0 == 1){//正转；
		if(goJoint(0,1000.1234)){
			axisstat0 = 2;
		}
	}
	else if(axisstat0 == 2){
		if(goJoint(0,-1000.1234)){
			axisstat0 = 3;
		}
	}
	else if(axisstat0 == 3){
		makecnt++;
		if(makecnt > 20000){
			axisstat0 = 1;
			makecnt = 0;
		}
	}
}

function task_slice()
{
	//axisswing();
	axisbelt();
	axiscake();
}

setPeriodFunc(task_slice);
```