

前面的文章我们介绍了带运行计时的电机控制函数块（[西门子SCL编程实例——带运行计时的电机控制](#)）。今天这篇文章，我们来介绍下如何实现多个电机自动调度（超时自动关闭并调度下一台电机）。
注：本文默认电机连接是使用断路器、接触器实现的直接启动电路，对于变频器类的控制需要做一些配置才能使用。



控制要求如下：

- 可设置每个电机的额定运行时间，当电机的实际运行时间大于额定运行时间时，则自动关闭并调度下一台可用电机；
- 当电机出现故障（比如断路器跳闸或超时无反馈），则电机状态变为不可用。不可用电机不参加调度；
- 当故障消除并复位后，电机状态重新变为可用，可重新参加调度；

具体如下：

打开博途开发环境，新建项目，添加用户自定义数据类型：typeMotorSchedule，如下图所示：

typeMotorSchedule					
	名称	数据类型	默认值	设定值	注释
1	▼ Command	Struct		☐	——命令——
2	start	Bool	false	☐	启动命令
3	stop	Bool	false	☐	停止命令
4	▼ Para	Struct		☐	——参数——
5	feedbackTime	Time	T# 2s	☐	反馈监视时间
6	normalRunTime	DInt	200	☐	设定运行时间，单位为秒
7	▼ ExtraSignal	Struct		☐	——其它信号——
8	fuse	Bool	false	☐	断路器，1=OK
9	feedbackFuse	Bool	false	☐	断路器反馈
10	▼ State	Struct		☐	——状态——
11	Q	Bool	false	☐	输出
12	error	Bool	false	☐	错误位
13	release	Bool	false	☐	1=可用，0=不可用
14	selected	Bool	false	☐	1=被调度选中，0=未选中
15	status	Word	16#0	☐	状态字
16	actualRunningTime	UDInt	0	☐	实际运行时间，单位为秒

其中：

- Command：包括启动（start）和停止（stop），用来启动或停止电机；
- Para：电机控制参数，包括反馈时间和额定运行时间，单位，秒；
- ExtraSignal：外部信号，包括断路器和接触器反馈；
- State：状态，包括：输出位（Q）、错误位（error）、释放（release，表示电机是否可用）、选中（selected，表示电机是否被选中）、状态（status，电机状态，非0等于出错）、实际运行时间（actualRunningTime，单位，秒）；

添加函数块，命名为：FB5017_MotorScheduler，声明变量如下图所示：

FB5017_MotorScheduler							
	名称	数据类型	默认值	保持	设定值	监控	注释
1	▼ Input				☐		
2	Init	Bool	false	非保持	☐		初始化调度程序。上升沿信号有效
3	Enable	Bool	false	非保持	☐		1=使能调度程序
4	Reset	Bool	false	非保持	☐		
5	GlobalTime	UDInt	0	非保持	☐		全局时间信号。单位：秒
6	NormalRunTime	DInt	200	非保持	☐		额定运行时间。单位：秒
7	▼ Output				☐		
8	Error	Bool	false	非保持	☐		1=有错误。0=无错误
9	State	DInt	0	在 IDB 中...	☐		状态
10	indexSelected	DInt	0	非保持	☐		被选中电机的序号
11	▼ InOut				☐		
12	▶ motors	Array[*] of *typeMotorSched...			☐		电机数组
13	▼ Static				☐		
14	▶ MotorControlWithRuntime	*FB5007_MotorControlWithR...			☒		电机控制块
15	statInitRisingEdge	Bool	false	非保持	☐		初始化上升沿信号
16	statInitRisingEdgeHF	Bool	false	非保持	☐		初始化上升沿信号辅助变量
17	resetRisingEdge	Bool	false	非保持	☒		
18	resetRisingEdgeHF	Bool	false	非保持	☐		
19	statIndex	DInt	0	非保持	☐		
20	statReSelection	Bool	false	非保持	☐		1=需要重新选择电机
21	statInitalized	Bool	false	非保持	☐		
22	statState	DInt	0	在 IDB 中...	☐		
23	▼ statStateSub AT *statState*	Array[0..1] of Int		在 IDB 中...	☐		
24	statStateSub[0]	Int		在 IDB 中...	☐		
25	statStateSub[1]	Int		在 IDB 中...	☐		
26	▼ Temp				☐		
27	tmpUBound	DInt			☐		数组上限
28	tmpLBound	DInt			☐		数组下限
29	i	DInt			☐		循环变量
30	▼ Constant				☐		
31	<新增>				☐		

函数块代码比较长，我将其分成多个区域：

- 功能说明；
- 初始化（initialize）；
- 复位（reset）；
- 调度（sheduler）；

具体如下：

功能说明：

```

1 曰(*)
2  Copyrights @Founderchip
3  =====
4  功能说明:
5      该程序实现电机, 依赖于用户自定义类型typeMotorSchedule。
6      可设定额定运行时间, 当某台电机的实际运行时间大于设定时间时,
7      会自动关闭当前电机并启动下一个电机。
8      当电机出现功能故障时, 比如断路器跳闸, 或没有反馈信号时,
9      调度程序会跳过该电机, 自动选择可调度的电机。
10     在进行电机调度之前, 应先进行初始化 (Init) 设置,
11     然后使能 (Enable) 调度程序。
12 输入参数:
13     Init           : 初始化调度程序, 上升沿信号有效
14     Enable        : 1=使能调度程序
15     Reset         : 复位信号
16     GlobalTime     : 全局时间信号, 单位, 秒
17     NormalRunTime  : 额定运行时间, 单位, 秒
18 输出参数:
19     Error          : True=调查程序发现错误
20     State          : 电机状态 (DInt)  Int1 + Int0
21                     int1: 错误类型
22                     int0: 故障电机编号 (索引)
23     indexSelected  : 被选中的电机编号
24 输入/输出参数:
25     motors         : 电机数组 (typeMotorSchedule)
26
27 作者:
28     北岛李工
29     2022-3-4
30  -----
31 修改日志:
32
33 2023-3-4 v1.0 版本(首发)          北岛李工
34
35  -----

```

初始化:

```

38 //数组上限
39 #tmpUBound := UPPER_BOUND (ARR := #motors, DIM := 1);
40 //数组下限
41 #tmpLBound := LOWER_BOUND (ARR := #motors, DIM := 1);
42 REGION Initialize
43     //初始化上升沿信号
44     #statInitRisingEdge := #Init AND NOT #statInitRisingEdgeHF;
45     #statInitRisingEdgeHF := #Init;
46     IF #statInitRisingEdge THEN
47         FOR #i := #tmpLBound TO #tmpUBound DO
48             #motors[#i].State.selected := FALSE;
49             #motors[#i].State.release := TRUE;
50             #motors[#i].State.Q := FALSE;
51             #motors[#i].State.error := FALSE;
52             #motors[#i].State.status := 0;
53             #motors[#i].State.actualRunningTime := 0;
54             #motors[#i].Para.normalRunTime := #NormalRunTime;
55             //默认从最小序号电机开始
56             #statIndex := #tmpLBound;
57         END_FOR;
58         #statInitalized := TRUE;
59     END_IF;
60 END_REGION

```

复位:

```

61 REGION reset
62     //复位
63     //上升沿信号
64     #resetRisingEdge := #Reset AND NOT #resetRisingEdgeHF;
65     #resetRisingEdgeHF := #Reset;
66     IF #resetRisingEdge THEN
67         FOR #i := #tmpLBound TO #tmpUBound DO
68             IF #motors[#i].State.error THEN
69                 #motors[#i].State.error := FALSE;
70                 #motors[#i].State.status := 0;
71                 #motors[#i].State.actualRunningTime := 0;
72                 #motors[#i].State.release := TRUE;
73             END_IF;
74         END_FOR;
75         #statState:=0;
76     END_IF;
77 END_REGION

```

调度:

```

78 IF NOT #statInitalized THEN
79     #statStateSub[0] := 1; //错误代码
80 END_IF;
81 //使能调度器
82 IF #Enable THEN
83     REGION scheduler
84         //重新选择电机
85         IF #statReSelection THEN
86             #motors[#statIndex].Command.start := FALSE;
87             #motors[#statIndex].Command.stop := FALSE;
88             #motors[#statIndex].State.Q := FALSE;
89             #motors[#statIndex].State.actualRunningTime := 0;
90             #statIndex += 1;
91             IF #statIndex > #tmpUBound THEN
92                 #statIndex := #tmpLBound;
93             END_IF;
94             //复位静态变量
95             #MotorControlWithRuntime.Q := FALSE;
96             #MotorControlWithRuntime.Error := FALSE;
97             #MotorControlWithRuntime.Diagnose := 0;
98             #MotorControlWithRuntime.Runtime := 0;
99             #MotorControlWithRuntime.statStartPosEdge := FALSE;
100             #MotorControlWithRuntime.statStopPosEdge := FALSE;
101             #statReSelection := FALSE;
102         END_IF;
103
104         IF #motors[#statIndex].State.release THEN
105             IF #motors[#statIndex].State.Q THEN //电机已经启动
106                 //比较运行时间与设定时间
107                 IF #motors[#statIndex].State.actualRunningTime >
108                     #motors[#statIndex].Para.normalRunTime THEN
109                     #motors[#statIndex].Command.stop := TRUE;
110                     #motors[#statIndex].Command.start := FALSE;
111                     #motors[#statIndex].State.selected := FALSE;
112                     //需要重新选择电机
113                     #statReSelection := TRUE;
114                 END_IF;
115             ELSE
116                 #motors[#statIndex].Command.start := TRUE; //启动
117                 #motors[#statIndex].Command.stop := FALSE;
118                 #motors[#statIndex].State.selected := TRUE;
119             END_IF;
120         ELSE
121             #motors[#statIndex].State.selected := FALSE;
122             //需要重新选择电机
123             #statReSelection := TRUE;
124         END_IF;
125     END_REGION

```

```

126  REGION motorControl
127  //电机控制
128  IF #motors[#statIndex].State.selected THEN
129      // 调用电机控制函数块
130      #MotorControlWithRuntime(Fuse := #motors[#statIndex].ExtraSignal.fuse,
131                               Start := #motors[#statIndex].Command.start,
132                               Stop := #motors[#statIndex].Command.stop,
133                               FeedbackSignal := #motors[#statIndex].ExtraSignal.feedbackFuse,
134                               GlobalTime := #GlobalTime,
135                               Q => #motors[#statIndex].State.Q,
136                               Error => #motors[#statIndex].State.error,
137                               Diagnose => #motors[#statIndex].State.status,
138                               Runtime := #motors[#statIndex].State.actualRunningTime
139      );
140      //复位命令
141      #motors[#statIndex].Command.start := FALSE;
142      #motors[#statIndex].Command.stop := FALSE;
143      //如果出错，则该电机不释放(not release)
144      IF #motors[#statIndex].State.error THEN
145          #motors[#statIndex].State.release := FALSE;
146          #statStateSub[0] := 3; //故障类别
147          #statStateSub[1] := DINT_TO_INT(#statIndex); //故障电机编号
148      END_IF;
149      //
150      END_IF;
151  END_REGION
152  #indexSelected := #statIndex;
153 ELSE
154  IF #statInitalized THEN
155      #motors[#statIndex].State.Q := FALSE;
156  END_IF;
157  END_IF;
158  //电机控制状态
159  IF #statState > 0 THEN
160      #Error := TRUE;
161  ELSE
162      #Error := FALSE;
163  END_IF;
164  //输出
165  #State := #statState;

```

使用该函数块时需要先初始化，然后使能。当出现错误时，要进行复位。

可以在全局数据块中创建电机数组（typeMotorSchedule），然后将断路器、接触器的信号写入电机数组的ExtraSignal中，并将电机状态State.Q输出到硬件的输出通道中。

我创建了测试函数块及全局数据块对电机调度程序进行了测试，包括断路器的错误测试，结果是OK的。

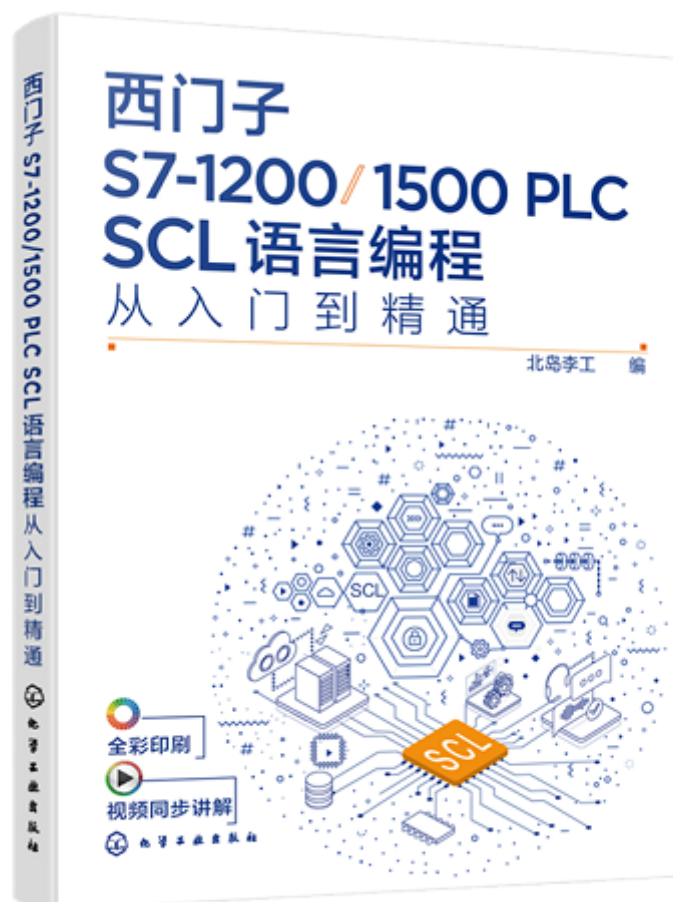
motorScheduler ▸ PLC_2 [CPU 1515-2 PN] ▸ 程序块 ▸ DB200_Global [DB200]						
保持实际值 快照 将快照值复制到起始值中 将起始值加载为实际值						
DB200_Global						
	名称	数据类型	起始值	快照	监视值	保持
1	Static					☐
2	globalSecond	UDInt	0	—	4080	☐
3	actualRuntime	UDInt	0	—	0	☐
4	indexSelected	DInt	0	—	2	☐
5	errorSchedule	Bool	false	—	FALSE	☐
6	stateSchedule	DInt	0	—	0	☐
7	motors	Array[0..4] of "type..."				☐
8	motors[0]	"typeMotorSchedule"				☐
9	motors[1]	"typeMotorSchedule"				☐
10	motors[2]	"typeMotorSchedule"				☐
11	Command	Struct				☐
12	Para	Struct				☐
13	ExtraSignal	Struct				☐
14	fuse	Bool	false	—	TRUE	☐
15	feedbackFuse	Bool	false	—	FALSE	☐
16	State	Struct				☐
17	Q	Bool	false	—	TRUE	☐
18	error	Bool	false	—	FALSE	☐
19	release	Bool	false	—	TRUE	☐
20	selected	Bool	false	—	TRUE	☐
21	status	Word	16#0	—	16#0000	☐
22	actualRunni...	UDInt	0	—	26	☐
23	motors[3]	"typeMotorSchedule"				☐
24	Command	Struct				☐

如果你有任何问题，欢迎留言。

下面是西门子SCL编程的文章归档链接：

《《 西门子SCL编程入门到精通文章归档 》》

我的书《西门子S7-1200/1500 PLC SCL语言编程 ——从入门到精通》从硬件到软件，比较详细的介绍了SCL语言的编程，感兴趣的话可以扫描下面的二维码查看：



识别图中小
程序码购买