使用数据日志的示例程序

简介

在以下示例程序中，创建一个数据日志，将一条或多条记录写入该数据日志，关闭和打开该数据日志，并基于该数据日志创建新文件。在该示例中，还可以删除数据日志。该示例中还可以编辑最多八个数据日志。

有关各个指令的详细信息，请使用相应的链接打开相应的帮助说明。

使用数据记录的一般注意事项

- 执行“DataLogCreate”和“DataLogNew”指令时，将自动打开所创建的数据记录。
- CPU 从 RUN 切换至 STOP，或者 CPU 重新启动后，将自动关闭数据记录。
- 要执行“DataLogWrite”指令，必须打开数据记录。
- 对于 S7-1200 CPU，最多可同时打开八个数据记录。对于 S7-1500 CPU，最多可同时打开十个数据记录。

要求

在 CPU 属性中进行以下设置：

- 在“PROFINET 接口 > 访问 Web 服务器”(PROFINET interface > Access to the web server) 中，激活 Web 服务器访问。
- 在“Web 服务器”(Web server) 条目中，启用该模块的 Web 服务器。
- 在“用户管理”(User administration) 部分，创建一个具有文件读写和删除权限的新用户。

数据的存储

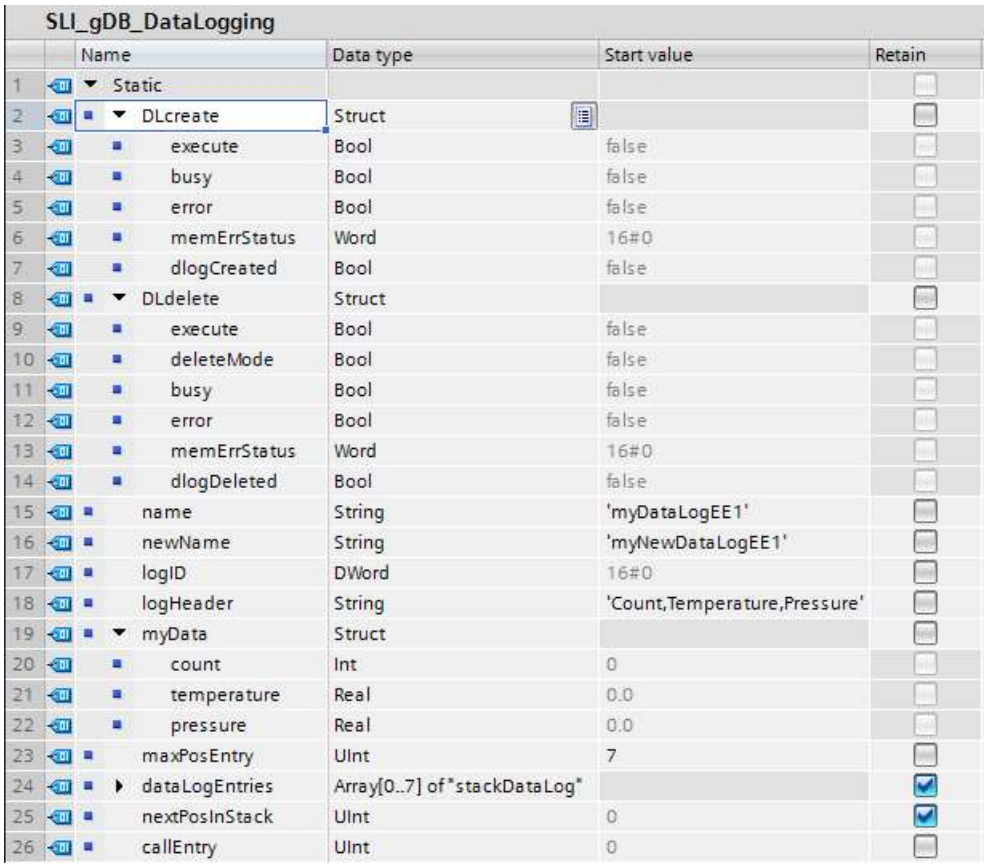
调用和编辑某个数据日志时，需要该数据日志的名称和 ID。为数据日志条目创建存储器，以防这些数据丢失。可通过 PLC 数据类型“stackDataLog”定义数据日志条目的结构。

stackDataLog

	Name	Data type	Default value
1	name	String	''
2	ID	DWord	16#0
3	DLclosed	Bool	false

创建以下变量，在全局数据块 (SLI_gDB_Datalogging) 中进行数据存储。这其中还有用于数据日志条目的存储器和用于控制存储器的变量。

注：如果还想要从 CPU 的装载存储器中物理删除数据日志，请使用值为“TRUE”的“DLdelete.deleteMode”变量。



SLI_gDB_DataLogging

	Name	Data type	Start value	Retain
1	Static			
2	DLcreate	Struct		
3	execute	Bool	false	
4	busy	Bool	false	
5	error	Bool	false	
6	memErrStatus	Word	16#0	
7	dlogCreated	Bool	false	
8	DLdelete	Struct		
9	execute	Bool	false	
10	deleteMode	Bool	false	
11	busy	Bool	false	
12	error	Bool	false	
13	memErrStatus	Word	16#0	
14	dlogDeleted	Bool	false	
15	name	String	'myDataLogEE1'	
16	newName	String	'myNewDataLogEE1'	
17	logID	DWord	16#0	
18	logHeader	String	'Count,Temperature,Pressure'	
19	myData	Struct		
20	count	Int	0	
21	temperature	Real	0.0	
22	pressure	Real	0.0	
23	maxPosEntry	UInt	7	
24	dataLogEntries	Array[0..7] of "stackDataLog"		
25	nextPosInStack	UInt	0	
26	callEntry	UInt	0	

数据块（“SLI_gDB_Datalogging”）用于创建 (DataLogCreate) 或删除 (DataLogDelete) 一个数据日志。将“myData”结构的 3 个条目用作过程值：count、temerature 和 pressure。在数据块中，暂时存储这三个值，然后使用“DataLogWrite”指令将其作为数据记录传送到数据记录。

数据记录在以下条目之后组成：

- 数据记录号（自动分配）
- 日期（当对 DataLogCreate 使用“1”时在 TIMESTAMP 参数中自动分配）。
- 时间（当对 DataLogCreate 使用“1”时在 TIMESTAMP 参数中自动分配）。
- 结构“myData”中“count”的当前值。
- 结构“myData”中“temperature”的当前值。
- 结构“myData”中“pressure”的当前值。

使用全局数据块“SLI_gDB_DataLogW”，为写入数据记录（DataLogWrite：）提供过程值。

SLI_gDB_DataLogW

	Name	Data type	Start value	Retain
1	Static			
2	DLwrite	Struct		
3	execute	Bool	false	
4	preset	Struct		
5	count	Int	5	
6	scaledTemp	Real	25.7	
7	scaledPressure	Real	45.0	
8	busy	Bool	false	
9	error	Bool	false	
10	status	Word	16#0	
11	memErrStatus	Word	16#0	
12	memDone	Bool	false	
13	DLclose	Struct		
14	execute	Bool	false	
15	busy	Bool	false	
16	error	Bool	false	
17	status	Word	16#0	
18	memDone	Bool	false	
19	DLopen	Struct		
20	execute	Bool	false	
21	busy	Bool	false	
22	error	Bool	false	
23	status	Word	16#0	
24	memDone	Bool	false	
25	DLnewfile	Struct		
26	execute	Bool	false	
27	busy	Bool	false	
28	error	Bool	false	
29	status	Word	16#0	
30	newFileCreated	Bool	false	

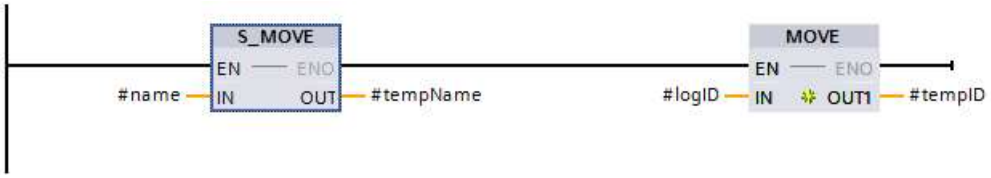
函数“SLI_FC_saveEntry_DataLog”：参数互连

使用函数“SLI_FC_saveEntry_DataLog”保存数据日志的名称和 ID。在其中创建以下局部变量。

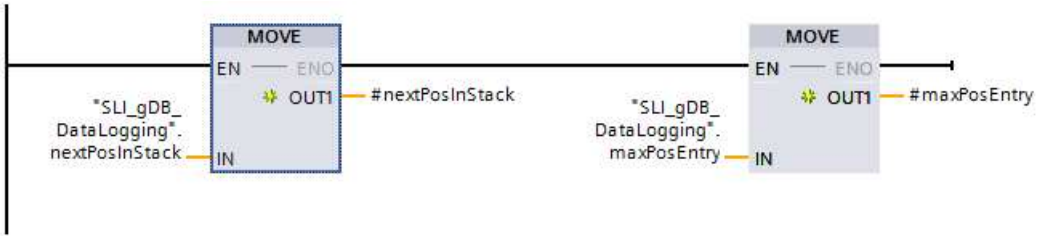
SLI_FC_saveEntry_DataLog

	Name	Data type
1	Input	
2	name	String
3	logID	DWord
4	Output	
5	<Add new>	
6	InOut	
7	<Add new>	
8	Temp	
9	nextPosInStack	UInt
10	maxPosEntry	UInt
11	tempName	String
12	tempID	DWord

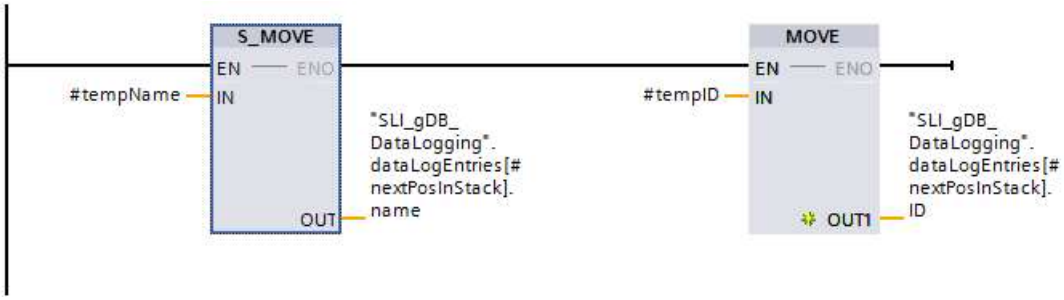
程序段 1：通过输入参数“#name”和“#logID”来传输变量“name”和“ logID”的值。



程序段 2：“nextPosInstack”变量指定了“dataLogEntries”数组中要用于数据日志条目的存储位置。“maxPosEntry”变量指定了“dataLogEntries”数组中最后一个单元格。按如下步骤将变量“nextPosInstack”与“maxPosEntry”的值互连，用于进一步处理。

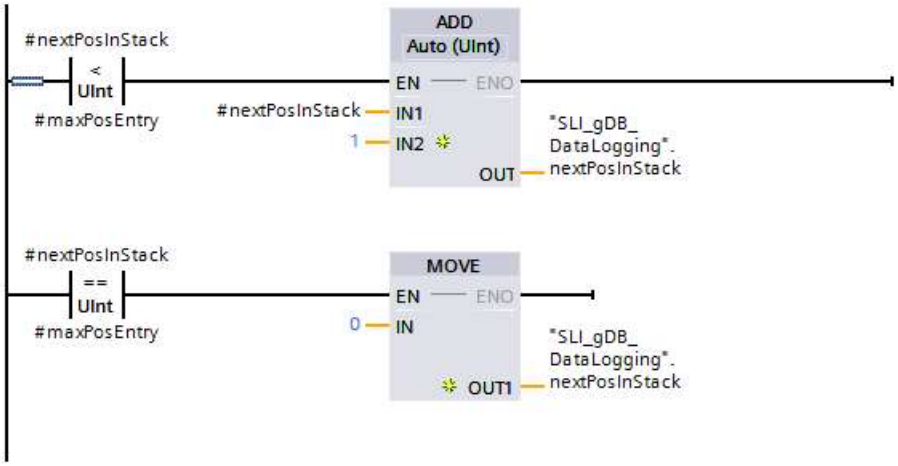


程序段 3：按以下方式将名称和 ID 存储在“dataLogEntries”数组中。



程序段 4：为确保将新的数据日志存储在存储器的不同条目中，请递增变量“nextPosInStack”。

注：当达到存储限制（“#maxPosEntry”）时，“nextPosInStack”变量不再递增，而是复位为值“0”。这表示新数据日志的数据会覆盖旧数据日志中的数据。



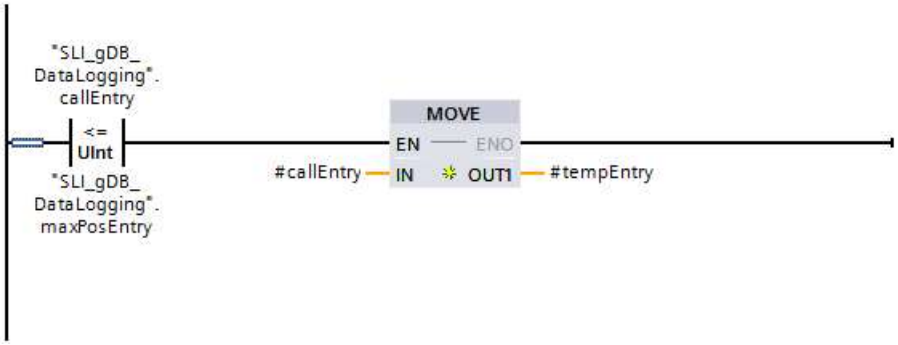
函数“SLI_FC_callEntry_DataLog”：参数互连

要调用选定数据日志条目的名称和 ID，请创建函数“SLI_FC_callEntry_DataLog”。在其中创建以下局部变量。通过“callEntry”变量指定选择的数据日志条目。

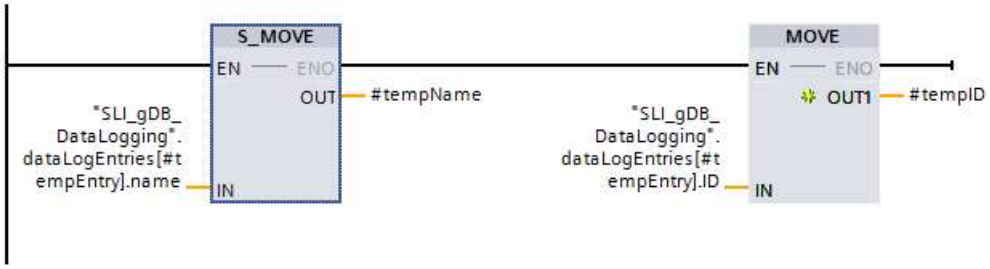
SLI_FC_callEntry_DataLog

		Name	Data type
1		Input	
2		callEntry	UInt
3		Output	
4		name	String
5		logID	DWord
6		InOut	
7		<Add new>	
8		Temp	
9		tempEntry	UInt
10		tempName	String
11		tempID	DWord

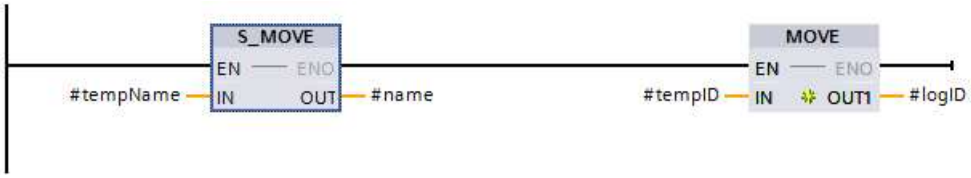
程序段 1：按如下步骤将变量“callEntry”的值互连，用于进一步处理。



程序段 2：按如下方式调用已创建数据日志的名称和 ID。



程序段 3：通过输出参数“#name”和“#logID”来传输变量“dataLogEntries.name”和“dataLogEntries.ID”的值。



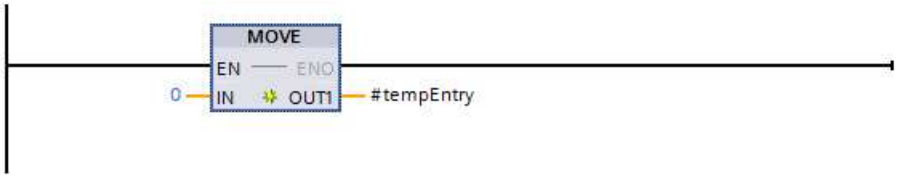
函数“SLI_FC_deleteEntry_DataLog”：参数互连

要删除数据日志条目中的数据，请使用函数“SLI_FC_deleteEntry_DataLog”。在其中创建以下局部变量。

SLI_FC_deleteEntry_DataLog

		Name	Data type	Default value
1		Input		
2		deleteEntry	UInt	
3		Output		
4		<Add new>		
5		InOut		
6		<Add new>		
7		Temp		
8		tempEntry	UInt	
9		Constant		
10		EMPTYSTRING	String	''

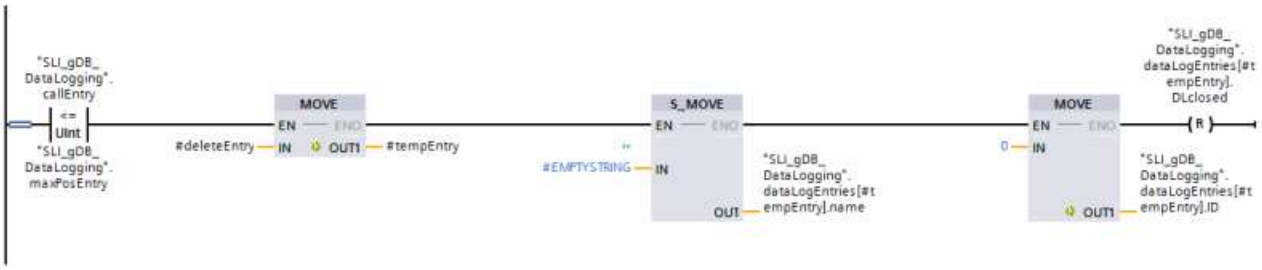
程序段 1：随后确保临时变量“#tempEntry”有值。



程序段 2：如果变量“callEntry”的值不大于变量“maxPosEntry”的值，则会发生以下情况：

- 变量“callEntry”的值将传输到输入参数 #deleteEntry。通过变量“callEntry”选择要删除的数据日志条目。
- 将会复位选定数据日志条目的变量“dataLogEntries.name”、“dataLogEntries.ID”和“dataLogEntries.DLclosed”。

这些互连呈现如下形式。



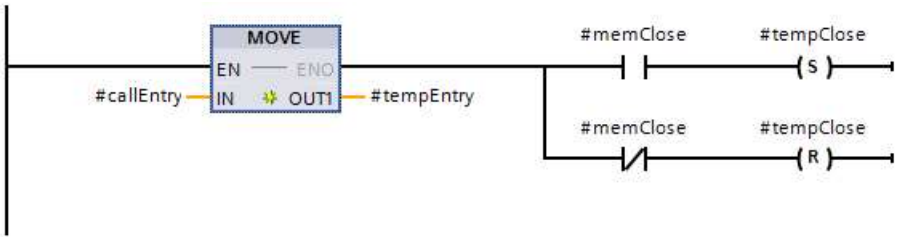
函数“SLI_FC_memCloseEntry_DataLog”：参数互连

要保存数据日志的状态（打开或关闭），请使用函数“SLI_FC_memCloseEntry_DataLog”。在其中创建以下局部变量。

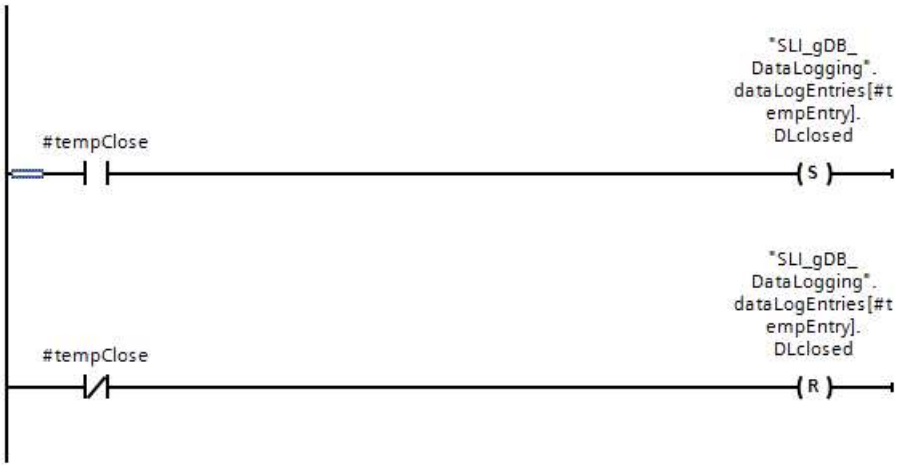
SLI_FC_memCloseEntry_DataLog

		Name	Data type
1		Input	
2		callEntry	UInt
3		memClose	Bool
4		Output	
5		<Add new>	
6		InOut	
7		<Add new>	
8		Temp	
9		tempEntry	UInt
10		tempClose	Bool

程序段 1：要选择数据日志条目，请传输输入参数“#callEntry”的值。此外，还需要通过输入参数“#mem-Close”传输数据日志的状态。



程序段 2：将数据日志的状态保存在适当的数据日志条目中。这些互连呈现如下形式。



函数块“SLI_FB_DataLogging”：参数互连

创建一个函数块“SLI_FB_DataLogging”作为示例程序的中央块。在该函数块中创建以下局部变量。

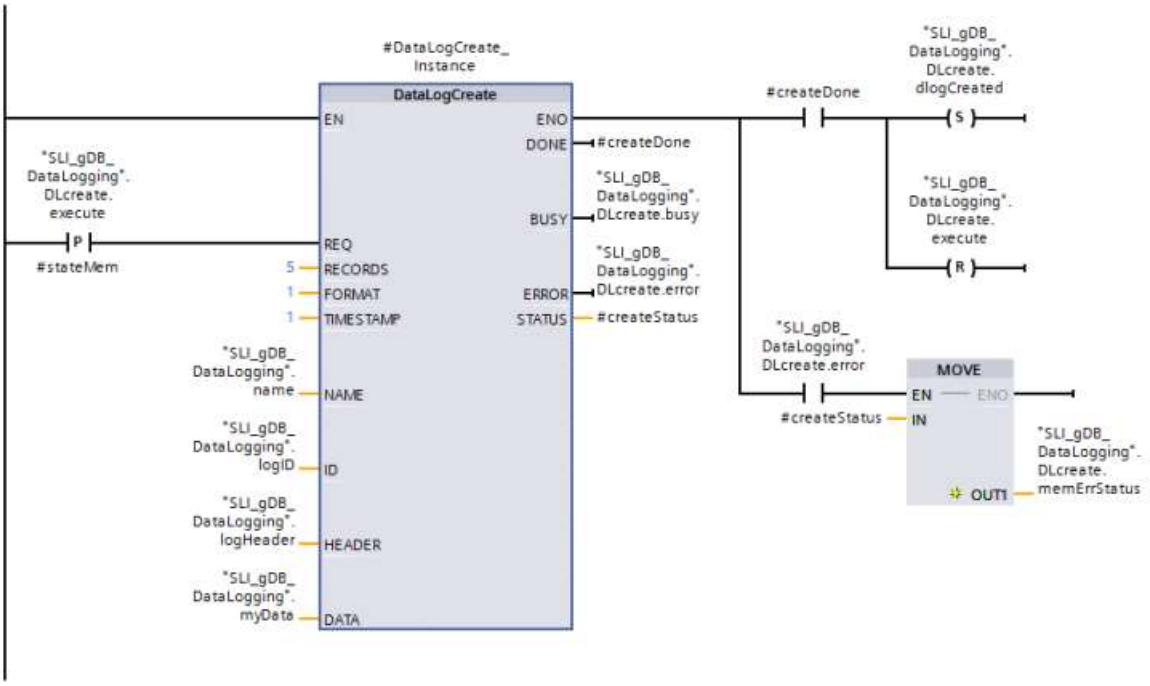
7		Static			
8		createDone	Bool	false	Non-retain
9		createStatus	Word	16#0	Non-retain
10		deleteDone	Bool	false	Non-retain
11		deleteStatus	Word	16#0	Non-retain
12		writeDone	Bool	false	Non-retain
13		executeClose	Bool	false	Non-retain
14		closeDone	Bool	false	Non-retain
15		openDone	Bool	false	Non-retain
16		memOpenDone	Bool	false	Non-retain
17		newFileDone	Bool	false	Non-retain
18		newFileStatus	Word	16#0	Non-retain
19		stateMem	Bool	false	Non-retain
20		stateMem2	Bool	false	Non-retain
21		stateMem3	Bool	false	Non-retain
22		stateMem4	Bool	false	Non-retain
23		stateMem5	Bool	false	Non-retain
24		stateMem6	Bool	false	Non-retain
25		DataLogCreate_Instan..	DataLogCreate		
26		DataLogWrite_Instance	DataLogWrite		
27		DataLogClose_Instance	DataLogClose		
28		DataLogOpen_Instance	DataLogOpen		
29		DataLogNewFile_Insta..	DataLogNewFile		
30		DataLogDelete_Instan..	DataLogDelete		
31		Temp			
32		<Add new>			
33		Constant			
34		EMPTYSTRING	String	''	

在该函数块中创建以下互连。在程序循环 OB (OB1) 中调用该函数块。

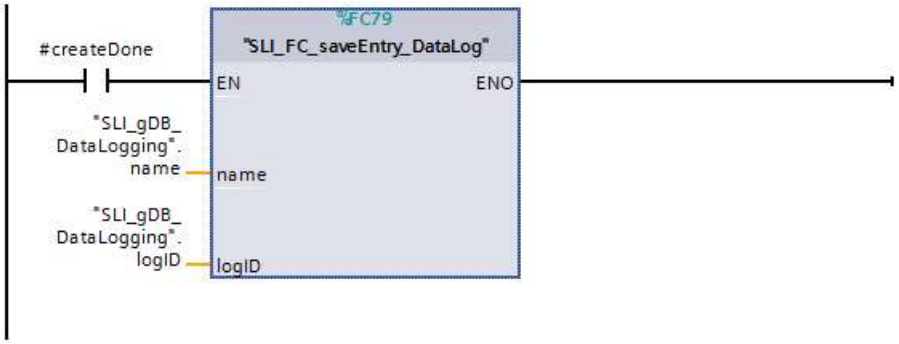
程序段 1：在指令 DataLogCreate 的输入参数 REQ（“DLcreate.execute”）的上升沿处，启动数据日志的创建。

DataLogCreate 的输出参数 DONE（“#createDone”）只适用于一个循环。因此，需将其值保存在变量“DLcreate.dlogCreated”中。该过程会将变量“DLcreate.execute”复位。

如果发生错误，则将状态（“DLcreate.status”）保存在“DLcreate.memErrStatus”变量中。

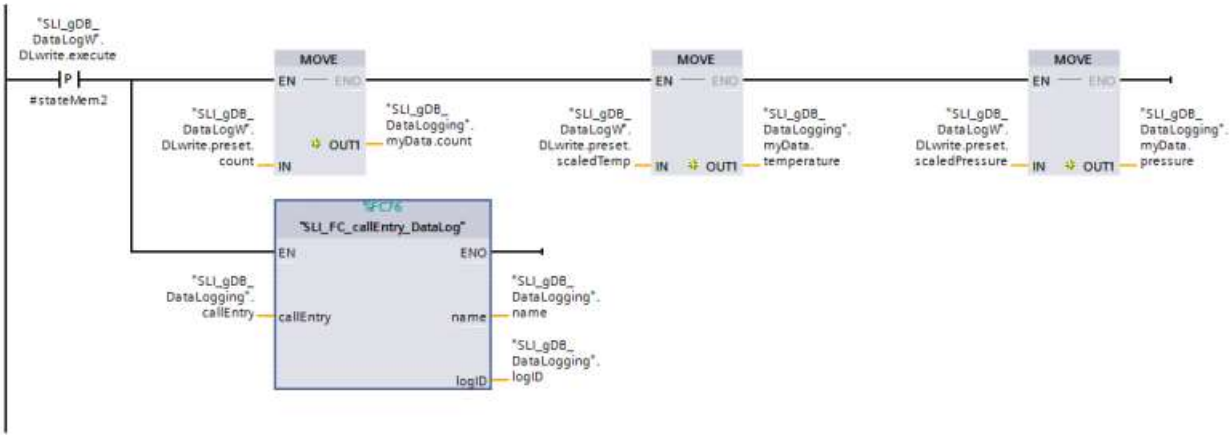


程序段 2：通过使用函数“SLI_FC_saveEntry_DataLog”保存数据日志的名称和 ID。如果参数 DONE（“#createDone”）的信号状态为“TRUE”，则执行存储操作。



程序段 3：在上升沿处，将触发将新过程值存储在 myData 结构中。这一步用于临时在数据块“SLI_gDB_DataLogging”中存储所需的过程值。

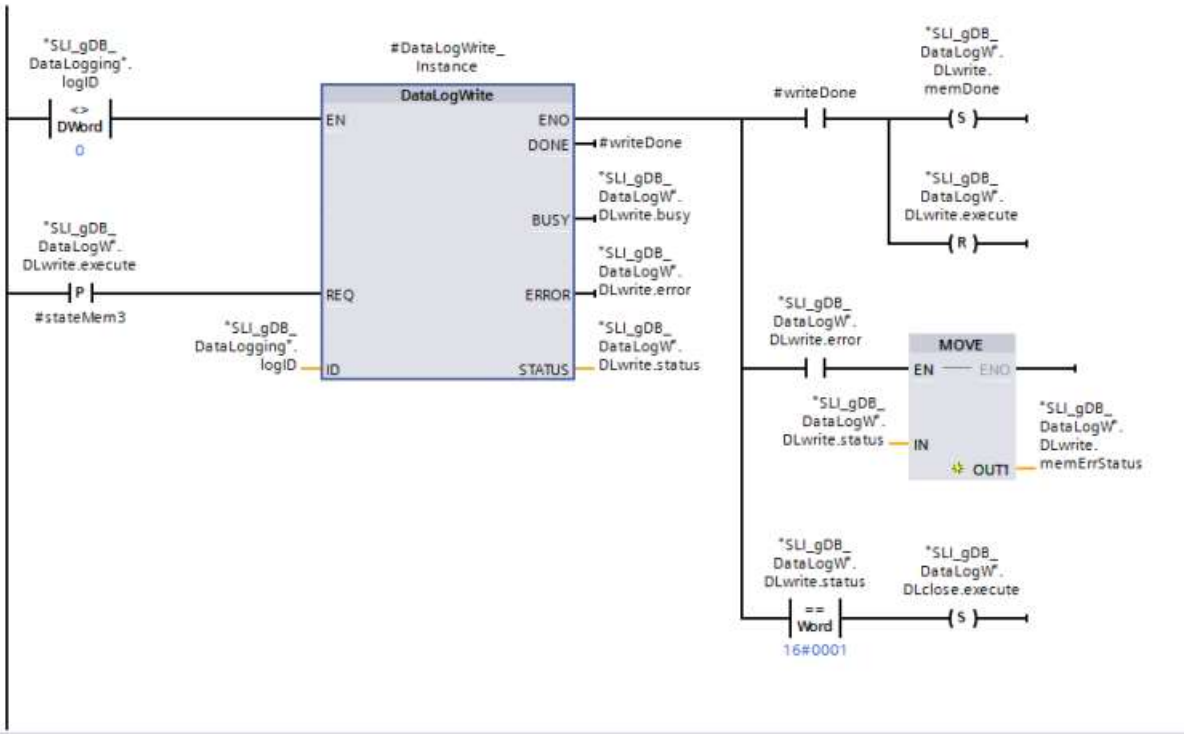
此外，还可使用函数“SLI_FC_callEntry_DataLog”调用所需的数据日志。通过“callEntry”变量，可根据“dataLogEntries”数组中的数据日志条目选择数据日志。



程序段 4：如果已创建一个数据日志，并且变量“logID”的值不为“0”，则置位 DataLogWrite 的输入 EN。执行查询是因为生成过程跨越了多个循环，且必须在完成后才能执行写操作。在输入参数 REQ（“DLwrite.execute”）的上升沿处，将触发写入数据记录。

DataLogWrite 的成功状态（“#writeDone”为“TRUE”）通过“DLwrite.memDone”变量进行存储。该过程会将变量“DLwrite.execute”复位。

如果发生错误，则将状态（“DLwrite.status”）保存在“DLwrite.memErrStatus”变量中。如果状态为“16#0001”，指令“DataLogWrite”会自动启动数据日志的关闭（“DLclose.execute”为“TRUE”）。



说明

数据日志中的数据记录数目

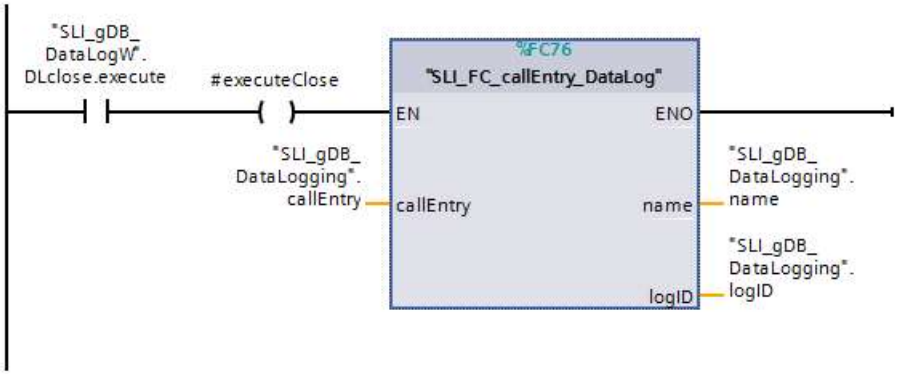
数据日志设计为包含 5 条数据记录（请参见“程序段 1”）。

这意味着：

- 在 5 条数据记录之后，使用指令 DataLogWrite，将值“0001”输出到参数 STATUS（“DLwrite.status”）中。

- 在文件末尾，创建最后一个可能的数据记录。如果创建了其它数据记录，则覆盖较早的数据记录。

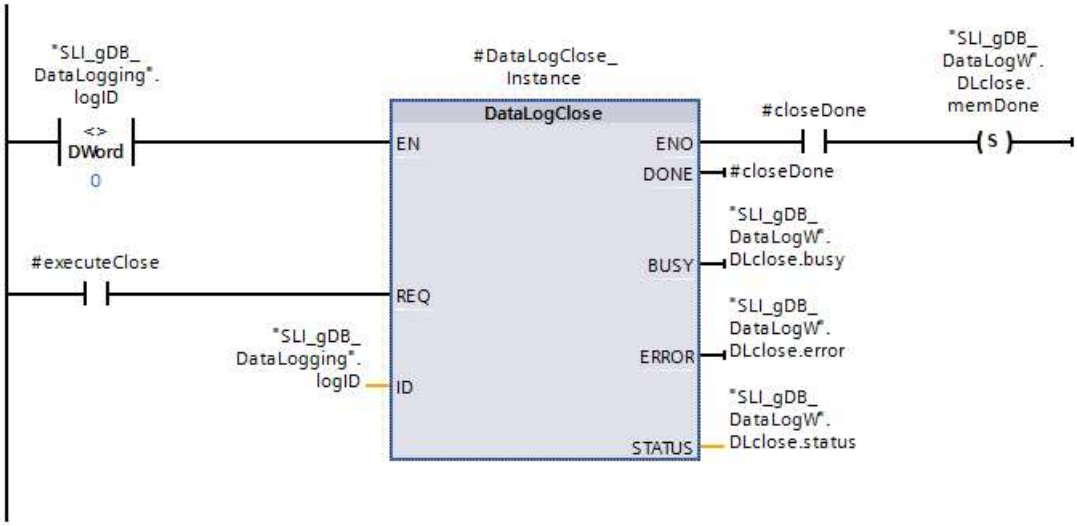
程序段 5：在关闭数据日志之前，请(通过“callEntry”)选择要关闭的数据日志。当常开触点(“DLclose.execute”) 的信号状态为“TRUE”时，根据“dataLogEntries”数组中的条目来调用所需的数据日志。此外，将“#executeClose”设置为“TRUE”。为此，请创建以下互连。



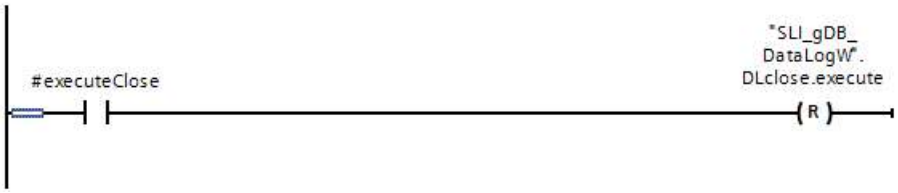
程序段 6：写入最后一个数据记录或者处理完数据日志之后关闭数据日志。

为此，将执行 DataLogClose 指令的 REQ (“#executeClose”) 输入置位。数据日志关闭后，无法再写入数据记录。

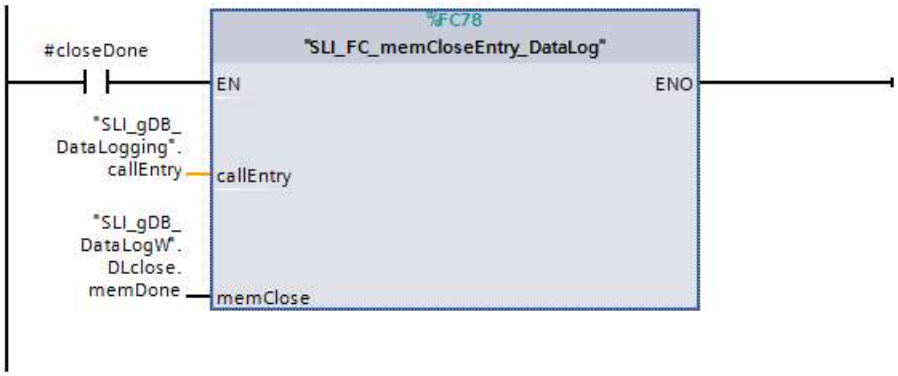
DataLogClose 的成功状态 (“#closeDone”为“TRUE”) 通过“DLclose.memDone”变量进行存储。



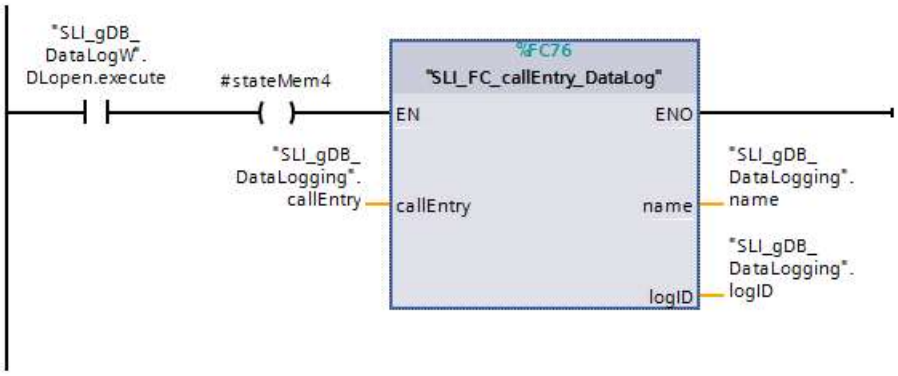
程序段 7：按以下步骤将“DLclose.execute”变量自动复位。



程序段 8：在以下互连的基础上，将数据日志的“关闭”状态 (“DLclose.memDone”为“TRUE”) 存储在数据日志条目中 (“dataLogEntries.DLclosed”为“TRUE”)。

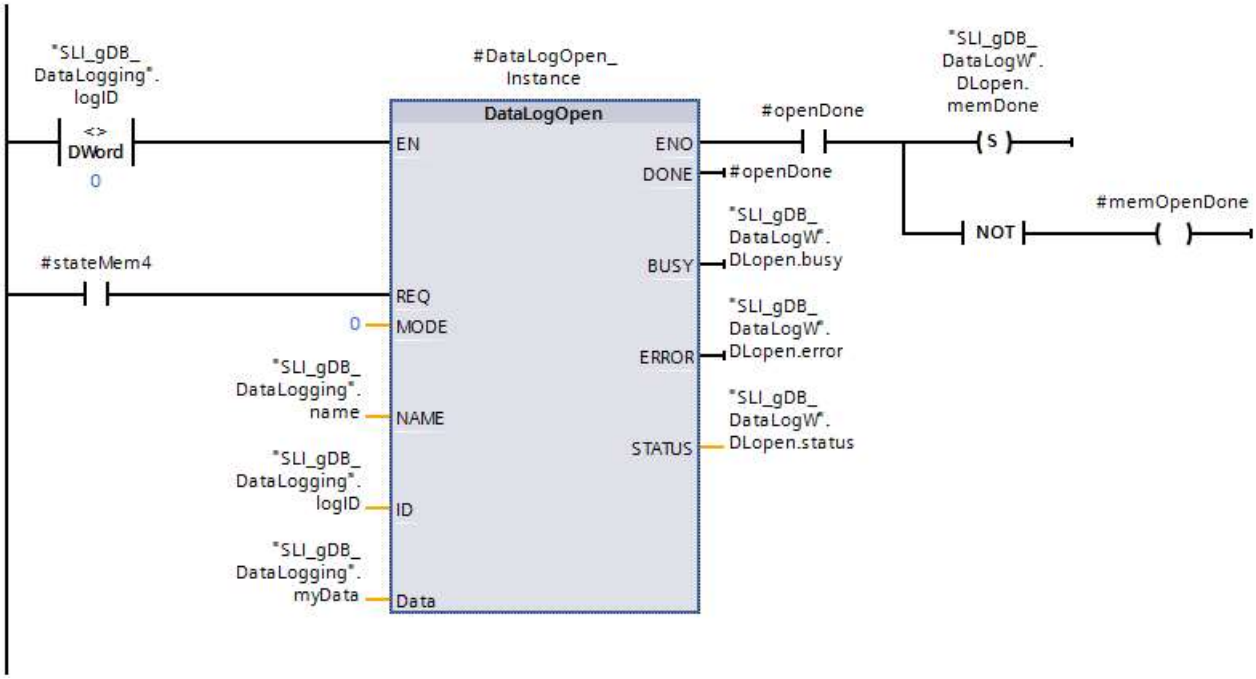


程序段 9：在打开数据日志之前，请（通过“callEntry”）选择要打开的数据日志。当常开触点（“DLopen.execute”）的信号状态为“TRUE”时，根据“dataLogEntries”数组中的条目来调用所需的数据日志。为此，请创建以下互连。

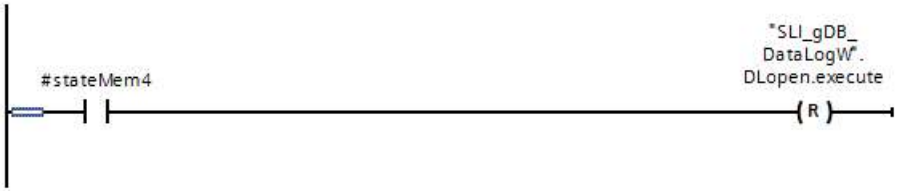


程序段 10：必须使用 DataLogOpen 指令再次打开数据日志以便稍后再写入数据记录。此后，如果使用 DataLogWrite 写入其它数据记录，则始终覆盖最早的数据记录。

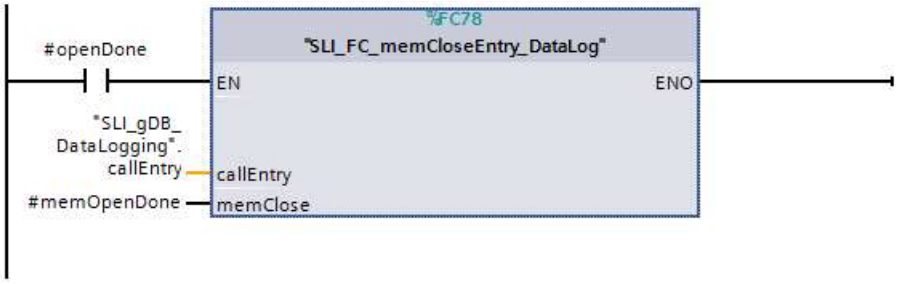
DataLogOpen 的成功状态（“#openDone”为“TRUE”）通过“DLopen.memDone”变量进行存储。“#openDone”的否定结果通过“#memOpenDone”变量进行保存。



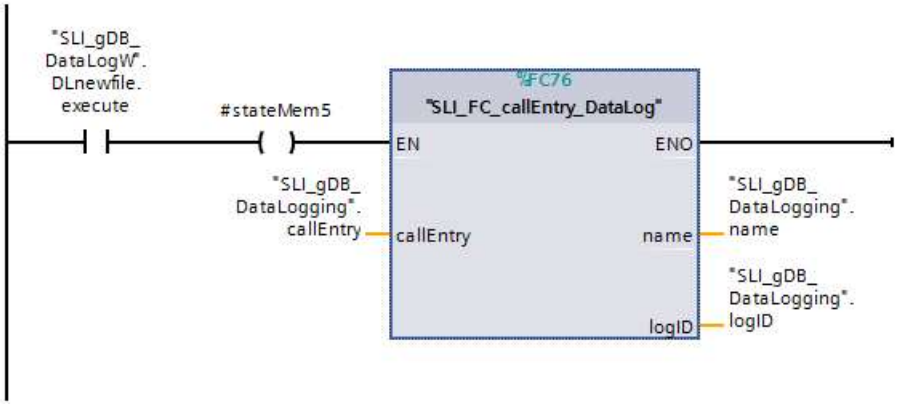
程序段 11：按以下步骤将“DLopen.execute”变量自动复位。



程序段 12：在以下互连的基础上，将数据日志的“打开”状态存储在数据日志条目中（“#memClose”或“dataLogEntries.DLclosed”为“FALSE”）。

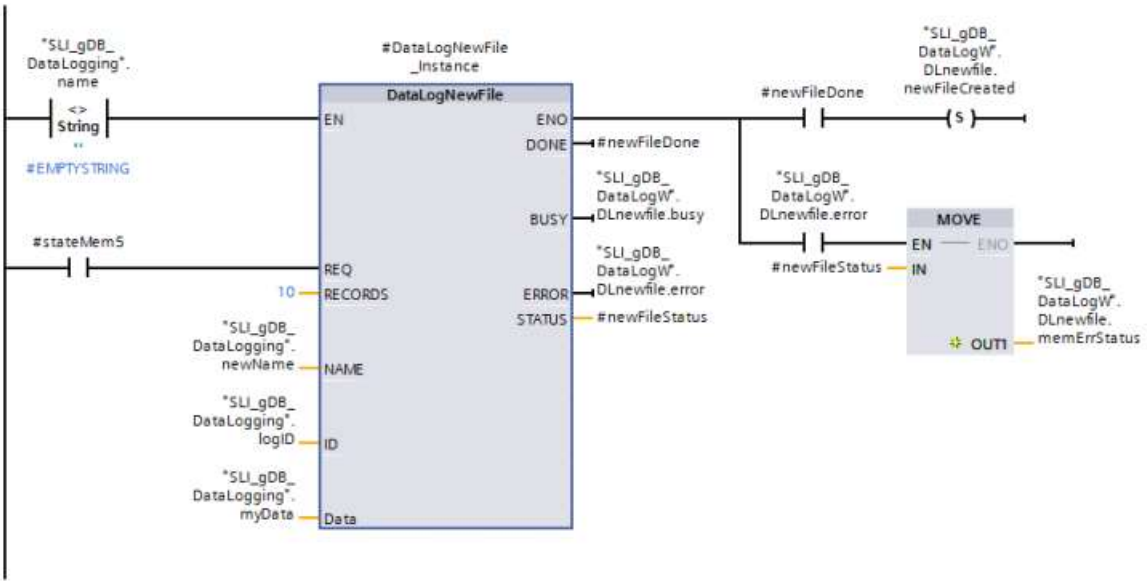


程序段 13：要在数据日志的基础上创建一个新文件，首先必须（通过“callEntry”）选择要使用的数据日志。当常开触点（“DLnewfile.execute”）的信号状态为“TRUE”时，根据“dataLogEntries”数组中的条目来调用所需的数据日志。为此，请创建以下互连。



程序段 14：如果不想覆盖较早的数据记录，可以使用“DataLogNewFile”指令创建具有相同结构的新数据记录。为此，要在指令的 ID 参数中输入您要复制其结构的现有数据记录的 ID。在“DataLogNewFile”指令执行完毕后，将为新数据记录分配唯一新 ID 值。

在输入参数 REQ（“#stateMem5”）的上升沿时，将触发创建过程。DataLogNewFile 的成功状态（“#newFileDone”为“TRUE”）通过“DLnewfile.memDone”变量进行存储。如果发生错误，则将状态（“#newFileStatus”）保存在“DLnewfile.memErrStatus”变量中。

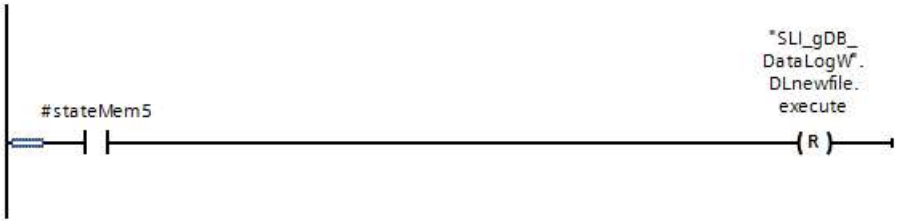


说明

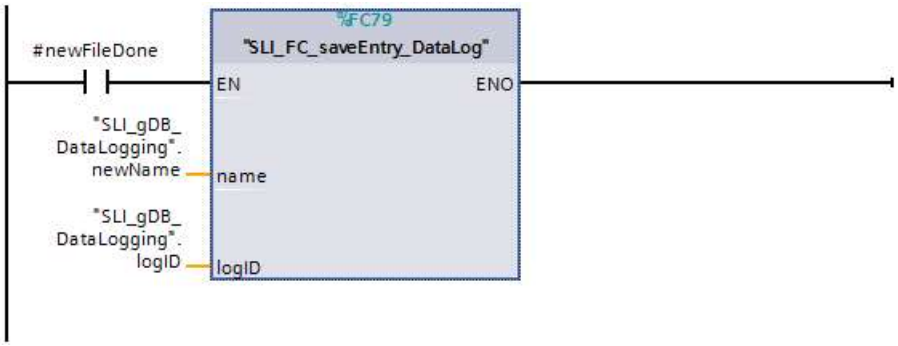
跨多个循环调用

请注意，DataLogNewFile 的调用也要跨多个循环。比如，通过对 DataLogWrite 指令的 BUSY 参数采样，可防止指令“DataLogNewFile”过早执行。

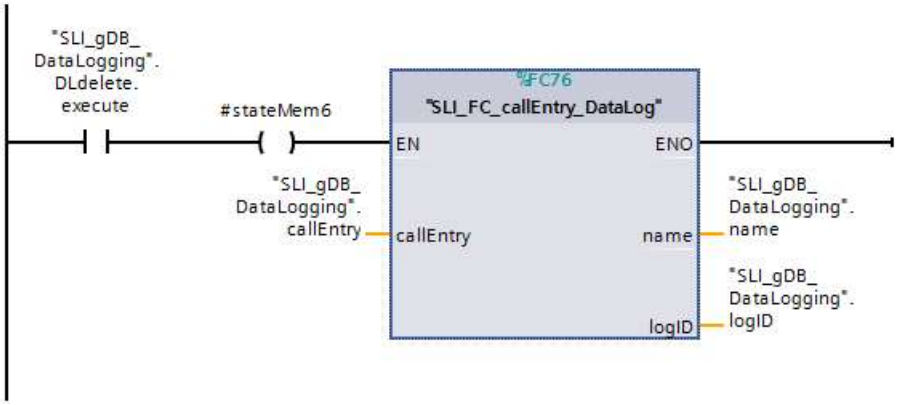
程序段 15：按以下步骤将“DLnewfile.execute”变量自动复位。



程序段 16：通过使用函数“SLI_FC_saveEntry_DataLog”保存数据日志的名称和 ID。如果参数 DONE（“#newFileDone”）的信号状态为“TRUE”，则执行存储操作。

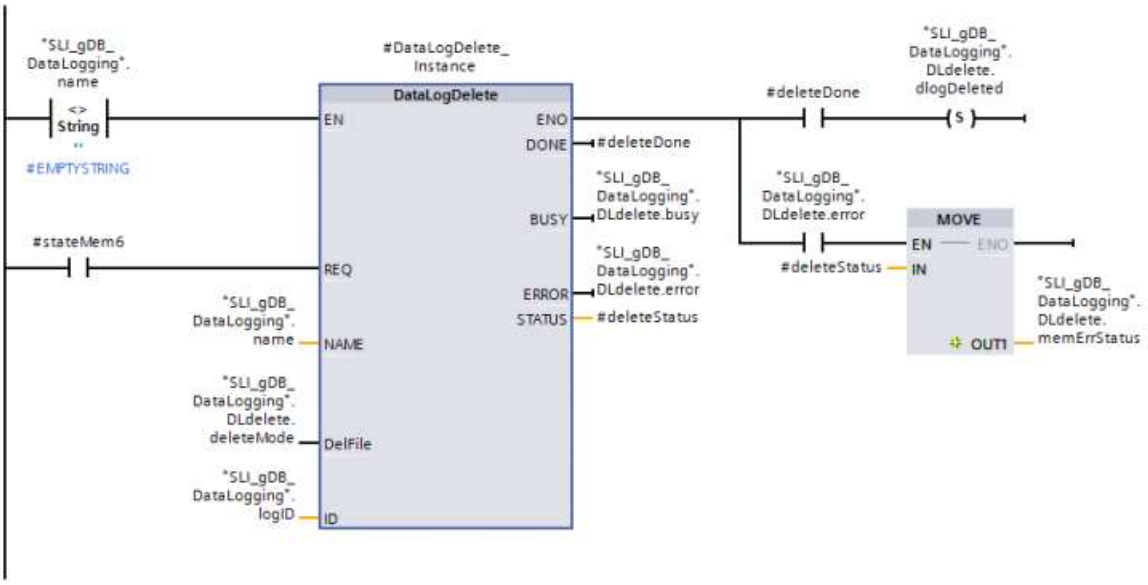


程序段 17：要删除数据日志，首先必须（通过“callEntry”）选择要删除的数据日志。当常开触点（“DLdelete.execute”）的信号状态为“TRUE”时，根据“dataLogEntries”数组中的条目来调用所需的数据日志。为此，请创建以下互连。

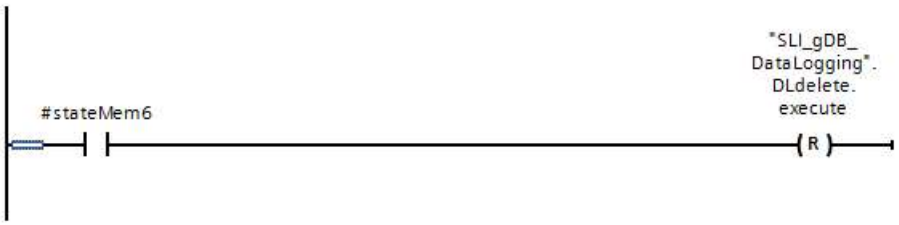


程序段 18：通过使用指令“DataLogDelete”删除不再需要的数据日志。

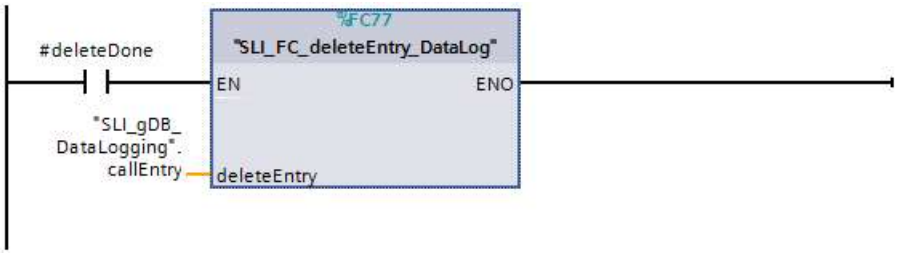
在输入参数 REQ（“#stateMem6”）的上升沿时，将触发数据日志的删除过程。DataLogDelete 的成功状态（DONE 为“TRUE”）通过“DLdelete.dlogDeleted”变量进行存储。如果发生错误，则将状态（“#deleteStatus”）保存在“DLdelete.memErrStatus”变量中。



程序段 19：按以下步骤将“DLdelete.execute”变量自动复位。



程序段 20：通过使用函数“SLI_FC_deleteEntry_DataLog”删除数据日志的条目。如果参数 DONE（“#deleteDone”）的信号状态为“TRUE”，则执行删除操作。



结果

通过 Web 服务器打开写入的数据

通过 Web 服务器，可显示该示例程序创建的数据日志。为此，使用 Internet 浏览器打开 Web 服务器并打开目录“\DataLogs”。

CSV 文件的内容

- 使用 DataLogCreate 指令创建数据记录时，数据记录的最大数量设置为“5”。如果不超过此数字，所有写入的数据记录都会包含在数据记录中。

	A	B	C	D	E	F
1	Record	Date	UTC Time	Count	Temperature	Pressure
2	1	9/30/2010	20:26:56	1	9.86E+01	3.52E+01
3	2	9/30/2010	20:28:43	2	1.00E+02	3.73E+01
4	3	9/30/2010	20:29:03	3	9.99E+01	3.68E+01
5	4	9/30/2010	20:29:21	4	9.95E+01	3.64E+01
6	5	9/30/2010	20:30:19	5	9.92E+01	3.74E+01
7						

- 如果再添加其它数据记录，最旧的数据记录 (Record 1) 将被覆盖。

	A	B	C	D	E	F
1	Record	Date	UTC Time	Count	Temperature	Pressure
2	6	9/30/2010	20:32:03	6	9.86E+01	3.58E+01
3	2	9/30/2010	20:28:43	2	1.00E+02	3.73E+01
4	3	9/30/2010	20:29:03	3	9.99E+01	3.68E+01
5	4	9/30/2010	20:29:21	4	9.95E+01	3.64E+01
6	5	9/30/2010	20:30:19	5	9.92E+01	3.74E+01
7						

程序代码

有关上述示例的更多信息和程序代码，请参见“**指令的示例库**”。