

Siemens
Intro to Structured Control Language (SCL)
in TIA Portal with S7-1200/1500

Kelly Anton | 10-24-2019

1

Agenda

- Brief Overview of SCL
- SCL Editor
- Create simple SCL block
- Debugging SCL
- Data types
- Program Control
- Math, Strings and Arrays
- Live Demos

2

SCL (Structured Control Language) - Defined

- High-level programming language based on PASCAL
- Text based language

Name	Data type	Default value	Comment
1	Input		
2	Output		
3	lowbound	Dint	
4	avgout	Real	
5	InOut		
6	array_to_be_sorted	Array["] of Real	
7	Temp		
8	size	Dint	
9	X	Dint	
10	Y	Dint	
11	tempX	Real	
12	tempY	Real	
13	mysum	Real	

```

1 #size := UPPER_BOUND(ARR := #array_to_be_sorted
2   , DIM := 1);
3
4 FOR #X := 1 TO #size DO
5   FOR #Y:= (#X +1) TO #size DO
6     IF #array_to_be_sorted[#X] > #array_to_be_sorted[#Y] THEN
7       #tempX := #array_to_be_sorted[#X];
8       #tempY := #array_to_be_sorted[#Y];
9       //flip
10      #array_to_be_sorted[#X] := #tempY;
11      #array_to_be_sorted[#Y] := #tempX;
12    END_IF;
13  END_FOR;
14 END_FOR;

```

3

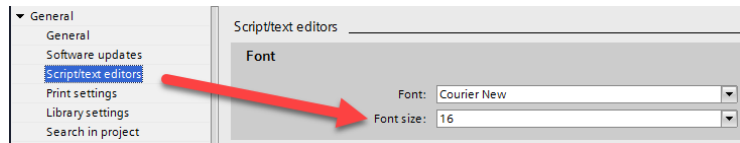
SCL – Common Uses

- Math
- String Operations
- Array Operations
- Communication
- Program Flow

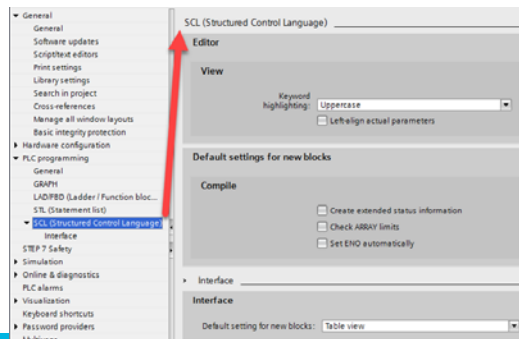
4

SCL - Options > Settings (Editor Settings)

- General > Script/text editors Font size



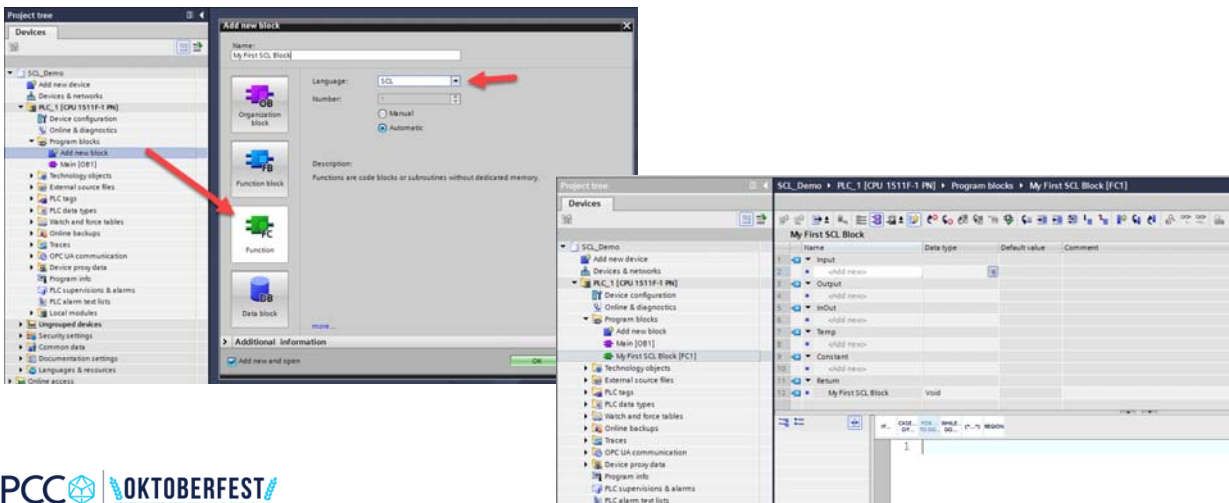
- PLC programming > SCL (Structured Control Language)



5

SCL - FC/FB Blocks

- Add New SCL FC/FB Block



6

SCL – FC/FB Block Parameters

- Create parameters in interface area just like LAD
- Step 7 5.x parameters in code. Easier in TIA.

The screenshot shows the 'My First SCL Return val' block configuration. The parameters are listed in a table below:

Name	Data type	Default value	Comment
Input			
MyInt1	Int		
MyInt2	Int		
Output			
<Add new>			
InOut			
<Add new>			
Temp			
MyTempResult	Int		
Constant			
<Add new>			
Return			
My First SCL Return val	Int		

Below the table, the SCL code is displayed:

```

1 #MyTempResult := #MyInt1 + #MyInt2;
2
3 //return a value to FC call
4
5 #"My First SCL Return val" := #MyTempResult;

```

7

SCL – FC/FB Block Parameters New Option with V15.1

- Interface area setting for Table view or Textual view
- Options > Settings > PLC Programming > SCL > Interface

The screenshot shows the SCL code for a function block:

```

1 FUNCTION "My_Text_InterfaceFC" : Void
2
3 VAR_INPUT
4   Num1 : Int;
5   Num2 : Real;
6 END_VAR
7
8 VAR_OUTPUT
9   Result : Real;
10 END_VAR
11
12 VAR_IN_OUT ... END_VAR
13
14
15
16 VAR_TEMP ... END_VAR
17
18
19
20 VAR CONSTANT ... END_VAR
21
22
23

```

Below the code, the 'Interface' settings are shown. The 'Default setting for new blocks' is set to 'Table view'.

The SCL code at the bottom of the screenshot is:

```

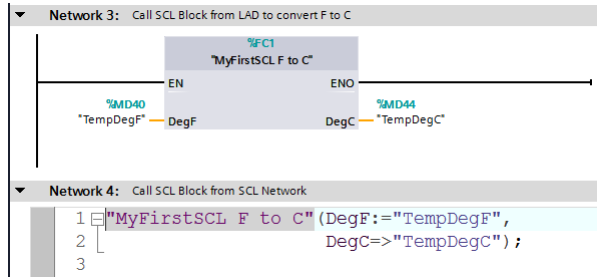
1 #Result := #Num1 + #Num2;

```

8

SCL - Calling FC's and FB's

- Call FC/FB from LAD
- Call FC from SCL
- Call FB from SCL



Network 7: Call My First SCL FB from SCL

```

1  [ ] "My_DB2" (Button:="Button_2", // Input
2  [   | FlashRate:="Clock_1Hz", // Input
3  [   | Light=>"Light_2"); // Output
4  [
5  [

```

Variable	Address
"My_DB2"	%DB2
"Button_2"	%I0.1
"Light_2"	%Q0.1

My_DB2 [DB2] Properties

General

Name: My_DB2
Type: DB
Language: DB
Number: 2
 Manual
 Automatic

Instance DB of My First SCL FB [FB1]

9

SCL – SCL Network Embedded in Ladder (LAD)

- Insert SCL Network in LAD...

Network 1: LAD Code

Network 2: SCL Code

```

1 //Embed SCL code in LAD block
2 "MyInt_Result" := "MyInt_1" + "MyInt_2";
3

```

Variable	Address	Value
"MyInt_Result"	%MW14	13
"MyInt_1"	%MW10	10
"MyInt_2"	%MW12	3

10

SCL - General Rules

- Instructions can span several lines
- Each instruction ends with a semicolon ;
- Not case sensitive

```

Network 4: Call SCL Block from SCL Network
1  "MyFirstSCL F to C" (DegF:="TempDegF",
2  DegC=>"TempDegC");

```

11

SCL - Comments

- Document your code
- Comments do not affect program execution

```

1  // Single line comment
2
3  (*
4     Comment a section is
5     for multiple lines of comments.
6
7  *)
8

```

12

SCL - Regions

- Used for readability of code

```

1 REGION Math Demos
2   REGION Integer Math Demo
3     // Nested region
4     "MyInt_Result" := "MyInt_1" + "MyInt_2";
5   END_REGION
6
7   REGION Real Math Demo
8     // Nested region
9     "MyReal_Result" := "MyReal_1" + 1.3;
10  END_REGION
11 END_REGION
12
13

```

13

SCL - Built-in functions

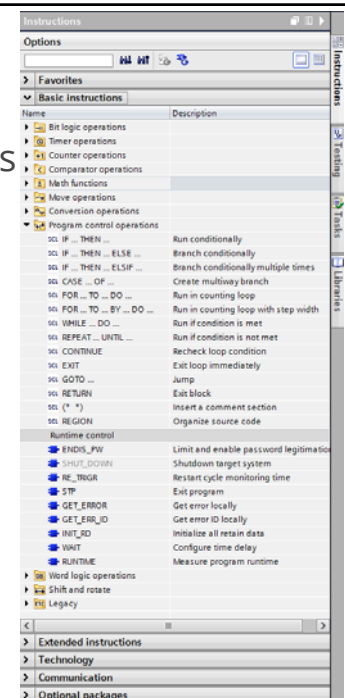
- Ctrl + Space brings up a list of functions & tags
- Use Instructions tab and drag into code
- Start typing ... to use intellisense

```

1 // Ctrl + Space bring up list of functions
2 // or start typing first few letters to filter
3

```

SCL	Keyword		
SCL_ABS_*			>>>
SCL_ACOS_*			>>>
SCL_AND			>>>
SCL_ASIN_*			>>>
SCL_ATAN_*			>>>
SCL_BCD16_TO_*			>>>
SCL_BCD32_TO_*			>>>
SCL_BOOL_TO_*			>>>



14

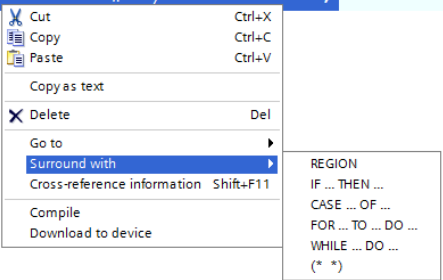
SCL - Surround with

- Quickly wrap selected lines of code with statement

```

1 // Use surround
2 // highlight line(s) right click for menu
3
4 #MyTempResult := #MyInt1 + 1;

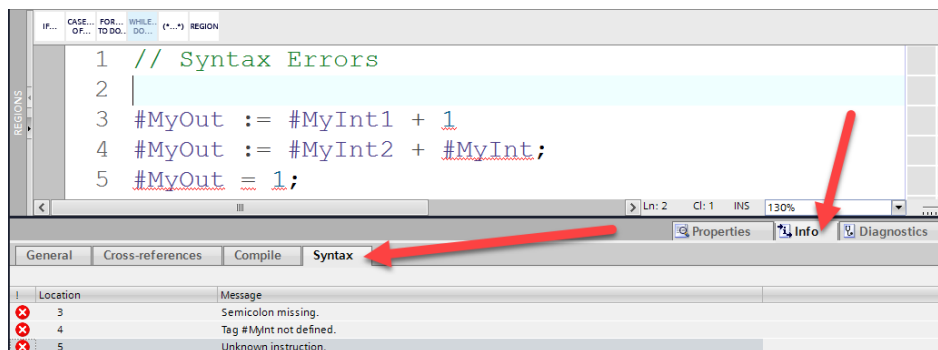
```



15

SCL - Debugging Syntax Errors

- Missing ; at end of statement
- Used = instead of := for assignment
- Start with first error because it could be causing other errors



```

1 // Syntax Errors
2 |
3 #MyOut := #MyInt1 + 1
4 #MyOut := #MyInt2 + #MyInt;
5 #MyOut = 1;

```

Location	Message
3	Semicolon missing.
4	Tag #MyInt not defined.
5	Unknown instruction.

16

SCL - Debugging Logic Errors


- Syntactically correct, but logic is incorrect
- Monitor all logic or monitor from selected line on
- Use Watch Tables

The screenshot shows the SIMATIC Manager interface for a project named 'SCL_Demo'. The main window displays SCL code for a function block 'My First SCL [FC1]'. The code includes a comment '// Monitoring with glasses' and two assignment statements for '#MyOut'. A watch table on the right side of the code editor shows the current values of these variables.

Variable	Value
#MyOut	1
#MyInt1	0
#MyOut	11
#MyOut	1
#MyInt2	10

17

SCL - Debugging Logic with Breakpoints

- As of V15 the following CPU's support breakpoints
 - S7300/400 and S7-1500 (firmware 2.5 or later)
- Single step
- Run to cursor
-  Breakpoint not allowed.

The screenshot shows the SIMATIC Manager interface for a project named 'SCL_Demo'. The main window displays SCL code for a function block 'My First SCL Breakpoints [FC1]'. The code includes several conditional statements and assignments for '#MyOut'. A watch table on the right side of the code editor shows the current values of these variables. A red arrow points to the 'Breakpoints' section in the 'Options' panel on the right, which lists several breakpoints set on the code.

Variable	Value
#MyOut	0
Result	TRUE
#In_1	41
#MyOut	20
Result	TRUE
#In_1	41
#MyOut	30
Result	TRUE
#In_1	41
#MyOut	40

18

SCL - Data types and conversion

- Implicit conversion will take place or use CONVERT

```

1 // Using different data types
2 // Word type for BCD_Val
3 "BCD_Val" := "MyInt_1";          3 "BCD_Val" := INT_TO_BCD16("MyInt_1");

```

The screenshot shows the SIMATIC Manager interface. A warning message is displayed in the 'Messages' pane: 'The sign or the accuracy of the value may be lost.' The 'CONVERT' dialog box is open, showing 'Source type: INT' and 'Target type: _TO_'. The dialog lists various data types: BCD16, BOOL, BYTE, CHAR, DINT, DWORD, LINT, LREAL, LWORD, REAL, and SINT. A red arrow points from the warning message to the dialog box. The 'Properties' pane on the right shows 'Conversion operations' with a list of functions including CONVERT, ROUND, CEIL, FLOOR, TRUNC, SCALE_X, NORM_X, REF, Variant, Legacy, Program control operations, Word logic operations, and Extended instructions.

19

SCL - Typed and non-typed constants

- Implicit conversion will take place
- It is best to type the constants, eliminates yellow warnings
 - INT#, DINT#, REAL# etc.

The screenshot shows the SIMATIC Manager interface with SCL code and a variable declaration table. The code includes comments and assignments for 'MyReal_Result' and 'MyInt_1'. The table shows the values of these variables after compilation.

"MyReal_Result"	%MD24	-15536.0
"MyInt_1"	%MW10	0
"MyReal_Result"	%MD24	50000.0
"MyInt_1"	%MW10	0
"MyReal_Result"	%MD24	3.5
"MyInt_1"	%MW10	0
"MyReal_Result"	%MD24	3.5
"MyInt_1"	%MW10	0

The 'Messages' pane shows two warnings: 'The sign or the accuracy of the value may be lost.' (Location 12) and 'Value of constant outside the permitted range.' (Location 9).

20

SCL – Result data Type of Arithmetic functions

- Two fixed-point numbers with sign, result receives larger type
 - **INT + DINT = DINT**
- Two fixed-point numbers without sign, result receives larger type
 - **USINT + UDINT = UDINT**
- One fixed-point number with sign and the other does not, result receives next larger type with sign
 - **SINT + USINT = INT**
- One fixed-point number and floating point, result receives floating point type
 - **INT + REAL = REAL**
- Two floating-point types, result receives larger floating point type
 - **REAL + LREAL = LREAL**

21

SCL – Arithmetic, Logical and Relational expressions

Operation	Operator
Power	**
Unary plus	+
Unary minus	-
Multiplication	*
Division	/
Modulo function	MOD

Operation	Operator
Negation (generate one's complement)	NOT
AND logic operation	AND or &
OR logic operation	OR
EXCLUSIVE OR logic operation	XOR

Operation	Operator
Compare for equal, not equal	=, <>
Compare for less than, less than-equal to, greater than, greater than or equal to	<, <=, >, >=

22

SCL – Program Control with If ... Then ...

- Basic Instructions – Drag into SCL code

```

1 // Program control IF
2 IF condition THEN
3     // Statement section IF
4     ;
5 ELSE
6     // Statement section ELSE
7     ;
8 END_IF;

```

```

1 // Program control IF
2
3 IF "MyInt_1" > 20 THEN
4     "Light_2" := True;
5 ELSE
6     "Light_2" := False;
7 END_IF;
8

```

Name	Description
scl IF ... THEN ...	Run conditionally
scl IF ... THEN ... ELSE ...	Branch conditionally
scl IF ... THEN ... ELSEIF ...	Branch conditionally multiple tim...
scl CASE ... OF ...	Create multiway branch
scl FOR ... TO ... DO ...	Run in counting loop
scl FOR ... TO ... BY ... DO ...	Run in counting loop with step wi...
scl WHILE ... DO ...	Run if condition is met
scl REPEAT ... UNTIL ...	Run if condition is not met
scl CONTINUE	Recheck loop condition
scl EXIT	Exit loop immediately
scl GOTO ...	Jump
scl RETURN	Exit block
scl (* *)	Insert a comment section
scl REGION	Organize source code

23

SCL – Program Control with CASE ... OF ...

```

1 // Program control CASE
2 CASE variable name OF
3     1: // Statement section case 1
4     ;
5     2..4: // Statement section case 2 to 4
6     ;
7     ELSE // Statement section ELSE
8     ;
9 END_CASE;

```

Name	Data type	Default value	Comment
Input			
Mode	Int		
Output			
Mode1Selected	Bool		
Mode234Selected	Bool		

```

1 // Program control CASE
2 CASE #Mode OF
3     1:
4         // Statement section case 1
5         #Mode1Selected := TRUE;
6     2..4:
7         // Statement section case 2 to 4
8         #Mode234Selected:= TRUE;
9     ELSE
10        // Statement section ELSE Mode <> 1-4
11        #Mode1Selected := FALSE;
12        #Mode234Selected := FALSE;
13 END_CASE;

```

24

SCL – Program Control with For, While and Repeat loops

```

1 // Program control with For Loop
2
3 FOR counter := start_count TO end_count DO
4     // Statement section FOR
5     ;
6 END_FOR;

```

```

1 // Program control with While Loop
2
3 WHILE condition DO
4     // Statement section WHILE
5     ;
6 END_WHILE;

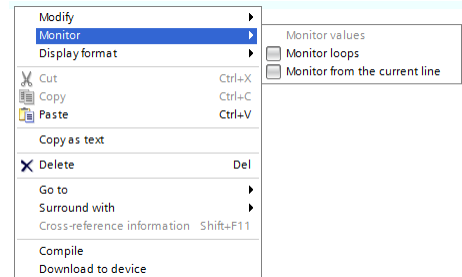
```

```

1 // Program control with Repeat Loop
2
3 REPEAT
4     // Statement section REPEAT
5     ;
6 UNTIL condition END_REPEAT;

```

Tip: Enable Monitor Loops



25

SCL – Program Control with For, While and Repeat loops

```

1 // Program control with loops
2
3 "MyInt_1" := 0;
4
5 FOR #i := 1 TO 5 DO
6     "MyInt_1" := "MyInt_1" + 1;
7 END_FOR;
8
9 WHILE "MyInt_1" > 0 DO
10    "MyInt_1" := "MyInt_1" - 1;
11 END_WHILE;
12
13 REPEAT
14    "MyInt_1" := "MyInt_1" + 2;
15 UNTIL "MyInt_1" = 10 END_REPEAT;

```

26

SCL - Program Control with For, While and Repeat loops

- CONTINUE
 - Finishes current execution of FOR, WHILE, or REPEAT loop
 - Skips remaining lines and jumps to loop check condition
- EXIT
 - Leaves a FOR, WHILE, or REPEAT at any point in loop
 - Skips remaining lines while exiting loop

```

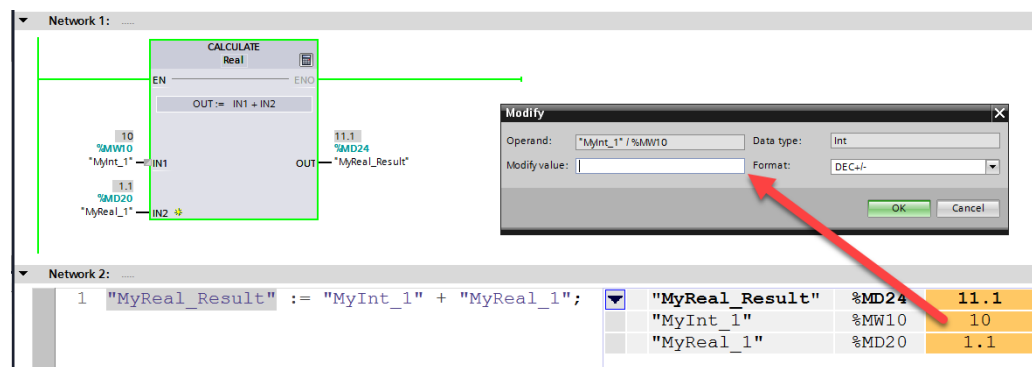
8 FOR #i := 1 TO 5 DO
9     IF #MyA_Total > 10 THEN
10        CONTINUE;           // top of loop
11    END_IF;
12    #MyA_Total := #MyA_Total + #MyA[#i];
13    IF "Button_1" THEN
14        EXIT;                // jump out of loop
15    END_IF;
16 END_FOR;

```

27

SCL - Math compared to CALCULATE LAD Block

- CALCULATE block uses INx parameters in expressions
- SCL expression uses actual tag names including constants
- SCL easier to read



Network 1: ...

Network 2: ...

1 "MyReal_Result" := "MyInt_1" + "MyReal_1";

"MyReal_Result"	%MD24	11.1
"MyInt_1"	%MW10	10
"MyReal_1"	%MD20	1.1

28

SCL - String processing

```

1 (* find first,last name delimited by comma
2   search in FirstAndLastNameString
3   store first name in #FirstNameString
4   store last name in #LastNameString
5   return 0 if names extracted, return 1 if no comma
6 *)
7
8 #CommaPos := FIND(IN1 := #FirstAndLastNameString,
9                 IN2 := WSTRING#',' );
10
11 IF #CommaPos > 0 THEN
12     #FirstNameString := LEFT(IN := #FirstAndLastNameString,
13                             L := #CommaPos - 1);
14     #LastNameString := MID(IN := #FirstAndLastNameString,
15                            L := 254,
16                            P := #CommaPos + 1);
17     #My First SCL Strings := 0;
18 ELSE
19     #FirstNameString := WSTRING#'';
20     #LastNameString := WSTRING#'';
21     #My First SCL Strings := 1;
22 END_IF;
    
```

Extended instructions	
Name	Description
String - Char	
S_CONV	Convert character string
STRG_VAL	Convert character string to numerical value
VAL_STRG	Convert numerical value to character string
Strg_TO_Chars	Convert character string to Array of CHAR
Chars_TO_Strg	Convert Array of CHAR to character string
MAX_LEN	Determine the length of a character string
JOIN	Join multiple strings
SPLIT	Splitting an array of characters into multiple strings
ATH	Convert ASCII string to hexadecimal number
HTA	Convert hexadecimal number to ASCII string
LEN	Determine the length of a character string
CONCAT	Combine character strings
LEFT	Read the left characters of a character string
RIGHT	Read the right characters of a character string
MID	Read the middle characters of a character string
DELETE	Delete characters in a character string
INSERT	Insert characters in a character string
REPLACE	Replace characters in a character string
FIND	Find characters in a character string

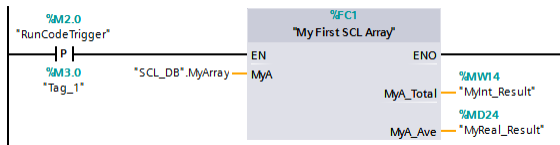


29

SCL - Array Processing

- Array defined in DB

MyArray	Array[1..5] of Int	
MyArray[1]	Int	1
MyArray[2]	Int	2
MyArray[3]	Int	3
MyArray[4]	Int	4
MyArray[5]	Int	5



Name	Data type	Default value	Comment
Input			
MyA	Array[*] of Int		
MyA[*]	Int		
Output			
MyA_Total	Int		
MyA_Ave	Real		
InOut			
Temp			
size	Dint		
start	Dint		
i	Int		

```

1 #size := UPPER_BOUND(ARR := #MyA, DIM := 1);
2 #start := LOWER_BOUND(ARR := #MyA, DIM := 1);
3
4 #MyA_Total := 0;
5 #MyA_Ave := 0;
6
7 FOR #i := #start TO #size DO
8     #MyA_Total := #MyA_Total + "SCL_DB".MyArray[#i];
9 END_FOR;
10
11 #MyA_Ave := #MyA_Total / #size;
    
```



30

SCL – Editing Tips with Smart Editor

- SCL Editor is more than a simple text editor, watch for colors

```
1 // Intellisense for browsing functions, tags, db elements
2 // Enter DB name followed by period
3 "SCL_DB".
```

MyString1	String			
MyString2	String			
NewString	String			

```
1 // Watch for Blue color selections
2 // Dbl click text to get Blue, Start typing to use intellisense
3 "SCL_DB".NewString :=CONCAT(IN1:=string_in, IN2:=string_in.)
```

```
1 // Watch for Yellow color selections when adding instructions
2 // When Yellow and typing you are just adding text
3 "SCL_DB".NewString := CONCAT(IN1:=stringTEXT_in, IN2:=string_in.)
```

```
1 // Watch for Light Green color selections after adding ; to end of statement
2 // Now you can press enter at the comma to create a new line in middle of instruction
3 "SCL_DB".NewString := CONCAT(IN1 := stringTEXT_in, IN2 := string_in.);
```

```
1 // Watch for Gray color selections (all uses highlighted)
2 "SCL_DB".NewString := CONCAT(IN1 := "SCL_DB".MyString1, IN2 := "SCL_DB".MyString1);
```

31

Live Demos

- Embed SCL in LAD Block
- Create SCL Block
- Array Processing
- Breakpoints

32

SCL - Resources

- Structured Control Language (SCL V4, V5.0) for S7-300/S7-400 Programming
<https://support.industry.siemens.com/cs/us/en/view/1137188>
- S7-SCL V5.3 for S7-300/400 Getting Started
<https://support.industry.siemens.com/cs/us/en/view/18735131>
- STEP 7 Basic/Professional V15.1 and SIMATIC WinCC V15.1 System Manual
<https://support.industry.siemens.com/cs/us/en/view/109755202>
- Berger Book- Automating with SIMATIC S7-1500

SCL Chapter