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

Kelly Anton | 10-24-2019

Agenda

• Brief Overview of SCL
• SCL Editor
• Create simple SCL block
• Debugging SCL
• Data types
• Program Control
• Math, Strings and Arrays
• Live Demos
SCL (Structured Control Language) - Defined

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

SCL – Common Uses

• Math
• String Operations
• Array Operations
• Communication
• Program Flow
SCL – Options > Settings (Editor Settings)

- General > Script/text editors Font size

- PLC programming > SCL (Structured Control Language)

SCL – FC/FB Blocks

- Add New SCL FC/FB Block
SCL – FC/FB Block Parameters

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

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
SCL - Calling FC’s and FB’s

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

SCL – SCL Network Embedded in Ladder (LAD)

- Insert SCL Network in LAD...
**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", DegC="TempDegC");
```

**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 *)
8
```
SCL – Regions

- Used for readability of code

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
SCL – Surround with

- Quickly wrap selected lines of code with statement

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

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

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

10/2/2019
**SCL – Debugging Logic Errors**

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

**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.
SCL – Data types and conversion

- Implicit conversion will take place or use CONVERT

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

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.

```plaintext
5 // 50000 is converted to int so it becomes -12536
6 "MyReal_Result" := "MyInt_1" + DINT#50000;
7 "MyReal_Result" := "MyInt_1" + 3.5;
8 "MyReal_Result" := "MyInt_1" + REAL#3.5;
```
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`

---

SCL – Arithmetic, Logical and Relational expressions

<table>
<thead>
<tr>
<th>Operation</th>
<th>Operator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power</td>
<td><code>^</code></td>
</tr>
<tr>
<td>Unary plus</td>
<td><code>+</code></td>
</tr>
<tr>
<td>Unary minus</td>
<td><code>-</code></td>
</tr>
<tr>
<td>Multiplication</td>
<td><code>*</code></td>
</tr>
<tr>
<td>Division</td>
<td><code>/</code></td>
</tr>
<tr>
<td>Modulo function</td>
<td><code>MOD</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Operation</th>
<th>Operator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Negation (generate one's complement)</td>
<td><code>NOT</code></td>
</tr>
<tr>
<td>AND logic operation</td>
<td><code>AND</code></td>
</tr>
<tr>
<td>OR logic operation</td>
<td><code>OR</code></td>
</tr>
<tr>
<td>EXCLUSIVE OR logic operation</td>
<td><code>XOR</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Operation</th>
<th>Operator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compare for equal, not equal</td>
<td><code>=</code> <code>&lt;&gt;</code></td>
</tr>
<tr>
<td>Compare for less than, less than or equal to</td>
<td><code>&lt;</code>, <code>&lt;=</code></td>
</tr>
<tr>
<td>Greater than, greater than or equal to</td>
<td><code>&gt;</code>, <code>&gt;=</code></td>
</tr>
</tbody>
</table>
SCL – Program Control with If ... Then ...

- Basic Instructions – Drag into SCL code

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

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

```scl
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;
```
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 END_FOR;

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

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

Tip: Enable Monitor Loops

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;
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

```plaintext
FOR #i := 1 TO 5 DO
  IF #MyA_Total > 10 THEN
    CONTINUE; // top of loop
  END_IF;
  #MyA_Total := #MyA_Total + #MyA[#i];
  IF "Button_1" THEN
    EXIT; // jump out of loop
  END_IF;
END_FOR;
```

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
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;

SCL – Array Processing

- Array defined in DB

```c
MyArray = [1..5] of int
MyArray[2] = 3
MyArray[4] = 4
```

```c
#size := UPPER_BOUND(ARR := #MyA, DIM := 1);
#start := LOWERBOUND(ARR := #MyA, DIM := 1);
#MyA_Total := 0;
#MyA_Ave := 0;
FOR #i := #start TO #size DO
  #MyA_Total := #MyA_Total + "SCL_DB".MyArray[#i];
END_FOR;
#MyA_Ave := #MyA_Total / #size;
```
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",

<table>
<thead>
<tr>
<th>MString1 String</th>
</tr>
</thead>
<tbody>
<tr>
<td>MString2 String</td>
</tr>
<tr>
<td>NewString String</td>
</tr>
</tbody>
</table>

1 // Watch for Blue color selections
2 // Db1 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 := _string_text_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 := _string_text_in_, IN2 := _string_in_);
```

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
- S7-SCL V5.3 for S7-300/400 Getting Started
- STEP 7 Basic/Professional V15.1 and SIMATIC WinCC V15.1 System Manual
- Berger Book- Automating with SIMATIC S7-1500
  SCL Chapter