Ladder Logic / Diagrams

CPE200, Fall 2023

Hank Dietz

http://aggregate.org/hankd/

University of
Kentucky

Why are we doing this?

- Because career fair folks asked for it… ;-)
- Ladder logic is ***still*** commonly used
  - PLCs (Programmable Logic Controllers)
  - Makes programming look like circuits; originally switches and relays
  - Each “rung” is a rule relating inputs & output; sort-of like an if-then statement

Ladder Basics

```
-[ ]- or -] [-     Normally open contact
-[\]- or -]\[-     Normally closed contact
-[blocktype]-      Special block

-[ ]--[ ]-         Series means AND

-+-[ ]-+-          Parallel means OR
 |     |
 +-[ ]-+

-( )-              Normally inactive coil (output)
-(\)-              Normally active coil (output)
```

LED is on

```
||     ||
|| LED ||
||-( )-||
||     ||
```

LED is on while Switch is pressed

```
||                        ||
||   Switch        LED    ||
||-----[ ]---------( )----||
||                        ||
```

LED is on while Switch0 is pressed AND Switch1 isn’t

```
||                     ||
|| Switch0 Switch1 LED ||
||---[ ]-----[\]---( )-||
||                     ||
```

LED is on while Switch0 is pressed OR Switch1 isn’t

```
||                 ||
||   Switch0   LED ||
||-+---[ ]---+-( )-||
|| |         |     ||
|| | Switch1 |     ||
|| +---[\]---+     ||
||                 ||
```

LED0 is as LED before, but LED1 is on while LED0 is off

```
||                  ||
||   Switch0   LED0 ||
||-+---[ ]---+-( )--||
|| |         |      ||
|| | Switch1 |      ||
|| +---[\]---+      ||
||                  ||
|| LED0         LED1 ||
||-[\]---------( )--||
||                  ||
```

An example: `Alarm` sounds when either sensor is triggered

```
||                   ||
||   Sensor0   Alarm ||
||-+---[ ]---+--( )--||
|| |         |       ||
|| | Sensor1 |       ||
|| +---[ ]---+       ||
||                   ||
```

But we also want `Alarm` if a wire to a sensor breaks...

```
||                       ||
|| Sensor0 Sensor1  AOK  ||
||---[\]-----[\]----( )--||
||                       ||
|| AOK             Alarm ||
||-[\]--------------( )--||
||                       ||
```

Implementations?

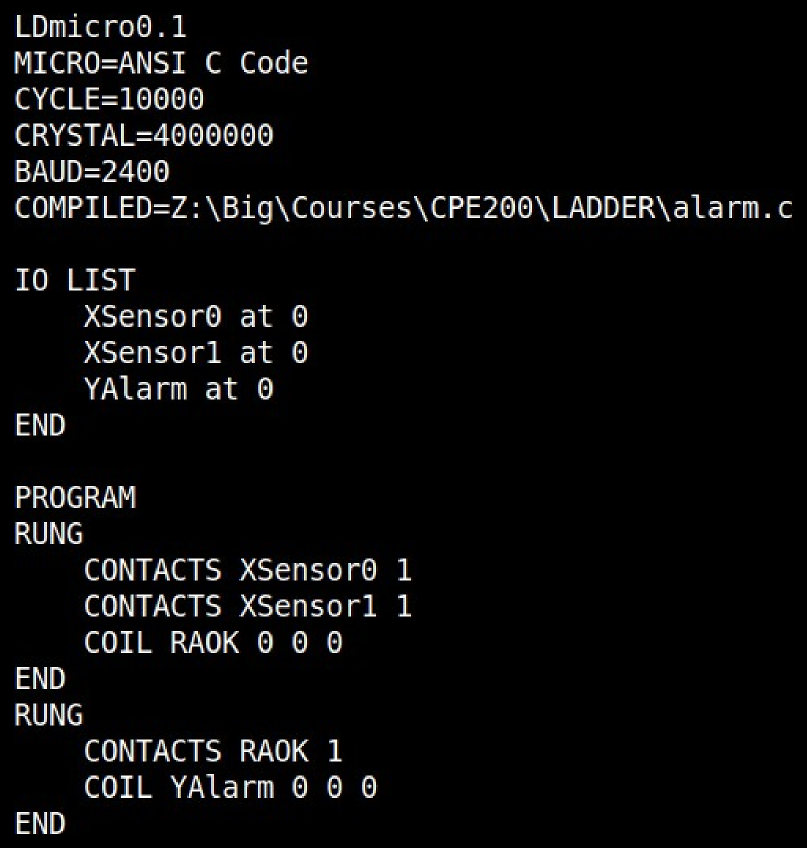
- There is a standard: IEC 61131-3, however, Allen Bradley, Siemens, etc., PLCs differ a bit
- **LDmicro** is an old free editor+compiler `https://cq.cx/ladder.pl`
- **OpenPLC** is open-source PLC software compliant with IEC 61131-3

  `https://autonomylogic.com/docs/openplc-overview/`

LDmicro description of Alarm

```
LDmicro export text
for 'ANSI C Code', 4.000000 MHz crystal, 10.0 ms cycle time


LADDER DIAGRAM:

   ||                                                    ||
   ||    XSensor0         XSensor1            RAOK       ||
 1 ||-------]/[--------------]/[---------------( )-------||
   ||                                                    ||
   ||                                                    ||
   ||                                                    ||
   ||                                                    ||
   ||      RAOK                              YAlarm      ||
 2 ||-------]/[--------------------------------( )-------||
   ||                                                    ||
   ||                                                    ||
   ||                                                    ||
   ||------[END]-----------------------------------------||
   ||                                                    ||
   ||                                                    ||


I/O ASSIGNMENT:

  Name                        | Type                | Pin
 -----------------------------+--------------------+------
  XSensor0                    | digital in          | (not assigned)
  XSensor1                    | digital in          | (not assigned)
  YAlarm                      | digital out         | (not assigned)
  RAOK                        | int. relay          |
```

LDmicro structured text for Alarm



```
LDmicro0.1
MICRO=ANSI C Code
CYCLE=10000
CRYSTAL=4000000
BAUD=2400
COMPILED=Z:\Big\Courses\CPE200\LADDER\alarm.c

IO LIST
    XSensor0 at 0
    XSensor1 at 0
    YAlarm at 0
END

PROGRAM
RUNG
    CONTACTS XSensor0 1
    CONTACTS XSensor1 1
    COIL RAOK 0 0 0
END
RUNG
    CONTACTS RAOK 1
    COIL YAlarm 0 0 0
END
```

LDmicro C code for Alarm

```
/* U_xxx symbols correspond to user-defined names. There is such a symbol
   for every internal relay, variable, timer, and so on in the ladder
   program. I_xxx symbols are internally generated. */
STATIC BOOL I_b_mcr = 0;
#define Read_I_b_mcr() I_b_mcr
#define Write_I_b_mcr(x) I_b_mcr = x
STATIC BOOL I_b_rung_top = 0;
#define Read_I_b_rung_top() I_b_rung_top
#define Write_I_b_rung_top(x) I_b_rung_top = x

/* You provide this function. */
PROTO(extern BOOL Read_U_b_XSensor0(void);)


/* You provide this function. */
PROTO(extern BOOL Read_U_b_XSensor1(void);)

STATIC BOOL U_b_RAOK = 0;
#define Read_U_b_RAOK() U_b_RAOK
#define Write_U_b_RAOK(x) U_b_RAOK = x

/* You provide these functions. */
PROTO(BOOL Read_U_b_YAlarm(void);)
PROTO(void Write_U_b_YAlarm(BOOL v);)
```

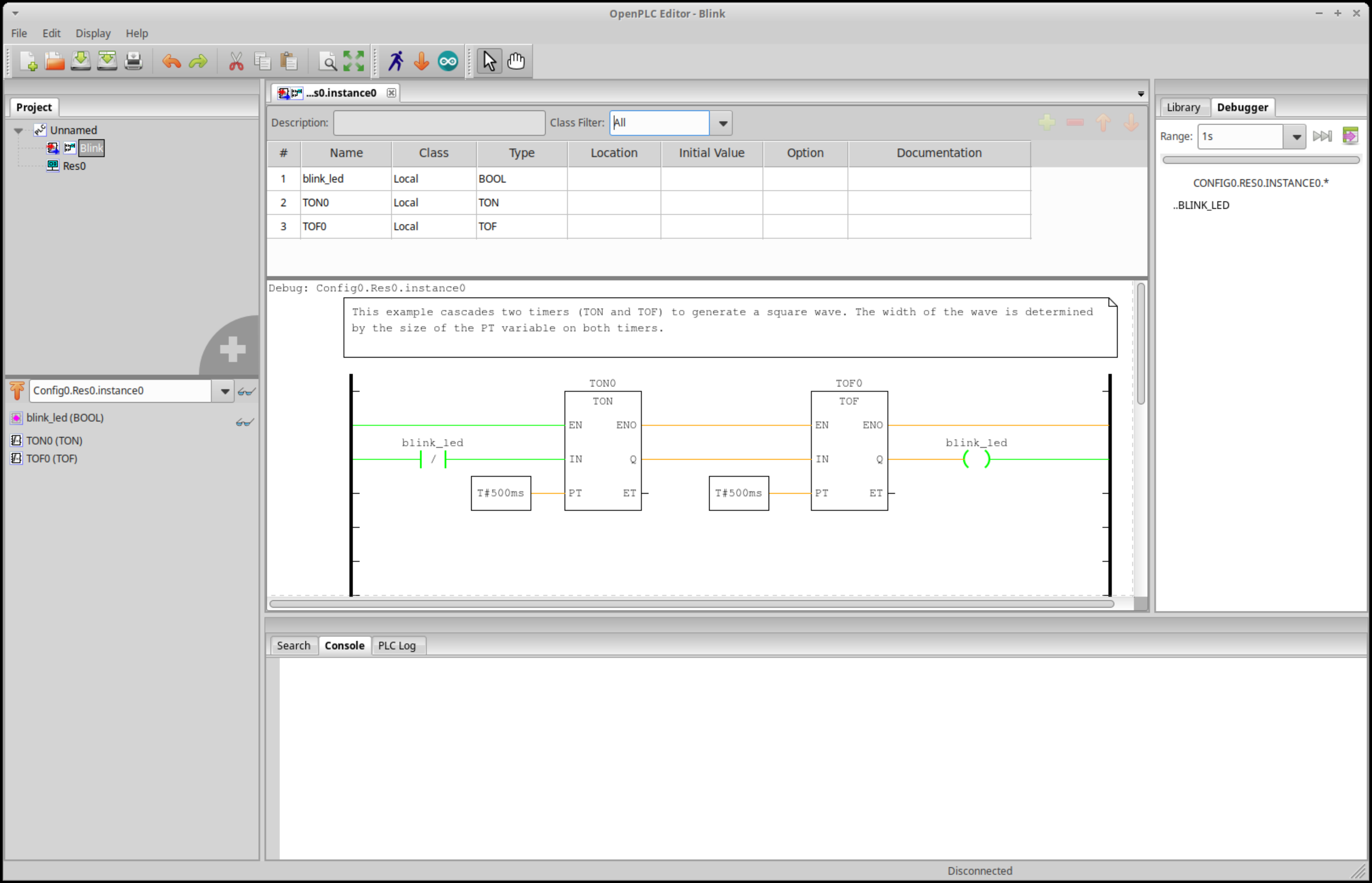
```
/* Call this function once per PLC cycle. You are responsible for calling
   it at the interval that you specified in the MCU configuration when you
   generated this code. */
void PlcCycle(void)
{
    Write_I_b_mcr(1);

    /* start rung 1 */
    Write_I_b_rung_top(Read_I_b_mcr());

    /* start series [ */
    if(Read_U_b_XSensor0()) {
        Write_I_b_rung_top(0);
    }

    if(Read_U_b_XSensor1()) {
        Write_I_b_rung_top(0);
    }

    Write_U_b_RAOK(Read_I_b_rung_top());

    /* ] finish series */

    /* start rung 2 */
    Write_I_b_rung_top(Read_I_b_mcr());

    /* start series [ */
    if(Read_U_b_RAOK()) {
        Write_I_b_rung_top(0);
    }

    Write_U_b_YAlarm(Read_I_b_rung_top());

    /* ] finish series */
}
```

Special Blocks for LDmicro...

`-[OSR]-`	One shot rising; `OSF` for falling
`-[TON]-`	Turn On with delay; `TOF` for off
`-[CTU]-`	Count Up; `CTD` Down; `CTC` Circular
`-[EQU]-`	== comparison; also `NEQ` != and `GRT` >; `GEQ` >=; `LES` <; `LEQ` <=
`-[MOV]-`	Move; also `ADD`; `SUB`; `MUL`; `DIV`

Others include ADC read, PWM output, shift register, look-up table, and many more...

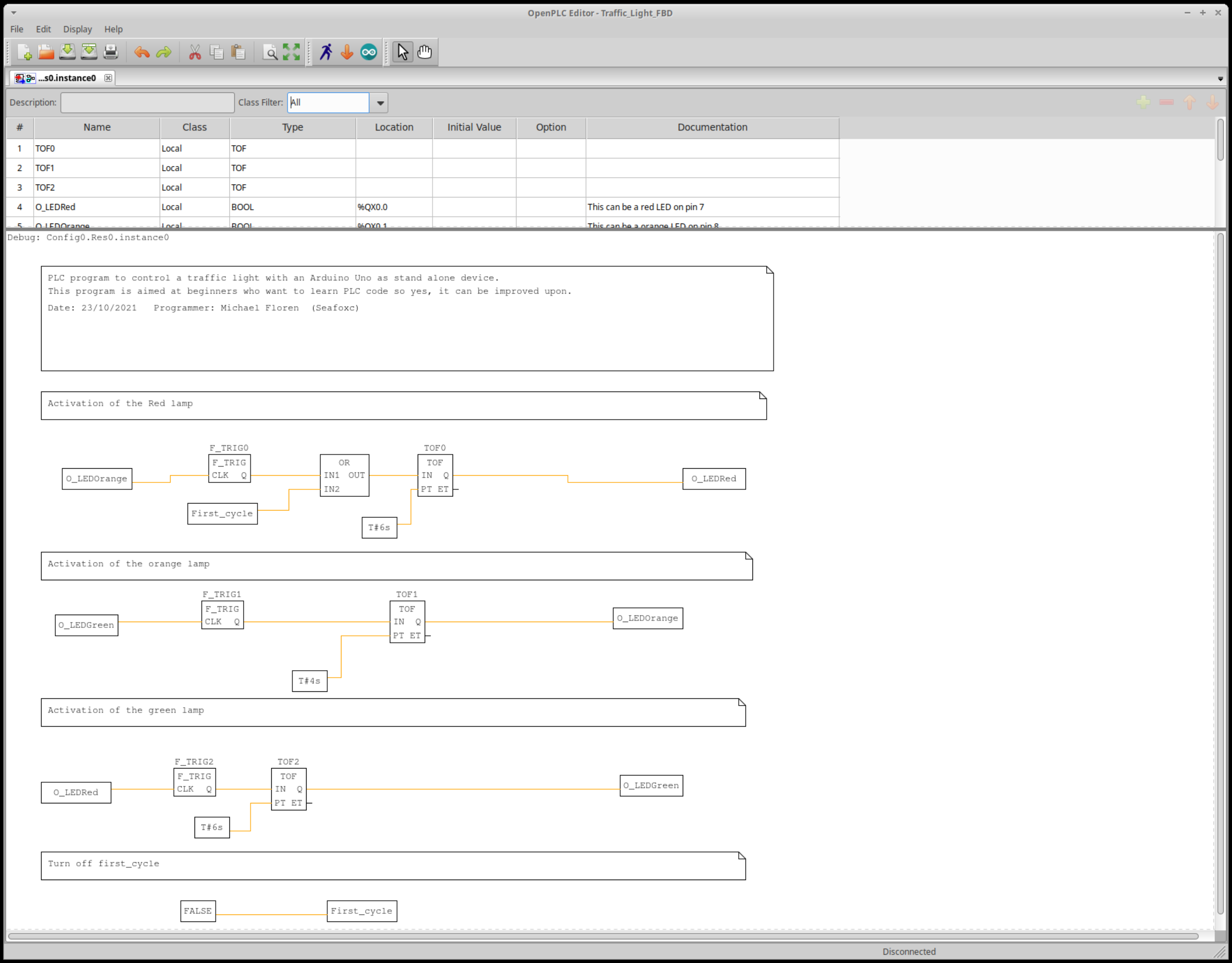
OpenPLC Editor: Blink a LED



#	Name	Class	Type	Location	Initial Value	Option	Documentation
1	blink_led	Local	BOOL				
2	TON0	Local	TON				
3	TOF0	Local	TOF				

This example cascades two timers (TON and TOF) to generate a square wave. The width of the wave is determined by the size of the PT variable on both timers.

OpenPLC Editor: Traffic Light



Description:

Class Filter: All

#	Name	Class	Type	Location	Initial Value	Option	Documentation
1	TOF0	Local	TOF				
2	TOF1	Local	TOF				
3	TOF2	Local	TOF				
4	O_LEDRed	Local	BOOL	%QX0.0			This can be a red LED on pin 7
5	O_LEDOrange	Local	BOOL	%QX0.1			This can be a orange LED on pin 8

Debug: Config0.Res0.instance0

```
PLC program to control a traffic light with an Arduino Uno as stand alone device.
This program is aimed at beginners who want to learn PLC code so yes, it can be improved upon.
Date: 23/10/2021   Programmer: Michael Floren  (Seafoxc)
```

Activation of the Red lamp

Activation of the orange lamp

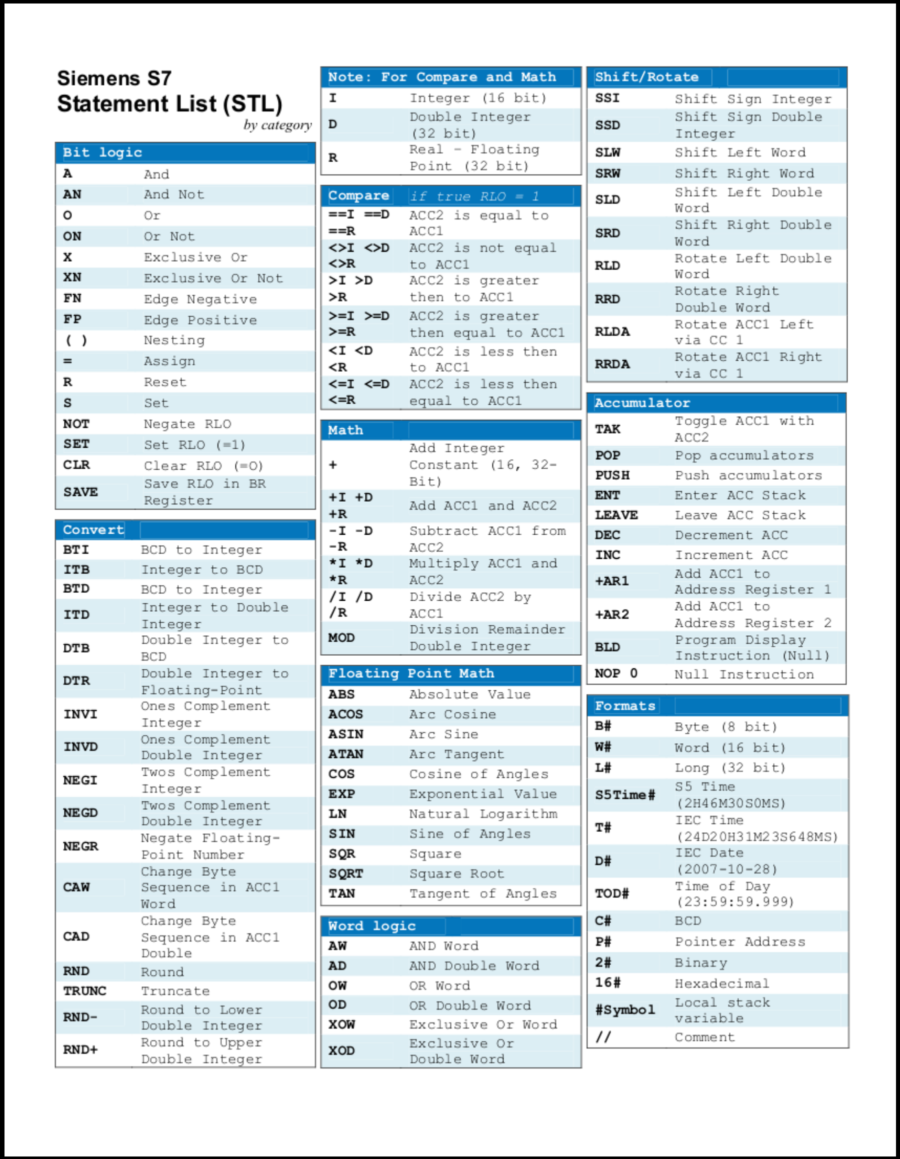
Activation of the green lamp

Turn off first_cycle

IEC 61131-3 Standard: PLC Programming Languages

- We'll stick to LD for CPE200, but...
- IEC 61131-3 lists **5 programming languages**:
  - IL: Instruction List (text)
  - ST: Structured Text (text)
  - LD: Ladder Diagram (graphical)
  - FBD: Function Block Diagram (graphical)
  - SFC: Sequential Function Chart (mixed)

IL Reference Card (Siemens)



Siemens S7 Statement List (STL)

by category

Bit logic	
A	And
AN	And Not
O	Or
ON	Or Not
X	Exclusive Or
XN	Exclusive Or Not
FN	Edge Negative
FP	Edge Positive
( )	Nesting
=	Assign
R	Reset
S	Set
NOT	Negate RLO
SET	Set RLO (=1)
CLR	Clear RLO (=0)
SAVE	Save RLO in BR Register

Convert	
BTI	BCD to Integer
ITB	Integer to BCD
BTD	BCD to Integer
ITD	Integer to Double Integer
DTB	Double Integer to BCD
DTR	Double Integer to Floating-Point
INVI	Ones Complement Integer
INVD	Ones Complement Double Integer
NEGI	Twos Complement Integer
NEGD	Twos Complement Double Integer
NEGR	Negate Floating-Point Number
CAW	Change Byte Sequence in ACC1 Word
CAD	Change Byte Sequence in ACC1 Double
RND	Round
TRUNC	Truncate
RND-	Round to Lower Double Integer
RND+	Round to Upper Double Integer

Note: For Compare and Math	
I	Integer (16 bit)
D	Double Integer (32 bit)
R	Real - Floating Point (32 bit)

Compare	*if true RLO = 1*
==I ==D ==R	ACC2 is equal to ACC1
<>I <>D <>R	ACC2 is not equal to ACC1
>I >D >R	ACC2 is greater then to ACC1
>=I >=D >=R	ACC2 is greater then equal to ACC1
<I <D <R	ACC2 is less then to ACC1
<=I <=D <=R	ACC2 is less then equal to ACC1

Math	
+	Add Integer Constant (16, 32-Bit)
+I +D +R	Add ACC1 and ACC2
-I -D -R	Subtract ACC1 from ACC2
***I *D *R**	Multiply ACC1 and ACC2
/I /D /R	Divide ACC2 by ACC1
MOD	Division Remainder Double Integer

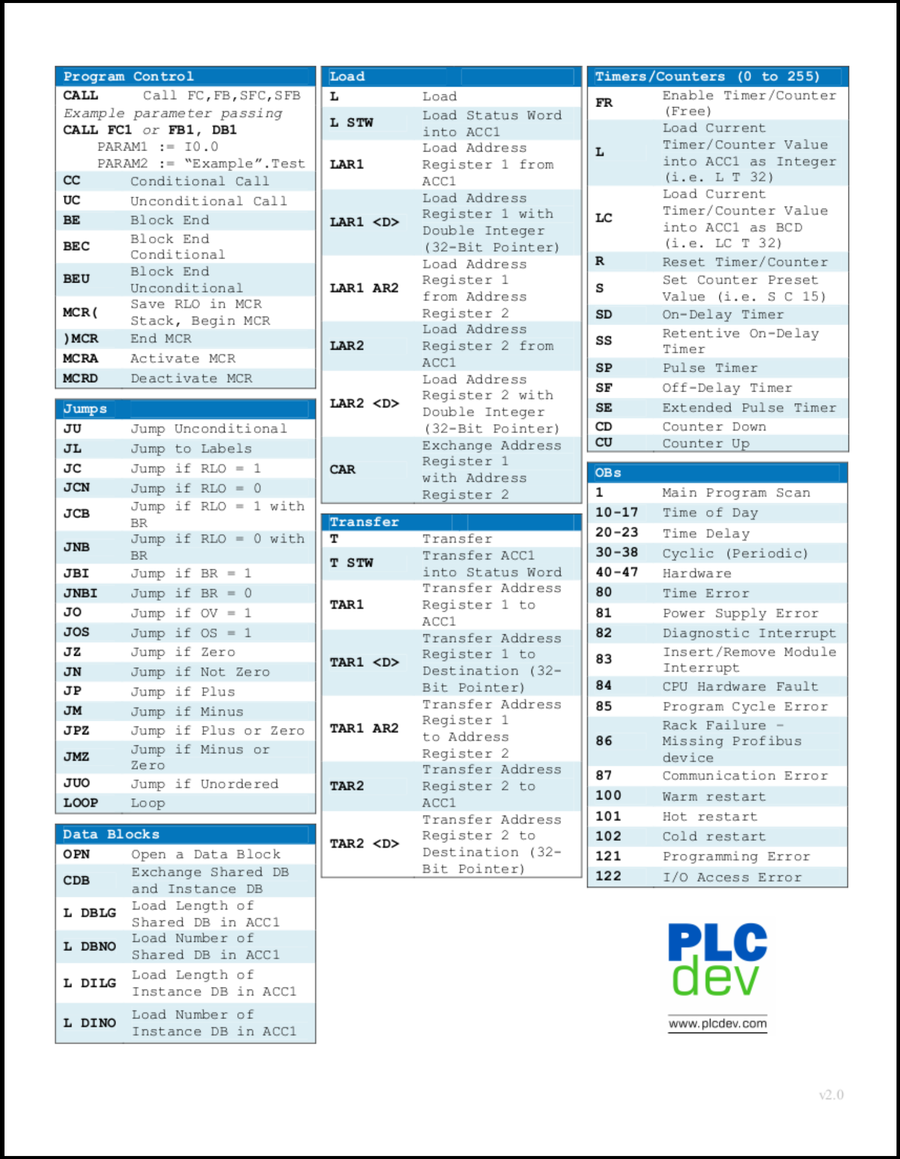
Floating Point Math	
ABS	Absolute Value
ACOS	Arc Cosine
ASIN	Arc Sine
ATAN	Arc Tangent
COS	Cosine of Angles
EXP	Exponential Value
LN	Natural Logarithm
SIN	Sine of Angles
SQR	Square
SQRT	Square Root
TAN	Tangent of Angles

Word logic	
AW	AND Word
AD	AND Double Word
OW	OR Word
OD	OR Double Word
XOW	Exclusive Or Word
XOD	Exclusive Or Double Word

Shift/Rotate	
SSI	Shift Sign Integer
SSD	Shift Sign Double Integer
SLW	Shift Left Word
SRW	Shift Right Word
SLD	Shift Left Double Word
SRD	Shift Right Double Word
RLD	Rotate Left Double Word
RRD	Rotate Right Double Word
RLDA	Rotate ACC1 Left via CC 1
RRDA	Rotate ACC1 Right via CC 1

Accumulator	
TAK	Toggle ACC1 with ACC2
POP	Pop accumulators
PUSH	Push accumulators
ENT	Enter ACC Stack
LEAVE	Leave ACC Stack
DEC	Decrement ACC
INC	Increment ACC
+AR1	Add ACC1 to Address Register 1
+AR2	Add ACC1 to Address Register 2
BLD	Program Display Instruction (Null)
NOP 0	Null Instruction

Formats	
B#	Byte (8 bit)
W#	Word (16 bit)
L#	Long (32 bit)
S5Time#	S5 Time (2H46M30S0MS)
T#	IEC Time (24D20H31M23S648MS)
D#	IEC Date (2007-10-28)
TOD#	Time of Day (23:59:59.999)
C#	BCD
P#	Pointer Address
2#	Binary
16#	Hexadecimal
#Symbol	Local stack variable
//	Comment



Program Control	
CALL	Call FC,FB,SFC,SFB
Example parameter passing	
CALL FC1 *or* **FB1, DB1**	
PARAM1 := I0.0	
PARAM2 := “Example”.Test	
CC	Conditional Call
UC	Unconditional Call
BE	Block End
BEC	Block End Conditional
BEU	Block End Unconditional
MCR(	Save RLO in MCR Stack, Begin MCR
)MCR	End MCR
MCRA	Activate MCR
MCRD	Deactivate MCR

Jumps	
JU	Jump Unconditional
JL	Jump to Labels
JC	Jump if RLO = 1
JCN	Jump if RLO = 0
JCB	Jump if RLO = 1 with BR
JNB	Jump if RLO = 0 with BR
JBI	Jump if BR = 1
JNBI	Jump if BR = 0
JO	Jump if OV = 1
JOS	Jump if OS = 1
JZ	Jump if Zero
JN	Jump if Not Zero
JP	Jump if Plus
JM	Jump if Minus
JPZ	Jump if Plus or Zero
JMZ	Jump if Minus or Zero
JUO	Jump if Unordered
LOOP	Loop

Data Blocks	
OPN	Open a Data Block
CDB	Exchange Shared DB and Instance DB
L DBLG	Load Length of Shared DB in ACC1
L DBNO	Load Number of Shared DB in ACC1
L DILG	Load Length of Instance DB in ACC1
L DINO	Load Number of Instance DB in ACC1

Load	
L	Load
L STW	Load Status Word into ACC1
LAR1	Load Address Register 1 from ACC1
LAR1 <D>	Load Address Register 1 with Double Integer (32-Bit Pointer)
LAR1 AR2	Load Address Register 1 from Address Register 2
LAR2	Load Address Register 2 from ACC1
LAR2 <D>	Load Address Register 2 with Double Integer (32-Bit Pointer)
CAR	Exchange Address Register 1 with Address Register 2

Transfer	
T	Transfer
T STW	Transfer ACC1 into Status Word
TAR1	Transfer Address Register 1 to ACC1
TAR1 <D>	Transfer Address Register 1 to Destination (32-Bit Pointer)
TAR1 AR2	Transfer Address Register 1 to Address Register 2
TAR2	Transfer Address Register 2 to ACC1
TAR2 <D>	Transfer Address Register 2 to Destination (32-Bit Pointer)

Timers/Counters (0 to 255)	
FR	Enable Timer/Counter (Free)
L	Load Current Timer/Counter Value into ACC1 as Integer (i.e. L T 32)
LC	Load Current Timer/Counter Value into ACC1 as BCD (i.e. LC T 32)
R	Reset Timer/Counter
S	Set Counter Preset Value (i.e. S C 15)
SD	On-Delay Timer
SS	Retentive On-Delay Timer
SP	Pulse Timer
SF	Off-Delay Timer
SE	Extended Pulse Timer
CD	Counter Down
CU	Counter Up

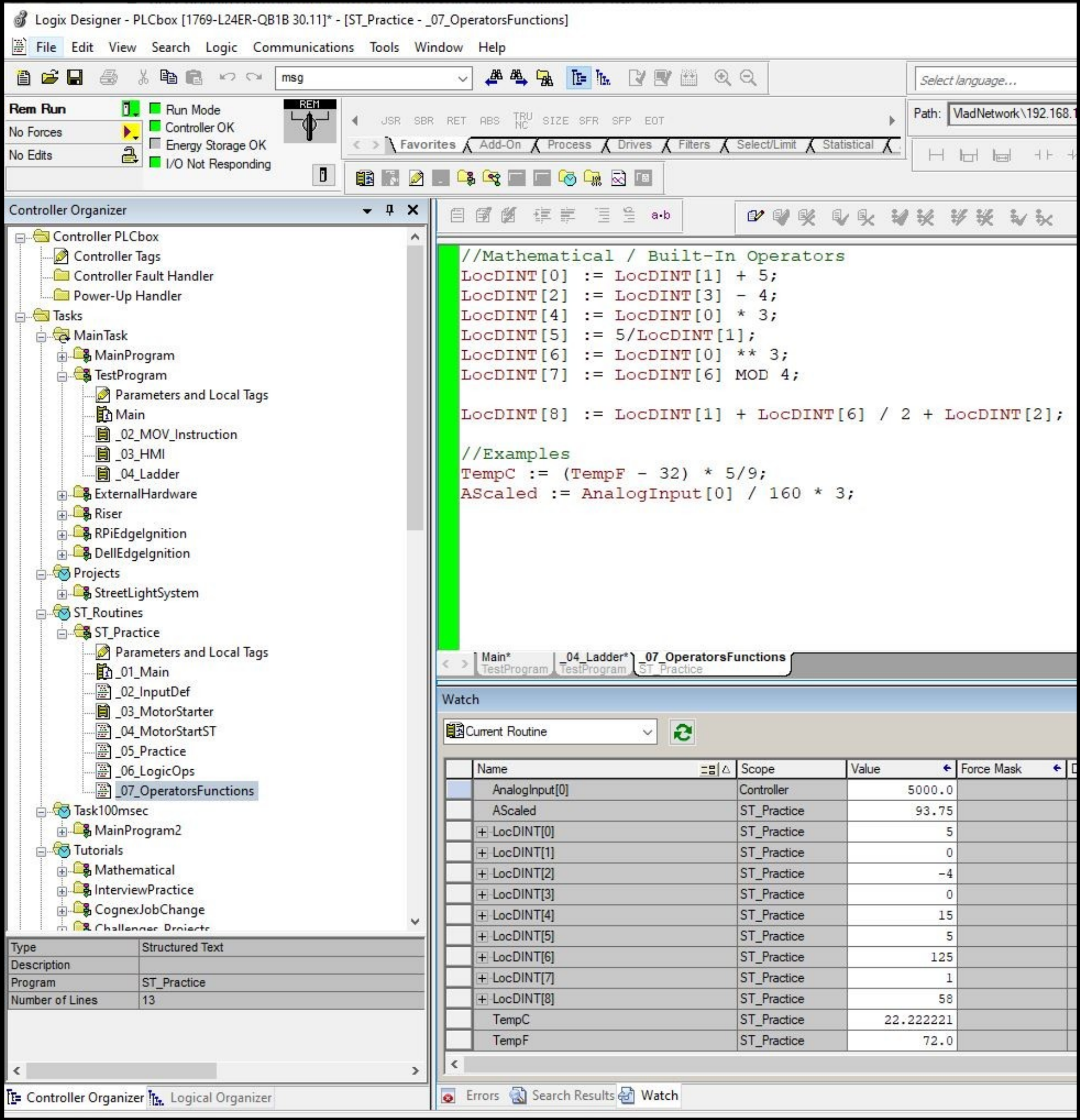
OBs	
1	Main Program Scan
10-17	Time of Day
20-23	Time Delay
30-38	Cyclic (Periodic)
40-47	Hardware
80	Time Error
81	Power Supply Error
82	Diagnostic Interrupt
83	Insert/Remove Module Interrupt
84	CPU Hardware Fault
85	Program Cycle Error
86	Rack Failure - Missing Profibus device
87	Communication Error
100	Warm restart
101	Hot restart
102	Cold restart
121	Programming Error
122	I/O Access Error

PLC dev

www.plcdev.com

v2.0

ST Example from SolisPLC



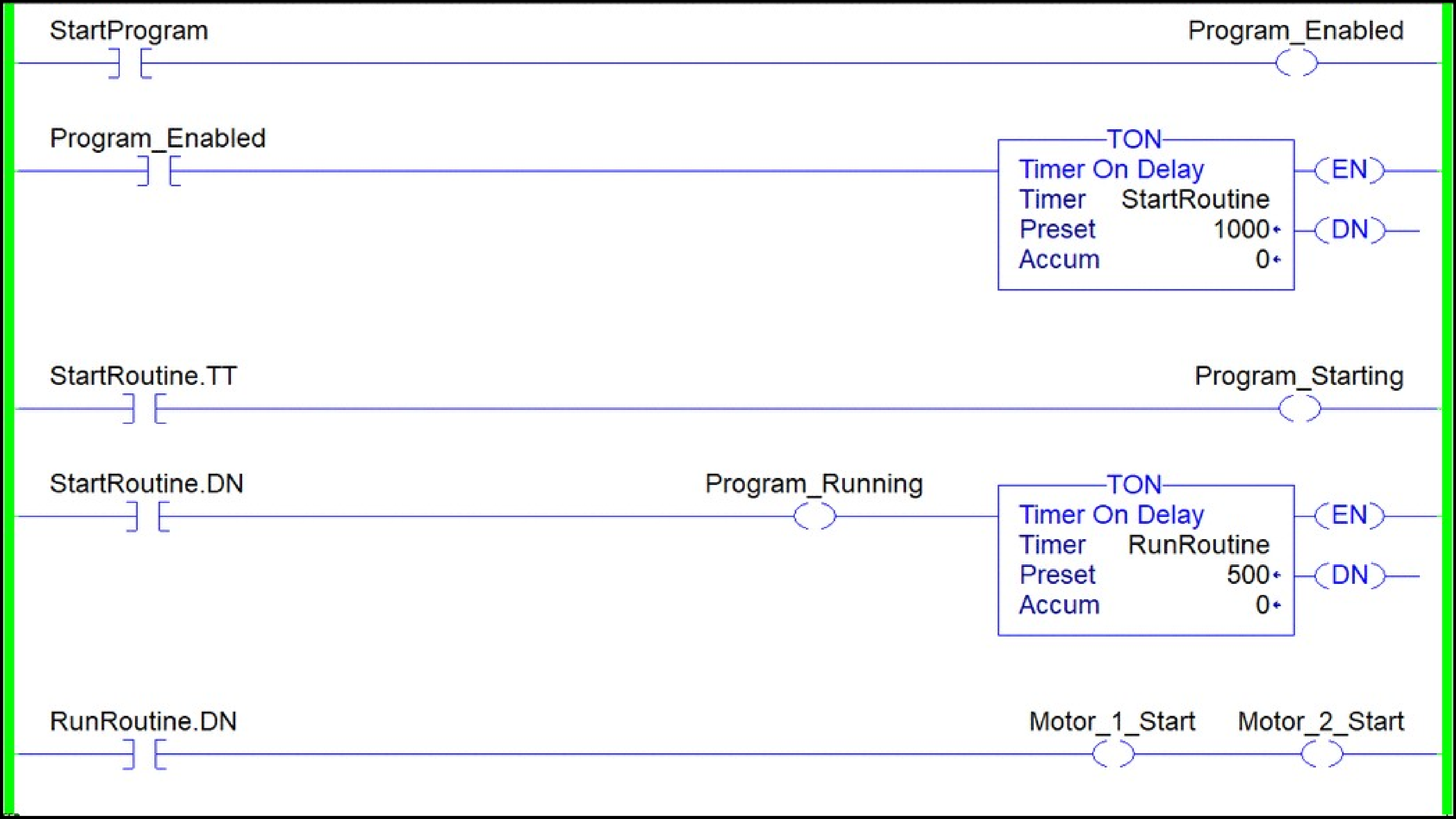
Logix Designer - PLCbox [1769-L24ER-QB1B 30.11]* - [ST_Practice - _07_OperatorsFunctions]

```
//Mathematical / Built-In Operators
LocDINT[0] := LocDINT[1] + 5;
LocDINT[2] := LocDINT[3] - 4;
LocDINT[4] := LocDINT[0] * 3;
LocDINT[5] := 5/LocDINT[1];
LocDINT[6] := LocDINT[0] ** 3;
LocDINT[7] := LocDINT[6] MOD 4;

LocDINT[8] := LocDINT[1] + LocDINT[6] / 2 + LocDINT[2];

//Examples
TempC := (TempF - 32) * 5/9;
AScaled := AnalogInput[0] / 160 * 3;
```

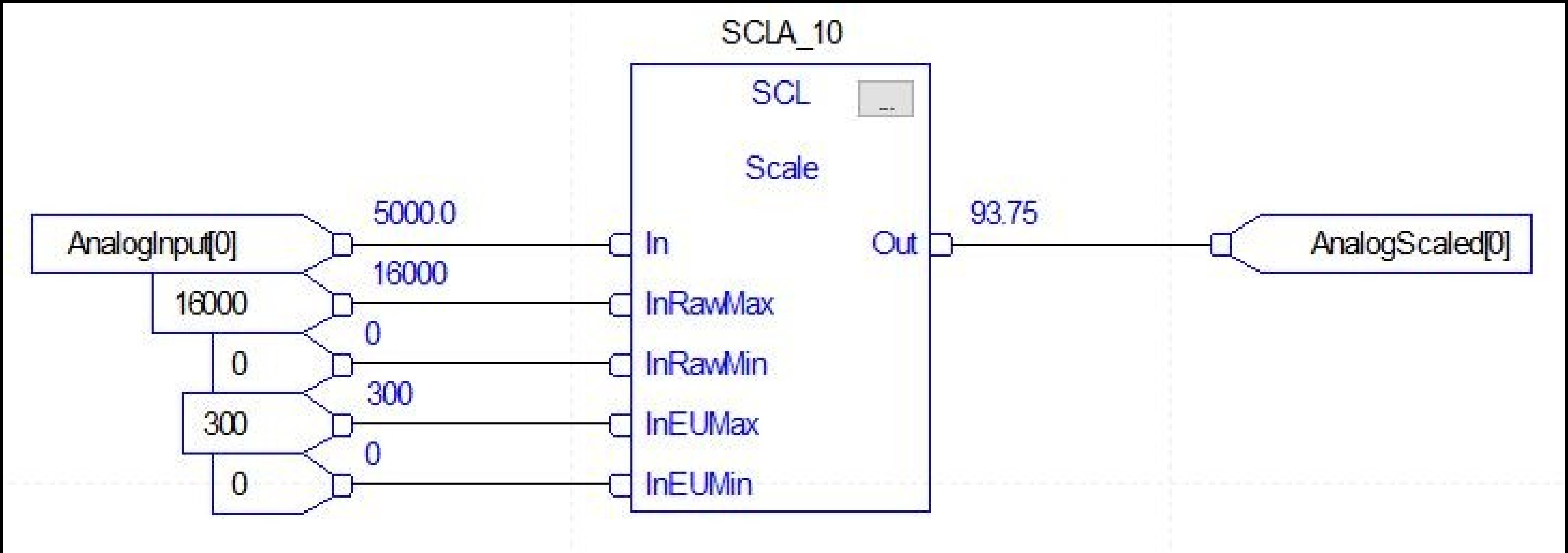
Name	Scope	Value	Force Mask
AnalogInput[0]	Controller	5000.0	
AScaled	ST_Practice	93.75	
+ LocDINT[0]	ST_Practice	5	
+ LocDINT[1]	ST_Practice	0	
+ LocDINT[2]	ST_Practice	-4	
+ LocDINT[3]	ST_Practice	0	
+ LocDINT[4]	ST_Practice	15	
+ LocDINT[5]	ST_Practice	5	
+ LocDINT[6]	ST_Practice	125	
+ LocDINT[7]	ST_Practice	1	
+ LocDINT[8]	ST_Practice	58	
TempC	ST_Practice	22.222221	
TempF	ST_Practice	72.0	

Type	Structured Text
Description	
Program	ST_Practice
Number of Lines	13

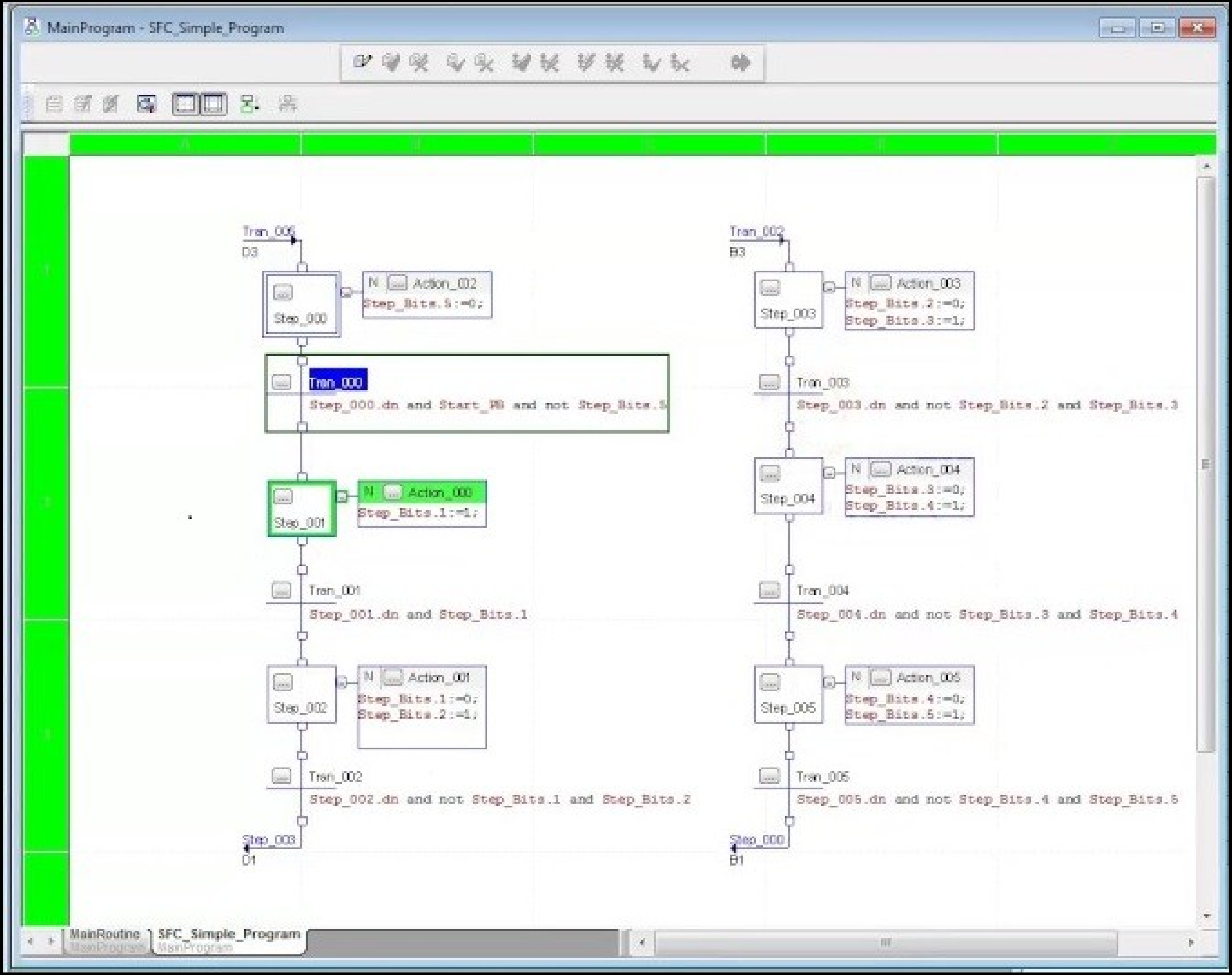
LD Example from SolisPLC



FBD Example from SolisPLC



SFC Example from SolisPLC



```
Tran_005
D3
Step_000
N  Action_002
Step_Bits.5:=0;

Tran_000
Step_000.dn and Start_PB and not Step_Bits.5

Step_001
N  Action_000
Step_Bits.1:=1;

Tran_001
Step_001.dn and Step_Bits.1

Step_002
N  Action_001
Step_Bits.1:=0;
Step_Bits.2:=1;

Tran_002
Step_002.dn and not Step_Bits.1 and Step_Bits.2

Step_003
D1
```

```
Tran_002
B3
Step_003
N  Action_003
Step_Bits.2:=0;
Step_Bits.3:=1;

Tran_003
Step_003.dn and not Step_Bits.2 and Step_Bits.3

Step_004
N  Action_004
Step_Bits.3:=0;
Step_Bits.4:=1;

Tran_004
Step_004.dn and not Step_Bits.3 and Step_Bits.4

Step_005
N  Action_005
Step_Bits.4:=0;
Step_Bits.5:=1;

Tran_005
Step_005.dn and not Step_Bits.4 and Step_Bits.5

Step_000
B1
```