

Introduction to IEC-61131

Automation Trends

- **Growing Demands on Control Systems**
 - **Applications Becoming More Sophisticated**
 - **Real-time Data Links to IT Systems**
 - **“Make to Order” Manufacturing Demands**
Process Synchronization
 - **Control to Optimize Processes**
Higher Throughput, Higher Quality, Save Energy
 - **Real-time Maintenance Information**
- **Collaborative Engineering**
Sharing Applications
Consistency

Control Engineering Objectives

- **Higher Quality**
- **Higher Efficiency**
- **Higher Productivity**
- **Greater Flexibility**

IEC-61131-3

Industrial Control Programming Standard

Common Programming

Look & Feel

*You might think of IEC-61131 as the “Basic”
of the control industry..*

IEC-61131 Languages

The screenshot displays the WAGO-IO-PRO 32 - Unit123.pro software interface. The main window shows a variable declaration list on the left and a ladder logic diagram in the center. The variable list includes:

- 0013 TimeP1: TIME := #5s;
- 0014 TimeP2: TIME := #10s;
- 0015 Step_12: BOOL;
- 0016 Override_12: BOOL;
- 0017 Unit_12: BOOL;
- 0018 Overload_12: BOOL;
- 0019 Divert_6: BOOL;
- 0020 Gatge_6: BOOL;
- 0021 r1: BOOL;
- 0022 Sinus: BOOL;
- 0023 run: BOOL;
- 0024 run_string: BOOL;
- 0025 updirection: BOOL;
- 0026 yVal: BOOL;
- 0027 END_VAR

The ladder logic diagram shows a sequence of steps: Init, Process_1, Process_2, Continue, Next_Batch, Ready, and back to Init. The Next_Batch step is currently active.

Three code windows are open:

- Action Process_2 (IL):**

```
0001 LD r1
0002 SIN
0003 MUL 1000
0004 ST Sinus
0005 LD r1
0006 MUL 200
```
- Action Next_Batch (FBD):** A function block diagram showing Proc_1_Done and Proc_2_Done connected to an AND gate, which is connected to NextBatch, which is connected to another AND gate, which is connected to Ready.
- Transition Ready (ST):**

```
0001 IF NOT run THEN
0002 RETURN;
0003 END_IF;
0004 run_string:= 'Stop';
0005
0006 IF updirection THEN
0007 IF yVal = -100 THEN
0008 updirection :=FALSE;
0009 THEN
```
- Action Process_1 (LD):**

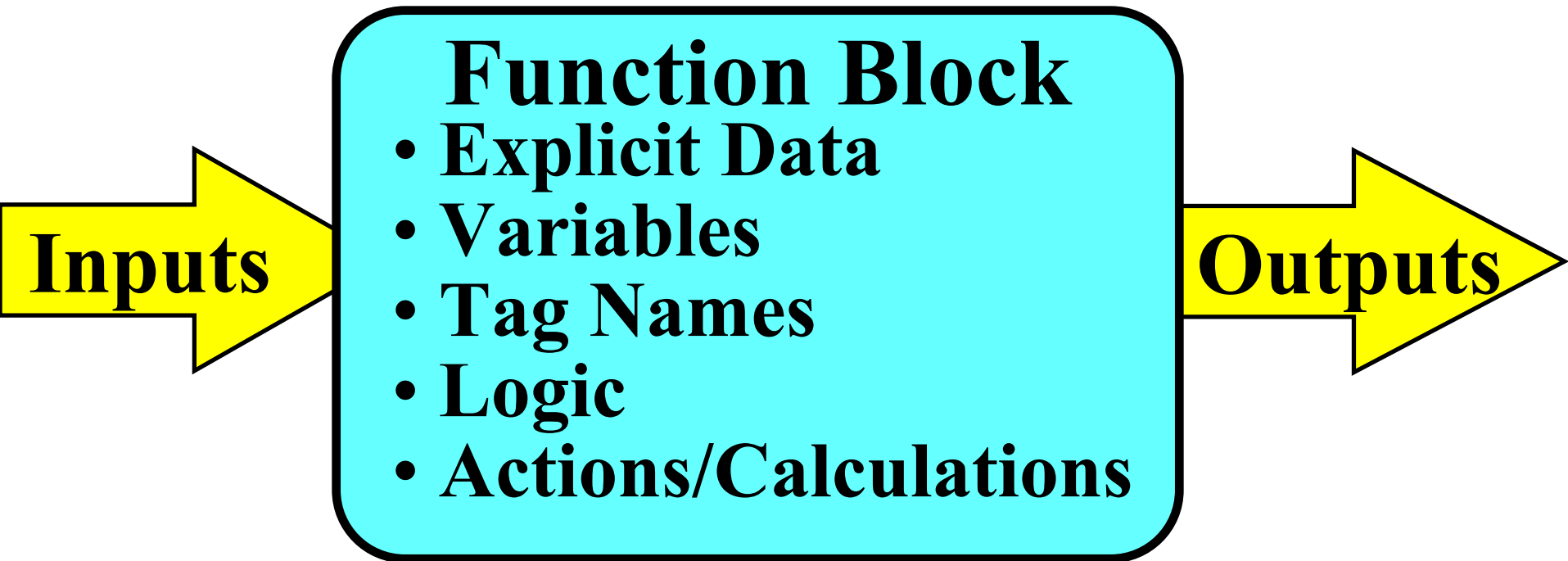
```
0001 TRUE StartProc_1
0002 Step_12 Unit_12
Override_12
Overload_12
```

The bottom status bar shows ONLINE, OV, and READ indicators, along with the time 3:31 PM.

Function Blocks

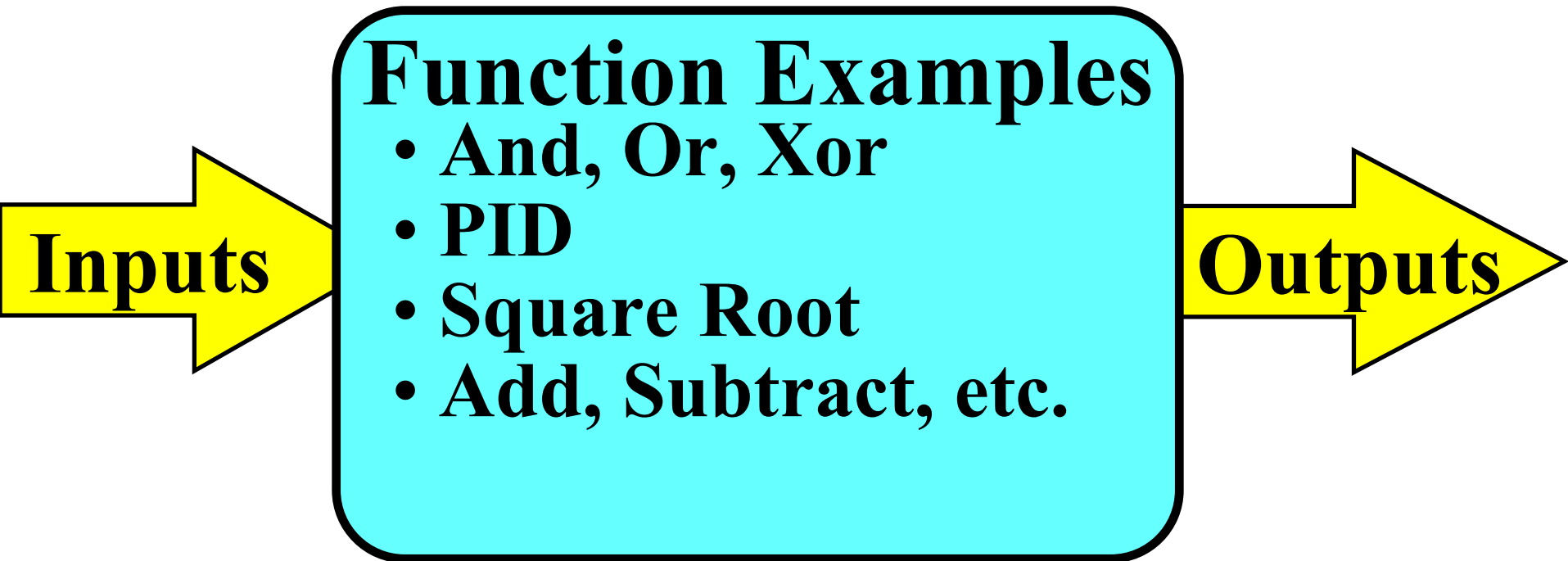
Function Block...

Self Contained Building Blocks



Function Block...

Self Contained Building Blocks



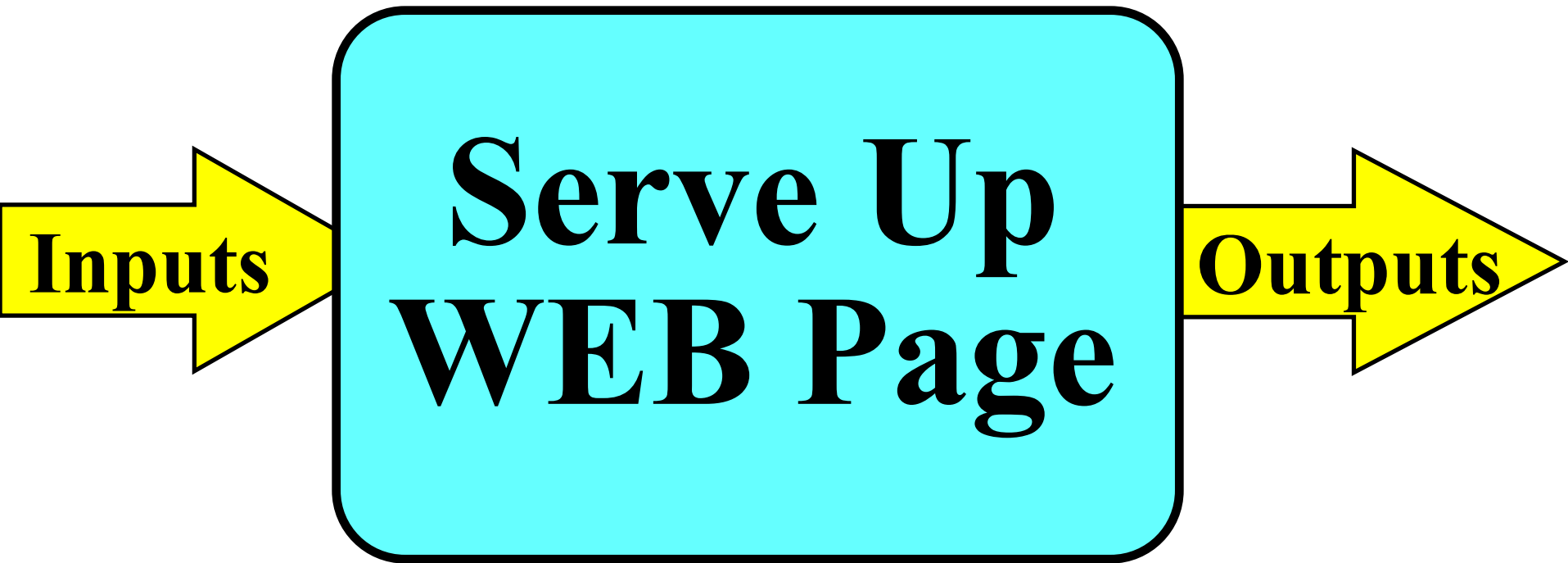
Function Block...

Self Contained Building Blocks



Function Block...

Self Contained Building Blocks



Function Block...

Self Contained Building Blocks

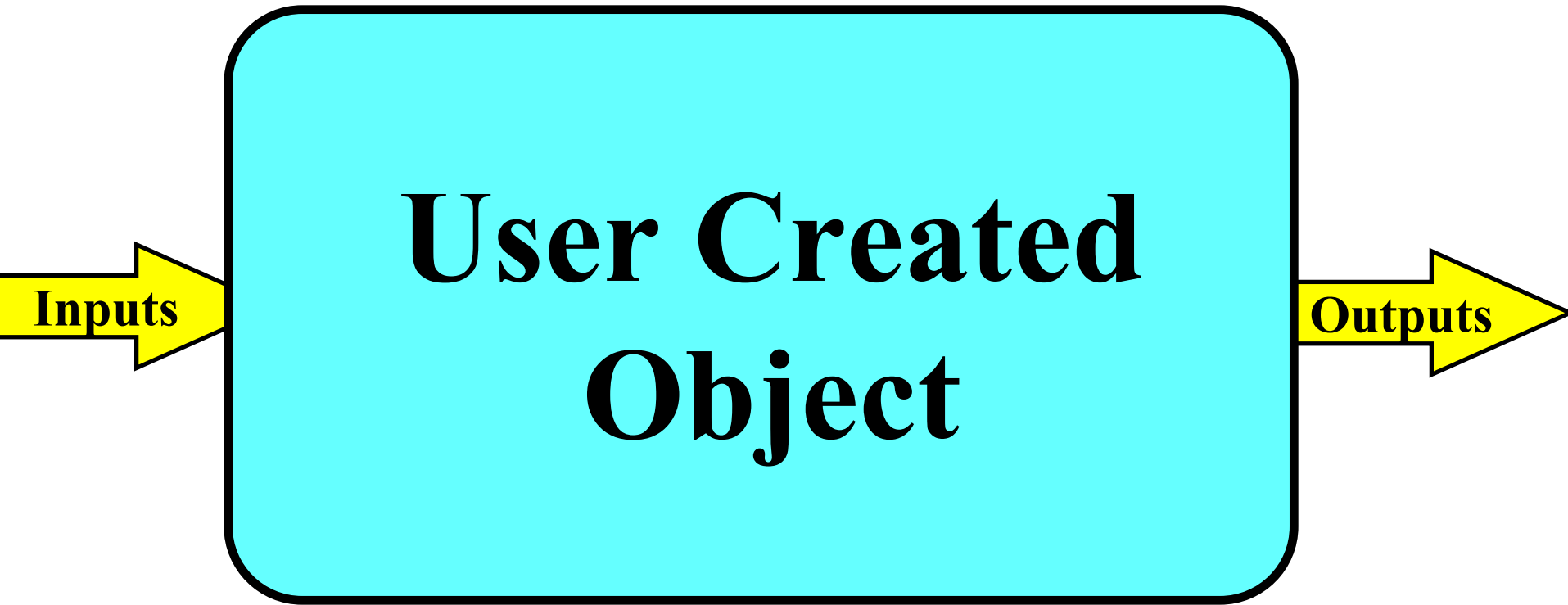


Function Block...

Self Contained Building Blocks

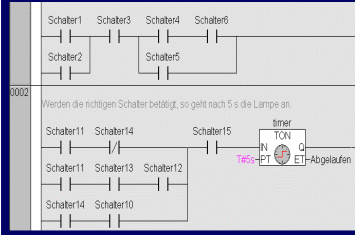


Encapsulation



Built from standard blocks ...

Ladder Logic



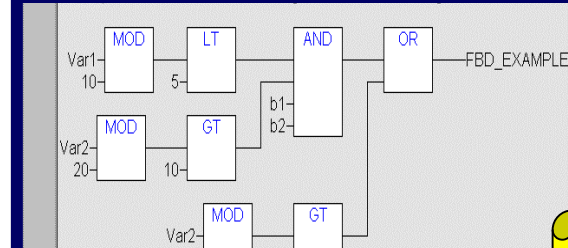
Instruction List

```
0003 SIN          1000
0004 MUL          1000
0005 ST          sinus
0006 (* Den Cosinus einer Zahl *)
0007 LD          r1
0008 COS
0009 MUL          1000
0010 ST          cosinus
0011
0012 (* Die Zahl weiterschalten *)
0013 LD          r1
0014 ADD          0.1
0015 ST          r1
```

Structured Text

```
0002 IF NOT run THEN
0003   RETURN;
0004 END_IF;
0005 run_string := 'Stop';
0006
0007 IF updirection THEN
0008   IF yVal = -100 THEN
0009     updirection := FALSE;
0010   IF xVal = 0 THEN
0011     rightdirection := TRUE;
0012   ELSE
0013     ...
0014 END
0015 ELSE
0016   yVal
```

Function Block



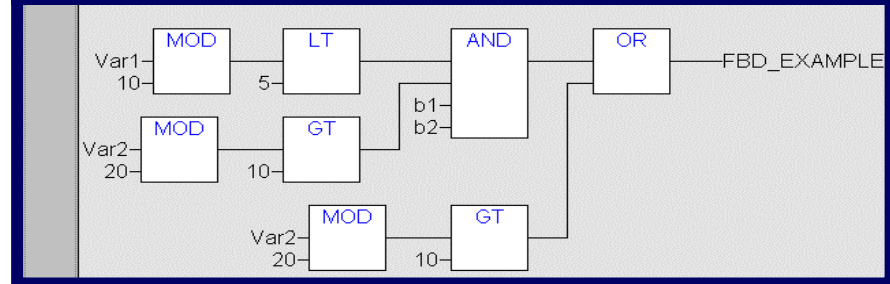
Build your own function blocks with Ladder Logic, Instruction List, or Structured Text

Create Your Own Standards

Reuse Tested Code
Easily Understood
Self Documenting
Password Protect



Function Block



IEC-61131-3 Editors

IDE

Integrated Design Environment

IEC-61131-3 Editors

IDE

(Integrated Design Environment)

- **Process Design**
- **Logic Simulation**
- **Automatic Documentation**
- **Online Debug Environment**

IDE

Integrated Design Environment

Editor

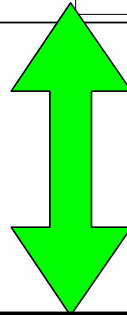
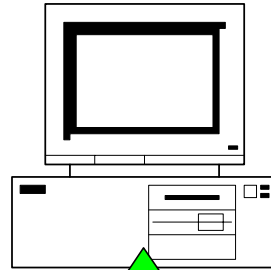
Debug Tools

Breakpoints

Watch Windows

Strip Chart Recorder

Integrated HMI



IEC-61131 Controller
Standard Function Blocks
User Created Function Blocks

IDE

Define Application



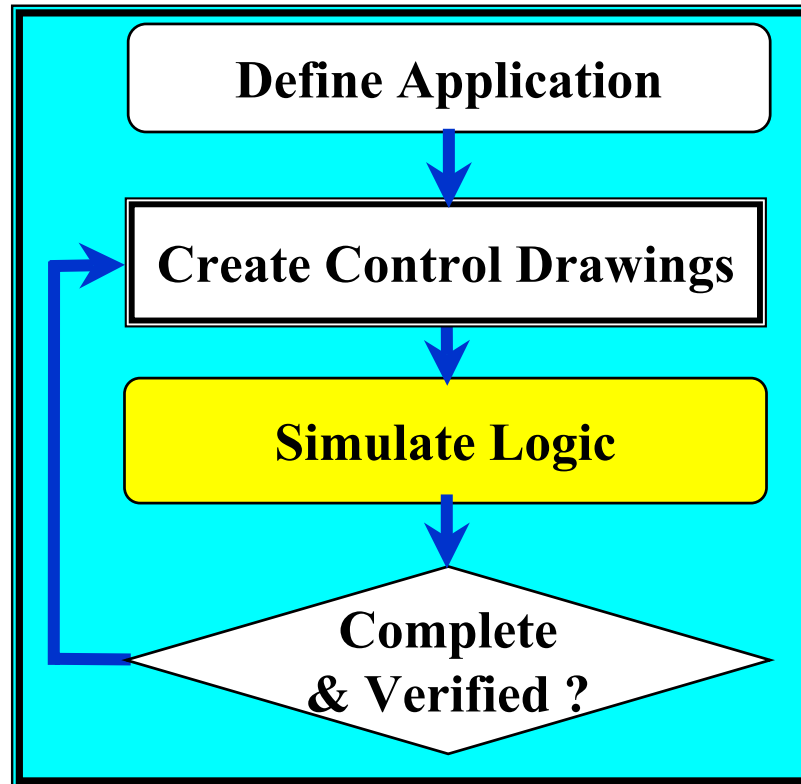
Create Control Drawings



Simulate Logic

**Immediate
Test & Verification**

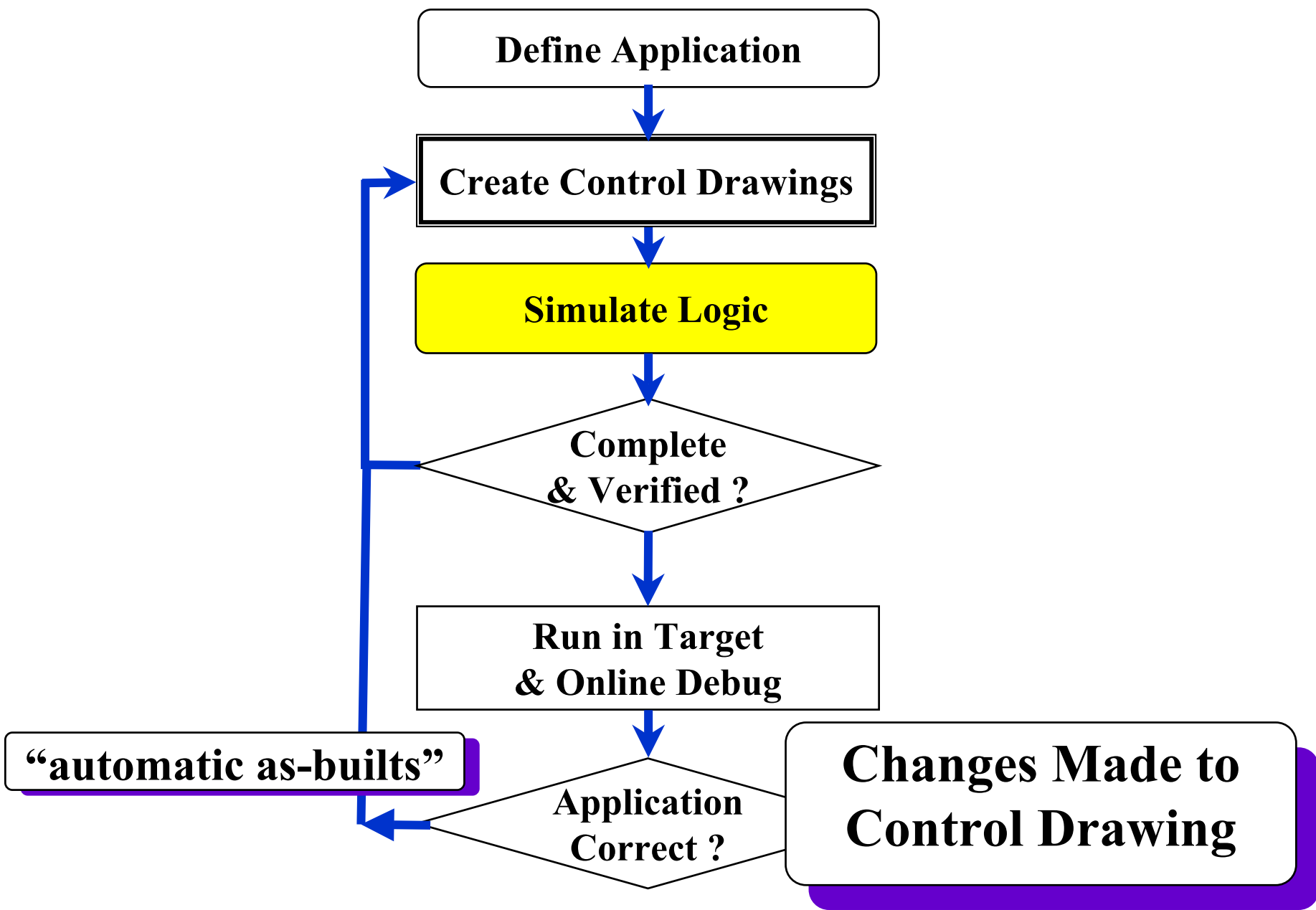
**Tested
Reusable
Functions**



Portable Applications

- Reusable Program Logic
- Reusable User Created Objects
- Symbolic Tags
- Promotes Standardization

**Changes
Documented
Automatically**



IEC-61131 Advantages

- ◆ Significantly Better Price/Performance**
- ◆ Leverages Existing Staff**
- ◆ Lower Implementation Cost**
- ◆ Lower Maintenance Cost**
- ◆ Easily Scaled to Needs**
- ◆ Flexibility for Productivity Enhancements**
- ◆ Interfaces to Information & Business Systems**

Resulting in ...

- **Improved Productivity**
- **Improved Efficiency**
- **Improved Quality**
- **Improved Performance**
- **Protected Investment**

Positive Investment Decision

IEC-61131 Standard

So What Who Cares?

- **Training**
- **Collaborative Engineering**
- **Quality**
- **Software Maintenance Costs**

PLCopen

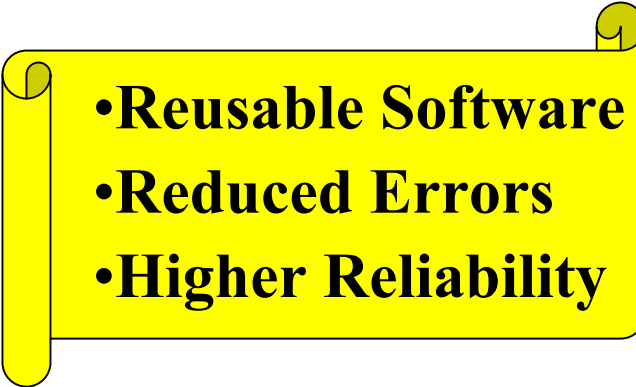
So What Who Cares?

- **Open Architecture Advocate**
- **Training**
- **Forum for Users**
- **Extending Standard -User Needs**

PLCopen

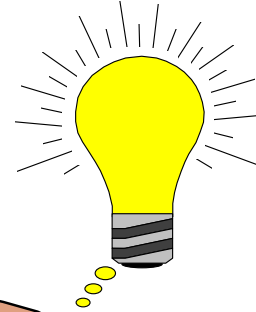
IEC 61131-3 Global Standard

Industrial Control Programming

- **Standardizes Programming Interface**
 - **Standardizes Implementation**
 - **Standardizes Specification**
 - **Standardizes Maintenance**
 - **Standardizes Design**
 - **Standardizes Training**
- 
- **Reusable Software**
 - **Reduced Errors**
 - **Higher Reliability**

Control Engineers...Adding Value

Challenge is to creatively
apply technology to gain
competitive advantage.



**IEC-61131
Technology
Provides the tools...**