IEC 61131-3 Basics and PLCopen

European Panasonic Headquarter, Germany

created by
Ralf Wohlschlaeger
General Manager Factory Automation
Panasonic Electric Works (Europe) AG

Chairman of PLCopen PC1 committee (promotion)
Contents:

- What is IEC 61131-3?
  - History
  - Advantages
  - Explanation

- What is PLCopen?
  - Organisation
  - Current topics
The Way to IEC61131-3 Programming

NEMA Programmable Controllers Committee formed (USA)
GRAFCET (France)
DIN 40719, Function Charts (Germany)
NEMA ICS-3-304, Programmable Controllers (USA)
IEC SC65A/WG6 formed
DIN 19 239, Programmable Controller (Germany)
IEC 65A(Sec)38, Programmable Controllers
MIL-STD-1815 Ada (USA)
IEC SC65A(Sec)49, PC Languages
IEC 848, Function Charts
IEC SC65A(Sec)67
IEC 64A(Sec)90
IEC 1131-3
IEC 61131-3
Type 3 report recommendation
IEC 61131-3
name change
IEC 61131-3
second edition

Source: Dr. J. Christensen (-1995) / R. Wohlschlaeger (-2003)
### Style of Software

<table>
<thead>
<tr>
<th>Conventional styled software</th>
<th>IEC 61131-3 styled software</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direct hardware address:</td>
<td>IEC address:</td>
</tr>
<tr>
<td>- X0, X1...Y0,Y1....DT0,DT1.....</td>
<td>- %IX0.....%QX0.....%MW5.0</td>
</tr>
<tr>
<td>1 Program from start to end</td>
<td>- each Variable have a name</td>
</tr>
<tr>
<td></td>
<td>- each Variable have a data type</td>
</tr>
<tr>
<td></td>
<td>- global and local Variables</td>
</tr>
</tbody>
</table>

Panasonic Control FPWIN Pro can use both styles.

Both styles can be mixed.
IEC 61131-3
An internationally accepted standard

- Unified rules in systems worldwide, reduces misunderstandings and shortens training
- Reuse of ready-made Functions and Function Blocks, saves time for programming and debugging
- Better overview through structure and modularity
- Fewer errors through defined data types and encapsulation
- Safe investment due to standardisation
Examples of IEC 61131-3 advantages

**Variables**:
- better documentation --> programming by names / symbols
- I/O connection list already stored in the project
- Base for the re-use of software

**POUs**:
- structured programming
- well defined interface --> other variables can be used in other projects
- re-use of Function Blocks saves time and debugging

**SFC**:
- flowchart on the monitor
- divide big programs into small and easy parts
- top down development / bottom up --> well structured
- different languages in the program
- easy debugging and error locating - only the current step is active

**General**:
- save training time for programmers
- enables parallel software development by more programmers
- certification ensures users to protect their investments for the future
Use Variable Names instead of Addresses
Conventional programming requires different functions for e.g.:

- **16-bit**
  - F22_ADD2
  - EN
  - s1
  - s2

- **32-bit**
  - F23_ADD2
  - EN
  - s1
  - s2

- **4-digit BCD data**
  - F42_ADD2
  - EN
  - s1
  - s2

- **8-digit BCD data**
  - F43_ADD2
  - EN
  - s1
  - s2

- **Floating point data**
  - F310_FADD
  - EN
  - s1
  - s2

**Flexible IEC instructions:**
1 function instead of several

Input data must be of the same data type!
The IEC 61131 Standard - The PLC Standard

Part 1  General overview, definitions
Part 2  Hardware
  - I/O signals, safety requirements, environment
Part 3  Programming Languages
Part 4  User Guidelines
Part 5  Communication
Part 6  Reserved
Part 7  Fuzzy control
Part 8  Technical Report

International Standard
IEC 61131-3 Software Model

Configuration A

Resource L

Task1

Program P1

Task2

Program P2

Task3

Program P3

Task4

Program P4

FB1

FB2

FB3

FB4

global and direct addressed variables

access paths

Task association

Access path association
IEC 61131-3: The 5 Programming Languages and The Common Elements

- Character set (English........)
- Data types (BOOL, WORD, INTEGER.................)
- Variables (VAR, VAR_input, VAR_output........)
- POUs, Program Organisation Units (Function, Function Block...)
- SFC Elements (Steps, Transitions..............................)
- Configuration elements: (Tasks)

• Basis for software re-use
### IEC 61131-3 Elementary Data Types

<table>
<thead>
<tr>
<th>No.</th>
<th>Keyword</th>
<th>Data Type</th>
<th>Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>BOOL</td>
<td>Boolean</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>SINT</td>
<td>Short integer</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>INT</td>
<td>Integer</td>
<td>16</td>
</tr>
<tr>
<td>4</td>
<td>DINT</td>
<td>Double integer</td>
<td>32</td>
</tr>
<tr>
<td>5</td>
<td>LINT</td>
<td>Long integer</td>
<td>64</td>
</tr>
<tr>
<td>6</td>
<td>USINT</td>
<td>Unsigned short integer</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>UINT</td>
<td>Unsigned integer</td>
<td>16</td>
</tr>
<tr>
<td>8</td>
<td>UDINT</td>
<td>Unsigned double integer</td>
<td>32</td>
</tr>
<tr>
<td>9</td>
<td>ULINT</td>
<td>Unsigned long integer</td>
<td>64</td>
</tr>
<tr>
<td>10</td>
<td>REAL</td>
<td>Real numbers</td>
<td>32</td>
</tr>
<tr>
<td>11</td>
<td>LREAL</td>
<td>Long reals</td>
<td>64</td>
</tr>
<tr>
<td>12</td>
<td>TIME</td>
<td>Duration</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>DATE</td>
<td>Date (only)</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>TIME_OF_DAY</td>
<td>Time of day (only)</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>DATE_AND_TIME</td>
<td>Date and time of day</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>STRING</td>
<td>Character string</td>
<td>8</td>
</tr>
<tr>
<td>17</td>
<td>BYTE</td>
<td>Bit string of length 8</td>
<td>8</td>
</tr>
<tr>
<td>18</td>
<td>WORD</td>
<td>Bit string of length 16</td>
<td>16</td>
</tr>
<tr>
<td>19</td>
<td>DWORD</td>
<td>Bit string of length 32</td>
<td>32</td>
</tr>
<tr>
<td>20</td>
<td>LWORD</td>
<td>Bit string of length 64</td>
<td>64</td>
</tr>
</tbody>
</table>
The 5 Languages of IEC 61131-3

**Instruction List**
- LD A
- ANDN B
- ST C

**Structured Text**
```
C := A AND NOT B
```

**Sequential Function Chart**
- **Step 1**: N FILL
- **Transition 1**
- **Step 2**: S Empty
- **Transition 2**
- **Step 3**

**Function Block Diagram**
```
AND
A  B  C
```

**Ladder Diagram**
```
A   B                    C
-| |--
|/\-----------------( )
```
POU = Program Organization Unit

- A POU consists of a header (variable declaration) and the body (instructions)
- POUs enable the re-use of software from macro level (Programs) to micro level (FB and Functions)

<table>
<thead>
<tr>
<th>POU Type</th>
<th>Replicated as:</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program</td>
<td>Program instance</td>
<td>Main program</td>
</tr>
<tr>
<td>Function Block</td>
<td>FB instance</td>
<td>Subroutine with own memory, several inputs and outputs possible</td>
</tr>
<tr>
<td>Function</td>
<td>Function</td>
<td>Subroutine without own memory</td>
</tr>
</tbody>
</table>
Function Blocks can be easily reused

1. Function Block

2. Variable Interface

<table>
<thead>
<tr>
<th>FU-Block Header</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Class</strong></td>
</tr>
<tr>
<td>0 VAR_INPUT</td>
</tr>
<tr>
<td>1 VAR_INPUT</td>
</tr>
<tr>
<td>2 VAR_INPUT</td>
</tr>
<tr>
<td>3 VAR_OUTPUT</td>
</tr>
</tbody>
</table>

3. Program body

Program once
reuse always

LD IN_1
AND IN_2
OR OUT_1
AND IN_3
ST OUT_1
Easy Programming of FBs and FUN
Easy Programming of FBs and FUN

Define inputs and outputs

Program FB contents

<table>
<thead>
<tr>
<th>Class</th>
<th>Identifier</th>
<th>Type</th>
<th>Initial</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>VAR_INPUT</td>
<td>In1</td>
<td>BOOL</td>
<td>FALSE</td>
</tr>
<tr>
<td>1</td>
<td>VAR_INPUT</td>
<td>In2</td>
<td>BOOL</td>
<td>FALSE</td>
</tr>
<tr>
<td>2</td>
<td>VAR_INPUT</td>
<td>In3</td>
<td>BOOL</td>
<td>FALSE</td>
</tr>
<tr>
<td>3</td>
<td>VAR_OUTPUT</td>
<td>Out1</td>
<td>BOOL</td>
<td>FALSE</td>
</tr>
</tbody>
</table>
Library Concept

Libraries:

- IEC_Standard_Lib
- Vendor_Lib
- Pulsed_Lib
- Communication_Lib
- PID_Lib
- Special_Project_Lib

Special_Lib

- TWO_TRIP
- POSITION_2_AX
- POSITION_3_AX
- WAIT_10s
- ELEVAT_4_FL
  ...

- Self-created FBs can be stored in libraries
- Comfortable structuring and sorting in the libraries
- Know-how protection of FBs and libraries
- Easy reuse of tested software --> saves time
PLCopen is a World-wide association

Main Office in Europe

Office in North America

Office in Japan
PLCopen was founded on June 15, 1992 in Giessen, Germany. Target was to promote IEC 61131-3, inform customers and give more weight to the IEC 61131-3 standard.
PLCopen *Mission*

We want to be the leading association resolving topics related to control programming to support the use of international standards in this field.
PLCopen is a World-wide association

> 80 members (June 2004)

from 19 countries all over the world

Suppliers, institutes and users

See newsletter / website for up-to-date list
The Essence of Compliance

- The IEC 61131 standard gives rules for compliance
- Certification guides users towards real IEC 61131-3 programming systems (e.g. PLCopen certified products)

Without testing there is no standard

Meanwhile only truly compliant IEC 61131-3 systems are promoted as IEC 61131-3 products
TC3: PLCopen Compliance Levels

Conformity Level & Reusability Level
Re-usability of Function (Block) libraries

Base Level
Portability of minimal systems
Certification

Certified products can use these logos

<table>
<thead>
<tr>
<th>Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Base Level:</strong></td>
<td>first step into IEC 61131-3 software</td>
</tr>
<tr>
<td><strong>Conformity Level:</strong></td>
<td>conforms to IEC 61131-3 based on supported data types</td>
</tr>
<tr>
<td><strong>Reusability Level:</strong></td>
<td>reuse of IEC 61131-3 Function Blocks based on supported data types</td>
</tr>
<tr>
<td><strong>Motion Control:</strong></td>
<td>certified Function Blocks according the Motion Control specification</td>
</tr>
<tr>
<td><strong>XML:</strong></td>
<td>Opening up the development environments by specifying XML formats for IEC 61131-3</td>
</tr>
</tbody>
</table>
Function Block exchange in ST language is possible:
- if both systems have the Reusability Level ST
- the used instructions are IEC 61131-3 instructions
- the same data types are available
Revolutionizing the industry with a global standard

Mechanics do not help anymore, a standard with software is possible

Reduce maintainance and sanitation

Less hardware parts, more software

The solution is........

........Software
Motion Control Standardization means:

- Hardware independent Software Development
- Consistent Development Environment
- Consistent Installation and Maintenance Interface

Same ‘Look and Feel’

IEC 61131-3 is a good base
The PLCopen Task Force Motion Control

Initiated by Users
..to fulfil their requirements

Goal:
To harmonize the access for Motion Control across different platforms
during development, installation and maintenance
based on the IEC 61131-3 environment

Users:
Bosch Packaging
Kuka
Kloekner Tevopharm
Focke EKB
Hershey Foods
Tetra Pak

Suppliers:
Siemens
Elau
Beckhoff Industrial Electronics
SEW Eurodrive
Mitsubishi Electric Europe
Cross Hueller
Lenze
Parker Hannifin
ISG Stuttgart
Control Techniques
Phoenix Contact
Keba
KW Software
Rockwell Automation
Nyquist
Baumueller
infoteam Software
Rexroth Indramat
HW Independence via Function Blocks

Software View
Encapsulation / Information Hiding

Hardware View

Inputs
Name
Outputs

I/F

Sercos Drive
Motor

E

I/F

PWM
Drive
Motor

E
Example of a Function Block

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>AXIS_REF</td>
<td>BOOL</td>
</tr>
<tr>
<td>Execute</td>
<td>BOOL</td>
</tr>
<tr>
<td>Position</td>
<td>REAL</td>
</tr>
<tr>
<td>Velocity</td>
<td>REAL</td>
</tr>
<tr>
<td>Acceleration</td>
<td>REAL</td>
</tr>
<tr>
<td>Deceleration</td>
<td>REAL</td>
</tr>
<tr>
<td>Jerk</td>
<td>REAL</td>
</tr>
<tr>
<td>Direction</td>
<td>MC_Direction</td>
</tr>
<tr>
<td>CommandAborted</td>
<td>BOOL</td>
</tr>
<tr>
<td>Error</td>
<td>BOOL</td>
</tr>
<tr>
<td>ErrorID</td>
<td>WORD</td>
</tr>
<tr>
<td>Done</td>
<td>BOOL</td>
</tr>
<tr>
<td>Axis</td>
<td>REAL</td>
</tr>
<tr>
<td>ErrorID</td>
<td>WORD</td>
</tr>
</tbody>
</table>

**FB-Name** | **MC_MoveAbsolute**

This function block commands a controlled motion at a specified absolute position.
Thank you !