



*Technology Training that Works*

---

# Practical Industrial Programming using 61131-3 for PLCs

---

## Contents

---

<b>1</b>	<b>An introduction - IEC 1131-3 on PLC programming</b>	<b>1</b>
1.1	Development & growth of Programmable Controllers	2
1.2	Need for standardization in programming approach	3
1.3	Drawbacks in conventional programming methodology	4
1.4	Features of IEC-1131-3 language definition	10
1.5	Summary	11
<b>2</b>	<b>PLC software architecture</b>	<b>12</b>
2.1	Software quality attributes	12
2.2	IEC software architecture	13
2.3	Component parts of IEC Software architecture	14
2.3.1	Configuration	14
2.3.2	Resource	15
2.3.3	Task	15
2.3.4	Program	15
2.4	Functions and function blocks	16
2.5	Local and global variables	17
2.5.1	Directly represented variable	17
2.5.2	Access paths	18
2.5.3	Execution control	18
2.6	Mapping software model to real life systems-Examples	18
2.7	Applications	20
2.7.1	Program Organization Unit (POU)	21
2.7.2	Hierarchical design	22
2.7.3	Communications	22
2.8	Conclusion	24
<b>3</b>	<b>Common elements in IEC-1131-3</b>	<b>25</b>
3.1	Common elements	25
3.1.1	Character set	26
3.1.2	Identifiers	26
3.1.3	Keywords	26
3.1.4	Comments	27



*Technology Training that Works*

<b>3.2</b>	<b>Elementary data types</b>	<b>27</b>
3.2.1	Integer	27
3.2.2	Real or floating point	28
3.2.3	Time or duration	28
3.2.4	Date and time data	29
3.2.5	String	30
3.2.6	Bit string	30
<b>3.3</b>	<b>Generic data type</b>	<b>31</b>
3.3.1	Initial values	31
<b>3.4</b>	<b>Derived data types</b>	<b>31</b>
3.4.1	Derived directly from elementary type	31
3.4.2	Structured data type	32
3.4.3	Enumerated data type	32
3.4.4	Sub range data types	32
3.4.5	Array data types	33
3.4.6	Default initial values of derived data type	33
<b>3.5</b>	<b>Variables</b>	<b>34</b>
3.5.1	Internal variable	34
3.5.2	Input variables	35
3.5.3	Output variables	35
3.5.4	Input/output variables	35
3.5.5	Global variable and external variable	36
3.5.6	Temporary variables	36
3.5.7	Directly represented variable	36
3.5.8	Access variables	37
3.5.9	Variable attributes	37
<b>3.6</b>	<b>Variable initialization</b>	<b>38</b>
<b>3.7</b>	<b>Functions</b>	<b>38</b>
3.7.1	Execution control	41
3.7.2	Function blocks	42
<b>3.8</b>	<b>Programs</b>	<b>44</b>
<b>3.9</b>	<b>Resource</b>	<b>44</b>
<b>3.10</b>	<b>Tasks</b>	<b>44</b>
3.10.1	Non-preemptive scheduling	45
3.10.2	Preemptive scheduling	45
3.10.3	Task assignment	46
3.10.4	Configuration	47
<b>3.11</b>	<b>Summary</b>	<b>47</b>
<b>4</b>	<b>Structured text</b>	<b>48</b>
<b>4.1</b>	<b>Introduction</b>	<b>48</b>
<b>4.2</b>	<b>Statements used for assignments</b>	<b>49</b>
<b>4.3</b>	<b>Expressions</b>	<b>49</b>
<b>4.4</b>	<b>Evaluating an expression</b>	<b>50</b>
<b>4.5</b>	<b>Statements</b>	<b>51</b>
4.5.1	Function block calls	51



*Technology Training that Works*

<b>4.6</b>	<b>Conditional statements</b>	<b>52</b>
4.6.1	IF...THEN...ELSE	52
4.6.2	CASE statement	53
<b>4.7</b>	<b>Iteration statements</b>	<b>53</b>
4.7.1	FOR ... DO	53
4.7.2	WHILE ... DO	54
4.7.3	REPEAT ... UNTIL	54
4.7.4	EXIT	54
4.7.5	RETURN	55
<b>4.8</b>	<b>Implementation dependence</b>	<b>55</b>
<b>4.9</b>	<b>Summary</b>	<b>55</b>
<hr/>		
<b>5</b>	<b>Function block diagram</b>	<b>57</b>
<hr/>		
<b>5.1</b>	<b>Introduction</b>	<b>57</b>
<b>5.2</b>	<b>Basics</b>	<b>58</b>
<b>5.3</b>	<b>Methodology</b>	<b>58</b>
<b>5.4</b>	<b>Signal flow</b>	<b>60</b>
<b>5.5</b>	<b>Feedback path</b>	<b>61</b>
<b>5.6</b>	<b>Network layout</b>	<b>61</b>
<b>5.7</b>	<b>Function execution control</b>	<b>62</b>
<b>5.8</b>	<b>Jumps and labels</b>	<b>62</b>
<b>5.9</b>	<b>Network evaluation rules</b>	<b>63</b>
<b>5.10</b>	<b>Summary</b>	<b>64</b>
<hr/>		
<b>6</b>	<b>Ladder diagrams</b>	<b>65</b>
<hr/>		
<b>6.1</b>	<b>Introduction</b>	<b>65</b>
<b>6.2</b>	<b>Basic concept</b>	<b>66</b>
<b>6.3</b>	<b>Graphical symbols used in ladder diagram</b>	<b>66</b>
<b>6.4</b>	<b>Boolean expressions using ladder diagrams</b>	<b>69</b>
<b>6.5</b>	<b>Integrating functions and function blocks within ladder diagrams</b>	<b>70</b>
<b>6.6</b>	<b>Feedback paths</b>	<b>71</b>
<b>6.7</b>	<b>Jumps and labels</b>	<b>72</b>
<b>6.8</b>	<b>Network evaluation rules</b>	<b>72</b>
<b>6.9</b>	<b>Portability</b>	<b>73</b>
<b>6.10</b>	<b>Summary</b>	<b>74</b>
<hr/>		
<b>7</b>	<b>Instruction list</b>	<b>75</b>
<hr/>		
<b>7.1</b>	<b>Introduction</b>	<b>75</b>
<b>7.2</b>	<b>Structure of IL programming language</b>	<b>75</b>
7.2.1	Basics	75



*Technology Training that Works*

7.2.2	Instruction structure	76
7.2.3	Comparison with Structured Text (ST) language	77
7.2.4	General semantics of IL expressions	77
7.2.5	Modifiers for deferred execution	78
7.2.6	Other modifiers	79
<b>7.3</b>	<b>Calling functions and function blocks</b>	<b>80</b>
7.3.1	Function block - Formal call with an input list	80
7.3.2	Function block - Informal call	80
7.3.3	Function block - Call with an input list	81
7.3.4	Calling a function - Formal call	81
7.3.5	Calling a function - Informal call	81
<b>7.4</b>	<b>Portability and other issues</b>	<b>82</b>
<b>7.5</b>	<b>Summary</b>	<b>83</b>
<hr/>		
<b>8</b>	<b>Sequential Function Chart (SFC)</b>	<b>84</b>
<hr/>		
<b>8.1</b>	<b>Introduction to the basic concept of SFC</b>	<b>84</b>
8.1.1	Structure of SFC	85
<b>8.2</b>	<b>Steps</b>	<b>89</b>
8.2.1	Initial step	89
8.2.2	Normal step	89
<b>8.3</b>	<b>Transitions</b>	<b>91</b>
<b>8.4</b>	<b>Actions</b>	<b>92</b>
<b>8.5</b>	<b>Action qualifiers</b>	<b>95</b>
<b>8.6</b>	<b>Action control function block</b>	<b>98</b>
<b>8.7</b>	<b>Execution rules</b>	<b>98</b>
<b>8.8</b>	<b>Design safety issues</b>	<b>99</b>
<b>8.9</b>	<b>Top down design</b>	<b>100</b>
<b>8.10</b>	<b>Summary</b>	<b>101</b>