

IEC 61499 FUNCTION BLOCKS FOR EMBEDDED
AND DISTRIBUTED CONTROL SYSTEMS DESIGN
Second Edition

Valeriy Vyatkin
Auckland University, New Zealand



Copyright 2012 by ISA—International Society of Automation

67 Alexander Drive
P.O. Box 12277
Research Triangle Park, NC 27709

All rights reserved.

Printed in the United States of America.

10 9 8 7 6 5 4 3 2

ISBN: 978-1-936007-93-6

No part of this work may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior written permission of the publisher.

Notice

The information presented in this publication is for the general education of the reader. Because neither the author nor the publisher has any control over the use of the information by the reader, both the author and the publisher disclaim any and all liability of any kind arising out of such use. The reader is expected to exercise sound professional judgment in using any of the information presented in a particular application. Additionally, neither the author nor the publisher have investigated or considered the affect of any patents on the ability of the reader to use any of the information in a particular application. The reader is responsible for reviewing any possible patents that may effect any particular use of the information presented.

Any references to commercial products in the work are cited as examples only. Neither the author nor the publisher endorses any referenced commercial product. Any trademarks or tradenames referenced belong to the respective owner of the mark or name. Neither the author nor the publisher makes any representation regarding the availability of any referenced commercial product at any time. The manufacturer's instructions on use of any commercial product must be followed at all times, even if in conflict with the information in this publication.

Library of Congress Cataloging-in-Publication Data in Process

Contents

	The O ³ neida Publications Series	xi
	Acknowledgments	xiii
	Introduction	xv
	Introduction to the Second Edition	xvi
	How to use this book	xvi
	Book Web Site	xviii
1	Quick start.....	1
	System modelled with function blocks	1
	Function block internals	4
	Summary	5
	Review questions for Chapter 1	6
	Notes	6
2	Evolution of industrial automation technologies.....	7
	First generation	7
	Second generation	7
	Third generation	8
	Fourth generation	9
	Fifth generation	11
	Summary	11
	Review questions for Chapter 2	11
3	Automation: From mass production to flexibility.....	13
	Industrial trends	13
	Requirements for the new generation of automation	14
	Advantages and drawbacks of the PLC architecture	15
	Technical requirements for next-generation architectures	19
	Summary	19
	Review questions for Chapter 3	20
	Notes	20
4	Horizons of distributed intelligent automation.....	21
	Autonomous intelligent devices	21
	System modelling and simulation	21
	Service provision mechanism	23
	Intelligent integration	23
	Different hardware running same function blocks	23
	Key technologies provided by IEC 61499	26
	Main IEC 61499 benefits for PLC users	27
	Potential benefits of IEC 61499 for embedded control users	28
	Summary	28
	Review questions for Chapter 4	28
	Notes	28

5	Basic concepts of IEC 61499.....	29
	Events 29	
	Function blocks 29	
	Full definition of a function block type 31	
	Data types 32	
	Standard data types 32	
	Extended data types in FBDK/FBRT 34	
	ANY 34	
	COLOR 35	
	ARRAYS 35	
	MATRIX 35	
	Summary 36	
	Review questions for Chapter 5 36	
	Notes 36	
6	Function block development kit.....	37
	General information on FBDK 37	
	Installation of FBDK 38	
	Accessing documentation 39	
	Create a basic function block type 39	
	Compile a function block 42	
	Test a function block 43	
	Create and run a system configuration 43	
	Combine FBDK with JAVA development and debugging in Eclipse 43	
	Create a new project in Eclipse 44	
	Set a class to run 44	
	Run FBDK from Eclipse 45	
	Troubleshooting FBDK 46	
	Summary 47	
	Review questions for Chapter 6 47	
7	Basic function blocks.....	49
	Capsule for functions 49	
	Execution Control 49	
	Syntax of ECC transitions 50	
	ECC transition evaluation 52	
	How does a basic function block work? 53	
	Simple function blocks in FBDK 53	
	Standard libraries 55	
	Tutorial 1: Modifying an existing basic function block type 55	
	Summary 60	
	Review questions for Chapter 7 61	
	Notes 61	
8	Composite function blocks.....	63
	Connections 64	
	Execution Control 66	
	Operations with events 66	
	Tutorial 2: Create and test a composite function block type 67	
	Summary 67	
	Review questions for Chapter 8 69	

9	Applications and sub-applications.....	71
	Application 71	
	Sub-applications 72	
	Summary 73	
	Review questions for Chapter 9 73	
10	Models for devices and resources	75
	Device 75	
	Resource 76	
	Classes of devices 76	
	Device type definition 77	
	PC-based remote device 78	
	Embedded device 80	
	Configure Netmaster 80	
	The usual way of executing <i>Java</i> applications 81	
	TINI specific 81	
	Create a version of FBRT executable on TINI 82	
	Resource type definition 82	
	Device management 82	
	Summary 83	
	Review questions for Chapter 10 84	
	Notes 84	
11	Distributed system configurations.....	85
	System configuration 85	
	Tutorial 3: Using a modified function block type in a system configuration 91	
	Modifying an existing system configuration 91	
	Testing the composite function block USE_XYZ 94	
	Summary 94	
	Review questions for Chapter 11 95	
12	Service interface function blocks	97
	Services 97	
	Standard input and output names of service interface function block parameters 101	
	Event inputs 101	
	Event outputs 101	
	Data inputs 102	
	Data outputs 102	
	Communication function blocks 103	
	Communication function blocks of FBDK 103	
	Local communications 105	
	How to develop a service interface function block in FBDK 107	
	Process interface: read inputs and write outputs 108	
	Simulation of service interface function blocks 109	
	Cyclic scan data sampling 109	
	Example of device type definition for Netmaster 110	
	Summary 110	
	Exercises for Chapter 12 112	

13	Simple application with decentralized control	115
	Description of a controlled object	115
	Controller design	116
	Controller implementation	117
	System configuration	119
	Debugging	120
	Distribution	120
	Summary	123
	Review questions for Chapter 13	123
14	User interface function blocks	125
	Human-machine interface	125
	IN_CHOICE—drop-down list	125
	IN_MATRIX—input display for the matrix	126
	IN_TEXT—area for text input	126
	IN_BOOL—labelled checkbox to enter a Boolean value	126
	IN_EVENT—event modelled by button	127
	IN_ANY—text input field	127
	IN_ARRAY—input area for array	127
	IN_ENUM—drop-down list	128
	RADIO_BOOL—choice between two alternatives	128
	Data Output Function Blocks	128
	DIAG_LOG—Time-stamped log for diagnostic events	128
	OUT_BOOL—colored circle with a label	128
	OUT_ANY—textual display of any data type	129
	OUT_ARRAY—single-line text field with array value	129
	OUT_EVENT—event display	129
	OUT_MATRIX	130
	OUT_TEXT	130
	Input and output function block FB_SLIDER	130
	JAVA implementation details	130
	Flow Layout in PANEL_RESOURCE and VIEW_PANEL resource types	130
	VIEW_PANEL resource type	131
	Animation	131
	Drawbacks of the standard visualization blocks of FBDK	133
	VHMI Package of visualization function blocks	133
	Device type: ImageDev	133
	Resource type: ImageResource	133
	Render	134
	Render with rotation FB type—RenderRot	135
	IN_OUT_DISPLAY_BUTTON	135
	IN_PRESS_BUTTON	135
	Installation	136
	Example	137
	View implementation of a mechatronic device	137
	Summary	140
	Review questions for Chapter 14	140
15	Model/view/control design pattern	141
	Design methodology	141
	Sketch	141

	Building views	142
	Implementation of the multi-layer architecture	142
	Animation	144
	Models	146
	Control	148
	Diagnostics	149
	Distribution	150
	Physical design	152
	More MVC examples	153
	Summary	153
	Review questions for Chapter 15	153
16	Automation objects for efficient system integration	155
	Object-oriented design in industrial automation	155
	Drawbacks of MVC design	155
	Adapter interface function blocks	156
	The benefit of using adapters	156
	Function blocks in automation objects	158
	Reconfiguration	161
	Development process	163
	Develop interfaces	163
	Adapter wrappers	163
	Controllers	164
	Encapsulate process dependencies to function blocks	164
	Summary	165
	Review questions for Chapter 16	165
17	New business models.....	167
	Structure of the automation market	167
	Opportunities for automation technology vendors	169
	Automation users: Machine builders, system integrators, and production enterprises	169
	Knowledge economy in automation	170
	Repository of automation intellectual property	172
	Summary	172
	Review questions for Chapter 17	173
	Notes	173
18	Execution rules	175
	Introduction: Semantic problems of IEC 61499	175
	Semantics of a basic function block	176
	Order of transition evaluation	176
	Input event lifetime	178
	What to do with ‘non-eventful’ transitions?	179
	Emitting output events from function blocks	180
	Real-life illustrative example	180
	Illustrating semantic problems in the example	182
	Semantics of FB networks	183
	Proposal to fix the FB semantics	185
	Conclusion	186

19	Tools for function block design.....	189
	Overview of available tools and run-time platforms	189
	ISaGRAF	189
	Overview	189
	Reference example	190
	Comparing Execution Model of ISaGRAF with FBDK	192
	Implementation of LedChaser in FBDK	195
	Distributed systems	196
	Pilot case studies	197
	Hardware platforms	200
	Problems of ISaGRAF Semantics	201
	ISaGRAF: Summary	204
	NxtStudio	204
	Overview of the tool	206
	Support of MVC design pattern	206
	Distribution	208
	Execution	208
	Pilot applications	210
	4DIAC IDE and Forte	211
	FBench	212
	CORFU Engineering Support System	214
20	Conclusion.....	217
A	Event function blocks	219
B	IEC 61499–compliance profile for feasibility demonstrations	225
	Part 4. Configurability agreement	225
	4.1 Software tools	225
	4.2 Device management services	225
	4.3 Devices	226
	4.4 FBMGT Document Type Definition (DTD)	226
	4.5 Request/Response semantics	227
CD	Proposed corrections to IEC 61499	239
	Overview of corrections	239
	Temporary variables in algorithms	239
	Syntax and semantics of ECC	239
	Use of adapters	240
	Network Segments	241
	Interoperation with programmable controllers (Annex D6)	241
	D.6.1 Introduction	241
	D.6.2 Service conventions	241
	D.6.3 Function block types	242
	Bibliography	249
	Index	255

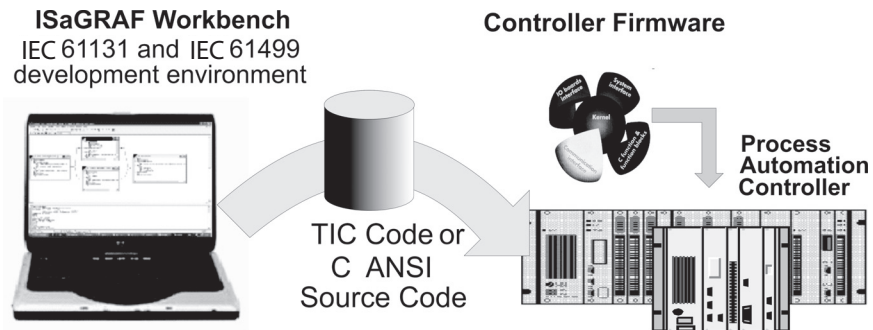


Figure 19-1 Structure of ISaGRAF: Workbench and Firmware for IEC 61499 and IEC 61131-3 system development.

At the same time, not all concepts of IEC 61499 have been implemented. The communication between the application parts is achieved through network variables instead of service interface FBs. The appearance of FBs in ISaGRAF follows IEC 61499, but their internals look a bit different. The Execution Control Chart of a basic FB, for example, is defined in a way similar to the IEC 61131-3 Sequential Function Chart language. The event-driven IEC 61499 FBs are executed on top of a cyclic-scanned and time-triggered IEC 61131-3 runtime system.

ISaGRAF allows the use of the traditional PLC programming languages of the IEC 61131-3 standard in function blocks; for example, it allows ladder logic, which is shown in Figure 19-2.

Reference example

The ‘LED Chaser’ reference example provided with the ISaGRAF demo kit (Figure 19-3) will be used to illustrate differences between this implementation and its FBDK interpretation. The demo kit consists of 3 identical microprocessor devices with some input/output ports connected with 2 buttons, 2 switches and four light emitting diodes (LED).

The ‘LedChaser’ application works as follows. On start up, the light begins ‘running’ from the leftmost LED (LED3) to the right. The direction changes when the light reaches the rightmost LED, and so on. Pressing the B1 button on either device increases the speed of running, and pressing B2 decreases it. At any given time there is only one lit lamp in all three devices. An FB network controller for this system in the ISaGRAF version of IEC 61499 function blocks is shown in Figure 19-4. The control is decentralised; pressing a button in any device will immediately impact on the speed of the run, even when the light is already ‘running’ in another device.

If you press a button on device 3 while the light is in device 1, for example, the event will go to the PeriodManagement block, and then a new period will be transmitted to LedSequence1. As a result, the speed (of the light movement in device 1) will change instantaneously.

The program includes six instances of the ‘ButtonManagement’ FB, one per button. This FB detects a depressed button, and generates an event signal, which is sent to the ‘PeriodCalc’ FB, and that generates the value of time interval between the illuminations of two neighboring LEDs. This value is passed to the FBs responsible for sending signals to the lamps within one device. These are the three instances of the ‘LedSequence’ FB type. This FB has two input events: ‘Inc’ and ‘Dec’. When Inc is received, the FB sequentially displays all 5 possible patterns, in binary coding: ‘0000’, ‘1000’, ‘0100’, ‘0010’, ‘0001’ and ‘0000’, delaying the duration between each change of the output pattern.

The ‘LedSequence’ is a basic function block, whose logic is defined by means of an Execution Control Chart (ECC). In ISaGRAF, the ECC is implemented using a Sequential Function Chart (SFC) programming language. Apparently, ISaGRAF developers wanted to re-use the existing SFC editor, on account of control engineers’ familiarity with this language.

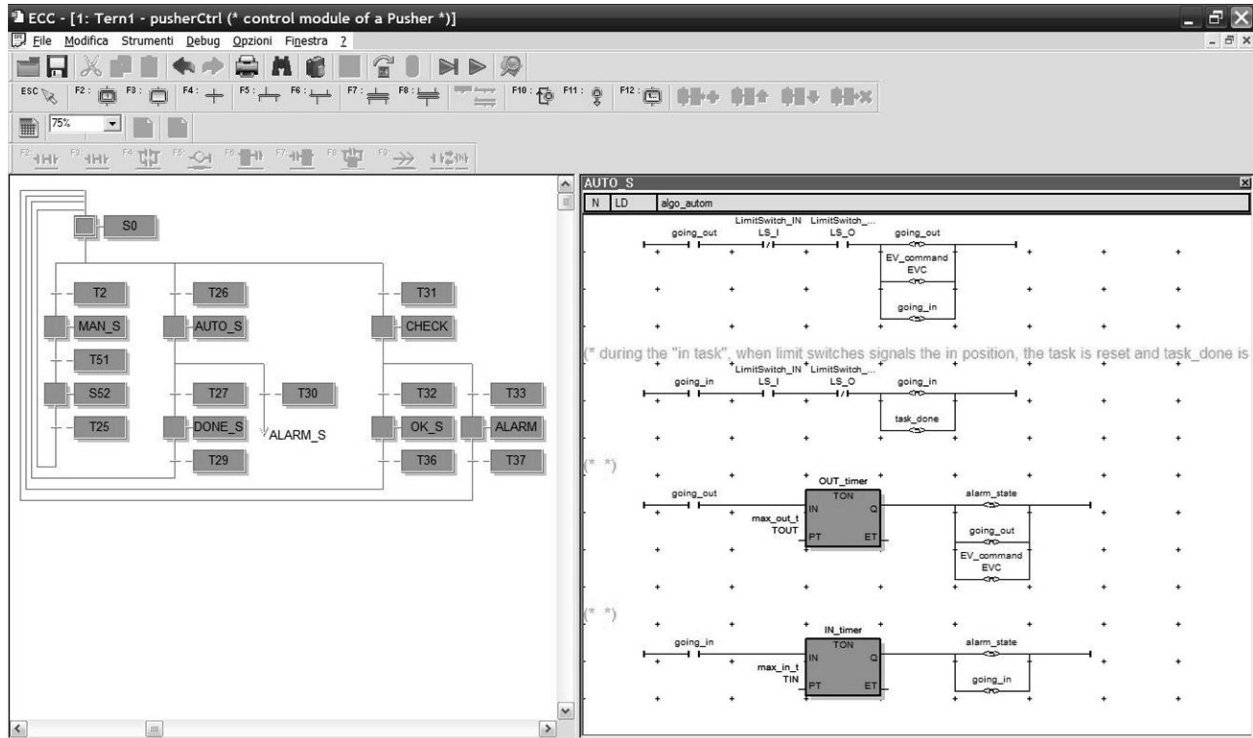


Figure 19-2 ISaGRAF workbench enables the use of Ladder Logic to program algorithms in basic function blocks.

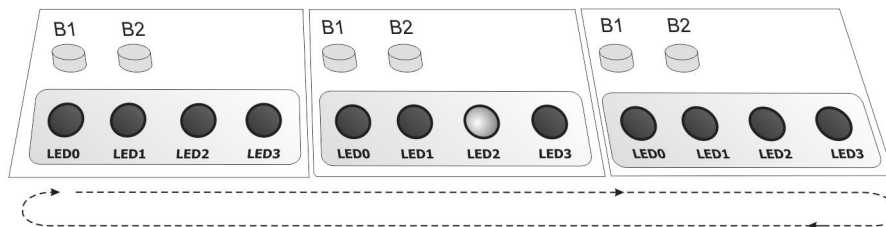


Figure 19-3 LED Chaser system.

Despite the similar look, the ECC semantics is different from pure SFC semantics in IEC 61131-3. In the “pure” SFC, no more than one transition per scan can occur. After a transition, SFC is “put on hold”.

In IEC 61499, the ECC execution continues until there is no transition that evaluates to TRUE. If evaluation of a transition between steps ends in FALSE, the block’s execution in this scan stops. In the next scan, it will be resumed at the same place with a new evaluation of the transition. Consider, for example, the SFC of the ‘LedSequence’ FB, shown in Figure 19-5.

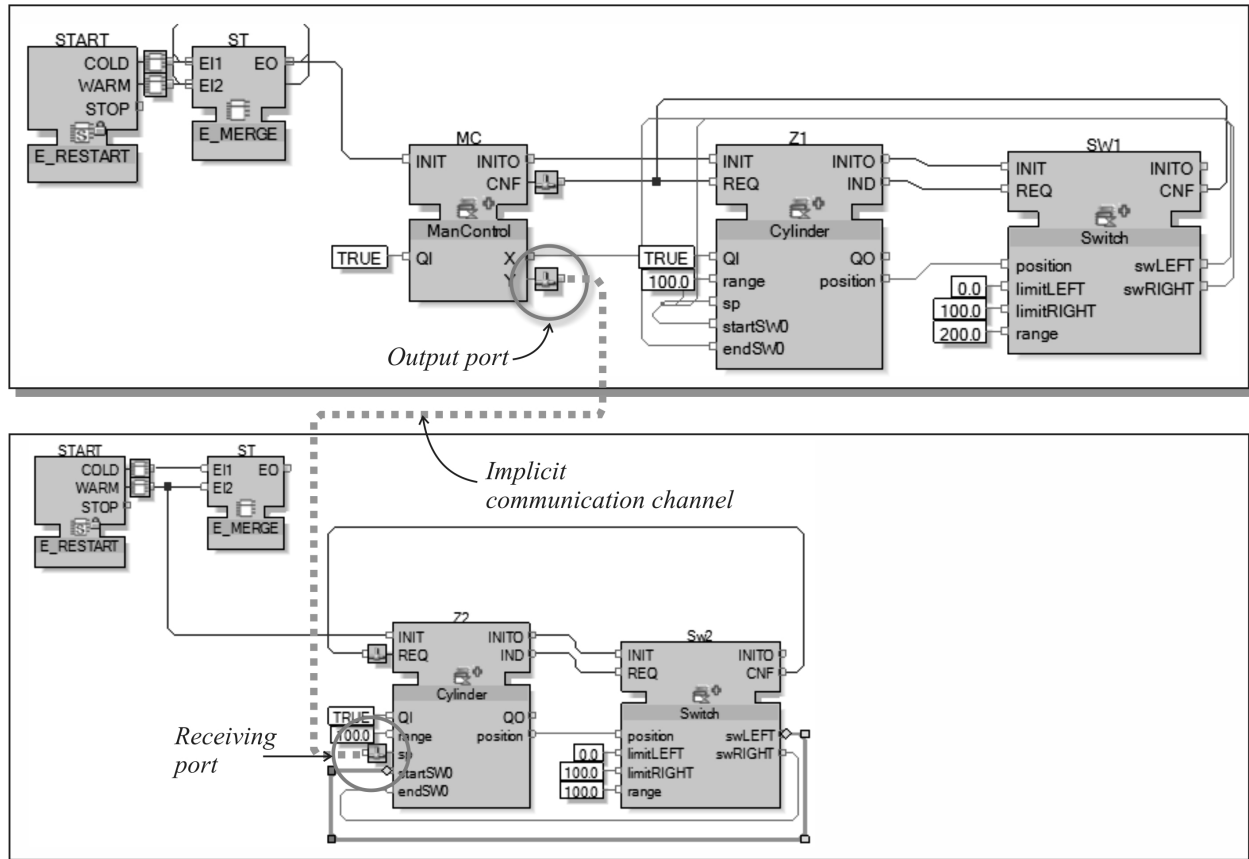


Figure 19-27 The “TwoCylinders” application distributed to two devices.

microprocessor code using a cross-compiler. It is linked with the run-time libraries responsible for scheduling events.

NxtStudio supports device class 2 specification, i.e., it allows uploading new function block types to remote devices. Thus, it extends the capabilities of FORTE, which (like FBRT) supports only class 1 devices.

One should note some difference in the ECC execution rules of FBDK and ISaGRAF. The differences are illustrated in the reference example in Figure 19-28.

All three outputs will be emitted simultaneously on arrival of the first event of the sequence, because the lifetime of events in NxtStudio differs from those in FBDK and ISaGRAF. An event is “alive” until some state has not been visited twice (to avoid loops), or until no more ECC transitions evaluate to TRUE.

Pilot applications

NxtControl has already applied its technology in a few dozen projects in the domain of building automation. The largest project is a training centre building with 19 control devices controlling about 2500 I/Os (heating, ventilation, air-condition, lighting, etc.) with IEC 61499.

The range of hardware platforms compliant with NxtStudio, includes industrial personal computers (IPC) of Siemens (MicroBox), Beckhoff (CX 1020), and WAGO (IPC 871). Some of these devices are shown in Figure 19-29. Like PLCs, these devices are hardened and certified for use in tough industrial environments.

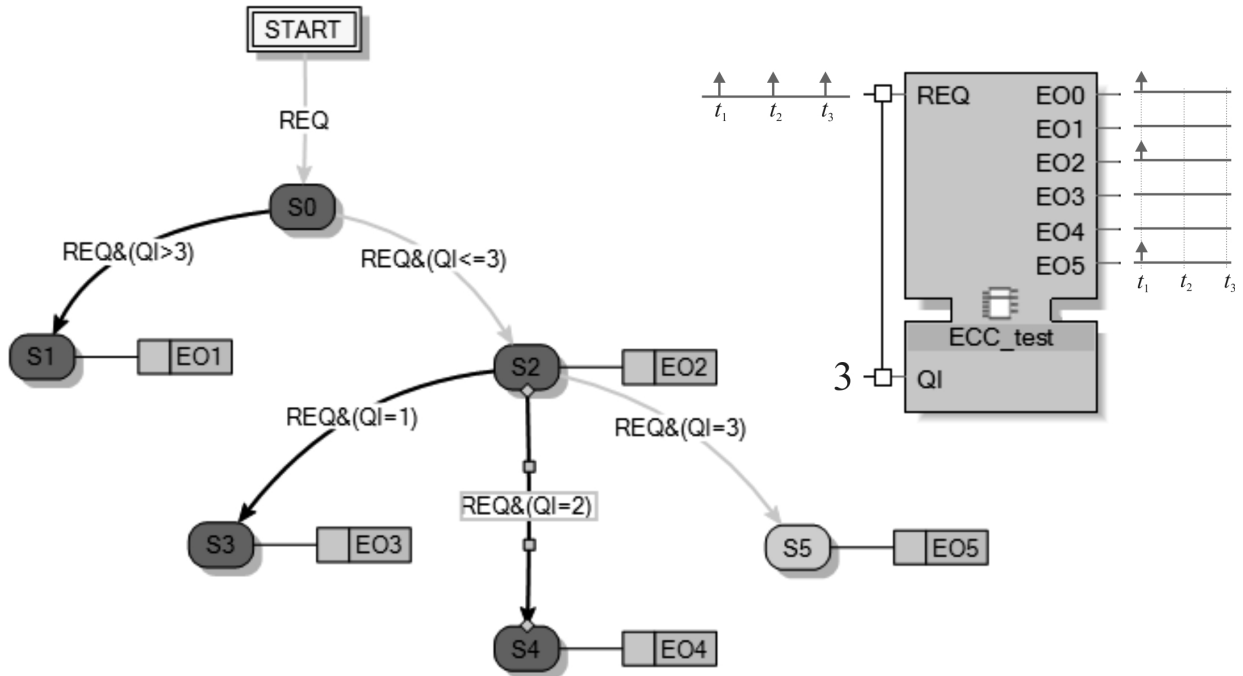


Figure 19-28 Execution of the benchmarking FB in NxtStudio.

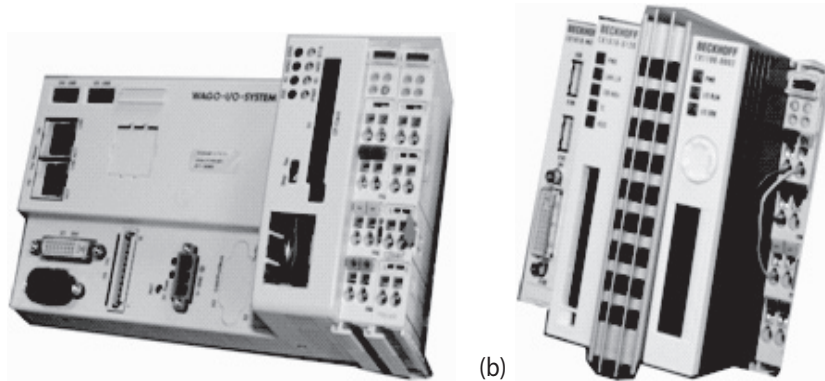


Figure 19-29 a) WAGO IPC 871; b) Beckhoff CX 1020.

Focusing on the European automation market, NxtControl has developed support for field area networks, such as Profibus and Ethernet. Through these interfaces, the compliant devices can communicate with a broad spectrum of automation devices, such as motion control devices.

NxtControl software is especially efficient in applications involving highly modular machines, such as material-handling systems. Figure 19-30 shows an example of a material-handling automation application implemented in NxtStudio. Observe the similarity between the structure of the physical part (right top) and the structure of the program (right bottom). The function blocks are CAT instances, implementing control and visualization of a conveyor group.

4DIAC IDE and Forte

The 4DIAC consortium was formed in 2007 by several academic and industrial organizations with the goal of delivering open source design tools and run-time platforms for IEC 61499 (4DIAC, 2009). The