

EPOS2 P

Positioning Controllers

Best Practice & Application Examples



Document ID: rel5877

PLEASE READ THIS FIRST

The present document represents a compilation of (hopefully) helpful “Good-to-Knows” that might come in handy in your daily work with EPOS2 P Positioning Controllers.

The individual chapters cover particular cases or scenarios and are intended to give you a hand for efficient setup and parameterization of your system.



We strongly stress the following facts:

- *The present document does not replace any other documentation covering the basic installation and/or parameterization described therein!*
 - *Also, any aspect in regard to health and safety, as well as to secure and safe operation are not covered in the present document – it is intended and must be understood as complimenting addition to those documents!*
-

TABLE OF CONTENTS

1	Introduction	5
1.1	Important Notice: Prerequisites for Permission to commence Installation. . . .	5
1.2	About this Document	5
1.2.1	Intended Purpose.	5
1.2.2	Target Audience.	5
1.2.3	Documentation Structure	6
1.2.4	How to use.	6
1.2.5	Symbols and Signs	7
1.3	Trademarks and Brand Names	8
2	Getting Started	9
2.1	Step 1: Open EPOS Studio Project	9
2.2	Step 2: Import Parameter	10
2.3	Step 3: Execute Startup Wizard.	11
2.4	Step 4: Execute Regulation Tuning	12
2.5	Step 5: Open IEC 61131 Program.	13
2.6	Step 6: Start IEC 61131 Program	15
2.7	How to use Example Program	17
2.7.1	How to watch Variables in the Source Code	17
2.7.2	How to watch State Changes.	17
2.7.3	How to trigger and acknowledge Errors	18
3	Best Practice Examples	19
3.1	«State Machine»	20
3.2	«Error Handling»	24
3.3	«Input/Output Handling»	28
3.4	«Homing»	32
3.5	«Positioning»	36
3.6	«Continuous Motion»	40
3.7	«Read Actual Value»	44
3.8	«Object Dictionary Access»	47
3.9	«Data Handling»	51
4	Application Examples	55
4.1	«Cyclic Motion»	56
4.2	«I/O Mode»	61
4.3	«Multiaxis Motion»	66
4.4	«Process Input Output»	71

••*page intentionally left blank*••

1 Introduction

1.1 Important Notice: Prerequisites for Permission to commence Installation

The EPOS2 P is considered as partly completed machinery according to EU directive 2006/42/EC, Article 2, Clause (g) and therefore **is only intended to be incorporated into or assembled with other machinery or other partly completed machinery or equipment.**



WARNING

Risk of Injury

Operating the device without the full compliance of the surrounding system with the EU directive 2006/42/EC may cause serious injuries!

- Do not operate the device, unless you have made sure that the other machinery fulfills the requirements stated in EU directive!
- Do not operate the device, unless the surrounding system fulfills all relevant health and safety aspects!
- Do not operate the device, unless all respective interfaces have been established and fulfill the stated requirements!

1.2 About this Document

1.2.1 Intended Purpose

The purpose of the present document is to provide you specific information to cover particular cases or scenarios that might come in handy during commissioning of your drive system. Each chapter covers a particular aspect of EPOS2 P programming. It may be part of a complete application (→“Application Examples” on page 4-55) or with a focus on a particular and single task (→“Best Practice Examples” on page 3-19). As an opening chapter (→as of page 2-9), you will find a generally applicable guide on how to handle the «EPOS Studio» during application programming.

Use for other and/or additional purposes is not permitted. maxon motor, the manufacturer of the equipment described, does not assume any liability for loss or damage that may arise from any other and/or additional use than the intended purpose.

Find the latest edition of the present document, as well as additional documentation and software to the EPOS2 P Positioning Controllers also on the internet: →www.maxonmotor.com

1.2.2 Target Audience

This document is meant for trained and skilled personnel working with the equipment described. It conveys information on how to understand and fulfill the respective work and duties.

This document is a reference book. It does require particular knowledge and expertise specific to the equipment described.

1.2.5 Symbols and Signs

Safety Alerts



Take note of when and why the alerts will be used and what the consequences are if you should fail to observe them!

Safety alerts are composed of...

- a signal word,
- a description of type and/or source of the danger,
- the consequence if the alert is being ignored, and
- explanations on how to avoid the hazard.

Following types will be used:

- 1) **DANGER**
Indicates an **imminently hazardous situation**. If not avoided, the situation **will** result in death or serious injury.
- 2) **WARNING**
Indicates a **potentially hazardous situation**. If not avoided, the situation **can** result in death or serious injury.
- 3) **CAUTION**
Indicates a **probable hazardous situation** and is also used to alert against unsafe practices. If not avoided, the situation **may** result in minor or moderate injury.

Example:



DANGER

High Voltage and/or Electrical Shock

Touching live wires causes death or serious injuries!

- Make sure that neither end of cable is connected to live power!
- Make sure that power source cannot be engaged while work is in process!
- Obey lock-out/tag-out procedures!
- Make sure to securely lock any power engaging equipment against unintentional engagement and tag with your name!

Prohibited Actions and Mandatory Actions

The signs define prohibitive actions. So, you **must not!**

Examples:



Do not touch!



Do not operate!

The signs point out actions to avoid a hazard. So, you **must!**

Examples:



Unplug!



Tag before work!

Informatory Signs



Requirement / Note / Remark

Indicates an action you must perform prior continuing or refers to information on a particular item.



Best Practice

Gives advice on the easiest and best way to proceed.



Material Damage

Points out information particular to potential damage of equipment.



Reference

Refers to particular information provided by other parties.

1.3 Trademarks and Brand Names

For easier legibility, registered brand names are listed below and will not be further tagged with their respective trademark. It must be understood that the brands (the below list is not necessarily concluding) are protected by copyright and/or other intellectual property rights even if their legal trademarks are omitted in the later course of this document.

Brand Name	Trademark Owner
CANopen® CiA®	© CiA CAN in Automation e.V, DE-Nuremberg
OpenPCS	© infoteam Software AG, DE-Bubenreuth
Windows®	© Microsoft Corporation, USA-Redmond, WA

Table 1-2 Brand Names and Trademark Owners

1.4 Copyright

© 2016 maxon motor. All rights reserved.

The present document – including all parts thereof – is protected by copyright. Any use (including reproduction, translation, microfilming and other means of electronic data processing) beyond the narrow restrictions of the copyright law without the prior approval of maxon motor ag, is not permitted and subject to persecution under the applicable law.

maxon motor ag

Brünigstrasse 220

P.O.Box 263

CH-6072 Sachseln

Switzerland

Phone +41 41 666 15 00

Fax +41 41 666 16 50

www.maxonmotor.com

2 Getting Started



Note

The following description will serve as a guide on how to handle the «EPOS Studio» during application programming. Handling and respective settings are shown by means of an exemplary program using the Best Practice Example “State Machine”. The in fact information visualized may slightly differ depending on your actual inputs.

2.1 Step 1: Open EPOS Studio Project

In this section you will open the EPOS Studio project file containing the system topology information.



Note

You will need the «EPOS Studio» Version 1.42 or higher to open the project files

- 1) Configure EPOS2 P hardware:
 - a) Switch off the EPOS2 P power supply.
 - b) Configure the EPOS2 P as Node 1 (CAN-ID Switch).
- 2) Load «EPOS Studio» project:

Doubleclick file «StateMachineProject.pjm» in your example directory.
- 3) Change communication properties:
 - a) Click «Communication» in the Page Navigator.

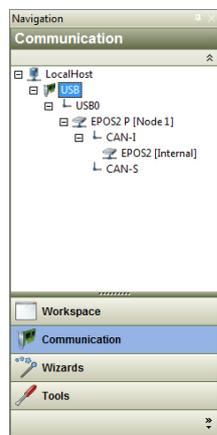


Figure 2-2 Page Navigator – Communication

- b) Click right on «USB», then click «Properties».
- c) Select the port you will be using.

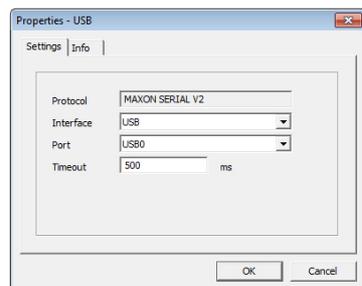


Figure 2-3 Communication Properties

- d) Click "OK" to close the dialog.
- 4) Establish the online communication to the device:
 - a) Switch on the EPOS2 P power supply.
 - b) Click "Connect All" button in the toolbar.



Figure 2-4 Connect All

- c) The following window will appear.

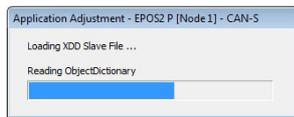


Figure 2-5 Application Adjustment

- d) Upon completion of the application adjustment, the EPOS2 P will be online.

2.2 Step 2: Import Parameter

In this section you will load the parameters to all devices in the project. The parameters are predefined for this example and available as XDC or DCF file for each device.

- 1) Import parameters to EPOS2 P [Node 1]:
 - a) Click "Wizards" in the Page Navigator.
 - b) Select "EPOS2 P [Node 1]" in the Device Selection Combo Box.
 - c) Doubleclick "Parameter Export/Import" in the Wizard Tree.

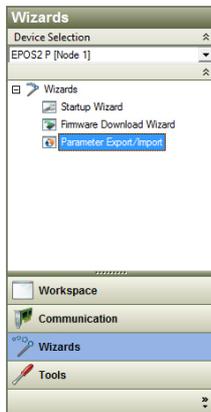


Figure 2-6 Page Navigator – Parameter Export/Import

- d) Select «EPOS2 P [Node 1]» and «EPOS2 [Internal]» parameter file, then click «Import Parameters from files».

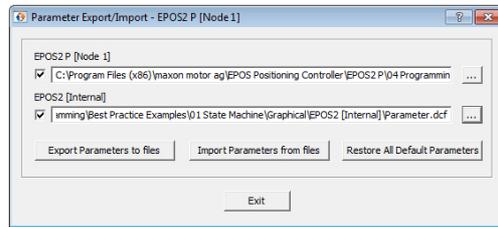


Figure 2-7 Parameter Export/Import

- e) Upon successful parameter import, click «Exit».

2.3 Step 3: Execute Startup Wizard

In this section you will execute the startup wizard. The startup wizard will help you to change the pre-defined motor and sensor parameters to your specific hardware configuration.



Note

For details → «EPOS2 P 24/5 Getting Started», chapter «Installation and Configuration».

- 1) Initiate Startup Wizard of EPOS2 P [Node 1]:
 - a) Click «Wizards» in the Page Navigator.
 - b) Select «EPOS2 P [Node 1]» from the Device Selection Combo Box.
 - c) Doubleclick «Startup Wizard» in the Wizard Tree.

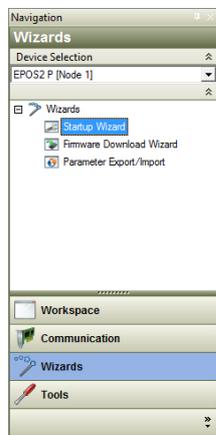


Figure 2-8 Page Navigator – Startup Wizard

- 2) Execute Startup Wizard of EPOS2 P [Node 1]:
Follow the instruction of the startup wizard to configure the EPOS2 P drive system.

2.4 Step 4: Execute Regulation Tuning

In this section you will execute «Auto Tuning» for all devices. Regulation tuning will help you to change the predefined regulation parameters to your specific load.



Note

For details → «EPOS2 P 24/5 Getting Started», chapter “Regulation Gains Tuning”.

- 1) Start Regulation Tuning of EPOS2 [Internal]:
 - a) Click «Wizards» in the Page Navigator.
 - b) Select «EPOS2 [Internal]» from the Device Selection Combo Box.
 - c) Doubleclick «Regulation Tuning» in the Wizard Tree.

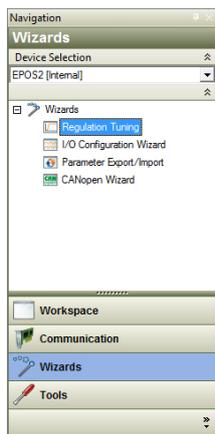


Figure 2-9 Page Navigator – Regulation Tuning

- 2) Execute «Auto Tuning» of EPOS2 [Internal].

2.5 Step 5: Open IEC 61131 Program

In this section you will open the IEC 61131 example program.

- 1) Open "IEC-61131 Programming":
 - a) Click "Tools" in the Page Navigator.
 - b) Select "EPOS2 P [Node 1]" from the Device Selection Combo Box.
 - c) Doubleclick "IEC-61131 Programming" in the Tools Tree.

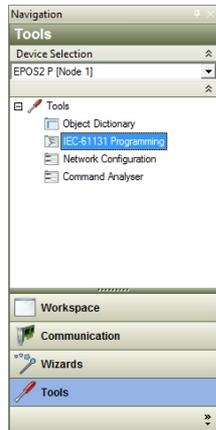


Figure 2-10 Page Navigator – IEC-61131 Programming

- 2) Open IEC-61131 Example Program:
 - a) Click "Browse Project" to open the IEC-61131 project file.

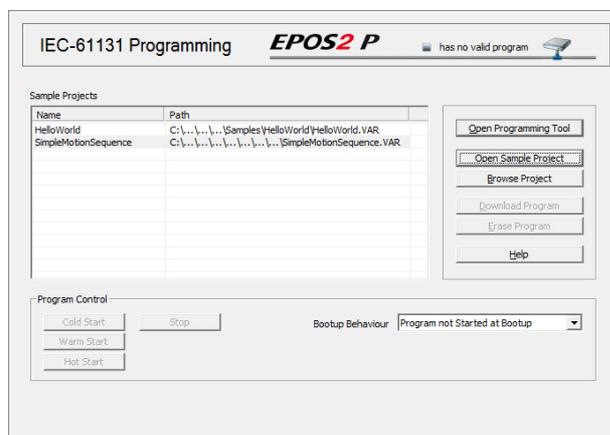


Figure 2-11 IEC-61131 Programming Tool

Getting Started

Step 5: Open IEC 61131 Program

- b) Select OpenPCS project file (*.VAR) from working directory "...EPOS2 P [Node 1]IEC 61131 Program\StateMachineProject.VAR", then click «Open».

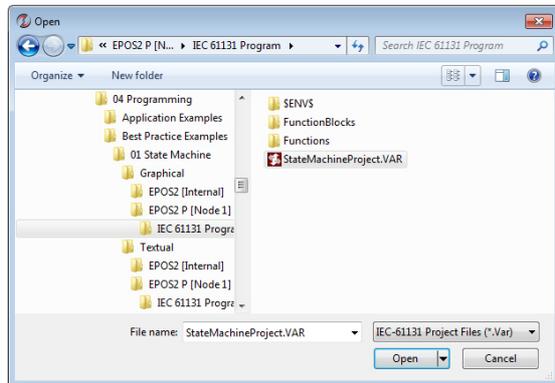


Figure 2-12 Browse IEC-61131 Project

- c) The programming environment «OpenPCS» will be opened.

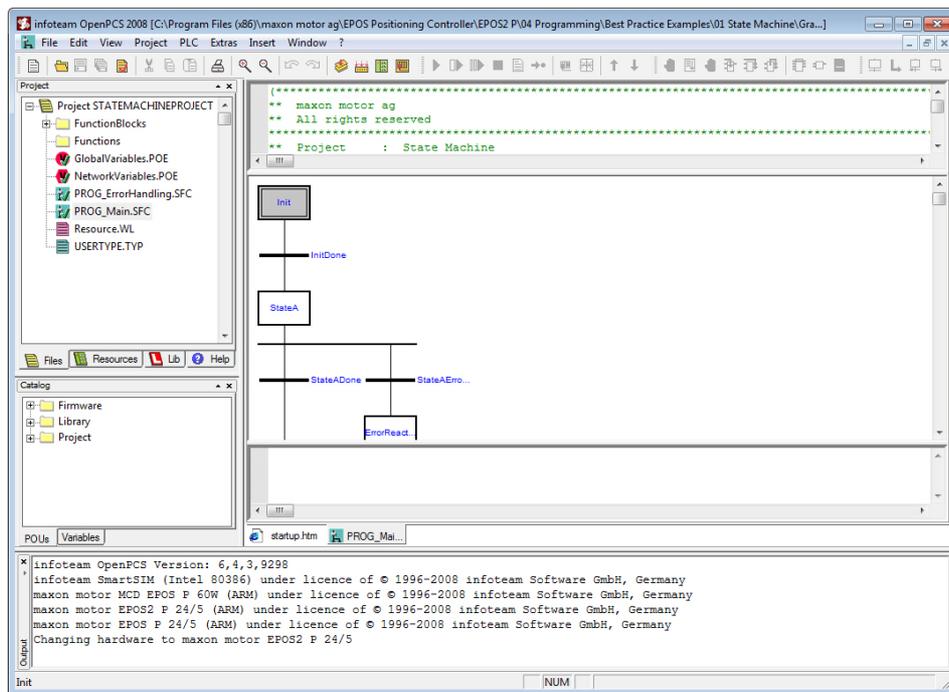


Figure 2-13 IEC-61131 Project

2.6 Step 6: Start IEC 61131 Program

In this section you will compile and start the IEC 61131 example program.

- 1) Edit Connection Settings:
 - a) Select **PLC** menu, then click **Connections**.
 - b) Select **ProxyEpos2_USB**, then click **Edit**.

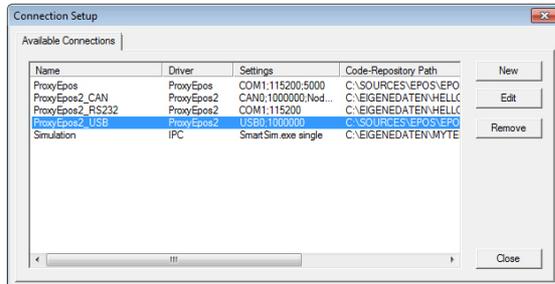


Figure 2-14 Connection Setup

- c) Click **Settings** to change connection settings.
- d) Select the used communication settings that you are using.

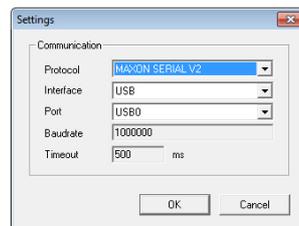
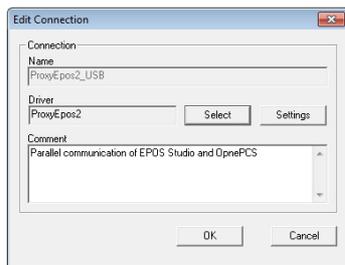


Figure 2-15 Edit Connection

- e) Close all dialogs by clicking **OK**.
- 2) Compile Program:
Click **Build Active Resource** button in the toolbar.



Figure 2-16 Build Active Resource

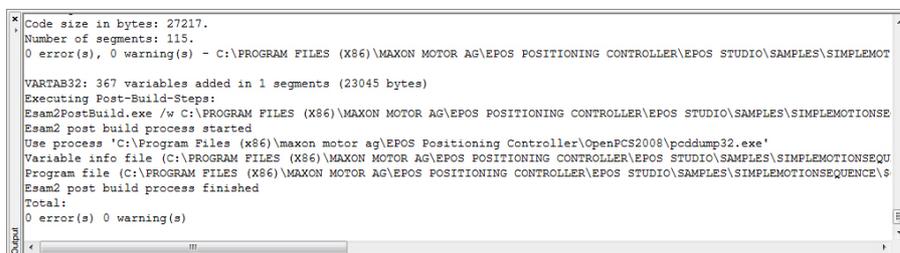


Figure 2-17 Build Result

3) Download Program:

- a) Click "Go Online/Offline" button in the toolbar.



Figure 2-18 Go Online

- b) Click "Yes" to download the program to the EPOS2 P.

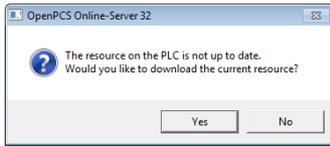


Figure 2-19 Download Program

4) Click "Coldstart" button in the toolbar to start the program:



Figure 2-20 Coldstart

2.7 How to use Example Program

2.7.1 How to watch Variables in the Source Code

- 1) Open Source Code:
 - a) Doubleclick a file in the project explorer (e.g. PROG_MAIN.ST).

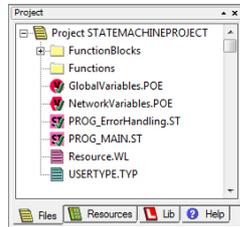


Figure 2-21 Project Explorer

- b) Click «Monitor/Edit» button in the toolbar to switch the source code window into debugging mode.



Figure 2-22 Monitor/Edit

- c) Move the cursor over a variable in the source code to see the actual value.

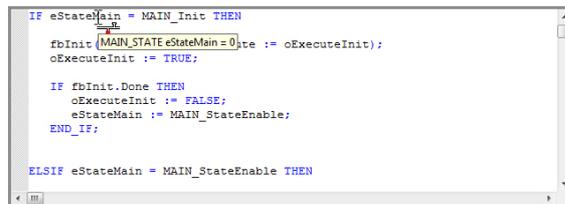


Figure 2-23 Watch Variable in Source Code

2.7.2 How to watch State Changes

- 1) Select «Resources» tab in the project explorer.

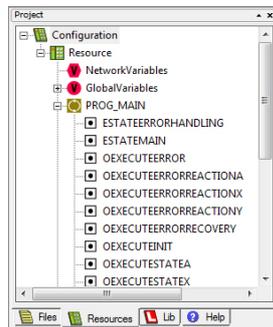


Figure 2-24 Project Explorer Resources

- 2) Add the following variables to the watch window. Do so by clicking right on the items in the “Resources” window, then click “Add To Watchlist”.

Instancepath	Name	Value	Type	Address	Force	Comment
PROG_MAIN.FBSTATEY	DONE	TRUE	BOOL			
PROG_MAIN.FBSTATEY.FBTIMER	ET	10s0ms	TIME			
PROG_MAIN.FBSTATEX	DONE	FALSE	BOOL			
PROG_MAIN.FBSTATEX.FBTIMER	ET	2s318ms	TIME			
PROG_MAIN.FBSTATEA	DONE	TRUE	BOOL			
PROG_MAIN.FBSTATEA.FBTIMER	ET	10s0ms	TIME			
PROG_MAIN.FBINIT	DONE	TRUE	BOOL			
PROG_MAIN.FBINIT.FBTIMER	ET	10s0ms	TIME			
PROG_MAIN	ESTATEMAIN	3	ENUM			
PROG_ERRORHANDLING	ESTATEERRORHANDLING	0	ENUM			

Figure 2-25 Watch Variables for State Changes

- 3) Watch State Changes:
 - a) Check on the state variables to see the active state and on the timer values to see when a transition is triggered.
 - b) Check on the digital output 1 to see state changes between State X and State Y.

2.7.3 How to trigger and acknowledge Errors

- 1) Select “Resources” tab in the project explorer (→ Figure 2-24).
- 2) Add the following variables to the watch window. Do so by clicking right on the items in the “Resources” window, then click “Add To Watchlist”.

Instancepath	Name	Value	Type	Address	Force	Comment
PROG_ERRORHANDLING.FBEH_ERRORDETECTION	OERRORTRIGGER	FALSE	BOOL			
PROG_ERRORHANDLING.FBEH_ERROR	OERRORRECOVERY	FALSE	BOOL			
PROG_MAIN.FBERRORRECOVERY	DONE	FALSE	BOOL			
PROG_MAIN.FBERRORRECOVERY.FBTIMER	ET	0ms	TIME			
PROG_MAIN.FBERRORREACTIONY	DONE	FALSE	BOOL			
PROG_MAIN.FBERRORREACTIONY.FBTIMER	ET	0ms	TIME			
PROG_MAIN.FBERRORREACTIONX	DONE	FALSE	BOOL			
PROG_MAIN.FBERRORREACTIONX.FBTIMER	ET	0ms	TIME			
PROG_MAIN	ESTATEMAIN	3	ENUM			
PROG_ERRORHANDLING	ESTATEERRORHANDLING	0	ENUM			

Figure 2-26 Watchlist Resources

- 3) Trigger Error:
 - a) Wait until the Main State Machine is in State X or State Y.
 - b) Doubleclick the variable “oErrorTrigger” and set the value to TRUE.
- 4) Acknowledge Error:
 - a) Wait until the Main and ErrorHandling State Machine has changed to the state “Error”.
 - b) Doubleclick the variable “oErrorRecovery” and set the value to TRUE.
 - c) See how the Main State Machine is restarting.



Note

Make sure that the variables are set to “FALSE” before changing. Only a positive edge of the signal triggers a reaction.

3 Best Practice Examples

The «Best Practice Examples» feature individual aspects of EPOS2 P programming. The described cases may be part of an overall application but keep their focus on single, isolated task in application programming.

Contents

3.1 «State Machine» 3-20

3.2 «Error Handling» 3-24

3.3 «Input/Output Handling» 3-28

3.4 «Homing» 3-32

3.5 «Positioning» 3-36

3.6 «Continuous Motion» 3-40

3.7 «Read Actual Value» 3-44

3.8 «Object Dictionary Access» 3-47

3.9 «Data Handling» 3-51

Scope

Hardware	Order #	Firmware Version	Reference
EPOS2		0x2120h or higher	Firmware Specification
EPOS2 P		0x0200h or higher	Firmware Specification Programming Reference Cable Starting Set Hardware Reference

Table 3-3 Best Practice Examples – covered Hardware and required Documents

Tools

Tools	Description
Software	«EPOS Studio» Version 1.42 or higher

Table 3-4 Best Practice Examples – recommended Tools

3.1 «State Machine»

3.1.1 In Brief

A state machine is the basis and starting point of every EPOS P program. Its implementation represents the framework for all other examples.

The present example explains how to implement a state machine including states and transitions.

3.1.2 Functionality

The example consists of two state machines. Thereby, one state machine implements the application process while the other implements error handling. No functionality is implemented within the different states. As a placeholder, a timer is simulating the delay of a real process functionality.

3.1.2.1 Main State Machine

The Main State Machine is an example for a typical application process.

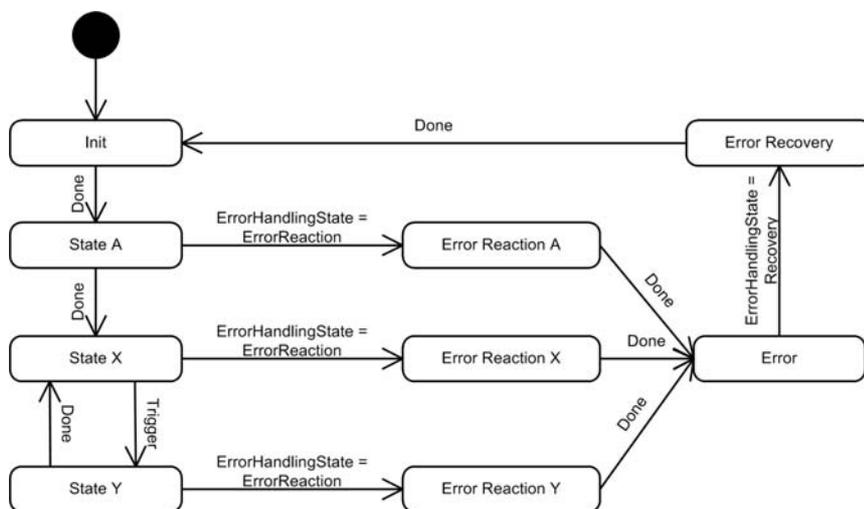


Figure 3-27 State Machine – Main State Machine

State	Description
Init	The initialization state without functionality
State A	A process state without functionality
State X	Digital Output 1 is set inactive
State Y	Digital Output 1 is set active
Error Reaction A	An error reaction state being entered when an error is detected in State A
Error Reaction X	An error reaction state being entered when an error is detected in State X
Error Reaction Y	An error reaction state being entered when an error is detected in State Y
Error	After an error reaction, the process remains in State “Error” until the error is acknowledged
Error Recovery	After acknowledgment, the process is recovered and restarted

Table 3-5 State Machine – Main State Machine (States)

Transition		Description
Init	→ State A	Done = Timer of 10 s has elapsed
State A	→ State X	Done = Timer of 10 s has elapsed
State X	→ State Y	Done = Timer of 10 s has elapsed
State Y	→ State X	Done = Timer of 10 s has elapsed
State A	→ Error Reaction A	ErrorHandling State = Error Reaction
State X	→ Error Reaction X	ErrorHandling State = Error Reaction
State Y	→ Error Reaction Y	ErrorHandling State = Error Reaction
Error Reaction A	→ Error	Done = Timer of 10 s has elapsed
Error Reaction X	→ Error	Done = Timer of 10 s has elapsed
Error Reaction Y	→ Error	Done = Timer of 10 s has elapsed
Error	→ Error Recovery	ErrorHandling State = Error Recovery
Error Recovery	→ Init	Done = Timer of 10 s has elapsed

Table 3-6 State Machine – Main State Machine (Transitions)

3.1.2.2 Error Handling State Machine

The Error Handling State Machine monitors the process and coordinates error reactions and error recoveries of the processes. Error handling is implemented as a separate state machine allowing the implementation of more than one process state machines.

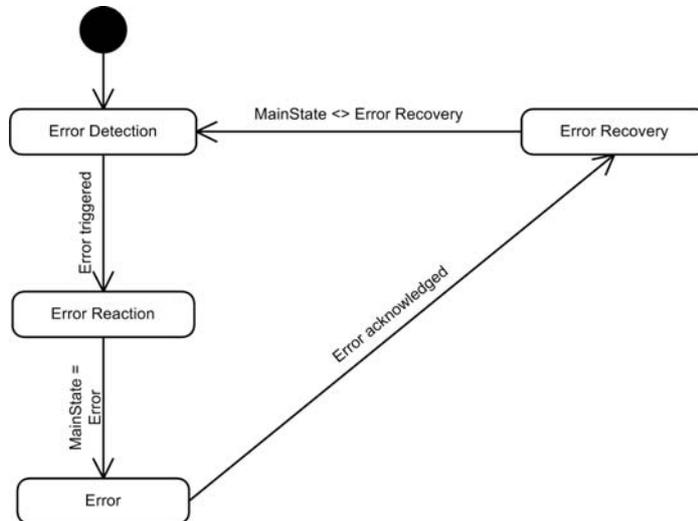


Figure 3-28 State Machine – Error Handling State Machine

State	Description
Error Detection	Monitors the variable “oErrorTrigger”.
Error Reaction	Signals to the Main State Machine that an error reaction must be started. The Error Handling State Machine remains in this state until all processes have accomplished their error reactions.
Error	Entered when the Main State Machine has finished the error reaction
Error Recovery	Signals to the Main State Machines that the error is acknowledged and that the process can be restarted. The Error Handling State Machine remains in this state until all processes have been restarted.

Table 3-7 State Machine – Error Handling State Machine (States)

Transition	Description
Error Detection → Error Reaction	Variable “oErrorTrigger” = FALSE → TRUE
Error Reaction → Error	Process State = Error
Error → Error Recovery	Variable “oErrorRecovery” = FALSE → TRUE
Error Recovery → Error Detection	Process State ≠ Error Recovery

Table 3-8 State Machine – Error Handling State Machine (Transitions)

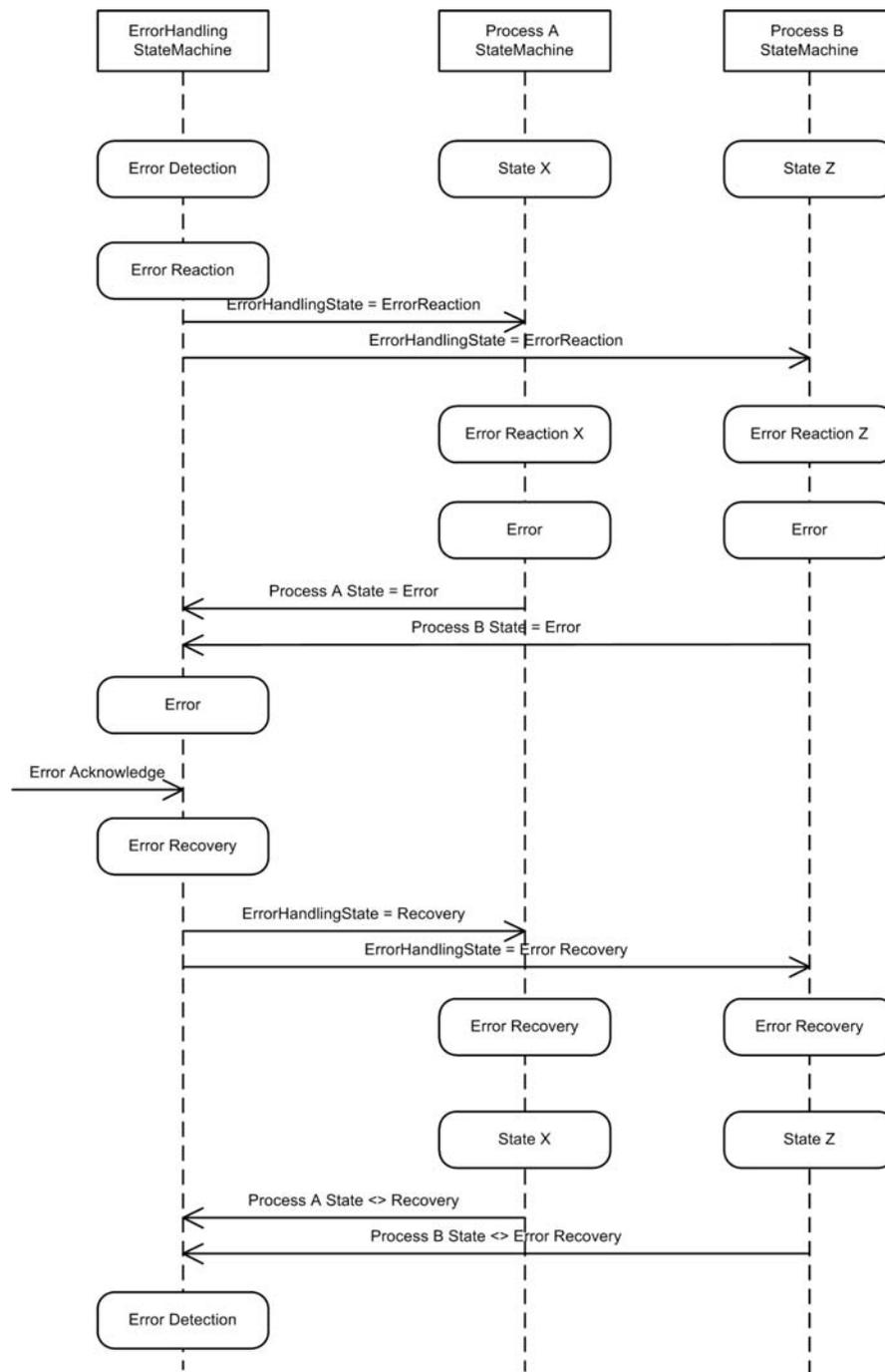


Figure 3-29 State Machine – State Machine Interactions for more than one Process



Note

For an example on how to handle the «EPOS Studio» during application programming → chapter “2 Getting Started” on page 2-9.

3.2 «Error Handling»

3.2.1 In Brief

The example demonstrates the usage of the Error Handling State Machine. The state machine detects axis-related errors, communication errors, and is gathering information on the individual error cause. The error information is shown in separate variables on the debug screen.

3.2.2 Functionality

The example consists of two state machines. Thereby, one state machine implements the application process while the other implements error handling. Basically, the Main State Machine is equal to the framework example → “«State Machine»” on page 3-20. In addition, handling of information from communication errors is added. As a result, the Error Handling State Machine has a real functionality. Axis-related errors and communication errors can be detected with state “EH_ErrorDetection” rather than just simulated. The state “EH_Error” is used to collect error information.

3.2.2.1 Main State Machine

The Main State Machine is an example for a typical application process.

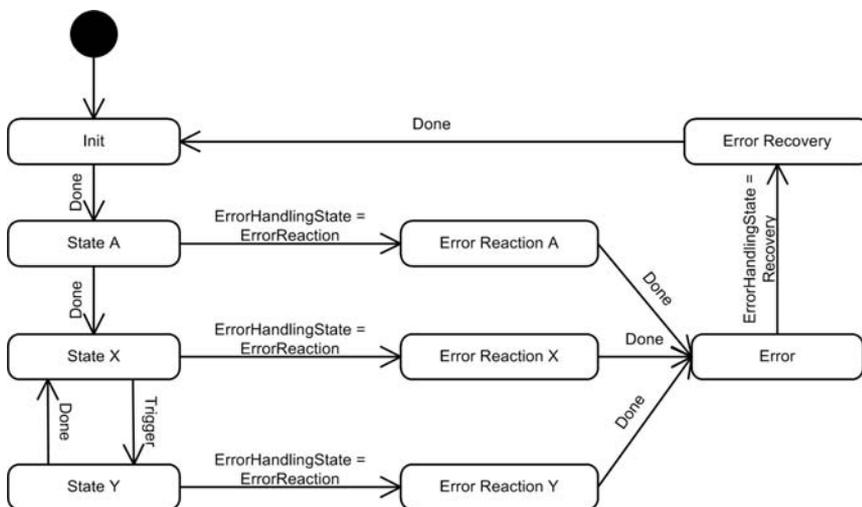


Figure 3-30 Error Handling – Main State Machine

State	Description
Init	The initialization state without functionality
State A	A process state without functionality
State X	Digital Output 1 is set inactive
State Y	Digital Output 1 is set active
Error Reaction A	An error reaction state being entered when an error is detected in State A
Error Reaction X	An error reaction state being entered when an error is detected in State X
Error Reaction Y	An error reaction state being entered when an error is detected in State Y
Error	After an error reaction, the process remains in “Error” state until the error is acknowledged
Error Recovery	After acknowledgment, the process is recovered and restarted

Table 3-9 Error Handling – Main State Machine (States)

Transition		Description
Init	→ State A	Done = Timer of 10 s has elapsed
State A	→ State X	Done = Timer of 10 s has elapsed
State X	→ State Y	Done = Timer of 10 s has elapsed
State Y	→ State X	Done = Timer of 10 s has elapsed
State A	→ Error Reaction A	ErrorHandling State = Error Reaction
State X	→ Error Reaction X	ErrorHandling State = Error Reaction
State Y	→ Error Reaction Y	ErrorHandling State = Error Reaction
Error Reaction A	→ Error	Done = Timer of 10 s has elapsed
Error Reaction X	→ Error	Done = Timer of 10 s has elapsed
Error Reaction Y	→ Error	Done = Timer of 10 s has elapsed
Error	→ Error Recovery	ErrorHandling State = Error Recovery
Error Recovery	→ Init	Done = Timer of 10 s has elapsed, all axis-related errors are reset, and global variables for error handling are set to the initial condition

Table 3-10 Error Handling – Main State Machine (Transitions)

3.2.2.2 Error Handling State Machine

The Error Handling State Machine monitors the process and coordinates error reactions and error recoveries of the processes. Error handling is implemented as a separate state machine allowing the implementation of more than one process state machines.

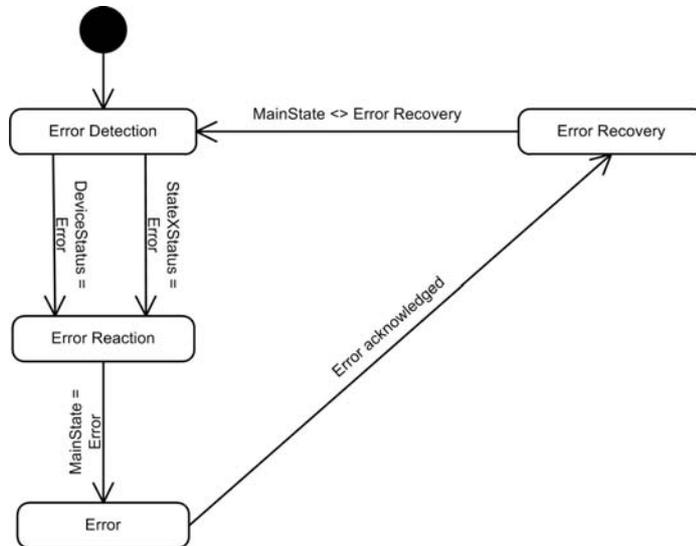


Figure 3-31 Error Handling – Error Handling State Machine

State	Description
Error Detection	Monitors the variable “oErrorTrigger”.
Error Reaction	Signals to the Main State Machine that an error reaction must be started. The Error Handling State Machine remains in this state until all processes have accomplished their error reactions.
Error	Entered when the Main State Machine has finished the error reaction
Error Recovery	Signals to the Main State Machines that the error is acknowledged and that the process can be restarted. The Error Handling State Machine remains in this state until all processes have been restarted.

Table 3-11 Error Handling – Error Handling State Machine (States)

Transition	Description
Error Detection → Error Reaction	Variable “oErrorTrigger” = FALSE → TRUE
Error Reaction → Error	Process State = Error
Error → Error Recovery	Variable “oErrorRecovery” = FALSE → TRUE
Error Recovery → Error Detection	Process State ≠ Error Recovery

Table 3-12 Error Handling – Error Handling State Machine (Transitions)

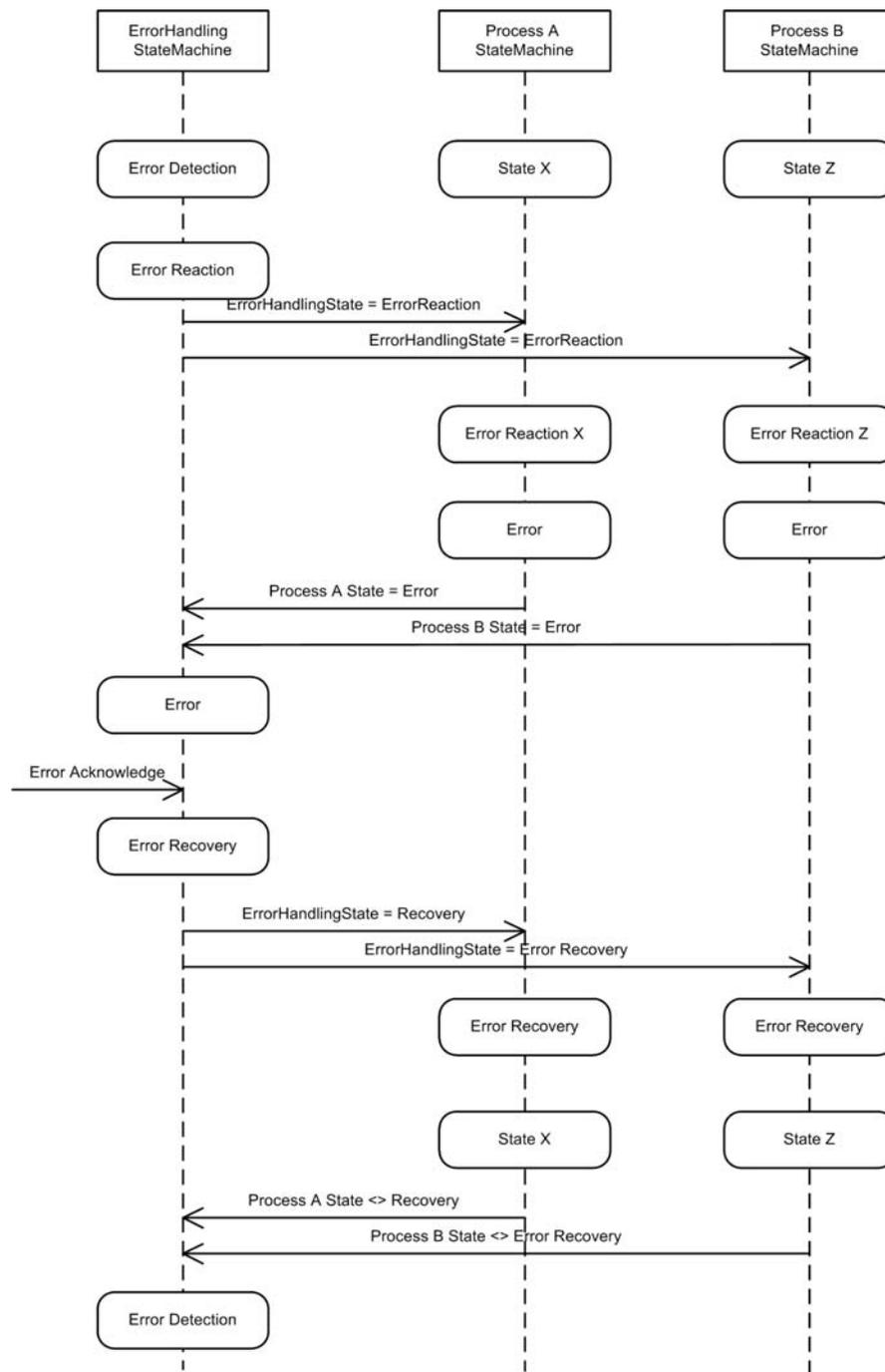


Figure 3-32 Error Handling – State Machine Interactions for more than one Process



Note

For an example on how to handle the «EPOS Studio» during application programming →chapter “2 Getting Started” on page 2-9.

3.3 «Input/Output Handling»

3.3.1 In Brief

The example demonstrates how to read digital and analog inputs and how to write digital outputs.

3.3.2 Functionality

The example consists of two state machines. Thereby, one state machine implements the application process while the other implements error handling. State changes of the Main State Machine are triggered by timers. Transition to the next state will be done when time has elapsed and functionalities have correctly finished. The state “Digital Input” reads the digital inputs, state “Analog Input” reads the two analog inputs and generates their average. The state “Digital Output” writes the digital outputs.

3.3.2.1 Main State Machine

The Main State Machine is an example for a typical application process.

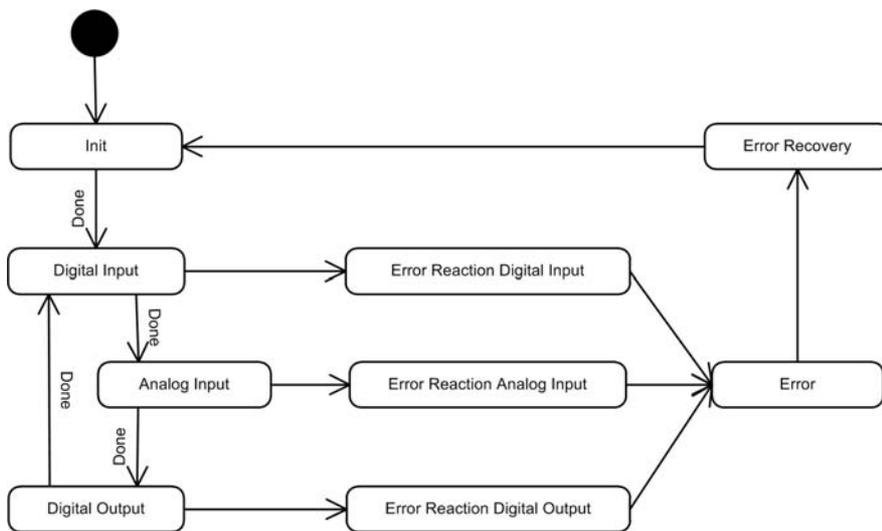


Figure 3-33 Input/Output Handling – Main State Machine

State	Description
Init	The initialization state without functionality
State Digital Input	A process state reading the digital inputs
State Analog Input	A process state reading the analog inputs and generates their average
State Digital Output	A process state writing the digital outputs
Error Reaction Digital Input	An error reaction state being entered when an error is detected in State “Digital Input”
Error Reaction Analog Input	An error reaction state being entered when an error is detected in State “Analog Input”
Error Reaction Digital Output	An error reaction state being entered when an error is detected in State “Digital Output”
Error	After an error reaction, the process remains in State “Error” until the error is acknowledged
Error Recovery	After acknowledgment, the process is recovered and restarted

Table 3-13 Input/Output Handling – Main State Machine (States)

Transition		Description
Init	→ State Digital Input	Done = Timer of 10 s has elapsed
State Digital Input	→ State Analog Input	Done = Timer of 10 s has elapsed and reading of digital inputs has correctly finished
State Analog Input	→ State Digital Output	Done = Timer of 10 s has elapsed and reading of analog inputs has correctly finished
State Digital Output	→ State Digital Input	Done = Timer of 10 s has elapsed and writing of digital outputs has correctly finished
State Digital Input	→ Error Reaction Digital Input	ErrorHandling State = Error Reaction
State Analog Input	→ Error Reaction Analog Input	ErrorHandling State = Error Reaction
State Digital Output	→ Error Reaction Digital Output	ErrorHandling State = Error Reaction
Error Reaction Digital Input	→ Error	Done = Timer of 10 s has elapsed
Error Reaction Analog Input	→ Error	Done = Timer of 10 s has elapsed
Error Reaction Digital Output	→ Error	Done = Timer of 10 s has elapsed
Error	→ Error Recovery	ErrorHandling State = Error Recovery
Error Recovery	→ Init	Done = Timer of 10 s has elapsed, all axis-related errors are reset, and global variables for error handling are set to the initial condition

Table 3-14 Input/Output Handling – Main State Machine (Transitions)

3.3.2.2 Error Handling State Machine

The Error Handling State Machine monitors the process and coordinates error reactions and error recoveries of the processes. Error handling is implemented as a separate state machine allowing the implementation of more than one process state machines.

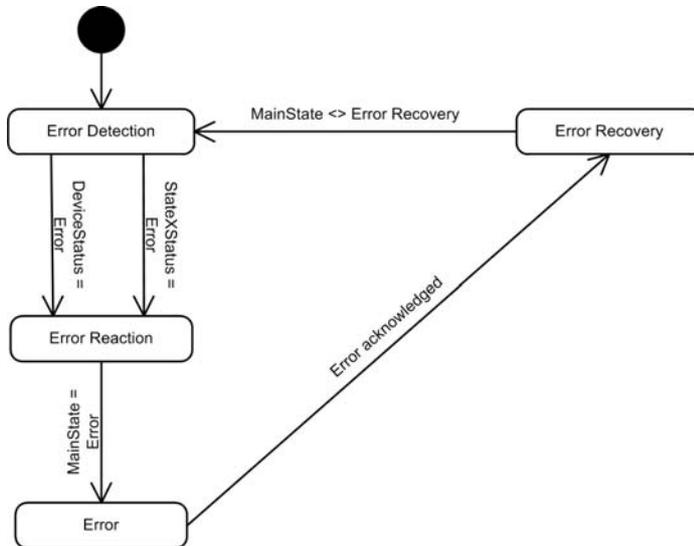


Figure 3-34 Input/Output Handling – Error Handling State Machine

State	Description
Error Detection	Monitors the variable “oErrorTrigger”.
Error Reaction	Signals to the Main State Machine that an error reaction must be started. The Error Handling State Machine remains in this state until all processes have accomplished their error reactions.
Error	Entered when the Main State Machine has finished the error reaction
Error Recovery	Signals to the Main State Machines that the error is acknowledged and that the process can be restarted. The Error Handling State Machine remains in this state until all processes have been restarted.

Table 3-15 Input/Output Handling – Error Handling State Machine (States)

Transition	Description
Error Detection → Error Reaction	Variable “oErrorTrigger” = FALSE → TRUE
Error Reaction → Error	Process State = Error
Error → Error Recovery	Variable “oErrorRecovery” = FALSE → TRUE
Error Recovery → Error Detection	Process State ≠ Error Recovery

Table 3-16 Input/Output Handling – Error Handling State Machine (Transitions)

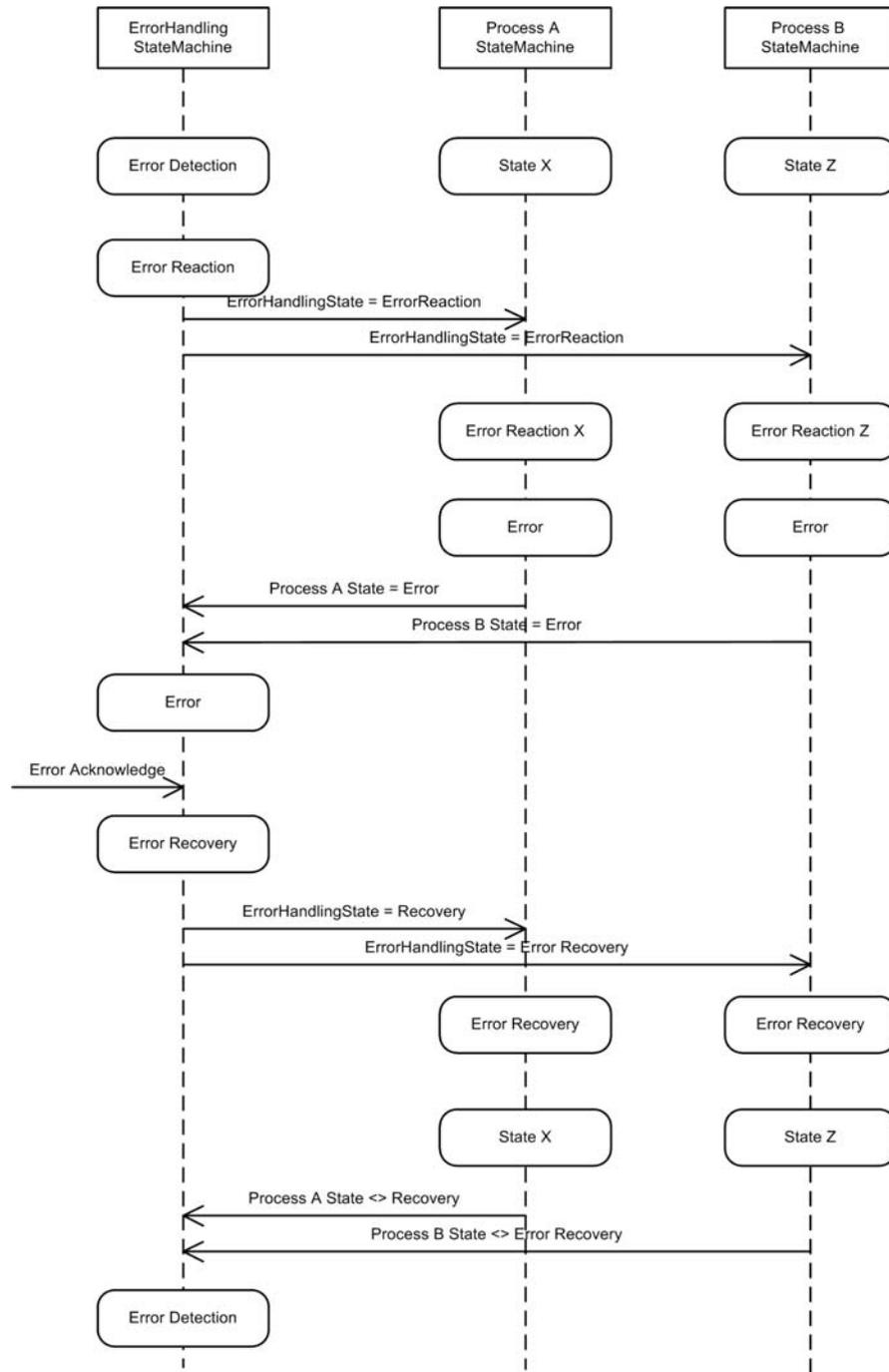


Figure 3-35 Input/Output Handling – State Machine Interactions for more than one Process



Note

For an example on how to handle the «EPOS Studio» during application programming → chapter “2 Getting Started” on page 2-9.

3.4 «Homing»

3.4.1 In Brief

The example demonstrates how to configure, start, and stop a homing procedure.

3.4.2 Functionality

The example consists of two state machines. Thereby, one state machine implements the application process while the other implements error handling. The state "Init" has no functionality. The Main State Machine consists of nine states. The states are used to demonstrate a typical homing sequence. Error handling stops the axis if an error is detected. Error handling stops the axis if an error is detected.

3.4.2.1 Main State Machine

The Main State Machine is an example for a typical application process.

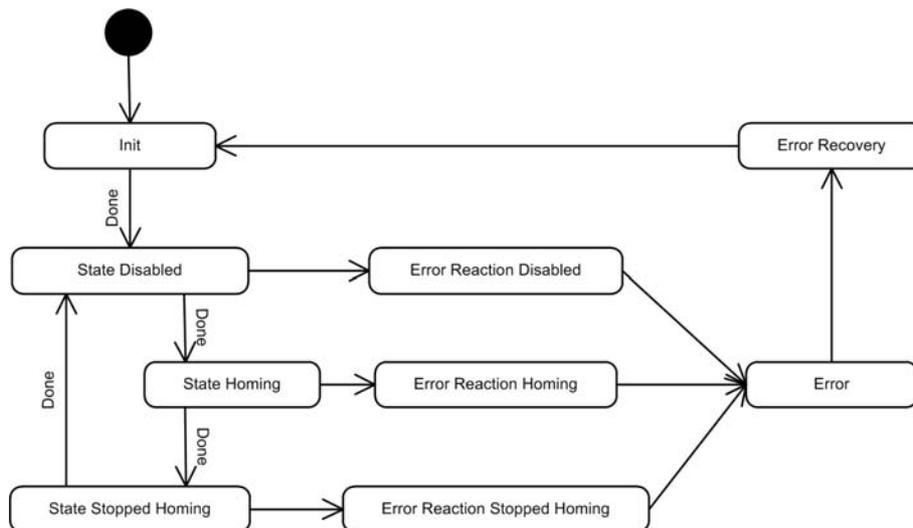


Figure 3-36 Homing – Main State Machine



Note

The current threshold homing method is configured for all homing procedures. Block the motor axis by hand to abort homing.

State	Description
Init	The initialization state without functionality
State Disabled	A process state being entered after "Init" or after one complete process cycle
State Homing	A process state setting the EPOS "Enabled", configures, and starts a homing procedure
State Stopped Homing	A process state stopping the homing procedure and sets the EPOS "Disabled"
Error Reaction Disabled	An error reaction state being entered when an error is detected in State "Disabled"
Error Reaction Homing	An error reaction state being entered when an error is detected in State "Homing"
Error Reaction Stopped Homing	An error reaction state being entered when an error is detected in State "Stopped Homing"

State	Description
Error	After an error reaction, the process remains in State “Error” until the error is acknowledged
Error Recovery	After acknowledgment, the process is recovered and restarted

Table 3-17 Homing – Main State Machine (States)

Transition	Description
Init → State Disabled	Done = TRUE
State Disabled → State Homing	Done = Timer of 5 s has elapsed
State Homing → State Stopped Homing	Done = The homing procedure has finished. <i>Note: Block motor axis by hand to finish homing</i>
State Stopped Homing → State Disabled	Done = The axis is at standstill and the EPOS “Disabled”
State Disabled → Error Reaction Disabled	ErrorHandling State = Error Reaction
State Homing → Error Reaction Homing	ErrorHandling State = Error Reaction
State Stopped Homing → Error Reaction Stopped Homing	ErrorHandling State = Error Reaction
Error Reaction Disabled → Error	Done = Timer of 10 s has elapsed
Error Reaction Homing → Error	Done = The axis was stopped
Error Reaction Stopped Homing → Error	Done = The axis was stopped
Error → Error Recovery	ErrorHandling State = Error Recovery
Error Recovery → Init	Done = Timer of 10 s has elapsed and all axis-related errors are reset

Table 3-18 Homing – Main State Machine (Transitions)

3.4.2.2 Error Handling State Machine

The Error Handling State Machine monitors the process and coordinates error reactions and error recoveries of the processes. Error handling is implemented as a separate state machine allowing the implementation of more than one process state machines.

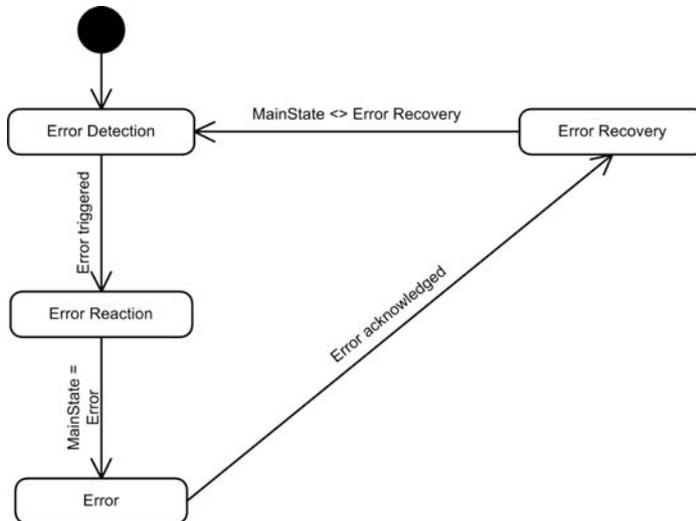


Figure 3-37 Homing – Error Handling State Machine

State	Description
Error Detection	Monitors the variable “oErrorTrigger”.
Error Reaction	Signals to the Main State Machine that an error reaction must be started. The Error Handling State Machine remains in this state until all processes have accomplished their error reactions.
Error	Entered when the Main State Machine has finished the error reaction
Error Recovery	Signals to the Main State Machines that the error is acknowledged and that the process can be restarted. The Error Handling State Machine remains in this state until all processes have been restarted.

Table 3-19 Homing – Error Handling State Machine (States)

Transition	Description
Error Detection → Error Reaction	Variable “oErrorTrigger” = FALSE → TRUE
Error Reaction → Error	Process State = Error
Error → Error Recovery	Variable “oErrorRecovery” = FALSE → TRUE
Error Recovery → Error Detection	Process State ≠ Error Recovery

Table 3-20 Homing – Error Handling State Machine (Transitions)

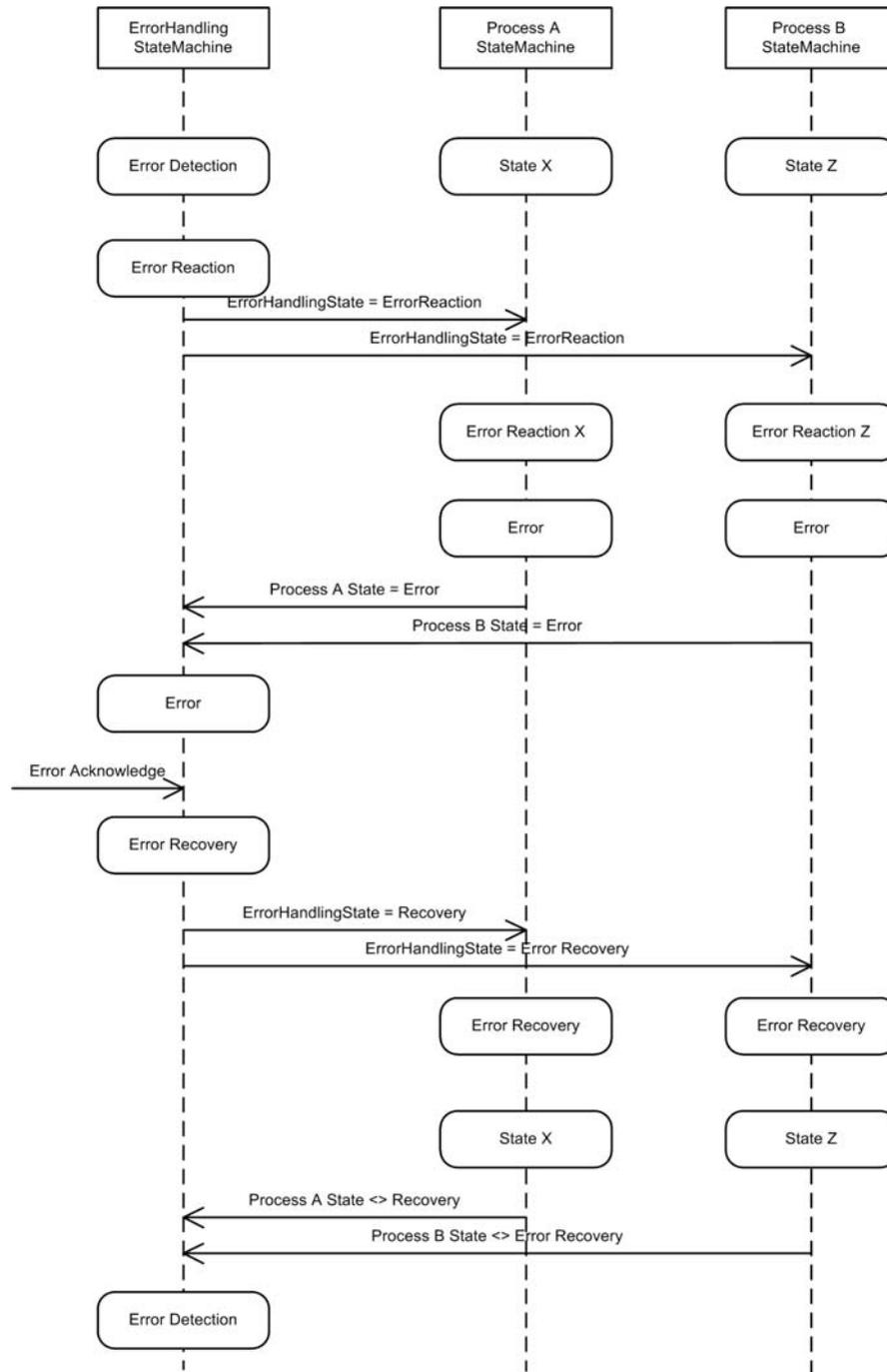


Figure 3-38 Homing – State Machine Interactions for more than one Process



Note

For an example on how to handle the «EPOS Studio» during application programming → chapter “2 Getting Started” on page 2-9.

3.5 «Positioning»

3.5.1 In Brief

The example demonstrates how to execute positioning operations.

3.5.2 Functionality

The example consists of two state machines. Thereby, one state machine implements the application process while the other implements error handling. The Main State Machine consists of eleven states. The states realize three different kinds of positioning operations. The error handling stops the axis if an error is detected.

3.5.2.1 Main State Machine

The Main State Machine is an example for a typical application process.

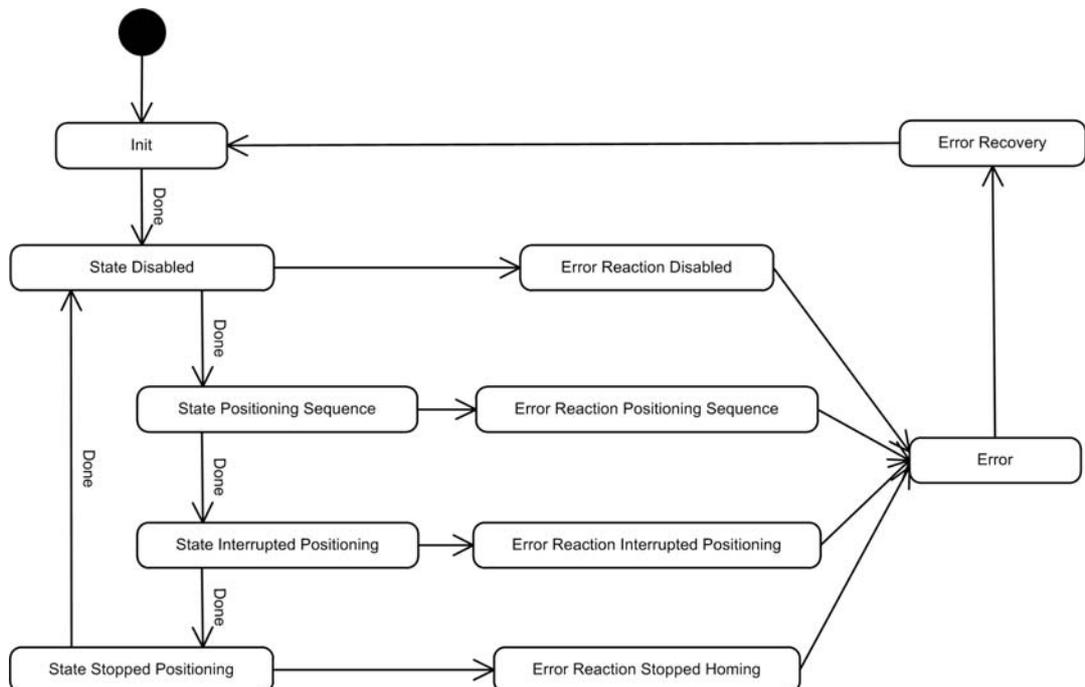


Figure 3-39 Positioning – Main State Machine

State	Description
Init	The initialization state without functionality
State Disabled	A process state being entered after the Init state or after one complete process cycle
State Positioning Sequence	A process state setting the EPOS “Enabled” and starts two sequential relative positioning operations
State Interrupted Positioning	A process state executing an interrupted positioning operation
State Stopped Positioning	A process state stopping positioning operation and sets the EPOS “Disabled”.
Error Reaction Disabled	An error reaction state being entered when an error is detected in State “Disabled”
Error Reaction Positioning Sequence	An error reaction state being entered when an error is detected in State “Positioning Sequence”

State	Description
Error Reaction Interrupted Positioning	An error reaction state being entered when an error is detected in State "Interrupted Positioning"
Error Reaction Stopped Positioning	An error reaction state being entered when an error is detected in State "Stopped Positioning"
Error	After an error reaction, the process remains in State "Error" until the error is acknowledged
Error Recovery	After acknowledgment, the process is recovered and restarted

Table 3-21 Positioning – Main State Machine (States)

Transition	Description
Init → State Disabled	Done = TRUE
State Disabled → State Positioning Sequence	Done = Timer of 5 s has elapsed
State Positioning Sequence → State Interrupted Positioning	Done = Both sequential relative positioning operations are completely finished
State Interrupted Positioning → State Stopped Positioning	Done = The interrupted positioning operation is completely finished
State Stopped Positioning → State Disabled	Done = The axis is at standstill and the EPOS is "Disabled"
State Disabled → Error Reaction Disabled	ErrorHandling State = Error Reaction
State Positioning Sequence → Error Reaction Positioning Sequence	ErrorHandling State = Error Reaction
State Interrupted Positioning → Error Reaction Interrupted Positioning	ErrorHandling State = Error Reaction
State Stopped Positioning → Error Reaction Stopped Positioning	ErrorHandling State = Error Reaction
Error Reaction Disabled → Error	Done = Timer of 10 s has elapsed
Error Reaction Positioning Sequence → Error	Done = The axis was stopped
Error Reaction Interrupted Positioning → Error	Done = The axis was stopped
Error Reaction Stopped Positioning → Error	Done = The axis was stopped
Error → Error Recovery	ErrorHandling State = Error Recovery

Transition	Description
Error Recovery → Init	Done = Timer of 10 s has elapsed and all axis-related errors are reset

Table 3-22 Positioning – Main State Machine (Transitions)

3.5.2.2 Error Handling State Machine

The Error Handling State Machine monitors the process and coordinates error reactions and error recoveries of the processes. Error handling is implemented as a separate state machine allowing the implementation of more than one process state machines.

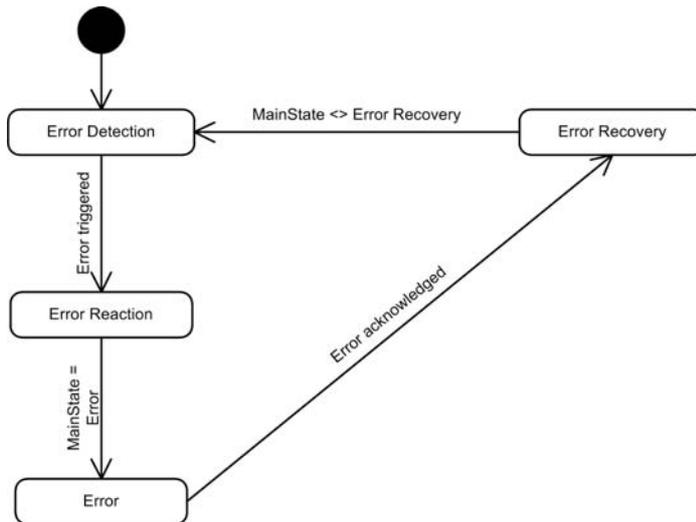


Figure 3-40 Positioning – Error Handling State Machine

State	Description
Error Detection	Monitors the variable “oErrorTrigger”.
Error Reaction	Signals to the Main State Machine that an error reaction must be started. The Error Handling State Machine remains in this state until all processes have accomplished their error reactions.
Error	Entered when the Main State Machine has finished the error reaction
Error Recovery	Signals to the Main State Machines that the error is acknowledged and that the process can be restarted. The Error Handling State Machine remains in this state until all processes have been restarted.

Table 3-23 Positioning – Error Handling State Machine (States)

Transition	Description
Error Detection → Error Reaction	Variable “oErrorTrigger” = FALSE → TRUE
Error Reaction → Error	Process State = Error
Error → Error Recovery	Variable “oErrorRecovery” = FALSE → TRUE
Error Recovery → Error Detection	Process State ≠ Error Recovery

Table 3-24 Positioning – Error Handling State Machine (Transitions)

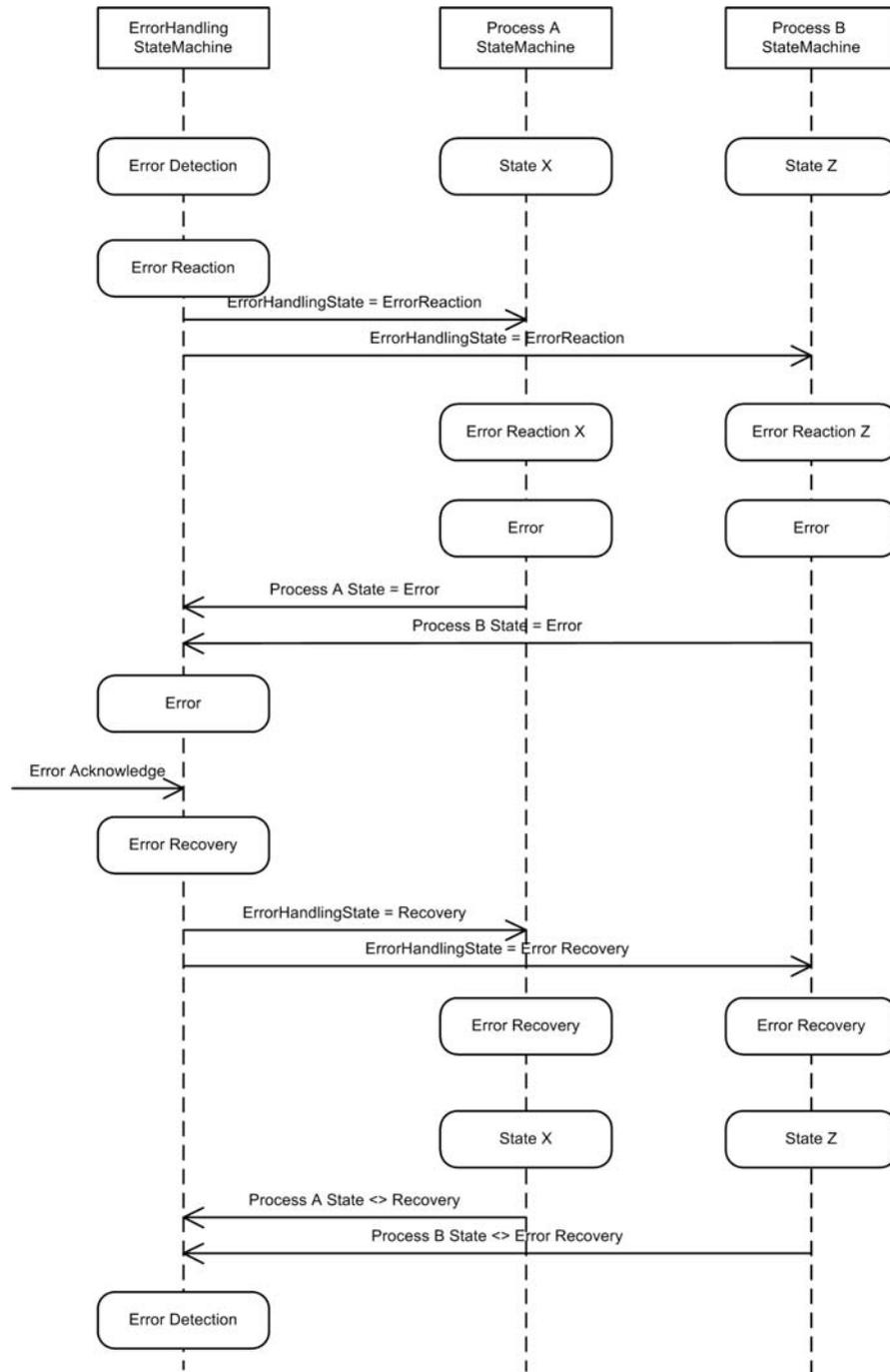


Figure 3-41 Positioning – State Machine Interactions for more than one Process



Note

For an example on how to handle the «EPOS Studio» during application programming → chapter “2 Getting Started” on page 2-9.

3.6 «Continuous Motion»

3.6.1 In Brief

The example demonstrates how to execute continuous motions. Shown will be different kinds of continuous motion; two sequential continuous, an interrupted continuous, and stopping the continuous motion.

3.6.2 Functionality

The example consists of two state machines. Thereby, one state machine implements the application process while the other implements error handling. The Main State Machine consists of eleven states. The states realize three different types of continuous motion. Error handling is stopping the axis if an error is detected.

3.6.2.1 Main State Machine

The Main State Machine is an example for a continuous motion process.

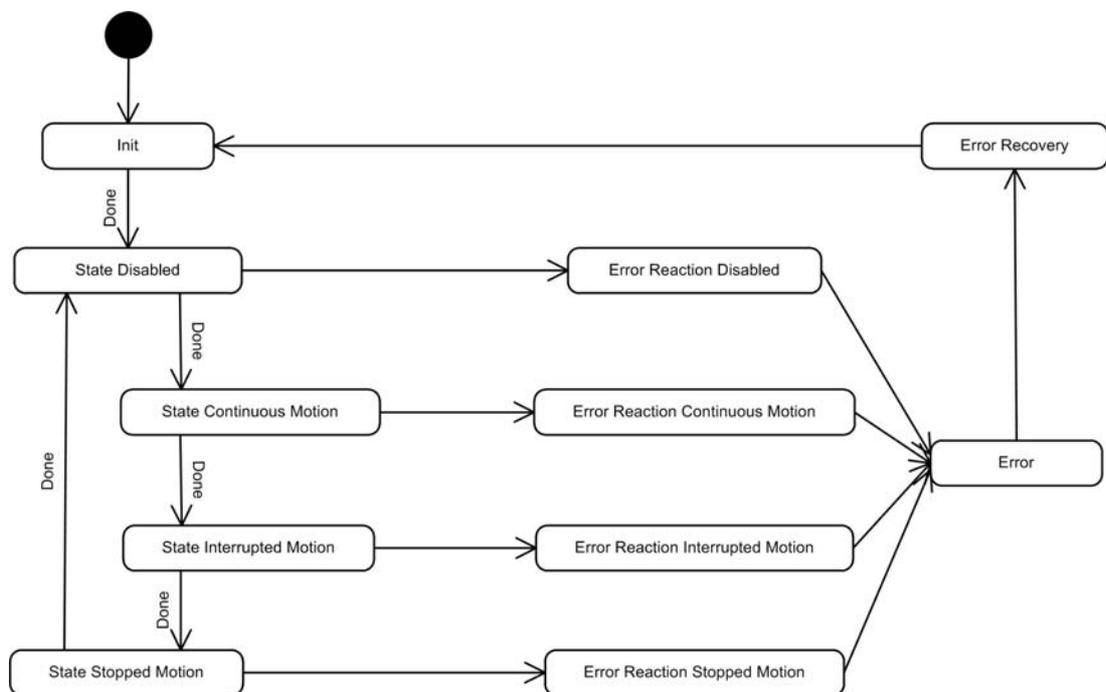


Figure 3-42 Continuous Motion – Main State Machine

State	Description
Init	The initialization state without functionality
State Disabled	A process state being entered after “Init” state or after one complete process cycle
State Continuous Motion	A process state setting the EPOS “Enabled” and starts two sequential continuous motions
State Interrupted Continuous Motion	A process state executing an interrupted continuous motion
State Stopped Continuous Motion	A process state stopping the continuous motion and sets the EPOS “Disabled”
Error Reaction Disabled	An error reaction state being entered when an error is detected in State “Disabled”

State	Description
Error Reaction Continuous Motion	An error reaction state being entered when an error is detected in State “Continuous Motion”
Error Reaction Interrupted Motion	An error reaction state being entered when an error is detected in State “Interrupted Motion”
Error Reaction Stopped Motion	An error reaction state being entered when an error is detected in State “Stopped Motion”
Error	After an error reaction, the process remains in State “Error” until the error is acknowledged
Error Recovery	After acknowledgment, the process is recovered and restarted

Table 3-25 Continuous Motion – Main State Machine (States)

Transition	Description
Init → State Disabled	Done = TRUE
State Disabled → State Continuous Motion	Done = Timer of 5 s has elapsed
State Continuous Motion → State Interrupted Motion	Done = Both sequential continuous motions are correctly finished
State Interrupted Motion → State Stopped Motion	Done = The interrupted continuous motion is correctly finished
State Stopped Continuous Motion → State Disabled	Done = The axis is at standstill and the EPOS “Disabled”
State Disabled → Error Reaction Disabled	ErrorHandling State = Error Reaction
State Continuous Motion → Error Reaction Continuous Motion	ErrorHandling State = Error Reaction
State Interrupted Motion → Error Reaction Interrupted Motion	ErrorHandling State = Error Reaction
State Stopped Motion → Error Reaction Stopped Motion	ErrorHandling State = Error Reaction
Error Reaction Disabled → Error	Done = Timer of 10 s has elapsed
Error Reaction Continuous Motion → Error	Done = The axis was stopped
Error Reaction Interrupted Motion → Error	Done = The axis was stopped
Error Reaction Stopped Motion → Error	Done = The axis was stopped
Error → Error Recovery	ErrorHandling State = Error Recovery
Error Recovery → Init	Done = Timer of 10 s has elapsed and all axis-related errors are reset

Table 3-26 Continuous Motion – Main State Machine (Transitions)

3.6.2.2 Error Handling State Machine

The Error Handling State Machine monitors the process and coordinates error reactions and error recoveries of the processes. Error handling is implemented as a separate state machine allowing the implementation of more than one process state machines.

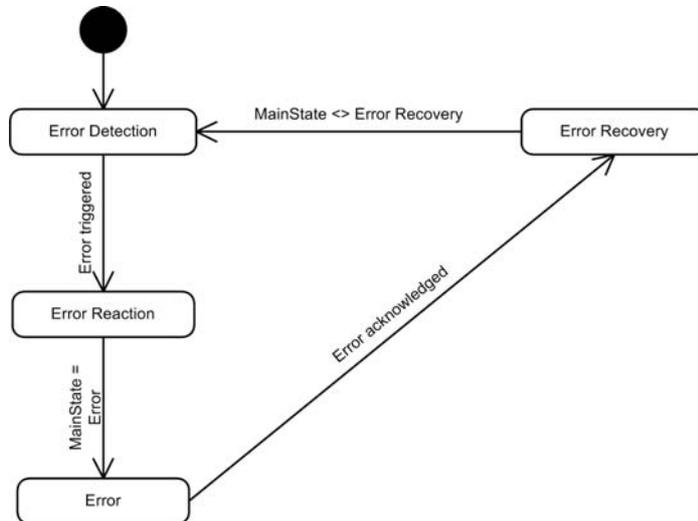


Figure 3-43 Continuous Motion – Error Handling State Machine

State	Description
Error Detection	Monitors the variable “oErrorTrigger”.
Error Reaction	Signals to the Main State Machine that an error reaction must be started. The Error Handling State Machine remains in this state until all processes have accomplished their error reactions.
Error	Entered when the Main State Machine has finished the error reaction
Error Recovery	Signals to the Main State Machines that the error is acknowledged and that the process can be restarted. The Error Handling State Machine remains in this state until all processes have been restarted.

Table 3-27 Continuous Motion – Error Handling State Machine (States)

Transition	Description
Error Detection → Error Reaction	Variable “oErrorTrigger” = FALSE → TRUE
Error Reaction → Error	Process State = Error
Error → Error Recovery	Variable “oErrorRecovery” = FALSE → TRUE
Error Recovery → Error Detection	Process State ≠ Error Recovery

Table 3-28 Continuous Motion – Error Handling State Machine (Transitions)

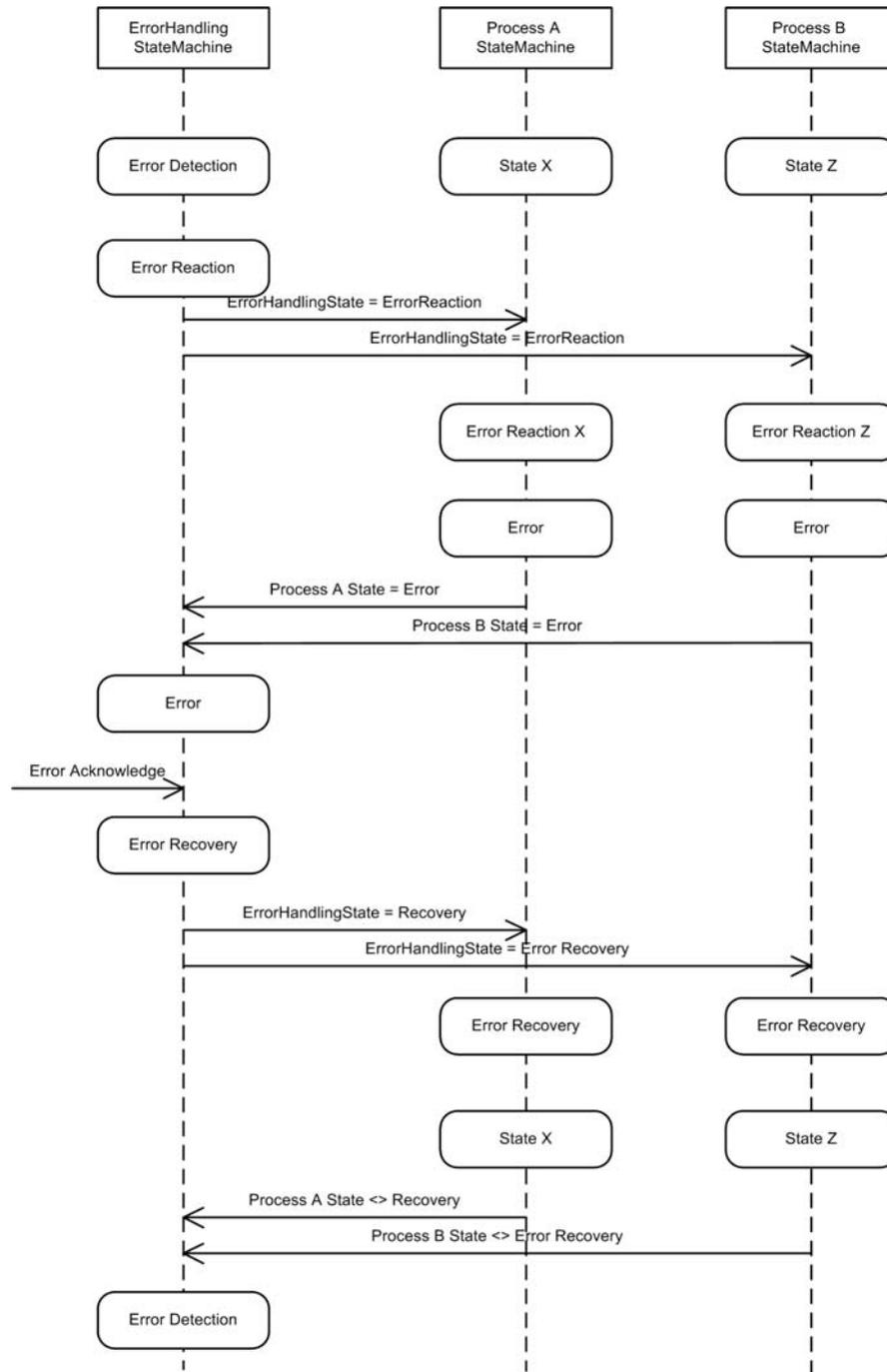


Figure 3-44 Continuous Motion – State Machine Interactions for more than one Process



Note

For an example on how to handle the «EPOS Studio» during application programming → chapter “2 Getting Started” on page 2-9.

3.7 «Read Actual Value»

3.7.1 In Brief

The example demonstrates how to read the actual position, actual velocity, and actual current of the EPOS. You may observe the results with variables “dActualPosition”, “dActualVelocity”, and “wActualCurrent”.

3.7.2 Functionality

The example consists of two state machines. Thereby, one state machine implements the application process while the other implements error handling. State “Init” has no functionality implemented and state “Reading” realizes the reading of the actual values.

3.7.2.1 Main State Machine

The Main State Machine is an example for a typical application process.

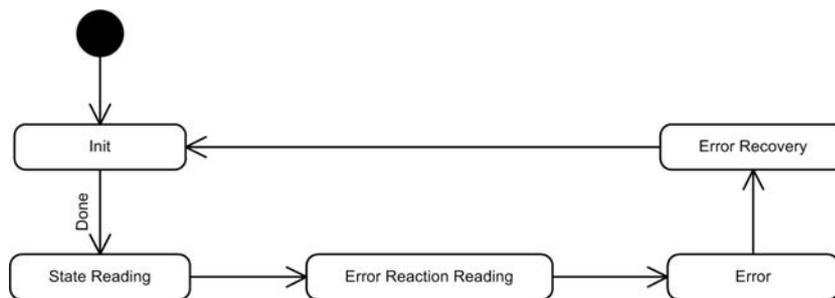


Figure 3-45 Read Actual Value – Main State Machine

State	Description
Init	The initialization state without functionality
State Reading	A process state for reading the actual values
Error Reaction Reading	An error reaction state being entered when an error is detected in State “Reading”
Error	After an error reaction, the process remains in State “Error” until the error is acknowledged
Error Recovery	After acknowledgment, the process is recovered and restarted

Table 3-29 Read Actual Value – Main State Machine (States)

Transition	Description
Init → State Reading	Done = TRUE
State Reading → Error Reaction Reading	ErrorHandling State = Error Reaction
Error Reaction Reading → Error	Done = Timer of 10 s has elapsed
Error → Error Recovery	ErrorHandling State = Error Recovery
Error Recovery → Init	Done = Timer of 10 s has elapsed

Table 3-30 Read Actual Value – Main State Machine (Transitions)

3.7.2.2 Error Handling State Machine

The Error Handling State Machine monitors the process and coordinates error reactions and error recoveries of the processes. Error handling is implemented as a separate state machine allowing the implementation of more than one process state machines.

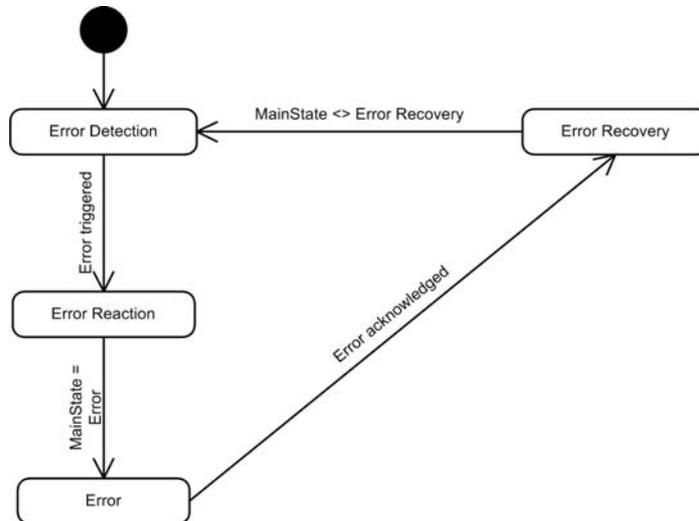


Figure 3-46 Read Actual Value – Error Handling State Machine

State	Description
Error Detection	Monitors the variable “oErrorTrigger”.
Error Reaction	Signals to the Main State Machine that an error reaction must be started. The Error Handling State Machine remains in this state until all processes have accomplished their error reactions.
Error	Entered when the Main State Machine has finished the error reaction
Error Recovery	Signals to the Main State Machines that the error is acknowledged and that the process can be restarted. The Error Handling State Machine remains in this state until all processes have been restarted.

Table 3-31 Read Actual Value – Error Handling State Machine (States)

Transition	Description
Error Detection → Error Reaction	Variable “oErrorTrigger” = FALSE → TRUE
Error Reaction → Error	Process State = Error
Error → Error Recovery	Variable “oErrorRecovery” = FALSE → TRUE
Error Recovery → Error Detection	Process State ≠ Error Recovery

Table 3-32 Read Actual Value – Error Handling State Machine (Transitions)

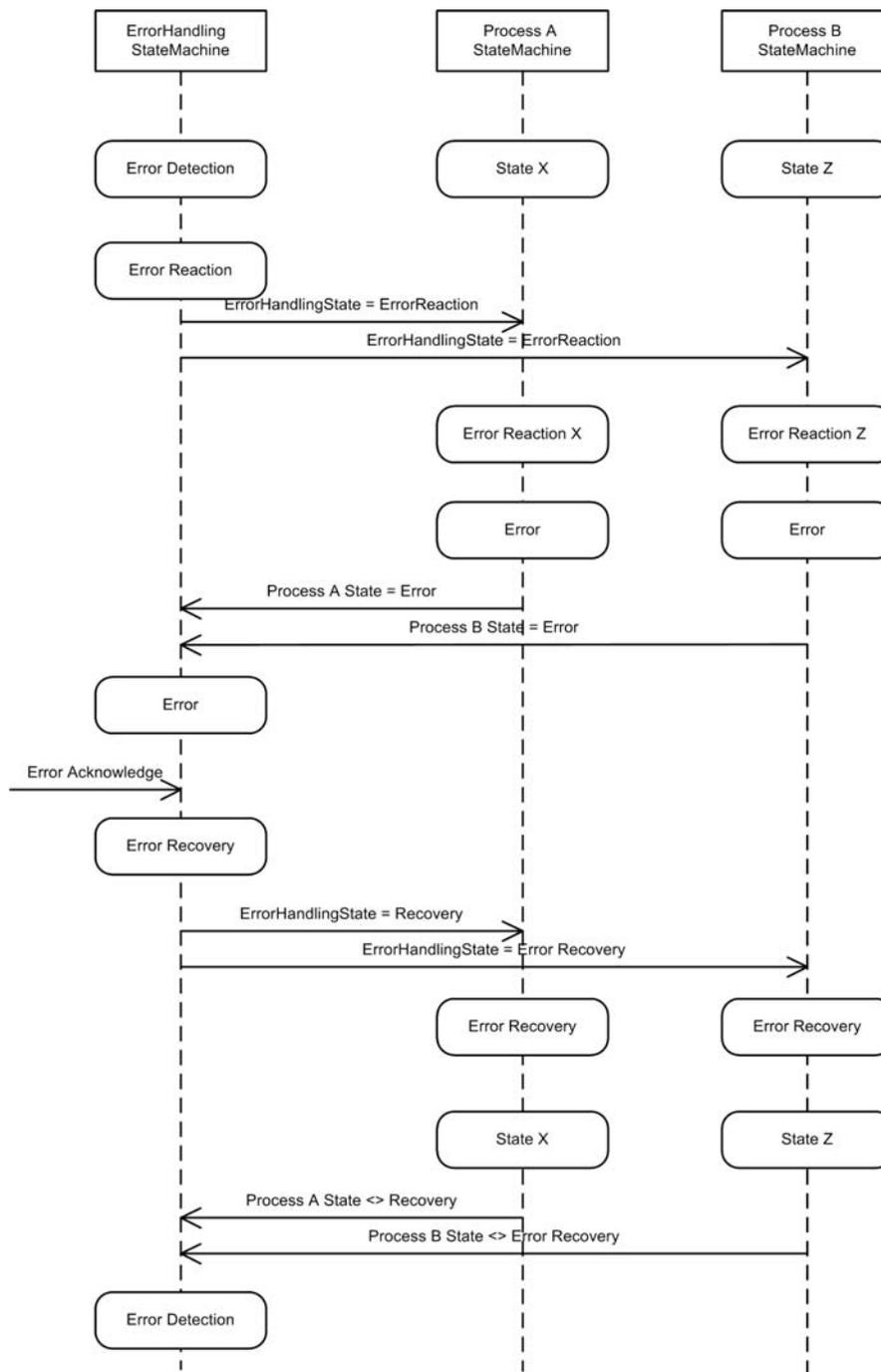


Figure 3-47 Read Actual Value – State Machine Interactions for more than one Process



Note

For an example on how to handle the «EPOS Studio» during application programming →chapter “2 Getting Started” on page 2-9.

3.8 «Object Dictionary Access»

3.8.1 In Brief

The example demonstrates how to read an object from the “ObjectDictionary”.

3.8.2 Functionality

The example consists of two state machines. Thereby, one state machine implements the application process while the other implements error handling. State “Init” has no functionality. State “Object Reading” executes read operations and state “Object Writing” executes write operations of objects from the “ObjectDictionary”.

3.8.2.1 Main State Machine

The Main State Machine is an example for a reading and writing object process.

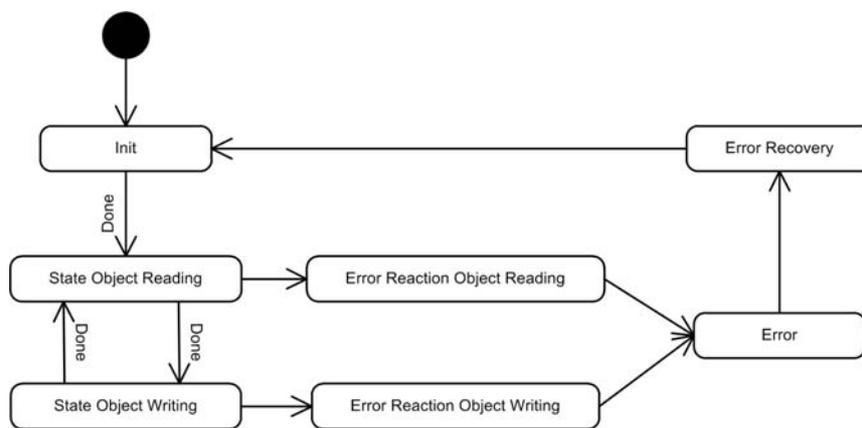


Figure 3-48 Object Dictionary Access – Main State Machine

State	Description
Init	The initialization state without functionality
State Object Reading	A process state reading objects
State Object Writing	A process state writing objects
Error Reaction Object Reading	An error reaction state being entered when an error is detected in State “Object Reading”
Error Reaction Object Writing	An error reaction state being entered when an error is detected in State “Object Writing”
Error	After an error reaction, the process remains in State “Error” until the error is acknowledged
Error Recovery	After acknowledgment, the process is recovered and restarted

Table 3-33 Object Dictionary Access – Main State Machine (States)

Transition		Description
Init	→ State Object Reading	Done = TRUE
State Object Reading	→ State Object Writing	Done = Timer of 5 s has elapsed and the objects are read
State Object Writing	→ State Object Reading	Done = Timer of 5 s has elapsed and the objects are written
State Object Reading	→ Error Reaction Object Reading	ErrorHandling State = Error Reaction
State Object Writing	→ Error Reaction Object Writing	ErrorHandling State = Error Reaction
Error Reaction Object Reading	→ Error	Done = Timer of 10 s has elapsed
Error Reaction Object Writing	→ Error	Done = Timer of 10 s has elapsed
Error	→ Error Recovery	ErrorHandling State = Error Recovery
Error Recovery	→ Init	Done = Timer of 10 s has elapsed and all axis-related errors are reset

Table 3-34 Object Dictionary Access – Main State Machine (Transitions)

3.8.2.2 Error Handling State Machine

The Error Handling State Machine monitors the process and coordinates error reactions and error recoveries of the processes. Error handling is implemented as a separate state machine allowing the implementation of more than one process state machines.

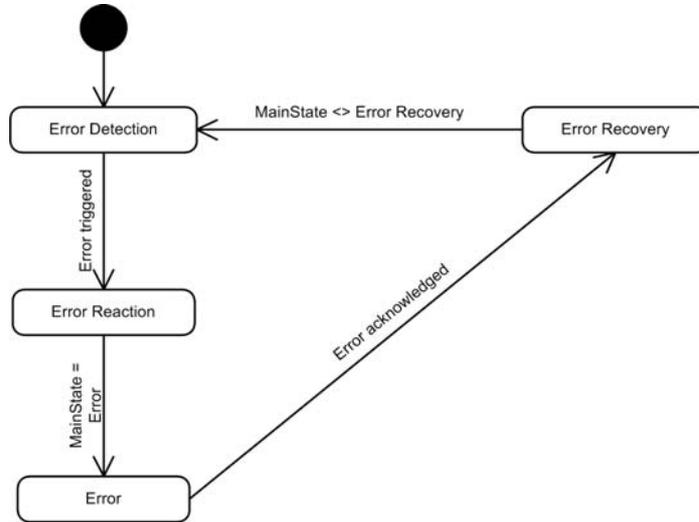


Figure 3-49 Object Dictionary Access – Error Handling State Machine

State	Description
Error Detection	Monitors the variable “oErrorTrigger”.
Error Reaction	Signals to the Main State Machine that an error reaction must be started. The Error Handling State Machine remains in this state until all processes have accomplished their error reactions.
Error	Entered when the Main State Machine has finished the error reaction
Error Recovery	Signals to the Main State Machines that the error is acknowledged and that the process can be restarted. The Error Handling State Machine remains in this state until all processes have been restarted.

Table 3-35 Object Dictionary Access – Error Handling State Machine (States)

Transition	Description
Error Detection → Error Reaction	Variable “oErrorTrigger” = FALSE → TRUE
Error Reaction → Error	Process State = Error
Error → Error Recovery	Variable “oErrorRecovery” = FALSE → TRUE
Error Recovery → Error Detection	Process State ≠ Error Recovery

Table 3-36 Object Dictionary Access – Error Handling State Machine (Transitions)

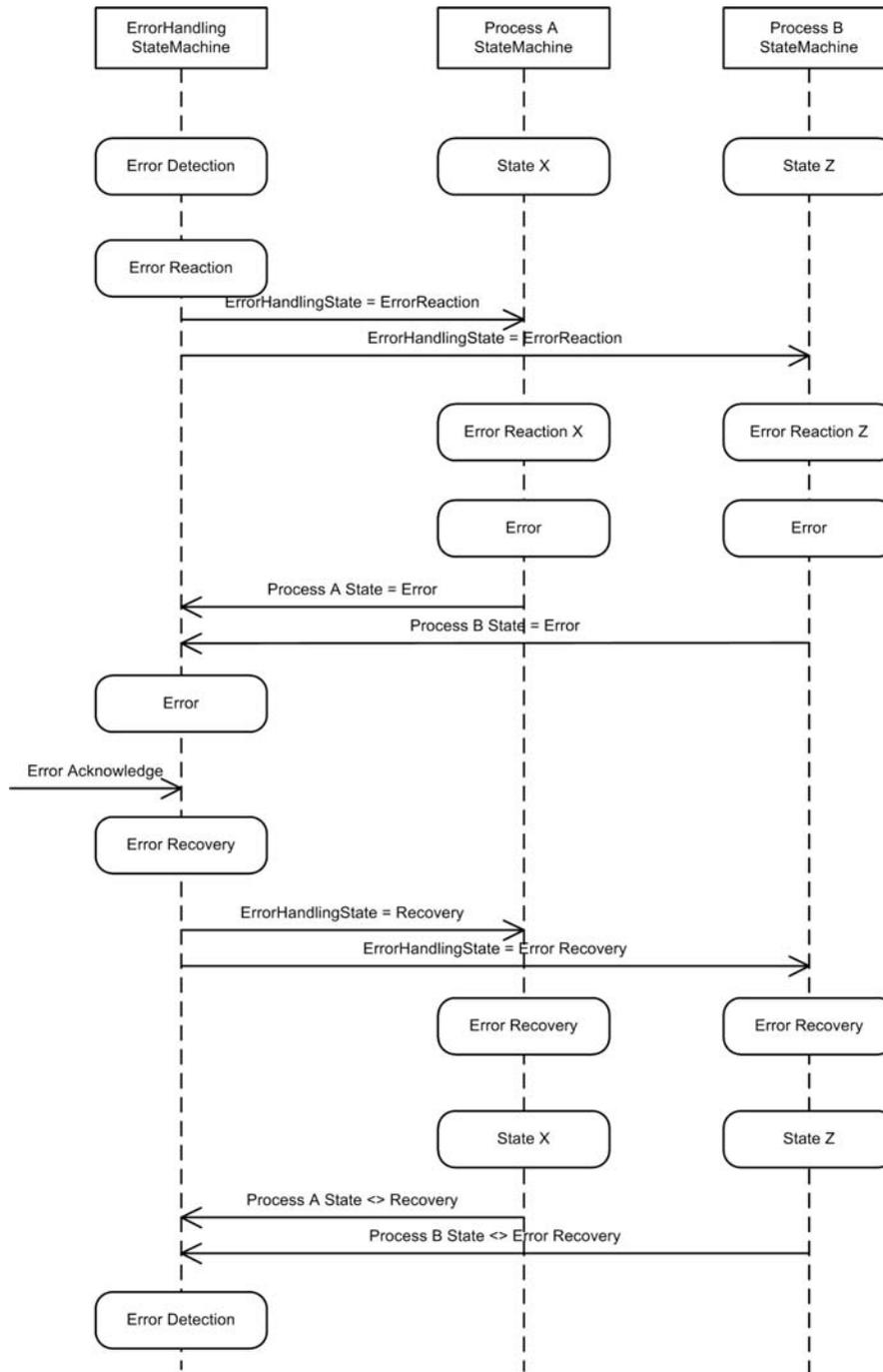


Figure 3-50 Object Dictionary Access – State Machine Interactions for more than one Process



Note

For an example on how to handle the «EPOS Studio» during application programming →chapter “2 Getting Started” on page 2-9.

3.9 «Data Handling»

3.9.1 In Brief

The example demonstrates how to process data. The example is used to read and write bits and to convert data types.

3.9.2 Functionality

The example consists of two state machines. Thereby, one state machine implements the application process while the other implements error handling. State “Init” has no functionality. State “BitHandling” shows how to read and write bits and State “TypeConversions” consists of type conversion functions from any “INT” type to “UDINT”, and vice versa.

3.9.2.1 Main State Machine

The Main State Machine is an example for a data handling process.

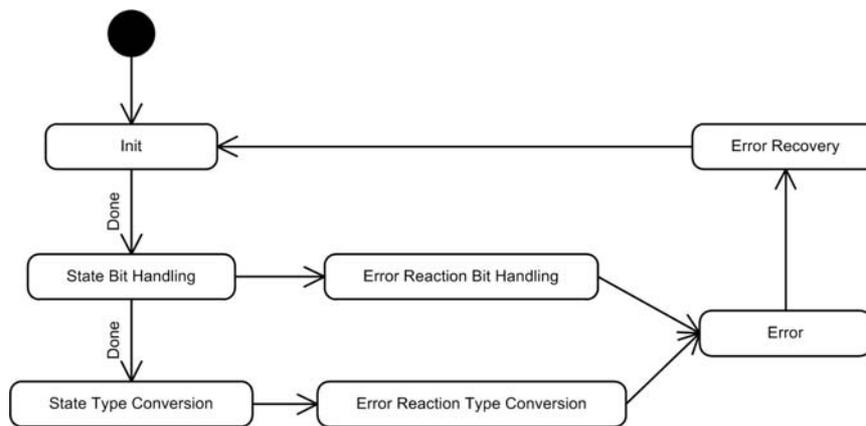


Figure 3-51 Data Handling – Main State Machine

State	Description
Init	The initialization state without functionality
State Bit Handling	A process state showing how to read and write bits
State Type Conversion	A process state showing how to convert data types
Error Reaction Bit Handling	An error reaction state being entered when an error is detected in State “Bit Handling”
Error Reaction Type Conversion	An error reaction state being entered when an error is detected in State “Type Conversion”
Error	After an error reaction, the process remains in State “Error” until the error is acknowledged
Error Recovery	After acknowledgment, the process is recovered and restarted

Table 3-37 Data Handling – Main State Machine (States)

Transition		Description
Init	→ State Bit Handling	Done = TRUE
State Bit Handling	→ State Type Conversion	Done = All bits are read and written
State Bit Handling	→ Error Reaction Bit Handling	ErrorHandling State = Error Reaction
State Type Conversion	→ Error Reaction Type Conversion	ErrorHandling State = Error Reaction
Error Reaction Bit Handling	→ Error	Done = TRUE
Error Reaction Type Conversion	→ Error	Done = TRUE
Error	→ Error Recovery	ErrorHandling State = Error Recovery
Error Recovery	→ Init	Done = Timer of 10 s has elapsed and all axis-related errors are reset

Table 3-38 Data Handling – Main State Machine (Transitions)

3.9.2.2 Error Handling State Machine

The Error Handling State Machine monitors the process and coordinates error reactions and error recoveries of the processes. Error handling is implemented as a separate state machine allowing the implementation of more than one process state machines.

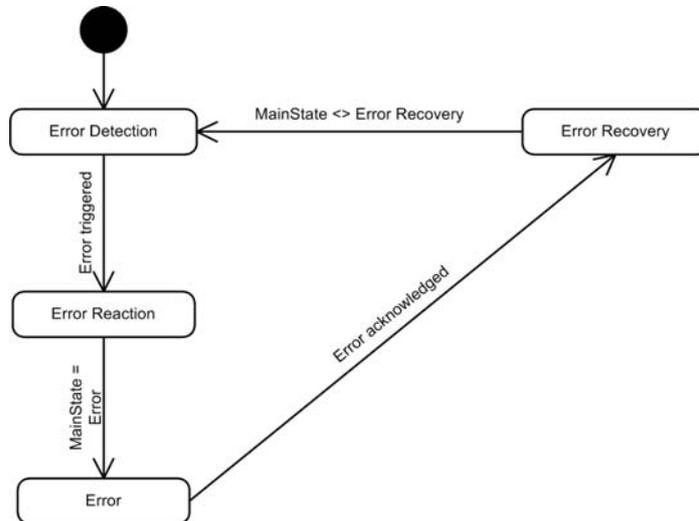


Figure 3-52 Data Handling – Error Handling State Machine

State	Description
Error Detection	Monitors the variable “oErrorTrigger”.
Error Reaction	Signals to the Main State Machine that an error reaction must be started. The Error Handling State Machine remains in this state until all processes have accomplished their error reactions.
Error	Entered when the Main State Machine has finished the error reaction
Error Recovery	Signals to the Main State Machines that the error is acknowledged and that the process can be restarted. The Error Handling State Machine remains in this state until all processes have been restarted.

Table 3-39 Data Handling – Error Handling State Machine (States)

Transition	Description
Error Detection → Error Reaction	Variable “oErrorTrigger” = FALSE → TRUE
Error Reaction → Error	Process State = Error
Error → Error Recovery	Variable “oErrorRecovery” = FALSE → TRUE
Error Recovery → Error Detection	Process State ≠ Error Recovery

Table 3-40 Data Handling – Error Handling State Machine (Transitions)

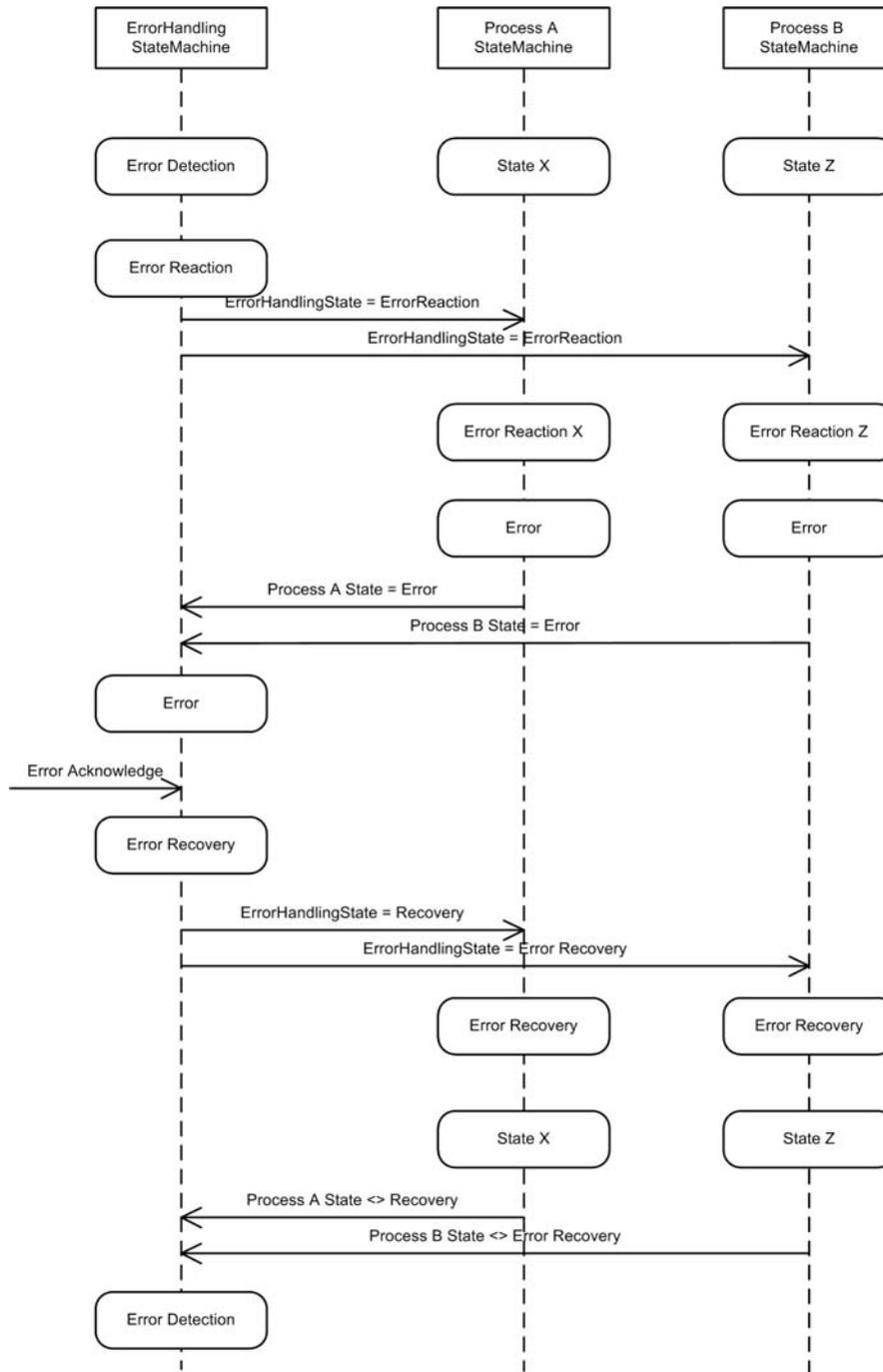


Figure 3-53 Data Handling – State Machine Interactions for more than one Process



Note

For an example on how to handle the «EPOS Studio» during application programming →chapter “2 Getting Started” on page 2-9.

4 Application Examples

The «Application Examples» describe complete, self-contained application scenarios of EPOS2 P programming. The described cases may consist of individual «Best Practice Examples».

Contents

4.1 «Cyclic Motion»	4-56
4.2 «I/O Mode»	4-61
4.3 «Multiaxis Motion»	4-66
4.4 «Process Input Output»	4-71

Scope

Hardware	Order #	Firmware Version	Reference
EPOS2		0x2120h or higher	Firmware Specification
EPOS2 P		0x0200h or higher	Firmware Specification Programming Reference Cable Starting Set Hardware Reference

Table 4-41 Application Examples – covered Hardware and required Documents

Tools

Tools	Description
Software	«EPOS Studio» Version 1.42 or higher

Table 4-42 Application Examples – recommended Tools

4.1 «Cyclic Motion»

4.1.1 In Brief

The example describes typical motion sequences with one axis featuring «Homing», «Continuous Motion», and «Positioning».

4.1.2 Functionality

The example consists of two state machines. Thereby, one state machine implements the application process while the other implements error handling. With digital input 1, you can set the EPOS2 to the state “Enabled”. Upon completion of the homing procedure, the program stays in state “WaitForGo”. The transition to state “SearchPosition” will be done with a positive edge at digital input 2. To stop the cyclic movement, digital input 3 ought to be set. Error handling will stop the axis if an error occurs.

4.1.2.1 Main State Machine

The Main State Machine is an example for a motion sequence application process.

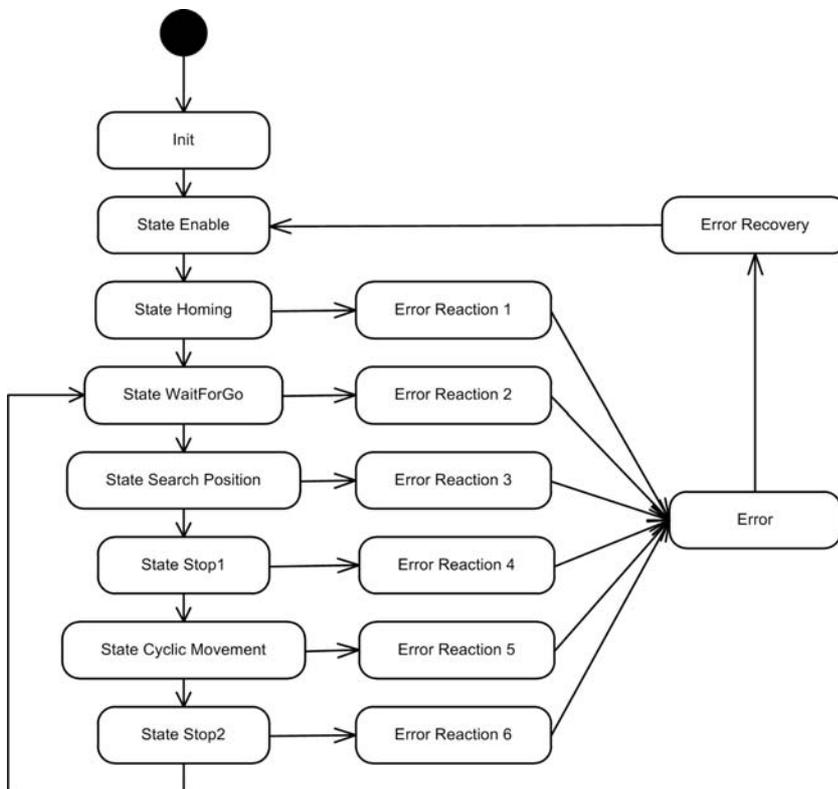


Figure 4-54 Cyclic Motion – Main State Machine

State	Description
Init	The initialization state resetting all digital outputs and axis-related errors and sets the EPOS2 to state "Disabled"
State Enable	A process state setting the EPOS2 to state "Enabled"
State Homing	A process state executing a homing procedure with home position = 0
State WaitForGo	A process state reading digital input 2
State Search Position	A process state starting a continuous motion and stops it if the actual current is exceeding the defined current threshold
State Stop1	A process state stopping the axis and stores the standstill position
State Cyclic Movement	A process state implementing a cyclic motion between two positions, shows the cyclic motion with digital output 2, and leaves the state if a positive edge at digital input 3 has been detected
State Stop2	A process state stopping the axis
Error Reaction 1	An error reaction state being entered when an error is detected in state "Homing"
Error Reaction 2	An error reaction state being entered when an error is detected in state "WaitForGo"
Error Reaction 3	An error reaction state being entered when an error is detected in state "Search Position"
Error Reaction 4	An error reaction state being entered when an error is detected in state "Stop1"
Error Reaction 5	An error reaction state being entered when an error is detected in state "Cyclic Movement"
Error Reaction 6	An error reaction state being entered when an error is detected in state "Stop2"
Error	After an error reaction, the process remains in "Error" state until the error is acknowledged
Error Recover	After acknowledgment, the process is recovered and restarted

Table 4-43 Cyclic Motion – Main State Machine (States)

Transition	Description
Init → State Enable	Done = All digital outputs and axis-related errors are reset and the EPOS2 is set to the state "Disabled"
State Enable → State Homing	Done = Value of the digital input 1 has been set
State Homing → State WaitForGo	Done = Homing procedure is correctly finished
State WaitForGo → State Search Position	Done = Value of the digital input 2 has been set
State Search Position → State Stop1	Done = A continuous motion is started and the actual current exceeding the defined current threshold
State Stop1 → State Cyclic Motion	Done = The axis is standstill
State Cyclic Motion → State Stop2	Done = Value of the digital input 3 has been set
State Stop2 → State WaitForGo	Done = The axis is standstill
State Homing → Error Reaction 1	ErrorHandling State = Error Reaction
State WaitForGo → Error Reaction 2	ErrorHandling State = Error Reaction
State Search Position → Error Reaction 3	ErrorHandling State = Error Reaction
State Stop1 → Error Reaction 4	ErrorHandling State = Error Reaction
State Cyclic Motion → Error Reaction 5	ErrorHandling State = Error Reaction
State Stop2 → Error Reaction 6	ErrorHandling State = Error Reaction
Error Reaction 1 → Error	Done = The axis was stopped
Error Reaction 2 → Error	Done = The axis was stopped
Error Reaction 3 → Error	Done = The axis was stopped
Error Reaction 4 → Error	Done = The axis was stopped
Error Reaction 5 → Error	Done = The axis was stopped
Error Reaction 6 → Error	Done = The axis was stopped
Error → Error Recovery	ErrorHandling State = Error Recovery
Error Recovery → State Enable	Done = All digital outputs and all axis-related errors are reset and the global variables for error handling are set to the initial condition

Table 4-44 Cyclic Motion – Main State Machine (Transitions)

4.1.2.2 Error Handling State Machine

The Error Handling State Machine monitors the process and coordinates error reactions and error recoveries of the processes. Error handling is implemented as a separate state machine allowing the implementation of more than one process state machines.

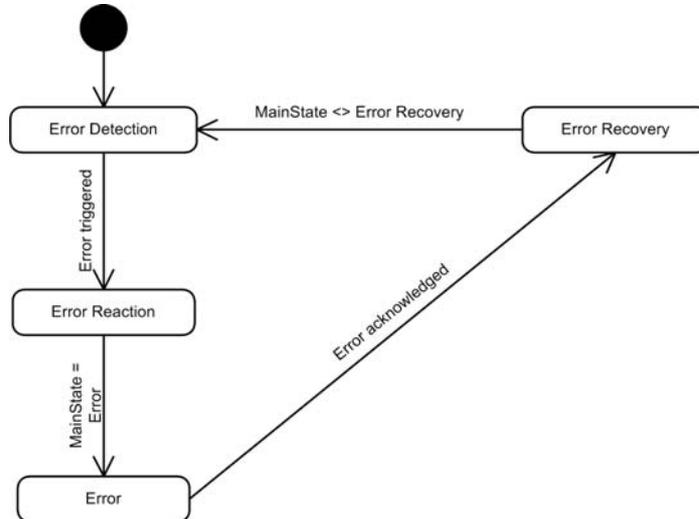


Figure 4-55 Cyclic Motion – Error Handling State Machine

State	Description
Error Detection	Monitors the Main State Machine and all devices involved
Error Reaction	Signals to the Main State Machine that an error reaction must be started. The Error Handling State Machine remains in this state until all processes have accomplished their error reactions.
Error	Entered when the Main State Machine has finished the error reaction and showing all error information
Error Recovery	Signals to the Main State Machines that the error is acknowledged and the process can be restarted. The Error Handling State Machine remains in this state until all processes have been restarted.

Table 4-45 Cyclic Motion – Error Handling State Machine (States)

Transition	Description
Error Detection → Error Reaction	An axis-related error or communication error detected
Error Reaction → Error	Process State = Error
Error → Error Recovery	Variable “oErrorRecovery” = FALSE → TRUE
Error Recovery → Error Detection	Process State ≠ Error Recovery

Table 4-46 Cyclic Motion – Error Handling State Machine (Transitions)

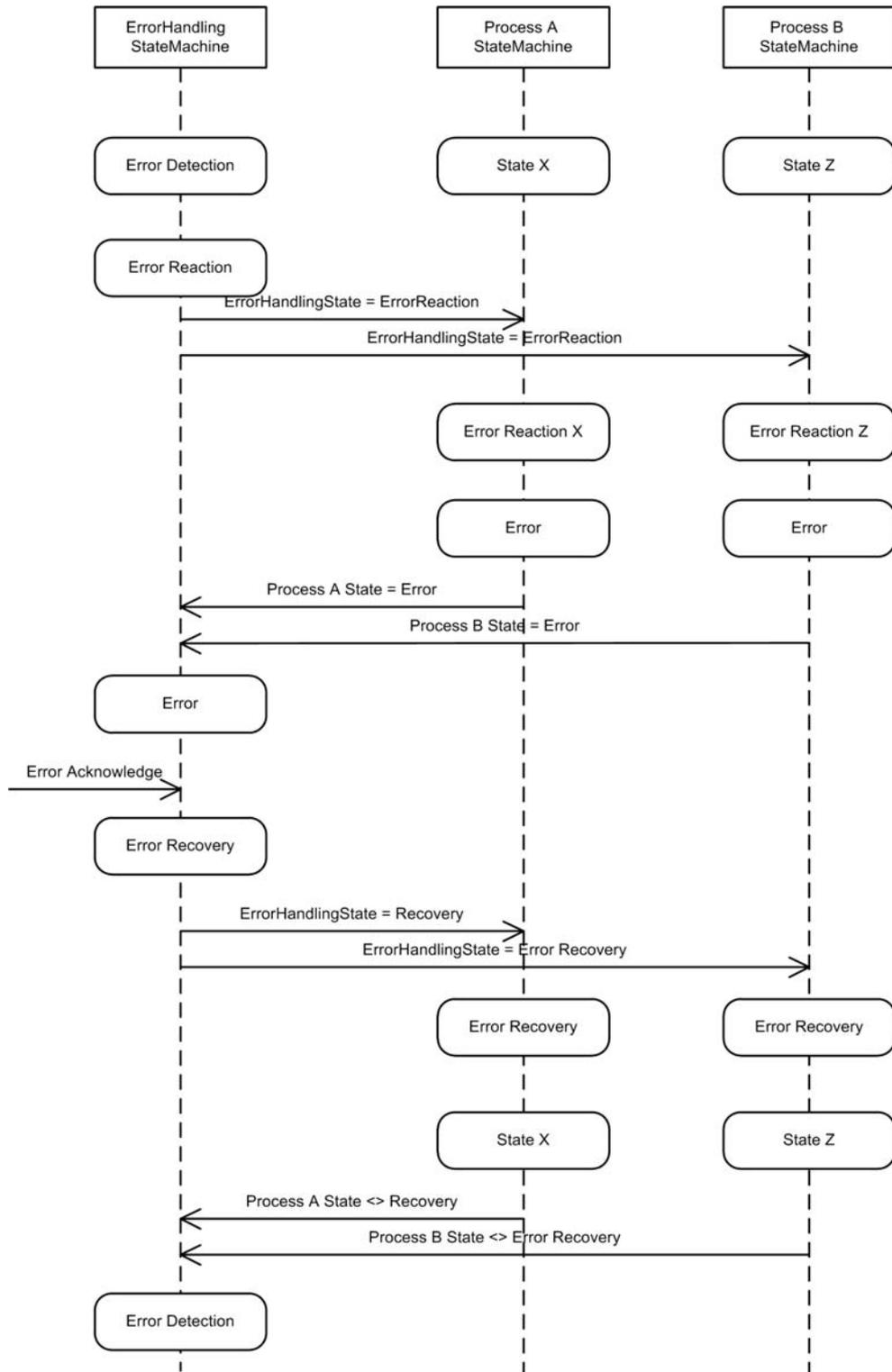


Figure 4-56 Cyclic Motion – State Machine Interactions for more than one Process



Note

For an example on how to handle the «EPOS Studio» during application programming →chapter “2 Getting Started” on page 2-9.

4.2 «I/O Mode»

4.2.1 In Brief

The example describes I/O-triggered motions with one axis.

4.2.2 Functionality

The example consists of two state machines. Thereby, one state machine implements the application process while the other implements error handling. Different functionalities are started and stopped with digital inputs. Digital outputs signal the actual state.

4.2.2.1 Main State Machine

The Main State Machine is an example for a motion sequence application process.

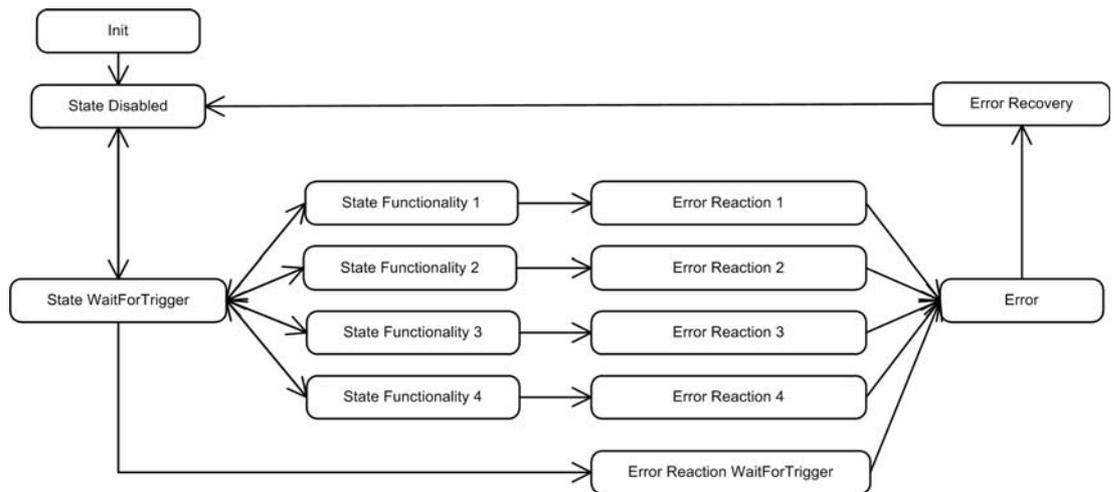


Figure 4-57 I/O Mode – Main State Machine

State	Description
Init	The initialization state resetting all digital outputs and axis-related errors and sets the EPOS2 to state "Disabled"
State Disabled	A process state disabling the axis power stage. Leaving this state, the power stage is enabled.
State WaitForTrigger	A process state switching to the respective functionality state by reading digital inputs or sets the EPOS2 to state "Disabled"
State Functionality 1	A process state executing a homing procedure
State Functionality 2	A process state starting a relative movement with 5000 qc
State Functionality 3	A process state starting a continuous movement with 1000 rpm
State Functionality 4	A process state stopping the axis
Error Reaction 1	An error reaction state being entered when an error is detected in State "Functionality 1"
Error Reaction 2	An error reaction state being entered when an error is detected in State "Functionality 2"
Error Reaction 3	An error reaction state being entered when an error is detected in State "Functionality 3"
Error Reaction 4	An error reaction state being entered when an error is detected in State "Functionality 4"
Error Reaction WaitForTrigger	An error reaction state being entered when an error is detected in State Functionality "WaitForTrigger"
Error	After an error reaction, the process remains in "Error" state until the error is acknowledged
Error Recovery	After acknowledgment, the process is recovered and restarted

Table 4-47 I/O Mode – Main State Machine (States)

Transition		Description
Init	→ State Disabled	Done = All digital outputs and axis-related errors are reset and the EPOS2 is set to the state "Disabled"
State Disabled	→ State WaitForTrigger	Done = Value of the digital input 1 has been set. The EPOS2 is set to State "Enabled".
State WaitForTrigger	→ State Disabled	Done = Value of the digital input 1 has been reset
State WaitForTrigger	→ State Functionality 1	Done = FunctionalityNumber = 2#0001 and digital input 2 (Trigger) has a positive edge
State WaitForTrigger	→ State Functionality 2	Done = FunctionalityNumber = 2#0010 and digital input 2 (Trigger) has a positive edge
State WaitForTrigger	→ State Functionality 3	Done = FunctionalityNumber = 2#0011 and digital input 2 (Trigger) has a positive edge
State WaitForTrigger	→ State Functionality 4	Done = FunctionalityNumber = 2#0100 and digital input 2 (Trigger) has a positive edge
State Functionality 1	→ State WaitForTrigger	Done = Homing procedure has correctly finished
State Functionality 2	→ State WaitForTrigger	Done = Relative movement has correctly started
State Functionality 3	→ State WaitForTrigger	Done = Continuous movement has correctly started
State Functionality 4	→ State WaitForTrigger	Done = The axis is in standstill
State WaitForTrigger	→ Error Reaction WaitForTrigger	ErrorHandling State = Error Reaction
State Functionality 1	→ Error Reaction 1	ErrorHandling State = Error Reaction
State Functionality 2	→ Error Reaction 2	ErrorHandling State = Error Reaction
State Functionality 3	→ Error Reaction 3	ErrorHandling State = Error Reaction
State Functionality 4	→ Error Reaction 4	ErrorHandling State = Error Reaction
Error Reaction WaitForTrigger	→ Error	Done = The axis was stopped
Error Reaction 1	→ Error	Done = The axis was stopped
Error Reaction 2	→ Error	Done = The axis was stopped
Error Reaction 3	→ Error	Done = The axis was stopped
Error Reaction 4	→ Error	Done = The axis was stopped
Error	→ Error Recovery	ErrorHandling State = Error Recovery
Error Recovery	→ State Disabled	Done = All digital outputs and all axis-related errors are reset and the global variables for error handling are set to the initial condition

Table 4-48 I/O Mode – Main State Machine (Transitions)

4.2.2.2 Error Handling State Machine

The Error Handling State Machine monitors the process and coordinates error reactions and error recoveries of the processes. Error handling is implemented as a separate state machine allowing the implementation of more than one process state machines.

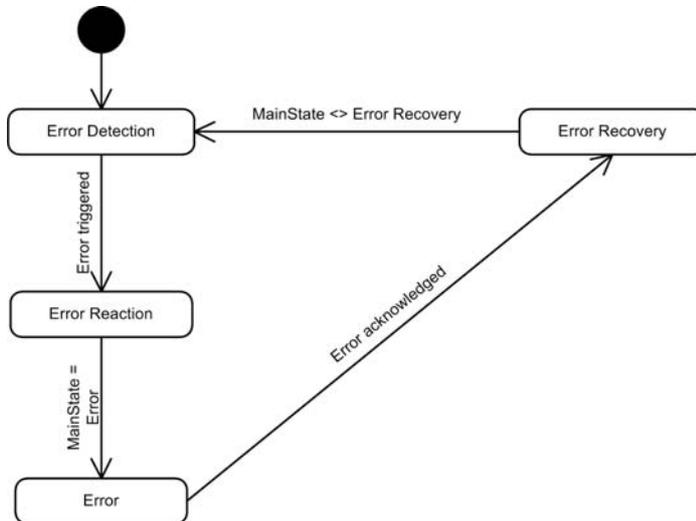


Figure 4-58 I/O Mode – Error Handling State Machine

State	Description
Error Detection	Monitors the Main State Machine and all devices involved
Error Reaction	Signals to the Main State Machine that an error reaction must be started. The Error Handling State Machine remains in this state until all processes have accomplished their error reactions.
Error	Entered when the Main State Machine has finished the error reaction and showing all error information
Error Recovery	Signals to the Main State Machines that the error is acknowledged and the process can be restarted. The Error Handling State Machine remains in this state until all processes have been restarted.

Table 4-49 I/O Mode – Error Handling State Machine (States)

Transition	Description
Error Detection → Error Reaction	An axis-related error or communication error detected
Error Reaction → Error	Process State = Error
Error → Error Recovery	Variable “oErrorRecovery” = FALSE → TRUE
Error Recovery → Error Detection	Process State ≠ Error Recovery

Table 4-50 I/O Mode – Error Handling State Machine (Transitions)

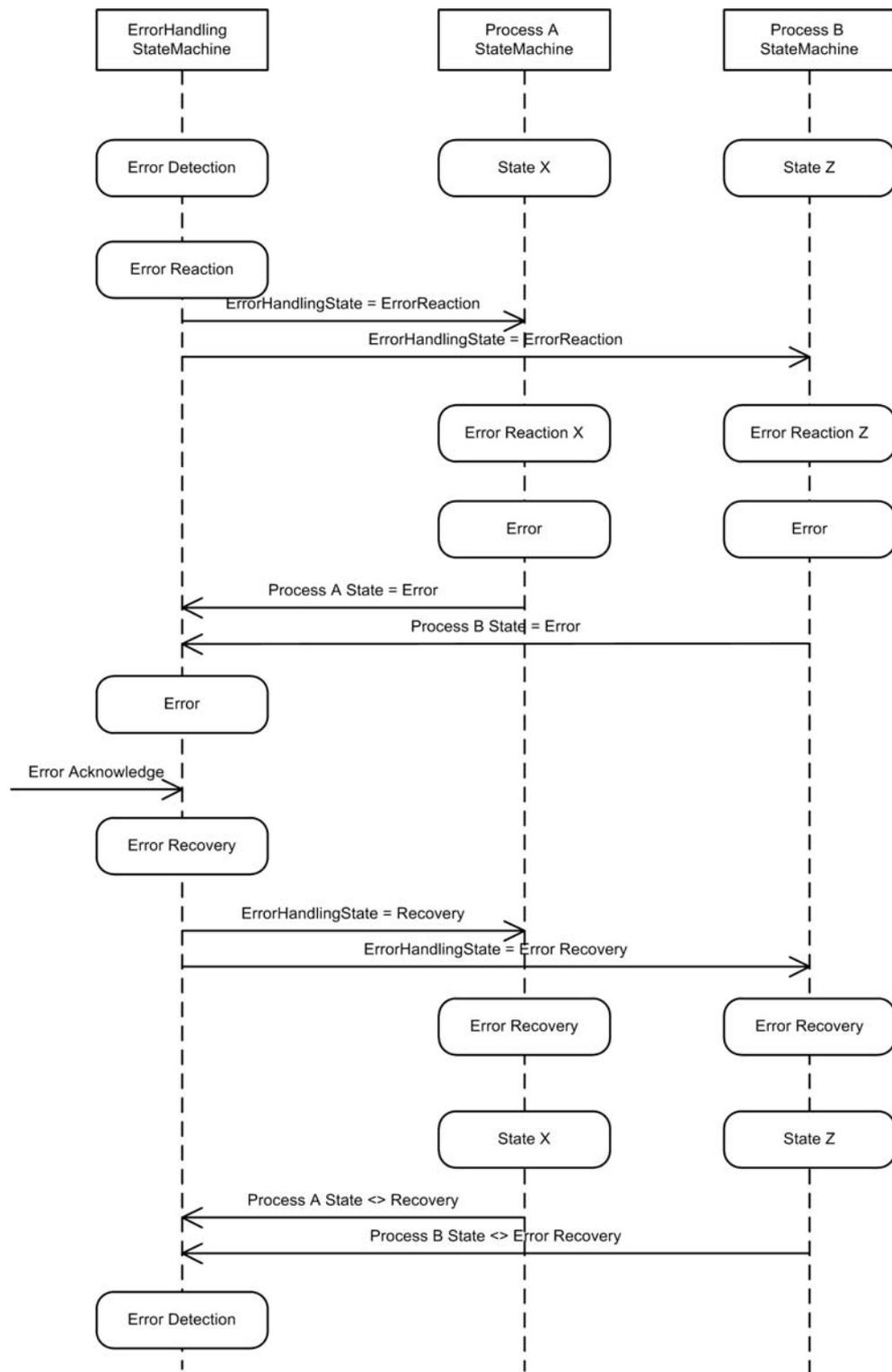


Figure 4-59 I/O Mode – State Machine Interactions for more than one Process



Note

For an example on how to handle the «EPOS Studio» during application programming →chapter “2 Getting Started” on page 2-9.

4.3 «Multiaxis Motion»

4.3.1 In Brief

The example demonstrates how to implement coordinated motions with two axes.

4.3.2 Functionality

The example consists of two state machines. One state machine implements the application process and the second implements the error handling.

The example shows how to coordinate motions with more than one axis. A homing procedure is done for both axes during the initialization. Afterwards, the axes perform relative movements alternately. After two motions, both axes drive back to their initial positions at the same time. Error handling stops the axes if an error is detected.

4.3.2.1 Main State Machine

The Main State Machine is an example for coordinated motions with two axes.

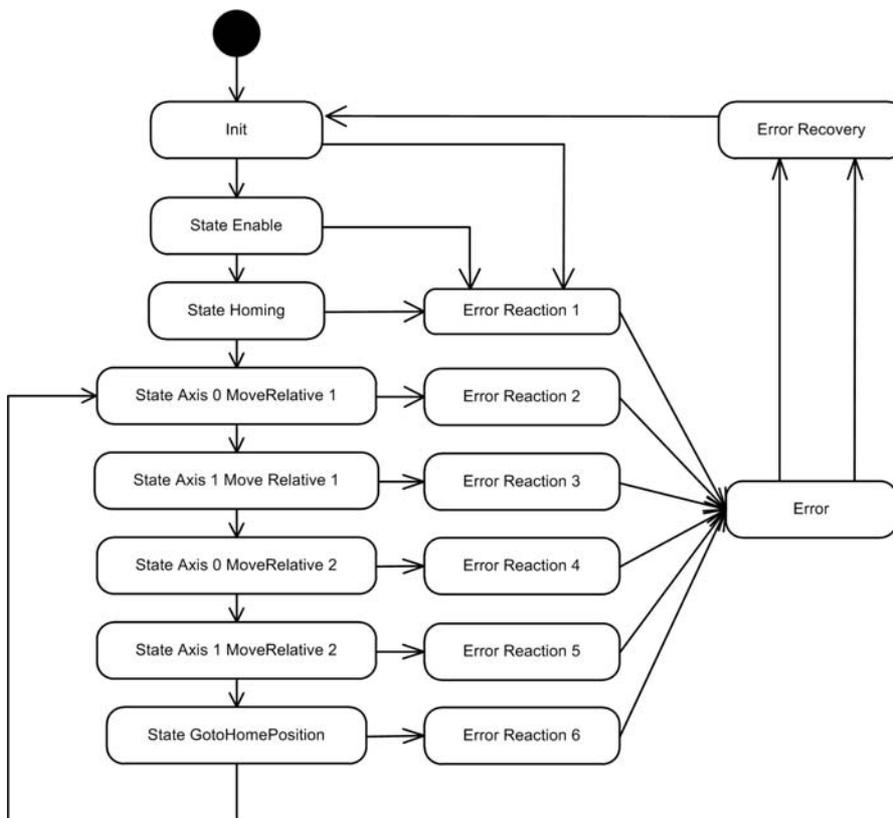


Figure 4-60 Multiaxis Motion – Main State Machine

State	Description
Init	The initialization state resetting all digital outputs and axis-related errors and sets the EPOS2 to state "Disabled"
State Enable	A process state setting the EPOS2 to the state "Enabled"
State Homing	A process state executing a homing procedure for both axes with home position = 0
State Axis 0 MoveRelative 1	A process state starting a relative positioning of the axis 0 to the position 5000
State Axis 1 MoveRelative 1	A process state starting a relative positioning of the axis 1 to the position 5000
State Axis 0 MoveRelative 2	A process state, starting a relative positioning of the axis 0 to the position 10000
State Axis 1 MoveRelative 2	A process state starting a relative positioning of the axis 1 to the position 10000
State GotoHomePosition	A process state starting an absolute positioning for both axes to drive back to their initial positions at the same time
Error Reaction 1	An error reaction state being entered when an error is detected in State "Homing"
Error Reaction 2	An error reaction state being entered when an error is detected in State "Axis0_MoveRelative1"
Error Reaction 3	An error reaction state being entered when an error is detected in State "Axis1_MoveRelative1"
Error Reaction 4	An error reaction state being entered when an error is detected in State "Axis0_MoveRelative2"
Error Reaction 5	An error reaction state being entered when an error is detected in State "Axis1_MoveRelative2"
Error Reaction 6	An error reaction state being entered when an error is detected in State "GotoHomePosition"
Error	After an error reaction, the process remains in "Error" state until the error is acknowledged
Error Recovery	After acknowledgment, the process is recovered and restarted

Table 4-51 Multiaxis Motion – Main State Machine (States)

Transition		Description
Init	→ State Enable	Done = All digital outputs and axis-related errors are reset and the EPOS2 is set to the state "Enable"
State Enable	→ State Homing	Done = Value of digital input 1 has been set
State Homing	→ State Axis 0 MoveRelative 1	Done = Homing procedure has correctly finished
State Axis 0 MoveRelative 1	→ State Axis 1 MoveRelative 1	Done = The first relative positioning of axis 0 has correctly finished
State Axis 1 MoveRelative 1	→ State Axis 0 MoveRelative 2	Done = The first relative positioning of axis 1 has correctly finished
State Axis 0 MoveRelative 2	→ State Axis 1 MoveRelative 2	Done = The second relative positioning of axis 0 has correctly finished
State Axis 1 MoveRelative 2	→ State GotoHomePosition	Done = The second relative positioning of axis 1 has correctly finished
State GotoHomePosition	→ State Axis 0 MoveRelative 1	Done = Both absolute positioning operations are correctly finished
State Homing	→ Error Reaction 1	ErrorHandling State = Error Reaction
State Axis 0 MoveRelative 1	→ Error Reaction 2	ErrorHandling State = Error Reaction
State Axis 1 MoveRelative 1	→ Error Reaction 3	ErrorHandling State = Error Reaction
State Axis 0 MoveRelative 2	→ Error Reaction 4	ErrorHandling State = Error Reaction
State Axis 1 MoveRelative 2	→ Error Reaction 5	ErrorHandling State = Error Reaction
State GotoHomePosition	→ Error Reaction 6	ErrorHandling State = Error Reaction
Error Reaction 1	→ Error	Done = The axis was stopped
Error Reaction 2	→ Error	Done = The axis was stopped
Error Reaction 3	→ Error	Done = The axis was stopped
Error Reaction 4	→ Error	Done = The axis was stopped
Error Reaction 5	→ Error	Done = The axis was stopped
Error Reaction 6	→ Error	Done = The axis was stopped
Error	→ Error Recovery	ErrorHandling State = Error Recovery
Error Recovery	→ Init	Done = All digital outputs and all axis-related errors are reset and the global variables for error handling are set to the initial condition

Table 4-52 Multiaxis Motion – Main State Machine (Transitions)

4.3.2.2 Error Handling State Machine

The Error Handling State Machine monitors the process and coordinates error reactions and error recoveries of the processes. Error handling is implemented as a separate state machine allowing the implementation of more than one process state machines.

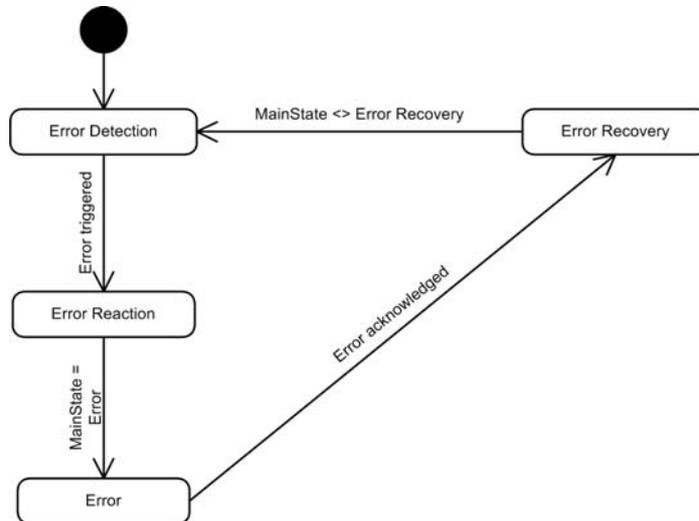


Figure 4-61 Multiaxis Motion – Error Handling State Machine

State	Description
Error Detection	Monitors the Main State Machine and all devices involved
Error Reaction	Signals to the Main State Machine that an error reaction must be started. The Error Handling State Machine remains in this state until all processes have accomplished their error reactions.
Error	Entered when the Main State Machine has finished the error reaction and showing all error information
Error Recovery	Signals to the Main State Machines that the error is acknowledged and the process can be restarted. The Error Handling State Machine remains in this state until all processes have been restarted.

Table 4-53 Multiaxis Motion – Error Handling State Machine (States)

Transition	Description
Error Detection → Error Reaction	An axis-related error or communication error detected
Error Reaction → Error	Process State = Error
Error → Error Recovery	Variable “oErrorRecovery” = FALSE → TRUE
Error Recovery → Error Detection	Process State ≠ Error Recovery

Table 4-54 Multiaxis Motion – Error Handling State Machine (Transitions)

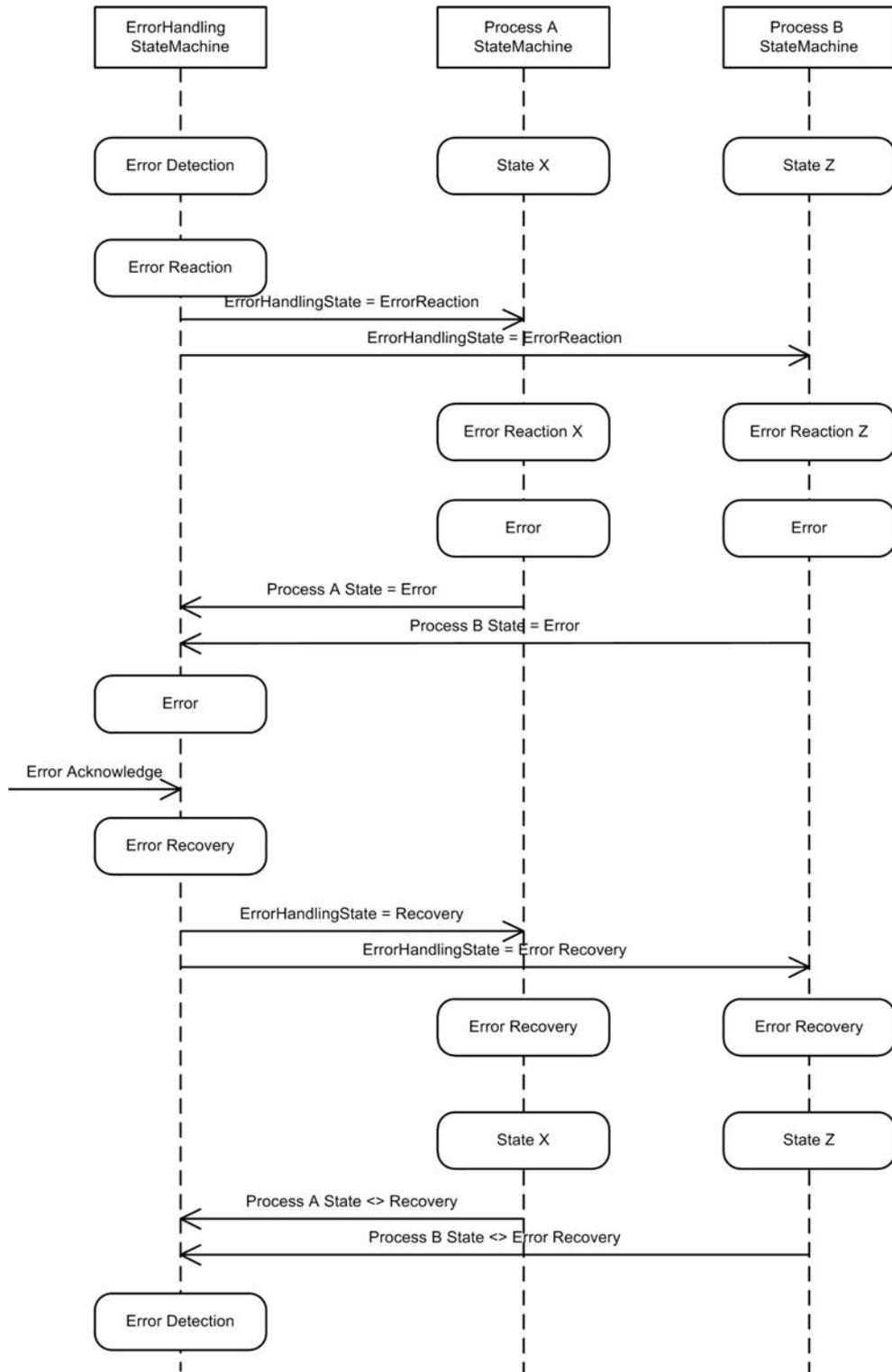


Figure 4-62 Multiaxis Motion – State Machine Interactions for more than one Process



Note

For an example on how to handle the «EPOS Studio» during application programming → chapter “2 Getting Started” on page 2-9.

4.4 «Process Input Output»

4.4.1 In Brief

The example demonstrates a typical supervisory control application. As a supervisor program, a HMI application based on the Windows DLL is used. The supervisor controls a cyclic movement between two positions. The amplitude of the cycle can be configured. The cycle movements can be started and stopped. The status of the cycle movement is read and displayed on the graphical user interface.

4.4.2 Functionality

The example consists of two state machines. One state machine implements the application process and the second implements the error handling.

The program can be controlled by writing the process input variables. A positive edge of the process variable “StartCycles” initiates a cyclic movement between two positions. The amplitude of the movement can be configured by writing the process variable “Amplitude”. In case of error, a positive edge of the process variable “ResetError” can be written to acknowledge the error. The output process variables “Running”, “Error”, and “Count” can be used to monitor the cycle movement.

4.4.2.1 Main State Machine

The Main State Machine is an example for an application process controlled by a supervisor.

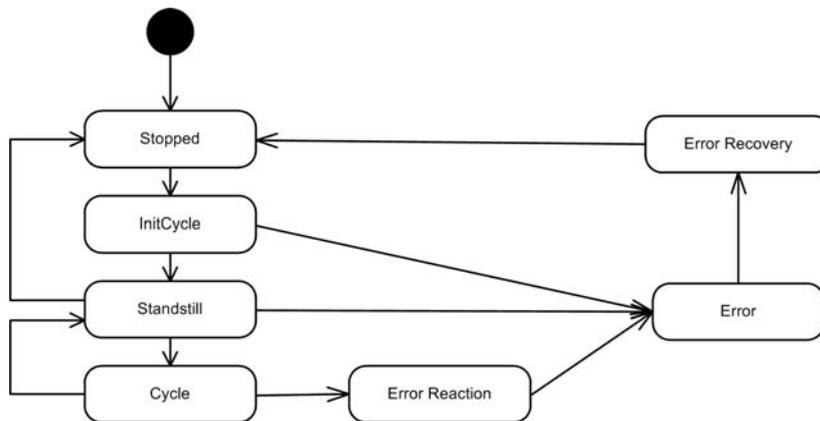


Figure 4-63 Process Input Output – Main State Machine

State	Description
Stopped	A process state setting the EPOS2 to state “Disabled”. A positive edge of variable “StartCycles” triggers a transition to state “InitCycle”
InitCycle	A process state setting the EPOS2 to state “Enabled” and prepares the application for the cycle movement
Standstill	A process state waiting for 1 second between two sequential cycles. A negative edge of variable “StartCycles” triggers a transition to state “Stopped”.
Cycle	A process state starting two sequential positioning operations. The second positioning is delayed by 1 second.
Error Reaction	An error reaction state being entered as an error is detected in “Cycle”
Error	After an error reaction, the process remains in “Error” state until the error is acknowledged. A positive edge of variable “ResetError” triggers a transition to state “ErrorRecovery”.
Error Recover	After acknowledgment, the process is recovered and restarted

Table 4-55 Process Input Output – Main State Machine (States)

Transition		Description
Stopped	→ InitCycle	A positive edge of variable "StartCycles"
InitCycle	→ Standstill	Initialization of the cycle movement is done
Standstill	→ Cycle	Timer of 1 s has elapsed
Cycle	→ Standstill	Two positioning movements are executed
Standstill	→ Stopped	A negative edge of variable "StartCycles"
Cycle	→ Error Reaction	ErrorHandling State = Error Reaction
Stopped	→ Error	ErrorHandling State = Error Reaction
InitCycle	→ Error	ErrorHandling State = Error Reaction
Standstill	→ Error	ErrorHandling State = Error Reaction
Error Reaction	→ Error	ErrorHandling State = Error Reaction
Error	→ Error Recovery	ErrorHandling State = Error Recovery
Error Recovery	→ Stopped	Done = All axis-related errors are reset and the global variables for error handling are set to the initial condition

Table 4-56 Process Input Output – Main State Machine (Transitions)

4.4.2.2 Error Handling State Machine

The Error Handling State Machine monitors the process and coordinates error reactions and error recoveries of the processes. Error handling is implemented as a separate state machine allowing the implementation of more than one process state machines.

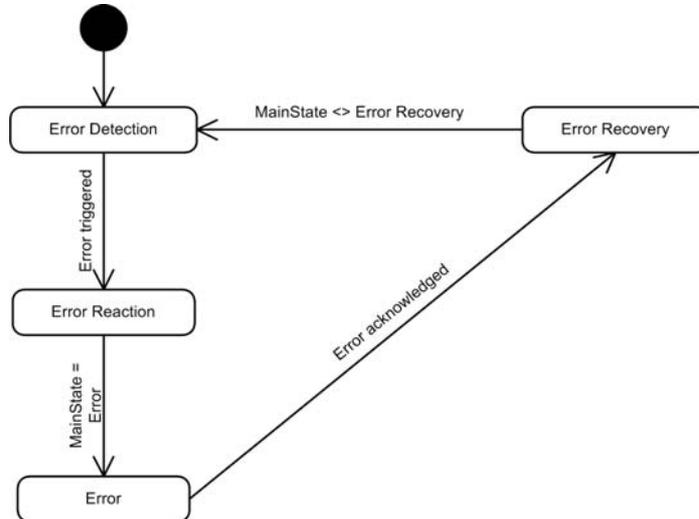


Figure 4-64 Process Input Output – Error Handling State Machine

State	Description
Error Detection	Monitors the Main State Machine and all devices involved
Error Reaction	Signals to the Main State Machine that an error reaction must be started. The Error Handling State Machine remains in this state until all processes have accomplished their error reactions.
Error	Entered when the Main State Machine has finished the error reaction and showing all error information
Error Recovery	Signals to the Main State Machines that the error is acknowledged and the process can be restarted. The Error Handling State Machine remains in this state until all processes have been restarted.

Table 4-57 Process Input Output – Error Handling State Machine (States)

Transition	Description
Error Detection → Error Reaction	An axis-related error or communication error detected
Error Reaction → Error	Process State = Error
Error → Error Recovery	A positive edge of variable “ResetError”
Error Recovery → Error Detection	Process State ≠ Error Recovery

Table 4-58 Process Input Output – Error Handling State Machine (Transitions)

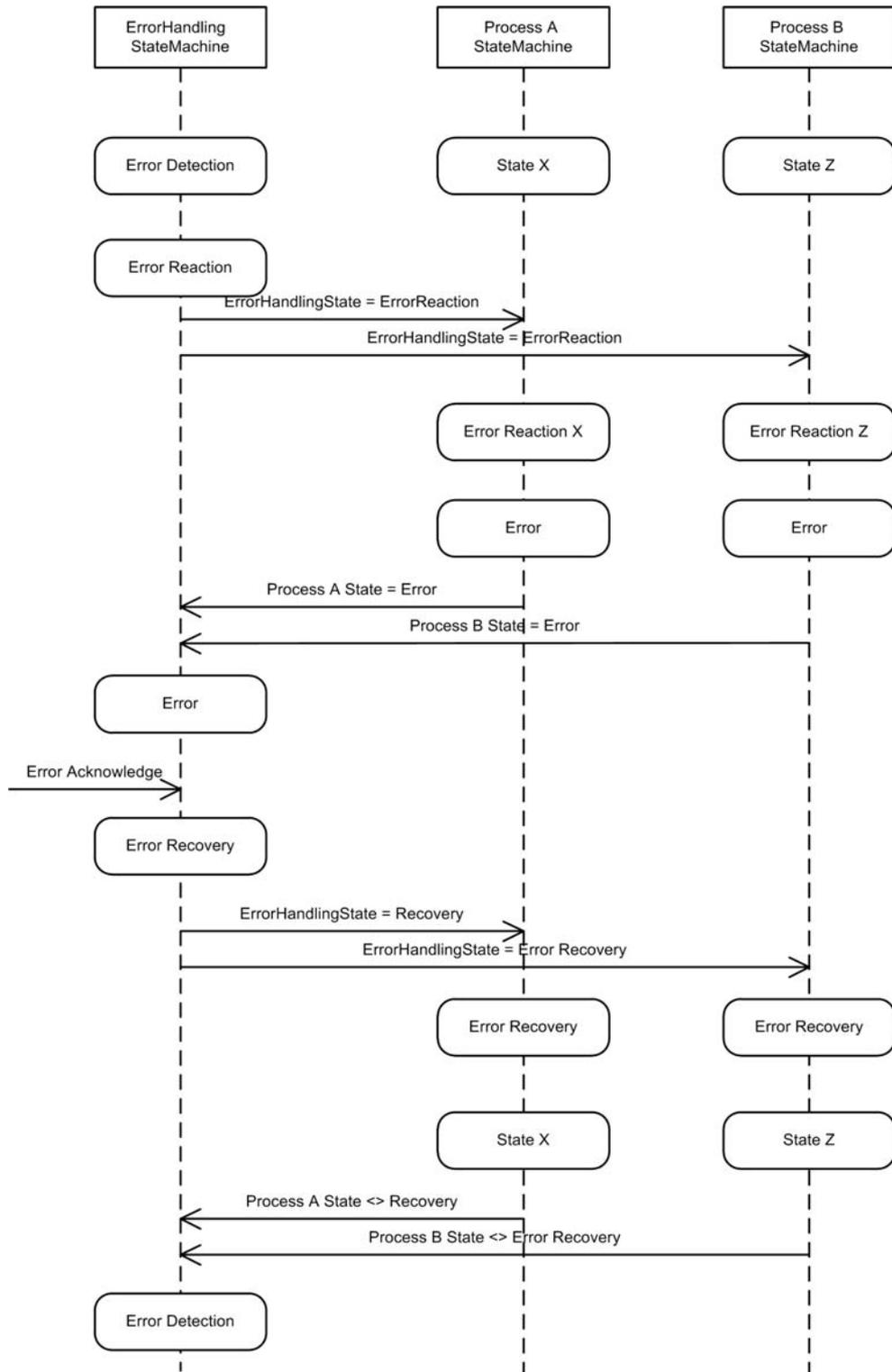


Figure 4-65 Process Input Output – State Machine Interactions for more than one Process



Note

For an example on how to handle the «EPOS Studio» during application programming → chapter “2 Getting Started” on page 2-9.

LIST OF FIGURES

Figure 1-1 Documentation Structure6

Figure 2-2 Page Navigator – Communication9

Figure 2-3 Communication Properties9

Figure 2-4 Connect All10

Figure 2-5 Application Adjustment10

Figure 2-6 Page Navigator – Parameter Export/Import10

Figure 2-7 Parameter Export/Import11

Figure 2-8 Page Navigator – Startup Wizard11

Figure 2-9 Page Navigator – Regulation Tuning12

Figure 2-10 Page Navigator – IEC-61131 Programming13

Figure 2-11 IEC-61131 Programming Tool13

Figure 2-12 Browse IEC-61131 Project14

Figure 2-13 IEC-61131 Project14

Figure 2-14 Connection Setup15

Figure 2-15 Edit Connection15

Figure 2-16 Build Active Resource15

Figure 2-17 Build Result15

Figure 2-18 Go Online16

Figure 2-19 Download Program16

Figure 2-20 Coldstart16

Figure 2-21 Project Explorer17

Figure 2-22 Monitor/Edit17

Figure 2-23 Watch Variable in Source Code17

Figure 2-24 Project Explorer Resources17

Figure 2-25 Watch Variables for State Changes18

Figure 2-26 Watchlist Resources18

Figure 3-27 State Machine – Main State Machine20

Figure 3-28 State Machine – Error Handling State Machine22

Figure 3-29 State Machine – State Machine Interactions for more than one Process23

Figure 3-30 Error Handling – Main State Machine24

Figure 3-31 Error Handling – Error Handling State Machine26

Figure 3-32 Error Handling – State Machine Interactions for more than one Process27

Figure 3-33 Input/Output Handling – Main State Machine28

Figure 3-34 Input/Output Handling – Error Handling State Machine30

Figure 3-35 Input/Output Handling – State Machine Interactions for more than one Process31

Figure 3-36 Homing – Main State Machine32

Figure 3-37 Homing – Error Handling State Machine34

Figure 3-38 Homing – State Machine Interactions for more than one Process35

Figure 3-39 Positioning – Main State Machine36

Figure 3-40 Positioning – Error Handling State Machine38

Figure 3-41 Positioning – State Machine Interactions for more than one Process39

Figure 3-42 Continuous Motion – Main State Machine40

Figure 3-43	Continuous Motion – Error Handling State Machine	42
Figure 3-44	Continuous Motion – State Machine Interactions for more than one Process	43
Figure 3-45	Read Actual Value – Main State Machine	44
Figure 3-46	Read Actual Value – Error Handling State Machine	45
Figure 3-47	Read Actual Value – State Machine Interactions for more than one Process	46
Figure 3-48	Object Dictionary Access – Main State Machine	47
Figure 3-49	Object Dictionary Access – Error Handling State Machine	49
Figure 3-50	Object Dictionary Access – State Machine Interactions for more than one Process	50
Figure 3-51	Data Handling – Main State Machine	51
Figure 3-52	Data Handling – Error Handling State Machine	53
Figure 3-53	Data Handling – State Machine Interactions for more than one Process	54
Figure 4-54	Cyclic Motion – Main State Machine	56
Figure 4-55	Cyclic Motion – Error Handling State Machine	59
Figure 4-56	Cyclic Motion – State Machine Interactions for more than one Process	60
Figure 4-57	I/O Mode – Main State Machine	61
Figure 4-58	I/O Mode – Error Handling State Machine	64
Figure 4-59	I/O Mode – State Machine Interactions for more than one Process	65
Figure 4-60	Multiaxis Motion – Main State Machine	66
Figure 4-61	Multiaxis Motion – Error Handling State Machine	69
Figure 4-62	Multiaxis Motion – State Machine Interactions for more than one Process	70
Figure 4-63	Process Input Output – Main State Machine	71
Figure 4-64	Process Input Output – Error Handling State Machine	73
Figure 4-65	Process Input Output – State Machine Interactions for more than one Process	74

LIST OF TABLES

Table 1-1	Notations used in this Document	6
Table 1-2	Brand Names and Trademark Owners	8
Table 3-3	Best Practice Examples – covered Hardware and required Documents	19
Table 3-4	Best Practice Examples – recommended Tools	19
Table 3-5	State Machine – Main State Machine (States)	20
Table 3-6	State Machine – Main State Machine (Transitions)	21
Table 3-7	State Machine – Error Handling State Machine (States)	22
Table 3-8	State Machine – Error Handling State Machine (Transitions)	22
Table 3-9	Error Handling – Main State Machine (States)	24
Table 3-10	Error Handling – Main State Machine (Transitions)	25
Table 3-11	Error Handling – Error Handling State Machine (States)	26
Table 3-12	Error Handling – Error Handling State Machine (Transitions)	26
Table 3-13	Input/Output Handling – Main State Machine (States)	28
Table 3-14	Input/Output Handling – Main State Machine (Transitions)	29
Table 3-15	Input/Output Handling – Error Handling State Machine (States)	30
Table 3-16	Input/Output Handling – Error Handling State Machine (Transitions)	30
Table 3-17	Homing – Main State Machine (States)	33
Table 3-18	Homing – Main State Machine (Transitions)	33
Table 3-19	Homing – Error Handling State Machine (States)	34
Table 3-20	Homing – Error Handling State Machine (Transitions)	34
Table 3-21	Positioning – Main State Machine (States)	37
Table 3-22	Positioning – Main State Machine (Transitions)	38
Table 3-23	Positioning – Error Handling State Machine (States)	38
Table 3-24	Positioning – Error Handling State Machine (Transitions)	38
Table 3-25	Continuous Motion – Main State Machine (States)	41
Table 3-26	Continuous Motion – Main State Machine (Transitions)	41
Table 3-27	Continuous Motion – Error Handling State Machine (States)	42
Table 3-28	Continuous Motion – Error Handling State Machine (Transitions)	42
Table 3-29	Read Actual Value – Main State Machine (States)	44
Table 3-30	Read Actual Value – Main State Machine (Transitions)	44
Table 3-31	Read Actual Value – Error Handling State Machine (States)	45
Table 3-32	Read Actual Value – Error Handling State Machine (Transitions)	45
Table 3-33	Object Dictionary Access – Main State Machine (States)	47
Table 3-34	Object Dictionary Access – Main State Machine (Transitions)	48
Table 3-35	Object Dictionary Access – Error Handling State Machine (States)	49
Table 3-36	Object Dictionary Access – Error Handling State Machine (Transitions)	49
Table 3-37	Data Handling – Main State Machine (States)	51
Table 3-38	Data Handling – Main State Machine (Transitions)	52
Table 3-39	Data Handling – Error Handling State Machine (States)	53
Table 3-40	Data Handling – Error Handling State Machine (Transitions)	53
Table 4-41	Application Examples – covered Hardware and required Documents	55
Table 4-42	Application Examples – recommended Tools	55

Table 4-43	Cyclic Motion – Main State Machine (States)	57
Table 4-44	Cyclic Motion – Main State Machine (Transitions)	58
Table 4-45	Cyclic Motion – Error Handling State Machine (States)	59
Table 4-46	Cyclic Motion – Error Handling State Machine (Transitions)	59
Table 4-47	I/O Mode – Main State Machine (States)	62
Table 4-48	I/O Mode – Main State Machine (Transitions)	63
Table 4-49	I/O Mode – Error Handling State Machine (States)	64
Table 4-50	I/O Mode – Error Handling State Machine (Transitions)	64
Table 4-51	Multiaxis Motion – Main State Machine (States)	67
Table 4-52	Multiaxis Motion – Main State Machine (Transitions)	68
Table 4-53	Multiaxis Motion – Error Handling State Machine (States)	69
Table 4-54	Multiaxis Motion – Error Handling State Machine (Transitions)	69
Table 4-55	Process Input Output – Main State Machine (States)	71
Table 4-56	Process Input Output – Main State Machine (Transitions)	72
Table 4-57	Process Input Output – Error Handling State Machine (States)	73
Table 4-58	Process Input Output – Error Handling State Machine (Transitions)	73

INDEX

A

alerts **7**
applicable EU directive **5**

E

EU directive, applicable **5**

H

how to
interpret icons (and signs) used in the document **7**
read this document **2**

I

incorporation into surrounding system **5**
informatory signs **8**

M

mandatory action signs **7**

N

non-compliance of surrounding system **2**

O

operating license **5**
other machinery (incorporation into) **5**

P

prerequisites prior installation **5**
prohibitive signs **7**
purpose of this document **5**

S

safety alerts **7**
signs
informative **8**
mandatory **7**
prohibitive **7**
signs used **7**
symbols used **7**

© 2016 maxon motor. All rights reserved.

The present document – including all parts thereof – is protected by copyright. Any use (including reproduction, translation, microfilming and other means of electronic data processing) beyond the narrow restrictions of the copyright law without the prior approval of maxon motor ag, is not permitted and subject to persecution under the applicable law.

maxon motor ag

Brünigstrasse 220
P.O.Box 263
CH-6072 Sachseln
Switzerland

Phone +41 41 666 15 00

Fax +41 41 666 16 50

www.maxonmotor.com