

SIEMENS

SIMATIC

S7-HiGraph V5.3 Programming State Graphs

Programming and Operating Manual

Preface

Product overview

1

Installing

2

Designing a program using
the example of a drill

3

User interface

4

Working with S7-HiGraph

5

Process error diagnostics

6

User program runtime
on the PLC

7

STL language description

8

Example configurations

9

Control elements

10

Technical support

11

Safety Guidelines

This manual contains notices which you should observe to ensure your own personal safety as well as to avoid property damage. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring to property damage only have no safety alert symbol.



Danger

indicates an imminently hazardous situation which, if not avoided, will result in death or serious injury.



Warning

indicates a potentially hazardous situation which, if not avoided, could result in death or serious injury.



Caution

used with the safety alert symbol indicates a potentially hazardous situation which, if not avoided, may result in minor or moderate injury.

Caution

used without safety alert symbol indicates a potentially hazardous situation which, if not avoided, may result in property damage.

Notice

used without the safety alert symbol indicates a potential situation which, if not avoided, may result in an undesirable result or state.

When several danger levels apply, the notices of the highest level (lower number) are always displayed. If a notice refers to personal damages with the safety alert symbol, then another notice may be added warning of property damage.

Qualified Personnel

The device/system may only be set up and operated in conjunction with this documentation. Only qualified personnel should be allowed to install and work on the equipment. Qualified persons are defined as persons who are authorized to commission, to earth, and to tag circuits, equipment and systems in accordance with established safety practices and standards.

Intended Use

Please note the following:



Warning

This device and its components may only be used for the applications described in the catalog or technical description, and only in connection with devices or components from other manufacturers approved or recommended by Siemens.

This product can only function correctly and safely if it is transported, stored, set up and installed correctly, and operated and maintained as recommended.

Trademarks

All designations marked with ® are registered trademarks of Siemens AG. Other designations in this documentation might be trademarks which, if used by third parties for their purposes, might infringe upon the rights of the proprietors.

Copyright Siemens AG, 2004. All rights reserved

Reproduction, transmission or use of this document or its contents is not permitted without express written authority. Offenders will be liable for damages. All rights, including rights created by patent grant or registration of a utility model or design, are reserved.

Disclaimer of Liability

We have checked the contents of this manual for agreement with the hardware and software described. Since deviations cannot be precluded entirely, we cannot guarantee full agreement. However, the data in the manual are reviewed regularly, and any necessary corrections will be included in subsequent editions. Suggestions for improvement are welcomed.

Siemens AG
Automation and Drives Group
P.O. Box 4848, D-90327 Nuremberg (Germany)

Siemens AG 2004
Technical data subject to change

Preface

Purpose of this documentation

This manual provides a comprehensive overview of programming with S7-HiGraph. It supports you in installing and commissioning the software. The manual introduces the procedures in creating a user program, and offers information on the layout of user programs and the various language elements.

The readership of this documentation are adequately qualified persons who are working in the areas of programming, engineering, commissioning and servicing of automation systems.

We recommend that you familiarize yourself with the example in Chapter 2 "Designing a Program Based on the Example of a Drill." This chapter shows you an easy way of getting started in programming with S7-HiGraph.

Basics required

A general knowledge of automation technology is required in order to understand this manual.

We also assume that the user is firm in using computers or equipment similar to a (programming devices, for example) under the operating systems MS Windows, MS Windows 2000 Professional or MS Windows XP Professional. S7-HiGraph is based on STEP 7 basic software, so you should also know how to handle the basic software as described in the "Programming with STEP 7 V5.x" manual.

Where this manual applies

The manual applies to the S7-HiGraph programming software, version 5.3.

S7-HiGraph and STEP 7 basic software documentation package

The table below gives you an overview of the STEP 7 and S7-HiGraph documentation:

Manual	Purpose	Order no.
S7-HiGraph basics and references <ul style="list-style-type: none"> • S7-HiGraph for S7-300/400: Programming state graphs 	Basic and reference information explaining how to create a program, and the structure of the user programs and the various language elements.	Element of the S7-HiGraph software
STEP 7 basic knowledge with <ul style="list-style-type: none"> • Getting started and exercises with STEP 7 V5.3 • Programming with STEP 7 V5.3 • Configuring hardware and connections in STEP 7 V5.3 • Migration manual, converting from S5 to S7 	Basic information for technical personnel, describing how to implement control tasks using STEP 7 and S7-300/400 CPUs.	6ES7810-4CA07-8BW0
STEP 7 reference information <ul style="list-style-type: none"> • LAD/FBD/STL for S7-300/400 CPUs manuals • Standard and system functions for S7-300/400 CPUs 	Reference information on the programming languages LAD/FBD/STL and standard and system functions, supplementary to the STEP 7 knowledge base.	6ES7810-4CA07-8BW1

Manual and Online Help

The manual represents an extract of the Online Help. You can easily switch between the manual and the Online Help because of their identical structure.

The manual describes the basics in using S7-HiGraph. The Online Help contains further reference information.

Using the help functions

Online Help system

The Online Help system offers you direct and on-the-spot information. You can quickly and directly look up information without having to search through manuals.

Calling the Online Help

The online help can be called up by various means:

- Select a "Help" menu command. Commands available:
 - Help topics: offers various methods for accessing help information.
 - Context-sensitive Help (F1): Provides information on the selected object, active dialog box or window.
 - Introduction: Provides a brief overview of the application, and its vital features and functional scope.
 - Getting started: summarizes the essential steps in creating your first successful task.
 - Using Help: shows how to find information in the help system.
 - About: provides information on the current software version.
- Click "Help" in a dialog box. The help relating to this dialog box appears.
- In a window or dialog box, position the cursor on the topic on which you require help, then press F1 or select **Help > Context-Sensitive Help**.
- Pick up the question mark cursor from the toolbar and then click the element on which you need help.

Navigating in the Online Help

The Online Help system offers three tabs:

- Click the "Content" tab to access the table of contents. Browse the table of contents to navigate to the relevant topic.
- The "Index" tab is used to find a specific topic, based on an alphabetical search string.
- Click the "Find" tab to access the full text search function. There you can enter your specific search string.

Jumps to related topics

You can jump to related topics using the menu bar of the Online Help. Commands of this menu bar:

- "In section" shows all subheadings of the current topic.
- "Instructions" contains jumps to user instructions associated with the current topic.
- "Examples" contains jumps to examples of the current topic.
- "Basics" contains jumps to background information on the current topic.
- "Options" allows you to navigate up and down in the structure.

Printing the Online Help

- Select **Options > Open PDF** in the Online Help to print the Help in PDF format.

Printing one or several help topics

- Click "Print" in the help topic you want to print.
- To print a group of associated topics, select a book in the help contents and then click "Print."

Table of contents

Preface	iii
1 Product overview	1-1
1.1 Overview of S7-HiGraph	1-1
1.2 S7-HiGraph programming language	1-2
1.3 Functions in S7-HiGraph	1-4
1.4 What's new in V5.3?	1-5
2 Installing	2-1
2.1 Automation License Manager	2-1
2.1.1 License agreements with Automation License Manager	2-1
2.1.2 Installing Automation License Manager	2-3
2.1.3 Rules for handling License Keys	2-4
2.2 Installing S7-HiGraph	2-5
2.2.1 Installing S7-HiGraph	2-5
2.2.2 Running Setup	2-6
2.2.3 Installation information	2-7
2.2.4 Uninstalling S7-HiGraph	2-7
3 Designing a program using the example of a drill	3-1
3.1 Welcome to the S7-HiGraph example for newcomers	3-1
3.2 Requirements for executing the sample program	3-2
3.3 Automation task drilling machine	3-3
3.4 Steps in creating the "Drilling Machine" sample program	3-5
3.5 Step 1: Define the function units and state graphs	3-6
3.6 Step 2: Designing the state graph	3-8
3.7 Step 3: Defining the system signals	3-10
3.8 Step 4: Create a "HiGr_Exp" project in SIMATIC Manager	3-11
3.9 Step 5: Creating a symbol table	3-13
3.10 Step 6: Creating a state graph	3-13
3.11 Step 7: Declaring the variables	3-14
3.12 Step 8: Inserting the states and transitions	3-15
3.13 Step 9: Enter the state name	3-16
3.14 Step 10: Entering actions and transitions	3-16

3.15	Step 11: Creating a graph group.....	3-18
3.16	Step 12: Assigning the current parameters.....	3-21
3.17	Step 13: Compiling the graph group	3-24
3.18	Step 14: Programming OB1	3-25
3.19	Step 15: Downloading the user program	3-27
3.20	Step 16: Debugging the user program.....	3-27
4	User interface	4-1
4.1	User interface.....	4-1
4.2	Customizing the working area.....	4-3
4.2.1	Arranging working windows	4-3
4.2.2	Working with the session memory	4-3
4.2.3	Working with the object-specific session memory	4-4
4.2.4	Setting the colors and fonts for the working windows.....	4-5
4.2.5	Enlarging and reducing the view.....	4-5
4.2.6	Customizing the size of the drawing area.....	4-6
4.2.7	Setting the grid points	4-6
4.2.8	Showing and hiding instructions or features	4-6
5	Working with S7-HiGraph	5-1
5.1	Reusability of state graphs (type / instance concept)	5-1
5.2	Steps in creating a program.....	5-2
5.3	Setting up a STEP 7 project.....	5-3
5.4	Starting S7-HiGraph.....	5-3
5.5	Creating and opening state graphs.....	5-4
5.6	Declaring variables.....	5-5
5.6.1	Meaning of the variable declaration	5-5
5.6.2	The variable declaration view.....	5-6
5.6.3	Declaration sections.....	5-7
5.6.4	Information in the variable detail view.....	5-8
5.6.5	Steps in entering the variable declaration.....	5-8
5.6.6	Default variables	5-9
5.6.7	Interaction between variable declarations and instructions	5-12
5.6.8	Interaction between variable declarations and current parameter assignments	5-13
5.7	Programming the structure of a state graph	5-14
5.7.1	Elements of a state graph	5-14
5.7.2	Rules for the structure of state graphs and groups.....	5-15
5.7.3	Options of aligning graphic objects	5-15
5.7.4	States	5-16
5.7.4.1	Steps in inserting states	5-17
5.7.4.2	Assigning the state name, number and comment	5-17
5.7.4.3	Assigning state features.....	5-18
5.7.4.4	Handling states	5-19

5.7.5	Transitions.....	5-20
5.7.5.1	Steps in inserting transitions.....	5-21
5.7.5.2	Specifying the transition priority.....	5-21
5.7.5.3	Assigning the transition name and comment.....	5-22
5.7.5.4	Assigning transition features.....	5-22
5.7.5.5	Handling transitions	5-23
5.7.6	Permanent instructions	5-25
5.7.6.1	Steps in inserting permanent instructions.....	5-25
5.8	Programming instructions	5-26
5.8.1	Instructions in states and transitions / permanent Instructions.....	5-26
5.8.2	Instruction types	5-27
5.8.3	Rules for entering STL instructions.....	5-28
5.8.4	Settings for STL Instructions.....	5-29
5.8.5	Steps in entering STL instructions	5-30
5.8.6	Programming with absolute or symbolic addresses	5-31
5.8.6.1	Entering symbols	5-32
5.8.6.2	Entering symbolic addresses in the program.....	5-32
5.8.6.3	Selecting the symbolic / absolute address format	5-33
5.9	Programming waiting, monitoring and delay times.....	5-34
5.9.1	Programming waiting times.....	5-34
5.9.1.1	Waiting times.....	5-34
5.9.1.2	To program waiting times.....	5-34
5.9.2	Programming monitoring times.....	5-36
5.9.2.1	Monitoring times.....	5-36
5.9.2.2	To program monitoring times.....	5-36
5.9.3	Programming delay times	5-37
5.9.3.1	Delay times	5-37
5.9.3.2	To program delay times	5-38
5.10	Programming operating modes.....	5-39
5.10.1	Operating modes	5-39
5.10.2	Steps in programming operating modes.....	5-39
5.11	Programming graph groups	5-40
5.11.1	Graph groups	5-40
5.11.2	Steps in programming graph groups	5-42
5.12	Programming the exchange of messages between state graphs.....	5-43
5.12.1	Basics of exchanging messages.....	5-43
5.12.2	Basic procedure of message programming	5-44
5.12.3	Declaration of message variables.....	5-44
5.12.4	Programming statements for messages	5-45
5.12.5	Interconnecting incoming and outgoing messages.....	5-45
5.13	Viewing reference data	5-47
5.13.1	Overview of available reference data	5-47
5.13.2	Generating and viewing reference data.....	5-48
5.13.3	Quick positioning to locations in the program.....	5-48
5.13.4	S7-HiGraph-specific information in the reference data.....	5-50

5.14	Save	5-51
5.15	Compilation	5-52
5.15.1	Compiler settings	5-53
5.16	Calling and loading the S7-HiGraph FC.....	5-54
5.16.1	Calling the FC in an S7 Program	5-54
5.16.2	Example of an OB 1 with FC call	5-55
5.16.3	Requirements for downloading	5-56
5.16.4	Downloading the user program.....	5-56
5.16.5	Downloading the FC and corresponding DB/diagnostic DB	5-57
5.16.6	ONLINE delta downloads.....	5-57
5.17	Debugging the program	5-58
5.17.1	Monitoring the program status	5-58
5.17.1.1	Requirements for starting the program status monitoring function	5-58
5.17.1.2	View in program status.....	5-59
5.17.1.3	Steps in viewing the program status	5-61
5.17.2	Debugging using the STEP 7 debugging functions	5-62
5.17.2.1	Check Block Consistency.....	5-62
5.17.2.2	Monitoring and controlling variables	5-62
5.17.2.3	Evaluating the diagnostic buffer.....	5-63
5.17.2.4	Evaluating CPU messages	5-64
5.18	Printing	5-65
5.18.1	Creating a program documentation	5-65
5.18.2	Steps in printing	5-66
5.18.3	Setting grayscale or color printing.....	5-67
5.18.4	Setting the page format for printing.....	5-67
5.18.5	Setting up headers and footers.....	5-68
5.18.6	Opening the print preview	5-68
5.18.7	Showing page frames	5-68
5.19	Using data of older S7-HiGraph versions	5-69
5.19.1	Converting HiGraph 2.6 / 2.7 programs	5-69
5.19.1.1	Converting individual graph groups	5-69
5.19.1.2	Converting projects	5-70
5.19.2	Using programs created in S7-HiGraph V4.x / 5.0/V5.2	5-71
5.19.2.1	Defining the save format of new sources.....	5-72
5.19.2.2	Determining the current save format.....	5-72
5.19.2.3	Converting sources to V5.3.....	5-72
5.19.2.4	Maintaining sources in their current format.....	5-72
5.19.2.5	Converting source code from V5.3 down to V5.0/V5.2	5-72

6	Process error diagnostics	6-1
6.1	Standard diagnostics with the help of ProTool / ProAgent	6-1
6.1.1	Standard diagnostics functions	6-1
6.1.2	Tools for standard diagnostics	6-2
6.1.3	Interaction between S7-HiGraph - AS - HMI	6-3
6.1.4	Requirements of standard diagnostics	6-4
6.1.5	How to generate standard diagnostics data	6-5
6.1.6	Output of messages to the message screen	6-6
6.1.6.1	Message texts on message screens	6-7
6.1.6.2	How to define message texts in S7-HiGraph	6-9
6.1.6.3	To set the acknowledgment function	6-9
6.1.7	Output of initial and current values to the detail screen	6-10
6.1.7.1	How to define the screen output of initial and current values	6-11
6.1.8	Motion monitoring and control on the motion screen	6-12
6.1.8.1	How to define the motion view for a state graph	6-13
6.1.8.2	How to define the motion view for an instance	6-14
6.1.8.3	Example of the configuration of texts and addresses on the motion screen	6-15
6.1.9	Visualization of units on the overview screen	6-17
6.1.9.1	How to define the view of an instance unit	6-17
6.1.9.2	How to define the view of a graph group unit	6-17
6.1.10	Transition handling in ProAgent	6-18
6.1.11	Programming guidelines for obtaining correct results in a ProAgent analysis	6-19
6.2	Diagnostics by means of format converter	6-21
6.2.1	Interaction between S7-HiGraph, the AS and the OP	6-22
6.2.2	Message events	6-23
6.2.3	Requirements of format converter diagnostics	6-23
6.2.4	How you generate diagnostic data for format converter diagnostics	6-24
7	User program runtime on the PLC	7-1
7.1	Rules for cyclic execution of a state	7-1
7.2	Overview of cyclic execution of a state	7-2
7.3	Reaction on startup and hot restart	7-3
7.4	Reaction on POWER OFF / ON	7-4
7.5	Reaction of default variables during load operations or at startup	7-6
7.6	Memory requirements of the user program	7-6
8	STL language description	8-1
8.1	STL instructions	8-1
8.2	Valid data types	8-10

9	Example configurations	9-1
9.1	"Drill unit" configuration, example	9-1
9.1.1	Learning objectives	9-1
9.1.2	"Transfer line" automation task	9-2
9.1.3	Expansion with control and operating devices	9-3
9.1.4	Determining the function units to be controlled	9-4
9.1.5	Assignment of function units and state graphs	9-5
9.1.6	Creation of graph group	9-6
9.1.7	Defining the program structure	9-8
9.1.8	Overview: State graph and graph groups for the drill unit	9-9
9.1.9	State graph for controlling the operation enable signal	9-11
9.1.10	State graphs for controlling the operating modes	9-13
9.1.11	State graph for coordination of the drill unit	9-15
9.1.12	Motor state graph	9-17
9.1.13	Status graph Valve_2E	9-19
9.1.14	Clamp state graph	9-21
9.1.15	Compiler settings for the sample configuration	9-23
9.2	Example of the configuration "Drill unit with standard diagnostics"	9-24
9.2.1	Configuring diagnostics for the "Operating modes" state graph	9-24
9.2.2	Configuring diagnostics for the "Operation enables" state graph	9-25
9.2.3	Diagnostic configuration in the motion state graphs	9-26
9.2.4	Diagnostics configuration in graph groups	9-29
10	Control elements.....	10-1
10.1	Calling up the shortcut menus	10-1
10.2	Tooltips.....	10-1
10.3	Symbols.....	10-3
10.3.1	Toolbar icons.....	10-3
10.4	Keyboard control	10-5
10.4.1	Selecting menu commands with keystrokes	10-5
10.4.2	Moving the cursor in the text editor	10-6
10.4.3	Moving the cursor in dialog boxes	10-6
10.4.4	Selecting menu commands form the menu bar / pop-up menu	10-7
10.4.5	Selecting texts with key commands	10-7
10.4.6	Accessing the help with key commands	10-8
10.4.7	International/German keyboard layout.....	10-8
11	Technical support	11-1
11.1	Further technical support	11-1
11.2	Service & Support on the Internet.....	11-1
11.3	A&D Technical Support.....	11-2
11.4	Training Center	11-2

Glossary

Index

Tables

Table 8-1	Bit logic operations.....	8-1
Table 8-2	Word logic operations	8-2
Table 8-3	Time operations	8-2
Table 8-4	Counter instructions	8-2
Table 8-5	Load and transfer operations	8-3
Table 8-6	Comparison operations	8-3
Table 8-7	Block calls	8-3
Table 8-8	Fixed point maths.....	8-4
Table 8-9	Floating point (real number) arithmetic	8-4
Table 8-10	Rotate and shift instructions.....	8-5
Table 8-11	Accumulator operation instructions.....	8-5
Table 8-12	Conversion instructions.....	8-6
Table 8-13	STL instructions, sorted by mnemonics	8-6
Table 8-14	Valid data types	8-10

Product overview

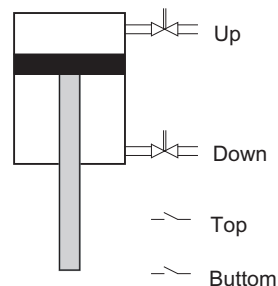
1.1 Overview of S7-HiGraph

S7-HiGraph enhances STEP 7 functionality with a graphic interface for programming state graphs.

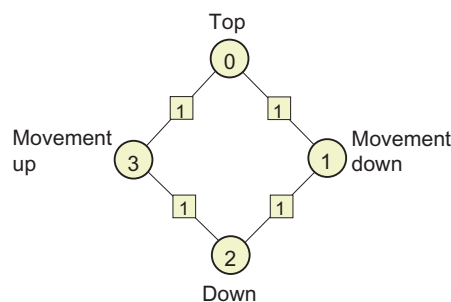
Mapping the process image to state graphs

State graphs represent an efficient tool for controlling and mapping SIMATIC objects. The program structure is oriented on the technological objects of the process for easy interpretation.

Valves with the states
"Top" and "Bottom" and
the movements "Up" and
"Down"



Representation of states
in a state graph



Advantages of the graphical layout

The program structure is shown and documented in graphic format. The clear graphic layout facilitates service and maintenance and enhances modification options.

Reusing state graphs

Once created, the state graphs are always available for reuse in other applications and central editing. This reuse feature is particularly practical when used in conjunction with standardized function units, for example, valves or motors.

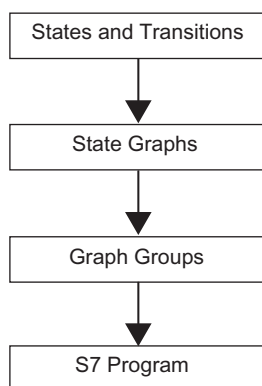
1.2 S7-HiGraph programming language

Preliminary aspects: Creating images of technological units

Basic requirement in programming is to divide your automation task into several technological function units. A functional unit may consist of a mechanical component (for example, a valve), or represent a conceptional unit (for example, "operating mode control".) The reaction of each functional unit is then described with the help of a state graph.

Structure of an S7-HiGraph program

Hierarchy of S7-HiGraph programs:



States

Defines the states a functional unit can assume in the process. A valve may assume the "open" or "closed" state, for example.

Transitions

Transitions contain step conditions which must be satisfied in order to change a status.

State graphs

A state graph describes the reaction of each functional unit in the process. It usually contains several states and transitions.

Graph groups

State graphs are organized in groups in order to generate a control program for the entire process, based on the individual state graphs. These may correspond with the mechanical functional units of a machine, for example. One of the state graphs within a group can be defined as a coordinator.

State graphs can communicate by exchanging messages.

S7 program

Blocks of the S7 program:

- Function (FC)

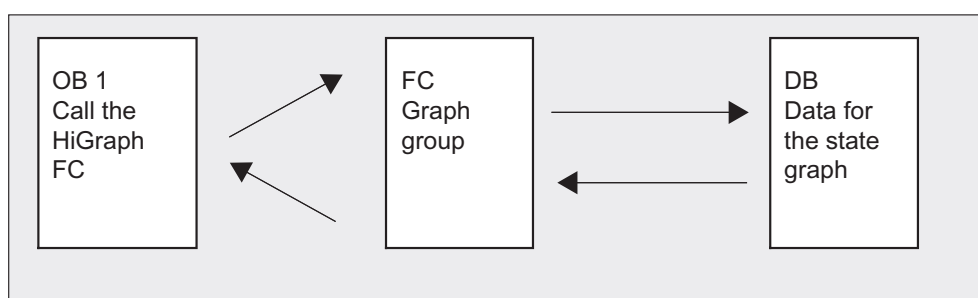
A function (FC) is generated when a graph group is compiled

- Data block (DB)

The DB is also generated during compilation. It contains the data of the various state graphs.

- Calling block (OB)

In order for the loaded S7-HiGraph program to be executed in the CPU, the S7-HiGraph FC must be called in a cyclic block (for example, OB1.)



1.3 Functions in S7-HiGraph

Overview of tasks

S7-HiGraph provides various functions for handling your specific tasks. We distinguish the following tasks:

- Programming
- Monitoring
- Debugging
- Diagnostics

Programming

- Convenient engineering environment based on Windows standards.
- Programming of actions in states and transitions in the STL programming language.
- Call of STEP 7 code blocks (FC, SFC, FB, SFB with STL, LAD, FBD, or SCL instructions) from the state graph
- Programming of waiting, monitoring and delay times without using S7 timers: This allows you to save S7 timers, which are available only in limited numbers on the CPUs.

The various timers have the following functions:

- Waiting times delay processing of a status.
- Monitoring times monitor the processing time of a state.
- Delay times prevent a reaction of the CPU to short signal pulses.

Monitoring

You can program global monitoring functions for a state graph. Specific events (for example, emergency off) can thus be monitored centrally, irrespective of the active state.

Debugging

A monitoring mode is available for debugging the system. This function can be used to visualize the status of a functional unit at the time of its transition to the current state, and the relevant step conditions. This enhanced transparency also facilitates online diagnostics when the system is in run.

Process error diagnostics

S7-HiGraph contains an integral interface for process error diagnostics: Error states, monitoring timeout and messages can be visualized on an HMI. This requires only slightly more programming time.

1.4 What's new in V5.3?

New functionality in V5.3

Version 5.3 of the S7-HiGraph programming software contains the following enhancements or changes compared to V5.2:

Installation requirements

- MS Windows 2000 Professional or MS Windows XP Professional.
- STEP 7 V5.3 or higher.

New licensing concept

A new licensing concept is available, starting at S7-HiGraph V5.3. License agreements are no longer granted by means of authorization, but rather by means of License Key. License Keys are managed with the help of Automation License Manager. The AuthorsW program is now obsolete

Printing in color

Users can now print S7-HiGraph documents in color.

Delay times

You can use delay times to prevent a reaction of the PLC to short signal pulses.

Modified step reaction when executing cyclic actions when RLO = 0

The compiler option "Cyclic actions with RLO=0" was tuned. This ensures that all signals set while dwelling in a certain state are reset when this state is exited.

Session memory

Enhanced "Session memory" function:

- Documents are opened with the saved screen settings when started in SIMATIC Manager.
- When the user sets the status mode, documents are reopened with the screen settings saved for this status mode. Open documents remain open and are not changed.

Installing

2.1 Automation License Manager

2.1.1 License agreements with Automation License Manager

Automation License Manager

Use of the programming software requires a product-specific License Key (license agreement). Starting with S7-HiGraph V5.3, this key is installed using Automation License Manager.

Automation License Manager is a software product of Siemens AG, and is a global system tool for used to handle License Keys (technical license representatives.)

Automation License Manager is found:

- on the S7-HiGraph or STEP 7 product CDs
- as a download on the Internet pages of Siemens AG A&D Customer Support.

Automation License Manager provides an integrated Online Help system. You can call this help after installation context-sensitive by pressing F1, or by selecting the **Help > Help on License Manager** command. The Help system provides detailed information about the functionality and handling of Automation License Managers.

Licenses

The STEP 7 program packages are protected by licenses which users need to obtain. A license is issued for granting rights of use for relevant products. The representatives of these rights are:

- the CoL (Certificate of License) and
- the License Key.

Certificate of License (CoL)

The "Certificate of License" included with the relevant product represents legal proof of the rights of use. The relevant product may only be used by the licensee (CoL) or his representatives.

License Key

A License Key is the technical representative of a license (electronic license stamp.)

SIEMENS AG issues a License Key for all software protected by legal license agreement. A software can only be used if a corresponding license is found on a computer, and only in accordance with the license agreement and conditions of use associated with this License Key.

Users can store License Keys and transfer these between storage media as follows:

- on License Key floppy disks,
- on local hard disk drives and
- on hard disk drives of computers on the network.

For further information on handling License Keys, please refer to the Automation License Manager Online Help.

Note

- You may use S7-HiGraph over a short period without License Key in order to get acquainted with its user interface and functionality.
 - Unrestricted use in accordance with legal license agreements, however, is only possible after you installed the License Key.
 - If you have not installed the License Key, you are requested at regular intervals to do so.
-

License types

Siemens AG distinguishes between the following user-oriented license types for its software products. The behavior of the software is controlled by different License Keys for these license types. The type of use is derived from the relevant Certificate of License.

License type	Description
Single License	Use of the software on any computer without time limit.
Floating License	Software license obtained on a network ("remote" usage) without time limit.
Trial License	The use of the software is restricted to: <ul style="list-style-type: none">• a maximum period of 14 days,• a certain number of days, starting at initial use,• use for testing and validation (disclaimer of liability.)
Upgrade License	An upgrade may require a specific system state: <ul style="list-style-type: none">• An Upgrade License can be used to convert an "Old"Version x to a Version >x+... license.• An upgrade may be required after an extension of data volumes.

2.1.2 Installing Automation License Manager

Installing Automation License Manager

Automation License Manager is installed by a Setup program. The installation software for Automation License Manager is found on the S7-HiGraph or STEP 7 product CDs.

You can install Automation License Manager alongside with S7-HiGraph, or decide to do so later.

Note

- For detailed information on Setup procedures for Automation License Managers, refer to the current readme.wri.
 - The Automation License Manager Online Help system provides all essential information on the functionality and handling of License Keys.
-

Installing License Keys at a later time

When you start the STEP 7 software without having installed the License, the system outputs a corresponding message.

Note

- You may use software over a short period without License Key in order to get acquainted with its user interface and functionality.
 - Unrestricted use in accordance with legal license agreements, however, is only possible after you installed the License Key.
 - If you have not installed the License Key, you are requested at regular intervals to do so.
-

To install License Keys separately:

- Installing License Keys from floppy disks
- Installing License Keys by means of download from the Web (requires ordering)
- Use of Floating License Keys available on the network.

For detailed information on procedures, refer to the Automation License Manager Online Help. You can call this help after installation context-sensitive by pressing F1, or by selecting the Help > Help on License Manager command.

Note

- License Keys can only be used under Windows 2000/XP if installed on a hard disk drive with write access.
 - Floating Licenses can also be used via the network, that is, "remotely."
-

2.1.3 Rules for handling License Keys

Further information

You can open the Automation License Manager Online Help context-sensitive by pressing F1, or by selecting the **Help > Help on Automation License Manager** command.

This help provides all information on the functionality and handling of License Keys.

Caution

Note the information on handling License Keys contained in the Online Help and readme.wri for Automation License Manager. If neglected, you run the risk of irrevocably losing your License Keys.

2.2 Installing S7-HiGraph

2.2.1 Installing S7-HiGraph

Installation requirements

- Operating system Microsoft Windows 2000/XP
- SIMATIC STEP 7 basis package. For information on the version required, refer to the readme.wri file
- A PC or PG. For requirements, refer to readme.wri
A programming device (PG) is a special, compact Personal Computer for industrial use. It is fully equipped for programming SIMATIC automation systems.
- Storage capacity:
For free hard disk space requirements, refer to the "readme" file.
- MPI interface (option):
The MPI interface between the engineering system (PG or PC) and the PLC is only required when using MPI for communication with the PLC. You require either
 - a PC/MPI cable that is connected to the communication port of your device, or
 - an MPI module on your PG/PC.Some PGs are equipped with an integrated MPI interface.
- External prommer (option):
When using a PC, you only need an external prommer if you want to program EPROMs.

Installation procedure

S7-HiGraph contains a Setup program for automatic installation. Setup is executed using standard Windows software setup procedures.

Setup is completely menu controlled.

2.2.2 Running Setup

Preparations for installation

- Start Windows to run Setup
- To run Setup from CD-ROM, place the CD-ROM into the CD-ROM drive of your PC.

Running Setup

To run Setup:

1. Open the "Control panel" in Windows, then double-click "Add/Remove Programs" to open the software setup dialog box.
2. Follow the Setup instructions on the screen.

Setup guides you through the installation process. You can always select the next or previous step.

Setup offers you various dialog boxes containing questions or options. Please read the information below in order to be able to quickly and easily confirm the dialogs.

If an S7-HiGraph is already installed ...

if Setup finds an S7-HiGraph installation on your computer, it outputs a corresponding message and the following options:

- Cancel setup, in order to uninstall the old version of S7-HiGraph under Windows and then run Setup again, or
- Continue Setup and overwrite the old version.

You should uninstall any older version before you run Setup in order to maintain a clean software structure. To simply overwrite the older version also has the disadvantage, that any remaining old elements may not be removed when you uninstall it at a later time.

2.2.3 Installation information

About installing License Keys

Setup scans the hard disk drive to find a corresponding License Key. If a valid License Key is not found, you are notified that the software can only be used with a License Key. You can then choose to install the License Keys right away, or resume setup and install it at a later time. In the first case, insert your License Key disk when Setup asks you to do so.

Setup errors

The following errors cancel Setup:

- You more than likely have not started Setup under Windows if an initialization error occurs when you try run Setup.exe.
- Insufficient hard disk space: You need a sufficient amount of free space on your hard disk drive (see the readme), depending on the scope of your installation.
- Defective CD / disk: Please contact your local SIEMENS representative if you find a defective CD / disk.
- Operating error: Run Setup again and follow the instructions carefully.

At the end of installation...

A corresponding screen message informs you that Setup has successfully installed your software.

If Setup has modified any DOS files, you are prompted to restart Windows. You can run S7-HiGraph after the restart.

You can also run S7-HiGraph directly after the completion of Setup.

2.2.4 Uninstalling S7-HiGraph

Procedure:

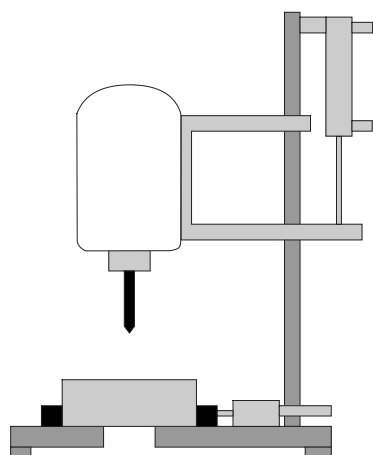
Use the standard Windows uninstall procedure:

1. Open the "Control panel" in Windows, then double-click "Software" to open the software setup dialog box.
2. Select S7-HiGraph from the software list. Click "Remove" to uninstall the software.
3. If the "Remove free file" dialog box appears, click "No" if you are uncertain.

Designing a program using the example of a drill

3.1 Welcome to the S7-HiGraph example for newcomers

This Getting Started shows you within one hour flat how to use S7-HiGraph to create a program for the automation of the drilling machine described below.



Learning objectives

In the first step, you learn how to efficiently configure an S7-HiGraph program. In the next steps you are introduced to all tasks you need to carry out in SIMATIC Manager and S7-HiGraph:

- Creating a program
- Downloading a program to the CPU
- Debugging a program

Storage location of the sample project

The correctly programmed "Zen03_02_HiGraph_Drillmac" project for this Getting Started is included in your software package. After installation, you can find it in the STEP7\Examples folder.

3.2 Requirements for executing the sample program

Various conditions must be satisfied before you can start your specific tasks.

Programming the example

Hardware and software components required:

- a programming device / PC
- the STEP 7 basic package and the S7-HiGraph optional package.

Debugging the sample program

In order to download and debug the sample program, you also need the following components:

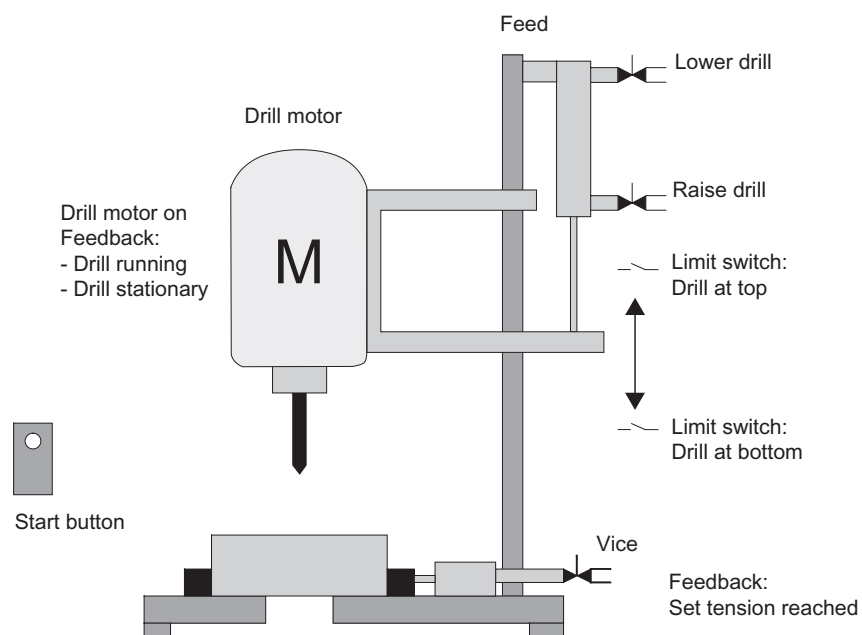
- either:
an AS with digital I/O module (8DI+8DO).
In this example we deploy an S7-300 with CPU 314. However, S7-HiGraph programs can also be executed on an S7-400 automation system,
- or:
S7-PLCSIM optional package for simulating an S7-300 / S7-400 CPU.

3.3 Automation task drilling machine

Create a sample program for a drilling machine according to the following specifications:

- Technology screen
- Basic state
- Functional diagram

Technology screen: Structure of the drilling machine



Basic state

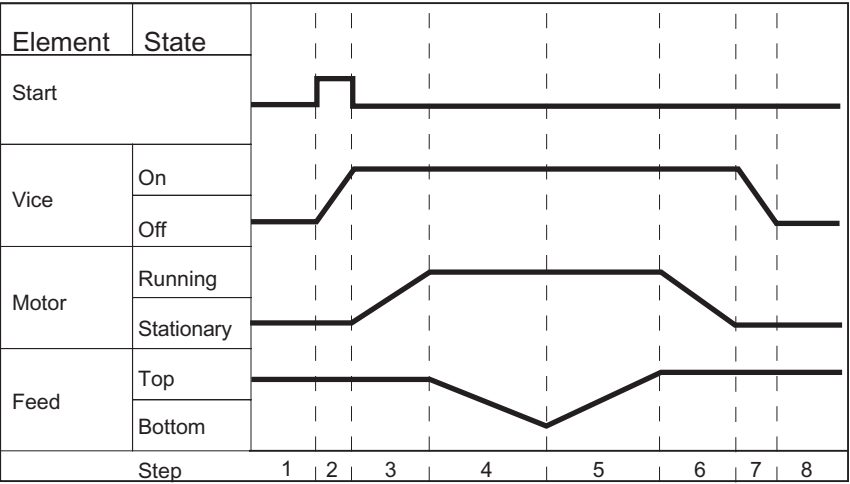
Definition of the basic state of the drilling machine:

- Drill motor is idle.
- Feed / drill in top position.
- No part clamped.

Drilling sequence

- The drilling sequence is divided into the following steps:
- 1. Insert the part, then press the Start button to run the machine
 - 2. Clamp the part (until the set clamping pressure is reached)
 - 3. The drill motor starts
 - 4. Use the feed to lower the drill to the lower set position
 - 5. Use the feed to raise the drill to the upper set position
 - 6. Switch off the drill motor
 - 7. Open the clamp
 - 8. Remove the part

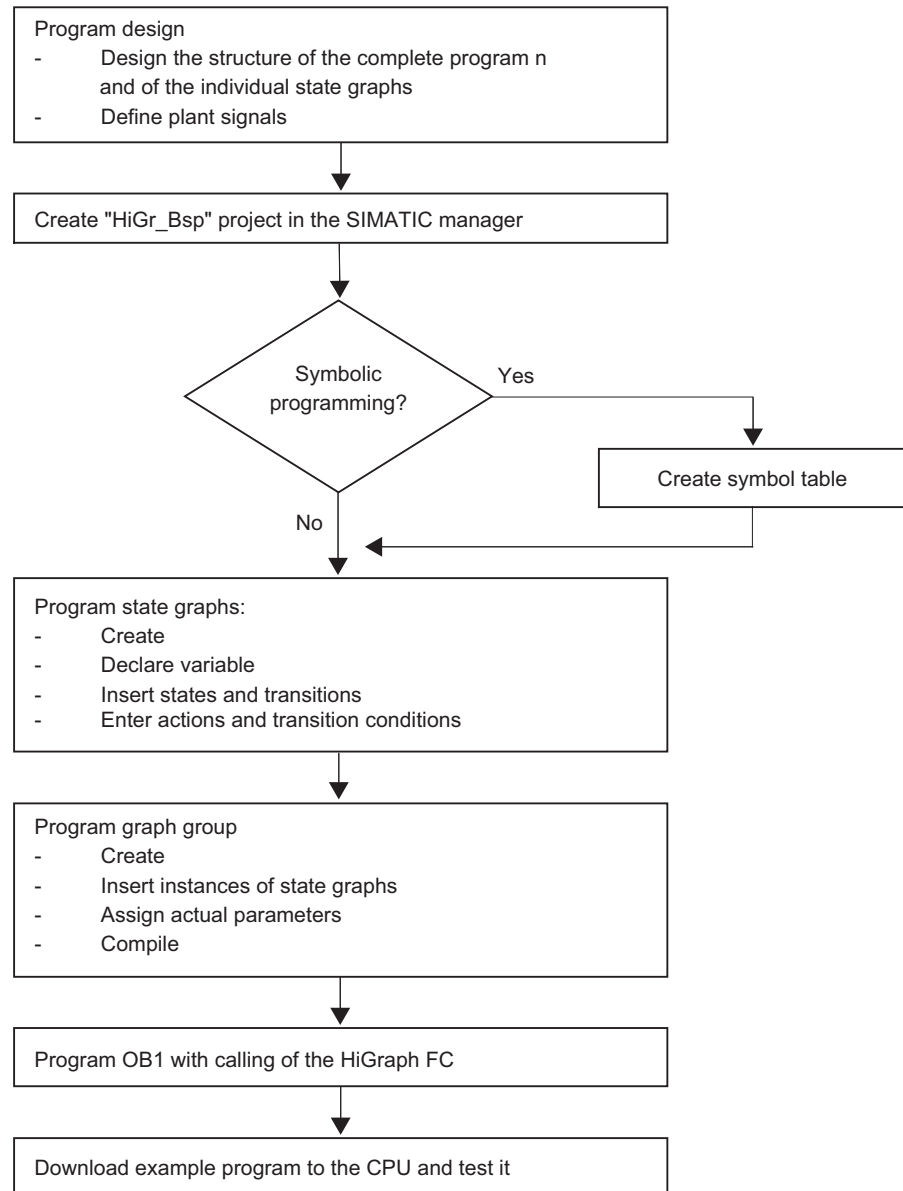
The function diagram below shows the drilling sequence:



3.4 Steps in creating the "Drilling Machine" sample program

Overview of steps

The flow chart below illustrates the various steps. The various steps shown below are described in detail.



3.5 Step 1: Define the function units and state graphs

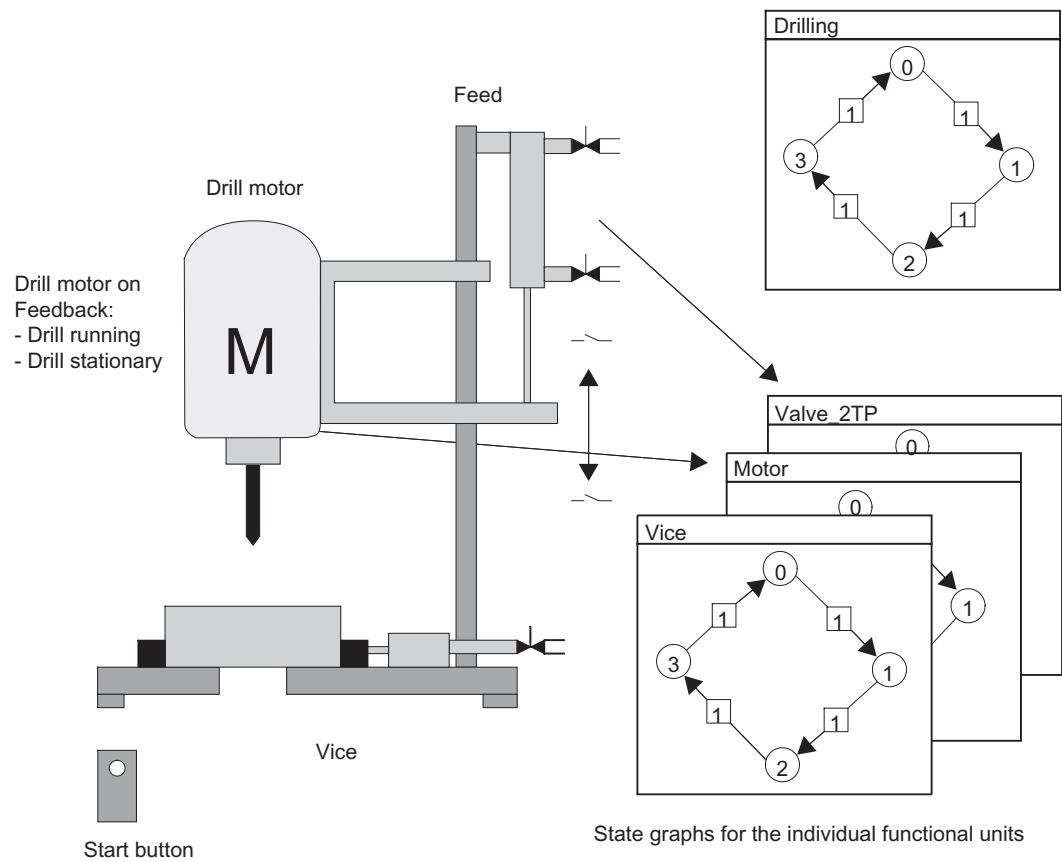
Follow these rules:

- One state graph is required per function unit or task.
One state graph is usually installed for each mechanical component of a process.
- State graphs are used for further functions, for example, control of operating modes or operation enables.
- One or several higher-priority state graphs are used to coordinate state graph groups.
This structure is created within the graph groups.

Procedure:

1. Divide the drilling machine into the following function units:
 - "Drilling motor"
 - "Feed" The feed function is implemented by means of a valve with two limit positions.
 - "Clamp"
2. Assign the following state graphs for controlling the function units:
 - "Motor"
 - "Valve_2E"
 - "Vice"
3. Use the "Drill" state graph for coordination

State graph "Drilling"



3.6 Step 2: Designing the state graph

In this example, we are going to program the "Valve_2E" state graph. Further state graphs required are included in your "ZEn03_01_HiGraph_Drilmac" sample project.

Requirements

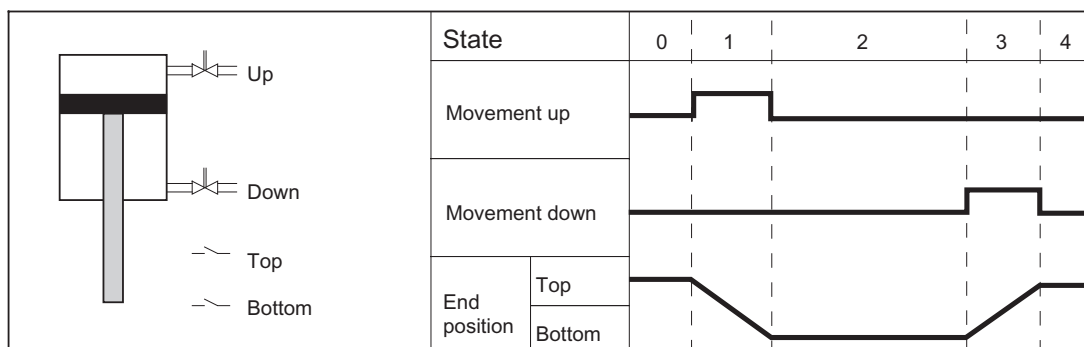
The sample program is based on the following conditions:

- The function unit on which the "Valve_2E" state graph is based is a valve unit with two limit positions.
- The solenoid valves only need to be operated for the motion phase. The valve remains in the relevant limit position.

Defining elements of the function unit

Elements required in this example:

- A solenoid valve for the "up" motion
- A solenoid valve for the "down" motion
- A limit switch for the "top" limit
- A limit switch for the "bottom" limit



Defining the states

Define the valve unit states. The table below shows the names used in this example:

no.	State	Description
0	Initialization	An initialization state is required in every state graph. In this state it is possible to check whether the functional unit is in a defined home position. If required, it can be returned to its initial position.
1	"Top" limit position	Drill in the upper limit position
2	"Down" motion	Drill moves down
3	"Bottom" limit position	Drill in the lower limit position
4	"Up" motion	Drill moves up

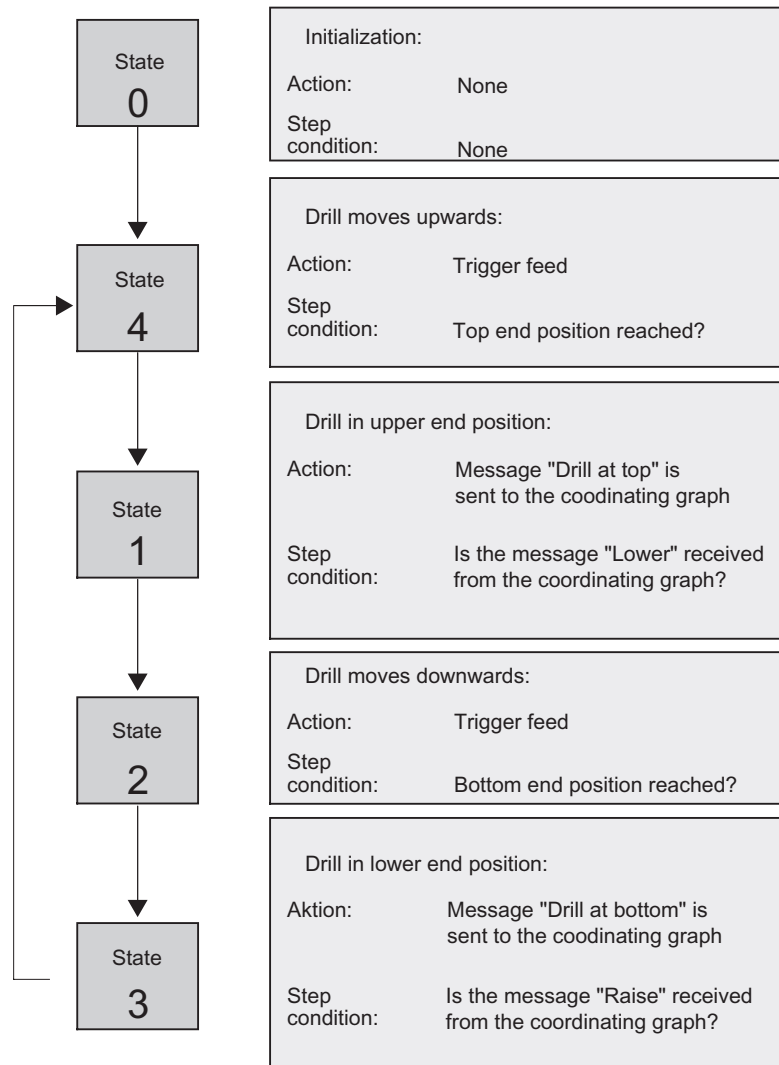
Defining the status change

Define the status change. Status changes required in this example:

- The "Drilling" state graph determines the valve unit transition to the next state. For this purpose it sends messages to the "Valve_2E."
- When the valve reaches the limit, a message is returned to the "Drilling" state graph.

Designing the state graph

Create the state graph based on the earlier specifications:



3.7 Step 3: Defining the system signals

Requirements

After you have split the drilling process into its basic functions, you should define the corresponding inputs and outputs for each state. The concept is based on the technical diagram and the flow chart.

Follow these steps:

1. List the corresponding inputs and outputs of the drilling machine in an assignment table.
2. In addition to the absolute inputs and outputs, enter your symbolic names (for example, I 0.4 "Tension_reached")
3. You should also comment your program appropriately ("feedback for set part clamping pressure reached", for example.)

Assignment table for the sample program

In the drilling machine example, we assume that the switches and contactors of the drilling machine are controlled via the inputs and outputs of the digital I/O module of the S7-300 CPU. The I/O module used has 8 inputs and 8 outputs. The default values of the input and output addresses of the module on slot 4 are: I 0.0 to I 0.7 and O 0.0 to O 0.7.

The resultant assignment table:

Address, absolute	Address, symbolic	Description
Inputs in the program		
I 0.0	Drill_motor_running	Feedback for "Drill running at set speed"
I 0.1	Drill_motor_stopped	Feedback for "Drill stopped"
I 0.2	Drill_at_bottom	Limit switch for "Drill in bottom position"
I 0.3	Drill_at_top	Limit switch for "Drill in top position"
I 0.4	Clamping_pressure_reached	Feedback message "Set part clamping pressure reached"
I 0.7	Start_button	Button for starting the drill
Outputs in the program		
O 0.0	Drill_motor_on	Switch drill motor on
O 0.1	Drill_Down	Use the feed to move the drill into its bottom limit position
O 0.2	Drill_Up	Use the feed to raise the drill to the upper limit position
O 0.3	Clamp_part	Clamp / lock part at a set pressure

3.8 Step 4: Create a "HiGr_Exp" project in SIMATIC Manager

Requirements

Prerequisite for programming with S7-HiGraph is a project for saving the data of your S7-HiGraph program.

To create a project in SIMATIC Manager...

1. Select **File > Wizard "New Project"**
to open the STEP 7 wizard. This tool supports you in creating the project.
2. Enter the following information
 - Which CPU are you using in your project?
Specify your CPU. A CPU 314 is used in the included example.
 - Which blocks do you want to add?
Select OB1.
 - What is the name of your project?
Enter the name "HiGr_Exp."

Project structure

Projects for state graph programming are the same as other projects in STEP 7.

The STEP 7 Wizard now creates a root folder for the station you selected. This contains a nested folder with the selected CPU. It contains the S7 program, including the block, symbol and source folders.

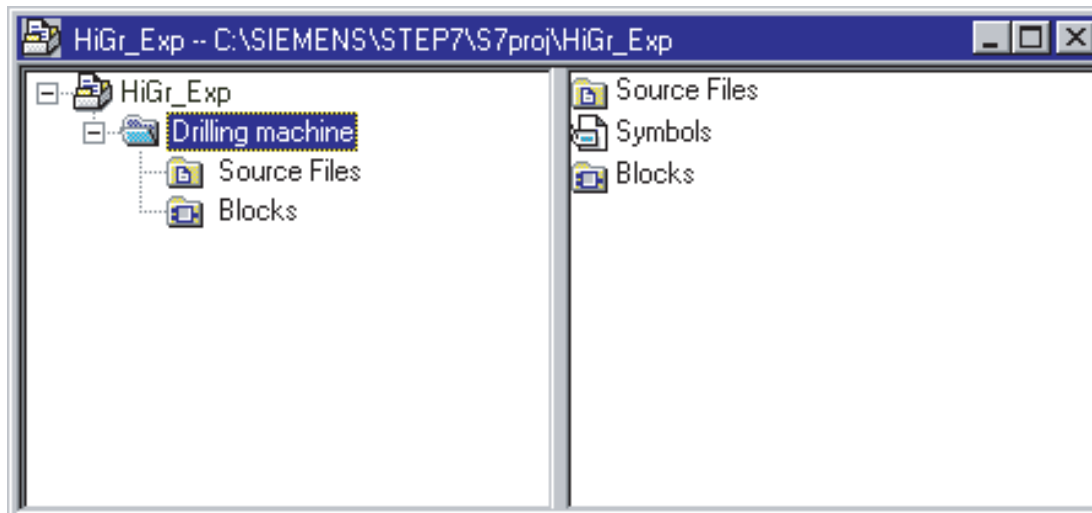
An "S7 program" folder is automatically generated for each CPU you have assigned in your project configuration. This folder is used to store the blocks, sources and symbols of the user program.

To rename the S7 program:

1. In SIMATIC Manager, select the "S7 Program" folder.
2. Assign the name "Drilling machine" to the S7 program (menu command **Edit>Rename**).

Structure of the sample project

The figure below shows the structure of the sample project in SIMATIC Manager.



3.9 Step 5: Creating a symbol table

Because you are going to use symbolic addresses for programming, you should now create a symbol table.

Procedure:

1. Open the symbol table in the "Drilling machine" folder by double-clicking the "Symbols" folder.
2. Edit the table as shown in the figure below:

Symbol table

	Status	Symbol	Address		Data type	Comment
1		Drill_Motor_On	A	0.0	BOOL	Switch on drill motor
2		Lower_Drill	A	0.1	BOOL	Lower drill via feed to lowest limit
3		Raise_Drill	A	0.2	BOOL	Raise drill via feed to highest limit
4		Clamp_Workpiece	A	0.3	BOOL	Clamp/hold workpiece with set tension
5		DB_GG_Drilling	DB	1	DB 1	DB for drilling graph group
6		Drill_Motor_Running	E	0.0	BOOL	Feedback signal for "drill running at set speed"
7		Drill_Motor_Stopped	E	0.1	BOOL	Feedback signal for "drill stationary"
8		Drill_at_Bottom	E	0.2	BOOL	Limit switch for "drill at lowest position"
9		Drill_at_Top	E	0.3	BOOL	Limit switch for "drill at highest position"
10		Tension_Reached	E	0.4	BOOL	Feedback signal for "workpiece set tension reached"
11		Start_Button	E	0.7	BOOL	Start button for drilling machine
12		GG_Drilling	FC	1	FC 1	FC for drilling graph group
13		CYCL_EXC	OB	1	OB 1	
14						

3.10 Step 6: Creating a state graph

In this introductory example, we are going to program only the "Valve_2E" state graph. The state graphs also required are included in your "ZEn03_01_S7HiGraph_Drillmac" sample project.

State graphs are created in the "Sources" folder of the S7 program

Procedure:


1. In SIMATIC Manager, open the "Sources" folder in the S7 program "Drilling machine".
2. Select **Insert > S7 Software > Status Graph**.
3. Name the created state graph "Valve_2E."

3.11 Step 7: Declaring the variables

Variable overview in S7-HiGraph

Variables are declared directly in the variable overview of S7-HiGraph. The variable overview contains various declaration sections. These already contain default variables S7-HiGraph automatically enters in the declaration when a state graph is created.

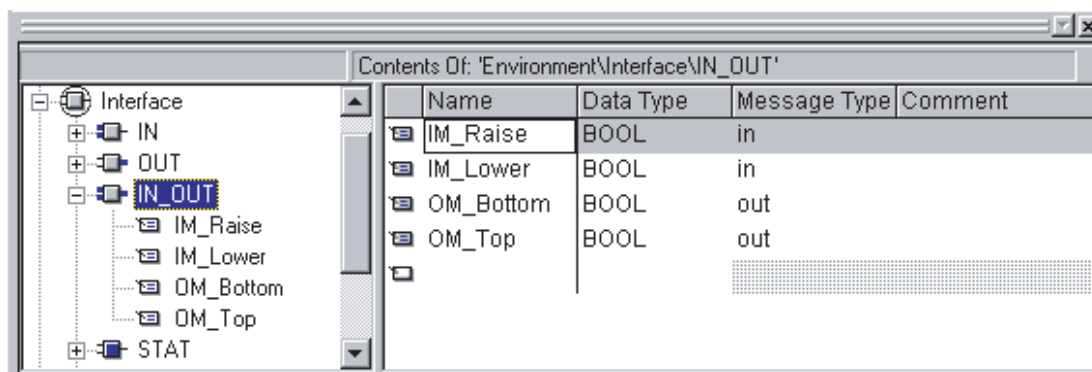
Procedure:

1. In SIMATIC Manager, select the "Sources" folder and double-click "Valve_2E" to start S7-HiGraph.
2. Click  to open the variable overview. There you can view the default variables in S7-HiGraph.
3. Select the relevant declaration section, then select **Insert > Declaration line**.
4. Change to the detail view and select the "Variables" tab.
5. Type in the variable name, the data type and the message type as shown in the table below.

Declaration section	Name	Data type	Message
IN	Top	Bool	
	Bottom	Bool	
OUT	Up	Bool	
	Down	Bool	
IN_OUT	IM_Up	Bool	In
	IM_Down	Bool	In
	OM_Top	Bool	Out
	OM_Bottom	Bool	Out

Filled out variable detail view



The figure below shows the filled out variable detail view with the selected declaration section IN_OUT.

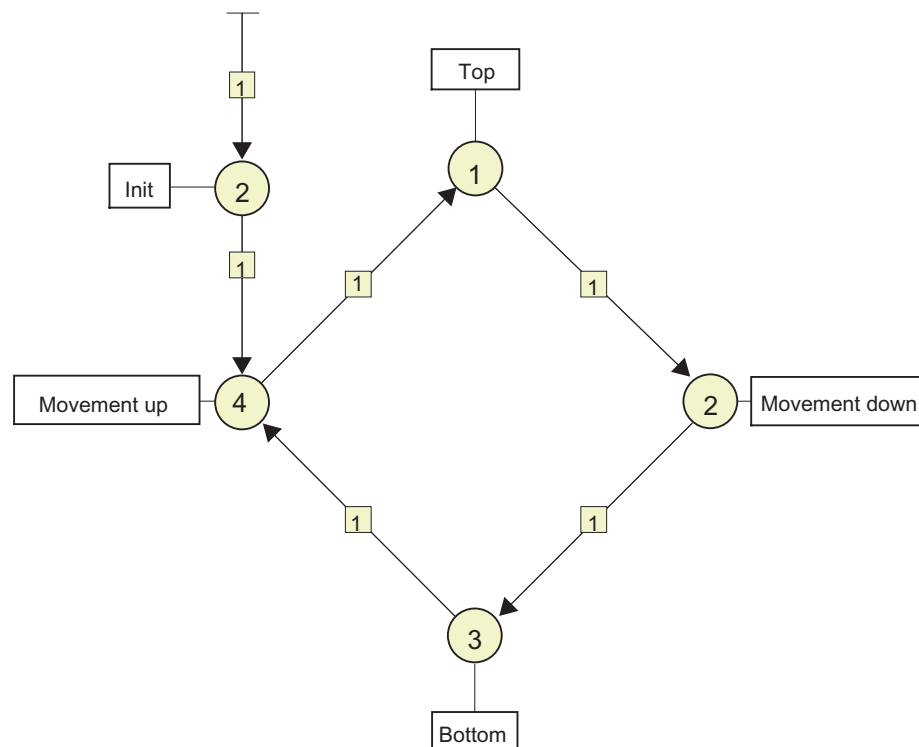


3.12 Step 8: Inserting the states and transitions

Now insert the states and transitions for state graphs in the editing window as shown in the figure below.

Procedure:

1. Select **Insert > Status**  and then insert the states 1 to 4.
2. For precise positioning, select **Options > Align**.
3. Select **Insert > Transition**  and then interconnect the states.



Note

Always begin and end a transition in the center of a state circle. Only this method ensures that the transition has a connection to the state. Transition ends which do not have a connection to a state are identified by a small crossline. These are treated as special forms of transitions (as Return or Any transitions).

3.13 Step 9: Enter the state name

To improve the structural layout, each status is assigned a self-explanatory name:


Procedure:

1. Select the status, then select **Edit > Object Properties**.
This command can also be called with a right-click.
2. Type in a name in the "Name" input box.
The name is shown in a box next to the state.
3. Drag-and-drop the small box to a suitable position on the drawing area.

3.14 Step 10: Entering actions and transitions

You need to program the actions and transition conditions belonging to the state graphs.

Procedure:

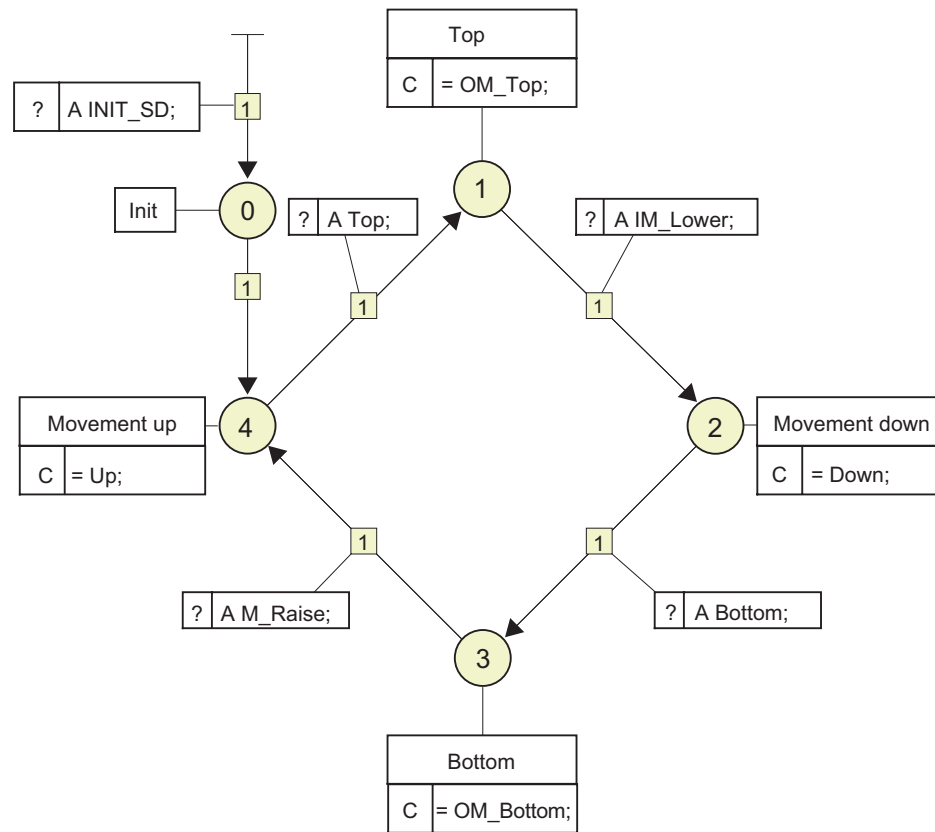
1. In the first step, select state 4.
2. Next, open the detail view by selecting **View > Details**. 
3. Select the "Instructions" tab.
4. Select the "Cyclic actions" instruction type on the left window pane.
5. Right-click and select "Insert."
A new statement line is inserted.
6. Select the new instruction and enter =Up on the right window pane. Always terminate a statement with a semicolon.
7. Now, click all further states consecutively and enter the corresponding instructions as shown in the figure below.

Note

STEP 7 works with a global set of valid keywords, e.g. "Down" for output bytes. If you want to use this kind of string as an address, indicate it using the symbol ID #. The symbol ID prevents the string from being recognized as a keyword.

8. Next, select the state 4 to state 1 transition.
9. Select the "Conditions" instruction type on the left window pane.
10. Right-click and select "Insert."
A new condition is inserted.
11. Enter the condition U Top and terminate the line with a semicolon.
12. Use the same procedure for all other transitions as shown in the figure below.

13. Select **File > Save** 



3.15 Step 11: Creating a graph group

Overview

Graph groups are created in four separate steps:

Step	What?
Preparation	Copy further state graphs This step is only required in this sample project: You successfully created the "Valve_2E" state graph. All other state graphs are found in the included "ZEn03_01_HIGRAPH_DrilMac" sample project.
A	Create the graph group In the graph group, define the order in which the state graphs are called cyclically in the program. Graph groups and state graphs are saved to the same folder.
B	Insert and name the instances of the state graphs The call of a state graph in a graph group is called an instance. Add the instances of the state graphs to the graph group and assign meaningful names to them.
C	Specify the run sequence The order by which the instances are executed is determined in the run sequence.

Preparation: How to copy further state graphs

1. Change to SIMATIC Manager.
2. Copy the "Motor", "Vice" and "Drill" state graphs from the "ZEn03_01_HIGRAPH_DrilMac" program.
3. Paste these graphs into the "Sources" folder of your program.

Partial step A: To create the graph group

1. In SIMATIC Manager, open the "Sources" folder in the S7 program "Drilling machine".
2. Select **Insert > S7 Software > Graph Group**.
3. Assign the name "Drilling_Machine" to the graph group you created and then open it with double-click.
You now see an empty drawing area.

Partial step B: To insert and rename instances

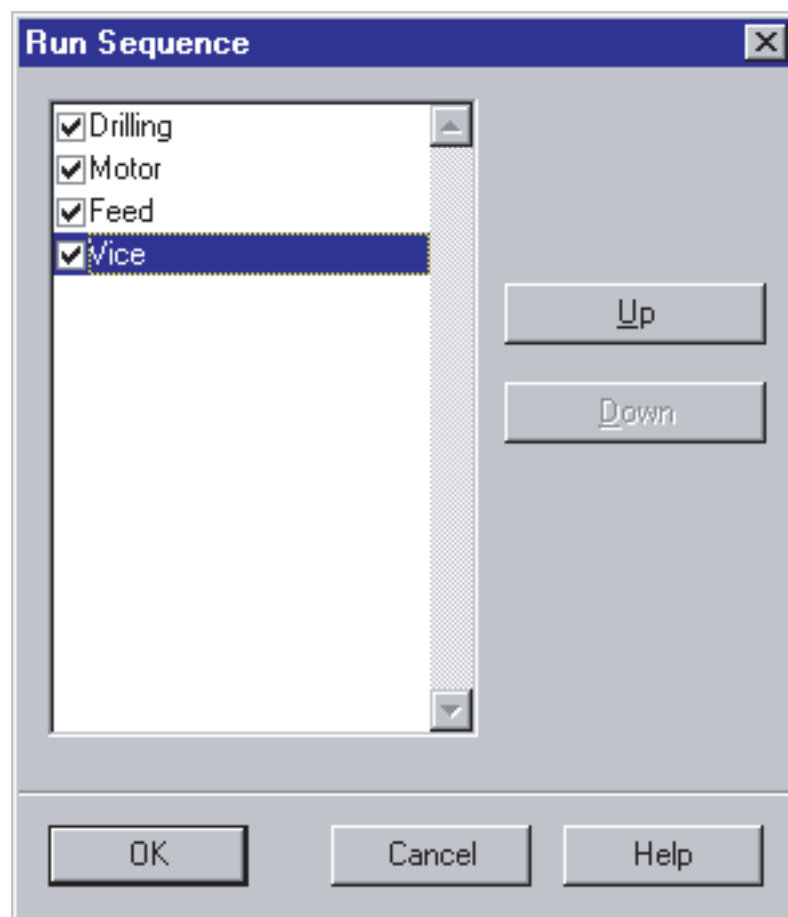
1. Select **Insert > Instance**
2. On the next dialog box, select state graph "Valve_2E."
3. Place the instance onto the drawing area.
4. Repeat this operation to insert the instances of all four state graphs
5. Now select **Edit > Object Properties** to call the "Instance properties" dialog box and assign meaningful names to the instances.

6. Enter the following names in the "Name" input box.

Instance of the state graph	Name
Valve_2E	Feed
Motor	Drilling motor
Vice	Clamp
Drill	Drill

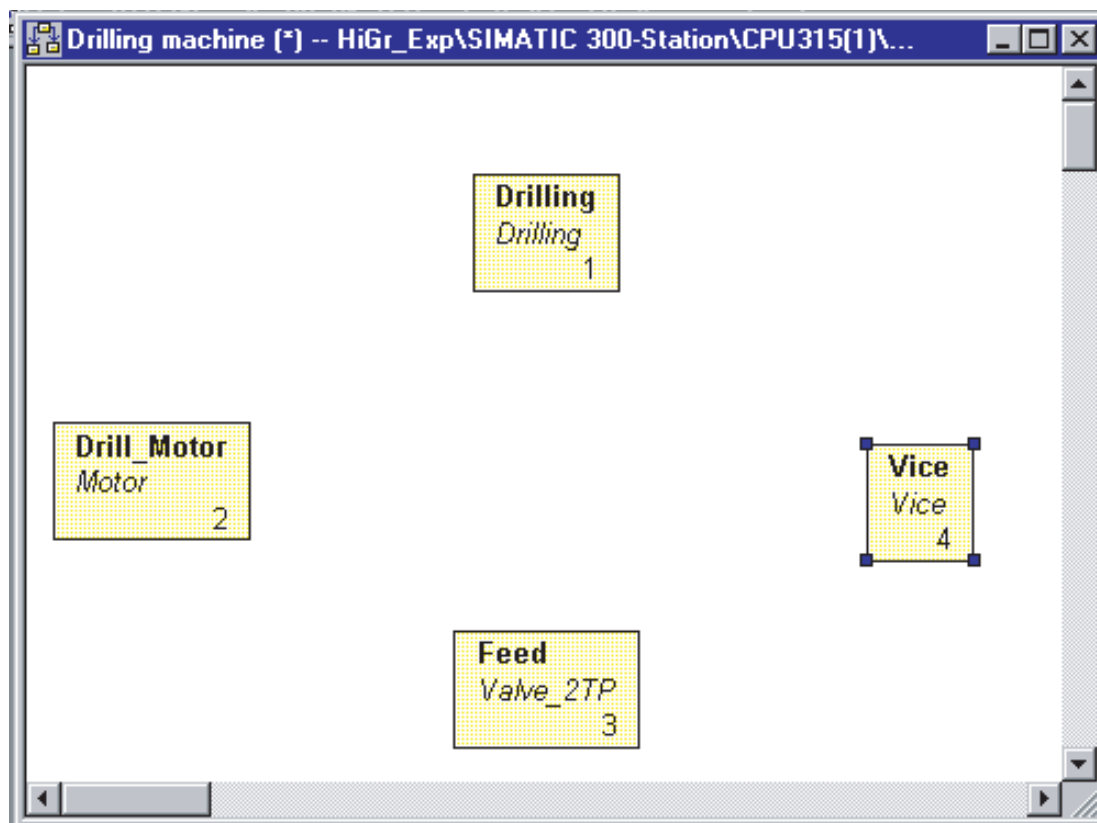
Partial step C: To define the run sequence

1. Select **Edit > Run Sequence**
2. Use the "Up" and "Down" buttons to set the run sequence:



Result

This is what your graph group looks like after you have completed the steps described earlier:



3.16 Step 12: Assigning the current parameters

In the graph group, assign current parameters to the formal parameters of the instances.

To assign current parameters

Open the detail view by selecting **View > Details** , then select the "Current Parameters" tab.

1. Select an instance and enter the current parameters listed below (shown in bold letters.)

Current parameters of the "Feed" instance

Area	Name	Data type	Current parameter	Message
IN	Top	Bool	Drill_at_top	
	Bottom	Bool	Drill_at_bottom	
OUT	Up	Bool	Drill_Up	
	Down	Bool	Drill_Down	
IN_OUT	IM_Up	Bool		In
	IM_Down	Bool		In
	OM_Top	Bool	Drilling.IM_Top	Out
	OM_Bottom	Bool	Drilling.IM_Bottom	Out

Current parameters of the "Drill_motor" instance

Area	Name	Data type	Current parameter	Message
IN	Motor_running	Bool	Drill_motor_running	
	Motor_stopped	Bool	Drill_motor_stopped	
OUT	Motor_On	Bool	Drill_motor_on	
IN_OUT	IM_Motor_Start	Bool		In
	IM_Motor_Stop	Bool		In
	OM_Motor_running	Bool	Drilling.IM_Motor_running	Out
	OM_motor_stopped	Bool	Drilling.IM_motor_stopped	Out

Current parameters of the "Vice" instance

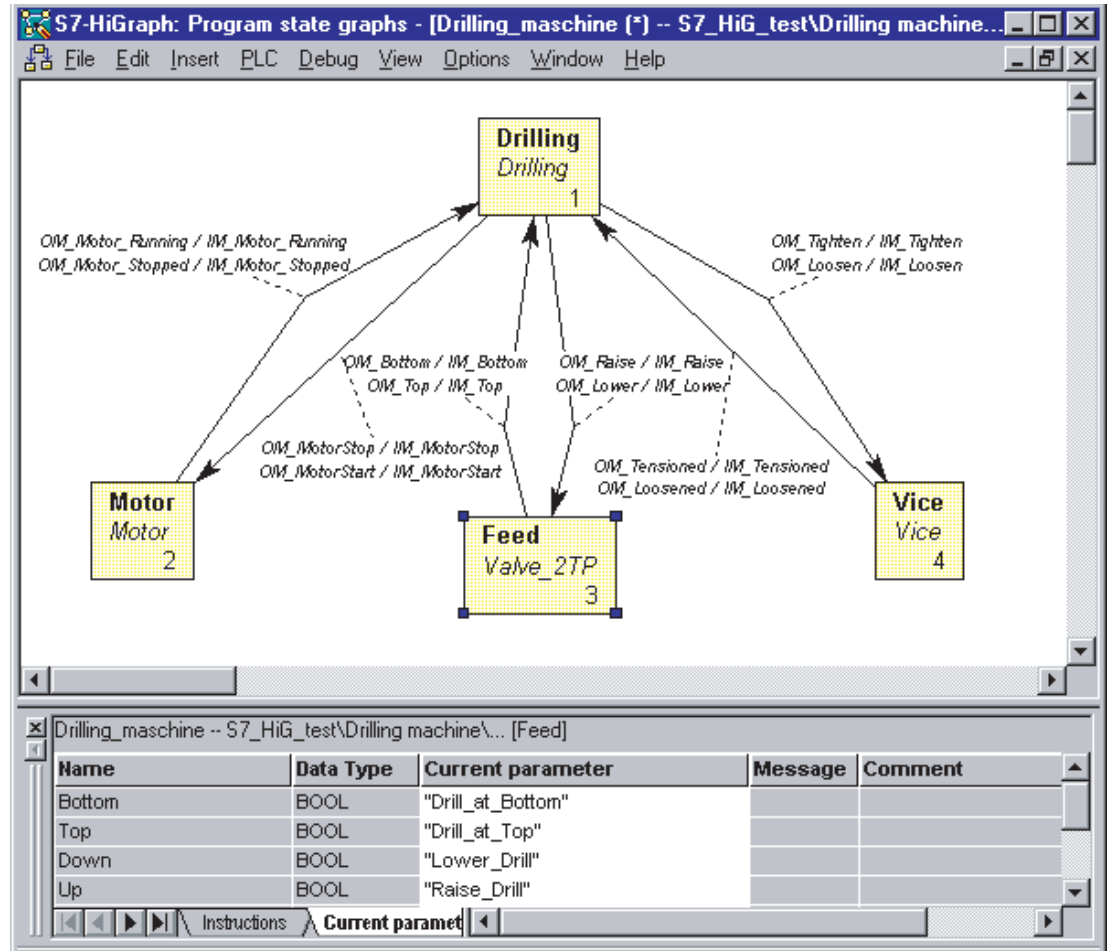
Area	Name	Data type	Current parameter	Message
IN	Pressure_reached	Bool	Clamping_pressure_reached	
OUT	Clamping	Bool	Clamp_part	
IN_OUT	IM_Clamping	Bool		In
	IM_Release	Bool		In
	OM_Clamped	Bool	Drilling.IM_Clamped	Out
	OM_Released	Bool	Drilling.IM_Released	Out

Current parameters of the "Drilling" instance

Area	Name	Data type	Current parameter	Message
IN	Start	Bool	Start_button	
IN_OUT	OM_Motor_Start	Bool	Drill_motor.IM_Motor_Start	Out
	OM_Motor_Stop	Bool	Drill_motor.IM_Motor_Stop	Out
	IM_Motor_running	Bool		In
	IM_Motor_Stopped	Bool		In
	OM_Lowering	Bool	Feed.IM_Down	Out
	OM_Raising	Bool	Feed.IM_Up	Out
	IM_Bottom	Bool		In
	IM_Top	Bool		In
	OM_Clamping	Bool	Vice.IM_Clamping	Out
	OM_Release	Bool	Vice.IM_Release	Out
	IM_Clamped	Bool		In
	IM_Released	Bool		In


Result

Structure of the graph group after you have entered the current parameters:



3.17 Step 13: Compiling the graph group

Procedure:

1. Select **Options > Settings Drilling Machine** , then select the "Compile" tab.
2. Enter the name of the blocks to be generated. Use symbolic names in this example.
 - Symbolic name of the FC: GG_Drilling_Machine
 - Symbolic name of the DB: DB_GG_Drilling_Machine
3. Set the "Cyclic actions with RLO = 0" option.
4. The remaining options do not have to be changed.
5. Select **File > Compile** 

Result

This action compiles the graph group.

3.18 Step 14: Programming OB1


In order to process the S7-HiGraph program for the drilling machine in the automation system, it must be called in OB1

Initialization of the state graphs

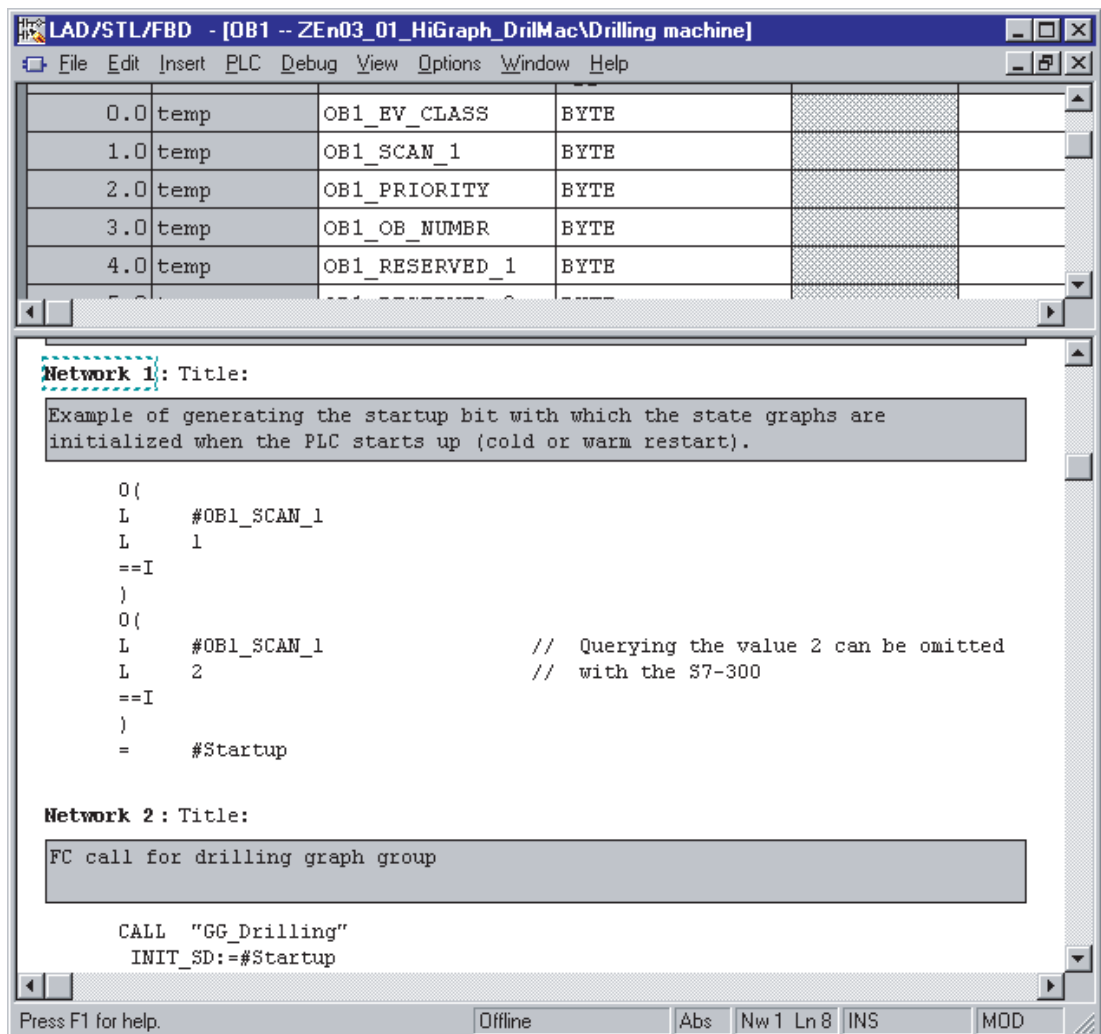
The state graphs in the graph group are initialized by the call of OB1.. The initialization is performed by means of the "INIT_SD" parameter which is contained in every graph group. Configure this parameter so that signal "1" is set after the control system is switched on, and "0" signal is set in the subsequent cycles. The signal can be generated based on the OB1 start info (#OB1_SCAN_1 variable) and stored in a temporary variable of OB1.

You program OB1 in the LAD/STL/FBD editor of the STEP 7 basic package.

Follow these steps:

1. Open OB1 in the LAD/STL/FBD editor
2. Declare a "Startup" variable of the data type BOOL.
3. Program the call of the S7-HiGraph FC as shown in the figure below.
4. Select **File > Save** 

Representation of OB1



LAD/STL/FBD - [OB1 -- ZEn03_01_HiGraph_DrillMac\Drilling machine]

Address	Variable	Symbol	DataType
0.0	temp	OB1_EV_CLASS	BYTE
1.0	temp	OB1_SCAN_1	BYTE
2.0	temp	OB1_PRIORITY	BYTE
3.0	temp	OB1_OB_NUMBR	BYTE
4.0	temp	OB1_RESERVED_1	BYTE

Network 1: Title:
Example of generating the startup bit with which the state graphs are initialized when the PLC starts up (cold or warm restart).

```

O(
  L      #OB1_SCAN_1
  L      1
  ==I
)
O(
  L      #OB1_SCAN_1           // Querying the value 2 can be omitted
  L      2                     // with the S7-300
  ==I
)
=      #Startup
    
```

Network 2: Title:
FC call for drilling graph group

```


CALL  "GG_Drilling"
INIT_SD:=#Startup
    
```

Press F1 for help. Offline Abs Nw 1 Ln 8 INS MOD

3.19 Step 15: Downloading the user program

You have to download the full "Drilling_Machine" user program (OB 1, FC, DB) to the CPU of the automation system using SIMATIC Manager.



Procedure:

1. Set the CPU to STOP
2. In your project "HiGr_Exp", open the CPU you assigned to the user program.
3. Open the S7 program and select the "Blocks" folder.
4. Select **PLC > Download** 

3.20 Step 16: Debugging the user program

You need to debug the program before you release it for runtime.

Follow these steps:

1. Set the CPU to RUN.
2. Open the graph group and select **Debug > Monitor** . You now see information on processing the graph group. The current state of each instance is shown.
3. Select an instance, then select **Edit > Open Object**.
The instance is opened ONLINE, including the following information:
 - The active state is highlighted in color.
 - The transition which lead to this state and the last active state are highlighted by shading
 - A table opens with detailed status information on the highest-priority transition outgoing from the active state.
4. Disable the **Debug > Monitor**  function to terminate monitoring mode.

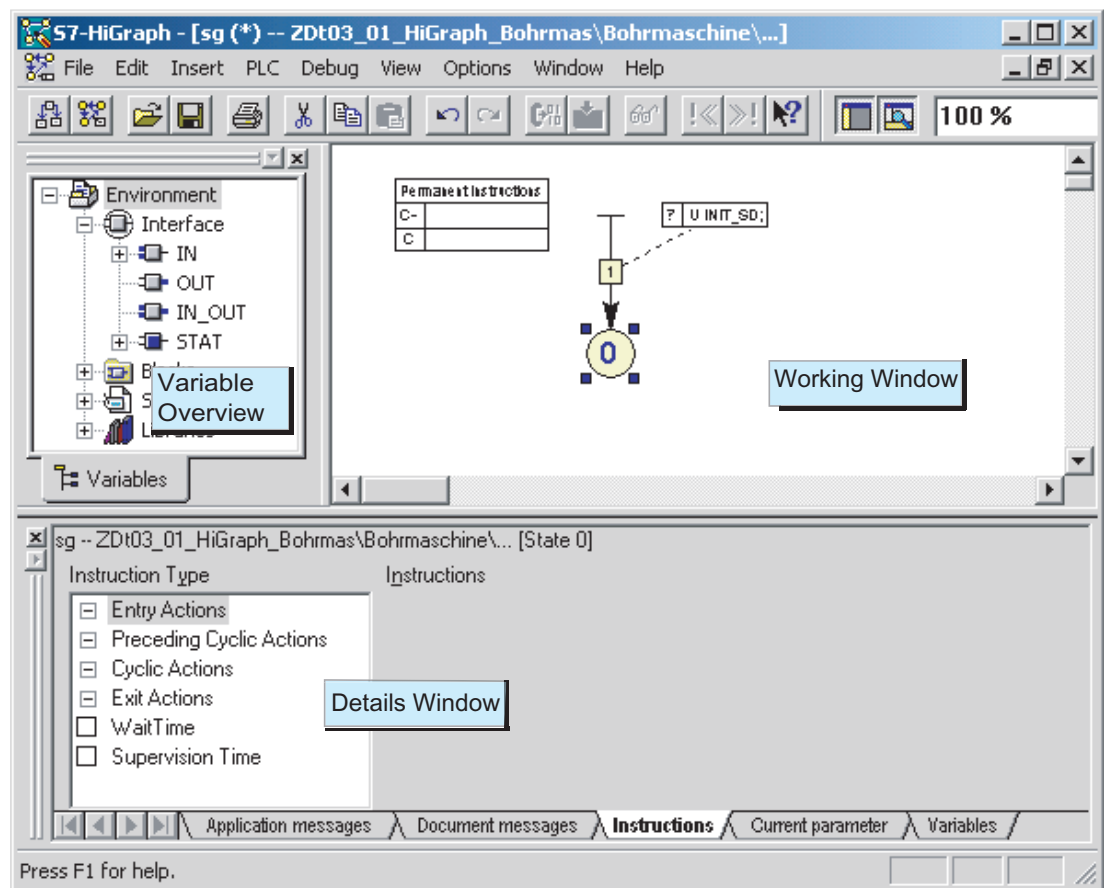
User interface

4.1 User interface

S7-HiGraph user interface

The following figure shows S7-HiGraph with a newly created state graph.

The S7-HiGraph user interface consists of various windows which you can hide or show as required. The functions of the various views are described below.



Working window

You create the diagrams of state graphs and graph groups in the working window.

Variable overview

Enter the variable declaration of the processed state graph in the variable overview.

You are shown additional program elements:

- Symbols from the symbol table
- Preconfigured blocks from the block folder
- Blocks from libraries.

You can open the window by selecting **View > Variable Overview**.

Detailed information about the declared variables and program elements are available in the details view ("Variables" tab).

To open a comprehensive Help on declaring variables, select a declaration section or a variable and then press F1.

Detail view

Select **View > Details** to open the detail view. This view contains all the relevant details you require for programming. Information is distributed to the following tabs:

Tab	Function
Application output	The "Application Messages" tab shows error messages and warnings that occur during compilation. This window is opened after each compiler run. The messages may relate to errors in graph groups or in their state group instances.
Document output	The "Document Messages" tab shows syntax errors found in the currently opened state graph or graph group.
Instructions	In the "Instructions" tab, you program the contents of states, transitions and permanent instructions.
Current parameter	In the "Current parameters" tab, you define the current parameters of instances.
Variables	<p>The "Variables" tab shows detailed information about the declared variables, based on your selection in the variable overview. The following details may be shown:</p> <ul style="list-style-type: none">• on variables• on symbols from the symbol table• on preconfigured blocks from the block folder• on blocks from the libraries <p>You can select Insert > Declaration Line to declare new variables.</p>

4.2 Customizing the working area

4.2.1 Arranging working windows

You can adapt the layout of windows opened with S7-HiGraph to suit your requirements. The following functions are available.

Show / hide and move the variable overview and details view

- You can select **View** to show or hide the variable overview and details view.
- In order to move these views, click the inner window edge and drag it to the desired position.

Arranging several working windows

- If several windows are opened, you can use the **Window > Arrange** menu command to cascade them, position them horizontally or vertically next to each other.

Hiding / showing the status bar and toolbars

- Select **View > Status Bar** or **View > Toolbars**.

4.2.2 Working with the session memory

S7-HiGraph can save the size and position of a window, the source file opened in it and the relevant screen section settings and restore these at a later time.

What is saved?

The following window settings are saved:

- Size and position of the working window
- Visibility and position of the tool bar and status bar
- Sources contained in the window
- Zoom factor, grid, page frames
- Visibility and position of the variable overview
- Displayed active state of the source

Automatic saving and restoring the layout / contents of windows

Default setting of the S7-HiGraph session memory:

- The window arrangement and content is automatically saved when you close S7-HiGraph
- The saved window layout / content is automatically restored the next time you start S7-HiGraph. Hence, you always start at the same position for your programming session, and you are saved from having to open and position document windows manually.

To change this behavior:

1. Select **File > Application Settings**
2. Select the "General" tab.
3. Deactivate the option "When exiting the application, save position and window arrangement".

Manual saving and restoring of the window layout/contents

You can also save the window settings manually:

1. Customize the working area layout to suit your requirements.
2. Select **Window > Save Arrangement** to save the current window settings and contents.
3. You can then restore your settings by selecting **Windows > Restore Arrangement**.

4.2.3 Working with the object-specific session memory

Beginning with V5.2, sources have the ability to save the window position and other window properties as object-specific properties. You can customize the settings separately for each source.

What additional information is saved?

In addition to the settings described in the "Session Memory" section, as of V5.2, sources are saved in the settings as follows:

- For graph group
 - Open instances, including their settings (zoom factor, grids, etc.)
 - Instances opened in status mode, including their settings (zoom factor, grids, etc.)
 - Visualization of external messages
- For state graphs:
 - Visibility of features
 - Visibility of instructions / permanent instructions

Automatic saving and restoring of object window arrangements / contents

Default setting of the S7-HiGraph session memory:

- The window arrangement is automatically saved when a document is saved.
- The saved window arrangement is automatically restored when you open a document.

To change this behavior:

1. Select **File > Application Settings**
2. Select the "General" tab.
3. Activate or deactivate the corresponding options.

4.2.4 Setting the colors and fonts for the working windows

You can customize the text fonts and element colors in the working windows.

Setting colors

The color settings are globally valid for the applications

1. Select **Options > Application Settings**.
2. Select the "Colors" tab, then select the relevant colors for the various elements.

Customizing the font and font size

You can define these settings separately for each state graph:

1. Select **Options > Settings for Graph Groups / State Graphs**.
2. Select the "Fonts" tab, then select the relevant fonts for the various elements.

4.2.5 Enlarging and reducing the view

You can enlarge or reduce the view of the graphics elements by setting a zoom factor.

Entering a numerical value for the zoom factor

You can explicitly define a certain zoom factor.

1. Click the zoom box on the toolbar.
2. Select a zoom factor from the drop-down list.

Entering a relative zoom factor

Or you may use one of the following commands of the **View** menu:

Use the following menu command...	...for the following purpose
Zoom > Zoom In	To enlarge the view step-by-step.
Zoom > Zoom Out	To reduce the view step-by-step.
Zoom > Normal Size	To restore the specified normal size.
Zoom > Zoom Factor...	To enter a user-specific zoom factor.
Zoom > Area Used	To select a zoom factor which displays all the objects in the working window.
Zoom > Select Zoom Area	To use the mouse to select an area you want to enlarge.
Zoom > Apply to All Windows	To apply the zoom factor of the active window to all open windows.

4.2.6 Customizing the size of the drawing area

The drawing area is the area on which you can position objects. You can customize the dimensions of this area.

Follow these steps:

1. Select **Options > Settings for Graph Groups / State Graphs**.
2. Set the required size (in mm) in the "Graphics" tab.

4.2.7 Setting the grid points

Setting a grid for the drawing area

- Select **View > Grid Points**

Setting the grid points

1. Select **Options > Settings for Graph Groups / State Graphs**.
2. Make your settings in the "Graphic" tab.

4.2.8 Showing and hiding instructions or features

You can show or hide the following elements in order to optimize the graphic view of a state graph:

- Instructions
- Permanent instructions
- Features of states and transitions

Procedure:

- Select **View > Show With > Statement Lists / Features / Permanent Instructions**.

Working with S7-HiGraph

5.1 Reusability of state graphs (type / instance concept)

Reusing state graphs

The state graphs you created for a certain functional unit can be reused at other program points at which a similar functional unit is required.

All the state graphs which you have programmed within an S7 program are saved centrally in the "Sources" folder. From there, you can add them as often as required to one or more graph groups, and thus call them.

The call of a state graph in a graph group is called an instance.

Central modification of state graphs

State graphs can be edited centrally: All modifications of a state graph apply to all of its instances.

Principle of reuse / central modification

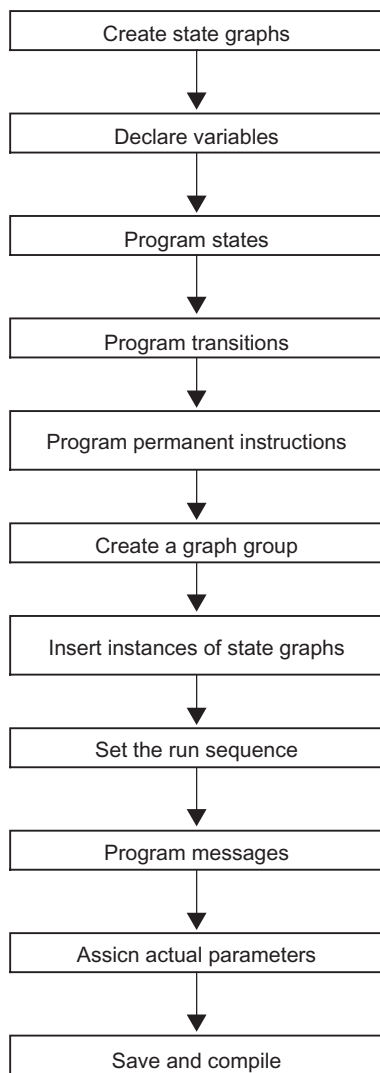
To enable multiple reuse of state graphs, declare all addresses used in a state graph as formal parameter (placeholder.)

Assign the current parameters (physical I/O, for example) to these formal parameters when you insert the graph into a graph group. This allows you to assign each instance of a state graph separate current parameters.

5.2 Steps in creating a program

Steps in creating a program

The flow chart provides an overview of all necessary steps.



5.3 Setting up a STEP 7 project

Before you start programming in S7-HiGraph, you first need to create a project and the corresponding symbol table in SIMATIC Manager:

Creating a project

1. In SIMATIC Manager, select **File > New > Project**.
2. Enter a name for the project and open it.

Creating a symbol table

If you want to use symbolic addresses for programming, it is advisable to create the symbol table before you start programming.

1. Open the symbol table of your S7 program in SIMATIC Manager.
2. Enter the required symbols in the table.

Note

STEP 7 provides a Wizard which helps you in creating a complete project structure. Select **File > Wizard "New Project"** to open the Wizard.

5.4 Starting S7-HiGraph

Starting it from the Windows user interface

After you have installed the software on your programming device/personal computer, you can call up the S7-HiGraph via the "Start" command button on the Windows task bar in (entry under "SIMATIC/STEP 7").

Starting S7-HiGraph from SIMATIC Manager

In SIMATIC Manager, you can also start S7-HiGraph by double-clicking a graph group or a state graph in the "Sources" folder.

5.5 Creating and opening state graphs

Creating state graphs

You can create state graphs either in SIMATIC Manager or in S7-HiGraph.

In the application...	..proceed as follows:
SIMATIC Manager	<ol style="list-style-type: none"> 1. Open the "Sources" folder in your S7 program. 2. Select Insert > S7 Software > Status Graph.
S7-HiGraph	<ol style="list-style-type: none"> 1. Select File > New State Graph. The "New" dialog box opens. The source file of your S7 Program is already selected and the state graph has already been entered as object type. 2. Type in the desired name in the "File name" input box and confirm your entries with "OK."

A new state graph contains an initial state, a startup transition and the permanent instructions list.

Opening state graphs

You can open state graphs either in SIMATIC Manager or in S7-HiGraph.

In the application...	..proceed as follows:
SIMATIC Manager	<ol style="list-style-type: none"> 1. Open the "Sources" folder in your S7 program. 2. You can open existing state graphs in this folder with double-click.
S7-HiGraph	<ol style="list-style-type: none"> 1. Select File > Open. 2. On the next dialog box, select the "Sources" folder of your S7 program. 3. Select the "State graph" from the "Object type" selection box. 4. Type in the relevant name in the "File name" input box and confirm your entries with "OK."

5.6 Declaring variables

5.6.1 Meaning of the variable declaration

Definition

A variable is a dynamic program value (parameter.) All variables used in a block must be declared separately. This variable declaration represents the definition.

In the variable declaration, the user determines that a parameter is a variable, and specifies its type by assigning it to a data type. The declaration reserves sufficient memory in the data block.

Meaning of the variable declaration in S7-HiGraph

Define all parameters used in a state graph as variables, so that you can reuse the state graphs without having to adapt the parameters.

These are in particular:

- Input, output and in-out parameters of the state graph.
These parameters form the "interface" used in the program for calling the state graph.
- Parameter used by the state graph to exchange messages.
- Further variables you may want to use for programming

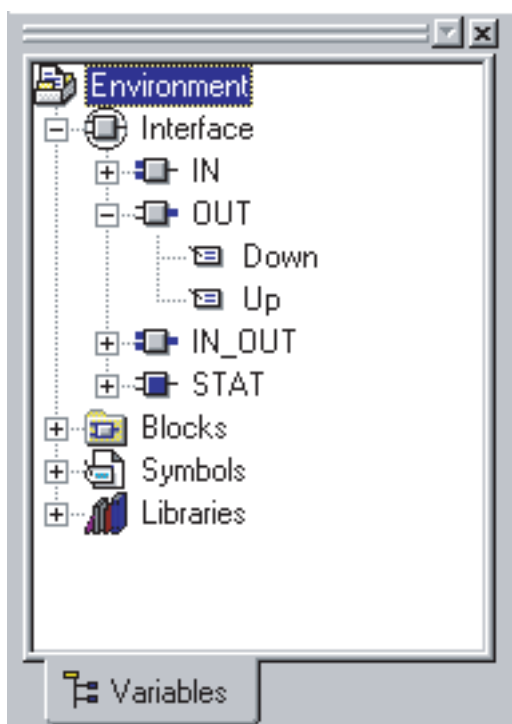
5.6.2 The variable declaration view

The variable declaration consists of the variable overview and of the variable detail view. You can show or hide these views by calling the View menu.

Variable overview

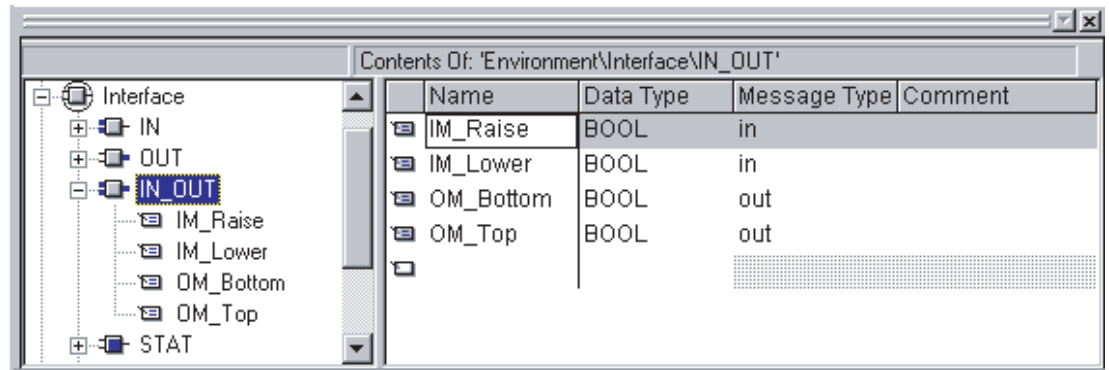
The variable overview is always located on the left side of the working area. This area shows the declaration sections (IN, OUT, IN_OUT, STAT).

It also shows elements of the current project, for example, the symbol table and blocks of the block folder and libraries.



Variable detail view

Select the "variables" tab in the detail view to view details about the declared variables, based on your selection in the variable overview.



5.6.3 Declaration sections

Sections in the variable declaration

The variable declaration is organized by the following declaration sections:

Declaration section	Meaning
IN	Contains the input parameters of the state graph and the default "AutomaticMode" and "ManualMode" variables.
OUT	Contains the output parameters of the state graph.
IN_OUT	Contains the in/out parameters of the state graph. Parameters you want to use to exchange messages have to be declared in this section.
STAT	Contains the static variables which are not used as formal parameters. The variables are allocated directly in the data block. These are: <ul style="list-style-type: none"> Default variables of S7-HiGraph. These cannot be edited. User-specific local static variables.

5.6.4 Information in the variable detail view

Information in the variable detail view

The variable detail view contains information in various columns

The table below shows these columns and their meaning:

Column	Meaning	Possible values	Default
Name	Symbolic name of the variable	Variable naming conventions: <ul style="list-style-type: none"> Valid characters: letters, numerals and the underscore (_). A name always begins with a letter or an underscore. A name may not be delimited with an underscore character. Two consecutive underscore characters are not permitted. Keywords are not allowed 	-
Data type	Data type of the variables	BOOL, INT, WORD, etc. (a selection is offered)	BOOL
Message type	Messages are used to coordinate state graphs. Messages must be declared in the IN_OUT declaration section. The "Message type" column therefore appears only in this section.	IN for incoming messages, OUT for outgoing messages.	-
Comment	Comment on variable documentation	Can be edited freely	-

5.6.5 Steps in entering the variable declaration

Procedure:

1. Select the relevant declaration section from the variable overview.
2. Type in the variable name in the "Name" column of the detail view.

You can choose one of two options:

- Enter the name in the last free line.
- Select **New Declaration Line** from the shortcut menu, then add a new line at any position for entering the name.

3. Press [ENTER].

This confirms your entries and adds an empty line to the variable declaration.

4. A further variable name can now be entered in the new line.
5. The "Data type" BOOL is entered automatically. To specify a different data type for the variable, click the arrow in the "Data type" column, then select a data type from the drop-down list.

5.6.6 Default variables

Use of default variables:

Programming in S7-HiGraph is facilitated by a number of default variables.

These are variables which you can

- dynamically assign values in the program
- and request their values.

Declaring the default variables

These variables are entered automatically in the variable declaration when a state graph is created. You can neither rename, nor delete these variables, nor change their data type.

Functions of the various default variables

Available default variables:

Default variables	Meaning	Declaration section	Data type	Assigned by user	Name in HiGraph V2.7
ManualMode	Input variable for setting manual mode. If this variable carries a "1" signal , only the transitions with the "Manual" attribute are checked. The variable may not carry a "1" signal at the same time as AutomaticMode.	IN	BOOL	x	BA_MANUAL
AutomaticMode	Input variable for setting auto mode. If this variable carries a "1" signal , only the transitions with the "Auto" attribute are checked. The variable may not carry a "1" signal at the same time as ManualMode.	IN	BOOL	x	BA_AUTO
INIT_SD	The INIT_SD variable serves as startup parameter. When the variable carries a "1" signal , a startup is signaled to the state graph.	STAT	BOOL	x	STARTUP
CurrentState	Current state	STAT	WORD		CURRENT

Default variables	Meaning	Declaration section	Data type	Assigned by user	Name in HiGraph V2.7
	This variable can be read in conditions. It contains the number of the current state. *				_STATE
PreviousState	Previous state This variable can be read in conditions. It contains the number of the last active state. *	STAT	WORD		LAST_STATE
StateChange	State transition This variable can be read in conditions. It carries a 1 signal in those cycles in which a state transition takes place. In all other cycles it carries the signal 0.*	STAT	BOOL		STATUS_CHANGE
ST_Expired	Monitoring time expired This variable can be read in conditions. *	STAT	BOOL		ST_ERROR
ST_ExpiredPrev	Monitoring time of the last state has expired This variable can be read in conditions *	STAT	BOOL		T_ERROR_OLD
ST_Stop	Monitoring time stopped The monitoring time is interrupted as long as this variable carries a "1" signal.	STAT	BOOL	x	STOP_WATCH TIME
ST_CurrValue	Monitoring time to go	STAT	DWORD		-
ST_Valid	Monitoring time active This variable is only of internal significance	STAT	BOOL		-
WT_Expired	Waiting time expired This variable can be read in conditions.	STAT	BOOL		-
WT_Stop	Waiting time stopped The waiting time is interrupted as long as this variable carries a "1" signal.	STAT	BOOL	x	STOP_WAIT TIME
WT_CurrValue	Waiting time to go	STAT	DWORD		-

Default variables	Meaning	Declaration section	Data type	Assigned by user	Name in HiGraph V2.7
WT_Valid	Waiting time active This variable is only of internal significance	STAT	BOOL		-
DT_Expired	Delay time expired This variable will only be created when the first delay time is configured.	STAT	BOOL		-
UsrMsgSend	Message state active This variable carries a "1" signal when a message state is active (only relevant for diagnostics with format converter.)	STAT	BOOL		-
UsrMsgQuit	Input variable for error/ message acknowledgement (only relevant for diagnostics with format converter).	IN	BOOL	X	-
UsrMsgStat	This variable is only of internal significance (only relevant for diagnostics with format converter).	STAT	WORD		-

* The section "Overview of cyclic execution of a state" explains in detail when and for which period the variables are set.

Inactive variable

The following variables are inactive immediately after a new state graph has been created:

- CurrentState
- PreviousState
- StateChange
- WT_Stop
- ST_Stop
- DT_Expired

To enable the variables

Procedure:

1. Select the variable from the variable overview.
2. Select **Edit > Object Properties**
3. On the next dialog box, select the "Attributes" tab.
4. Set the value "true" at the "S7_active" attribute.

5.6.7 Interaction between variable declarations and instructions

S7-HiGraph automatically applies the

variables from the variable declaration are used in the instructions you program in states and transitions. Changes in the variable declaration therefore always have an effect on one or more instructions. S7-HiGraph tracks such changes automatically in order to save you the tedious task of tracking them by hand.

Effects of changes in the variable declaration:

Changes in the variable declaration have in particular the following effect:

Action in the variable declaration	Reaction in the instructions
Correct change of a name without changing the data type	The variable immediately appears in all instructions under its new name
Changing the data type	Invalid instructions may become valid and, vice versa, valid instructions may become invalid.
A valid name is replaced with an invalid name	Instructions are not changed
Deleting a variable which is used in instructions	Valid instruction becomes invalid

5.6.8 Interaction between variable declarations and current parameter assignments

Interaction between variable declarations and current parameters

After you have inserted a state graph as an instance into a graph group, assign current parameters to the variables used in the state graph.

Subsequent changes in the variable declaration may affect current parameter assignments and, in the worst case, invalidate them.

Effects of changes in the variable declaration:

Changes in the variable declaration have in particular the following effects on state graph instances:

Action in the variable declaration	Reaction in the current parameter assignment
Correct change of a name without changing the data type	The old name is retained in the current parameter assignment, but is displayed in red. The new name is added, but is not assigned current parameters. All you do now is to transfer the current parameter assignment of the invalid name to the new name and then delete the name marked in red.
Changing the data type	Invalid assignments may become valid and valid assignments may become invalid.
A valid name is replaced with an invalid name	Assignment is not changed.
Deleting a variable which is used in instructions	Valid assignment becomes invalid.

Note

Closed graph groups are ignored when the variable names are being adapted.

5.7 Programming the structure of a state graph

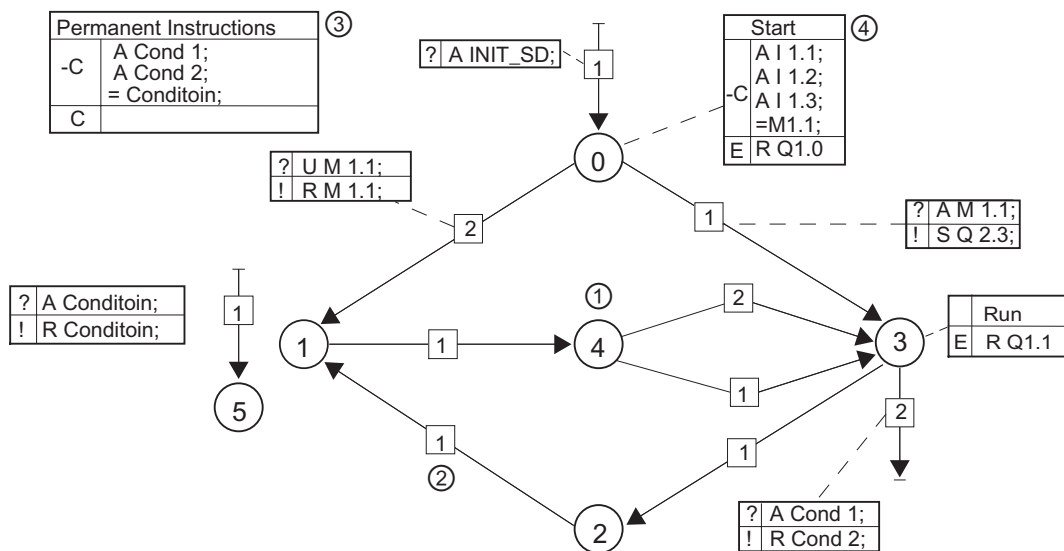
5.7.1 Elements of a state graph

Elements of a state graph

Elements in the graphic representation of a state graph:

- States (1)
- Transitions (2)
- Permanent instructions (3)
- Instructions in states or transitions (4)

The figure below shows an example of the structure and elements of a state graph.



5.7.2 Rules for the structure of state graphs and groups

Data volume

State graphs and graph groups must conform to the following data volume:

- Maximum number of elements in a state graph:
 - 4090 States
 - 4090 Transitions
- A graph group may contain up to 255 instances.

Required memory space

CPU memory requirements for S7-HiGraph programs are determined by the number of states, transitions and instances. For further information, refer to "Memory requirements of the user program."

See also

Memory requirements of the user program (Page 7-6)

5.7.3 Options of aligning graphic objects

Using the grid

The grid serves as an aid for the precise alignment and positioning of objects. To use the grid:

Function	Menu command
Setting a grid for the drawing area	View > Grid
Setting the grid points	Options > Settings for Graph Groups / State Graphs
Moving selected objects to the next grid position	Options > Align to Grid
Automatic alignment of inserted or moved objects to the grid	Options > Snap to Grid

Aligning to other objects

The following menu commands facilitate symmetrical alignment of elements:

Function	Procedure
Aligning several objects to a common vertical or horizontal position	Options > Align > To Object > Vertically/Horizontally
To align objects with even spacing	Options > Align > To Distance > Vertically/Horizontally
To place an object before or after another object	Options > Forwards or Options > Backwards

Layout on the page

You can show the print page frames while drawing if you want to adapt the layout of the state graphs or graph groups to the format of the future print page.

Function	Menu command
Showing the print page frames	View > Print Page Frame
To center a selected object or group of objects exactly to the nearest print page.	Options > Align > To Page

Aligning lines

Transition and message lines do not always have to follow a straight course. If the graphics are complex, bending the lines may give you a better overview.

1. Lines can be kinked by clicking the square in the line middle and dragging it in any direction.

Whenever you kink a line, it is separated into two sections. Further nodes appear at the center of each line segment.

2. Click these nodes to kink the line segments once again.
3. Select **Options > Straighten Line** to straighten the lines again or delete individual nodes.

5.7.4 States

Definition

A state represents a programmed unit of a state graph. It represents the state a controlled technological function may assume. Only one state of the state graph is active during program execution.

Initial state

The initial state represents a special form of a state. It determines which state a functional unit should assume at power on. Each state graph requires an initial state.

In the initial state it is possible to check whether the functional unit is in a defined initial position. If required, it can be returned to its initial position.

Instructions in states

A states is assigned instructions which are executed when it is active. The time of execution of an instruction can be defined: At the entry into a state, when dwelling in a state, or when leaving a state.

Visualization in the state graph diagram

A state is indicated as a circle. Each state of a state graph is assigned a unique number. You can also assign names to the states in order to improve the view.

5.7.4.1 Steps in inserting states

Procedure:

1. Open a state graph window.
2. Select **Insert > State**.
The cursor then changes its shape to assumes an inserting function.
3. Click the relevant insert position.
4. Add further states.
5. Press ESC to exit insertion mode and return to editing mode.

5.7.4.2 Assigning the state name, number and comment

State properties

You can assign the following properties to the states. These do not have any influence on program runtime.

- **State name**
The state name is indicated in the instruction table of the state. These names are also indicated in the context of process error diagnostics.
- **Status number**
State graphs with successive numbering require the least memory. Inserted states are automatically assigned successive numbers. However, gaps in the numbering may develop when you delete states. In this case, you can change the numbering in order to eliminate these gaps.
- **Status comment**
A well commented user programs is easier to interpret and to maintain.

Assigning names, numbers and comments

1. Select a status.
2. Select **Edit > Object Properties**
3. On the next dialog box, enter the state name, number and comment.

5.7.4.3 Assigning state features

Status features

You can assign features to states for the purpose of diagnostics.

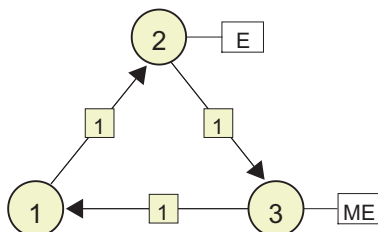
Characteristic	Function	Abbreviation
Error	Outputs an alarm message to the diagnostics program	F
Message	Outputs a status message to the diagnostics program	M

Assigning features

1. Select a status.
2. Select **Edit > Object Properties**
3. On the next dialog box, select the relevant features.

Visualization of features

The figure below shows a state graph containing states which are assigned features.



The F and M features may also be shown directly in the state circle.

1. Select **Options > Application Settings**.
2. Click the relevant option on the "Display" tab.

5.7.4.4 Handling states

Selecting

Select the state before you edit it. Options available:

Function	Procedure
Selecting individual states	Click the state.
Selecting several states	<ol style="list-style-type: none">1. Press CTRL and hold it down2. Click the various states in successive order.
Selecting several states using the lasso function.	<p>You can also select several states by using the lasso function.</p> <ol style="list-style-type: none">1. Place the mouse pointer on the drawing area.2. Keep the mouse button pressed and draw a line around the relevant states.
Selecting elements for copying	<p>You can also use the lasso function to select elements for copying:</p> <ol style="list-style-type: none">1. Place the mouse pointer on the drawing area.2. Press SHIFT.3. Keep the mouse button pressed and draw a line around the relevant elements.

Move

You can move states within a state graph

1. Select one or several states.
2. Click one of the selected states.
3. Drag-and-drop the selected element(s) to the relevant position.
4. Select **View > Update** if the screen display was distorted by the move operation.

Copy

You can copy states within a state graph and to other state graphs.

1. Select one or several states.
2. Select **Edit > Copy**.
3. Select **Edit > Paste**. The cursor then changes its shape and assumes an inserting function. Click with the insertion cursor on the relevant insert position on the drawing area.

Cut

You can cut selected states to copy these to the clipboard and paste these into another location.

1. Select a status.
2. Select **Edit > Cut**.

Delete

You can delete selected states without copying these to the clipboard.

1. Select a status.
2. Select **Edit > Delete**.

5.7.5 Transitions

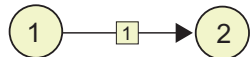
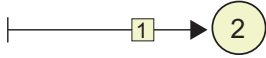
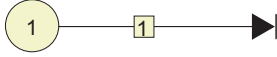
Definition

A transition controls the transition between two states. Each transition is assigned conditions which control the transition to the next state. A status transition is carried out if all conditions of a transition are satisfied.

Transition priority

Several transitions can lead from one state. In this case the transitions are assigned priorities. If the conditions of several transitions are satisfied, the transition with the highest priority is executed, the highest possible priority being 1.

Transition types

Transition type	Function	Visualization
Normal transition	A normal transition leads from one state to the next state.	
Any transition	<p>The Any transition leads from all states to a target state. Any transitions are processed continuously, irrespective of the current state of a state graph. They are used, for example, for permanent monitoring of high-priority conditions (for example, EMERGENCY-OFF.) If a monitoring event programmed in the Any transition is triggered, the system branches to the target state.</p> <p>If a state graph has several Any transitions, an individual priority is assigned to each Any transition. The priority of the Any transitions are evaluated separately from the priorities of other transitions: Any transitions always take priority over normal transitions.</p> <p>An Any transition which queries the predefined variable INIT_SD is treated as a startup transition. It is used to initialize the state graph.</p>	
Return transition	A Return transition leads from the current state back to the previously active state. Return transitions do not take priority over normal transitions.	

5.7.5.1 Steps in inserting transitions

Procedure:

1. Open a state graph window.
2. Select **Insert > Transition**.
The cursor then changes its shape to assumes an inserting function.
3. Click the start point of the transition.
4. Drag-and-drop the cursor to the destination.
5. Release the mouse button.
6. Add further transitions.
7. Press ESC to exit insertion mode and return to editing mode.

Transition types

The following transitions are created based on the position of their end points:

Transition type	Placing
Normal transition	between two states
Any transition	pointing from any point of the drawing area to a state
Return transition	outgoing from a state to any point of the drawing area

5.7.5.2 Specifying the transition priority

Transition priority

Several transitions can lead from one state. In this case the transitions are assigned priorities. If the conditions of several transitions are satisfied, the transition with the highest priority is executed, the highest possible priority being 1. The priority is indicated in a small square at the transition arrow.

Defining the priority

S7-HiGraph determines transition priorities based on the order in which transitions were created. To change an automatically set priority:

1. Select the transition.
2. Select **Edit > Object Properties**
3. Enter the priority on the next dialog box. Transition priorities do not have to be assigned in successive order.

5.7.5.3 Assigning the transition name and comment

Transition name and comment

You can assign names and comments to the transitions. These do not have any influence on program runtime.

Transition property	Function
Transition name	The transition name is indicated in the corresponding instruction table. The name is also shown on a connected HMI.
Transition comment	A well commented user programs is easier to interpret and to maintain. The comments are also shown on a connected HMI.

Procedure:

1. Select a transition.
2. Select **Edit > Object Properties**
3. On the next dialog box, enter the transition name and comment.

5.7.5.4 Assigning transition features

Transition features

You can assign transition features in order to program operating modes and evaluate waiting times.

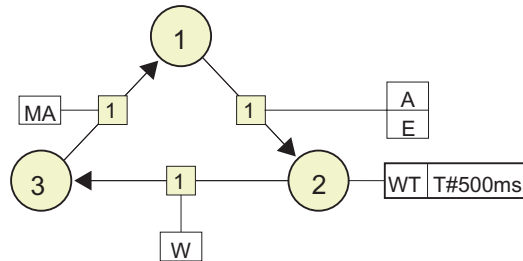
Characteristic	Function	Abbreviation
Manual	Transition is executed only in manual mode	H
Auto	Transition is executed only Auto mode.	A
Waits	The waiting time you configured in the output state is evaluated when executing a transition.	W
Error	Identifies the transition as an error transition. Such a transition is only used for documentation. It is not output to the HMI.	F

Assigning features

1. Select a transition.
2. Select **Edit > Object Properties**
3. On the next dialog box, select the relevant features.

Visualization of features

The figure below shows a state graph containing transitions which are assigned features.



The abbreviations MA, A, W and E can also be displayed directly in the transition node.

1. Select **Options > Application Settings**.
2. Select the relevant option on the "Display" tab.

5.7.5.5 Handling transitions

Selecting

Select the transition before you edit it. Options available:

Function	Procedure
Select individual transitions	Click the transition.
Selecting multiple transitions	<ol style="list-style-type: none"> 1. Press CTRL and hold it down. 2. Click the various transitions in successive order.
Selecting several transitions using the lasso function.	<p>You can also select several transitions by using the lasso function.</p> <ol style="list-style-type: none"> 1. Place the mouse pointer on the drawing area. 2. Keep the mouse button pressed and draw a line around the relevant transitions.
Selecting elements for copying	<p>You can also use the lasso function to select elements for copying:</p> <ol style="list-style-type: none"> 1. Place the mouse pointer on the drawing area. 2. Press SHIFT. 3. Keep the mouse button pressed and draw a line around the relevant elements.

Move

You can move complete transitions within a state graph

1. Draw a lasso around the entire transition.
2. Select **Edit > Cut**.
3. Select **Edit > Paste**.

The cursor then changes its shape and assumes an inserting function.

4. With the insertion cursor, click the relevant insert position on the drawing area.

Moving start or end points

You can move the start and end points of a transition in order to change the interconnections between states.

1. Select the start or end point of a transition.
2. Drag-and-drop it to the relevant state.

Copy

You can copy transitions within a state graph and to other state graphs.

1. Select a transition by clicking the box in the center of the transition.
2. Select **Edit > Copy**.
3. Select **Edit > Paste**. The cursor then changes its shape and assumes an inserting function. With the insertion cursor, click the relevant insert position on the drawing area.

Aligning transition lines

Transition lines do not always have to run along a straight course. If the graphics are complex, bending the lines may give you a better overview.

1. Transition lines can be kinked by clicking the square in the line middle and dragging it in any direction.

Whenever you kink a transition line, it is separated into two sections. Further nodes appear at the center of each line segment.

2. Click these nodes to kink the line segments once again.
3. Select **Options > Straighten Line** to straighten the lines again or delete individual nodes.

5.7.6 Permanent instructions

Definition

Permanent Instructions	
C-	
C	

Permanent instructions are executed once per execution cycle of the state graph, irrespective of the current state.

The following types of permanent instructions are available (refer to the sequence diagram in the section "Cyclic processing of a state in the PLC"):

Instruction types	ID	Description
Preceding cyclic actions (permanent)	(C-)	Are always executed at the beginning of a cycle.
Cyclic actions (permanent)	(C)	Are always executed at the end of a cycle.

Fields of application

In permanent instructions you can, for example, program the following operations centrally:

- Calculation of process variables which you query at several points.
- Acquisition and processing of events to which the system has to react, irrespective of the current state (for example, monitoring of a safety guard.)

See also

Overview of cyclic execution of a state (Page 7-2)

5.7.6.1 Steps in inserting permanent instructions

Procedure:

Each state graph contains an instruction table titled "Permanent instructions." There you enter the instructions.

1. Double-click in the "Permanent instructions" instruction table to open the "Instructions" tab.
2. Select an instruction type in the left-hand section of the tab (preceding cyclic action or cyclic action).
3. If there are no conditions yet, right click to open the context-sensitive menu and select **Insert** to create these.
4. Enter the instructions in the right-hand window pane.

5.8 Programming instructions

5.8.1 Instructions in states and transitions / permanent Instructions

Definition

Instructions represent process control commands. These are used to call blocks or control their process I/Os, for example.

Instructions can be assigned to states or transitions. Instructions are executed when a relevant state or transition is active.

You can program permanent instructions which are executed once in each state graph cycle, irrespective of states or transitions.

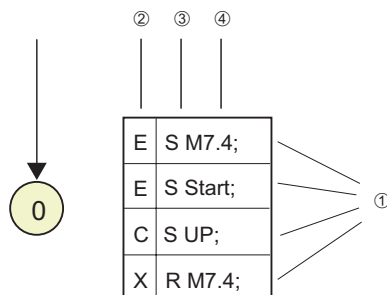
Instruction table

Instructions appear in the state graph diagram in table format:

The figure below shows an instruction table which is assigned to a state. The various elements are described in the paragraph below.

Information in the instruction table

The instruction table contains the following information:

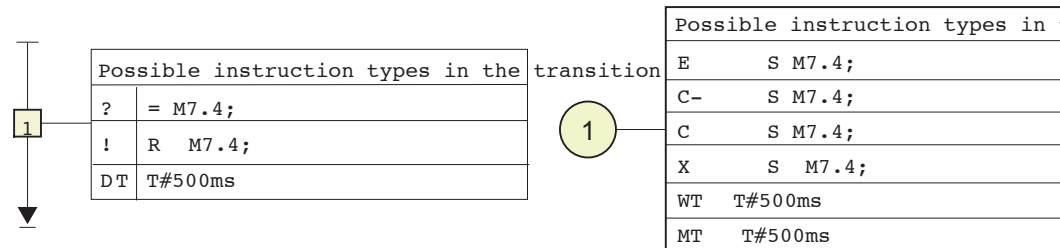


Element	Function
(1) Instruction blocks	An instruction block represents a line in the instruction table. It is advisable to install each instruction in its own block. However, you may also insert several instructions into a shared block.
(2) Instruction type	S7-HiGraph differentiates among several instruction types which determine more specifically when an instruction is executed (for example, when entering, during or leaving the state.) The instruction types are represented by the abbreviations E, C-, C and X. Several instruction blocks of the same type may exist (for example, the example in the figure contains two instruction blocks of Type E).
(3) (4) Instruction	The second column contains the actual instruction. It consists of two elements: <ul style="list-style-type: none"> An operation (3) which defines what the processor should do, for example, set (S) or reset (R) a bit or assign a value (=). An address (4) which defines what element the processor should use, for example, an input or output (I x.y, O x.y) or a flag (Mx.y)

5.8.2 Instruction types

Overview

The figure shows the instruction types available for states and transitions.



Function of the instruction types

The various instruction types have the following functions:

Instruction type	ID	Function	Can be used in
Entry actions	E	actions which are carried out once when entering a state	States
Preceding cyclic actions	C-	Actions executed while dwelling in a state before the step transitions are checked and containing state-specific logic operations for conditions.	States Permanent instructions
Cyclic actions	C	Actions which are carried out while dwelling in a state after the transitions have been checked	States Permanent instructions
Exit actions	X	Actions which are carried out once when exiting a state	States
Waiting times	WT	Specifies whether the control system is to dwell in a state for a minimum period.	States
Monitoring times	MT	Specifies whether the duration of dwelling in a state is to be monitored.	States
Delay times	DT	Defines the delay time for the action of a conditional transition.	Transitions
Conditions	?	These instructions describe the conditions which must be satisfied before a status transition can take place.	Transitions
Transition actions	!	These instructions are carried out once when the transition is triggered.	Transitions

5.8.3 Rules for entering STL instructions

Rules

The following basic rules must be observed when entering instructions in STL format:

Topic	Rule
Syntax	The syntax follows the rules for STL sources. For an exact description of the syntax refer to the online help for STL references.
Lines	Each instruction is written to a separate line.
Instruction blocks	The instructions of type can be distributed to different instruction blocks in order to improve the structure. This facilitates the automatic resetting of signals which were set while dwelling in the state.
Semicolon	Every line is terminated with a semicolon.
Upper case / Lower case	The program is not case-sensitive with respect to the entry of operations, symbols or absolute addresses.
Addresses	In order to ensure that state graphs can be used several times, you should only use declared variables as addresses. After you inserted the state graph as an instance in a graph group, you can assign symbolic or absolute addresses to these variables as current parameters.
RLO	Processing of an instruction table always starts with RLO = 1.
Nesting stack	Monitor the depth of the nesting stack yourself, because this is not checked during compiling. The nesting stack may contain up to seven entries. A violation of this limit is not reported as an error.
Indirect addressing	Indirect addressing is not permitted.

See also

Overview of cyclic execution of a state (Page 7-2)

Valid data types (Page 8-10)

5.8.4 Settings for STL Instructions

Setting the mnemonics

You have the following options of setting instruction mnemonics:

- German (for example, E1.0)
- English (for example, I 1.0)

Procedure:

1. Before you open an S7-HiGraph source in SIMATIC Manager, select **Options > Settings**
2. Set the mnemonics in the "Language" tab.

S7-HiGraph now interprets your entries in accordance with the mnemonics set.

Note

Please note that existing instructions are not automatically adapted to the new mnemonics settings.

See also

Valid data types (Page 8-10)

5.8.5 Steps in entering STL instructions

Procedure:

Enter the instructions as follows:

1. Double-click the element for which you want to program an instruction. This can be a state, a transition or the "Permanent instructions" field.
The "Instruction" tab opens.
2. Select an instruction type in the left-hand section of the "Instructions" tab (for example, entry action, cyclic action, etc.).
3. Determine an instruction block:
 - You can select an existing instruction block with a mouse click.
 - You can also select "Insert" from the shortcut menu to add a new instruction block.
4. Enter the instructions in the right-hand section of the "Instructions" tab.
5. In order to include symbolic names defined in the symbol table, select **Insert > Symbol**. This opens a list of all symbols. Select the relevant symbol.
6. The instructions are shown in a table on your working window. Drag-and-drop this table to a suitable position on the working window.

Syntax check

A syntax check is performed after you have concluded a line. The faulty line is highlighted in red color, and the corresponding syntax error is described in the "Document Message" tab. You can either eliminate the error immediately, or accept the faulty instruction and debug the line later.

Overview of available instructions

An overview of all instructions available for use in S7-HiGraph is found under "STL Language Description" in the "STL instructions" section.

See also

Valid data types (Page 8-10)

Overview of cyclic execution of a state (Page 7-2)

STL instructions (Page 8-1)

5.8.6 Programming with absolute or symbolic addresses

Advantages of symbolic programming

You use addresses such as I/O addresses, memory bits, counters, timers, data blocks and code blocks as current parameters for an S7-HiGraph program. You can address these parameters in your program by using absolute addresses (for example, I 1.1, M 2.0, FB21.) However, you considerably improve legibility of your programs by using symbolic names, rather than absolute addresses (for example, "Motor_On" or other descriptions according to one of the naming conventions used in your field of industry). This enables symbolic addressing in your user program.

Absolute addressing

An absolute address consists of an address ID and an address (O 4.0, I 1.1, M 2.0, FB 21, for example.)

Symbolic addressing

You achieve a clearer program layout and simplify troubleshooting by assigning symbolic names to the absolute addresses.

STEP 7 can automatically compile the symbolic names into the required absolute addresses. If you prefer using symbolic names to access inputs, outputs, flags, DBs, timers and counters, you first need to enable symbolic addressing of these data by assigning symbolic names to the absolute addresses.

You can assign address O 4.0 the symbolic name MOTOR_ON, and then use this as an address in a program instruction, for example. Symbolic addressing lets you easily check the consistency between program elements and components of the process control project.

Assignment of absolute and symbolic addresses in the symbol table

You assign symbolic names to the absolute addresses in the symbol table of your S7 program. There you can also assign comments to the symbols. The combination of short symbols and more detailed comments facilitates the efficient creation of programs and well-structured program documentation.

Note

For further information on symbolic programming, refer to the STEP 7 documentation.

5.8.6.1 Entering symbols

Entering several symbols in the symbol table

1. In S7-HiGraph, select **Options > Symbol Table**.
2. In the symbol table, assign symbols to the absolute addresses.

Entering single symbols in the dialog box

You can also define individual symbols in an open graph group without starting the symbol table. To do so:

1. Open the "Current parameter" tab.
2. Position the insert cursor on an entry in the "Current parameter" column.
3. Select **Edit > Edit Symbols**.
4. Now enter the following values:
 - absolute address of the symbol
 - data type of the address
 - name of the symbol.
 - Comment (optional).
5. Confirm your entries with OK. The entry is automatically added to the symbol table.

5.8.6.2 Entering symbolic addresses in the program

Selection list of symbolic addresses

You can open a selection list containing the symbols defined in the symbol table in order to support programming and the assignment of current parameters. You can then click the symbols to copy them directly to your program.

Procedure:

You can view the list of symbols using two different methods:

- To open this list once, select **Insert > Symbol (Ctrl + J)**.
- To open this list automatically, select **View > Show with > Symbol Selection**.

The list opens automatically when you place the cursor in the column "Current parameter" and enter a character.

5.8.6.3 Selecting the symbolic / absolute address format

Available layouts

You can choose from three representations of shared addresses:

Layout	System reaction to user input	Example
Absolute addresses	S7-HiGraph shows all addresses using their absolute address. S7-HiGraph converts symbolic names you enter into absolute addresses.	M1.0
Symbolic names	S7-HiGraph shows all addresses using their symbolic name. S7-HiGraph converts absolute address you enter into a symbolic name.	Motor_on
Shown as entered	S7-HiGraph does not perform a conversion	-

Setting the layout

- To set the "absolute addresses" format, set **View > Show with > Absolute Addresses**.
- To set the "symbolic name" format, set **View > Show with > Symbolic Addresses**.
- Reset these menu commands if you want to prevent a conversion in S7-HiGraph.

5.9 Programming waiting, monitoring and delay times

5.9.1 Programming waiting times

5.9.1.1 Waiting times

Definition

You can specify whether the control system is to hold a state for a minimum period before checking the step transitions.

You can specify the length of the waiting time as a constant or as a dynamic value. By using dynamic values (in the form of a formal parameter or a shared variable of the data type TIME), you can implement different waiting times in the various instances of a state graph.

5.9.1.2 To program waiting times

Basic steps in programming waiting times

Waiting time programming is divided into two steps:

1. Defining the waiting time
2. Make allowances for the waiting time in the transition.

Step 1: Defining the waiting time

1. Select a status.
2. Open the detail view.
3. Select the "Instructions" tab.
4. Select the "Waiting time" instruction type on the left section of the tab.
5. Enter the length of the waiting time on the right section of the tab:
 - Enter constants based on the STEP 7 time constant syntax:
T#<const>
<const>= nD (n Days) nH (n Hours) nM (n Minutes) nS (n Seconds) nMS (n Milliseconds), where n = number
For example:
T#3D4H2M1S44MS
T#2.5H
T#13S750MS
 - In order to define a dynamic value, enter a variable or shared variable declared for this state graph.

Step 2: Make allowances for the waiting time in the transition.

Next, you assign the "Waiting" attribute to those transitions which are to evaluate the waiting time:

1. Select the transition.
2. Select **Edit > Object Properties**
3. Set the "Waiting" attribute (check box).
4. Confirm your entry with OK.

Note

You can stop or interrupt the waiting time by setting the predefined variable WT_Stop.

See also

Overview of cyclic execution of a state (Page 7-2)

5.9.2 Programming monitoring times

5.9.2.1 Monitoring times

Definition

The monitoring time is used to monitor the time a control dwells in a certain state. The physical time measured is the monitoring time. The monitoring time therefore continues to run, even if cyclic execution of the state graph is interrupted (for example, at the change to manual mode.)

Editing monitoring times

If you have specified a monitoring time and the respective state is not left within the specified time, the program responds as follows.

- The default "ST_Expired" variable is set.
- An error message is output to the diagnostics program.

You can specify the length of the monitoring time in a constant or as dynamic values. By using a dynamic value in the form of a formal parameter or shared variable of the data type TIME, you can implement different monitoring times at the various instances of the state graph.

5.9.2.2 To program monitoring times

Procedure:

1. Select a state.
2. Open the detail view.
3. Select the "Instructions" tab.
4. Select the "Monitoring time" instruction type in the left-hand section of the tab.
5. Enter the length of the monitoring time in the right-hand section of the tab:
 - Enter constants based on the STEP 7 time constant syntax:
T#<const>
<const>= nD (n Days) nH (n Hours) nM (n Minutes) nS (n Seconds) nMS (n Milliseconds), where n = number
For example:
T#3D4H2M1S44MS
T#2.5H
T#13S750MS
 - In order to define a dynamic value, enter a variable or shared variable declared for this state graph.

Note

You can stop or interrupt the monitoring time by setting the predefined ST_Stop variable.

See also

Overview of cyclic execution of a state (Page 7-2)

5.9.3 Programming delay times

5.9.3.1 Delay times

Definition

You can use delay times to prevent a reaction of the PLC to short signal pulses.

A transition with delay time does not switch immediately after its condition is satisfied. Conditions must remain constant for the duration of a certain delay time before the transition takes place.

Editing delay times

Delay times always apply to the current cycle. All delay times are reset when a state graph is re-initialized.

Special features of the Any transitions

Special features of the Any transitions are:

- Reaction of S7-HiGraph when a state graph has several Any transitions:
The delay time of an Any transition is reset immediately when an Any transition of the same or higher priority is triggered.
- Delay times at an Any transition are not evaluated in the following situation:
 - the next step of the Any transition is active
and
 - the "Trigger Any transition once only" is set.

Constant or dynamic delay times

You can specify the length of the delay time as a constant or as a dynamic value. By using a dynamic value in the form of a formal parameter or shared variable of the data type TIME, you can implement different delay times at the various instances of the state graph.

V5.3 memory format

Delay times are only known to state graphs in V5.3 memory format. The state graph is automatically assigned memory format V5.3 when you program a delay time. The state graph can not be down converted to V5.2 format.

5.9.3.2 To program delay times

Requirements

The following conditions must be satisfied in order to program delay times for a state graph.

- The state graph is available in V5.3 format.
- The state graph is available in V5.3 format, and the "Convert to V5.3" option is set in the "Memory format" tab (application settings.)

Procedure:

1. Select a transition.
2. Open the detail view.
3. Select the "Instructions" tab.
4. Select the "Delay time" instruction type on the left section of the tab.
5. Enter the length of the delay time on the right section of the tab:
 - Enter constants based on the STEP 7 time constant syntax:
T#<const>
<const>= nD (n Days) nH (n Hours) nM (n Minutes) nS (n Seconds) nMS (n Milliseconds), where n = number
For example:
T#3D4H2M1S44MS
T#2.5H
T#13S750MS
 - In order to define a dynamic value, enter a variable or shared variable declared for this state graph.

See also

Determining the current save format (Page 5-72)

5.10 Programming operating modes

5.10.1 Operating modes

Definition

Operating modes define the method by which a machine or plant operates.

Operating mode	Function
Auto mode	Automatic program execution without operator intervention
Manual mode	Manual program execution. The operator sets each status.

See also

Overview of cyclic execution of a state (Page 7-2)

Default variables (Page 5-9)

5.10.2 Steps in programming operating modes

Transition features

S7-HiGraph supports you in implementing the operating modes.

To do so, set the "Auto" or "Manual" transition attributes in the state graphs. These attributes influence transitions to the next step as follows:

Transition features	Behavior of the control system
Auto	In addition to the transition conditions, the system evaluates a specific default variable (AutomaticMode.) A state transition takes place if <ul style="list-style-type: none">the transition conditions are satisfiedand the AutomaticMode variable has assumed the logical value 1.
Manual	In addition to the transition conditions, the system evaluates a specific default variable (ManualMode.) A state transition takes place if <ul style="list-style-type: none">the transition conditions are satisfiedand the ManualMode variable has assumed the logical value 1.
No attributes assigned	A status transition is executed place if the transition conditions are satisfied.

Assigning transition features

To assign the features:

1. Select a transition and then select **Edit > Object Properties**.
2. On the next dialog box, select the "Auto" or "Manual" features. Ensure that you explicitly assign an operating mode to each transition. There is no default operating mode for transitions.

Configuring the operating modes

You can implement various operating modes in your program by configuring higher-ranking state graphs which control the operating modes and provide data to the AutomaticMode or ManualMode variables.

Refer to the example for configuring operating modes in the "Notes on configuration" section.

5.11 Programming graph groups

5.11.1 Graph groups

Definition

A graph group contains a number of state graphs which can be compiled, downloaded and saved.

Function of graph groups in the S7-HiGraph program

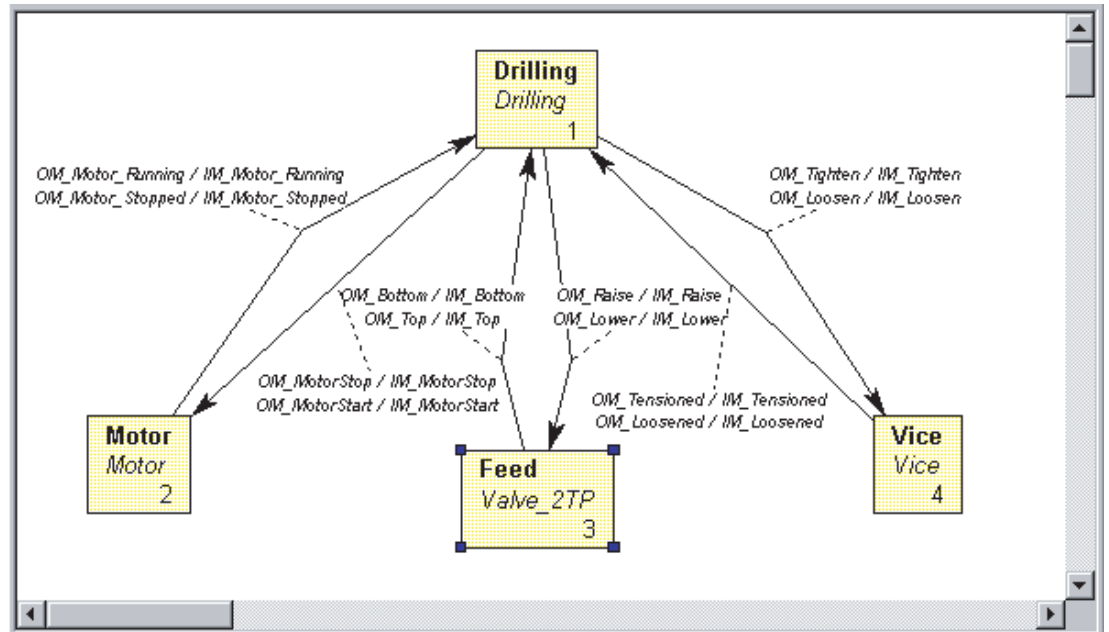
State graphs describe individual functional units of a machine. To describe an overall machine or plant, you coordinate a number of state graphs in a graph group.

Graph groups define the run sequence of the state graphs

A graph group defines a sequential order of calls of state graphs executed cyclically in the program. The call of a state graph is referred to as an instance. Instances are processed in the PLC based on a default run sequence.

Example

The figure below shows a graph group with the instances of several state graphs:



See also

Reusability of state graphs (type / instance concept) (Page 5-1)

5.11.2 Steps in programming graph groups

Step 1: Creating the graph group

1. Select **File > New Graph Group**.

The "New" dialog box opens.

You selected the "Sources" folder of your S7 program and entered the "Graph group" type under "Object type."

2. Enter a file name.

Step 2: Inserting instances

1. Select **Insert > Instance**
2. Select the relevant instance.
3. The instance is shown as a rectangular symbol on the working window.
4. Select **Edit > Object Properties**, then assign the instance a meaningful name in the next dialog box. This will not affect the file name of the state graph.

Note

By double-clicking on an instance, you can conveniently open the corresponding state graphs.

Note

A graph group may only contain instances of state graphs of the S7 program in which the graph group is also located.

Step 3: Changing the run sequence

1. Select **Edit > Run Sequence**
2. The "Run sequence" dialog box indicates the current sequential order. Use the arrow keys to change these.
The run sequence is indicated by a number in the lower right-hand corner of the instances.

Step 4: Assigning current parameters

You now have to assign current parameters to the variables used:

1. Select an instance.
2. Enter the current parameters in the "Current parameters" tab
3. In order to use symbolic names defined in the symbol table as current parameter, select **Insert > Symbol**.

See also

Programming with absolute or symbolic addresses (Page 5-31)

5.12 Programming the exchange of messages between state graphs

5.12.1 Basics of exchanging messages

Definition

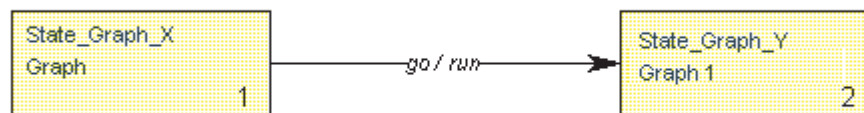
Messages are used for communication between state graphs. One state graph sets a signal that is received by another state graph. This is how state graphs can influence each others runtime.

Message types

Message type	Function
Internal message	Communication between state graphs of a graph group Communication takes place via a bit address in the S7-HiGraph DB.
External message	Communication between state graphs of different graph groups or between S7-HiGraph FCs and other STEP 7 programs. Communication is carried out via a shared bit address which you must provide yourself.

Display of messages

The signal exchange is clearly displayed in the graph group with arrows.



5.12.2 Basic procedure of message programming

Basic procedures for exchanging messages

A message is realized by means of a bit which is set by the sending state graph and evaluated in the receiving state graph.

However, you do not address this bit directly, but you rather use one local variable each in the two state graphs. This leads to a higher flexibility in multiple use (instantiation) of the state graphs.

Steps in programming messages

You always have to perform the following steps to program messages:

1. In the send state graph, declare a variable with the message type "out."
2. In the receive state graph, declare a variable with the message type "in."
3. Next, program an instruction in the sending state graph that sets or resets the signal state of the variable. In the receiving state graph, program a condition which scans the signal state of the variable.
4. Now insert instances of the two state graphs into one or more graph groups.
5. You now only have to specify which variables are to communicate with each other. This is achieved by means of assignments in the "Current parameters" tab.

5.12.3 Declaration of message variables

Procedure:

1. Open the relevant send state graph.
2. Declare an IN_OUT variable of the data type BOOL.
3. Assign the variable the message type "out."
To do so, click the types in the "Message type" column of the variable detail view, and then select the type from the list.
4. Open the relevant receive state graph.
5. Declare an IN_OUT variable of the data type BOOL.
6. Assign this variable the message type "in."

Note

The names of the two variables do not have to match.

System attribute S7_message

When you set a message type you implicitly assign the system attribute S7_message to the variable.

5.12.4 Programming statements for messages

Procedure:

1. In the state graph which is to send the message, program an instruction that assigns a signal state to the declared variable. (for example, S Message_out.)
2. In the state graph that is to receive the message, program an instruction that evaluates the signal state of the variable declared here (for example, U Message_in).
3. In the receiving transition, you can program an action to reset the respective bit once the message is received.

5.12.5 Interconnecting incoming and outgoing messages

Requirements

To interconnect these messages, you first need to add the instances of the send and receive state graphs to a graph group.

Procedure for internal messages

For internal messages, it is sufficient to tell the state graph where the message is to be sent.

1. Select the instance of the send state graph.
2. Open the "Current parameters" tab if necessary.
3. Select the sending variable from this tab.
4. Set the receiving variable as current parameter. Use the following syntax:
Name of the receiving state graph.Name of the message type "IN"

Example: *StatusGraph_Y.Message_in*

Note

You do not have to enter these names manually:

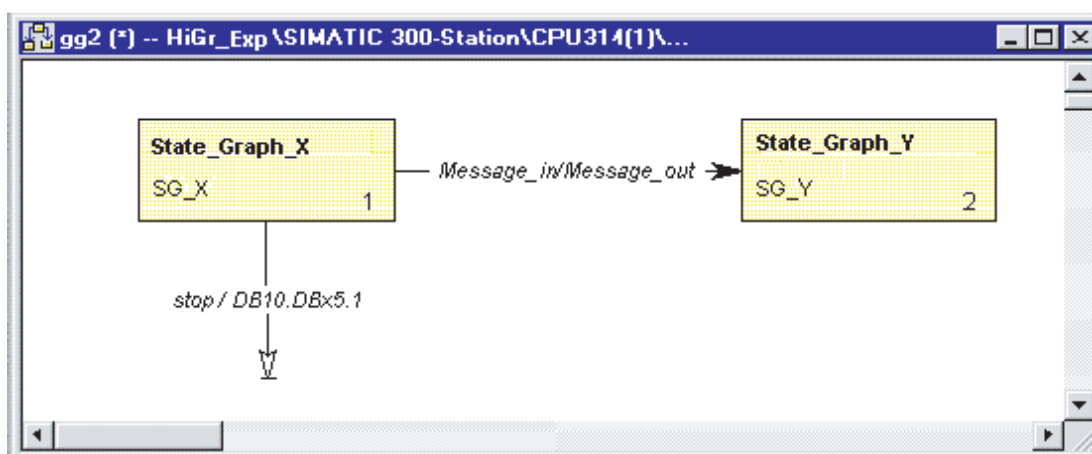
1. In the first step, select **Insert > Symbol** to open a list of state graphs containing incoming messages.
 2. Select a name from this list.
 3. Enter the following information using the keyboard.
 4. Again, select **Insert > Symbol** to open a list of incoming messages contained in the state graph.
 5. Select a name from this list.
-

Procedure for external messages:

1. Select the instance of the send state graph.
2. Select the send parameter from the "Current parameter" tab, then enter a shared bit address as current parameter (for example, DB10.DBx5.1).
3. Next, select the instance of the receiving state graph.
4. Assign the same shared bit address to the receiving parameter.

Result

The figure below shows a state graph sending an internal and an external message:



5.13 Viewing reference data

5.13.1 Overview of available reference data

You can generate and evaluate reference data in order to facilitate debugging and modification of your program. These reference data are used, for example, as

- overview of the entire user program
- basis for debugging and changes
- supplementation of the program documentation

Available reference data

The table below shows the reference data you can output:

Reference data	Contents
Cross-reference	Overview of addresses in the I, O, M, P, T, C and DB memory areas in the program.
Program structure	Call structure and overview of blocks within an S7 program and their dependencies
Assignment list	Shows the assignment for <ul style="list-style-type: none">• inputs, outputs, memory bits• timers and counters The overview of bit addresses in the I / O / M /timer / counter memory areas used in the application program forms a vital basis for troubleshooting or modifications.
Symbols not used	Overview of all symbols defined in the symbol table, but not used in the program.
List of addresses without symbol	Overview of all absolute addresses in the program for which you have not defined any symbols in the symbol table.

Note

For further information on this function, refer to the STEP 7 documentation.

5.13.2 Generating and viewing reference data

Automatic generation and update of reference data

You can generate or update reference data each time you compile a graph group.

1. Select **Options > Graph Group Settings**.
2. Open the "Compile" tab.
3. Set the "Generate reference data" option.

Viewing reference data

1. Select **Options > Reference Data > View**
2. If the reference data are not updated yet, you are prompted to generate or update these data.
3. Select the view.
4. Select **Reference Data > Close** to close the reference data again.

Further information

For further information on this function, refer to the STEP 7 help.

5.13.3 Quick positioning to locations in the program

Locations in the S7 program

You can use reference data to view all locations of an address in the S7 program.

S7-HiGraph outputs a list of locations, including the following information:

- Block in which the address is used
- Block symbol if it exists
- S7-HiGraph-specific information
- Address access mode: Read (R), write (W), read/write (RW), cannot be determined (?).

Display of all address locations in the program

Note

This function can only be executed if current reference data are available.

1. Open the relevant graph group or state graph.
2. Generate the reference data if not available.
3. Select an address in the current parameter or instruction window.
4. Select **Edit > Go To > Location**.
5. A list opens showing the address locations in your program.
6. You can now select a location from the list and use the "Go to" button to jump to the corresponding location in the program.

Display of local address locations

You can also jump to address locations in the active document. You can search for shared and local addresses. Reference data are not required for this operation.

- Select **Edit > Go To > Previous Location**.

5.13.4 S7-HiGraph-specific information in the reference data

S7-HiGraph-specific information

Language-specific information on S7-HiGraph is displayed in the cross-reference list and in the program structure. The abbreviations used there are explained in the following table:

Abbreviation	Description
HiGraph	Language used to generate the block.
Ixxx	Number of the instance in which the address is used.
Txxx	Priority of the transition in which the address is used.
aTxxx	Priority of the Any transition in which the address is used.
rTxxx	Priority of the Return transition in which the address is used.
Sxxx	Number of the state in which the address is used or the number of the source state of the specified transition.
C, C-, I, X, ?, !,	Type of instruction in which the address is used.
Lnxxx	Line within the instructions of a type in which the address is used.
M	Address is used in a property template.
Fp	Address is used in the current parameter assignment. It is entered there as the formal parameter of an instance.
Cp	Address is used in the current parameter assignment. It is entered there as the current parameter for a message.

5.14 Save

What you should know about saving data

In order to apply newly generated state graphs or graph groups or modifications in the PG database, you have to save them.

S7-HiGraph objects are saved in their current state to the "Sources" folder of the S7 program. The system does not perform a syntax check. You can also save objects which still contain errors so that you can continue editing them in a later session.

To save the objects

To save the S7-HiGraph objects:

1. Activate the working window of the object to be saved.
2. Select:
 - **File > Save** to save the object under the same name.
 - **File > Save as** to save the object under a different name or S7 program. Enter the new path or block in the next dialog box.

Note

Note that any changes made to a state graph affect all its instances once you save the state graph.

SIMATIC Manager also lets you save blocks or source files under other projects.

For information on memory requirements, refer to chapter "Memory requirements of the user program".

See also

Memory requirements of the user program (Page 7-6)

5.15 Compilation

What you should know about compiling

The S7-HiGraph compiler checks the syntax of the program, generates a function (FC) and a data block (DB), and saves these to the "Blocks" folder of the S7 program.

In S7-HiGraph, you always compile the full graph groups. Individual state graphs cannot be compiled.

Syntax error

All syntax errors are indicated in the "Application output" window. An executable block will not be created in this case. Warnings, however, do not influence the result of your compilation. An executable block is generated nonetheless.

Procedure:

1. In the first step, select **Options > Settings for State Graphs / State Groups**.
2. In the "Compile" tab, type in the name of the FC and DB and define any other compiler settings.
A DB or FC which already exists in the "Blocks" folder under the specified name will be overwritten during compilation.
3. Select **File > Compile**
The function compiles the graph group in your active graph group window.
If you opened the state graph window, the function checks in which graph group the state graph is instantiated and then compiles this group. If the state graph is instantiated in several graph groups, we recommend that you open and explicitly compile the relevant graph group.
4. The "Application output" window opens after compilation. Check for any errors.
5. Follow the on-screen instructions to eliminate the errors. Double-click the error message to jump to the error location.
6. Use the **Edit > Go To** command to jump to the previous or next error.
7. After you have eliminated all errors, compile the graph group once more.

5.15.1 Compiler settings

You need to make certain settings in the "Compile" tab to control the results of your compilation. Some of these settings are vital for any online delta downloads at a later time.

Compiler settings available

Option	Effect
FC	Name of the FC to be generated, absolute (for example, FC99) or symbolic (for example, GG_Drill)
DB	Name of the DB to be generated, absolute (for example, DB99) or symbolic (for example, DB_GG_Drill)
Restructure data block	The DB is recreated during compilation under consideration of the entered reserved memory ("Reserve memory (words) in the DB" option). Warning: If you set this option, you may not perform an online delta download. Set the CPU to STOP for the duration of one cycle to download the data. To be able to reload ONLINE, you first need to compile your program with the options "Memory reserves (words) in data block" and "Reconstruct data block" and reset both options for all other compilations.
Switch Any transitions only once	This option has the effect that an Any transition is no longer executed if the control system is already in the target state of the Any transition. If the conditions of the Any transition are still satisfied, however, the regular transitions which can be activated are no longer executed because the higher-priority Any transition takes precedence, although it does not make use of it. Please note that this option increases the code volume. We recommend instead that you explicitly execute a reset command (for example, in the transition action) to reset the conditions that triggered the Any transition.
Cyclic actions with RLO=0	This option has the effect that all instructions and set commands are executed in the cyclic actions of a state once more with RLO=0 when the state is left, and thus resets all signals set while dwelling in the state. The process is carried out after the transition actions and before the exit actions of the state are executed.
Include preceding cyclic actions in the entry cycle	When this option is set, preceding cyclic actions of the state (C-) are executed in the entry cycle (that is, the cycle in which a status transition takes place). Otherwise, these actions are ignored in the entry cycle.
Memory reserve (words) in DB:	This option ensures that sufficient memory space is reserved in the DB for additional state graphs and messages. This specification is important if you want to perform an online delta download. CAUTION: This option is only effective if the data block is restructured during compiling. To be able to reload ONLINE, you first need to compile your program with the options "Memory reserves (words) in data block" and "Reconstruct data block" and reset both options for all other compilations.
Generate reference data	The compiler automatically generates reference data when this option is set.

See also

ONLINE delta downloads (Page 5-57)

Downloading the user program (Page 5-56)

5.16 Calling and loading the S7-HiGraph FC

5.16.1 Calling the FC in an S7 Program

Calling the FC

In order to allow execution of the S7-HiGraph program in the AS, it is called in a cyclic block (OB1, for example.)

You program the calling block, for example, using the LAD/STL/FBD editor of the STEP 7 basic package.

Initialization of the state graphs

The call initializes the state graphs in the status groups. The initialization is performed by means of the "INIT_SD" parameter which is contained in each state graph. Configure this parameter so that a "1" is set when the control system is switched on, and a "0" is set in subsequent cycles.

Note

The signal can be generated by means of the OB 1 start info (#OB1_SCAN_1 variable) and saved to a temporary variable in OB1.

5.16.2 Example of an OB 1 with FC call

Example

```
ORGANIZATION_BLOCK "CYCL_EXC"
TITLE =
VERSION : 0.0

VAR_TEMP
OB1_EV_CLASS : BYTE //Bits 0 to 3=1 (coming), bits 4 to 7=1 (event class 1)
OB1_SCAN_1 : BYTE ; //1 (Cold restart scan 1 of OB 1), 3 (Scan 2-n of OB
//1)
OB1_PRIORITY : BYTE ; //1 (Priority 1 is lowest)
OB1_OB_NUMBR : BYTE ; //1 (Organization block 1, OB1)
OB1_RESERVED_1 : BYTE ; //Reserved for system
OB1_RESERVED_2 : BYTE ; //Reserved for system
OB1_PREV_CYCLE : INT ; //Cycle time of previous OB1 scan (ms.)
OB1_MIN_CYCLE : INT ; //Minimum cycle time of OB1 (ms.)
OB1_MAX_CYCLE : INT ; //Maximum cycle time of OB1 (ms.)
OB1_DATE_TIME : DATE_AND_TIME ; //Date and time OB1 started
INIT_SD: BOOL ;
END_VAR
BEGIN
NETWORK
TITLE =
// This example calls a graph group from
//OB 1:
L OB1_SCAN_1; // Example of the generation of the startup bit.
L 1;
==I;
= #INIT_SD; //In the first OB1 cycle the signal = "1"
CALL "Drill"
init_sd:=#INIT_SD
END_ORGANIZATION_BLOCK
```

5.16.3 Requirements for downloading

Requirements

The following conditions must be satisfied in order to download the user program to the AS:

- The program which is to be downloaded is compiled.
- You programmed the call of the S7-HiGraph-FC from a cyclically executed block.
- The programming device is interconnected with the AS.

5.16.4 Downloading the user program

Elements of the user program

The user program may contain the following components:

- S7-HiGraph FC
- S7-HiGraph DB
- S7-HiGraph diagnostics DB
- Calling block (OB, FB or FC)
- Other blocks used in the program
- HiGraphErrEmitterFB (FB20), if format converter diagnostics is required
- HiGraphMsgEmitterFC (FC101), if format converter diagnostics is required
- HiGraphUnivEmitterFC (FC102), if standard diagnostics is required
- Alarm_S (SFC18) and Alarm_SQ (SFC 17), if standard diagnostics is required

Procedure:

Use SIMATIC Manager to download the program.

1. Set the CPU to STOP.
2. In SIMATIC Manager, open the S7 program in which you saved the sources.
3. Select the relevant blocks from the "Blocks" folder:
4. Select **PLC > Download**

5.16.5 Downloading the FC and corresponding DB/diagnostic DB

You can also download the S7-HiGraph FC and the corresponding DBs using the S7-HiGraph application.

Procedure:

1. Open a graph group.
2. Select **PLC > Download**
3. On the "Download" dialog box, specify whether you want to download the DB or the diagnostic DB alongside with the FC.
4. Confirm with OK.

5.16.6 ONLINE delta downloads

What is an "ONLINE delta download"

When certain conditions are satisfied, S7-HiGraph only downloads the minimum data volume to minimize the impact on program runtime.

You do not have to set the CPU to STOP. The controlled process is not affected by this download operation.

How to perform an ONLINE delta download

You can download any deltas in online mode. However, to add new state graphs and messages, the DB must provide sufficient memory resources.

Requirements for ONLINE delta downloads

The following conditions must be satisfied in order to enable delta downloads:

Requirements	Procedure
The DB in the CPU provides sufficient memory resources for additional state graphs and messages.	Before you initially compile the program: <ol style="list-style-type: none">1. Select Options > Settings for Graph Groups / State Graphs.2. Select the "Compile" tab. Allocate sufficient memory resources.
Since its last download, the program was not compiled with the "Restructure data block" option.	<ol style="list-style-type: none">1. Select Options > Settings for Graph Groups / State Graphs.2. Open the "Compile" tab.3. Make sure that the "Restructure data block" option is not set.

See also

Compiler settings (Page 5-53)

5.17 Debugging the program

5.17.1 Monitoring the program status

Useful features of the program status monitoring function

The program status monitoring function allows visualization of program execution in the CPU. The execution of individual states and transitions is visualized on a faceplate, including current information on the instruction tables currently being processed.

This allows debugging of errors that were not indicated by the formal consistency check function when you created the program or by the syntax check during compilation. These errors are, for example:

- Programming errors, for example, invalid monitoring times
- Logical errors in the program structure, that is, the programmed states and transitions do not match the relevant process runtime.

Note

Make allowances for the fact that debugging may delay program execution and thus cause malfunctions or violations of the set cycle time limits.



Warning

Visualization of the program status while the system is in RUN may cause serious damage to persons or equipment if malfunctions or program errors occur!

Always ensure that dangerous states are excluded before you execute this function!

5.17.1.1 Requirements for starting the program status monitoring function

Requirements

The following conditions must be satisfied before you can execute the monitoring functions:

- The PG is connected ONLINE to the CPU.
- The program is compiled.
- The program (comprising FB, DB, and OB) was downloaded to the CPU.
- The CPU is in RUN (read) or RUN-P mode (read and write).
- The S7-HiGraph FC is called from a cyclically executed block (for example, OB1).

5.17.1.2 View in program status

Options of visualizing the program status

You have two options of visualizing the program status:

- Program status in the Graph groups window (status overview)
- Program status in the state graph window

Monitoring options in the various windows:

Program status in the status group window

Provides the status overview:

- shows all instances of the current graph group.
- The current state is indicated in each instance.

Program status in the state graph window

Shows detailed information on the status of a selected instance.

- The active state is highlighted in color.
- The transition which led to this state is marked in color
- The last active state can be identified optionally in accordance with the setting in the "Status" tab, which you can call by selecting **Options > Application Settings**.
- A table showing status information on the transition with the highest priority class, outgoing from the active state. Provided it is being processed, you can also view information on a different instruction table by selecting a different transition.

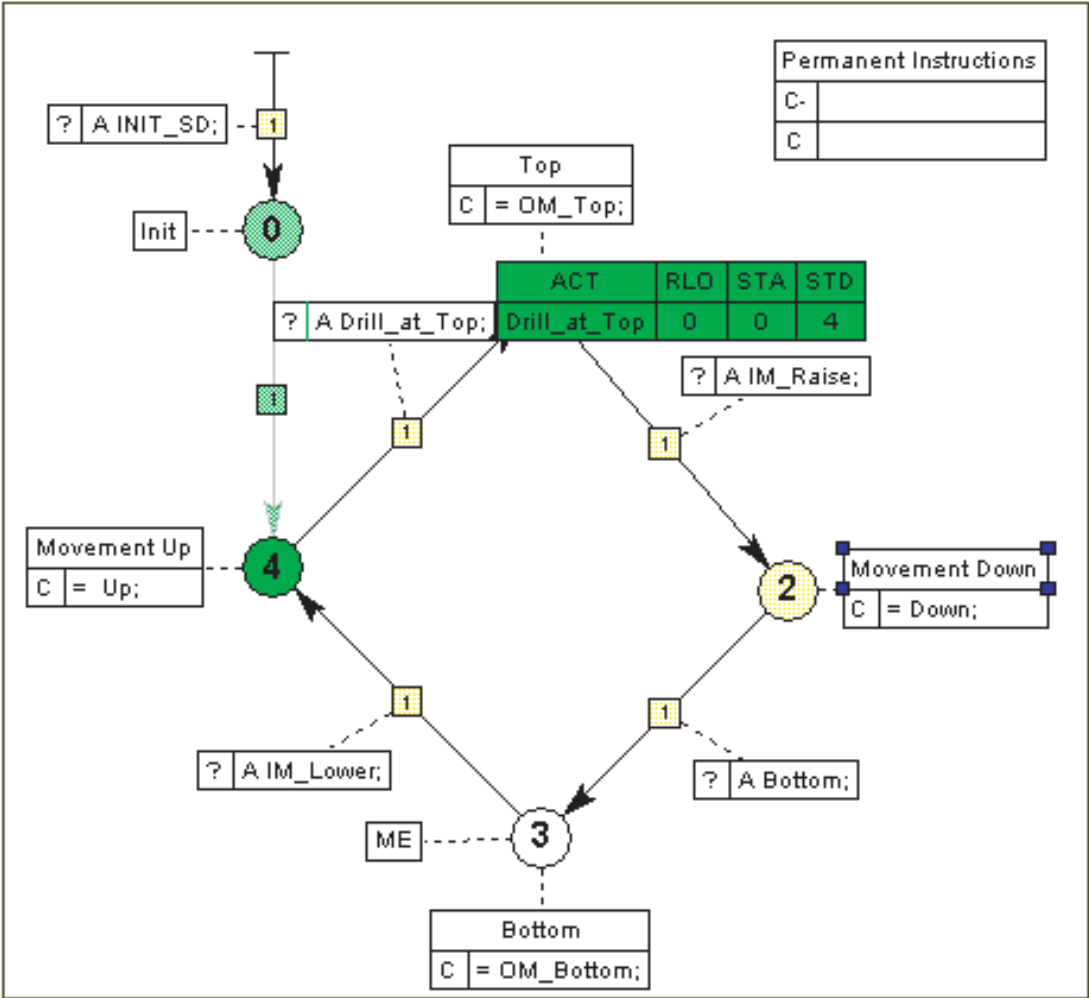
Table of the status information

The status information table contains:

Column	Meaning
RLO	The result of logic operation (RLO).
STA	The status bit.
STD	The standard status displays a timer word, counter word or the contents of ACCU 1, depending on the operation used in an instruction.
OPD	Current parameter which was assigned to the formal parameter of this instruction.

View of the program status

The figure below shows an example of a program status view:



5.17.1.3 Steps in viewing the program status

Procedure:

1. Select **Debug > Monitoring** while the graph group is open.
The status overview shows the status of this graph group and all selected instances.
2. Select an instance for monitoring. You can also select several instances by pressing the CTRL key and selecting instances successively.
3. Select **Edit > Open Object**.
Each selected instance is opened ONLINE. The view shows detailed status information.
4. The table initially outputs status information on the highest-priority transition of the active state. You may then select another instruction table to view its status information.
5. In order to select a different instance for monitoring, first return to the status overview to select the relevant instance, and then select **Edit > Open Object** once again.
6. To close the program status view, disable the **Debug > Monitor** command.



Warning

Visualization of the program status while the system is in RUN may cause serious damage to persons or equipment if malfunctions or program errors occur!

Always ensure that dangerous states are excluded before you execute this function!

5.17.2 Debugging using the STEP 7 debugging functions

5.17.2.1 Check Block Consistency

When do I have to check block consistency?

After you modified the interface of a block, all other blocks calling this block must be adapted. Otherwise, block inconsistency and time stamp conflicts may arise.

By selecting the STEP 7 "Check Block Consistency" function, you can initiate a consistency check of program changes at all S7 blocks in the block folder. This gives you a better idea of the effects of changes to interfaces on other blocks and enhances troubleshooting routines.

The function opens the corresponding editor and guides you to the positions where changes can be made at blocks in which inconsistencies cannot be automatically corrected. There you make the necessary changes. All block inconsistencies are corrected step by step.

Procedure:

1. Select a block folder in SIMATIC Manager.
2. Select **Edit > Check Block Consistency**.

Further information

For further information on this function, refer to the STEP 7 help.

5.17.2.2 Monitoring and controlling variables

How to use the "Controlling and monitoring variables" function

This function allows you to:

- view current values of variables from your user program (monitoring)
- Assign fixed values to the variables for debugging (controlling)

Procedure:

1. Select **Debug > Select and Monitor variable**.
2. Select the relevant instances and their variables in the next dialog box.
3. Click OK.

The STEP 7 variable table opens. It already contains the selected variables. Here you can monitor and control the variables.

Further information

For further information on this function, refer to the STEP 7 help.

5.17.2.3 Evaluating the diagnostic buffer

Using the diagnostics buffer

The following events are written to the diagnostic buffer of the AS during program runtime.

- Incoming/outgoing messages
- Incoming/outgoing errors

How the diagnostics buffer supports you in troubleshooting process errors.

Follow these steps:

The following conditions must be satisfied in order to write events to the diagnostics buffer:

Requirements	Procedure
The HiGraphErrEmitterFB (FB 20) and HiGraphMsgEmitterFC (FC 101) blocks as well as the system function block SFC 52 must exist in the "Blocks" folder of the S7 program.	The blocks are contained in the standard "S7-HiGraph Library." To copy these to the "Blocks" folder: <ol style="list-style-type: none">1. Open SIMATIC Manager.2. Open the S7-HiGraph library.
Diagnostics data for format conversion diagnostics are available for this graph group.	To generate format conversion diagnostics data, proceed as follows: <ol style="list-style-type: none">1. Select Options > Settings for Graph Groups / State Graphs.2. Select the "Diagnostics" tab.3. Set the "Format conversion diagnostics" option.

Further information

For further information on this function, refer to the STEP 7 help.

5.17.2.4 Evaluating CPU messages

Using CPU Messages

The "CPU messages" function also supports you in troubleshooting process errors.

Messages output:

- Status messages
- Error messages

Follow these steps:

1. Open SIMATIC Manager.
2. Select **PLC > CPU Messages**

Note

For further information on this function, refer to the STEP 7 documentation.

5.18 Printing

5.18.1 Creating a program documentation

After you created the program for your automation solution, you can use the integrated S7-HiGraph print function to output all essential data to a printer and thus create a program documentation.

Note

We recommend using a Postscript printer to print out the data.

Printable program elements (S7-HiGraph)

The following program components can be printed out directly in S7-HiGraph:

- The state graphs in graphical or text-based format
- The graph groups in graphical format
- The variable declaration with names, data types, message types and comments for the declared variables
- A list of settings for the graph group and state graphs
- A list of the current parameters, including the names of the formal parameters, the symbolic names and absolute addresses of the current parameters and the comments in the symbol table.

STEP 7 print functions

Further program data can be printed by using the STEP 7 print functions, for example:

- The object tree (project structure)
- The symbol table
- The diagnostic buffer contents
- The reference data
- A list of shared addresses used, including their symbolic names, absolute addresses and comments from the symbol table.

For further information on printing these objects, refer to the STEP 7 documentation.

Optional DOCPRO package

You can use the optional DOCPRO package to create, edit and print standardized circuit diagram manuals. This provides you with a system documentation which is compliant with DIN and ANSI standard.

5.18.2 Steps in printing

Procedure:

1. Open the state graph or the graph group which you want to print out.
2. Select **File > Print**.
3. Select a printer from the "Print" dialog box.
Click "Settings" to define further print settings for your selected printer.
4. Set the print scope in the same dialog box.
5. Define the program elements you want to print in the "Print content" box.
 - Set the "Current View" box to print the current screen content.
 - To print further program elements or other views, set the "According to settings" check box, then click "Settings."
6. Confirm with OK.
The file is output to the default printer.

Global print settings for the application

To output several objects with the same print settings, you can set global print options for your application.

1. Select **Options > Application Settings**.
2. Set the relevant options in the "Print Settings Graph Group" or "Print Settings Status Graph."

Document-specific print settings

You can customize the print settings for documents in V5.2 or higher format and save these alongside with the document. These local settings take priority over global application settings.

1. Select **Options > Settings for Graph Groups/State Graphs**.
2. Set your options in the "Print" tab.

5.18.3 Setting grayscale or color printing

S7-HiGraph documents are printed in grayscale by default. Elements shown with filling on the screen are printed without this filling.

You can customize these default settings, however:

- The document is printed in the screen colors.
- or
- The screen colors are converted into grayscales, that is, the documents are printed in grayscale.

Procedure:

1. Select **Options > Application Settings**.
2. Select the "Colors" tab.
3. Set the relevant option:
 - Color in print
 - Grayscales in print

5.18.4 Setting the page format for printing

Procedure:

1. Select **File > Page Setup**.
2. On the next dialog box, select the "Page format" tab to define the page layout for printing.

By selecting a corresponding page layout with margin (for example, A4 margin), a margin is created on the left side of the page which you can use for punching and filing.
3. Specify whether to use the text boxes only in the current session, or save the settings to the project for use in further sessions.
4. Confirm your entries with OK.
5. You can verify your settings in the page layout view by selecting the **File > Print Preview** command.

Note

The settings made (landscape or portrait) are only valid when you set the "Current View" option for printing.

The specific print settings for state graphs or graph groups apply if you set the "According to Settings" option.

5.18.5 Setting up headers and footers

You can define headers and footers for the printed document.

Follow these steps:

1. Select the project in SIMATIC Manager.
2. Select **File > Page Setup**.
3. Enter the relevant texts in the "Text boxes" tab.
4. Specify whether to use the text boxes only in the current session, or save the settings to the project for use in further sessions.

5.18.6 Opening the print preview

The print preview allows you to verify the settings before you output the document to a printer. You cannot edit the document in this print preview.

Follow these steps:

- Select **File > Print Preview**.

5.18.7 Showing page frames

What you should know about page frames

Page frames show the print dimensions of the pages.

Current print settings applied:

- Paper size
- Portrait/Landscape
- Zoom factor

Procedure:

To adapt the layout of state graphs or graph groups to the format of the future print page while drawing, you can show the page frame and align the objects to the frame.

1. Select **View > Page Frames** to show the page frames.
2. Use the **Options > Align > To Print Page** menu command to center a selected object or a group of objects exactly to the nearest print page.

Note

It is not possible to show page frames after setting the "Zoom to page" function.

5.19 Using data of older S7-HiGraph versions

5.19.1 Converting HiGraph 2.6 / 2.7 programs

Compatibility with HiGraph 2.6 / 2.7

Programs you created in HiGraph V2.6 and V2.7 and older version of STEP 7 can be reused in S7-HiGraph V5.x.

A full migration of programs of older HiGraph versions is not always possible.

Conversion options

Objects you may convert:

- individual graph groups
- STEP 7 projects

5.19.1.1 Converting individual graph groups

Follow these steps:

1. Start S7-HiGraph and open the graph group.
S7-HiGraph recognizes that the graph group originates from a previous program version and requests your confirmation to start conversion.
2. Confirm with "Yes."
The graph group is now automatically converted.
3. During conversion, you are prompted to adapt names, variables or symbols used in your old program name and which are no longer valid in S7-HiGraph V5.

Note

In order to suppress the request to rename variable, select **Options > Application Settings**, then set the corresponding option in the "Migration" tab. Note that this operation may generate faulty state graphs.

4. Now select **Options > Graph Group Settings**, and enter the FC and DB names in the "Compile" tab.
5. Set up the generation of diagnostic data for the FC in the "Diagnostics" tab.
6. Compile the graph group.

5.19.1.2 Converting projects

Follow these steps:

1. Open the project in SIMATIC Manager.
2. You are requested to confirm conversion when you open the project. Confirm with "No."

Note

The function converts only the project if you confirm with "Yes." The various S7-HiGraph files must be converted separately.

3. Select **File > Save as**.
4. Set the "Reorganize" option.
5. Type in the project name and its storage location, then select the "Project" file type.
6. Click "Save."
- Graph groups in the project created in older HiGraph versions are opened automatically and converted to S7-HiGraph V5.
7. Open one of the converted graph groups.
8. Select **Options > Graph Group Settings**. Specify the FC and DB name in the "Compile" tab. Set up the generation of diagnostic data for the FC in the "Diagnostics" tab.
9. Save and compile the graph group.
10. Repeat this operation for all graph groups.

Note

The basic system of STEP 7 V3 or higher is no longer case-sensitive with regard to symbols. If your old program contains case-sensitive symbol names, you will be made aware of this during conversion.

Note

A new block is required for the format converter diagnostics, namely the HiGraphErrEmitterFB function block (default setting FB 20). Copy this block from the example program to the source folder of your project. Renumber a block that already exists in your program under the number FB20 in the symbol table, then recompile the graph group.

5.19.2 Using programs created in S7-HiGraph V4.x / 5.0/V5.2

Save format

You can use various functions to edit a source, depending on the selected save format.

Functions	V4	V5	V5.2	V5.3
Basic functionality	x	x	x	x
Multiple instruction blocks of a type per state or transition		x	x	x
Symbolic / absolute view		x	x	x
Tooltip on current parameters in the status		x	x	x
GoTo error function		x	x	x
Indication of reference data for address locations		x	x	x
Standard diagnostics		x	x	x
Extended diagnostics with actual value calculation and instance-specific settings.			x	x
Session memory		x	x	x
Extended session memory			x	x
Document-specific print settings			x	x
Delay times				x

S7-HiGraph V4.x programs

Options of reusing state graphs or graph groups created in S7-HiGraph V4.x:

- Convert the source code to S7-HiGraph V5.x
- Process the sources in V4 format.

Note

Sources saved in V5.x format cannot be opened in S7-HiGraph V4.x or converted downwards.

S7-HiGraph V5.0/V5.2 programs

Options of reusing state graphs or graph groups created in S7-HiGraph V5.0/V5.2:

- Convert the source code to S7-HiGraph V5.3
- Further use of sources created in the previous format.

5.19.2.1 Defining the save format of new sources

Follow these steps:

1. Select **Options > Application Settings**.
2. Select the "Save format" tab.
3. Set the corresponding options under "New Status Graphs and Graph Groups."

5.19.2.2 Determining the current save format

Follow these steps:

1. Select **Options > Settings for Graph Groups/State Graphs**.
2. The "Save format" tab indicates the format in which the source was saved.

5.19.2.3 Converting sources to V5.3

Follow these steps:

1. Open the source you created in an older version.
2. You are asked to confirm conversion to V5.3 format.
3. Confirm with "Yes."

5.19.2.4 Maintaining sources in their current format

Procedure:

1. Select **Options > Application Settings**.
2. Select the "Save format" tab.
3. Set the "Retain source in set format without confirmation" option.

5.19.2.5 Converting source code from V5.3 down to V5.0/V5.2

Source code saved in V5.3 format can always be converted down to V5.0/V5.2. Status graphs with delay times form the exception. These state graphs can not be down converted.

Procedure:

1. Open the source file.
2. Select **Options > Settings for Graph Groups/State Graphs**.
3. Set the relevant option on the "Save format" tab.
4. Save the source code.

Process error diagnostics

6.1 Standard diagnostics with the help of ProTool / ProAgent

6.1.1 Standard diagnostics functions

Quick troubleshooting of any runtime errors in a machine or system is an important issue. The standardized diagnostic interface integrated in S7-HiGraph is an excellent tool for handling such error situations.

Diagnostic functions

The following diagnostic functions are available:

- Output of messages when the system enters an error, message or timeout state
- Determining the error triggering addresses (criteria analysis based on initial or actual values)
- Monitoring the movements of individual units in the machine/plant and manually controlled troubleshooting

Diagnostic screens on the HMI

Four default diagnostic screens are used on the HMI to visualize diagnostics information.

- The message screen
shows all queued error and status messages.
- The detail screen
shows the result of the criteria analysis of a message. This analysis is used to determine the message triggering event.
- The motion screen
shows the movements which are controlled by the state graphs of a graph group.
- The overview screen
shows the defined units in the plant.

6.1.2 Tools for standard diagnostics

Tools required

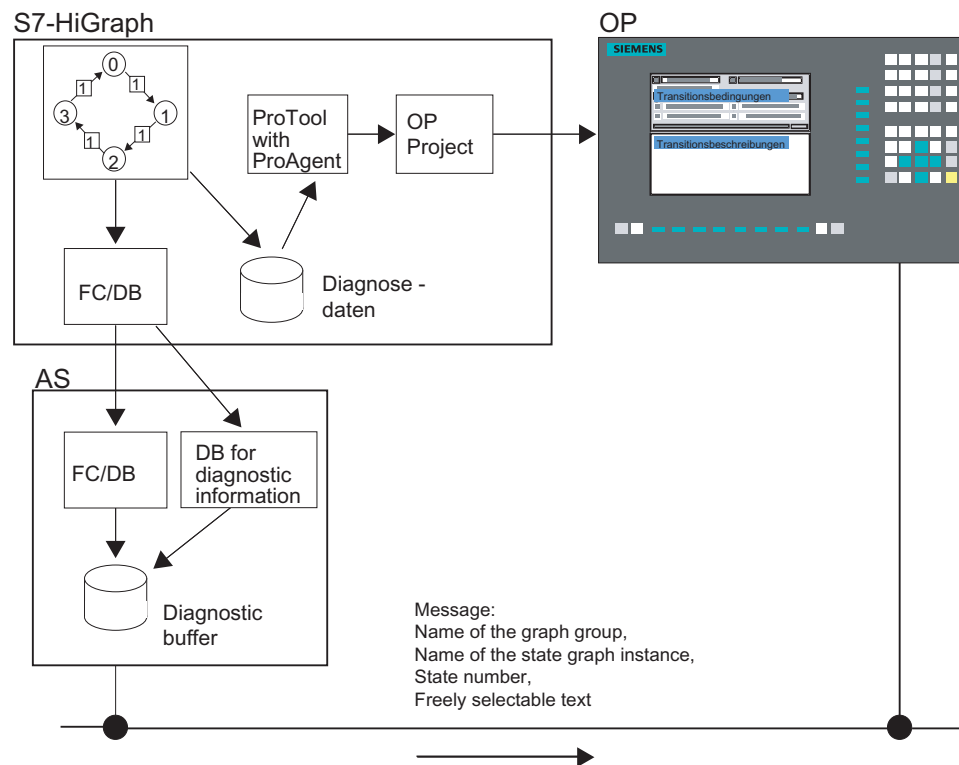
In addition to Hi-Graph, you need the following tools for standard diagnostics:

Tool	Function
ProTool	Visualization software. <ul style="list-style-type: none">• Outputs diagnostics data to the operating and monitoring system (HMI.)• Offers an option of creating diagnostics screens for the output of diagnostics data.
ProAgent	Configuration software. <ul style="list-style-type: none">• Tool used for seamless integration of diagnostics data into the HMI.• Offers diagnostic screen templates which can be configured to output all diagnostics data to the HMI. Involves only slight modification of the screen configuration.

6.1.3 Interaction between S7-HiGraph - AS - HMI

Graphical overview

The figure below demonstrates the interaction between a PG/PC and S7-HiGraph, the AS and the operating and monitoring system (HMI.)



Generating diagnostics information during compilation

The following data are generated when compilation is started:

- a function (FC) and a data block (DB) for the automation system
- a data block (DB) for diagnostic information, for example, lists of participating addresses
- Diagnostics data for the HMI:
 - Message texts
 - Symbols
 - Comments
 - Names of the graph groups
 - Instances
 - States
 - Transitions, etc.

Transfer of the function and DBs to the automation system.

S7-HiGraph downloads the function and data blocks of a graph group to the AS.

Transfer of diagnostics data to the HMI using ProTool / ProAgent

ProAgent prepares the diagnostics data in order to output these to an HMI by means of the visualization software ProTool Project.

Download of diagnostics information while the CPU is in RUN

When an error occurs while the CPU is in RUN, for example, a watchdog interrupt or timeout, the system transfers error messages indicating the message number and error triggering events to the HMI via MPI interface

6.1.4 Requirements of standard diagnostics

Requirements for using standard diagnostics functions

Requirements for using standard diagnostic functions:

- ProTool / ProAgent must be installed.
- The graph group and its state graphs must be available in the save format of S7-HiGraph V5 or higher.
- The diagnostic information is configured in the user program (for example, by assigning the "Error" or "Message" attributes to the states, or by configuring corresponding monitoring times.)
- The "Standard Diagnostics" is set in the "Diagnostics" tab (Settings for State Graphs/Graph Groups)
- The following blocks are available in the "Blocks" folder:
 - HiGraphUnivEmitterFC (FC 102)
 - HiGraphMotionUDT (UDT 2)
 - HiGraphUnitUDT (UDT3)

Note

These blocks are included in the "HiGraph Library." If not available in the "Blocks" folder yet, copy the blocks from the library to your project.

Requirements for using diagnostics functions in motion control

You need to create additional variables in order to be able to use diagnostics functions for motion control:

- To enable motion control, you must declare a variable of the type HiGraphMotionUDT in the STAT declaration section of the relevant state graph.
- To display instances of a motion in the overview screen of ProAgent, declare a variable of the type HiGraphUnitUDT in the STAT declaration section of the relevant state graph.

6.1.5 How to generate standard diagnostics data

Requirements

The graph group is opened.

Procedure:

1. Select **Options > Graph Group Settings**.
2. Set the "Standard Diagnostics" options on the "Diagnostics" tab.
3. On the same tab, specify the symbol or address of the DB to which the diagnostic data are saved.
4. Continue by making further diagnostics settings in the graph group or state graph for the output of data to the screens.
These settings are described under the following topics:
 - Output of messages to the message screen
 - Recording of initial/current values on the detail screen
 - Motion monitoring and control on the motion screen
 - Visualization of units on the overview screen
5. Compile the graph group.
6. Download all blocks of the "Blocks" folder to the AS:
 - FC and DB which contain the S7-HiGraph program
 - S7-HiGraph diagnostics DB
 - HiGraphUnivEmitterFC (FC 102)

Note

Diagnostic data can only be generated for graph groups created in HiGraph V5.0 or higher save format. You can set the save format on the "Save format" tab. To call this tab:

- for new graph groups, select **Options > Application Settings**
 - for existing graph groups, select **Options > Graph Group Settings**
-

6.1.6 Output of messages to the message screen

Two basic message classes

S7-HiGraph can transfer two different message classes to the HMI:

Message class	Description
Error messages	<p>An error message is output</p> <ul style="list-style-type: none">• when the program enters an "Error" state, or• after a monitoring time is exceeded (ST_Expired = 1.) <p>Error messages are not cleared until the program exits the triggering state.</p> <p>In addition, the user can set error message acknowledgement. In this case, the error message is not cleared until the message is acknowledged on the HMI and the function has exited the relevant state.</p>
Status messages	<p>A status message is output when the program enters a "Message" state.</p> <p>These messages indicate the process status. Status messages are mostly used to indicate system states or illegal operations.</p> <p>In contrast to error messages, the status messages are not saved to the archive. Their visualization and acknowledgement features are identical with those of the error messages.</p>

Acknowledgment of message classes

You can set this attribute separately for each message class.

6.1.6.1 Message texts on message screens

Function principle of message texts

Users can edit the default texts of S7-HiGraph which are output to the HMI. In the texts you create, you may use defined keywords which are replaced dynamically with current data when a message is generated. We distinguish the following keywords:

- Keywords in the message text in S7-HiGraph
- Keywords in ProTool / ProAgent

Keywords in the message text in S7-HiGraph

S7-HiGraph V5.2 or higher allows the use of the following keywords in message texts:

Keyword	Meaning	Replaced in the generated message by:
\$\$u\$\$	Instance name	Instance names
\$\$zg\$\$	Name of the state graph	Name of the state graph
\$\$u2\$\$	Name of the graph group	Name of the graph group
\$\$FCa\$\$	FC absolute	Absolute name, for example, FC 1
\$\$FCs\$\$	FC symbolic	Symbolic names, for example, DRILL_UNIT
\$\$FCt\$\$	Symbol comment of the FC	Symbol comment of the FC
\$\$DBa\$\$	DB absolute	Absolute name, for example, DB1
\$\$DBs\$\$	DB symbolic	Symbolic name, for example, DRILL_UNIT_DATA
\$\$DBt\$\$	Symbol comment of the DB	Symbol comment of the DB
\$\$si\$\$	First auxiliary process value: Number of the active state.	@1W%03u@
\$\$sn\$\$	First auxiliary process value: Name of the active state.	Cross-reference to an automatically generated text library, for example, @1W%t#S7HIG_FC1_S_ZG2@
\$\$ti\$\$	Second auxiliary process value: Number of the last triggered transition. The transition number is incremented internally in S7-HiGraph. This number is not visible on the user interface.	@2W%03u@
\$\$tn\$\$	Second auxiliary process value: Name of the last triggered transition.	Cross-reference to a generated text library, for example, @2W%t#S7HIG_FC1_T_ZG2@
\$\$isi\$\$	Third auxiliary process value: Number of the last state before the error or message state	@3W%03u@
\$\$Isn\$\$	Third auxiliary process value: Name of the last state before the error or message state.	@3W%03u@

Note

When S7-HiGraph keywords \$\$\$i\$\$\$ and \$\$ti\$\$\$ are being used, the state and transition numbers are output with three numerals. If this is not acceptable and you prefer a 2- or 3-digit numerical display, you can configure the messages by using keywords for the screen output of auxiliary process values as follows:

@1W%[i]u@, for output of the state number with i digits (i = 1 or 2)

@2W%[i]u@, for output of the transition number with i digits (i = 1 or 2)

Note

Earlier versions of ProTool/ProAgent evaluate the fourth auxiliary process value without error only if all values are output. All auxiliary process values must therefore be output for messages of error states that belong to a combination error. These are:

- \$\$\$i\$\$\$ or \$\$\$n\$\$\$ for the active state
- \$\$ti\$\$\$ or \$\$tn\$\$\$, for the last triggered transition
- \$\$li\$\$\$ or \$\$ln\$\$\$, for the last state before the error or message
- @4W%01b@, for the flag 'Switch to error state after timeout'
- The fourth auxiliary process value must be output last.

Note

Some versions of ProAgent show incorrect or no detail screen for some messages if the message format does not match. The detail screen is displayed correctly when the message contains the following auxiliary process values:

- The current state (keywords \$\$\$i\$\$\$ and \$\$\$n\$\$\$) and
- the last triggered transition (keywords \$\$ti\$\$\$ and \$\$tn\$\$\$) and
- the last state before the error (keywords \$\$li\$\$\$ and \$\$ln\$\$\$)
- the fourth auxiliary value must be output at the end of the message with @4W%01b@.

Keywords in ProTool / ProAgent

The following keywords are also interpreted in ProTool / ProAgent V6.0 or higher:

Keyword	Meaning
@ErrOpSym1@	Symbol of the first faulty address
@ErrOpCom1@	Comment of the first faulty address
@ErrOpAll@	Symbols and comments of all faulty addresses (separated by the pipeline character " ")

6.1.6.2 How to define message texts in S7-HiGraph

Requirements

S7-HiGraph is open.

Procedure:

1. Select **Options > Application Settings**.
2. Type in the relevant text on the "Diagnostics" tab.
These settings are saved when you compile new graph groups.

Example 1

To add new texts to existing graph groups you compiled with set standard diagnostics:

1. Select **Extras > Graph Group Settings**. Select the "Diagnostics" tab.
2. Set the "Compile and update messages" option.
3. Recompile the graph groups.

Example 2

To configure your messages in detail, call the STEP 7 message configuration dialog box:

1. Select the "Diagnostics" tab (Settings Graph Group/Status Graph)
2. Click "Edit".

6.1.6.3 To set the acknowledgment function

Requirements

A graph group or state graph is opened.

Note

After you opened a graph group, define the acknowledgment reactions for the entire graph group.

After you opened a state graph, set a graph-specific acknowledgment attribute. The settings in individual state graphs take higher priority.

Procedure:

1. Select **Options > Settings for Graph Groups/State Graphs**.
2. Select the "Standard Diagnostics" tab
3. Set "with acknowledgment" for error and/or status messages.

6.1.7 Output of initial and current values to the detail screen

Signal states of the addresses

The details screen shows the signal states of the message triggering addresses. We distinguish initial and current values.

Initial values

Initial values represent the signal states triggered at the time an error occurs.

The screen outputs the signal state which was triggered by the message event. The logic can be displayed both in STL or in FBD. A criteria analysis allows you to restrict screen output to signals involved in the error cause.

Current values

In addition to initial values, you can also output the current signal states of the message triggering addresses. Current values represent the signal states assumed after an error event is triggered and at a user-specific time. Current values can only be output as a supplement to initial values.

The function outputs the status pending at the time the user requests the current value on the HMI. The logic can be displayed both in STL or in FBD. A criteria analysis allows you to restrict the screen output to signals which are still involved in the error event.

Note

Current values can not be viewed in previous versions of ProAgent. For more detailed information, refer to the README.WRI file.

6.1.7.1 How to define the screen output of initial and current values

Requirements

A graph group or state graph is opened.

Note

After you opened a graph group, set the initial value acquisition function for the entire graph group.

When you open a state graph, set a graph-specific initial value acquisition attribute. Local settings in individual state graphs take higher priority.

Procedure:

1. Select **Options > Settings for Graph Groups/State Graphs**.
2. Select the "Diagnostics" tab.
3. Set the "with initial value acquisition" option in the "Diagnostics" tab for the relevant message class.
4. Set the "Enable selection of current value on the HMI" option for graph groups if you also want to view the current values.

6.1.8 Motion monitoring and control on the motion screen

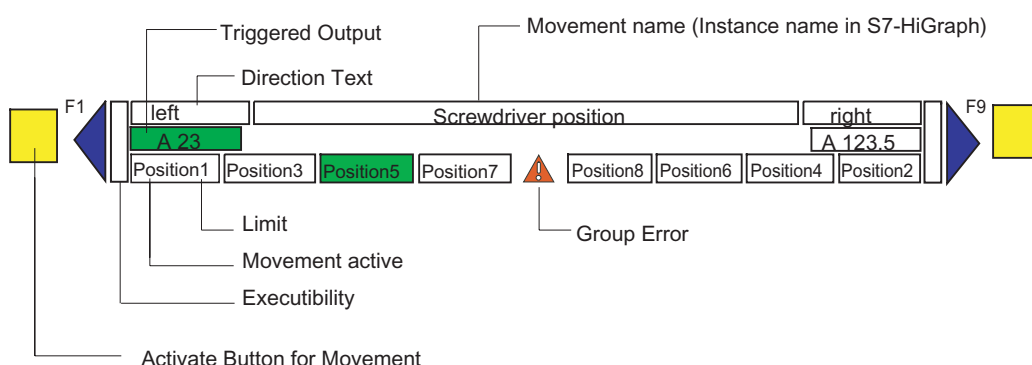
Functions and elements of a motion screen

Functions available on the motion screen:

- monitoring motions of individual units (=graph groups.)
- troubleshooting by means of controlled manual mode, or by moving the units with the help of the buttons next to the display

You can define the elements of a motion screen for graph groups. An instance-specific definition is only required for texts commenting the limit position and direction of motions.

All motions are represented by a "line" on the motion screen. A line contains the following information:



Data for the indicated texts and addresses

The indicated texts and addresses for field animation must be made available. The table below provides an overview of the motion elements and their configuration:

Element	S7-HiGraph configuration
Motion name	= <Instance name>
Controlled output	Symbol or address of the output which is entered in the "Settings for state graphs" ("Standard diagnostics" tab)
Direction text	Text which can be entered in the instance properties. The texts are initialized based on the entries in "Settings for state graphs" ("Standard diagnostics" tab.)
Limit position	Text which can be entered in the instance properties. The texts are initialized based on the entries in "Settings for state graphs" ("Standard diagnostics" tab.) A symbol or an address of the input associated with the relevant limit position is entered in the settings for the state graphs. Assign the corresponding <Movement UDT>.Final_Position[..] bit in the HiGraphMotionUDT for animation of the relevant limit field.
Group error	The group error bit is automatically set by S7-HiGraph when the code is generated
Executability	Executability<No.> bit from HiGraphMotionUDT which has to be set by the user program code. This bit indicates that manual motion control in the indicated direction is enabled. The respective arrow is color filled when this bit signal is "1".

Element	S7-HiGraph configuration
Motion active	Moving-Status<No.> bit from the HiGraphMotionUDT which has to be set by the user program code. This bit indicates a motion in the direction shown. The corresponding field is then highlighted in color.
Button for enabling the motion	<p>The user evaluates the corresponding signals based on the conditions and actions for manual operation, and depending on the HMI version.</p> <p>We distinguish two variants, depending on the HMI configuration:</p> <ul style="list-style-type: none"> Buttons on the side as softkeys: The button signals are declared in the UDT elements "Manual_Operation1" or "Manual_Operation2" Buttons on the side as direct buttons: The "Display_Order" array contains one bit for each line on the motion screen. When the line which is assigned to the instance is output on the motion screen, the line position bit is set in the UDT. This bit must be logically linked to the signal of the corresponding direct key.
Sequence of motions	In S7-HiGraph, the line numbers of motions are based on the execution sequence within a graph group

6.1.8.1 How to define the motion view for a state graph

Requirements

- The UDT 2 (HiGraphMotionUDT) block is saved to your S7 program
- The state graph is opened

Procedure:

1. Select **Options > Settings for Graph Groups/State Graphs** before you start compilation.
2. Select the "Standard Diagnostics" tab.
3. Set the "State graph realizes a movement" option.
4. In the "Movement UDT" input box of the "Standard Diagnostics" tab, enter a variable name for the HiGraphMotionUDT.
S7-HiGraph then generates an entry for the HiGraphMotionUDT in the variable declaration of the state graph.
5. Enter texts to describe both directions of movement.
6. Enter the addresses or texts which are assigned to the motion limit positions. You can use symbols from the symbol table or formal parameters from the variable declaration of the state graphs as addresses.
7. Set the signals of the HiGraphMotionUDT as described in "Monitoring and controlling motions on the motion screen." See also the example of the configuration of texts and addresses on the motion screen.

See also

Motion monitoring and control on the motion screen (Page 6-12)

Example of the configuration of texts and addresses on the motion screen. (Page 6-15)

6.1.8.2 How to define the motion view for an instance

Please note:

Note

You can set up this configuration only for sources of version 5.2 or higher.

These instance-specific settings take priority over those of the graph group.

Procedure:

1. Select an instance.
2. Select **Edit > Object Properties**
The "Instance properties" dialog box appears.
3. Make the relevant settings in the "Standard Diagnostics" tab.

6.1.8.3 Example of the configuration of texts and addresses on the motion screen.

Entries in the "Standard Diagnostics" tab

☐ Instances of the state group visible on display device as a unit
 Variable for units UDT:

☒ State graph realizes a movement
 Variable for movement UDT:

Text direction 1:

Start direction 1:

Text direction 2:

Start direction 2:

End positions:

	Address / Text	
1	Motor_running	
2	Motor_stopped	
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		

Example of instructions for the various elements

Permanent Instructions	
C-	//Limits U Motor_stopped; = Movement.Final_Position[0]; U Motor_running = Movement.Final_Position[1];
	//Enables U Enable_SensorPlus; = Movement.Executability2; UN Movement.Executability1; S Movement.Executability1;
	//Sensor signals U Movement.Manual_Operation1; U Movement.Executability1; = Man_Off; U Movement.Manual_Operation2; U Movement.Executability2; = Man_On;

- "Limits" element
The upstream cyclic actions in the "Limits" block animate the corresponding limit fields in the motion line (that is, the relevant field assumes a green color the limit switch is actuated)
- "Enables" element
The upstream cyclic instructions in the "Enables" block animate the fields for indicating the executability (that is, the corresponding field assumes a blue color when the conditions for manual operation by means of the corresponding direction key are satisfied)
- "Button signals" element
The upstream cyclic instructions in the "Button signals" block apply when display devices with softkeys are used, because the relevant signals in the UDTs are directly set by the HMI station when the direction keys are actuated.
- "Motion active" element
The signals for the animation of the "Motion active" fields can be set directly in the upstream cyclic actions of the corresponding states.

2	Startup	
	C	= MotorOn; = Movement.Moving_Status2;
	ÜZ	T#5000ms

6.1.9 Visualization of units on the overview screen

Visualization of units on the overview screen

The overview screen of ProAgent shows the machine or system units, including certain detailed information on the units. You can visualize graph groups and instances of a state graph as a unit.

6.1.9.1 How to define the view of an instance unit

Requirements

- The UDT 3 (HiGraphUnitUDT) block is saved to your S7 program
- The state graph is opened.

Procedure:

1. Select **Options > State Graph Settings**.
2. Set the "Instances of the state graph visible on the HMI as a unit" option on the "Standard Diagnostics" tab.
3. Next, type in a variable name in the "Variable for unit UDT" input box.
S7-HiGraph then generates an entry for the HiGraphUnitUDT in the variable declaration of the state graph.

6.1.9.2 How to define the view of a graph group unit

Requirements

- The UDT 3 (HiGraphUnitUDT) block is saved to your S7 program
- The graph group is opened.

Procedure:

1. Select **Options > Graph Group Settings**.
2. Set the "Graph groups visible on display device as a unit" option on the "Diagnostics" tab.
3. You can also specify the position of the unit on the overview screen.

6.1.10 Transition handling in ProAgent

Transitions relevant to error diagnostics

The transitions of the following conditions are used for error analysis in ProAgent:

- Transitions triggering an error or message state
- Transitions terminating a time monitored state. This includes all Any transitions.

Visualization of interim results of transitions in ProAgent

When using data or memory bits in these transition conditions containing interim results formed in the 'C-' instructions of the source states or 'C-' instructions of the permanent instructions of the state graphs, these data or memory bits for the fault analysis with ProAgent are replaced by the corresponding network or logical expression with which the intermediate result was formed.

Example of programming in S7-HiGraph and the corresponding visualization in ProAgent:

Visualization of the comparison operation	Example
Permanent instructions:	<div> <div>C</div> <div> U E1.0; U E1.1; =Intermediate result; </div> </div>
Instructions of the transition:	<div> <div>?</div> <div> U E3.3; U Intermediate result; </div> </div>
Visualization in ProAgent	U I3.3; U(U I1.0; U I1.1;);

6.1.11 Programming guidelines for obtaining correct results in a ProAgent analysis

Rules to be observed

You should always follow these programming guidelines in order to obtain correct analysis results in ProAgent:

- Where possible, use '=' assignments Networks terminated with Set and Reset are ignored in the analysis.
- Avoid assignments containing FP and FN operations.
- Always use expressions with bit instructions.
Exception: ProAgent can interpret the following commands:
 - SET
 - CLR
- Networks containing save command are ignored within transitions:
 - =
 - S
 - R
 - EP
 - EN

You should therefore avoid save commands to terminate diagnostics-relevant transitions. A transition terminated with a save command is ignored in the analysis.

- Do not declare assignments within parenthesis structures
- Always terminate diagnostics-relevant statements in C- with a save command, in order to ensure that these are evaluated correctly in the resultant network for ProAgent. (Blocks terminated only with '=' may be used)
- Always assign the results of comparison operations to an auxiliary variable.
- In the following example, the last two instructions are evaluated in the analysis, the previous expression "L Value1;" and "=" is ignored.

Example

Visualization of the comparison operation	Example
Programming in S7-HiGraph	<div style="border: 1px solid black; padding: 5px; display: inline-block;"> <pre> L Value1; L Value2; +I; ? L <I; = Intermediate result; U Intermediate result; U E7.4; </pre> </div>
Visualization in ProAgent	U interim result; U I 7.4

- If you need the result of a comparison operation in a diagnostics-relevant step condition, always terminate the comparison with an assignment to an auxiliary variable. The compare statements can be programmed in the 'C-' statements of the source state, in the permanent 'C-' statements or in '?' instructions of the transition.

Example 1: Comparison operations in the 'C' statements of the source state:

Visualization of the comparison operation	Example
Programming in S7-HiGraph	<div style="border: 1px solid black; padding: 5px; display: inline-block;"> <pre> L Value1; L Value2; <=I; C = M0.0; '? Instructions of the transition: U M0.0; </pre> </div>
Visualization in ProAgent	U M 0.0;

Example 2: Comparison in the '?' statements of the transition

Visualization of the comparison operation	
Programming in S7-HiGraph	<div style="border: 1px solid black; padding: 5px; display: inline-block;"> <pre> L Value1; L Value2; ? <=I; = M0.0; U M0.0; </pre> </div>
Visualization in ProAgent	U M 0.0;

6.2 Diagnostics by means of format converter

Function principle of the format converter

The format converter, too, can be used to generate diagnostics data. However, the diagnostic functions have certain limitations compared to standard diagnostics. For example, only the language scope of S7-HiGraph V2 is available.

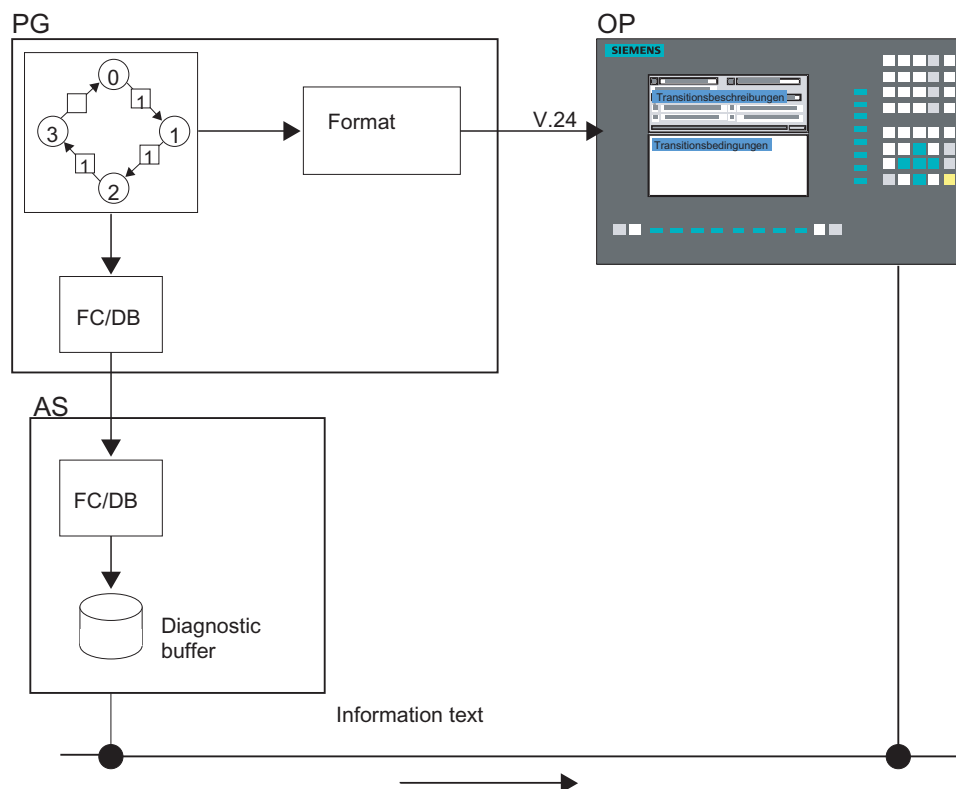
Note

You require the S7-HiGraph format converter for this type of diagnostics. The converter is not included in the standard package. Please order it separately from your SIEMENS representative and install it.

6.2.1 Interaction between S7-HiGraph, the AS and the OP

Graphical overview

The figure below shows the interaction between a programming device and S7-HiGraph, the AS and the HMI.



Generating diagnostics data

After you compiled an S7-HiGraph program, generate the diagnostic data for your OP. These diagnostics data contain all vital data of the state graphs (states and transitions outgoing from these, instructions, variables, etc.)

Diagnostics data for the OP

On the basis of these data, the HMI is able to visualize the error messages in plain text, perform a criteria analysis and mark the error triggering signals.

Transfer of data upon error events

When an error occurs in the control system, an error message is written to the diagnostic buffer of the AS and then transferred in a message frame to the OP. In addition to the date and time, this message frame contains an error message ID, or a status message ID, the number of the graph group (FC no.), the number of the instance, and the state number. Based on this code, the OP is then able to indicate the error or message as described above.

6.2.2 Message events

Message events in format converter diagnostics

The following events can trigger a message at the OP when using format-converter diagnostics.

- **Error states:**
The control system has entered an "Error" state. An error message is output to the OP when the error state is triggered. The error view is closed when the error state is cleared.
- **Violation of monitoring times (timeout):**
An error message is output to the OP if the monitoring time is exceeded ("1" value in the default variable ST_Expired). The error view is closed after the relevant timeout state is cleared.
- **Output of messages:**
A message is output at the HMI when the control system enters a "message" state. This message must be acknowledged by the operator.

Detailed information on message events

When an event occurs, the following detailed information is output to a connected HMI.

- Symbolic name of the event-triggering FC or graph group
- Name of the corresponding instance
- Name of the state

You can easily localize the relevant position in your S7-HiGraph program.

6.2.3 Requirements of format converter diagnostics

Requirements

- The S7-HiGraph format converter must be installed on your device in order to generate the diagnostic data.
- Blocks required in your project for the output of error states and messages
 - HiGraphErrEmitterFB (FB 20)
 - HiGraphMsgEmitterFC (FC 101)
- All necessary entries are made in the symbol table.

Note

If the block numbers in your program are already used, you can also assign other numbers in the symbol table and then recompile the block.

- The default "UsrMsgQuit" variable has the relevant current value. This variable is used to acknowledge messages.

Note

Note that the diagnostics function does not support the new functions in HiGraph 4.0 or later. These are:

- dynamic waiting and monitoring times
 - transition actions (!)
 - preceding cyclic actions (C-) in the state
 - dynamic current values for the variable types INT, DINT, REAL, TIME, DATE, TOD, S5TIME, CHAR
 - data types other than BOOL, WORD or DWORD for variables in the STAT declaration section
 - STL instructions other than those permitted in HiGraph 2.7.
 - shared addresses in instructions
 - instructions distributed to different instruction blocks.
-

6.2.4 How you generate diagnostic data for format converter diagnostics

These data must be prepared separately for visualization on an HMI.

Requirements

- An S7-HiGraph program was created.
- The relevant monitoring times or attributes are assigned.
- The graph groups are opened.

Procedure:

1. Select **Options > Settings for Graph Groups / State Graphs**.
2. Settings:
 - Set the "Format Converter Diagnostics" option on the "Diagnostics" tab if you are using source code in V5 save format.
 - This input is not required for source code in V4 save format.
3. Now compile all the graph groups in your S7 program.
4. Select **File > Convert format**. On the next dialog box, select the installation path of the format converter and the storage location for generated data.

You can select this command in any graph group. This converts all the graph groups of the S7 program.
5. Transfer the generated data to the OP.

User program runtime on the PLC

7.1 Rules for cyclic execution of a state

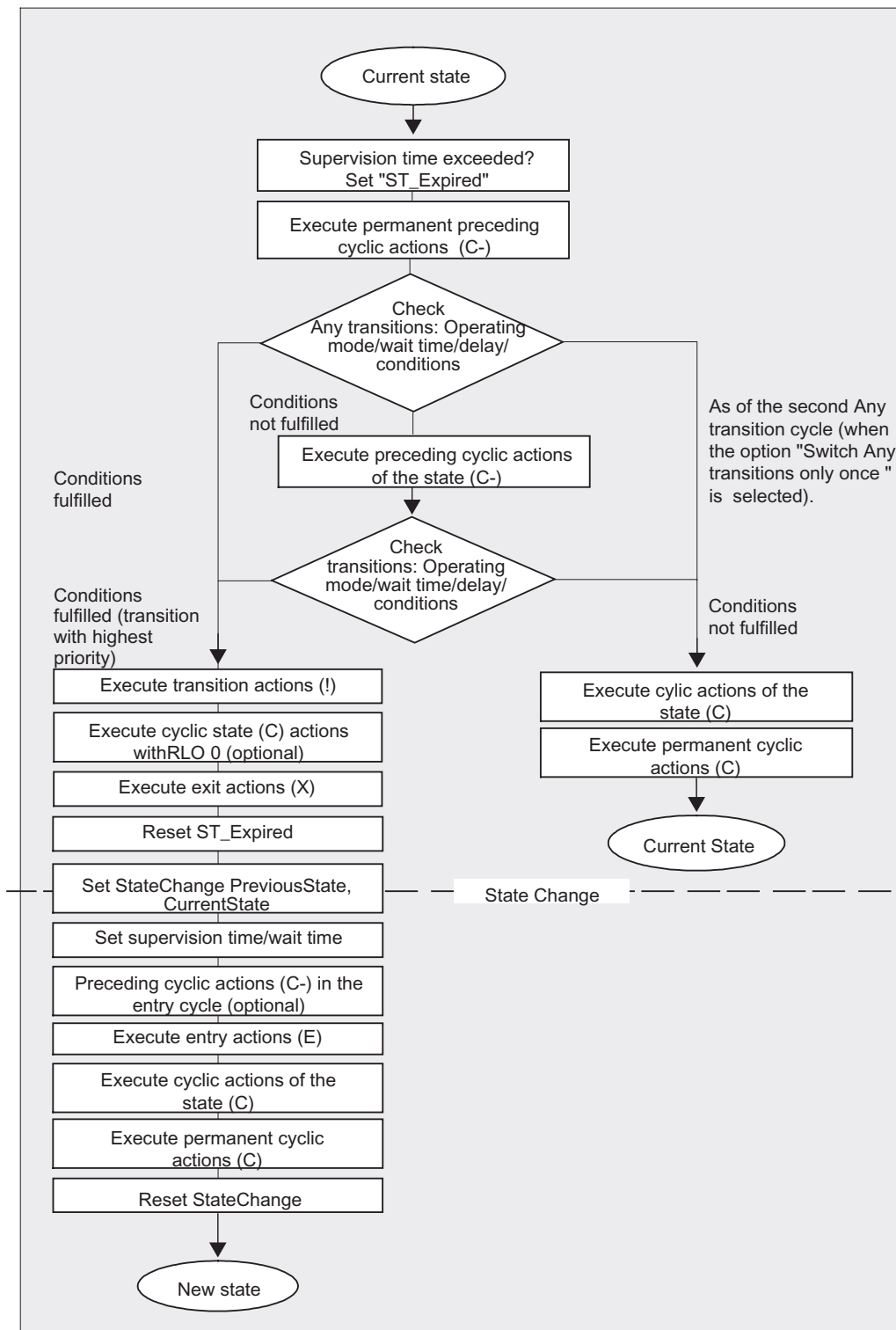
Rules

General rules:

- Only one status transition is made in a state graph cycle.
- Processing of an instruction table always starts with the result of logic operation RLO = 1.
- The status transition is executed when the following conditions are satisfied:
 - All conditions of a transition are satisfied, that is, RLO = 1 in the conditions of this transition,
 - the correct operating mode is set, provided an operating mode was programmed,
 - a waiting time is not set, or the waiting time has expired.

7.2 Overview of cyclic execution of a state

Cyclic execution of a state



7.3 Reaction on startup and hot restart

Initialization of the state graphs

Each graph group contains a formal parameter "INIT_SD."

At the call of a graph group in a cyclic block (OB1, for example), you need to provide data to this formal parameter in order to initialize the graph group.

The various state graphs contain the default "INIT_SD" variable. The signal state in the formal parameter of the graph group is automatically written to the default variables in the state graphs.

Reaction on startup / hot restart

Scheme of the startup / hot restart reaction of a state graph:

DB status	INIT_SD = 0	INIT_SD = 1
DB does not contain any data (DB reloaded or startup of the AS)	<ul style="list-style-type: none">• Reset of the internal incoming messages• Entering the initial state	<ul style="list-style-type: none">• Reset of the internal incoming messages• Entering the initial state
DB with valid content (DB in use by other application, AS restart)	<ul style="list-style-type: none">• Incoming internal messages are not reset automatically.• Normal operation	<ul style="list-style-type: none">• Incoming internal messages are not reset automatically.• Entering the initial state

7.4 Reaction on POWER OFF / ON

Cold restart or hot restart?

Startup modes depending on the CPU used

Reaction of the CPU when power is cycled OFF / ON	Description	Possible at
cold restart	CPU restarts	all CPUs
Hot restart	User program execution resumes at the break point after a hot restart.	This hot restart function is only available on S7-400 CPUs. It must be declared in the CPU parameter record under STEP 7.

Reaction of state graphs to hot restart

The reaction is not critical in an AS with hot restart function, because execution of the program can be resumed at the break point after POWER ON. However, the process image update has to be configured (the DELETE PIU option in HW Config is deactivated.)

The state graphs are executed as follows:

Break point	State graph execution
Break point at any time	<ul style="list-style-type: none"> An interrupted status transition is completed. Status n+1 is set. In the next step, a transition from state n+1 to state 0 is executed as a result of the startup transition.

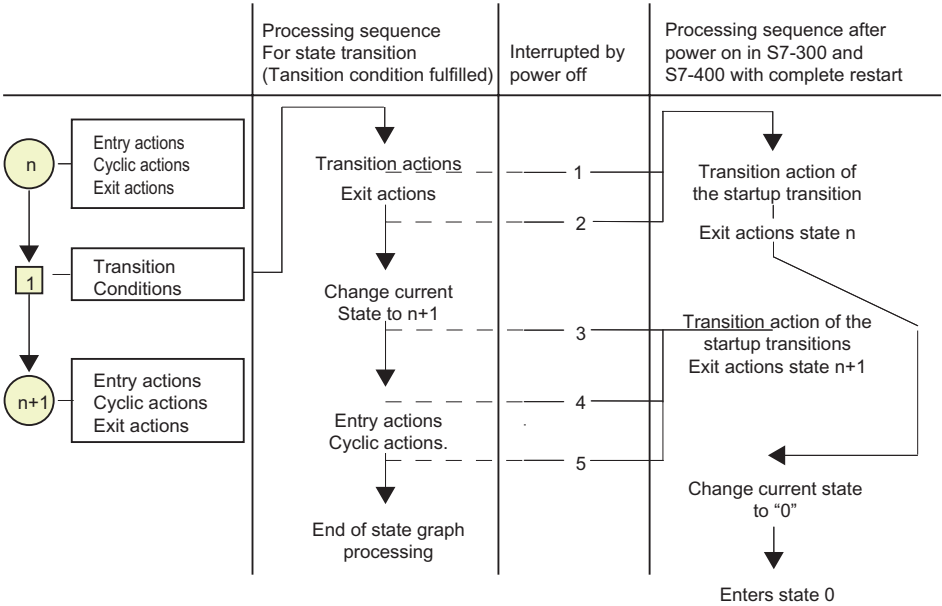
Reaction of state graphs to cold restart

If the AS performs, however, a cold restart after POWER ON, the reaction of the state graph depends on the break point at which execution was interrupted due to POWER OFF.

Break point	State graph execution
A status transition is not executed at the time of "POWER OFF."	<p>This reaction is not critical:</p> <ul style="list-style-type: none"> After POWER ON, the active state a transition had assumed before POWER OFF is reset to "0."
A status transition is executed at the time of "POWER OFF."	<p>Program execution can be interrupted at any point in the command sequence belonging to the transition.</p> <p>General rules:</p> <ul style="list-style-type: none"> Cyclic actions are executed with RLO=0 between the transition actions and the exit actions when option "Execute cyclic actions with RLO=0" is set in the "Compile" tab (settings). All instruction blocks (for example, input, output and cyclic actions, etc.) whose execution was canceled by a power off, remain partially executed. Note the reaction to interruptions due to power off shown at point 3 in the figure below: After POWER ON, a transition to state 0 is executed at the current state saved before POWER OFF. Result: Because the "CurrentState" variable was set to the new state before POWER OFF without execution of the entry action, the action it is not executed after power on.

Detail view: Reaction of state graphs to cold restart

The figure below shows the reaction of a state graph to POWER ON / OFF. It is assumed that the value in the formal parameter "INIT_SD" = "1", that is, the startup transition is fulfilled.



7.5 Reaction of default variables during load operations or at startup

Default variables in a state graph are provided actual values during graph runtime.

Reaction during load operations or at startup

The table shows which values are assigned to the variables in the following borderline cases of operation:

Contents of the variables...	CurrentState	PreviousState	ST_ExpiredPrev	StateChange
After the DB has been downloaded	0	0	0	TRUE
After the FC is reloaded in online mode	unchanged	unchanged	unchanged	unchanged
After hot restart of the AS	0	Current state before startup	Contents of the variable before startup	TRUE
After restart of the AS	0	0	0	TRUE

7.6 Memory requirements of the user program

Memory space for the FC

Element		Required memory space
Graph group	Fixed length of a graph group, excluding waiting or monitoring times	74 bytes
	Graph groups containing at least one waiting, monitoring or delay time.	plus 48 bytes
	GG with delay times	plus 20 bytes
State graphs	Fixed length of a state graph	52 bytes
	State graphs with any number of waiting times	plus 22 bytes
	State graphs with any number of monitoring times	plus 56 bytes
	State graphs with any number of message states	plus 60 bytes
	State graphs with permanent instructions	plus the length of the permanent instruction. Mean value of 4.3 bytes per STL instruction
States	Fixed length of a state	16 bytes
	States with waiting time	plus 8 bytes
	States with monitoring time	plus 8 bytes

Element		Required memory space
	States with entry, exit and cyclic actions	plus the length of the entry, exit and cyclic actions. Mean value of 4.3 bytes per STL instruction Note that cyclic actions are partially executed twice when "Cyclic action with RLO=0" is set.
	Diagnostics-relevant states* at standard diagnostics	plus 12 to 16 bytes
Transitions	Fixed length of a transition	20 bytes
	Transition with waiting time	plus 8 bytes
	Transition with delay time	plus max. 62 bytes
	Transition with MANUAL or AUTO mode	plus 8 bytes
	Transition with actions or conditions	plus 4 bytes, plus the size of the actions or conditions. Mean value of 4.3 bytes per STL instruction
	Diagnostics-relevant transitions* with standard diagnostics	plus 6 to 8 bytes
Any or Return transitions	Fixed length of Any or Return transitions	24 bytes
	Any or Return transitions with waiting time	plus 12 bytes
	Any or Return transitions with delay time	plus max. 62 bytes
	Any or Return transitions with MANUAL or AUTO mode	plus 8 bytes
	Any or Return transitions with actions or conditions	plus 4 bytes, plus the size of the actions or conditions. Mean value of 4.3 bytes per STL instruction

* Diagnostics-relevant states: States assigned a monitoring time, or with F or M attribute. Diagnostics-relevant transitions are used to exit states with monitoring function, or enter states with F or M attribute.

Note

The above listing only applies if all states are numbered successively in ascending order, starting at 0.

Memory space for the DB

The size of the DB consists of the sum of the graph group data and the STEP 7 management information for the DB itself (at present 38 bytes).

The data in the graph group can be calculated as follows:

- One word for an internal identifier
- 12 words per instance (13 if standard diagnostics is used)
- When using delay times:
 - approx. 4 bytes per graph group containing delay times.
 - approx. 4 bytes per delay time
- User-defined variables in the STAT declaration section
- Reserve memory you set in the "Compile" tab (**Options > Settings**).

Memory space for the diagnostic DB (if standard diagnostics is used)

Element		Required memory space
Graph group	Fixed length of a graph group	40 bytes
Instances	Numbered from 1, including all the gaps in the numbering	22 bytes per instance
Transitions (only diagnostics-relevant transitions are saved to the DB)	Fixed length of a transition	4 bytes per instance in which the transition is used
	STL instructions in diagnostics-relevant transitions	Plus 2-6 bytes per instruction
	At initial value detection	Plus 1 Bool per instruction
	At current value detection	Plus 10 bytes (this area is always created in each diagnostics DB)
	Current values selectable on the HMI (GG settings: Diagnostics > Enable selection of current values on the HMI)	plus 12 bytes
States (only diagnostics-relevant states are saved in the DB)	Fixed length of a state	12 bytes + 3 Booleans per instance in which the state is used. The Bool variables lie in 3 arrays.

STL language description

8.1 STL instructions

STL instructions

STL instructions available:

Table 8-1 Bit logic operations

SIMATIC mnemonics	International mnemonics, in case of deviations	Description
)		Nesting closed
=		Assign
CLR		Reset RLO (=0)
EN		Edge negative
EP		Edge positive
NOT		Negate RLO
O		OR
O(OR with nesting open
ON(OR NOT with nesting open
R		Reset
S		Set
SAVE		Save RLO in BIE bit
SET		Set RLO (= 1)
A	A	AND
A(A(AND with nesting open
AN	AN	AND NOT
AN(AN(AND NOT with nesting open
X		EXCLUSIVE OR
X(EXCLUSIVE OR with nesting open
XN		EXCLUSIVE OR NOT
XN(EXCLUSIVE OR NOT with nesting open

Table 8-2 Word logic operations

SIMATIC mnemonics	International mnemonics, in case of deviations	Description
OD		OR double word (32 bit)
OW		OR word (16 bit)
AD	AD	AND double word (32 bit)
AW	AW	AND word (16 bit)
XOD		Exclusive OR double word (32 bit)
XOW		Exclusive OR word (16 bit)

Table 8-3 Time operations

SIMATIC mnemonics	International mnemonics, in case of deviations	Description
EN		Enable timer (free, EN T 0 to T 255)
L		Load current timer as integer to ACCU 1
LC		Load current timer in BCD format to ACCU 1
R		Reset timer
SA	SF	Off-delay timer
SE	SD	On-delay timer
SI	SP	Pulse timer
SS		Retentive on-delay timer
SV	SE	Extended pulse timer

Table 8-4 Counter instructions

SIMATIC mnemonics	International mnemonics, in case of deviations	Description
EN		Enable counter (free, EN C 0 to C 255)
L		Load current counter as integer to ACCU 1
LC		Load current counter value as BCD to ACCU 1
R		Reset counter
S		Set counter preset value
CD	CD	Down count
CU	CU	Up count

Table 8-5 Load and transfer operations

SIMATIC mnemonics	International mnemonics, in case of deviations	Description
L		Load
T		Transfer
TAR	CAR	Exchange address register 1 with address register 2
TAR1	CAR1	Write address register 1 to ACCU 1
TAR2	CAR2	Write address register 2 to ACCU 1

Table 8-6 Comparison operations

SIMATIC mnemonics	International mnemonics, in case of deviations	Description
>I <I >=I <=I ==I <>I		Compare integers (16 bit)
>D <D >=D <=D ==D <>D		Compare integers (32 bit)
>R <R >=R <=R ==R <>R		Compare floating point numbers

Table 8-7 Block calls

SIMATIC mnemonics	International mnemonics, in case of deviations	Description
CALL		Block call At a block call enter the parameters in parentheses. Separate the parameters with comma character. Example of an FC call: CALL FC10 (param1 :=I 0.0, param2 :=I 0.1); Example of an FB call: CALL FB10, DB100 (para1 :=I 0.0, para2 :=I 0.1);

Table 8-8 Fixed point maths

SIMATIC mnemonics	International mnemonics, in case of deviations	Description
*D		Multiply ACCU 1 with 2 as integer (32 bit)
*I		Multiply ACCU 1 with 2 as integer (16 bit)
/D		Divide ACCU 2 by ACCU 1 as integer (32 bit)
/I		Divide ACCU 2 by ACCU 1 as integer (16 bit)
+		Add integer constant (Integer: 8, 16, 32 bit)
+D		Add ACCU 1 and ACCU 2 as integer (32 bit)
+I		Add ACCU 1 and ACCU 2 as integer (16 bit)
-D		Subtract ACCU 1 from ACCU 2 as integer (32 bit)
-I		Subtract ACCU 1 from ACCU 2 as integer (16 bit)
MOD		Division remainder double integer (32 bit)

Table 8-9 Floating point (real number) arithmetic

SIMATIC mnemonics	International mnemonics, in case of deviations	Description
*R		Multiply ACCU 1 with 2 as real number (32 bit, IEEE-FP)
/R		Divide ACCU 2 by ACCU 1 as real number (32 bit, IEEE-FP)
+R		Add ACCU 1 and ACCU 2 as real number (32 bit, IEEE-FP)
-R		Subtract ACCU 1 from ACCU 2 as real number (32 bit, IEEE-FP)
ABS		Absolute value of a real number (32 bit, IEEE-FP)
ACOS		Arc cosine of a real number (32 bit, IEEE-FP)
ASIN		Arc sine of a real number (32 bit, IEEE-FP)
ATAN		Arc tangent of a real number (32 bit, IEEE-FP)
COS		Cosine of a real number (32 bit, IEEE-FP)
EXP		Exponential of a real value (32 bit, IEEE-FP)
LN		Natural logarithm of a real number (32 bit, IEEE-FP)
SIN		Sine of a real number (32 bit, IEEE-FP)
SQR		Square of a real number (32 bit, IEEE-FP)
SQRT		Square root of a real number (32 bit, IEEE-FP)
TAN		Tangent of a real number (32 bit, IEEE-FP)

Table 8-10 Rotate and shift instructions

SIMATIC mnemonics	International mnemonics, in case of deviations	Description
RLD		Rotate left double word (32 bit)
RLDA		Rotate ACCU 1 left via CC1 (32 bit)
RRD		Rotate right double word (32 bit)
RRDA		Rotate ACCU 1 right via CC1 (32 bit)
SLD		Shift left double word (32 bit)
SLW		Shift left word (16 bit)
SRD		Shift right double word (32 bit)
SRW		Shift right word (16 bit)
SSD		Shift with sign integer (32 bit)
SSI		Shift with sign integer (16 bit)

Table 8-11 Accumulator operation instructions

SIMATIC mnemonics	International mnemonics, in case of deviations	Description
DEC		Decrement ACCU 1
INC		Increment ACCU 1
POP		CPU with two accumulators
POP		CPU with four accumulators
PUSH		CPU with two accumulators
PUSH		CPU with four accumulators
TAD	CAD	Change sequence of bytes in ACCU 1 (32 bit)
TAK		Exchange ACCU 1 with ACCU 2
TAW	CAW	Change sequence of bytes in ACCU 1 (16 bit)

Table 8-12 Conversion instructions

SIMATIC mnemonics	International mnemonics, in case of deviations	Description
BTD		BCD to integer (32 bit)
BTI		BCD to integer (16 bit)
DTB		Integer (32 bit) to BCD
DTR		Double integer (32 bit) to real number (32 bit; IEEE-FP)
INVD		1-complement integer (32 bit)
INVI		1-complement integer (16 bit)
ITB		Integer (16 bit) to BCD
ITD		Integer (16 bit) to double integer (32 bit)
NEGD		2-complement integer (32 bit)
NEGI		2-complement integer (16 bit)
NEGR		Change sign of real number
RND		Round to integer
RND-		Round to next lower integer
RND+		Round to next higher integer
TRUNC		Truncate

Table 8-13 STL instructions, sorted by mnemonics

SIMATIC mnemonics	International mnemonic (in case of deviations)	Description
)		Nesting closed
*D		Multiply ACCU 1 with 2 as integer (32 bit)
*I		Multiply ACCU 1 with 2 as integer (16 bit)
*R		Multiply ACCU 1 with 2 as real number (32 bit, IEEE-FP)
/D		Divide ACCU 2 by ACCU 1 as integer (32 bit)
/I		Divide ACCU 2 by ACCU 1 as integer (16 bit)
/R		Divide ACCU 2 by ACCU 1 as real number (32 bit, IEEE-FP)
+		Add integer constant (Integer: 8, 16, 32 bit)
+D		Add ACCU 1 and ACCU 2 as integer (32 bit)
+I		Add ACCU 1 and ACCU 2 as integer (16 bit)
+R		Add ACCU 1 and ACCU 2 as real number (32 bit, IEEE-FP)
-D		Subtract ACCU 1 from ACCU 2 as integer (32 bit)
-I		Subtract ACCU 1 from ACCU 2 as integer (16 bit)
-R		Subtract ACCU 1 from ACCU 2 as real number (32 bit, IEEE-FP)
=		Assign
==D		Compare integers (32 bit)
==I		Compare integers (16 bit)
==R		Compare floating point numbers
ABS		Absolute value of a real number (32 bit, IEEE-FP)
ACOS		Arc cosine of a real number (32 bit, IEEE-FP)
ASIN		Arc sine of a real number (32 bit, IEEE-FP)

SIMATIC mnemonics	International mnemonic (in case of deviations)	Description
ATAN		Arc tangent of a real number (32 bit, IEEE-FP)
BTD		BCD to integer (32 bit)
BTI		BCD to integer (16 bit)
CALL		At a block call from a HiGraph program, enter the parameters in parentheses. The individual parameters are separated by a comma. Example call FC: CALL FC10 (param1 :=I 0.0, param2 :=I 0.1); Example call FB: CALL FB10, DB100 (para1 :=I 0.0, para2 :=I 0.1);
CLR		Reset RLO (=0)
COS		Cosine of a real number (32 bit, IEEE-FP)
DEC		Decrement ACCU 1
DTB		Integer (32 bit) to BCD
DTR		Double integer (32 bit) to real number (32 bit; IEEE-FP)
EXP		Exponential of a real value (32 bit, IEEE-FP)
EN		Edge negative
EP		Edge positive
EN		Enable counter (free, EN C 0 to C 255)
EN		Enable timer (free, EN T 0 to T 255)
INC		Increment ACCU 1
INVD		1-complement integer (32 bit)
INVI		1-complement integer (16 bit)
ITB		Integer (16 bit) to BCD
ITD		Integer (16 bit) to double integer (32 bit)
L		Load
L		Load current timer as integer to ACCU 1
L		Load current counter as integer to ACCU 1
LC		Load current counter value as BCD to ACCU 1
LN		Natural logarithm of a real number (32 bit, IEEE-FP)
MOD		Division remainder double integer (32 bit)
NEGD		2-complement integer (32 bit)
NEGI		2-complement integer (16 bit)
NEGR		Change sign of real number
NOT		Negate RLO
O		OR
O(OR with nesting open
OD		OR double word (32 bit)
ON		OR NOT

SIMATIC mnemonics	International mnemonic (in case of deviations)	Description
ON(OR NOT with nesting open
OW		OR word (16 bit)
POP		CPU with two accumulators
POP		CPU with four accumulators
PUSH		CPU with four accumulators
R		Reset
R		Reset counter
R		Reset timer
RLD		Rotate left double word (32 bit)
RLDA		Rotate ACCU 1 left via CC1 (32 bit)
RND		Round to integer
RND-		Round to next lower integer
RND+		Round to next higher integer
RRD		Rotate right double word (32 bit)
RRDA		Rotate ACCU 1 right via CC1 (32 bit)
S		Set
S		Set counter preset value
SA	SF	Off-delay timer
SAVE		Save RLO in BIE bit
SE	SD	On-delay timer
SET		Set RLO (= 1)
SI	SP	Pulse timer
SIN		Sine of a real number (32 bit, IEEE-FP)
SLD		Shift left double word (32 bit)
SLW		Shift left word (16 bit)
SQR		Square of a real number (32 bit, IEEE-FP)
SQRT		Square root of a real number (32 bit, IEEE-FP)
SRD		Shift right double word (32 bit)
SRW		Shift right word (16 bit)
SS		Retentive on-delay timer
SSD		Shift with sign integer (32 bit)
SSI		Shift with sign integer (16 bit)
SV	SE	Extended pulse timer
T		Transfer
TAD	CAD	Change sequence of bytes in ACCU 1 (32 bit)
TAK		Exchange ACCU 1 with ACCU 2
TAN		Tangent of a real number (32 bit, IEEE-FP)
TAR	CAR	Exchange address register 1 with address register 2
TAR1	CAR1	Write address register 1 to ACCU 1
TAR2	CAR2	Write address register 2 to ACCU 1
TAW	CAW	Change sequence of bytes in ACCU 1 (16 bit)

SIMATIC mnemonics	International mnemonic (in case of deviations)	Description
TRUNC		Truncate
A	A	AND
A(A(AND with nesting open
AD	AD	AND double word (32 bit)
AN	AN	AND NOT
AN(AN(AND NOT with nesting open
AW	AW	AND word (16 bit)
X		EXCLUSIVE OR
X(EXCLUSIVE OR with nesting open
XN		EXCLUSIVE OR NOT
XN(EXCLUSIVE OR NOT with nesting open
XOD		Exclusive OR double word (32 bit)
XOW		Exclusive OR word (16 bit)
CD	CD	Down count
CU	CU	Up count

See also

Steps in entering STL instructions (Page 5-30)

8.2 Valid data types

Table of permitted data types

Table 8-14 Valid data types

Type / description	Size in bits	Format options	Range and numerical representation (lowest to highest value)	Example
BOOL (bits)	1	Boolean text	TRUE/FALSE	TRUE
BYTE (bytes)	8	Hexadecimal number	B#16#0 to B#16#FF	L B#16#10 L byte#16#10
WORD (Word)	16	Dual number Hex number BCD Unsigned decimal	#0 to 2#1111_1111_1111_1111 W#16#0 to W#16#FFFF C#0 to C#999 B#(0,0) to B#(255,255)	L 2#0001_0000_0000_0000 L W#16#1000 L word#16#1000 L C#998 L B#(10,20) L byte#(10,20)
DWORD (double word)	32	Dual number Hex number Decimal number unsigned	#0 to 2#1111_1111_1111_1111_1111_1111_1111_1111 DW#16#0000_0000 to DW#16#FFFF_FFFF B#(0,0,0,0) to B#(255,255,255,255)	#1000_0001_0001_1000_1011_1011_0111_1111 L DW#16#00A2_1234 L dword#16#00A2_1234 L B#(1, 14, 100, 120) L byte#(1,14,100,120)
INT (integer)	16	Decimal number, signed	-32768 to 32767	L 1
DINT (double integer, 32 bits)	32	Decimal number, signed	L#-2147483648 to L#2147483647	L L#1
REAL (floating-point number)	32	IEEE real number	Hi limit: $\pm 3.402823e+38$ Lo limit: $\pm 1.175495e-38$	L 1.234567e+13
S5TIME (SIMATIC time)	16	S5 time in steps of 10 ms (default value)	S5T#0H_0M_0S_10MS to S5T#2H_46M_30S_0MS and S5T#0H_0M_0S_0MS	L S5T#0H_1M_0S_0MS L S5TIME#1H_1M_0S_0MS
TIME (IEC time)	32	IEC time in steps of 1ms, integer with preceding sign	T#-24D_20H_31M_23S_648MS to T#24D_20H_31M_23S_647MS	L T#0D_1H_1M_0S_0MS L TIME#0D_1H_1M_0S_0MS
DATE (IEC date)	16	IEC date in steps of 1 day	D#1990-1-1 to D#2168-12-31	L D#1994-3-15 L DATE#1994-3-15
TIME_OF_DAY (time of day)	32	Time of day in steps of 1 ms	TOD#0:0:0.0 to TOD #23:59:59.999	L TOD#1:10:3.3 L TIME_OF_DAY#1:10:3.3
DATE_AND_TIME	64	Date and time of day	DT#1990-1-1-0:0:0.0 DT#2089-12-31-23:59:59.999	DT#1994-3-15:1:10:3.3 DATE_AND_TIME#1994-3-

Type / description	Size in bits	Format options	Range and numerical representation (lowest to highest value)	Example
(date and time of day)				15-1:10:3.3
CHAR (character)	8	ASCII characters	'A', 'B', etc.	'E'
String		ASCII string	STRING[n+2] n specifies the string length. Maximum length: 254 characters	'AB'
Pointer	48	Hexadecimal number		P#M50.0
Counter	16	Binary number in a word	T n n = CPU-specific	T 5
Timer	16	Binary number in a word	Tn n = CPU-specific	T 4

Example configurations

9.1 "Drill unit" configuration, example

9.1.1 Learning objectives

The "Drill machine" example for getting started was introduced in the programming with S7-HiGraph section. In real life you will have to solve far more complex automation tasks. This is why we shall introduce you to using S7-HiGraph for controlling entire systems in this documentation.

Learning objectives

We have configured a transfer line in which a processing station is used for controlling the already known drilling machine. Various additional functions are necessary in order to integrate the drill operation into the overall control of the transfer line. We have therefore extended the functionality of the drill unit compared to the drill machine as follows:

- A state graph for controlling the cooling media supply
- Operational enable signals
- Operating modes (Auto, Step, Manual / Setup)
- Monitoring functions

Basics required

Knowledge required in the following areas for comprehension of the sample project:

- S7-HiGraph
- SIMATIC
- Technical knowledge

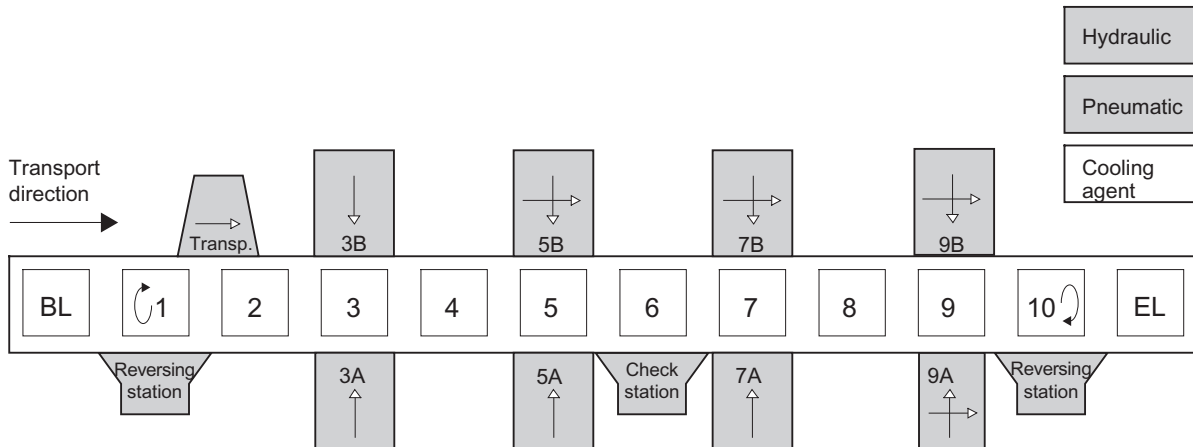
Storage location of the sample project

The "Drill unit" sample project ZEn03_02_HiGraph_DrilUnit is included in your software package. To configure the monitoring functions in the example, we have used format converter diagnostics function. The last part of the chapter provides configuration notes on using the standard diagnostics.

9.1.2 "Transfer line" automation task

"Transfer line" automation task

The task consists of the configuration of a transfer line with the following layout:



The stations 1 to 10 are assigned to processing units (3A, 3B, 5A, etc.), reversing stations (1, 10) or inspection stations (6). The line starts at the delivery point DEP, and ends at the discharge point DIP. The media supply (coolants, hydraulic and pneumatic system) is controlled in higher-ranking operations. One of the processing units used by the drill unit.

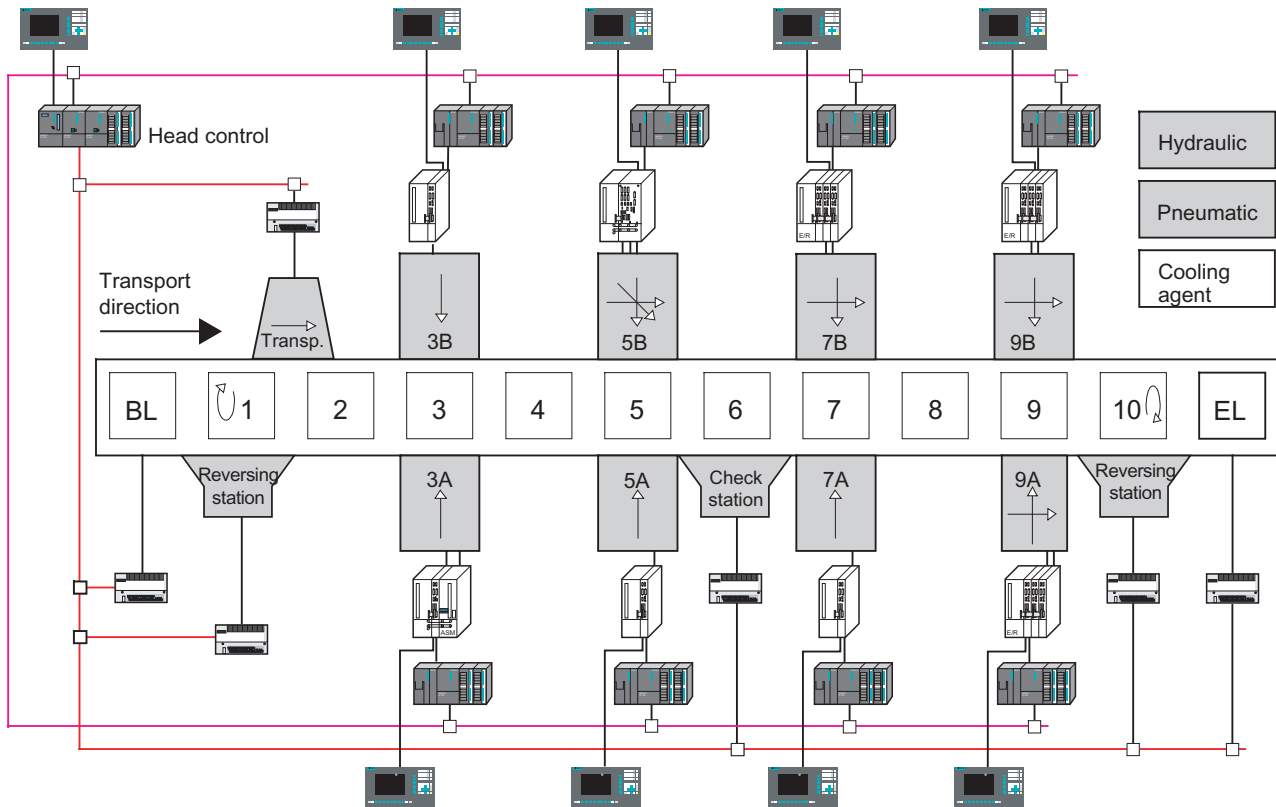
Function principle of the transfer line

A transfer line is characterized by the static, that is, simultaneous transportation of all parts. Each transport operation moves the parts forward to the next processing section.

9.1.3 Expansion with control and operating devices

Control and operating devices

The transfer line components defined in the layout are assigned the control and operating devices.



Each processing unit is usually assigned a control system with operator panel. Each unit is equipped with 1 to 4 controlled drives (axes / spindles), for which we require single-axis modules or NCs (SINUMERIK 810D/840D, for example.)

Master PLC

The master controls transportation and thus the coordination of the processing units. The delivery / discharge positions, reversing stations and the media supply system, for example, the hydraulic, pneumatic and coolant supply systems, are in many cases assigned to the master PLC, because it is not considered cost-effective in most applications to integrate an autarkic PLC.

Networking

All PLCs are networked via PROFIBUS DP. The distributed I/O is also interconnected to the master PLC via PROFIBUS DP.

The master PLC is interconnected with the factory mainframe for the acquisition of operational data (not shown on the layout).

9.1.4 Determining the function units to be controlled

After we have assigned the technological components to the control devices, we can define which functions to control with the help of these control devices.

Function units controlled by the master PLC

The table shows an example of function units controlled by the master PLC:

Functional unit	Function
Transport	The transport function consists of the "Up / Down" and "Advance / Retract" motions. The transport function is used to coordinate the entire transfer line, because parts are not processed unless the transport motion is completed and all parts are clamped. The next advance step is only executed when all units have completed processing.
Lubrication	Usually, the transfer line requires several lubricating circuits which are controlled centrally. These operations require the implementation of pumps, valves, filling level and pressure control systems.
Delivery / discharge	The master PLC controls functions for delivery, distribution, type control and safety doors at the delivery / discharge units
Auxiliary aggregates	The auxiliary aggregates include the media: hydraulic, pneumatic and coolants and the chip transport. The various media require a system consisting of pumps, valves, filling level control systems, filter monitoring systems, etc.
Reversing stations	In our example, the master PLC also controls the reversing stations. These stations require various tools, for example, for the "Clamp / Release" or "Up / Down" functions, reversing units, or grippers.

Function units assigned to the process control stations

The table below shows an example of function units controlled with the help of the process control stations:

Functional unit	Function
Processing	Basic processing elements (for drill operations, for example) are a drill spindle (drill drive), a slide block (for moving the spindle plus drill) and the coolants.
Clamping station	Each unit requires a clamping station which is used to secure the parts during processing. Up to four clamping cylinders may be used. After processing, the chips must be washed off in order to allow proper clamping of the next part.
Media	Central hydraulic, oil, air and coolant media supply systems of a transfer line require in particular only corresponding pressure and filling level monitoring functions.
Safety doors	The safety doors of a processing unit must be locked and unlocked. This also requires monitoring functions.

9.1.5 Assignment of function units and state graphs

The tables below show the state graphs required for the function units listed in the previous section. State graphs of a higher priority class were added for coordination tasks.

State graphs for the master PLC

Function group	Function	State graphs for
General		Operation enables, operating modes
Transport	Raise/Lower, Advance/Reverse	Coordination Raise/Lower, Advance/Reverse
Lubrication	Pumps, valves, filling level control, pressure control	Coordination, pumps, valves, filling level control, pressure control
Delivery / discharge	Delivery, distribution, type control, safety doors	Coordination of delivery feed, distribution, type control, safety doors
Auxiliary aggregates such as hydraulic units, pneumatic units, coolants, chip transport	Pumps, valves, filling level control, filter monitoring	Startup coordination pumps, valves, filling level control, filter monitoring
Reversing stations	Clamp / Release, Raise / Lower, reversing units, grippers, etc.	Coordination Clamp / Release, Raise / Lower reversing units, grippers, etc.

State graphs for a processing unit

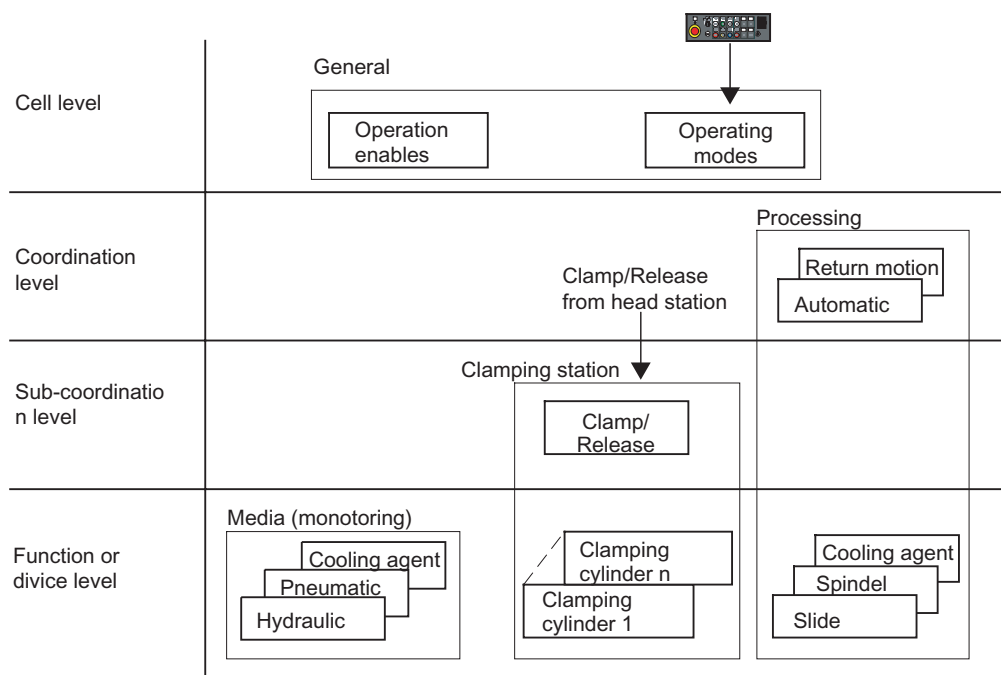
Function group	Function	State graphs for
General		Operation enables, operating modes
Processing	Spindle, slide block, coolant	Coordination Auto mode, coordination return flow spindle, slide block, coolant
Clamping station	Clamp / Release, Flushing	Coordination Clamp / Release, Flushing
Media	hydraulic system, pneumatic system, coolants	Monitoring of hydraulic system, pneumatic system, coolants
Safety doors	Locking / unlocking safety doors	Locking / unlocking safety doors

9.1.6 Creation of graph group

Layer structure of state graphs

State graphs can be organized by layers with the following meaning:

- Cell layer
Used to group essential functions (operation enables and operating modes.)
- Coordination layer
Contains the coordination functions, for example, for auto mode, retraction after process interrupts and similar.
- Coordination sublayer
For some applications it may be useful to group several functions in a coordination sublayer (the clamping cylinder of the clamp / release function, for example.)
- Function layer
Contains the state graphs used to control and monitor the various function units, for example, motors, valves, etc.



Creation of graph group

We distinguish three situations for which we usually create graph groups (indicated in the figure by the frame enclosing several state graphs):

- Graph group with coordination function

The graph group contains state graphs of the function layer, and higher-ranking state graphs for their coordination. Several state graphs may be used for coordination.

- Graph group without coordination function

This group is used only to group several state graphs (for example, hydraulic, pneumatic and coolant systems in the "auxiliary aggregates" graph group.)

- Graph group with coordination subfunctions

Here, the actual coordination is carried out by the master coordination function in the master PLC. The subcoordination shown in the example has the effect that several clamp cylinders seem to operate as a single function units (applies also the manual / setup control modes.)

9.1.7 Defining the program structure

Program elements required

S7-HiGraph generates an FC and a DB for each graph group. This FC is called in a block (OB, FB or FC).

In addition to the programs created in S7-HiGraph, we usually need further program elements.

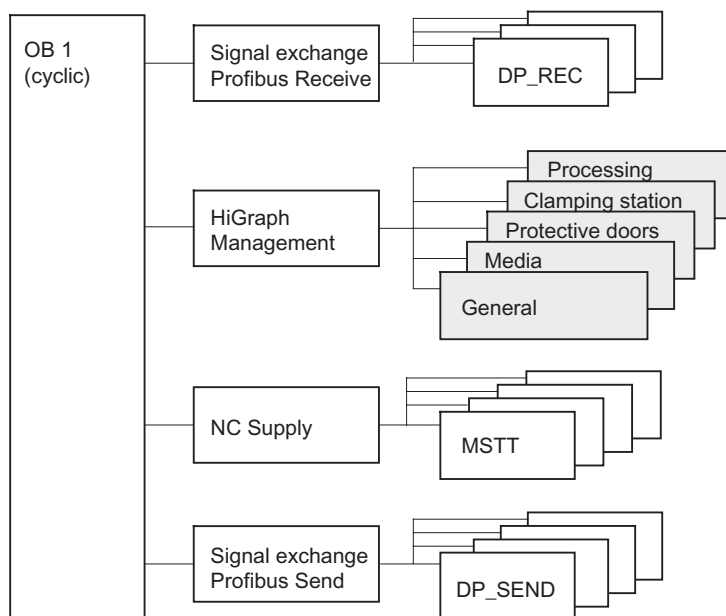
- Programs used to receive and prepare signals (signal exchange PROFIBUS Receive) output via PROFIBUS from the master PLC or transport control, or from the opposing processing unit.
- Program providing data to a connected NC station
- Program elements for sending data to the master PLC or to the opposing processing unit (signal exchange PROFIBUS Send).

Program structure

Useful program structure:

- All FCs generated in S7-HiGraph are called in a single "HiGraph Management" block
- All other program elements are also called in separate function blocks.
- All function blocks are called centrally and supplied with data by OB1.

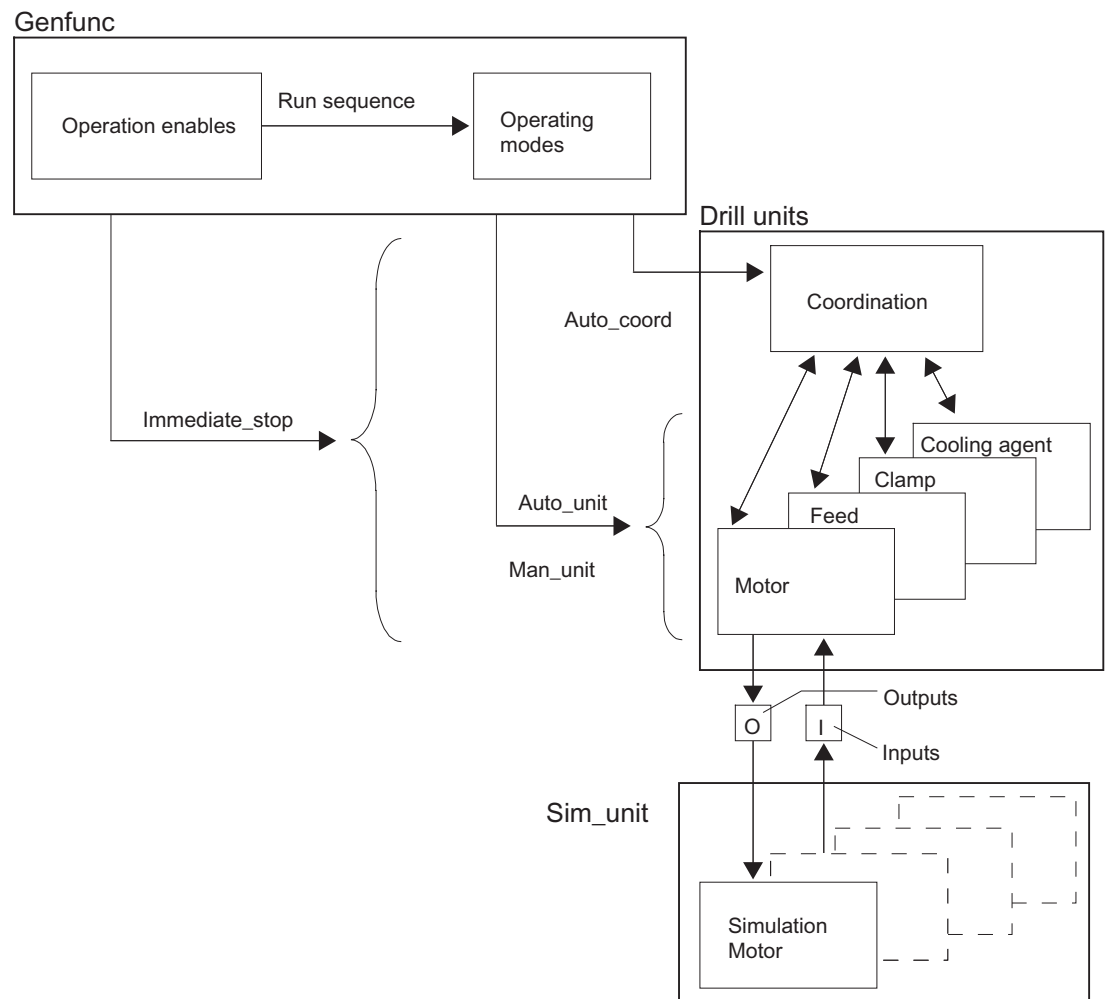
The figure below shows the program structure:



9.1.8 Overview: State graph and graph groups for the drill unit

Structure of state graphs and graph groups

How the graph groups and state graphs interact:



Graph group for operation enables and operating modes

A graph group contains functions for enabling operation and operating modes (GenFct.)

The state graph for enabling operation affects all state graphs by means of the operating mode state graph, because when runtime is disabled, all operating mode signals are also disabled. The graph also provides the "Instantaneous_Stop" signal which is evaluated in all relevant state graphs. This signal is used to freeze all motions as required.

The "Operating_mode" state graph provide the operating mode signals to all state graphs (including the state graphs for coordination, and the state graphs used to control and monitor the function units of the machine.)

Graph group for operation of the drill unit

A second graph group contains the state graph for operation of the drill unit. These state graphs usually control the actuators by means of output signals, and evaluate sensor signals.

Graph group for simulation

To simplify debugging, the included example contains functions for simulating actuators and sensors by means of state graphs. The third graph group contains the state graphs for this simulation.

9.1.9 State graph for controlling the operation enable signal

Production plants usually require hydraulic and pneumatic media supply systems which other function units need of operation.

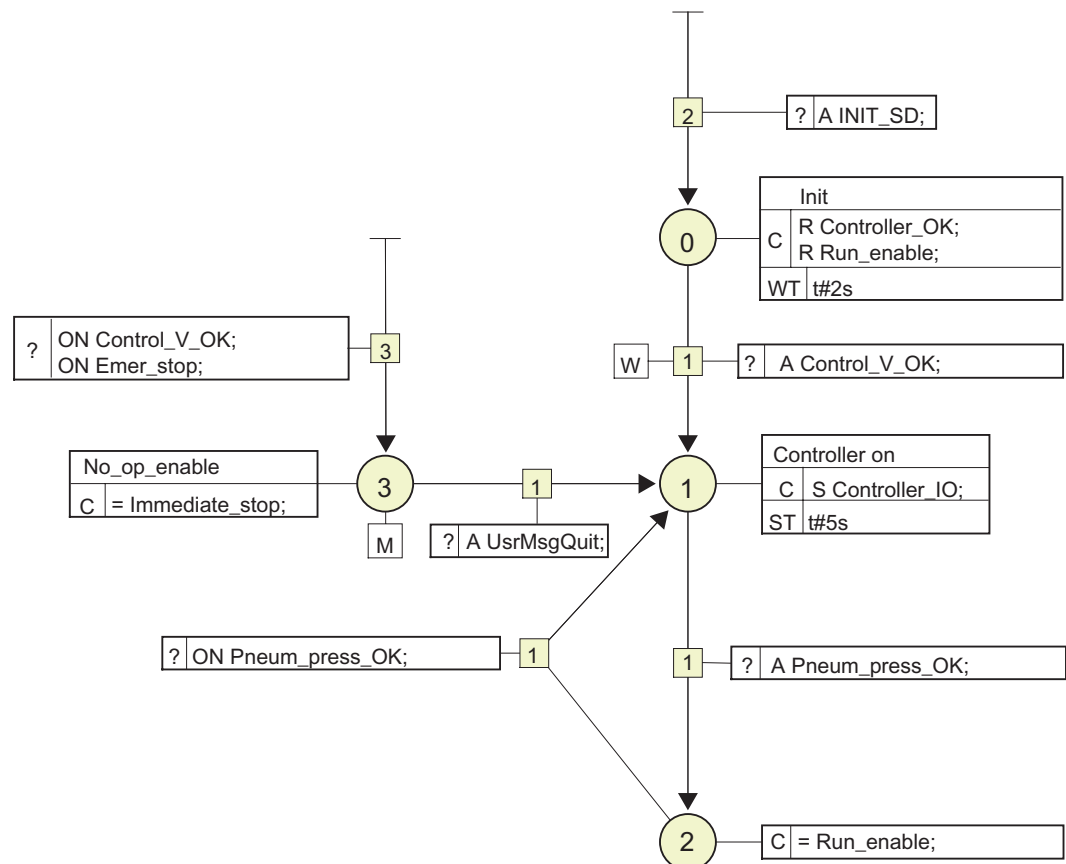
After startup of the plant, it is useful to switch on the hydraulic and pneumatic units at a central point and to create corresponding ready signals for other blocks.

Structure of the state graph

In our example, the following functions are controlled centrally:

- Control voltage
- Pneumatic system
- Evaluation of the "Emergency_Off" signal

These signals can be generated, for example, by means of a state graph as shown in the figure below:



Functions of the state graph

Functions provided by the state graph:

- After PLC startup, status 1 is reported with the "Controller_OK" signal in order to indicate that the control system has started and all power supplies are active. After the set pneumatic pressure is reached, the system status 2 enables the drill unit for operation (runtime enable.)
- Runtime is disabled again at the transition to 1 state when the pneumatic pressure drops below the set value. Motions already started will be completed. Runtime enable conditions are satisfied again when the pneumatic pressure reaches operational level.
- However, if the load power supply fails or an "Emergency_Off" signal is output, the system does not only reset the runtime enable signal, but also outputs the "Instantaneous_Stop" signal in status 3 in order to instantaneously freeze all motions. A message is also output to the operator panel. After the interrupt triggering event is cleared, operation of the transfer line is enabled again by pressing the acknowledgment button on the OP (with diagnostics by means of Format Converter.)

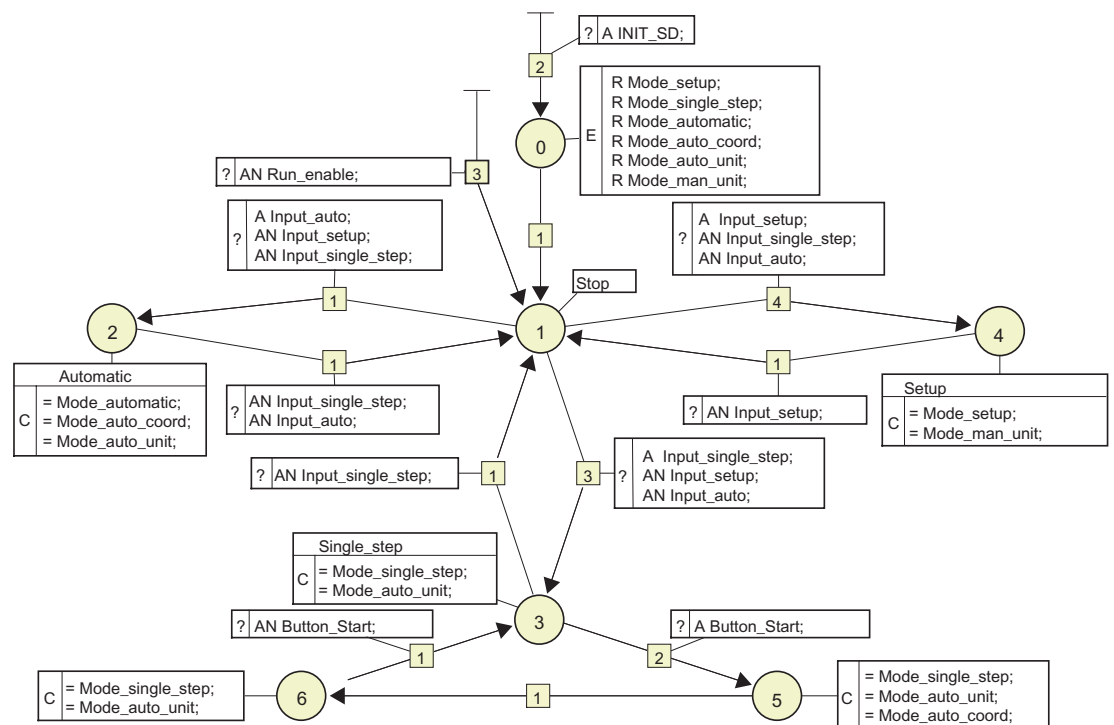
9.1.10 State graphs for controlling the operating modes

Structure of the state graph

The operating modes Auto, Step and Manual / Setup are selected for our example. The modes are set separately at the corresponding inputs (one input per mode.)

The output signals are used to indicate the selected operating mode and to control the nested state graphs.

Signal type	Signals
Signals for visualization	OM_Auto, OM_Step, OM_Setup
Signal for coordination	OM_Auto_Coord
Signals for function units	OM_Auto_Aggr, OM_Manual_Aggr



Functions of the state graph

Functions provided by the state graph:

- It decodes and debounces the signals of the mode selector switch. The corresponding operating mode is set when the signals have settled.
- The graph verifies the runtime enable signal continuously. Otherwise, an Any transition is generated in order to force status 1 and dwell in this state.
- It creates the operating mode signals for the state graphs on the coordination and function unit layer.

Syntax of the operating mode signals

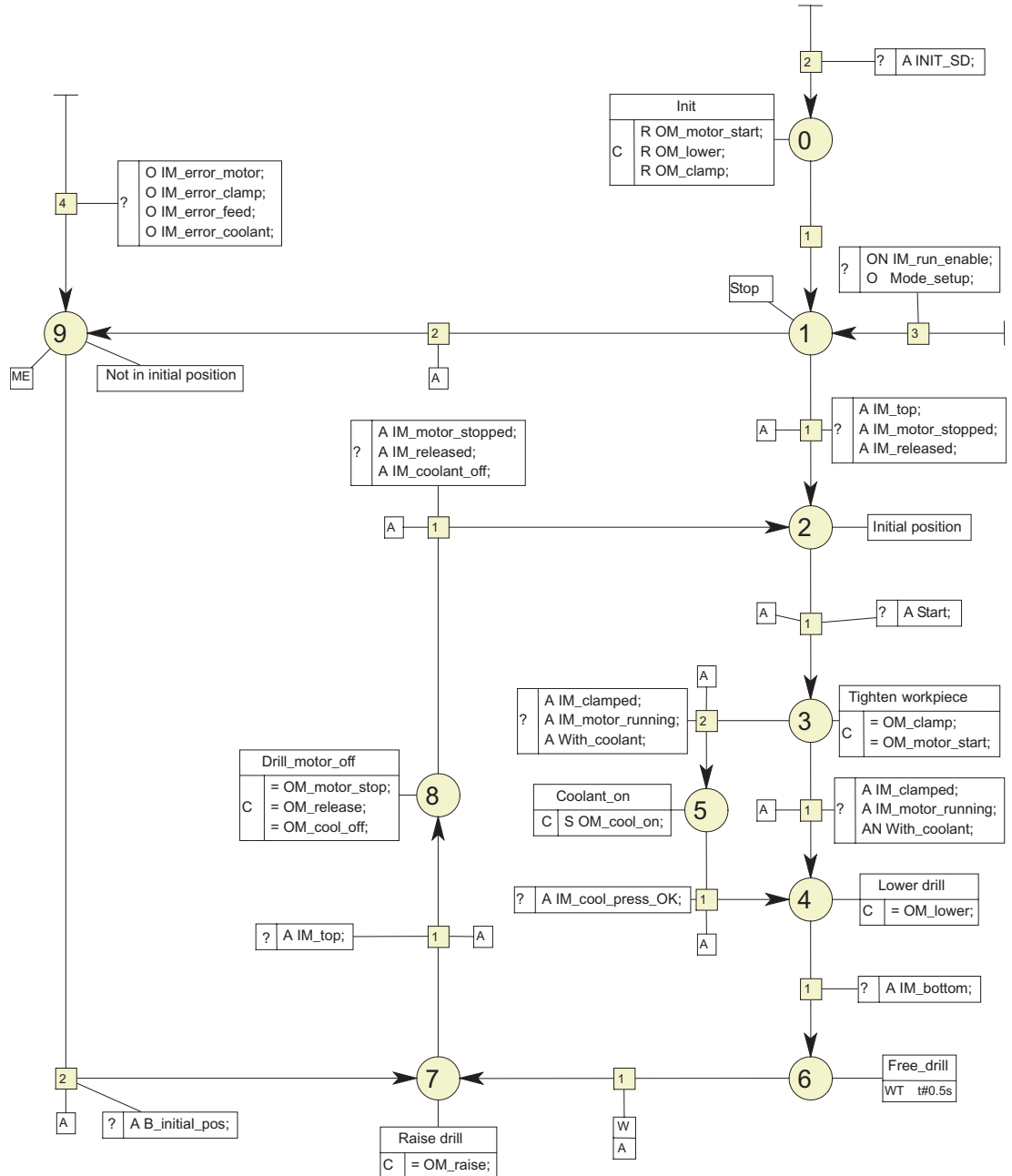
Operating mode signals are generated based on the following syntax:

- The input signals e_AutoMode, e_StepMode and e_SetupMode indicate which mode the operator selected.
- In Auto mode, the signals OM_Auto_Aggr and OM_Auto_Coord are set at the state graphs of the coordination and function unit layer.
- In Step mode, status 3 is set initially only at the state graphs of the function unit layer of auto mode (signal OM_Auto_Aggr). Because of the missing OM_Auto_Coord signal, the transitions with "Auto" attribute are disabled in the coordination state graph in order to suppress the transition to the next state. By pressing the Start button, the operator sets the OM_Auto_Coord signal at the coordination state graph for the period of one cycle, and thus enables the transition to the next state.
- In manual / setup mode, the OM_Manual_Aggr signal is used to enable manual mode at the state graphs on the function unit layer. Status 1 (HOLD) is set at the coordination state graph and thus disables it.

9.1.11 State graph for coordination of the drill unit

Structure of the state graph

Structure of the state graph for coordination of the drill unit:



Functions of the state graph

Functions of the coordination state graph "Drill":

- Initialization
- Auto mode
- Stepping mode
- Manual mode
- Cancel conditions

Initialization

Status 1 is set immediately after initialization. The state graph is frozen in this state by means of an Any transition if runtime is not enabled or Manual / Setup not set.

If the conditions in the Any transition are not satisfied, the home position of the drill unit is queried by means of transition 1 at status 1. If this is the current status, a transition to state 2 is executed and thus enables start of the drilling operation. If the drill unit is not in home position, transition 9 is used to set status 9 and a message is output on the HMI. The operator can use the T_Home_Position button to run the drill to home position.

Auto mode and stepping mode

In order to implement the operating modes, the relevant transitions are assigned the "Auto" attribute (indicated in the graphic view by a small flag with the letter "A".) Transitions with this attribute can only be switched when auto mode is set, or the start button is pressed in stepping mode.

In our example, the corresponding default variable "AutomaticMode" (section IN of the variable declaration) is set at the "OM_Auto_Coord" (M 10.5) flag by the "Operating modes" state graph. In auto mode, its status is always "1", and in "Stepping mode" it is "1" for the duration of one cycle after the start button is pressed.

Manual / Setup mode:

The "OM_Setup" signal is set to state 1 (HOLD) by means of an Any transition, thus disabling the state graph. Motions are then controlled directly using the button signals of the OP. The state graphs for the various function units take this into account.

Cancel situations:

A cancel situation is given when

- runtime enable is reset (set by the "Operation enables" state graph),
- the mode is toggled to Manual / Setup mode, or
- an error has occurred in the nested state graphs.

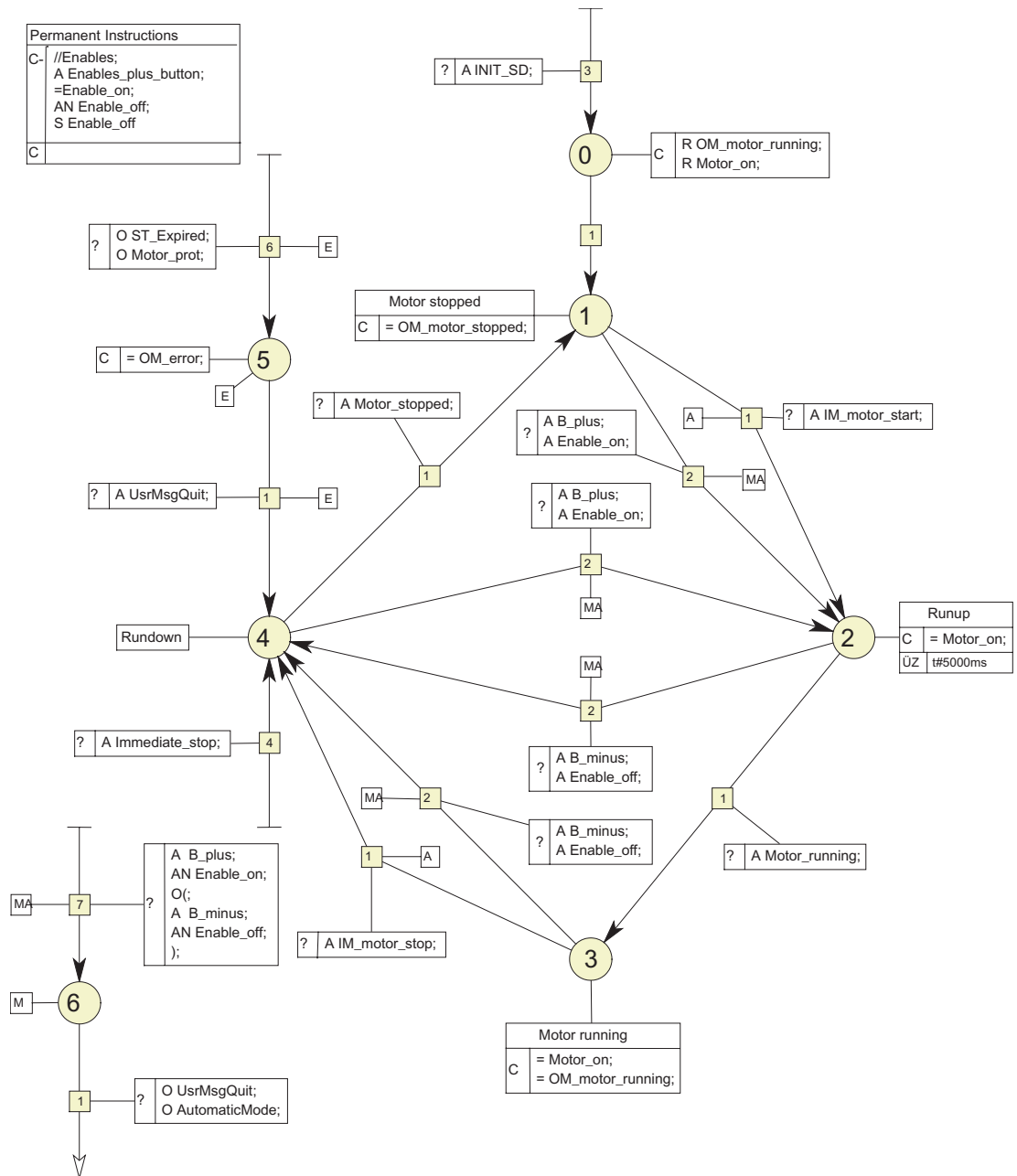
In the first two situations, the "Drill" state graph sets status 1 (HOLD), and in the last situation it sets status 9.

When the mode is toggled to Auto mode, a direct start of the drill unit is only possible when it is in home position. Otherwise, the operator has to run the unit to home position by pressing the "T_Home_Position" button.

9.1.12 Motor state graph

Structure of the state graph

Structure of the "Motor" state graph:



In our example we are going to use the state graph both for controlling the motor and the coolant, that is, we need to create two instances.

Functions of the state graph

Functions of the "Motor" state graph:

- Initialization
- Auto mode
- Manual / Setup mode
- Monitoring functions.

Initialization

The motor is always switched off during initialization.

Auto mode

In order to implement this operating mode, the relevant transitions are assigned the "Auto" attribute (in the graphical view, a small flag with an "A" letter.) That is, they can always be switched when the corresponding default variable "AutomaticMode" = 1. In auto and stepping mode, the signal is set by the "Operating modes" state graph. In those situations, the state graph receives its commands (messages) from the coordination graph.

Manual / Setup mode

"Manual / Setup" mode is implemented by means of transitions with "Manual" attribute (in the graphical view, a small flag with an "A" letter.) When this mode is set, the variable ManualMode = 1 and the relevant transitions are enabled.

The state graph can now be controlled by means of button signals. The conditions for switching on the motor in manual mode are included in the permanent instructions. Our example requires only the "Enable_ButtonPlus" condition to switch on the motor.

The "Enable_On" and "Enable_Off" signals can be used for visualization. For example, the currently permitted operation can be indicated on the HMI.

When the operator presses the Plus button, for example, although it is not allowed to switch on the drill motor, status 6 is set and a message is output on the OP.

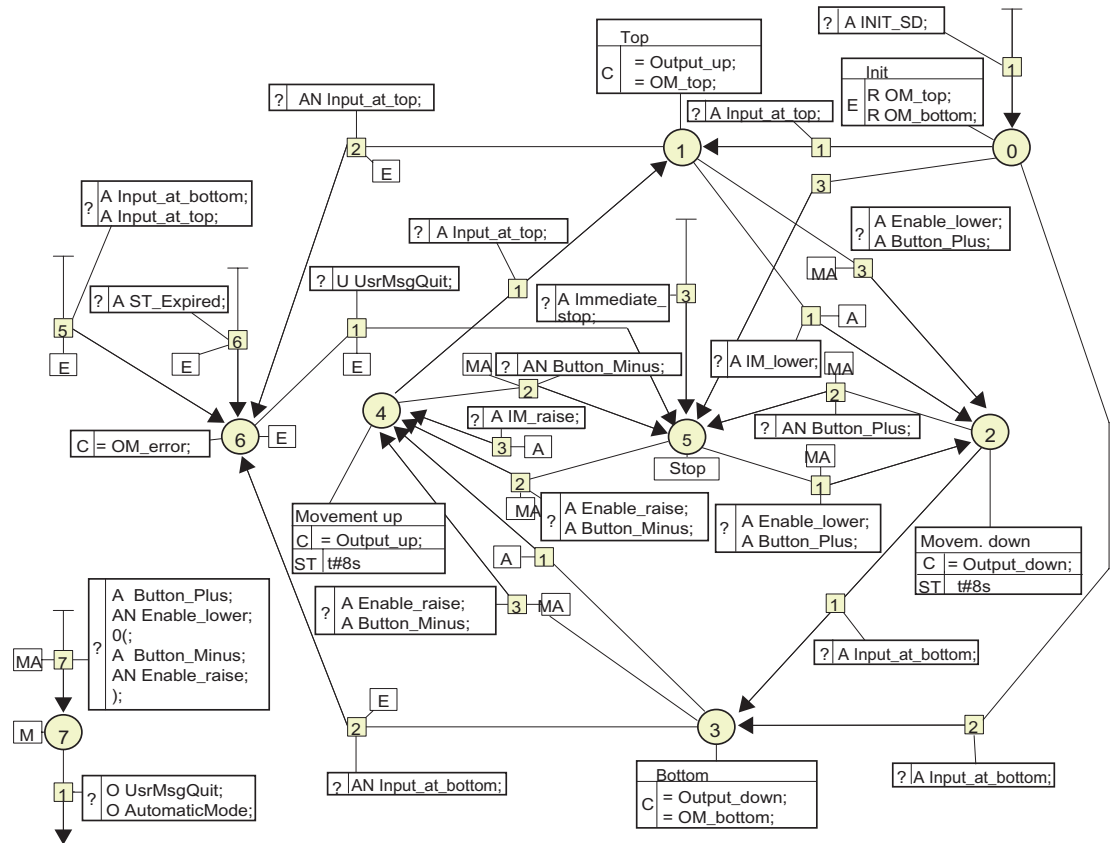
Monitoring functions

The motor state graph contains a startup watchdog timer. This timer function is defined in the status 2 (startup) instruction set. An error message is written to the diagnostic buffer of the As when the set startup time is violated. This message is acknowledged when the transition exits status 2. In our example we have not programmed any other reactions to this error state.

9.1.13 Status graph Valve_2E

Structure of the state graph

Structure of the "Valve_2E" state graph:



Functions of the state graph

Functions of the "Valve_2E" state graph:

- extended initialization function
- Auto mode
- Manual / Setup mode
- Monitoring functions
- Cancel

Initialization

The "Top" and "Bottom" limit positions are evaluated during the initialization. When the valve is in a defined limit position, the corresponding status (status 1 or 2) is assumed. When the valve is not in a defined limit position, status 5 (Hold) is assumed.

Auto mode

In order to implement this operating mode, the relevant transitions are assigned the "Auto" attribute (in the graphical view, a small flag with an "A" letter.) That is, they can always be switched when the corresponding default variable "AutomaticMode" = 1. In auto and stepping mode, the signal is set by the "Operating modes" state graph. In those situations, the state graph receives its commands (messages) from the coordination graph.

Manual / Setup mode

"Manual / Setup" mode is implemented by means of transitions with "Manual" attribute (in the graphical view, a small flag with an "A" letter.) When this mode is set, the variable ManualMode = 1 and the relevant transitions are enabled. The state graph can now be controlled by means of button signals.

Monitoring functions

Functions monitored:

- Time of motion
This monitoring function is implemented by setting monitoring times for status 1 and 4. If the permitted time of motion is exceeded, the "ST_Expired" variable is evaluated, and status 6 is set by means of the Any transition 4. This interrupts the motion.
- Faulty limit switch signals (double signal)
The Any transition 3 continuously monitors for faulty limit switch signals. When both limit switches output a limit position reached signal, status 6 is set. This status is assigned the "Error" attribute and thus triggers an error message.
- Drifting off limit position
Drifting off limit positions is monitored by evaluating the corresponding limit signals in the states 1 and 3 which are assigned to the limit positions. When the corresponding limit switch signal changes to "0" state, error state 6 is set and an error message is generated.

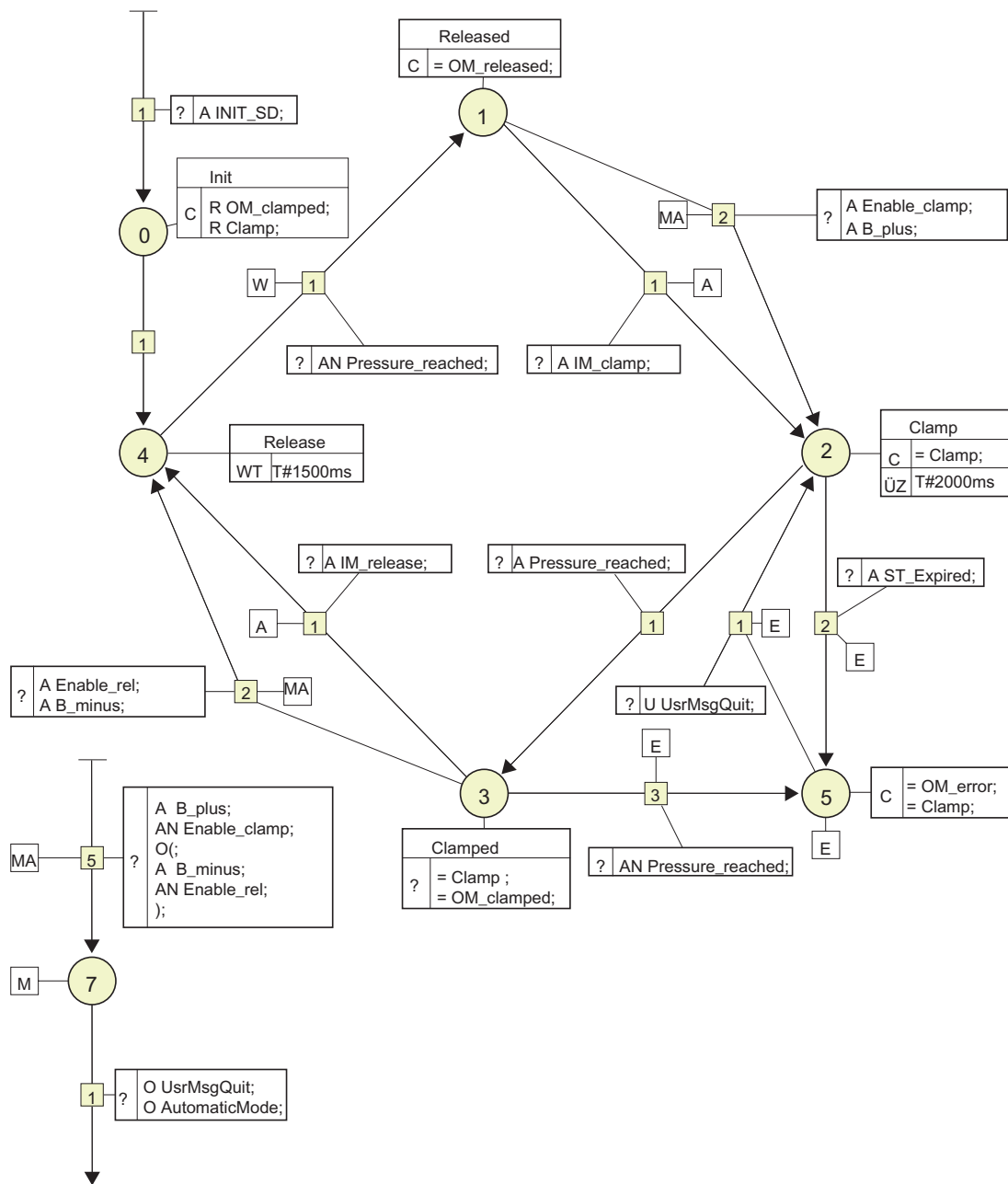
Cancel situations

When an "Instant_Stop" signal is output, an Any transition sets status 5 (Hold.)

9.1.14 Clamp state graph

Structure of the state graph

Structure of the state graph



Functions of the state graph

Functions of the "Clamp" state graph:

- Initialization
- Auto mode
- Manual / Setup mode
- Monitoring functions

Initialization

Status 4 (release) is set during initialization.

Auto mode

In order to implement this operating mode, the relevant transitions are assigned the "Auto" attribute (in the graphical view, a small flag with an "A" letter.) That is, they can always be switched when the corresponding default variable "AutomaticMode" = 1. In auto and stepping mode, the signal is set by the "Operating modes" state graph. In those situations, the state graph receives its commands (messages) from the coordination graph.

Manual / Setup mode

"Manual / Setup" mode is implemented by means of transitions with "Manual" attribute (in the graphical view, a small flag with an "A" letter.) When this mode is set, the variable ManualMode = 1 and the relevant transitions are enabled. The state graph can now be controlled by means of button signals.

Monitoring functions

Functions monitored:

- Duration of motion
This monitoring function is implemented by setting a monitoring time for status 2. If the permitted duration of motion is exceeded, the "ST_Expired" variable is evaluated, and transition 2 sets status 5. This interrupts the motion. This error has to be acknowledged in order to continue operation.
- Clamping pressure in clamped state
If the clamping pressure drops below the permissible limit during drilling, the system also branches to state 5 and the drilling process is interrupted.

9.1.15 Compiler settings for the sample configuration

Compiler settings

In order for the displayed state graphs to execute the desired functions, the following settings have to be carried out (select **Options > Settings for Graph Groups > Compile**):

Source	Settings
Graph group "General functions"	The "Cyclic actions with RLO = 0" option must be set. This has the effect that instructions modifying RLO are executed in the cyclic instructions of the states with RLO = 0 in the case of a state change.
Graph group "Drill unit"	In this graph group the options "Cyclic actions with RLO 0" and "Switch Any transition only once" must be activated.

9.2 Example of the configuration "Drill unit with standard diagnostics"

Learning objectives

The sample configuration "Drill unit with ProAgent" is based on the "Drill unit" example. You learn how to configure a diagnostics function for the "Drill unit" project.

Storage location of the sample project

An example of the configuration of the diagnostic inclusion is supplied in the "Drill unit" project included (program folder "Drill unit with ProAgent").

The "Drill unit" graph group contained in this example is based on an operator panel with soft keys. State graphs are also provided for other operator panels. However, they are not included in the graph group. The extensions of the state graph names provide information on the corresponding operator panels:

- Extension _SK: For operator panels with soft keys
- Extension _DK: For operator panels with direct keys
- Extension _TP: For touch panels

See also

Requirements of standard diagnostics (Page 6-4)

9.2.1 Configuring diagnostics for the "Operating modes" state graph

Diagnostics configuration at the state graph

The state graph described earlier for generating operating mode signals can be applied without changes.

View in ProAgent

It is only necessary to configure a corresponding field in which the set operating mode is shown in the ProAgent overview with ProTool means.

Selecting the operating mode

The operating mode is selected by means of switches or buttons at the operating panel.

See also

Requirements of standard diagnostics (Page 6-4)

9.2.2 Configuring diagnostics for the "Operation enables" state graph

Visualization of the state graph

In ProAgent, the state graphs for the coordination of a graph group are indicated on the overview screen.

Diagnostics configuration at the state graph

Requirements for configuring diagnostics functions:

- The state graphs are available in S7-HiGraph V5 file format.
- Set the "Instances of the state graph visible at the display unit as a unit" option (select **Options > Settings for Graph Groups/State Graphs**, "Diagnostics" tab.)
- If messages are to be generated in a state graph with coordination function, the initial value detection and display acknowledgement must be configured. (Select **Options > Settings for Graph Groups/State Graphs**, "Diagnostics" tab).

9.2.3 Diagnostic configuration in the motion state graphs

Visualization of the state graph

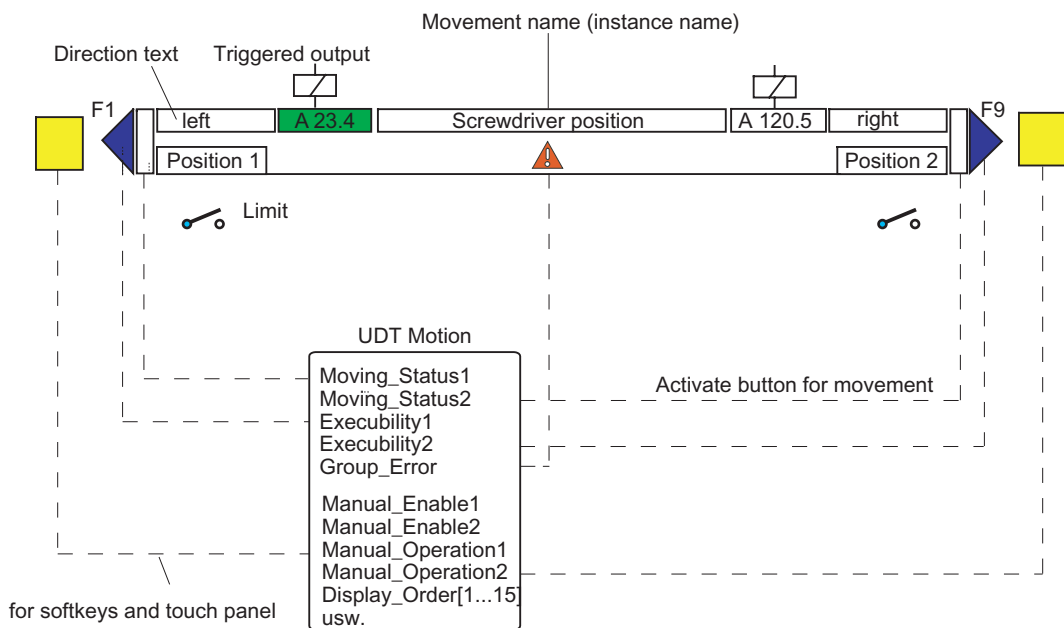
The ProAgent motion screen visualizes "motions", that is, the state of function units performing a motion on the machine or in the process (valves or motors, for example.) One "motion line" is reserved for each motion.

The motions can be operated in manual mode using the keys assigned to the line.

In S7-HiGraph, the display fields of a motion line have to be provided data and the key signals have to be evaluated.

Providing data to the display fields

Each motion line of a structure (HiGraphMotionUDT) serves as the data interface between the PLC and the OP, through which the required signals are exchanged. The following figure shows the relationship schematically.



- Setting the "Moving-Status<no.>" of the HiGraphMotionUDT in the motion states
- Setting manual mode interlock and enable conditions for the "Executability<no.>" signals of the HiGraphMotionUDT, for example, in the previous cyclic actions

- Depending on the layout of the keys for manual mode, either evaluation of the "Manual_Operation<No.>" signals, or of the "Display_Order" signals interconnected with the key signals in the transitions for manual mode.
- The following entries are to be carried out under Options > Settings for Graph Groups/State Graphs, "Standard diagnostics" tab:
 - Set the "State graph realizes a movement" option
 - Enter a variable name for the UDT (a variable of the type HiGraphMotionUDT is then created in the STAT declaration section by S7-HiGraph)
 - Enter direction texts
 - Specify the controlled outputs for the direction texts
 - Enter the address for the limits
 - If desired, carry out state-graph-specific settings for the initial-value detection and for message acknowledgement
 - If desired, set the "Instances of the state graph visible at the display unit as a unit" option.

Evaluation of the button signals

The button signals are configured in accordance with the OP design. We distinguish three OP variants:

OP with softkeys	In the case of softkeys, the "Manual_Enable1" or "Manual_Enable2" bits belonging to the motion are set as soon as the buttons next to the indicated motion are operated. These signals can then be evaluated in the transitions for manual mode.
OP with direct keys	<p>In the case of direct keys, the keys must be assigned to the indicated motions in the state graphs. For this purpose the OP sets that bit in the respective UDTs in the "Display-Order" bit arrays corresponding to the line indicating the relevant motion, that is, bit 3 is set when the motion is shown in the 3rd line, for example. In S7-HiGraph, it is advisable to configure a corresponding interconnection of the direct key signals with the "Display_Order" signals in the permanent instructions C-. This would take the following form for the "Direction movement minus" key, under the assumption that up to four movements are shown on the OP, for example:</p> <pre>// Button signals o(u Key_Minus1; u Motion.Display_Order[1];) o(u Key_Minus2; u Motion.Display_Order[2];) o(u Key_Minus3; u Motion.Display_Order[3];) o(u Key_Minus4; u Motion.Display_Order[4];) = KeyMinus; Note: The Key_Minus1 to Key_Minus4 signals are defined as shared signals in the symbol table, whereas the KeyMinus signal is defined in VarStat as a variable.</pre>
Touch panels (TPs)	<p>Softkeys are used in touch panels. In addition, the user must first select the motion which is to be operated in order to increase safety in operating. If a movement is selected, the Manual_Enable1 and Manual_Enable2 signals are set by the OP in the corresponding HiGraphMotionUDT. In these signals, the manual transitions must be interconnected with the softkey signals Manual_Operation1 and Manual_Operation2.</p> <p>The interconnection for the "Motion direction minus" key in the permanent instructions C- would have the following form for the Valve_2E state graph from the "Drill unit" graph group:</p> <pre>u Manual_Enable1; u Manual_Operation1; = KeyMinus;</pre>

9.2.4 Diagnostics configuration in graph groups

Diagnostic configuration at the graph groups

Select **Options > Settings for Graph Groups/State Graphs > Diagnostics**:

- Set the "Standard Diagnostics" option.
- In the "Diagnostic DB" input box, type in the name or number of the diagnostic DB which belongs to the graph group.
- If the graph group represents a technological unit which is to be displayed in the overview of ProAgent, set the " Graph group visible at the display unit as a unit" option.
- If initial value recording is required for error messages and/or operating messages, enable the corresponding fields.

Note

The initial values which belong to a message are indicated on the detail screen of ProAgent. This means that nothing is shown on the detail screen if initial value acquisition is not enabled in S7-HiGraph.

- If error and/or status messages need to be acknowledged, set the corresponding boxes.

Note

You can select **Options > Application Settings > Diagnostics** to adapt error and status message texts.

Control elements

10.1 Calling up the shortcut menus

Menu commands used frequently in the current context are also available in the shortcut menus:

Follow these steps:

1. Right-click an object or in a free area in a working window.
The shortcut menu opens.
2. Select a command.

10.2 Tooltips

A short description of the following elements of a state graph diagram opens when the cursor is positioned briefly on the element.

User interface objects

Object on which the cursor rests	View in the tooltip
State / state name	State number State name
Transition arrow / Transition name	Numbers of the previous and subsequent states Priority Name of the transition
Instance of a state graph	Run sequence Instance name Name of the state graph
Message arrow Message node	Names of the instances which send or receive the message Names of the message variables
Address from the variable declaration	Variable name Declaration section Data type Message type
Variable in the variable overview	Name Data type

Addresses during the execution of the program status

Address on which the cursor rests	View in the tooltip
Shared address	Symbolic name Absolute address Comment from the symbol table
Address from the variable declaration	Variable name Declaration section Data type Comment from the variable declaration



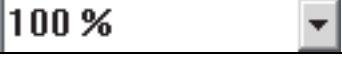










10.3 Symbols

10.3.1 Toolbar icons

List of icons

The toolbars contain the following icons which can be used for quick access to certain menu commands.

Symbol	Function
	New Graph Group (File Menu)
	New State Graph (File Menu)
	Open (File Menu)
	Save (File Menu)
	Print (File menu)
	Cut (Edit menu)
	Copy (Edit menu)
	Paste (Edit menu)
	Undo (Edit menu)
	Redo (Edit menu)
	Compile (File menu)
	Download (PLC menu)
	Monitor (Debug menu)
	Go To, Previous Error (Edit menu)
	Go To, Next Error (Edit menu)
	Context-sensitive Help (Help menu)

Symbol	Function
	Variable Overview (View menu)
	Details (View menu)
	Zoom, Zoom Factor (View menu)
	Zoom, Zoom In (View menu)
	Zoom, Zoom Out (View menu)
	Zoom, Select Zoom Area (View menu)
	Zoom, All Windows (View menu)
	New Window (Window menu)
	Arrange, Cascade (Window menu)
	Arrange, Vertically (Window menu)
	Arrange, Horizontally (Window menu)
	State (Insert menu)
	Transition (Insert menu)

Note

You can show or hide the icons of the toolbars by selecting **View > Toolbars**

10.4 Keyboard control

10.4.1 Selecting menu commands with keystrokes

All menu commands can be triggered by entering the corresponding ALT+key shortcut.

Procedure:

1. Press ALT and hold it down
2. Then press the underscored character in the relevant menu (for example, Alt+F for the "File" menu - if the "File" menu is entered in the menu bar).
The menu is expanded.
3. Next, press the underscored character in the relevant menu command (for example, N for "New".)
4. If the menu command has further submenus, these are then opened. Use the same procedure until you have selected the complete menu command by typing in the relevant character.

The menu command is triggered when you enter the last character of the keystroke.

Examples:

Menu commands	Keys
File > New	Alt + F, N
Window > Arrange > Horizontal	Alt + W, A, H

10.4.2 Moving the cursor in the text editor

Keystrokes for moving the cursor

Function	Keys
One line up.	ARROW UP
One line down.	ARROW DOWN
One character to the right.	ARROW RIGHT
One character to the left.	ARROW LEFT
One word to the right.	CTRL + ARROW RIGHT
One word to the left.	CTRL + ARROW LEFT
To the beginning of the line.	Pos1
To the end of the line.	Ende
One page up.	Bild-Auf
One page down.	Bild-Ab
To beginning of text.	CTRL + HOME
To end of text.	CTRL + END

10.4.3 Moving the cursor in dialog boxes

Keystrokes for moving the cursor in dialog boxes

Function	Keys
Move to the next input box (from left to right and from top to bottom).	TAB
Move one input box back.	SHIFT + TAB
Move to the input box that contains the underscored character X.	Alt + X
Selects objects from a selection list.	Arrow keys
Opens a selection list.	ALT + ARROW DOWN
Select an object or undo a selection.	SPACE BAR
Confirm entries and close the dialog box ("OK" command button).	Eingabetaste
Closes the dialog box without saving the selection ("Cancel" button).	ESC

10.4.4 Selecting menu commands form the menu bar / pop-up menu

Keystrokes for selecting menu commands

Function	Keys
To the menu bar.	F10
To the pop-up menu.	SHIFT + F10
To the menu containing the underscored letter X.	Alt + X
Assigned menu command.	Underscored letter in the menu command
One menu command to the left.	ARROW LEFT
One menu command to the right.	ARROW RIGHT
One menu command up.	ARROW UP
One menu command down.	ARROW DOWN
Activates the selected menu command.	Eingabetaste
Exits the menu or returns to the text.	ESC

10.4.5 Selecting texts with key commands

Keystrokes for marking text

Function	Keys
One character to the right.	SHIFT + ARROW RIGHT
One character to the left.	SHIFT + ARROW LEFT
One word to the right.	CTRL + SHIFT + ARROW RIGHT
One word to the left.	CTRL + SHIFT + ARROW LEFT
To the beginning of the line.	SHIFT + HOME
To the end of the line.	SHIFT + END
One line up.	SHIFT + ARROW UP
One line down.	SHIFT + ARROW DOWN
One page up.	SHIFT + PgUp
One page down.	SHIFT + PgDn
To file beginning.	CTRL + SHIFT + HOME
To file end.	CTRL + SHIFT + END

10.4.6 Accessing the help with key commands

Keystrokes for calling the Help

Function	Keys
Open the help.	F1 If there is a current context, for example, an opened dialog box, the corresponding help topic is called. Otherwise, the help table of contents is called up.
Click the question-mark symbol for the context-sensitive help	SHIFT + F1
Exit the help window and return to the editor window.	ALT + F4

10.4.7 International/German keyboard layout

German and international keyboard layout

International keyboard layout	German keyboard layout
Home	Pos1
End	Ende
PgUp	Bild-Auf
PgDn	Bild-Ab
Ctrl	Strg
Enter	Eingabetaste
Delete	Entf
Insert	Einfg
Shift	Umschalt

Technical support

11.1 Further technical support

Further support

If you have any questions relating to the products described in this manual and do not find the answers, please contact your Siemens partner at our local offices.

Your contact partner is found under:

<http://www.siemens.com/automation/partner>

The guide to our technical documentation for the various SIMATIC products and systems is found under:

<http://www.siemens.de/simatic-tech-doku-portal>

11.2 Service & Support on the Internet

Service & Support on the Internet

In addition to our documentation, we also offer you a comprehensive online knowledge base on the Internet.

<http://www.siemens.com/automation/service&support>

There you find:

- our Newsletter, which offers you the latest information on your products.
- your right documentation using our Service & Support search engine.
- a bulletin board where users and specialists exchange their knowledge worldwide.
- your local contact partner for Automation & Drives.
- information about on-site service, repairs and spare parts. And lots more under "Services".

11.3 A&D Technical Support

A&D Technical Support

Available 24 hours worldwide:

Worldwide (Nuremberg) Technical Support Local time: 0:00 to 24:00 / 365 days Phone: +49 (180) 5050-222 Fax: +49 (180) 5050-223 mailto:adsupport@siemens.com GMT: +1:00		
Europe / Africa (Nuremberg) Local time: Mo.-Fr. 8:00 to 17:00 Phone: +49 (180) 5050-222 Fax: +49 (180) 5050-223 mailto:adsupport@siemens.com GMT: +1:00	United States (Johnson City) Technical Support and Authorization Local time: Mo.-Fr. 8:00 to 17:00 Phone: +1 (423) 262 2522 Fax: +1 (423) 262 2289 mailto:simatic.hotline@sea.siemens.com GMT: -5:00	Asia / Australia (Beijing) Technical Support and Authorization Local time: Mo.-Fr. 8:00 to 17:00 Phone: +86 10 64 75 75 75 Fax: +86 10 64 74 74 74 mailto:adsupport.asia@siemens.com GMT: +8:00
German and English are generally spoken at our Technical Support and Authorization Centers.		

11.4 Training Center

Training Center

Siemens offers corresponding training courses to get you started with HiGraph and the S7 automation system. Please contact your regional Training Center, or the central Training Center in D90327 Nuremberg.

Phone: +49 (911) 895-3200.

Internet: <http://www.sitrain.com>

Glossary

Address

An address is part of a STEP 7 instruction and specifies with what the processor is to do something. It can be addressed either absolutely or symbolically.

Any transition

An Any transition is a particular type of transition. An Any transition goes from all states to a target state. Any transitions are continually processed independently of the current state of a state graph. Any transitions are used, for example, for the permanent supervision of invalid signals. If the supervision situation programmed in the Any transition occurs, the process branches to a target state.

Compilation

Compilation is the generation of an executable user program from a source file.

Current parameter

Current parameters replace the formal parameters when a state graph is compiled. Example: The formal parameter "Start" is replaced by the current parameter "I 3.6".

Cycle time

The cycle time is the time which the CPU needs to execute the user program once.

Data block (DB)

Data blocks (DB) are data areas in the user program which contain the user data. There are shared data blocks which can be accessed by all the code blocks and there are instance data blocks which are assigned to a particular FB call.

Data type

A data type is used to determine how the value of a variable or a constant is to be used in the user program.

In SIMATIC S7 there are two types of data types available to the user in accordance with IEC 1131-3: standard and derived data types.

Derived data type

Derived data types are created by the user by means of the data type declaration. They do not have an own name and can therefore not be used several times. A differentiation is made between fields and structures. The data types STRING and DATE AND TIME are such data types.

Diagnostic buffer

Buffered memory area in the CPU, in which diagnostic events are saved in the sequence of occurrence.

Fault signal

Displays a fault in the process.

Formal parameter

A formal parameter is a token value for the current parameter in the case of configurable code blocks. In the case of FBs and FCs the formal parameters are declared by the user. In the case of SFBs and SFCs they already exist. When the block is called, a current parameter is assigned to the formal parameter so that the called block operates with this current value. The formal parameters are part of the block-specific data of the block and are divided into the input, output and in/out parameters.

Functional unit

Functional units are the smallest physical objects within a plant or machine which can only have one state at any one time (for example, a valve). In S7-HiGraph functional units are represented by state graphs.

Graph group

A graph group is a number of state graphs belonging together which can be compiled, downloaded and saved. It defines an ordered sequence of calls to state graph, which is executed cyclically during the program execution.

Initial state

Specifies which state a functional unit should assume at power on.

Initial value

Signal state of the address which causes a fault or operation message.

Instance

In S7-HiGraph the term instance is used for the call of a state graph in a graph group.

Operation

An operation is part of a STEP 7 statement and specifies what the processor is to do.

Instruction

Smallest independent unit of a user program written in a text language. It represents an operation sequence for the processor.

Message

State graphs can influence one another in the way they are executed by exchanging messages.

Message acknowledgement

Input of the operator at the display unit with which he confirms that he has read a message. Messages which must be acknowledged may not disappear "unread" when the message cause no longer exists.

Message acknowledge memory

Memory area in the PLC for messages and message acknowledgements which occur in connection with a process error diagnostics.

Mnemonics

The mnemonics are the abbreviated representation of the addresses and programming instructions in the program (for example, "I" stands for input).

STEP 7 supports the international representation IEC (in English) and the SIMATIC representation (based on the German names for the instructions and the conventions for SIMATIC addressing).

Operating mode

The operating mode defines the method by which a machine or plant operates (for example in automatic mode, manual mode, setup mode).

Operation message

Operation messages indicate a status in the process.

Operation messages are often used to display invalid operations. Example: A motor is to be activated by operator control, although this is not allowed due to an open protective door.

Operator panel (OP)

Operator panel for rapidly accessing the machine, for example, in order to specify setpoint values or to output machine data.

Organization block

Organization blocks form the interface between the operating system of the CPU and the user program. The sequence for executing the user program is specified in the organization blocks.

Permanent instructions

Permanent instructions are executed once per execution cycle of a state graph irrespective of the current state.

Predefined variable

Predefined variables are variables which are entered automatically into the variable declaration when a state graph or a graph group is created.

Process error diagnostics

Localization of errors in the process (outside the PLC). For process error diagnostics you require a program item which can determine the error source for example, by comparing the setpoint and actual states of the process.

Project

A folder for all the objects of an automation solution irrespective of the number of stations, modules and their networking.

Return transition

A return transition returns from the current state to the previously active state.

Run sequence

Sequence in which the instances contained in a graph group are executed.

S7-HiGraph

Programming language for the comfortable functional description of technological objects in the form of state graphs.

S7-HiGraph source file

An S7-HiGraph source file is part of an S7 program that is created with S7-HiGraph and from which an executable function (FC) is generated by compilation.

S7 program

A folder for blocks, source files and charts for programmable S7 modules which also contains the symbol table.

SIMATIC Manager

Graphics user interface for SIMATIC users under Windows.

Standard data types

Standard data types are predefined data types in accordance with IEC 1131-3. Examples:

- "BOOL" defines a binary variable ("Bit");
- Data type "INT" defines a 16-bit fixed-point variable

Startup transition

Transition for initializing a state graph.

The startup transition ends in the default setting of state 0. It queries the preset variable INIT_SD, so that if this variable has the signal 1, it branches to state 0. If you make sure that the parameter "INIT_SD" has the signal 1 in the calling block on startup, the state graph will be initialized with this value.

State

Every state which a functional unit can have is represented by a state in the state graph.

A state graph can never be in more than one state at any one time. The states have instructions assigned to them which are executed if the state is active.

State graph

State graphs describe the behavior of functional units. They define states, which the functional units can have and the transitions between the states. The entire function of the plant or machine is represented by a combination of state graphs.

Statement List (STL)

The statement list (STL) is a machine text-based programming language. STL is the assembly language of STEP 5 and STEP 7. If a program is programmed in STL, the individual instruction statements correspond to the sequences with which the CPU executes the program.

Station

Device which can be connected to one or more subnets as a connected unit, for example, programmable logic controllers, programming devices, operator stations.

Status

The status is the designation for the signal state of an address in the programmable logic controller.

Status display

The status display is the display of the signal state of one or more addresses on the screen or display of a programming device connected online to the programmable logic controller.

Status overview

The status overview is the status display of a graph group.

Symbol

A symbol is a name defined by the user under observance of certain syntax rules. This name can be used after you have specified what it is to represent (for example, variable, block) for programming and for operator control and monitoring. Example: Address: I 5.0, Data type: Bool, Symbol: Emergency stop.

Symbol table

Table for assigning symbols (= names) to addresses for shared data and blocks. Examples: Emergency stop (symbol) - I 1.7 (address) or closed-loop controller (symbol) - SFB 24 (block).

System attributes

You can assign the following system attributes to parameters in HiGraph.

- **S7_active**
Displays whether the declaration of the parameter is active or inactive.
- **S7_message**
designates whether a variable is used for exchanging messages between state graphs.

Transition

A transition contains conditions which have to be fulfilled for the open-loop control to switch from one state to the next.

Transition priority

If several transitions are assigned to one state, a different priority is assigned to each transition. If the conditions for more than one transition are fulfilled, the transition with the highest priority switches to the next state.

Variable declaration

The variable declaration encompasses the specification of a symbolic name, a data type as well as any initial value and comment.

Index

"

"Current parameters" tab, 4-1

"Instructions" tab, 4-1

)

), 8-6

*

*D, 8-6

*I, 8-6

*R, 8-6

/

/D, 8-6

/I, 8-6

/R, 8-6

+

+, 8-6

+D, 8-6

+I, 8-6

+R, 8-6

=

=, 8-6

==D, 8-6

==I, 8-6

==R, 8-6

A

A, 8-9

A(, 8-9

ABS, 8-6

Absolute or symbolic programming, 5-31 - 5-33

Accessing the help with key commands, 10-8

ACCU operations (STL), 8-5

ACOS, 8-6

Actions, 5-12, 5-25 - 5-30, 8-6

AD, 8-9

Aligning graphics objects, 5-15

AN, 8-9

AN(, 8-9

Any transition, 5-20

Arranging windows, 4-3

Arranging working windows, 4-3, 4-5

ASIN, 8-6

Assigning features, 5-18

ATAN, 8-7

AutomaticMode, 5-9

Automation License Manager, 2-1, 2-3, 2-4

AW, 8-9

B

Basics of programming in S7-HiGraph, 5-1, 5-2

Bit logic operations (STL), 8-1

Block calls (STL), 8-3

Block structure, 1-1, 1-2

BOOL, 7-6

BTD, 8-7

BTI, 8-7

Byte, 7-6

C

CALL, 8-7

Calling the HiGraph FC, 5-54, 5-55

CD, 8-9

Certificate of License, 2-1

CHAR, 7-6

Character (CHAR), 7-6

- Check Block Consistency, 5-62
- CLR, 8-7
- cold restart, 7-3
- Comparison operations (STL), 8-3
- Compatibility with previous HiGraph versions, 5-71, 5-72
- Compilation, 5-52, 5-53
- Compiler settings, 5-53
- Conditions, 5-12, 5-25 - 5-30, 8-6
- Configuration based on the example of a transfer line, 9-1 - 9-6, 9-8, 9-9, 9-11, 9-13, 9-15, 9-17, 9-19, 9-21, 9-23
- Configuring standard diagnostics based on the example of a transfer line, 9-24 - 9-26, 9-29
- Controlling and monitoring variables, 5-62
- Conversion instructions (STL), 8-6
- Converting HiGraph 2.6 / 2.7 programs, 5-69, 5-70
- Copy
 - State, 5-19
 - Transition, 5-23
- COS, 8-7
- Count operations (STL), 8-2
- Counter, 7-6
- Criteria analysis, 6-10, 6-11
- CU, 8-9
- Current parameter
 - Current parameters for messages, 5-44
 - Graph groups: Programming messages, 5-44
 - Input, 5-42
 - Interaction between variable declarations and current parameter assignments, 5-13
 - Messages, 5-44
 - Printing, 5-65
 - Programming the exchange of messages between state graphs, 5-44
 - User interface, 4-1
- Current values, 6-10, 6-11
- CurrentState, 5-9
- Customizing the size of the drawing area, 4-6
- Cut
 - State, 5-19
 - Transition, 5-23
- Cyclic actions, 5-27
- Cyclic execution of a state graph, 7-1, 7-2

D

- D, 8-6
- Data types, 7-6
- Data volume, 5-15
- DATE, 7-6
- Date and time of day (DATE_AND_TIME), 7-6
- DATE_AND_TIME, 7-6

Debugging

- Monitoring the program status, 5-58, 5-59, 5-61
- STEP 7 debugger functions, 5-47
- Viewing reference data, 5-47
- DEC, 8-7
- Declaration properties dialog
 - See variable overview, 5-6
- Declaring variables
 - Columns in the variable detail view, 5-8
 - Declaration sections, 5-7
 - Declaring variables, 5-8
 - Default variables, 5-9
 - Meaning, 5-5
 - Steps in entering the variable declaration, 5-8, 5-44
 - Variable detail view, 5-8
 - variable overview, 4-1
 - Variable overview, 5-6, 5-7
- Default variables, 5-9
- Defining delay times, 5-37, 5-38
- Defining the DB name, 5-53
- Defining the FC name, 5-53
- Defining the monitoring times, 5-36
- Defining waiting times, 5-34
- Delay time, 5-37, 5-38
- Delta download, 5-57
- Detail screen (diagnostics), 6-10, 6-11
- Diagnostics, 6-1, 6-2
- Diagnostics by means of format converter, 6-22 - 6-24
- Diagnostics messages, 6-6, 6-7, 6-9, 6-23
- DINT, 7-6
- Double word (DWORD), 7-6
- Downloading the user program, 5-56, 5-57
- Drawing and positioning tools, 4-6
- Drawing area, 4-6
- DTB, 8-7
- DTR, 8-7
- DWORD, 7-6

E

- EN, 8-7
- Enhanced session memory for sources in V5.2 or Later, 4-4
- Enlarging and reducing the view, 4-5
- Entering the priority, 5-21
- Entry actions, 5-27
- EP, 8-7
- Error messages, 6-6, 6-7, 6-9
- Example of a drilling machine, 3-3 - 3-27
- Example of an application, 3-3 - 3-27
- Exit actions, 5-27
- EXP, 8-7

F

Features, 4-6, 5-18, 5-22
 Fixed point maths (STL), 8-4
 Floating point maths (STL), 8-4
 Font types, 4-5
 Formal parameters, 5-1
 Functions
 S7-HiGraph, 1-4

G

Generating and viewing reference data, 5-48
 Graph
 see state graph, 5-4
 Graph groups, 5-40, 5-42
 Assigning current parameters, 5-42
 Compilation, 5-52, 5-53
 Creating / opening, 5-42
 Defining the run sequence of the instances, 5-42
 Inserting status groups as instances, 5-42
 Printing, 5-65 - 5-68
 Programming messages, 5-43
 Programming with absolute or symbolic
 addresses, 5-31, 5-32, 5-33
 Grid, 4-6

H

HiGraph, 1-1, 1-2, 1-4
 Starting, 5-3
 User interface, 4-1
 HiGraph 2.6 / 2.7, 5-69, 5-70
 HiGraph V4.0 / V4.01, 5-71
 HiGraph V5.0, 5-72
 HiGraphInitValAcquEmitterFC (FC 103), 6-4
 HiGraphUnivEmitterFC (FC 102), 6-4
 Hot restart, 7-3

I

-I, 8-6
 IEC date (DATE), 7-6
 IEC time (TIME), 7-6
 INC, 8-7
 INIT_SD, 5-9, 5-54
 Initial download, 5-56, 5-57
 Initial state, 5-16
 Initial value acquisition, 6-10, 6-11
 Initialization, 7-3, 7-4
 Inserting permanent instructions, 5-25
 Inserting states, 5-17
 Inserting transitions, 5-21

Installation requirements, 2-5
 Installing Automation License Manager, 2-3
 Installing S7-HiGraph, 2-1, 2-5 - 2-7
 Instance, 5-40
 Insert, 5-40
 Programming with instances, 5-1
 Instance concept, 5-1
 Instructions, 5-12, 5-25 - 5-30
 Instructions in STL
 sorted by mnemonics, 8-6
 INT, 7-6
 Integer (INT), 7-6
 Integer, 32 bits (DINT), 7-6
 Interaction between S7-HiGraph - AS - HMI, 6-3
 Interaction between S7-HiGraph, the AS and OP
 (Format Converter), 6-22
 Interaction between variable declarations and current
 parameter assignments, 5-13
 Interaction between variable declarations and
 instructions, 5-12
 Interconnecting incoming
 and outgoing messages, 5-45
 International/German keyboard layout, 10-8
 Introduction, 1-1, 1-2, 1-4
 INVD, 8-7
 INVI, 8-7
 ITB, 8-7
 ITD, 8-7

K

Keyboard control, 10-6, 10-7, 10-8
 Keystrokes for menu commands, 10-5

L

L, 8-7
 LC, 8-7
 License agreements with Automation License
 Manager, 2-1
 License Key, 2-1, 2-4
 License Manager, 2-1
 License types
 Floating License, 2-2
 Single License, 2-2
 Trial License, 2-2
 Upgrade License, 2-2
 Limit position, 6-12 - 6-15
 LN, 8-7
 Load, 5-56, 5-57
 Load and transfer operations (STL), 8-3
 Locations of addresses in the program, 5-48

M

- ManualMode, 5-9
- Memory requirements of the user program, 7-6
- Message acknowledgment, 6-6, 6-7, 6-9
- Message screen (diagnostics), 6-6, 6-7, 6-9
- Message view, 4-1
- Messages, 5-43
- Migration, 5-69, 5-70
- Mnemonics, 5-29, 8-6
- MOD, 8-7
- Monitoring and controlling variables, 5-62
- Monitoring the program status, 5-58, 5-59, 5-61
- Monitoring time, 5-36, 6-6, 6-7, 6-9, 6-23
 - Copy: state, 5-19
 - Timeout, 6-6, 6-7, 6-9, 6-23
- Monitoring times, 5-27
- MonitoringTime, 5-36
- Motion monitoring and control on the motion screen, 6-12 - 6-15
- Motion screen (diagnostics), 6-12 - 6-15
- Moving the cursor in the text editor, 10-6
- Multiple instances of state graphs, 5-1

N

- NEGD, 8-7
- NEGI, 8-7
- NEGR, 8-7
- NOT, 8-7

O

- O, 8-7
- O(, 8-7
- OB 1, 5-54, 5-55
- OD, 8-7
- ON, 8-7
- ON(, 8-8
- ONLINE delta downloads, 5-57
- Opening state graphs, 5-4
- Output of messages to the message screen, 6-6, 6-7, 6-9
- Overview screen (diagnostics), 6-17
- OW, 8-8

P

- Page numbering, 5-65
- Page setup, 5-67
- Permanent instructions, 5-25
- PLC
 - Download to, 5-56, 5-57
- PLC, download to, 5-56, 5-57
- Pointer, 7-6
- POP, 8-8
- POWER ON / OFF, 7-4
 - Reaction to POWER ON / OFF, 7-4
- Preceding cyclic actions, 5-27
- PreviousState, 5-9
- Print settings
 - Global print settings for the application, 5-66
 - Graphical or text-based presentation of objects, 5-66
 - Order of print objects, 5-66
 - Zoom factor, 5-66
- Printing, 5-65, 5-67, 5-68
- ProAgent, 6-1, 6-2
- Process error diagnostics, 6-1 - 6-24
- Program status, 5-58, 5-59, 5-61
- Program structure, 1-1, 1-2
- Programming guidelines for standard diagnostics, 6-18, 6-19
- Programming operating modes, 5-39
- Programming the exchange of messages between state graphs, 5-43
- Programming the state
 - Assigning features, 5-18
 - Defining the delay time, 5-37, 5-38
 - Defining the monitoring time, 5-36
 - Defining waiting times, 5-34
 - Entering instructions, 5-30
 - Insert, 5-17
 - States, 5-16
- Programming the state graph
 - Assignment of function unit and state graph (example of a drilling machine), 3-8
 - Basics of programming, 5-1
 - Create the program structure (example of a drill), 3-6
 - Designing the state graphs (example of a drill), 3-8
 - Determine the state graphs required (example of a drill), 3-6
 - Open, 5-4
- Programming with symbolic addresses, 5-31 - 5-33
- ProTool, 6-1, 6-2
- PUSH, 8-8

Q

Quick positioning to locations in the program, 5-48

R

R, 8-8
 -R, 8-6
 Reaction on startup and hot restart, 7-3, 7-4
 REAL, 7-6
 Real number (REAL), 7-6
 Recording initial and current values on the detail screen, 6-10, 6-11
 Requirements for creating a program, 5-3
 Requirements of standard diagnostics, 6-4
 Return transition, 5-20
 right mouse button, 10-1
 RLD, 8-8
 RLDA, 8-8
 RND, 8-8
 RND-, 8-8
 RND+, 8-8
 Rotate and shift instructions (STL), 8-5
 RRD, 8-8
 RRDA, 8-8
 Rules for handling License Keys, 2-4
 Rules for the structure of state graphs and groups, 5-15
 Run level, 5-21
 see Priority, 5-21
 Run sequence, 5-42
 Running Setup, 2-6

S

S, 8-8
 S7_active, 5-9
 S7_message, 5-44
 S7-HiGraph, 1-1, 1-2
 Functions, 1-4
 installing, 2-5
 uninstalling, 2-7
 S7-HiGraph-specific abbreviations
 in the reference data, 5-50
 SA, 8-8
 Sample program, 3-3 - 3-27
 SAVE, 8-8
 Save format
 V4.x, V5.x, 5-72
 SE, 8-8
 Selecting menu commands from the menu bar using the keyboard, 10-6, 10-7
 Selecting texts with key commands, 10-7

Session memory, 4-3, 4-4
 SET, 8-8
 Setting colors, 4-5
 Setting the compiler parameters, 5-53
 Setting the font type in working windows, 4-5
 Setting up a project, 5-3
 Setting up a STEP 7 project, 5-3
 Setting up headers and footers, 5-68
 Settings, 4-5, 4-6, 5-53, 5-67
 Shortcut menus, 10-1
 SI, 8-8
 SIMATIC time, 7-6
 SIMATIC time (S5TIME), 7-6
 SIN, 8-8
 SLD, 8-8
 SLW, 8-8
 SQR, 8-8
 SQRT, 8-8
 SRD, 8-8
 SRW, 8-8
 SS, 8-8
 SSD, 8-8
 SSI, 8-8
 ST_CurrValue, 5-9
 ST_Expired, 5-9
 ST_ExpiredPrev, 5-9
 ST_Stop, 5-9
 ST_Valid, 5-9
 Standard diagnostics with the help of ProTool / ProAgent, 6-1 - 6-17
 Start parameter INIT_SD, 5-54
 Starting S7-HiGraph, 5-3
 Startup reaction, 7-3, 7-4
 Startup transition, 5-20
 State graph, 5-14, 7-1, 7-2
 Creating / opening, 5-4
 Printing, 5-65 - 5-68
 State name, 5-17
 StateChange, 5-9
 Status, 5-59, 5-61
 Status comment, 5-17
 Status messages, 6-6, 6-7, 6-9
 Status number, 5-17
 STEP 7 debugger functions, 5-47, 5-62 - 5-64
 Steps in creating a program, 5-2
 Steps in entering the message type, 5-44
 Steps in generating diagnostics data, 6-5, 6-24
 Steps in printing, 5-66
 Steps in programming instructions for messages, 5-45
 STL instructions, 8-6
 String, 7-6
 Structure
 Rules, 5-15
 Structure of a state graph, 5-14

SV, 8-8
Symbol table (example of a drilling machine), 3-13
Symbolic programming, 5-31, 5-32, 5-33

T

T, 8-8
TAD, 8-8
TAK, 8-8
TAN, 8-8
TAR, 8-8
TAR1, 8-8
TAR2, 8-8
TAW, 8-8
Template mechanism
 see Instance concept, 5-1
TIME, 7-6
Time operations (STL), 8-2
TIME_OF_DAY, 7-6
Time-of-day (TIME_OF_DAY), 7-6
Timer, 7-6
Toolbar icons, 10-3
Tooltips, 10-1
Transition actions, 5-27
Transitions, 5-20
 Copy, move, delete, 5-23
 Defining the priority, 5-21
 Insert, 5-21
 Transition features, 5-22
 Transition name, 5-22
 Transition properties, 5-22
TRUNC, 8-9

U

UDT_Motion, 6-4, 6-12 - 6-15
UDT_Unit, 6-4
Uninstalling S7-HiGraph, 2-7
Unit overview (diagnostics), 6-17
User interface, 4-1
 Customizing the user interface, 4-5, 4-6

V

variable declaration
 Printing, 5-65
Variable declaration
 Columns in the variable detail view, 5-8
 Declaration sections, 5-7
 Declaring variables, 5-8
 Default variables, 5-9

Interaction between variable declarations and
current parameter assignments, 5-13
Interaction between variable declarations and
instructions, 5-12
Meaning, 5-5
Steps in entering the variable declaration, 5-8
Variable detail view, 5-8
Variable overview, 4-1, 5-6, 5-7
Views, 5-6
Version 4.0/4.01, 5-71
Version V5.0, 5-72
Viewing reference data, 5-47, 5-48
Visualization of units on the overview screen, 6-17

W

Waiting time, 5-34
Waiting times, 5-27
WaitTime, 5-34
WORD, 7-6
Word (WORD), 7-6
Word logic operations (STL), 8-2
Working with state graphs and graph groups, 5-1
WT_CurrValue, 5-9
WT_Expired, 5-9
WT_Valid, 5-9

X

X, 8-9
X(, 8-9
XN, 8-9
XN(, 8-9
XOD, 8-9
XOW, 8-9

Z

Zoom, 4-5
Zooming in and out, 4-5