

SIEMENS

SIMATIC

S7-HiGraph for S7-300/400

Manual

Important Information,
Contents

Product Overview and
Installation **1**

Designing a Program Using
the Example of a Drill **2**

Working with S7-HiGraph **3**

Process Error Diagnosis **4**

STL Instruction Description **5**

Configuration Notes **6**

User Program Run Behavior
in the PLC **7**

Tips and Tricks **8**

Glossary

Index

01/2000
C79000-G7076-C527
Edition 01

Safety Guidelines

This manual contains notices which you should observe to ensure your own personal safety, as well as to protect the product and connected equipment. These notices are highlighted in the manual by a warning triangle and are marked as follows according to the level of danger:



Danger

indicates that death, severe personal injury or substantial property damage will result if proper precautions are not taken.



Warning

indicates that death, severe personal injury or substantial property damage can result if proper precautions are not taken.



Caution!

indicates that minor personal injury or property damage can result if proper precautions are not taken.

Note

draws your attention to particularly important information on the product, handling the product, or to a particular part of the documentation.

Qualified Personnel

Only qualified personnel should be allowed to install and work on this equipment. Qualified persons are defined as persons who are authorized to commission, to ground, and to tag circuits, equipment, and systems in accordance with established safety practices and standards.

Correct Usage

Please note the following:



Warning

This device and its components may only be used for the applications described in the catalog or the technical description, and only in connection with devices or components from other manufacturers which have been 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

SIMATIC®, SIMATIC HMI® and SIMATIC NET® are trademarks of Siemens AG.

Third parties using for their own purposes any other names in this document which refer to trademarks might infringe upon the rights of the trademark owners.

Copyright © Siemens AG 2000 All rights reserved

The 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.

Siemens AG
Bereich Automatisierungs- und Antriebstechnik
Geschäftsgebiet Industrie-Automatisierungssysteme
Postfach 4848, D- 90327 Nuernberg

Siemens Aktiengesellschaft

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 this manual are reviewed regularly and any necessary corrections included in subsequent editions. Suggestions for improvement are welcomed.

© Siemens AG 2000
Technical data subject to change.

C79000-G7076-C527



Important Information

Purpose of the manual

This manual is intended for programmers of HiGraph programs and for persons working in the fields of planning, commissioning and servicing PLCs.

If you are using HiGraph for the first time, we recommend that you familiarize yourself on the basis of the example in Chapter 2. It is an easy method of getting started in programming with HiGraph.

Required knowledge

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

In addition, knowledge of S7 programs is required. This can be looked up in the programming manual. Since HiGraph is based on the STEP 7 basic software, you should also know how to handle the basic software, as described in the STEP 7 User Manual.

Validity of the manual

The manual is valid for the S7-HiGraph programming software, Version 5.0.

Online help

In addition to the manual detailed support in using the software is provided by the online help integrated in the software.

The help system is integrated in the software through several interfaces:

- The **Help** menu contains several menu commands: **Contents** opens the table of contents of the help, **Introduction** provides an overview over HiGraph programming, **Using Help** provides detailed instructions on using the online help.
- The context-sensitive help provides information on the current context, for example on an opened dialog box or on active window. It can be used by clicking on the "Help" command button or by pressing F1.
- The status line provides a further form of context-sensitive help. A brief explanation of the respective menu command is displayed as soon as the cursor is positioned on the menu command.
- A brief explanation is also displayed for the icons in the toolbar as well as for the graphics elements in the state diagrams when the cursor is left briefly on the element.

If you would rather have the information of the online help in printed form, you can also print out individual help topics, books or the entire help.

User response to the documentation

We would appreciate your help in optimizing the documentation for you and future HiGraph users. Should you have any questions or remarks on this *manual* or on the *online help*, please fill out the questionnaire at the end of the manual and send it to the address specified there. Please also enter your personal evaluation there.

Reference to literature /.../

References to further documentation are made by means of literature numbers between slashes, for example /.../. You can use these numbers to ascertain the exact title of the documentation in the literature list at the end of the manual.

SIMATIC Training Center

We offer corresponding courses to help familiarize you with the SIMATIC S7 PLC. Please contact your regional training center or the central training center in D 90327 Nuremberg, Germany. Tel. +49 (911) 895 3154.

SIMATIC Customer Support Hotline

Available worldwide at all times:



Nuremberg SIMATIC BASIC Hotline Local time: Mo.-Fr. 7:00 to 17:00 Telephone: +49 (911) 895-7000 Fax: +49 (911) 895-7002 E-mail: simatic.support@Nbgm.siemens.de GMT: +1:00	Johnson City SIMATIC BASIC Hotline Local time: Mo.-Fr. 7:00 to 17:00 Telephone: +1 423 461-2522 Fax: +1 423 461-2231 E-mail: simatic.hotline@sea.siemens.com GMT: -5:00	Singapore SIMATIC BASIC Hotline Local time: Mo.-Fr. 8:30 to 17:30 Telephone: +65 740-7000 Fax: +65 740-7001 E-mail: simatic.hotline@sae.siemens.com.sg GMT: +8:00
Nuremberg SIMATIC Authorization Hotline Local time: Mo.-Fr. 7:00 to 17:00 Telephone: +49 (911) 895-7000 Fax: +49 (911) 895-7201 E-mail: authorization@Nbgm.siemens.de GMT: +1:00	Nuremberg SIMATIC Premium Hotline (Calls charged, only with SIMATIC Card) Time: Mo.-Fr. 0:00 to 24:00 Telephone: +49 (911) 895-7777 Fax: +49 (911) 895-7001 GMT: +01:00	

SIMATIC Customer Support Online Services

The SIMATIC Customer Support team provides you with comprehensive additional information on SIMATIC products in its online services:

- You can obtain general current information:
 - On the **Internet** at <http://www.ad.siemens.de/simatic>
- Current Product Information leaflets and downloads which you may find useful for your product are available:
 - On the **Internet** at <http://www.ad.siemens.de/simatic-cs>
 - Using the **Bulletin Board System** (BBS) in Nuremberg (*SIMATIC Customer Support Mailbox*) number +49 (911) 895-7100.

To dial the mailbox, use a modem with up to V.34 (28.8Kbps) with the following parameter settings: 8, N, 1, ANSI, or dial via ISDN (x.75, 64 Kbps).

Contents

Important Information

1	Product Overview and Installation	1-1
1.1	Overview of S7-HiGraph.....	1-1
1.2	What Has Changed from V4.01 to V5.0?	1-4
1.3	Installation and Authorization.....	1-6
2	Designing a Program Using the Example of a Drill	2-1
2.1	Welcome to the Example for Getting Started in HiGraph.....	2-1
2.2	Prerequisites.....	2-2
2.3	Automation Task Drilling Machine	2-3
2.4	Steps to Create the "Drilling Machine" Programming Example.....	2-5
	Step 1: Planning the Program Structure	2-6
	Step 2: Designing the State Graphs	2-7
	Step 3: Defining the Plant Signals	2-9
	Step 4: Creating the "HiGr_Exp" Example in the SIMATIC Manager	2-10
	Step 5: Creating a Symbol Table.....	2-11
	Step 6: Creating a State Graph and Starting S7-HiGraph	2-12
	Step 7: Declaring the Variables	2-13
	Step 8: Inserting the States and Transitions	2-14
	Step 9: Entering the Actions and Transition Conditions.....	2-15
	Step 10: Creating a Graph Group and Inserting Instances	2-16
	Step 11: Assigning the Current Parameters.....	2-18
	Step 12: Compiling the Graph Group.....	2-21
	Step 13: Including the HiGraph FC in a STEP 7 Program	2-22
	Step 14: Downloading and Debugging the User Program	2-23

3	Working with S7-HiGraph	3-1
3.1	Structure of a Program Consisting of State Graphs and Graph Groups (Instance Concept)	3-1
3.2	Steps for Creating a Program	3-2
3.3	Setting Up a STEP 7 Project	3-3
3.4	Starting S7-HiGraph and Creating State Graphs	3-4
3.5	Control Interface and Setting the Work Area	3-5
3.5.1	User Interface	3-5
3.5.2	Arranging Working Windows	3-6
3.5.3	Saving and restoring the Window Arrangement	3-7
3.5.4	Setting the Size of the Drawing Area	3-7
3.5.5	Enlarging and Reducing the View	3-7
3.5.6	Setting the Grid	3-8
3.5.7	Displaying and Hiding Instructions or Characteristics	3-8
3.5.8	Setting the Colors and Fonts for the Working Windows	3-8
3.5.9	Displaying Print Page Frames	3-9
3.6	Declaring Variables	3-10
3.6.1	Meaning of the Variable Declaration	3-10
3.6.2	The Variable Declaration Window	3-10
3.6.3	Declaration Sections	3-11
3.6.4	Columns in the Variable Declaration Window	3-11
3.6.5	Steps for Entering the Variable Declaration	3-12
3.6.6	Using Predefined Variables	3-12
3.6.7	Interaction between Variable Declarations and Instructions	3-15
3.6.8	Interaction between Variable Declarations and Current Parameter Assignments	3-16
3.7	Programming the Structure of a State Graph	3-17
3.7.1	Elements of a State Graph	3-17
3.7.2	Rules for the Structure of a State Graph	3-17
3.7.3	Possibilities of Aligning Graphical Objects	3-18
3.7.4	States	3-19
3.7.5	Transitions	3-22
3.7.6	Permanent Instructions	3-26
3.8	Programming Instructions	3-27
3.8.1	Instructions in States and Transitions/Permanent Instructions	3-27
3.8.2	Instruction Types	3-28
3.8.3	Rules for Entering STL Instructions	3-29
3.8.4	Settings for STL Instructions	3-30
3.8.5	Steps for Entering STL Instructions	3-30
3.9	Programming Waiting and Monitoring Times	3-31
3.9.1	Steps for Programming Waiting Times	3-31
3.9.2	Steps for Programming Monitoring Times	3-32
3.10	Programming Operating Modes	3-33
3.10.1	Operating Modes	3-33
3.10.2	Steps for Programming Operating Modes	3-33
3.11	Programming Graph Groups	3-34
3.11.1	Graph Groups	3-34
3.11.2	Steps for Programming Graph Groups	3-35
3.11.3	Programming with Absolute or Symbolic Addresses	3-36
3.12	Programming Messages between State Graphs	3-37
3.12.1	Basics of Exchanging Messages	3-37

3.13	Display Reference Data	3-40
3.14	Saving and Compiling	3-42
3.14.1	Saving State Graphs and Graph Groups	3-42
3.14.2	Compilation of the Program	3-42
3.14.3	Setting the Compilation Parameters	3-43
3.15	Calling and Loading S7-HiGraph FC.....	3-45
3.15.1	Calling the FC from an S7 Program.....	3-45
3.15.2	Requirements for Downloading.....	3-45
3.15.3	Downloading for the First Time.....	3-45
3.15.4	Reloading Changes ONLINE.....	3-46
3.16	Monitoring and Testing the Program.....	3-47
3.16.1	Monitoring the Program Status.....	3-47
3.16.2	Displaying in Program Status	3-47
3.16.3	Prerequisites for Starting the Program Status	3-49
3.16.4	Steps for Displaying the Program Status	3-49
3.16.5	STEP 7 Test Functions	3-50
3.17	Printing	3-52
3.17.1	Printing a Program Documentation.....	3-52
3.17.2	Printing Steps.....	3-52
3.18	Working with Data from OlderS7-HiGraph Versions.....	3-54
3.18.1	Converting Programs from HiGraph 2.6 / 2.7	3-54
3.18.2	Using Programs Created inS7-HiGraph Version V4.0/4.01	3-55
4	Process Error Diagnosis	4-1
4.1	Standard Diagnosis via ProTool/ProAgent.....	4-1
4.1.1	Standard Diagnosis via ProTool/ProAgent.....	4-1
4.1.2	Interaction between S7-HiGraph, the Automation System and the OP	4-2
4.1.3	Prerequisites for Standard Diagnosis.....	4-3
4.1.4	General Procedure for Creating Diagnostic Data (Standard Diagnosis).....	4-3
4.1.5	Displaying Messages in the Message Screen	4-4
4.1.6	Displaying of Initial Values in the Detail Screen	4-5
4.1.7	Displaying and Controlling Movements in the Movement Screen	4-6
4.1.8	Displaying Units in the Overview Screen	4-8
4.2	Diagnosis via Format Converter	4-9
4.2.1	Diagnosis via the Format Converter.....	4-9
4.2.2	Interaction between S7-HiGraph, the Automation System and the OP (Format Converter).....	4-10
4.2.3	Prerequisites for Diagnosis via the Format Converter	4-11
4.2.4	Generating Diagnostic Data for the Format Converter Diagnosis	4-11
5	STL Instruction Description	5-1
5.1	STL Instructions, Sorted by Instruction Families	5-1
5.2	STL Instructions, Sorted by Mnemonics.....	5-7
5.3	Valid Data Types.....	5-10

6	Configuration Notes	6-1
6.1	Introduction	6-1
6.2	Automation Task Transfer Line.....	6-2
6.3	Determining the Functions to be Controlled	6-4
6.4	Determining the State Graphs	6-5
6.5	Formation of Graph Groups.....	6-6
6.6	Specifying the Program Structure.....	6-8
6.7	Creating State Graphs.....	6-9
6.7.1	Overview: State Graphs and Graph Groups for the Drill Unit	6-9
6.7.2	State Graph for Controlling Operation Enables	6-10
6.7.3	State Graphs for Controlling Operating Modes	6-11
6.7.4	State Graph for Coordinating the Drill Unit.....	6-13
6.7.5	Motor State Graph	6-15
6.7.6	Clamp State Graph	6-17
6.7.7	Valve_2E State Graph	6-19
6.7.8	Compiler Settings.....	6-20
6.8	Standard Diagnosis Configuration	6-21
6.8.1	Information on the Supplied Example	6-21
6.8.2	State Graph for Generating the Operating-Mode Signals.....	6-21
6.8.3	State Graphs with Coordination Function	6-22
6.8.4	State Graphs which Realize a Movement	6-22
6.8.5	Diagnostic Configuration in the Graph Groups.....	6-25
7	User Program Run Behavior in the PLC	7-1
7.1	Cyclic Processing of a State in the PLC.....	7-1
7.2	Behavior on Startup and Restart	7-3
7.3	Memory Requirements of the User Program.....	7-6
8	Tips and Tricks	8-1
	Glossary.....	Glossary-1
	Index	Index-1

1 Product Overview and Installation

1.1 Overview of S7-HiGraph

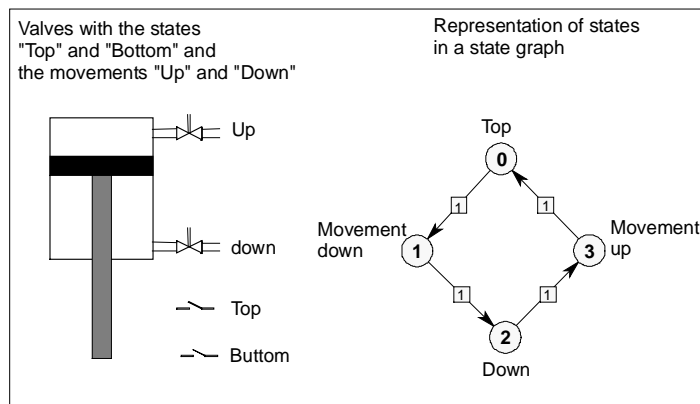
Application

S7-HiGraph extends the functional scope of STEP 7 to include a graphic programming method for state graphs.

With state graphs you can represent processes that you want to control with a SIMATIC programmable controller quickly and easily. The process is divided into individual functional units with a clearly defined functional scope. The behavior of each functional unit is described by means of a state graph.

The decisive advantage is that the program structure orientates itself to the technological objects involved in the process and is therefore easy to record.

The program structure is represented graphically and can be documented in the graphics. This representation is not only suitable for programmers of PLCs, but also for mechanical engineers, commissioning personnel, and service engineers.



Programming language S7-HiGraph

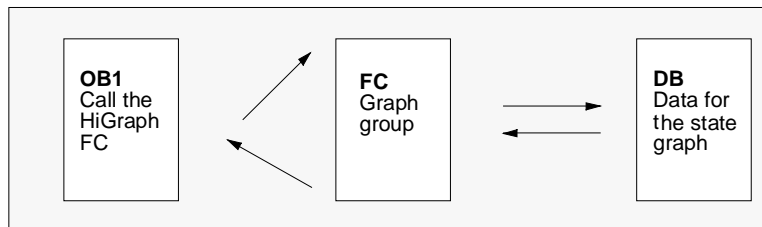
S7-HiGraph programs are structured as follows:

- As a prerequisite for programming, automation tasks are divided into individual functional units. A functional unit can consist of a mechanical component (for example, a valve) or represent a conceptual unit (for example, "operating mode control"). The behavior of each functional unit is then described with the help of a state graph.
- In the state graphs the states are defined which the functional units can assume. Actions can be triggered in states. The execution time of the action can be defined: While entering the state, while in the state or while exiting the state.
- Transitions contain transition conditions which initiate a state change when they are fulfilled. It is also possible to program actions which are executed as soon as a transition is carried out.
- The actions and conditions in the states and transitions are defined using a subset of the STEP 7 programming language STL (Statement List).
- In order to generate a control program for a complete process from the individual state graphs, the state graphs are grouped together in graph groups. These correspond, for example, to the mechanical functional units of a machine. Within a group a state graph can be employed as coordinator.
- The state graphs can communicate with one another by means of messages.

Blocks of the user program

When a graph group is compiled, a function (FC) and a data block (DB) are created. The data block contains the data for the individual state graphs.

In order for the loaded S7-HiGraph program to be able to run in the CPU, the S7-HiGraph FC must be called from a cyclically processed block (for example, OB 1).



Functions

S7-HiGraph offers the following functions:

- Comfortable development environment in Windows 95/98/NT standard.
- Programming of actions in states and transitions in STL.
- Calls of STEP 7 code blocks (FC, SFC, FB, SFB with STL, LAD, FBD, or SCL instructions) from the state graph.
- Programming of wait and monitoring times without using the S7 timers: One wait time and one monitoring time can be programmed for each state. The wait time can be used to delay the processing of a state. The monitoring time is used to monitor the execution time of states.
No S7 timers are required when programming wait and monitoring times. These are only available to a limited extent depending on the respective CPU.
- Monitoring functions can also be programmed for the whole state graph. Specific conditions (for example, emergency off) which arise can be monitored centrally, irrespective of the active state.
- Testing the behavior of the functional units with the ONLINE function "Status" where the currently active state, the last transition, and the previous state are identified, and information on the instructions in states and transitions is displayed.
- Diagnosis of process faults: Error states, monitoring timeouts, and messages can be displayed on an operator interface device.

1.2 What Has Changed from V4.01 to V5.0?

The S7-HiGraph version 5.0 is based on the functionality of earlier HiGraph versions and furthermore offers improved editing and input methods as well as the possibility of process error diagnosis. The following functions have been developed or extended:

Process error diagnosis via ProTool/ProAgent

The process error diagnosis allows rapid recognition, finding and eliminating process errors. The following diagnostic functions are available:

- Outputting of messages if the system enters an error state or if a monitoring time is exceeded
- Determining the addresses causing the error (starting-count criteria analysis)
- Monitoring the movements of the units in the machine/plant as well as troubleshooting by means of tracked manual mode

Improved support during programming

- You can use the menu command **View > Symbolic Addresses/Absolute Addresses** to toggle between absolute and symbolic display of the addresses in the program.
- In order to include symbolic names which are defined in the symbol table into an instruction, select the menu command **Insert > Symbol**. A list of all the symbols is then displayed.
- The points of use of an address in the program can be listed in an overview by using the menu command **Edit > Go to > Point of Application**. If you click on a point of use in the list, the program then jumps to this point.
- Linking of the incoming and outgoing messages in the current parameter window is supported by means of a selection list.
- A change in the variable name in the variable declaration is carried out automatically in the instructions of a state graph and in the current parameter assignment of a graph group.
- Instructions can now be arranged in instruction blocks. This facilitates automatic resetting of signals which were set while in the state.

Extended printing functions

- It is now possible to print out a list of the shared addresses used in S7-HiGraph.
- If you wish to adapt the layout of the state graphs or of the graph groups during drawing to the format of the print page to be used later, you can use the menu command **View > Print Page Frame** to have a frame displayed which shows the dimensions of the pages printed later. You can use the menu command **Options > Align > To Page** to center a marked object or a group of objects exactly on the print page.

Improved editor functions

- Instructions and current parameters are now displayed in a common window. The window is displayed as soon as you double-click on a state or an instance. If you deselect the element, the window is hidden again. You can also keep the window open constantly by selecting the corresponding menu command from the context menu.
- You can use the menu commands **Window > Save Arrangement** or **Window > Restore Arrangement** to save the settings for the windows and to restore them later.
- In order to structure the state graph more clearly, you can use the menu command **View > Details** to display or hide the following elements:
 - Instructions
 - Permanent instructions
 - Characteristics of states and transitions
- Copied or cut-out elements can be placed exactly at a desired position in the graphics by means of an interactive insertion cursor.

1.3 Installation and Authorization

System environment

The S7-HiGraph optional software package V5.0 runs on a programming device/PC with:

- Microsoft Windows 95/98/NT operating system
- STEP 7 Basic Package from Version 4.02.7 onwards or STEP 7 V5.0 from Service Pack 3 onwards (any corrections required for the basic package are supplied).

Hardware

The same requirements exist for S7-HiGraph as for the STEP 7 basic package. The S7-HiGraph optional package also requires at least 10 Mbytes of additional memory capacity, depending on the installation variant selected.

Starting the installation program

S7-HiGraph includes a SETUP program which carries out the installation automatically. Prompts on the screen guide you step by step through the whole installation procedure.

Proceed as follows:

1. Under Windows 95/98/NT start the dialog for installing software by double-clicking on the "Add/Remove Software" icon in the "Control Panel".
2. Click on "Install".
3. Insert the disk and click on "Continue". Windows now automatically searches for the installation program "Setup.exe".
4. Follow the instructions displayed by the installation program step by step.

On authorization

During installation the program checks whether the authorization required to use the S7-HiGraph programming software exists on the hard disk. If no authorization is found, a message is displayed to inform you that the software can only be used with authorization (user license). If you wish to, you can run the authorization program immediately or you can continue the installation and run the authorization at a later stage. In the first case you should insert the authorization diskette when prompted to do so.

Authorization diskette

A copy-protected authorization diskette is included with the scope of supply of the S7-HiGraph programming software. It contains the authorization and the program AUTHORSW required to display, install, and remove the authorization.



Caution

Read the information in the README.WRI file on the authorization diskette. If you do not adhere to these instructions, the authorization may be irretrievably lost.

If the authorization is lost ...

An authorization may be lost, for example, if a hard disk defect occurs and you did not have a chance to remove the authorization from the defective hard disk.

If you lose your authorization, you can fall back on the emergency license. It is also included on the authorization diskette. The emergency license allows you to continue using the software for a limited period of time. In this case, the time remaining until the license expires is displayed on the screen when you start it. Within this time period you should obtain a replacement for the lost authorization. Please contact your local Siemens distributor or sales office.

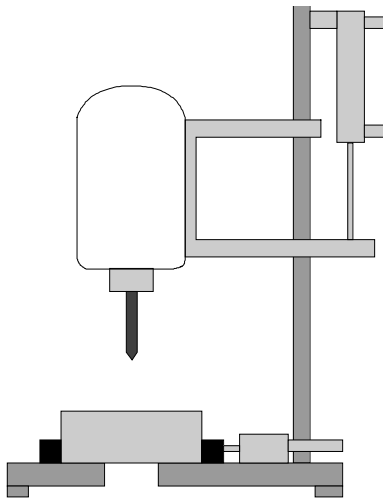
Note

For further information and rules on installing and removing the software please refer to the manual "Programming with STEP 7."

2 Designing a Program Using the Example of a Drill

2.1 Welcome to the Example for Getting Started in HiGraph

This example for getting started shows you in about an hour how to use S7-HiGraph in order to create a program for the automation of the following drilling machine.



You first learn how to plan and structure an S7-HiGraph program efficiently and are then taken step-by-step through all the tasks that you have to carry out in the SIMATIC Manager and in S7-HiGraph in order to

- create the program,
- download it to the CPU and
- debug it.

The correctly programmed example is included in the scope of delivery as the project ZEn03_01_HiGraph_DrillMac. After the installation it is positioned in the directory STEP7\Examples.

2.2 Prerequisites

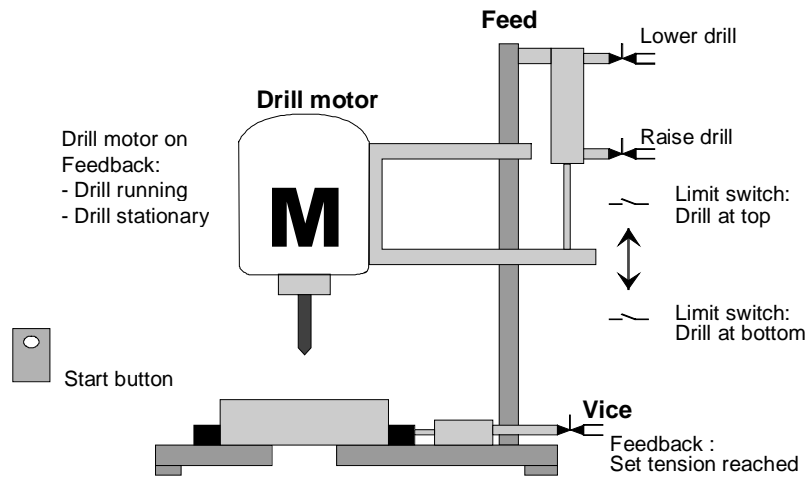
The following hardware and software components are required to program the example program as described:

- A programming device/personal computer on which the STEP 7 basic package and the S7- HiGraph optional package are installed.
- In order to download and debug the example program you require:
 - An automation system with a digital input-output module (8DI+8DO). In this example the S7-300 with CPU 314 is used. However, S7-HiGraph programs can also be executed on an S7-400 automation system.
 - Or the S7-PLCSIM S7-optional package in order to simulate a CPU of the series S7-300 or S7-400.

2.3 Automation Task Drilling Machine

The structure of the drilling machine, shown in the technical diagram, and the sequence of the drilling process, shown in the function diagram, are specified.

Technical diagram: Structure of the drilling machine



Basic state

The basic state of the drilling machine is defined as follows:

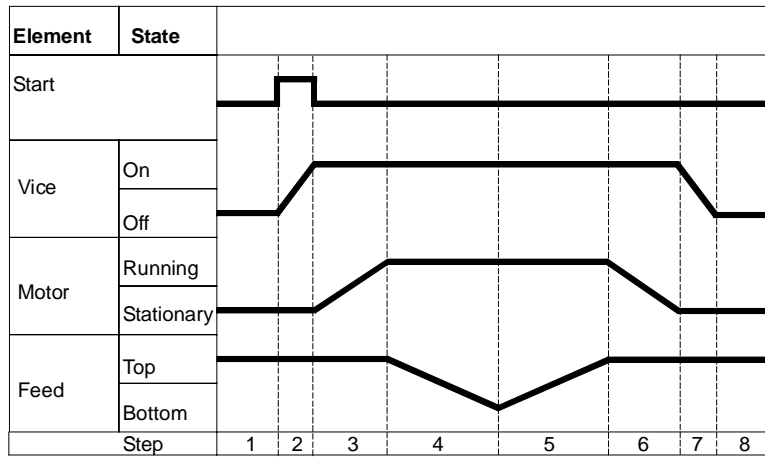
- Drill motor standing still.
- Feed/Drill is in the upper position.
- No workpiece is clamped.

Sequence during drilling

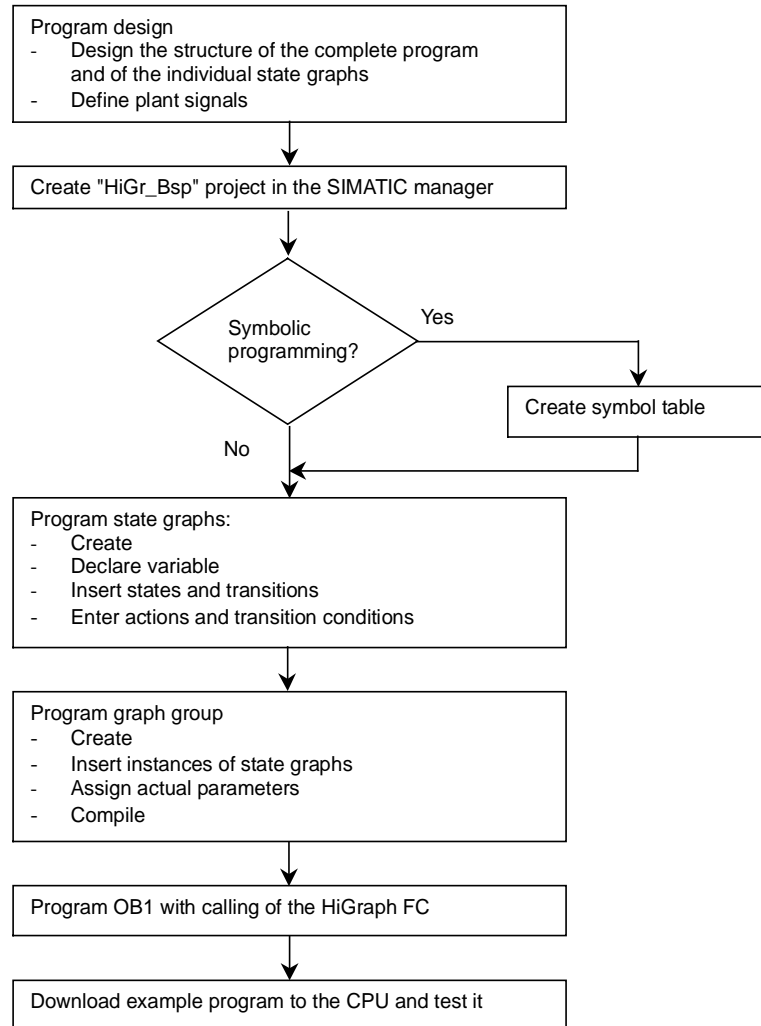
The drilling process is subdivided into the following steps:

1. Insert the workpiece and start the machine via the start pushbutton.
2. Clamp the workpiece (until the set clamping pressure is reached)
3. The drill motor starts up
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. Loosen the workpiece
8. Remove the workpiece

The following function diagram shows the sequence of the drilling process:



2.4 Steps to Create the "Drilling Machine" Programming Example



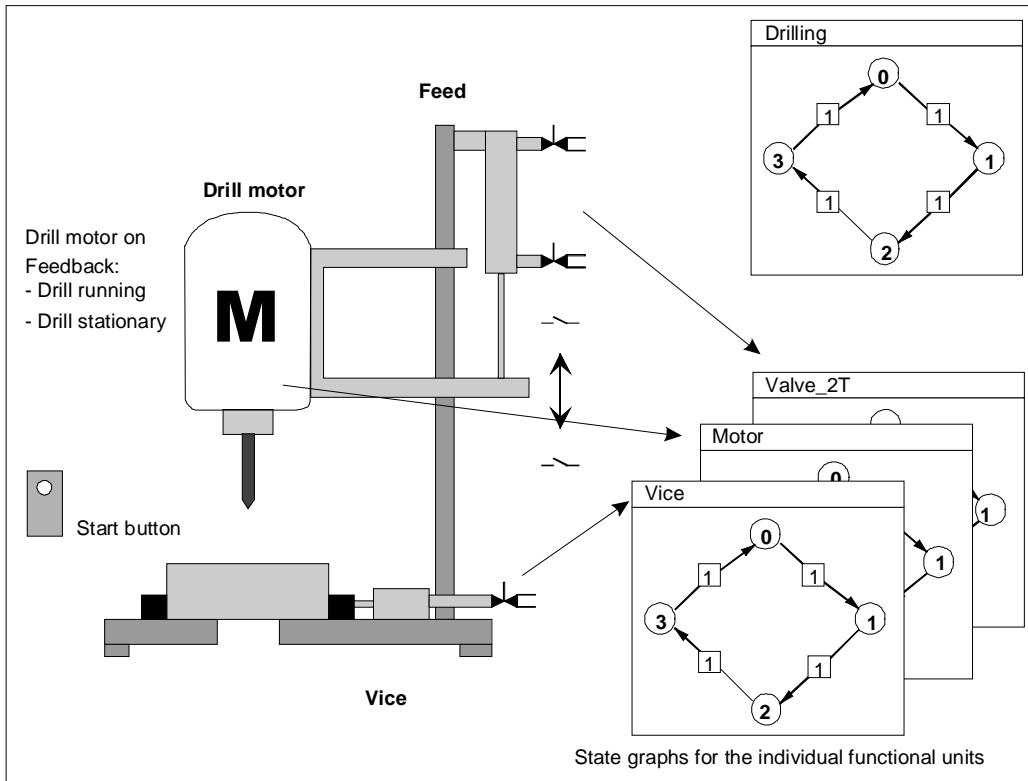
Step 1: Planning the Program Structure

First define which state graphs are required for the drilling machine example. The following rules apply:

- One state graph is required per function unit or task.
Usually one state graph is used for every mechanical component of a process. In addition there are further functions, such as for example, the control of the operating modes or the control of operation enables. These are also mapped to a state graph.
- The state graphs can be structured hierarchically. It is thus possible to insert one or more state graphs which coordinate the other state graphs in a graph group.

The drilling machine can be divided into the functional units "Drill_motor", "Feed" and "Vice". The feed is realized by means of a valve with two limits.

The state graphs "Motor", "Valve_2I" and "Vice" are required to control these functional units. The state graph for coordinating is to be called "Drilling".



Step 2: Designing the State Graphs

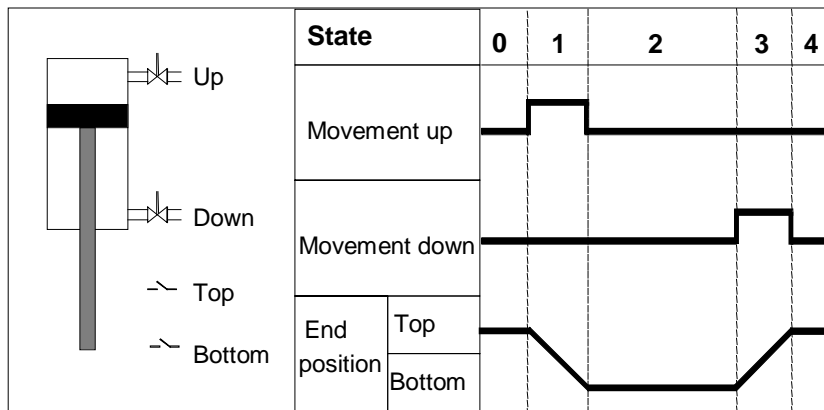
From the function unit to the state graph

In this example for getting started the state graph "Valve_2I" is to be programmed. The further state graphs required already exist in the supplied example project "ZEn03_01_HiGraph_DrilMac".

The functional unit on which the "Valve_2I" state graph is based is a valve unit with two limits. The valve unit consists of the following elements:

- A solenoid valve for the "Up" movement"
- A solenoid valve for the "Down" movement"
- A limit switch for the "Top" limit
- A limit switch for the "Bottom" limit.

It is assumed that the solenoid valves only have to be operated for the movement phase and that the valve remains in the respective limit.



Determining the states

The valve can thus assume the following states:

No.	State	Description
0	Initialization	A state for the initialization is required in every state graph. In the initial state it is possible to check whether the functional unit is in a defined initial position. If required, it can be brought to the initial position.
1	"Top" limit	Drill in the upper limit position
2	"Down" movement	Drill travels downwards
3	"Bottom" limit	Drill in the lower limit position
4	"Up" movement	Drill travels upwards

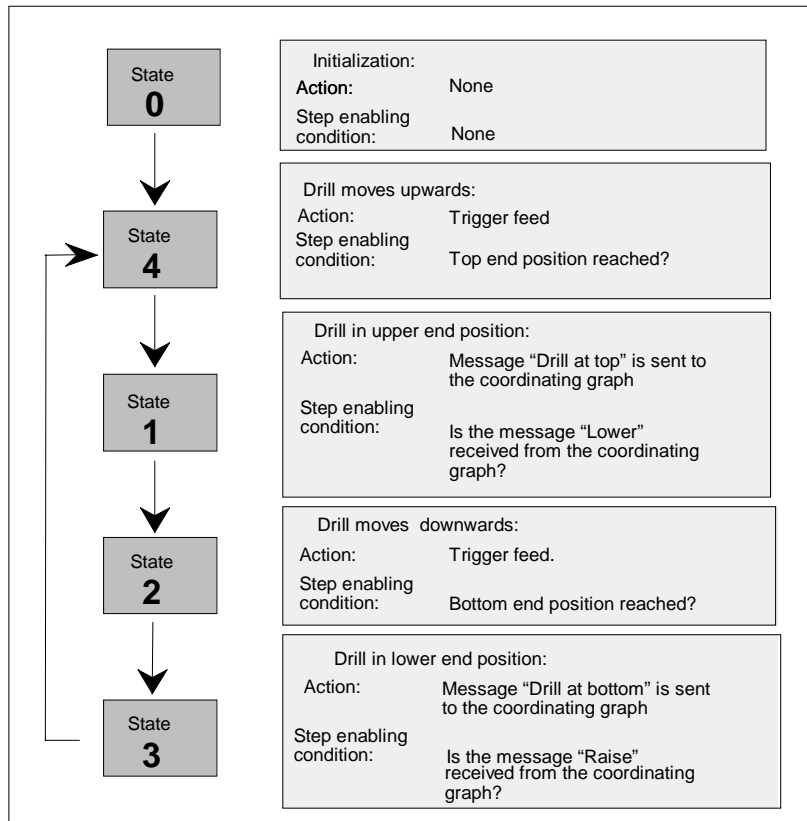
Determining the state change

The "Drilling" state graph determines when the valve unit changes from one state to the next. For this purpose it sends messages to the "Valve_2I".

When the valve reaches the limit, a message is returned to the "Drilling" state graph.

Designing the state graph

The structure of the "Valve_2I" state graph can now be specified on the basis of these specifications.



Step 3: Defining the Plant Signals

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

List the corresponding inputs and outputs of the drilling machine in an assignment table.

If you want to program your program symbolically, enter the desired symbolic designations (for example, I0.4 "Tension_reached") and any remarks required to understand the program better (for example, "Feedback for workpiece set clamping pressure reached") in addition to the absolute inputs and outputs.

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 input/output module of the S7-300 automation system. The input/output module has 8 inputs and 8 outputs. The default values of the input and output addresses of the module on Slot 4 are: I0.0 to I0.7 and O0.0 to O0.7.

Address, absolute	Address, symbolic	Description
Inputs in the program		
I 0.0	Drill_motor_running	Feedback for "Drill running with 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	Tension_reached	Feedback for "Workpiece set clamping pressure reached"
I 0.7	Start_button	Start button of the drilling machine
Outputs in the program		
Q 0.0	Drill_motor_on	Switch drill motor on
Q 0.1	Lower_drill	Use the feed to lower the drill to the lower limit
Q 0.2	Raise_drill	Use the feed to raise the drill to the upper limit
Q 0.3	Clamp_workpiece	Clamp/fix workpiece with set pressure

Step 4: Creating the "HiGr_Exp" Example in the SIMATIC Manager

Creating the project

Prerequisite for programming with S7-HiGraph is a project in which the data of the S7-HiGraph program is saved.

Projects for state graph programming is not different to other projects in STEP 7.

Proceed as follows in order to create a new project in the SIMATIC Manager.

1. Select the **File > "New Project" Wizard**
2. The STEP 7 Wizard which is now started supports you in creating the project. The Wizard prompts you to specify the following data:
 - **Which CPU are you using in your project?**
Specify your CPU. In the supplied example the CPU 314 is used.
 - **Which block do you want to add?**
Select the OB1.
 - **What do you want to call your project?**
Enter the name "HiGr_Exp".

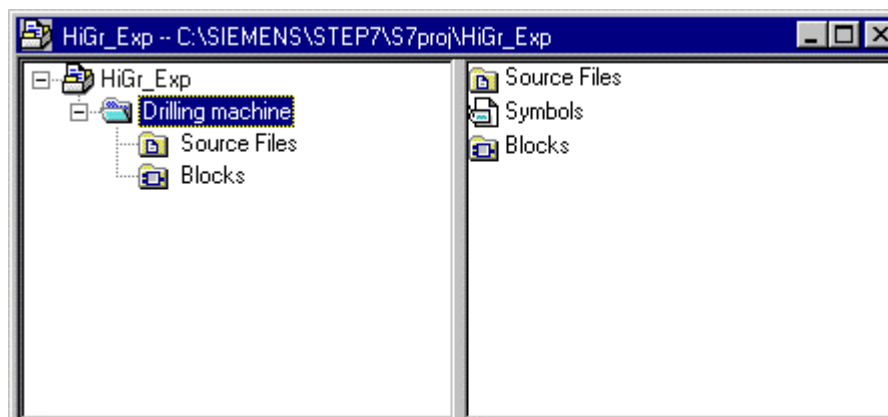
Project structure

The STEP 7 Wizard creates a folder for the station selected by you. This in turn contains a subfolder with the selected CPU. This contains the S7 program with folders for blocks, symbols and sources.

When configuring the structure, an "S7 Program" directory is automatically created for each CPU you have provided for. This directory serves as the folder for the blocks, the sources and the symbols of the user program.

- Call the S7 program "Drilling machine".

The following figure shows the structure of the example project.



Step 5: Creating a Symbol Table

Since you want to program the program with symbolic addresses, it is advisable to create the symbol table as the next step.

- For this purpose open the symbol table in the "Drilling machine" directory by double-clicking on the "Symbols" folder and edit the table as shown in the following figure.

	Symbol	Address	Data Type
1	TIME_TCK	SFC 64	SFC 64
2	WR_USMSG	SFC 52	SFC 52
3	RDSYSST	SFC 51	SFC 51
4	Clamp_Workpiece	Q 0.3	BOOL
5	Raise_Drill	Q 0.2	BOOL
6	Lower_Drill	Q 0.1	BOOL
7	Drill_Motor_On	Q 0.0	BOOL
8	CYCL_EXC	OB 1	OB 1
9	Start_Button	I 0.7	BOOL
10	Tension_Reached	I 0.4	BOOL
11	Drill_at_Top	I 0.3	BOOL
12	Drill_at_Bottom	I 0.2	BOOL
13	Drill_Motor_Stopped	I 0.1	BOOL
14	Drill_Motor_Running	I 0.0	BOOL
15	HiGraphMsgEmitterFC	FC 101	FC 101
16	GG_Drilling	FC 1	FC 1
17	HiGraphErrEmitterFB	FB 20	FB 20
18	DB_GG_Drilling	DB 1	DB 1

The HiGraphErrEmitterFB (FB 20) and HiGraphMsgEmitterFC (FC 101) blocks as well as the system function blocks SFC 51, SFC 52 and SFC 64 are required for diagnostic functions. The blocks are contained in the Standard library or in the HiGraph library.

Step 6: Creating a State Graph and Starting S7-HiGraph

Creating a state graph

Only the state graph "Valve_2I" is to be programmed in this introductory example. The further state graphs required already exist in the supplied example project "ZEn03_01_HIGRAPH_DrillMac".

State graphs are saved in the "Sources" folder of the S7 program.

Proceed as follows:


1. Open the "Sources" folder in the S7 program in the SIMATIC Manager.
2. Select the **Insert > S7 Software > State Graph** menu command.
3. Name the created state graph "Valve_2I".

Starting S7-HiGraph

The S7-HiGraph is started by double-clicking on the "Valve_2I" state graph in the "Sources" folder. "Valve_2I" is opened and already contains the first state (initial state) and the first transition which leads to this state.

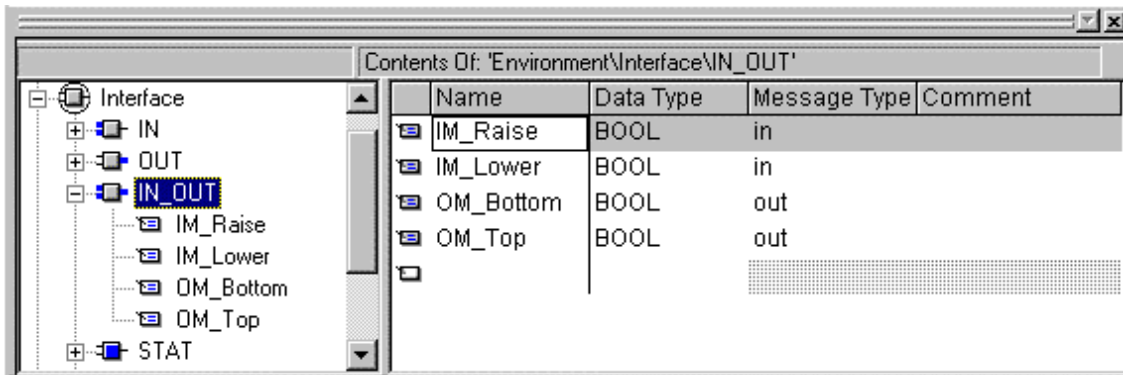
Step 7: Declaring the Variables

Now define the variables which the state graph will use.

1. Open the variable declaration window by using the command button .
2. The declaration sections are shown in the left-hand partial window. They contain predefined variables, which S7-HiGraph enters automatically into the declaration when creating a state graph. Enter the following additional variables. To do so double-click in the left-hand partial window on the desired declaration section and enter the variable name as well as the data type and the message type in the right-hand partial window.



Declaration section	Name	Data type	Message
IN	Top	BOOL	
	Bottom	BOOL	
OUT	Up	BOOL	
	Down	BOOL	
IN_OUT	IM_raise	BOOL	In
	IM_lower	BOOL	In
	OM_top	BOOL	Out
	OM_bottom	BOOL	Out

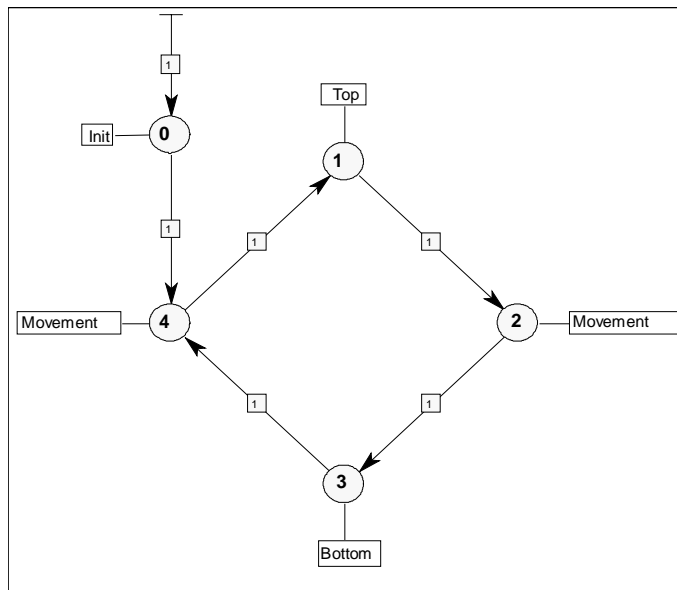
The following figure shows the filled-out variable declaration window. The IN_OUT declaration section is selected.



Step 8: Inserting the States and Transitions

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

1. Select the **Insert > State** menu command  and insert the states 1 - 4.
2. Use the **Options > Align** menu command to position them exactly.
3. Select the **Insert > Transition** menu command  and interconnect the states. 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).



Entering state names


Now enter a name for each state in order to improve the structural clarity:

1. For this purpose select the state and then select the **Edit > Object Properties** menu command. The command can also be called up by using the right-hand mouse button.
2. Enter a name in the "Name" input field.

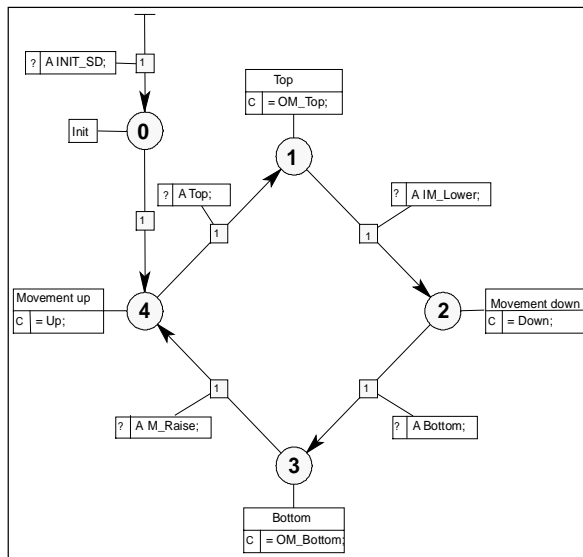
The name is displayed in a box next to the state. Use the mouse to drag the box to a suitable position on the drawing area.

Step 9: Entering the Actions and Transition Conditions


The following figure shows the actions and transition conditions which you have to program. Proceed as follows:

1. First select state 4.
2. Use the **View > Instructions/Current parameters**  menu command to have the input windows for instructions displayed.
3. Select the "Cyclic actions" instruction type in the left-hand partial window.
4. Press the right-hand mouse button and select the "Insert" menu command. A new instruction line is inserted.
5. Select the new instruction and enter =UP; in the right-hand partial window. Always complete instructions with a semicolon.
6. Now click all further states consecutively and enter the corresponding instructions.

Please note, that the address Down requires the symbol ID #Down in the instruction for state 2, as STEP 7 uses Down as the key word for an output byte.
7. Then select the transition from state 4 to state 1.
8. Select the "Conditions" instruction type in the left-hand partial window.
9. Press the right-hand mouse button and select the "Insert" menu command. A new condition is inserted.
10. Enter the condition U Top;. Here again always end the line with a semicolon.
11. Use the same procedure for all other transitions.



Saving

Now save the state graph by selecting the **File > Save** menu command .

Step 10: Creating a Graph Group and Inserting Instances

Copying further state graphs

You have now created the "Valve_2I" state graph successfully. In the next step copy the remaining state graphs needed into your S7 program.

- Change to the SIMATIC Manager and copy the state graphs "Motor", "Vice" and "Drill" from the program "ZEn03_01_HIGRAPH_DrillMac" to the "Sources" folder of your program.

Creating a graph group

In a graph group you define the sequence in which the state graphs are to be executed cyclically during the program execution.

Graph groups are created in the same file as the state graphs. Proceed as follows:

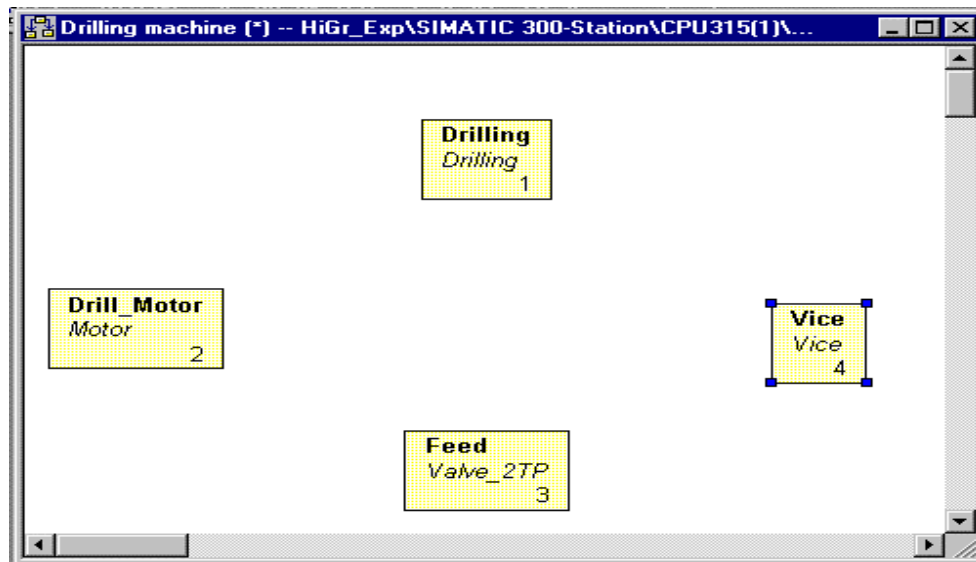
1. Open the "Sources" folder in the S7 program in the SIMATIC Manager.
2. Select the **Insert > S7 Software > Graph Group** menu command.
3. Name the created graph group "Drilling machine" and open it by double-clicking on it.

Inserting instances

After you have opened the "Drilling machine" graph group an empty drawing area is displayed into which you can insert the instances of the "Valve_2I", "Motor" and "Vice" state graphs.

1. Select the **Insert > Instance** menu command.
2. In the subsequent dialog box select the "Valve_2I" state graph.
3. Position the instance on the drawing area.
4. Repeat the process until you have inserted the instances of all four state graphs.
5. Now assign explanatory names to the instances by calling up the "Instance Properties" dialog box with the **Edit > Object Properties** menu command. Enter the following names in the "Name" input field.

Call the instance of the state graph...	By the name...
Valve_2I	Feed
Motor	Drill_motor
Vice	Vice
Drilling	Drilling

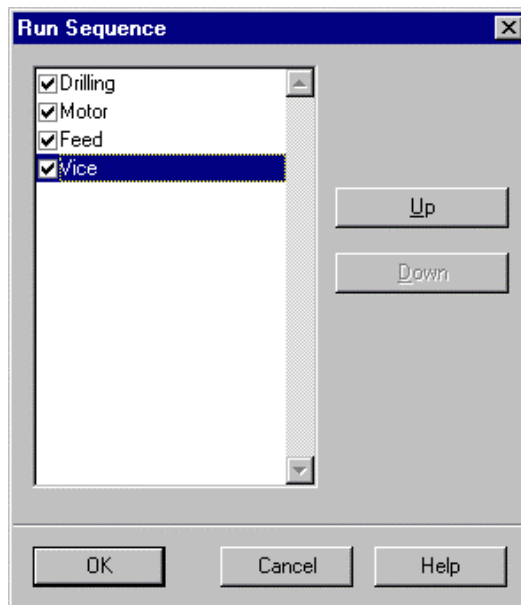


Specifying the run sequence

The instances should be executed in the following sequence:


1. Drilling
2. Drill_motor
3. Feed
4. Vice

Assign the correct position in the run sequence to the instances by selecting the **Edit > Run Sequence** menu command.



Step 11: Assigning the Current Parameters

In the graph group you assign current parameters to the formal parameters of the instances. Enter the current parameters as follows:

1. If the input window for the current parameters is not displayed, use the **View > Instructions/Current parameters** menu command  to have it displayed.
2. Select the "Feed" instance and enter the current parameters listed below (displayed in bold).
3. Use the same procedure for the "Drill_motor", "Vice" and "Drill" instances.

The following tables list the current parameters which have to be assigned to the instances.

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	Raise_drill	
	Down	BOOL	Lower_drill	
IN_OUT	IM_raise	BOOL		In
	IM_lower	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

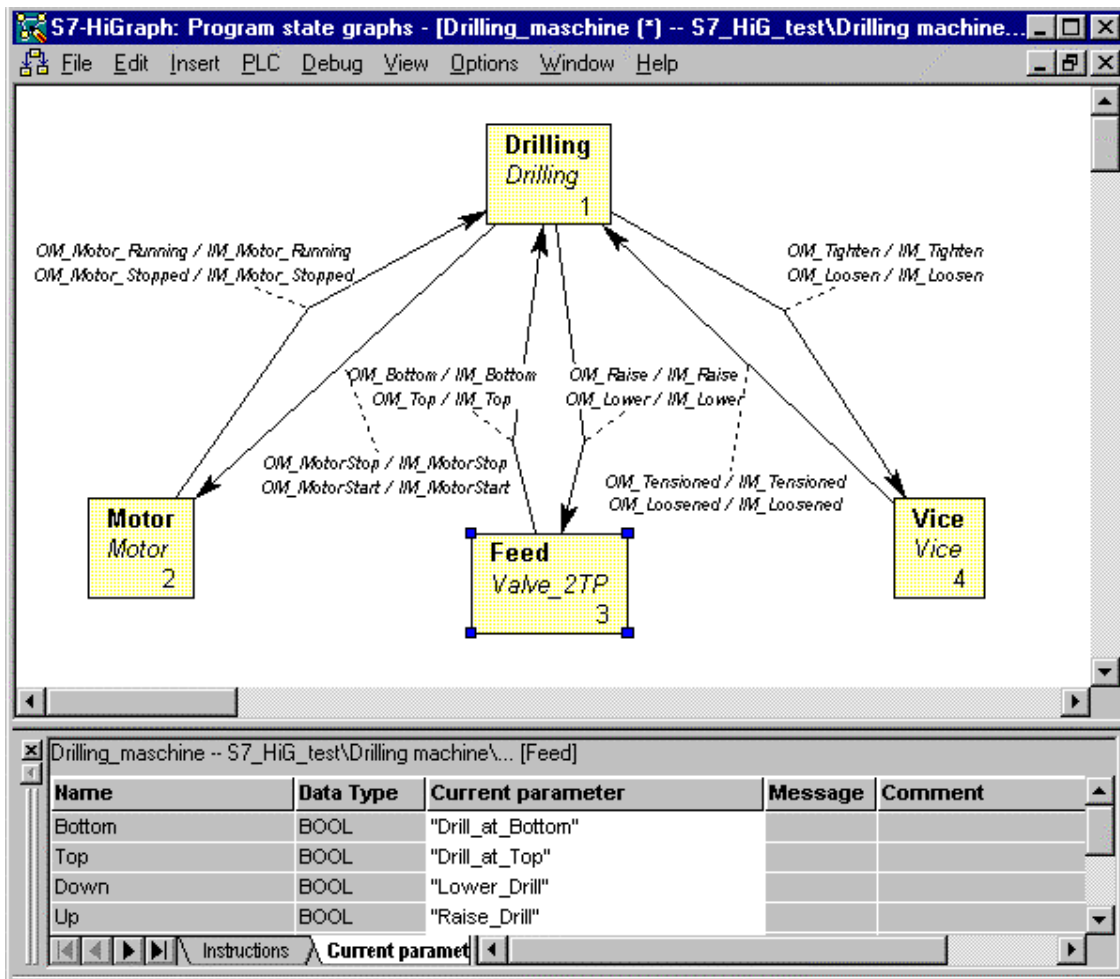
Current parameters of the "Vice" instance

Area	Name	Data type	Current parameter	Message
IN	TensionReached	BOOL	Tension_Reached	
OUT	Clamp	BOOL	Clamp_workpiece	
IN_OUT	IM_Tighten	BOOL		In
	IM_Loosen	BOOL		In
	OM_Tensioned	BOOL	Drilling.IM_Tensioned	Out
	OM_Loosened	BOOL	Drilling.IM_Loosened	Out

Current parameters of the "Drill" instance

Area	Name	Data type	Current parameter	Message
IN	Start	BOOL	Start_button	
IN_OUT	OM_motor_start	BOOL	Motor.IM_Motor_Start	Out
	OM_motor_stop	BOOL	Motor.IM_Motor_Stop	Out
	IM_motor_running	BOOL		In
	IM_motor_stopped	BOOL		In
	OM_lower	BOOL	Feed.IM_lower	Out
	OM_raise	BOOL	Feed.IM_raise	Out
	IM_bottom	BOOL		In
	IM_top	BOOL		In
	OM_Tighten	BOOL	Vice.IM_Tighten	Out
	OM_Loosen	BOOL	Vice.IM_Loosen	Out
	IM_Tensioned	BOOL		In
	IM_Loosened	BOOL		In

After the current parameters have been entered the graph group has the following structure:



Step 12: Compiling the Graph Group

Specifying the FC and DB

Enter the name of the blocks to be generated (FC and DB) in the "Compile" tab card (**Options > Settings for Graph Groups/State Graphs** menu command). In this example use the names FC1 and DB1 or the symbolic designators DB_GG_Drillingmachine and GG_Drillingmachine.

Specifying the compilation options

Further compilation options are offered in the "Compile" tab card. Activate the option "Cyclic actions with RLO = 0". The remaining options do not have to be changed.

Compiling a graph group

The graph group is compiled with the **File > Compile** menu command .

Step 13: Including the HiGraph FC in a STEP 7 Program

Calling the FC in the OB1

To process the S7-HiGraph program for the drilling machine in the automation system, it is called from the organization block OB1. Program the OB 1 in the LAD/STL/SFC editor of the STEP 7 basic package. The function (FC) generated by S7-HiGraph has a parameter "INIT_SD". This parameter is to be supplied in such a manner that signal "1" applies when the control system is activated and Signal "0" applies during the subsequent cycles. This initializes the state graphs in the graph group. The signal can be generated by means of the OB 1 start info (variable #OB1_SCAN_1) and saved in a temporary variable of the OB 1.

Please note that in addition, a variable "startup", data type BOOL has to be in the variable declaration of the OB.

The screenshot shows the Siemens STEP 7 LAD/STL/FBD editor. The title bar indicates the project is '[OB1 -- ZEn03_01_HiGraph_DrillMac\Drilling machine]'. The menu bar includes File, Edit, Insert, PLC, Debug, View, Options, Window, and Help. A table at the top lists the OB1 parameters:

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

Below the table, the editor shows two networks:

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

```

The status bar at the bottom shows 'Press F1 for help.', 'Offline', 'Abs', 'Nw 1 Ln 8', 'INS', and 'MOD'.


Compiling the OB 1

The OB 1 is compiled with the **File > Compile** menu command .

Step 14: Downloading and Debugging the User Program


You have to download the complete user program "Example" (OB 1, FC, DB) to the CPU of the automation system by means of the SIMATIC Manager.

Proceed as follows:


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

Debugging the user program

Proceed as follows in order to debug the S7-HiGraph program:

1. Set the CPU to RUN.
2. Open the graph group and select the **Debug > Monitor** menu command . Information on processing the graph group is now displayed. The current state of each instance is displayed.
3. Now mark one or several instances and select the **Edit > Open Object** menu command.

The instances are opened ONLINE, the following information is displayed:

- 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 with detailed status information is displayed for the transition outgoing from the active state with the highest priority.
4. The monitoring mode is terminated by deactivating the **Debug > Monitor** menu command .

3 Working with S7-HiGraph

3.1 Structure of a Program Consisting of State Graphs and Graph Groups (Instance Concept)

Function of state graphs

State graphs are program components which can be used several times. The state graphs which you have created for a certain functional unit can be used again 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 in one or more graph groups and thus call them.

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

Changes to state graphs can be carried out centrally: The changes carried out in a state graph act in all the instances of this state graph.

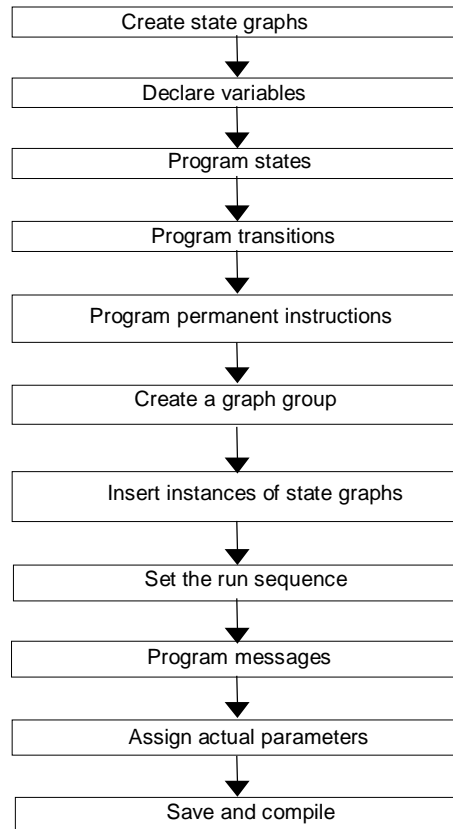
Function of graph groups

In a graph group you define an ordered sequence of calls of state graphs which is executed cyclically in the run sequence when the program is executed.

Declare all the signals used in a state graph as formal parameters so that you can use all the state graphs several times without having to adapt the addresses contained in them. For every call of a state graph define the current parameters of the respective state-graph interface in the graph group.

3.2 Steps for Creating a Program

The flowchart provides an overview of the steps needed to create an S7-HiGraph program. The individual steps are described in the following chapters.



3.3 Setting Up a STEP 7 Project

The following prerequisites must be created in the SIMATIC Manager before you begin to program with S7-HiGraph.

Creating a project

1. Select the **File > New > Project** menu command in the SIMATIC Manager.
2. Enter a name for the project and open it.

Creating a symbol table

If you want to program with symbolic addresses, it is advisable to create the symbol table before programming.

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

Note

From Version 4 on STEP 7 provides a Wizard which helps you in building a complete project structure. In order to start the Wizard use the **File > Wizard "New Project"** menu command.

3.4 Starting S7-HiGraph and Creating State Graphs

Starting 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 task bar in Windows 95/98/NT (entry under "Simatic/STEP 7").

Starting from the SIMATIC Manager

You can also start S7-HiGraph in the SIMATIC Manager by positioning the cursor on a graph group or a state graph in the "Sources" folder and double-clicking on it.

Creating and opening state graphs in S7-HiGraph

1. In order to create a state graph select the **File > New State Graph** menu command. In order to open an existing state graph select **File > Open**.
2. In the subsequent dialog box select the "Sources" folder of your S7 program.
3. Select the "state graph" type in the "Object type" selection field.
4. In order to create a state graph enter the desired name in the "File name" input field and confirm your input with "OK".
In order to open an existing state graph select the desired name and confirm your input with "OK".

Creating and opening state graphs in the SIMATIC Manager

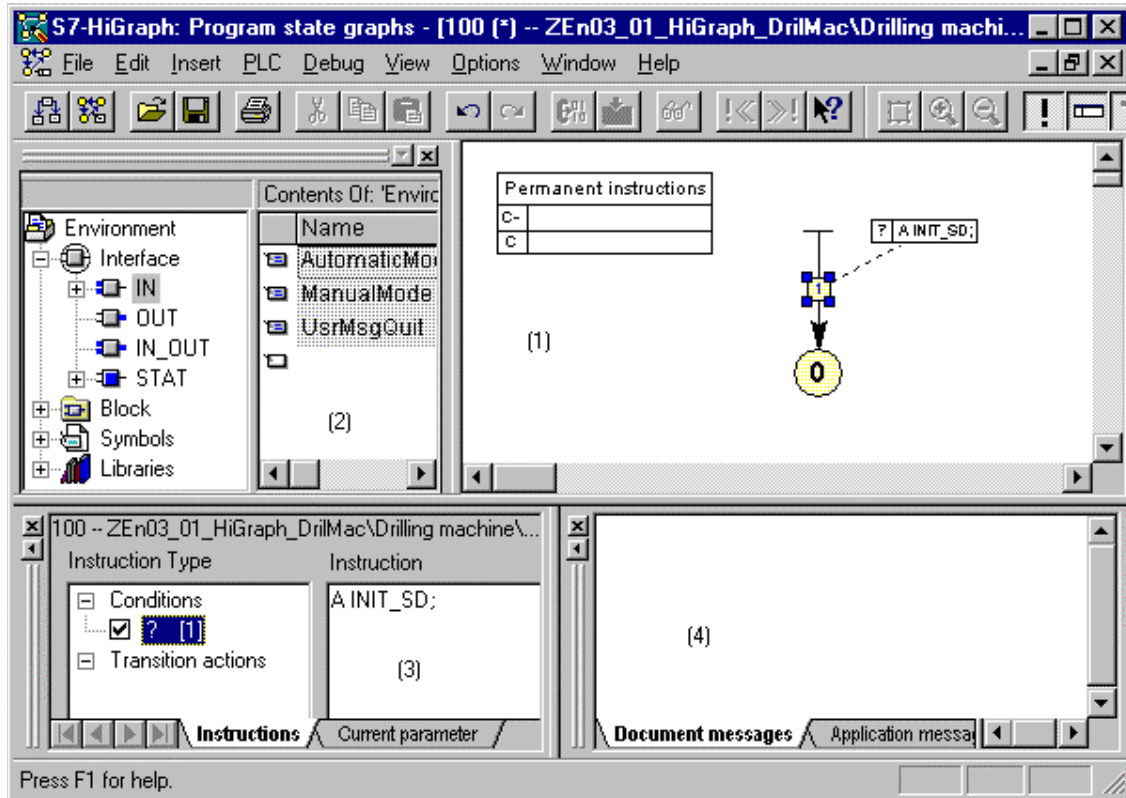
- Open the "Sources" folder in your S7 program.
- Existing state graphs in this folder can be opened by double-clicking on them.
- In order to create a state graph select the **Insert > S7 Software > State Graph** menu command.

A new state graph contains an initial state and a startup transition as well as the instruction table for permanent instructions.

3.5 Control Interface and Setting the Work Area

3.5.1 User Interface

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



Function of the windows

The S7-HiGraph user interface consists of various windows which you can hide or display as required. In order to use the available screen area optimally, you can use the mouse to change the size of the windows or to move them.

The individual windows have the following functions:

Window	Name/Function
(1)	The diagrams of the state graphs and graph groups are created in the working window.
(2)	Enter the variable declaration of the processed state graph in the variable declaration window. It can be displayed by using the View > Variables menu command.

Window	Name/Function
(3)	<p>The View > Instructions/Current parameters menu command is used to have the Instructions/Current Parameters window displayed. It consists of two tab cards:</p> <ul style="list-style-type: none"> • The "Instructions" tab card is active by default when you edit a state graph. Here you program the contents of states, transitions and permanent instructions. • The "Current parameters" tab card is active by default when you edit a graph group. Here you define the current parameters of instances.
(4)	<p>Errors and warnings occurring are output in the "Messages" window. Here you can choose between two register tabs:</p> <ul style="list-style-type: none"> • The "Document Messages" tab card shows syntax errors which were found in the state graph or the graph group currently opened. It is displayed automatically as soon as an error or an inconsistency occurs in the program. • Error messages and warnings arising during compiling are output in the "Application Messages" tab card. The window is displayed after every compilation run. The messages can refer to errors in the graph group or to state graphs instanced in it. <p>Use the View > Messages menu command to have the window displayed when required.</p>

3.5.2 Arranging Working Windows

You can change the positioning of the windows which exists when S7-HiGraph is opened and adapt it to your personal requirements.

The following functions are available:

- You can use the **View** menu to display or hide the output window for errors and warnings, the variable declaration window and the instruction window.
- The instruction window can also be opened by double-clicking on a state or a transition. In this case it is hidden again as soon as you click on another element in the working window. However, you can also keep the instruction window open by selecting the **Remain open** menu command from the pop-up menu.
- In order to move the output window for errors and warnings, the instruction window or the variable declaration window, click on the inner window edge and drag the window to the desired position. The partial windows can only be placed at the outer edge of the working surface.
- 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.
- The usual Windows commands can be used to minimize, maximize or close the windows.
- The status bar and toolbars can also be hidden. To do so use the **View > Status bar** or **View > Toolbars** menu commands.

3.5.3 Saving and Restoring the Window Arrangement

S7-HiGraph offers the possibility of saving the current arrangement of the windows and of restoring it at a later time. You can save one arrangement each for the state-graph and graph-group windows.

What is saved?

The following information is saved when you save the window arrangement:

- Size and position of the working window
- Zoom factor
- Variable declaration window displayed or not

Saving the window arrangement

In order to save the arrangement of the current window type select the **Window > Save Arrangement** menu command.

Restoring the window arrangement

In order to restore the saved arrangement select the **Window > Restore Arrangement** menu command.

3.5.4 Setting the Size of the Drawing Area

The drawing area is the area on which the objects can be positioned. To set the size of this area:

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

3.5.5 Enlarging and Reducing the View

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

Select one of the following menu commands:

View > Zoom	> Zoom In	In order to increase the view step-by-step.
	> Zoom Out	In order to decrease the view step-by-step.
	> Normal Size	In order to restore the specified normal size.
	> Area Used	In order to select a zoom factor which displays all the objects in the working window.

3.5.6 Setting the Grid

- The drawing area is filled with a grid when you select the **View > Grid Points** menu command.
- In order to set the grid size select the **Options > Settings for Graph Groups / State Graphs** menu command and enter the desired values in the "Graphics" register tab.

3.5.7 Displaying and Hiding Instructions or Characteristics

You can display or hide the following elements in order to optimize the graphics structure of a state graph:

- Instructions
- Permanent instructions
- Characteristics of states and transitions

To do so, use the **View > Details** menu command.

3.5.8 Setting the Colors and Fonts for the Working Windows

You can set the font of the texts as well as the colors of the elements in the working windows.

Colors can be set for the entire application:

1. Select the **Options > Application Settings** menu command.
2. Select the desired colors for the various elements in the "Colors" register tab.

The font and size of characters can be set individually for each state graph:

1. Select the **Options > Settings for Graph Groups / State Graphs** menu command.
2. Select the desired fonts for the various elements in the "Fonts" register tab.

3.5.9 Displaying Print Page Frames

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

The print page frames display the dimensions of the pages as they are printed later.

The current printer settings taken into consideration are:

- Paper size
- Portrait/Landscape
- Zoom factor

Have the print page frames displayed by using the **View > Print Page Frame** menu command. Use the **Options > Align > To 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 have print page frames displayed if you have selected the print setting "Zoom on one page".

3.6 Declaring Variables

3.6.1 Meaning of the Variable Declaration

Define all the parameters used in a state graph as variables (formal parameters) so that you can use the state graphs several times without having to adapt the parameters contained in them.

Specify these variables in the variable declaration. In addition you specify those variables here which are used to exchange messages.

The variable declaration has the following effects:

- The declaration reserves sufficient memory in the data block.
- The specification of input, output and in/out parameters defines the "interface" of the state graph.
- By assigning a "Message type" you can specify variables which are used to exchange messages between state graphs.

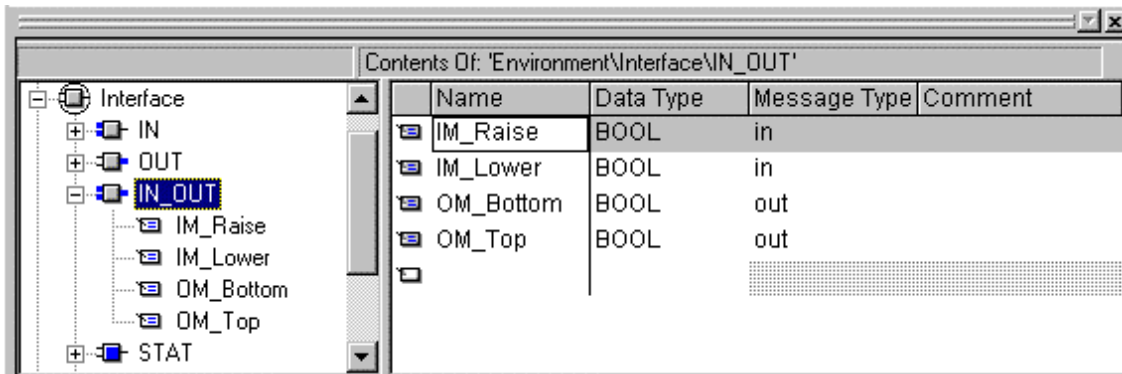
3.6.2 The Variable Declaration Window

The standard position of the variable declaration window is the upper section of the working area. It can be displayed and hidden by using the **View > Variables** menu command. Here you declare the variables which you want to use.

The variable declaration window is divided into two sections:

- The hierarchy window (left-hand partial window) displays the declaration sections (IN, OUT, IN_OUT, STAT). In addition the following elements from the current project are displayed:
 - Symbols contained in the symbol table
 - Blocks contained in the block folder

The detail window (right-hand partial window) contains columns for the name, data type, address, message type and comment on the variables.



3.6.3 Declaration Sections

The variable declaration is structured into the following declaration sections:

Declaration section	Meaning
IN	Contains the input parameters of the state graph and the predefined "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 which you want to use to exchange messages have to be declared here.
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"> • Variables predefined by S7-HiGraph. These cannot be edited. • Local static variables defined by you.

3.6.4 Columns in the Variable Declaration Window

The columns in the variable declaration window have the following meaning:

Column	Meaning	Possible values	Default
Name	Symbolic name of the variables	The following rules apply for variable names: <ul style="list-style-type: none"> • The valid characters are letters, numerals and the underline (_). • A name always begins with a letter or an underline. • A name may not end with an underline. • Two consecutive underlines are not permitted. • Key words are not allowed 	-
Data type	Data type of the variables	BOOL, INT, WORD, etc. (selection is offered)	BOOL
Message type	Messages are used to coordinate state graphs with each other. Messages must be declared in the IN_OUT declaration section. The "Message type" column is therefore only displayed in this section.	IN for incoming messages, OUT for outgoing messages.	-
Comment	Comment used to document the variables	Can be selected freely	-
Initial value	The initial value becomes the current value for the variable when the DB is saved for the first time unless you explicitly assign a current value.	The value has to be compatible with the data type.	A default value will be entered depending on the data type.

3.6.5 Steps for Entering the Variable Declaration

1. Select the desired declaration section in the left-hand section of the variable declaration window.
2. Enter the variable name in the "Name" column. You can either enter the name in the last free line or select the "New Declaration Line" menu command in the pop-up menu in order to insert a free line at any point.
3. Press [ENTER]. This confirms your input and inserts a further empty line into the variable declaration.
4. A further variable name can now be entered in the new line.
5. BOOL is entered automatically as the "Data type". If you want to specify a different data type for the variable, click on the arrow in the "Data type" column and select a data type from the list displayed.

3.6.6 Using Predefined Variables

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

These variables are entered automatically in the variable declaration when a state graph is created. The name and data type cannot be changed and the variables themselves cannot be deleted.

The following predefined variables are available:

<u>Predefined variables</u>	<u>Meaning</u>	<u>Decl. section</u>	<u>Data type</u>	<u>Assigned by user</u>	<u>Name in HiGraph V2.7</u>
ManualMode	Input variable used to set the Manual operating mode. If this variable carries the signal 1, only the transitions with the attribute "Manual" are checked. The variable may not carry signal 1 at the same time as AutomaticMode.	IN	BOOL	X	BA_MANUAL
AutomaticMode	Input variable used to set the Automatic operating mode. If this variable carries the signal 1, only the transitions with the attribute "Auto" are checked. The variable may not carry signal 1 at the same time as ManualMode.	IN	BOOL	X	BA_AUTO
INIT_SD	The variable INIT_SD serves as the startup parameter. If the variable carries the signal 1, initializing is signaled to the state graph.	STAT	BOOL	X	STARTUP
CurrentState	Current state This variable can be queried in conditions. It contains the number of the current state. *	STAT	WORD		CURRENT_STATE
PreviousState	Previous state This variable can be queried in conditions. It contains the number of the previous active state. *	STAT	WORD		PREVIOUS_STATE
StateChange	State change This variable can be queried in conditions. It carries the signal 1 in those cycles in which a state change takes place. In all other cycles it carries the signal 0.*	STAT	BOOL		STATE_CHANGE
ST_Expired	Monitoring time expired This variable can be queried in conditions. *	STAT	BOOL		ST_ERROR
ST_ExpiredPrev	Monitoring time of the last state has expired. This variable can be queried in conditions.	STAT	BOOL		ST_ERROR_PREV

<u>Predefined variables</u>	<u>Meaning</u>	<u>Decl. section</u>	<u>Data type</u>	<u>Assigned by user</u>	<u>Name in HiGraph V2.7</u>
ST_Stop	Monitoring time stopped The monitoring time is stopped as long as this variable carries the signal 1.	STAT	BOOL	x	STOP_WATCH TIME
ST_CurrValue	Remaining monitoring time	STAT	DWORD		-
ST_Valid	Monitoring time active. This variable only has an internal meaning	STAT	BOOL		-
WT_Expired	Waiting time expired This variable can be queried in conditions.	STAT	BOOL		-
WT_Stop	Waiting time stopped The waiting time is stopped as long as this variable carries the signal 1.	STAT	BOOL	X	STOP_WAIT TIME
WT_CurrValue	Remaining waiting time	STAT	DWORD		-
WT_Valid	Waiting time active. This variable only has an internal meaning	STAT	BOOL		-
UsrMsgSend	Message state active This variable carries the signal 1 when a message state is active (only relevant for diagnosis with format converter).	STAT	BOOL		-
UsrMsgQuit	Input variable for error/message acknowledgement (only relevant for diagnosis with format converter).	IN	BOOL	X	-
UsrMsgStat	This variable only has an internal meaning (only relevant for diagnosis with format converter).	STAT	WORD		-

* The section "Cyclic execution of a state" explains in detail when and for which period the variable is set.

System attribute S7_active

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

- CurrentState
- PreviousState
- StateChange
- WT_Stop
- ST_Stop

In order to activate these variables:

1. Select the variable in the variable declaration window and then select the **Edit > Properties** menu command.
2. Select the "Attribute" register tab in the subsequent dialog box and enter the value "true" at the "S7_active" attribute.

3.6.7 Interaction between Variable Declarations and Instructions

Variables from the variable declaration are used in the instructions which you program in the 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.

Changes in the variable declaration have the following effect:

Action in the variable declaration	Reaction in the instructions
Correct change of a name without changing the data type	The variable is displayed immediately in all the instructions with its new name
Change in the data type	If invalid instructions existed, they may become valid. If valid instructions existed, they may become invalid.
Correct name is changed into an invalid name	Instructions are not changed
Deleting a variable which is used in instructions	Valid instruction becomes invalid

3.6.8 Interaction between Variable Declarations and Current Parameter Assignments

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

Changes in the variable declaration of a state graph act as follows on the instance(s) of the state graph in the graph group:

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 entered additionally, but an current parameter assignment is not carried out. You now only have to transfer the current parameter assignment of the invalid name to the new name and then delete the name marked in red.
Change in the data type	If invalid assignments existed, they may become valid. If valid assignments existed, they may become invalid.
Correct name is changed into an invalid name	Assignment is not changed.
Deleting a variable which is used in instructions	Valid assignment becomes invalid.

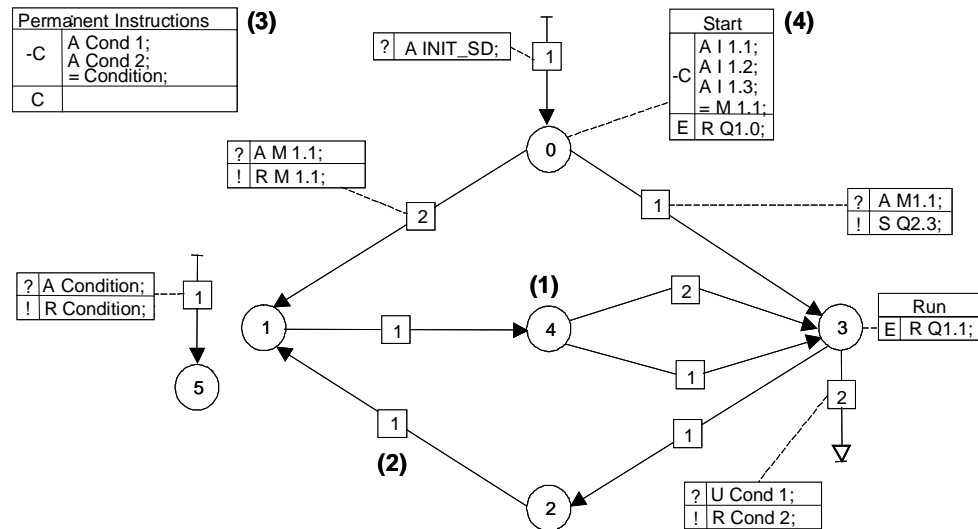
Note

Graph groups which are not opened are not taken into consideration when the variable names are adapted.

3.7 Programming the Structure of a State Graph

3.7.1 Elements of a State Graph

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



The graphic representation of a state graph consists of the following elements:

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

3.7.2 Rules for the Structure of a State Graph

State graphs and graph groups must remain within the following volume of data:

- A state graph can have the following maximum elements:
 - 255 states
 - 4090 transitions
- A graph group can contain a maximum of 255 instances.

For information on the memory which an S7-HiGraph program requires in the CPU refer to the Section "Cyclic processing of a state in the PLC".

3.7.3 Possibilities of Aligning Graphical Objects

Using the grid

The grid serves as an aid in aligning and positioning objects precisely.

- The drawing area is filled with a grid when you select the **View > Grid Points** menu command.
- In order to set the grid size select the **Options > Settings for Graph Groups/State Graphs** menu command and enter the desired values in the "Graphics" register tab.
- Activate the **Options > Align to Grid** menu command in order to move selected objects to the next grid point.
- Select the **Options > Snap to Grid** menu command in order to align objects automatically to the grid during insertion or moving.

Aligning to other objects

The following menu commands facilitate symmetrical alignment of elements:

- In order to align several objects in the same vertical or horizontal position, select several objects with the lasso and then select the **Options > Align > To Object > Vertically/Horizontally** menu command. Then click on the object to which the selected objects are to be aligned.
- In order to place objects equidistantly select the objects and then select the **Options > Align > To Distance > Vertically/Horizontally** menu command.
- In order to place an object before or after another object, use the **Options > Forwards** or **Options > Backwards** menu commands.

Positioning the page

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

- Have the print page frames displayed by using the **View > Print Page Frame** menu command.
- Use the **Options > Align > To Page** menu command to center a selected object or a group of objects exactly to the nearest print page.

Aligning lines

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

- Lines can be bent by clicking on the square in the line middle and dragging it in any direction.
- Whenever you bend a line, it is separated into two sections. Additional nodes from which the line sections can be bent again are displayed at the middle of each section.
- In order to straighten the lines select the **Options > Straighten Line** menu command or delete individual nodes.

3.7.4 States

States

Every state which a function unit can enter is represented by a state in the state graph. A state is represented as a circle. Every state has a unique number within the state graph. Names can also be assigned to the states in order to obtain a clearer overview.

Actions can be triggered in the states. The time at which the action is executed can be defined: For the entry in the state, during the state or when leaving the state.

Initial state

A state for the initialization is required in every state graph.

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

A state becomes the initial state when an Any transition branches into the state which queries the predefined variable INIT_SD.

Steps for inserting states

Proceed as follows in order to enter states:

1. Open a state graph window.
2. Select the **Insert > State** menu command. The cursor then changes its shape to an insertion cursor.
3. Click on the point at which the state is to be inserted.
4. Insert further states or press ESC in order to leave the insert mode and return to the editing mode.

Assigning the state name, number and comment

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

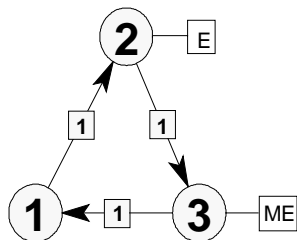
- State name
The state name is indicated in the instruction table of the state. The names are also displayed in the context of a process error diagnosis.
- State number
State graph with sequential numbering require the least memory. When states are inserted, state numbers are assigned sequentially automatically. However, gaps in the numbering occur when you delete states. In this case you can change the numbering in order to close this gap.
- State comment
A well-commented user program can be interpreted rapidly and is thus easy to update.

Assigning names, numbers and comments

When you have selected a state and then select the **Edit > Object Properties** menu command you can enter the name, number and comment of the state in the subsequent dialog box.

Assigning state characteristics

Characteristics can be assigned to states so that later diagnostics are possible. The following figure shows a state graph whose states have had characteristics assigned.



Characteristic	Function	Abbreviation
Error	Outputs an error message to the diagnostic program	E
Message	Outputs an operating message to the diagnostic program	ME

You can select the desired characteristic after you have selected a state and have then selected the **Edit > Object Properties** menu command.

Note

The abbreviations E and ME can optionally also be displayed in the state circle. To do so, select the **Options > Application Settings** menu command and activate the corresponding option in the "Display" register tab.

Handling states

Selecting:

Elements can be selected by various means:

- Individual elements can be selected by clicking on them.
- Several elements can be selected by using the lasso function. Position the cursor on the drawing area, keep the mouse button pressed and drag the cursor around the desired elements.
- If you press the [SHIFT] key as well while you are drawing a lasso, only the elements which can be copied are selected.
- Alternatively you can select several elements by keeping the [CTRL] key pressed while clicking on the elements.

Moving:

1. Select one or more states.
2. Click on one of the selected states and drag the selection to the desired position while keeping the mouse pressed.
3. Select the **View > Update** menu command if the screen display was distorted by the dragging operation.

Copying:

You can copy states both within a state graph as well as to other state graphs.

1. Select one or more states.
2. Select the **Edit > Copy** menu command.
3. Select the **Edit > Insert** menu command. The cursor then changes its shape to an insertion cursor. Click with the insertion cursor on the point in the drawing area at which you want to insert the state.

Cutting:

When you cut a selected object you place it into the clipboard. It can then be inserted at any other point.

- Select a state and then select the **Edit > Cut** menu command. Ensure that you only select the circles which have state numbers. You cannot cut elements if you select instruction tables or permanent instructions at the same time.

Deleting:

- Select a state and then select the **Edit > Delete** menu command. Ensure that you only select the circles which have state numbers. You cannot delete elements if you select instruction tables or permanent instructions at the same time.

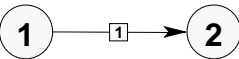
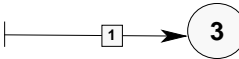
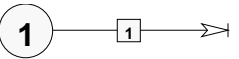
3.7.5 Transitions

Transitions

Transitions contain the transition conditions. A status change is carried out when all the conditions of a transition are fulfilled.

Several transitions can lead from one state. If the conditions of several transitions are fulfilled, the transition with the highest priority switches, the highest possible priority being 1.

Transition types

Transition type	Function	Display
Normal transition	A normal transition leads from a starting state to a subsequent state.	
Any transition	<p>An Any transition leads from all states to a target state. Any transitions are always processed, irrespective of the current state of a state graph. They are used, for example, for the permanent monitoring of higher-level conditions (for example, EMERGENCY-Off). If the monitoring case programmed in the Any transition arises, 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 the other transitions: All the Any transitions always have a higher priority than the 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	<p>A Return transition leads from the current state back to the previously active state.</p> <p>Return transitions do not have a higher priority than the normal transitions.</p>	

Steps for inserting transitions

Proceed as follows:

1. Open a state graph window.
2. Select the **Insert > Transition** menu command. The cursor then changes its shape to an insertion cursor.
3. Click on the output point of the transition.
4. Keep the mouse button pressed and drag the cursor to the target. When you are there, release the mouse.
5. Insert further transitions or press ESC in order to leave the insert mode and return to the editing mode.

Transition types

Depending on where the end points are placed the following transitions are created:

- A normal transition (between two states),
- An Any transition (pointing from any point of the drawing area to a state)
- Or a Return transition (pointing from a state to any point of the drawing area).

Specifying the transition priority

If several transitions leave a state, a different priority is assigned automatically to each transition. The priority is displayed in a small square at the transition arrow. The transition priorities do not have to be assigned without gaps.

If required, change the priorities by using the **Edit > Object Properties** menu command.

Assigning the transition name and comment

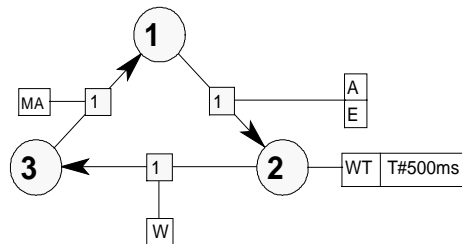
You can assign names and comments to the transitions. These properties do not have any influence on the program execution.

- Transition name
The transition name is displayed in the instruction table of the transition. The name is also displayed at a connected operator control and monitoring device.
- Transition comment
A well-commented user program can be interpreted rapidly and is thus easy to update. The comments are also displayed at a connected operator control and monitoring device.

When you have selected a transition and then select the **Edit > Object Properties** menu command you can enter the name and comment of the transition in the subsequent dialog box.

3.7.5.1 Assigning Transition Characteristics

You can assign transition characteristics in order to program operating modes and to take waiting times into consideration. The following figure shows a state graph whose transitions have had characteristics assigned.



<u>Characteristic</u>	<u>Function</u>	<u>Abbreviation</u>
Manual	Transition only switches in the manual operating mode	MA
Auto	Transition only switches in the automatic operating mode	A
Waits	When the transition is switched a waiting time which was planned in the output state is taken into consideration.	W
Error	Identifies the transition as an error transition.	E

You can select the desired characteristic after you have selected a transition and have then selected the **Edit > Object Properties** menu command.

Note

The abbreviations MA, A, W and E can also be displayed directly in the transition node. To do so, select the **Options > Application Settings** menu command and activate the corresponding option in the "Display" register tab.

Handling transitions

Selecting:

Elements can be selected by various means:

- Individual transitions can be selected by clicking on them.
- Several transitions can be selected by using the lasso function. Position the cursor on the drawing area, keep the mouse button pressed and drag the cursor around the desired elements.
- Alternatively you can select several transitions by keeping the [CTRL] key pressed while clicking on the transitions.

Moving:

1. Move around the entire transition with the lasso.
2. Select the **Edit > Cut** menu command. Select the **Edit > Insert** menu command. The cursor then changes its shape to an insertion cursor. Click with the insertion cursor on the point in the drawing area at which you want to insert the transition.

Moving starting or end points:

1. Select the starting or end point of a transition.
2. Drag it to the desired state while keeping the mouse button pressed.

Copying:

1. Select a transition by clicking on the box in the transition center.
2. Select the **Edit > Copy** menu command.
3. Select the **Edit > Insert** menu command. The cursor then changes its shape to an insertion cursor. Click with the insertion cursor on the point in the drawing area at which you want to insert the transition.

Aligning transition lines:

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

- Transition lines can be bent by clicking on the square in the transition center and dragging it in any direction.
- Whenever you bend a transition line, it is divided into two sections. Additional nodes from which the line sections can be bent again are displayed at the middle of each section.
- In order to straighten the lines select the **Options > Straighten Line** menu command or delete individual nodes.

3.7.6 Permanent Instructions

Permanent Instructions	
C-	
C	

Permanent instructions are executed once per execution cycle of the state graph, irrespective of the current state. In permanent instructions you can, for example, program the following processes centrally:

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

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

Instruction types	Identifier	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.

Steps for inserting permanent instructions

Every state graph contains an instruction table called "Permanent instructions". Enter the instructions here.

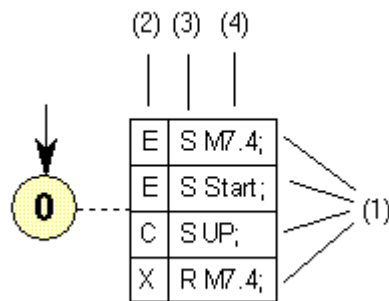
1. Double-click on the "Permanent instructions" instruction table in order to open the instruction window.
2. Select an instruction type in the left-hand section of the instruction window (preceding cyclic action or cyclic action).
3. Enter the instructions in the right-hand partial window.

3.8 Programming Instructions

3.8.1 Instructions in States and Transitions/Permanent Instructions

Instructions represent a process control command. They control, for example, inputs, outputs and bit memories or call blocks. You can assign states or transitions to instructions. In addition you can program so-called permanent instructions, which are executed irrespective of states or transitions.

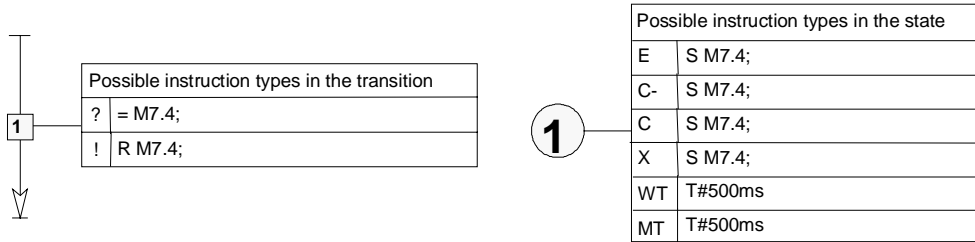
In a state graph diagram instructions are represented in tabular form:



The table contains the following information.

(1) Instruction block	Instructions are subdivided into instruction blocks. Each block is represented by a line in the instruction table. Several instruction blocks of the same type can occur (for example, the example in the figure contains two instruction blocks of Type E). It is advisable to place each instruction in its own block. However, if desired, you can also insert several instructions into a common block.
(2) Instruction type	S7-HiGraph differentiates among several instruction types which determine more specifically when an instruction is executed (for example, when entering the state, during the state or when leaving the state). The instruction types are represented by the abbreviations E, C-, C and X.
(3) (4) Instruction	The instruction itself consists of an operation (3) and an absolute or symbolic address (4)

3.8.2 Instruction Types



The following instruction types exist:

Instruction type	Identifier	Description	Can be used in
Entry actions	E	Actions which are carried out once when entering a state	States
Preceding cyclic actions	C-	Actions which are carried out during a state before the outgoing transitions are checked and which contain state-specific preceding logic operations for conditions.	States Permanent instructions
Cyclic actions	C	Actions which are carried out during a state after the transitions have been checked	States Permanent instructions
Exit actions	X	Actions which are carried out once when exiting from a state	States
Waiting times	WT	Specifies whether the control system is to stay in a state for a minimum period.	States
Monitoring times	MT	Specifies whether the duration of the state is to be monitored.	States
Conditions	?	These instructions describe the conditions which must be fulfilled before a state change can take place.	Transitions
Transition actions	!	These instructions are carried out once when the transition is activated.	Transitions

3.8.3 Rules for Entering STL Instructions

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

Topic	Rule
Syntax	The syntax follows the rules for STL sources. For an exact description of the syntax refer to the online help, "STL language description" chapter.
Lines	Each instruction stands in a separate line.
Instruction blocks	The instructions of one type can be arranged in various instruction blocks in order to improve the structure. This facilitates the automatic resetting of signals which were set during the state.
Semicolon	Every line ends with a semicolon.
Upper/lower case	The program is not case-sensitive for 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 such variables as addresses which you have declared in the variable declaration window. After you have 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 the result of logic operation RLO = 1.
Nesting stack	Monitor the depth of the nesting stack yourself, because this is not checked during compiling. The nesting stack can contain a maximum of seven entries. Exceeding of this limit is not rejected as an error.
Indirect addressing	Indirect addressing is not permitted.

3.8.4 Settings for STL Instructions

Setting the mnemonics

You can use two types of mnemonics when programming instructions:

- SIMATIC mnemonics (for example, E1.0)
- International mnemonics (for example, I1.0)

In order to inform S7-HiGraph which mnemonics you want to use, select the **Options > Settings** menu command and set the mnemonic in the "Language" register tab before opening an S7-HiGraph source in the SIMATIC Manager.

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

Note that instructions which have already been entered are not adapted automatically.

3.8.5 Steps for Entering STL Instructions

Enter the instructions as follows:

1. Double-click on the element for which you want to program an instruction. This can be a state, a transition or the field "Permanent instructions". The instruction window is opened.
2. Select an instruction type in the left-hand section of the instruction window (for example, entry action, cyclic action etc.).
3. The instructions of various types are arranged in blocks in order to improve the structure.
Select an existing instruction block, or select the "Insert" command from the pop-up menu in order to insert a new instruction block.
4. Enter the instructions in the right-hand partial window.
5. In order to use symbolic names which are defined in the symbol table, select the **Insert > Symbol** menu command. A list of all the symbols is then displayed.
6. A syntax check is carried out when you have finished entering a line. The incorrect line is displayed in red, the syntax error is described in a message window. You can either eliminate the error immediately or accept the incorrect instruction and correct the error later.
7. The instructions are displayed in a table in the working window. Drag this table to a suitable position in the working window.

The "STL language description" section contains an overview of all the instructions which can be used in S7-HiGraph.

3.9 Programming Waiting and Monitoring Times

3.9.1 Steps for Programming Waiting Times

You can specify whether the control system is to remain in a state for a minimum period before checking the outgoing transitions.

You can specify the length of the waiting time as an unchanging constant value or as a variable value. If you use a variable value (in the form of a formal parameter or a global variable), you can realize different waiting times in the various instances of the state graph.

Proceed as follows:

1. Double-click on a state in order to have the instruction window displayed.
2. Select the "Wait time" instruction type in the left-hand section of the window.
3. Enter the length of the waiting time in the right-hand section of the window:
 - Constant values are entered in accordance with 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 variable value, enter a variable declared for this state graph or a global variable.

You must then assign the "Waiting" attribute to those transitions which are to take the waiting time into consideration.

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

Note

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

3.9.2 Steps for Programming Monitoring Times

You can specify whether the period spent in a state is to be monitored. If you have specified a monitoring time and the respective state is not left within the specified time, the predefined variable "ST_Expired" is set. In addition an error message can be output to the diagnostic program.

You can specify the length of the monitoring period as an unchanging constant value or as a variable value. If you use a variable value (in the form of a formal parameter or a global variable), you can realize different monitoring times in the various instances of the state graph.

The real time is measured as the monitoring time. The monitoring time therefore continues to run, even if cyclic execution of the state graph is interrupted (for example, during alarm processing).

Proceed as follows:

1. Double-click on a state in order to have the instruction window displayed.
2. Select the "Monitoring time" instruction type in the left-hand section of the window.
3. Enter the length of the monitoring time in the right-hand section of the window:
 - Constant values are entered in accordance with 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 variable value, enter a variable declared for this state graph or a global variable.

Note

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

3.10 Programming Operating Modes

3.10.1 Operating Modes

You can create a program with operating modes by making the switch to the next transition dependent on certain input variables. The input variables are then queried in addition to the transition conditions.

The following operating modes are available:

Operating mode	Behavior of the control system
Auto	A state change takes place if the transition conditions are fulfilled and the AutomaticMode variable has the value 1.
Manual	A state change takes place if the transition conditions are fulfilled and the ManualMode variable has the value 1.
No operating mode assigned	A state change takes place if the transition conditions are fulfilled.

3.10.2 Steps for Programming Operating Modes

In order to program a program with different operating modes, first assign the attributes "Manual" or "Auto" to the transitions.

These attributes have the effect that S7-HiGraph also checks the predefined input variables AutomaticMode or ManualMode before the transition conditions during the program execution. A state change can only take place if the transition conditions are fulfilled and if the corresponding variable has the value 1.

The variables AutomaticMode or ManualMode can, for example, be supplied by a higher-level state graph which takes over the control of the operating modes.

Ensure that you assign an operating mode explicitly to each transition. The default setting is that no operating mode is assigned to the transitions.

Proceed as follows in order to assign the attributes:

1. Select a transition and then select the **Edit > Object Properties** menu command.
2. In the subsequent dialog box select the characteristics "Auto" or "Manual".

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

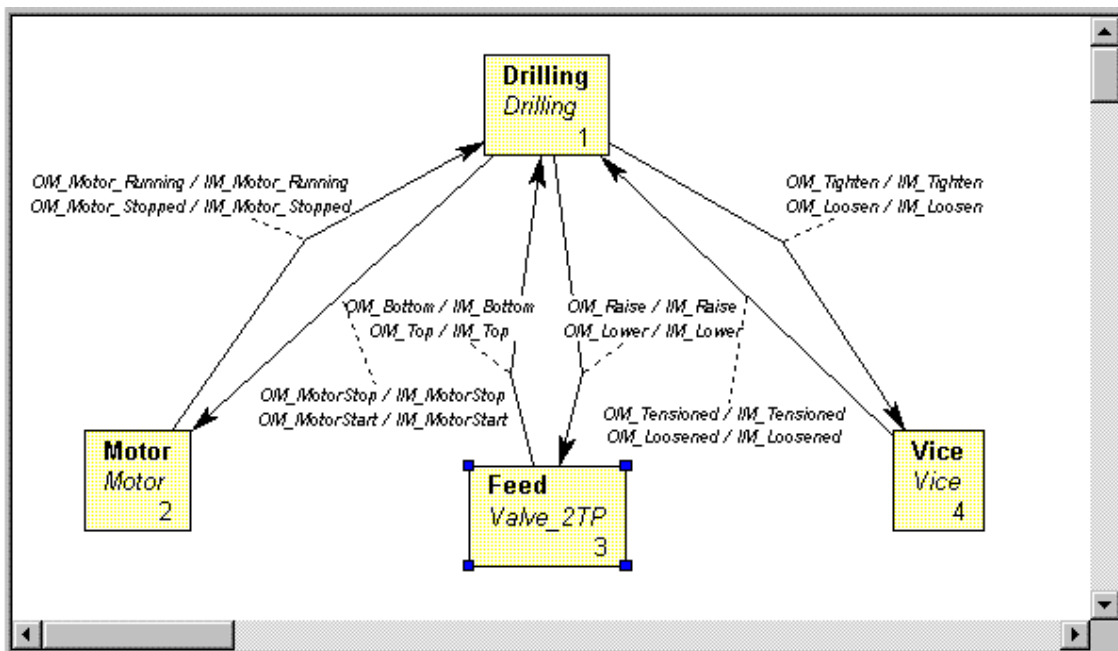
3.11 Programming Graph Groups

3.11.1 Graph Groups

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

A graph group defines an ordered sequence of calls to state graphs that are run cyclically during program execution. A call to a state graph is known as an instance. The instances are processed in the programmable controller in a specific run sequence.

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



3.11.2 Steps for Programming Graph Groups

Proceed as follows in order to create a graph group:

1. Create a graph group by selecting the menu command **File > New Graph Group** and specify "Graph group" as the object type.
2. In order to insert instances of state graphs into a graph group select the menu command **Insert > Instance**.
The instance is then represented as a rectangular symbol in the working window.
In order to assign meaningful names to the instances, select the menu command **Edit > Object Properties** and change the name in the subsequent 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.

1. You can change the run sequence by selecting the menu command **Edit > Run Sequence**.
The run sequence is indicated by a number in the lower right-hand corner of the instances.
2. You now have to assign current parameters to the variables used. For this purpose select the corresponding instance in order to open the "Instructions/Current parameters" window and enter the current parameters in the "Current parameters" register tab.
You can also use symbols that are defined in the symbol table as current parameters. In order to include symbol names which are defined in the symbol table select the menu command **Insert > Symbol**.

Variable declaration

The names of the instances you inserted are displayed as variables in the declaration section STAT in the variable declaration window of a graph group. You can display the variable declaration of the corresponding state graph by double-clicking on an instance name. You cannot edit the declaration.

3.11.3 Programming with Absolute or Symbolic Addresses

You use addresses such as I/O addresses, memory bits, counters, timers, data block and code blocks as the current parameter 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, your programs will be much easier to read if you use symbolic names instead of absolute addresses (for example, "Motor_On" or other descriptions according to one of the naming conventions used in your field of industry).

You assign symbolic names to the absolute addresses in the symbol table of your S7 program. Here you can also assign comments to the symbols. With the combination of short symbols and more detailed comments you can create effective programs and also produce good program documentation.

Calling the symbol table

1. Select the menu command **Options > Symbol Table**.
2. Assign symbols to the absolute addresses in the symbol table.

Automatic displaying of the symbols defined in the symbol table

You can have a list of the symbols defined in the symbol table and import symbols directly into the program in order to simplify the programming of instructions as well as the assignment of current parameters.

The list of symbols can be displayed by two different methods:

- You can use the menu command **Insert > Symbol** (Ctrl + J) to display the list when required.
- In the current parameters the list can also be displayed automatically as soon as you place the cursor in the column "Current parameter". For this purpose select the "Display automatically" option in the "Insert Symbol" register tab in the application settings.

You can determine the sorting and appearance of the list in the application settings ("Insert symbols" register tab).

Symbolic/Absolute representation of shared addresses

You can select between two representations of shared addresses:

- Representation of the absolute address (for example, M1.0)
- Representation of the symbolic name which was defined in the symbol table (for example, Motor_on)

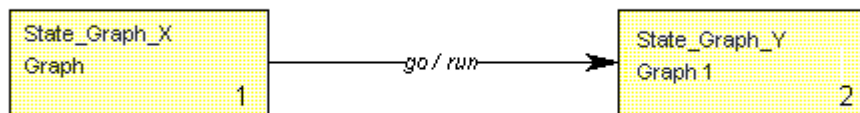
The representation form is set with the menu command **View > Symbolic Addresses** or **View > Absolute Addresses**.

When entering the addresses you can use either the absolute address or the symbolic name, irrespective of the setting specified above. S7-HiGraph converts your input correspondingly.

3.12 Programming Messages between State Graphs

3.12.1 Basics of Exchanging Messages

Messages are used to communicate between state graphs. One state graph sets a signal that is received by another state graph. The signal exchange is clearly displayed in the graph group with arrows.



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

Message types

Message type	Function
Internal message	Communication between state graphs of a graph group. Communication is carried out via a bit address in the 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.

Principle procedure for programming messages

1. A message is realized by means of a bit which is set by the sending state graph and which is 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 (instancing) of the state graphs.
2. One of the two message types "in" or "out" is assigned to the variable for incoming or outgoing messages.
3. Then 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. Then insert instances of the two state graphs into one or more graph groups.
5. You now have to specify which variables are to communicate with each other. This is attained by means of assignments in the current parameter window.

Declaration of variables for messages

1. Declare one variable each (IN_OUT) of the type BOOL in the sending and receiving state graph. The two variables do not have to have the same name.
2. Assign the message type "in" to one of the two variables, and the message type "out" to the other. The message type "in" stands for incoming messages, "out" for outgoing messages. For this purpose click in the "Message type" column of the variable declaration window and select the types from the displayed list.

System attribute S7_message

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

Programming the instructions for messages

1. In the state graph which is to send the message, program an instruction which 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 which queries the signal state of the variable declared here (for example, U Message_in). In the receiving transition you can program an action which resets the respective bit once the message is received.

Linking the incoming and outgoing messages

After you have inserted the instances of a sending and a receiving state graph into a graph group, link the incoming and outgoing messages.

In case of internal messages:

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

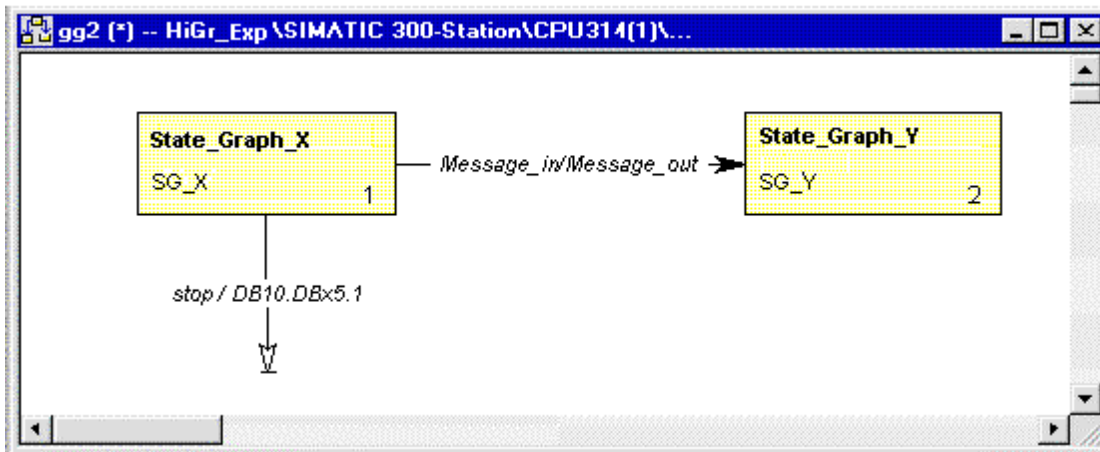
1. Select the instance of the sending state graph and, if necessary, unhide the current parameter window by using the menu command **View > Instructions/Current parameters**.
2. In the current parameter window select the sending variable and enter the receiving variable as the current parameter. Use the following syntax:
Name of the receiving state graph.Name of the message of type "IN"
Example:
`State_graph_Y.Message_in`

You do not have to enter this name manually, but can use the menu command **Insert > Symbol** to have a list of those state graphs displayed which contain the incoming messages. Enter the subsequent point on the keyboard and then select the menu command **Insert > Symbol** again to now have a list of the incoming messages of the selected state graph displayed.

In case of external messages:

1. Select the sending parameter in the current parameter window and enter a shared bit address as the current parameter (for example, DB10.DBx5.1).
2. Now select the instance of the receiving state graph and assign the same shared bit address to the receiving parameter.

The following figure shows a state graph that sends an internal and an external message.



3.13 Display Reference Data

Overview of the possible reference data

You can have the following reference data displayed in order to obtain an overview of the program:

Reference data	Contents
Cross reference	Overview of the use of addresses in the memory areas of I, Q, M, P, T, C and DB within the program.
Program structure	Call hierarchy of the blocks within an S7 program and overview of the blocks used and their dependencies.
Assignment list	Shows the assignment for: <ul style="list-style-type: none"> • Inputs, outputs, memory bits • Timers and counters <p>The overview over which bits of the addresses of the memory areas I, Q and M or which timers and counters within the user program are already assigned is an important basis for finding faults or for modifications.</p>
Symbols not used	Overview of all symbols defined in the symbol table but not used in the user program.
List of addresses without a symbol	Overview of all absolute addresses that were used in the program but for which no symbols are defined in the symbol table.

Note

You will find more information on this function in the STEP 7 documentation.

Generating and displaying reference data

Automatic generation/updating during compiling

Reference data are updated at every compilation of a graph group, if you have activated the compilation setting "Generate reference data" (menu command **Options > Settings for State Graphs/Graph Groups**).

Displaying reference data

Use the menu command **Options > Reference Data** to have the reference data displayed. Before the data are displayed the program checks whether the reference data are up to date. If not, new reference data are generated.

Rapid positioning to location in the program

You can use reference data to position on the locations of an address. Current reference data must be available for this purpose.

Follow the steps outlined below:

1. Open the desired graph group or the desired state graph.
2. Select an address in the current parameter or instruction window.
3. Select the menu command **Edit > Go To > Location**. A dialog box is now displayed which contains a list with the points of use of the address in the program.
4. You can now select a location in the list and use the "Go to" button to jump to the corresponding location in the program.

List of locations

The list of locations contains the following information:

- Block in which the address is used
- Block symbol if it exists
- Details
- S7-HiGraph-specific information
- Type of access to the address: Reading (R), writing (W), reading and writing (RW), cannot be determined (?).

S7-HiGraph-specific information in the reference data

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

Abbreviation	Instruction
HiGraph	Generation language of 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.
P	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.

3.14 Saving and Compiling

3.14.1 Saving State Graphs and Graph Groups

In order to read newly generated state graphs or graph groups or modifications in the programming device data management, you have to save them.

During saving you store S7-HiGraph objects in their current state in the "Source files" container of the S7 program. No syntax check is made. You can also save objects that still contain errors so that you can continue editing them in a later session.

Proceed as follows to save S7-HiGraph objects:

1. Activate the working window of the object to be saved.
2. Select
 - the menu command **File > Save**, if you want to save the object under the same name.
 - the menu command **File > Save As**, if you want to save the object under a different name or in a different S7 program. Enter the new path or the new block in the following dialog box.

Note

- Note that changes to a state graph have an effect on all its instances as soon as you save the state graph.
 - You can also save blocks or source files under other projects in the SIMATIC Manager.
 - For information on the memory requirements please also refer to the Section "Run behavior of an S7-HiGraph program".
-

3.14.2 Compilation of the Program

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

If syntax errors occur, these are displayed in the "Output" window. An executable block will not be created in this case. Warnings, however, do not affect the results of compilation; an executable logic block is created.

In S7-HiGraph you always compile the graph groups as a whole. Individual state graphs cannot be compiled.


Use the following steps to compile a program:

1. First select the menu command **Options > Settings for State Graphs/Graph Groups** and enter the name of the FC and DB and, if required, further compilation settings in the "Compile" register tab. If you specify a DB or FC name which already exists in the "Blocks" folder, the existing block is overwritten during compiling.
2. Select the menu command **File > Compile**.
If you are positioned in the working window of a graph group, the graph group is compiled.
If you are positioned in the working window of a state graph, the program first checks whether the graph is instanced in several graph groups. If so, you are prompted to specify which of the graph groups you wish to compile.
3. After compilation the "Output" window is opened. Check whether errors are displayed in it. Double-click the error message to jump to the position of the error. Follow the instructions in the window to correct the errors.
4. You can use the menu command **Edit > Go To** to jump to the previous or the next error.
5. After you have eliminated the compilation errors, compile the graph group once more.

3.14.3 Setting the Compilation Parameters

Some parameters have to be set for the compiling process in the "Compile" register tab. Some of the details are important if you want to reload changes ONLINE at a later stage.

Select the menu command **Options > Settings for Graph Groups/State Graphs**, in order to open the "Settings" dialog box and carry out the following settings in the "Compile" register tab:

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 DB 	The DB is recreated during compilation under consideration of the entered reserved memory ("Reserve memory (words) in the DB" option). Warning: If you activate this option, you may not reload the program changes during operation. In this case the CPU has to be set to STOP for the loading process.
Activate Any transition only once	This option has the effect that an Any transition is no longer activated if the open-loop control is already in the target state of the Any transition. If the condition of the Any transition continues to be fulfilled, this means, however, that the regular transitions which can be activated are no longer executed because the higher-priority Any transition takes precedence, but does not use it. Take into account that this option increases the code volume. We recommend instead that you explicitly reset the conditions that caused the Any transition to activate by using a reset command (for example, in the transition action).

Option	Effect
Cyclic actions with RLO 0	<p>This option has the effect that the cyclic actions of a state are executed once more with RLO=0 when a state is left so that all the signals which were set during the state are reset. The process is carried out after the transition actions have been carried out and before the exit actions of the state are executed. In order to use this option effectively, it is advisable to locate all the RLO-limiting instructions in own instruction blocks.</p> <p>Note that this option may lead to an increase in the code scope. Alternatively you can also reset the signals in the exit actions of the state.</p> <p>Also take into account that the S7 timers have to be reset.</p>
Preceding cyclic actions also in entry mode	<p>If this option has been activated, the preceding cyclic actions of the state (C-) are carried out in the entry cycle (meaning in the cycle in which a state change takes place). Otherwise these actions are ignored in the entry cycle.</p>
Generate reference data	<p>If you have selected this option, the reference data are generated automatically during compiling.</p>
Reserve memory (words) in the DB:	<p>This option ensures that memory is reserved in the data block for additional state graphs and messages. This specification is important if you want to reload the program changes online.</p> <p>CAUTION: This option is only effective if the DB is restructured during compiling. In order to reload ONLINE, you must therefore first compile the program with the options "Reserve memory (words) in the DB" and "Restructure DB" and then deactivate the two options for all further compiling processes.</p>

3.15 Calling and Loading S7-HiGraph FC

3.15.1 Calling the FC from an S7 Program

In order for the user program to run in the CPU, the FC that was generated when the graph group was compiled has to be called from a cyclically executed block (for example, OB1).

The calling block is programmed with one of the STEP 7 programming languages (for example, LAD, SFC, STL or also HiGraph).

Start parameter INIT_SD

The startup transition scans the startup parameter INIT_SD. If this value is 1, the startup transition is activated and the program branches to the startup state. If you ensure that the parameter INIT_SD has the value "1" on startup in the calling block, the state graph will be initialized with this value. In all further cycles INIT_SD must have the value 0.

The parameter can be supplied by using the OB1 startup information (variable #OB1_SCAN_1).

3.15.2 Requirements for Downloading

The following requirements must be fulfilled in order to download the user program to the PLC:

- The program which is to be downloaded must have been compiled free of errors.
- The call of the S7-HiGraph-FC from a cyclically executed block has been programmed.
- A connection must exist between the programming device and the programmable controller.

3.15.3 Downloading for the First Time

In order to download the complete user program (including OB 1) to the PLC:

1. Set the CPU to the STOP operating mode.
2. In the SIMATIC Manager open the S7 program in which you saved the sources.
3. Select the desired blocks in the "Blocks" folder:
 - HiGraph FC
 - HiGraph DB
 - Calling block (OB, FB or FC)

- HiGraphErrEmitterFB (FB20), if format converter diagnosis is desired
 - HiGraphMsgEmitterFC (FC101), if format converter diagnosis is desired
 - HiGraphUnivEmitterFC (FC102), if standard diagnosis is desired
 - Alarm_S (SFC17) and Alarm_SQ (SFC 18), if standard diagnosis is desired
4. Call up the **PLC > Download** menu command.

In order to only download the FC with the corresponding DB to the PLC:

5. Select the **PLC > Download** menu command while the graph group is open.
6. Specify in the "Download" dialog box whether you want to download the data block together with the FC to the CPU.

3.15.4 Reloading Changes ONLINE

You can reload a user program which you have already downloaded once to the CPU without having to set the CPU to the operating mode STOP. This makes it easy to integrate program changes.

You can reload any type of changes online. However, to add new state graphs and messages, sufficient memory reserves must be available in the data block.

The following prerequisites must be fulfilled before compilation in order to allow reloading:

- Before carrying out the first compilation of the program select adequate reserve memory for additional status graphs and messages in the "Compile" register tab (**Options > Settings for Graph Groups/State Graphs** menu command, "Compile" register tab).
- Ensure that the program has not been compiled with the option "Restructure data block" since the last downloading process (**Options > Settings for Graph Groups/State Graphs** menu command, "Compile" register tab).

If these conditions are fulfilled and you reload a program which is already in the CPU, S7-HiGraph always downloads the smallest amount of data required which has the smallest effects on the executing program. The controlled process is not affected by this download (bumpless reloading).

3.16 Monitoring and Testing the Program

3.16.1 Monitoring the Program Status

Monitoring the program status provides the possibility of visually tracking the execution of a program in the CPU. The current progress through the individual states and transitions is displayed and current information on the instruction tables currently being processed is shown on the screen.

This enables you to find errors that were not displayed by the formal consistency check when creating the program or by the syntax check during compilation. These errors are, for example:

- Programming errors, for example, incorrectly specified monitoring times
- Logic errors in the program structure, meaning the programmed states and transitions do not match the required process sequence.

However, take into account that the debugging function delays the program execution and can thus cause malfunctions or the cycle time to be exceeded.

There are two display possibilities for the program status:

- Program status of graph groups (status overview)
- Program status of state graphs



Warning

Debugging while the plant is running can cause serious damage to persons or equipment if malfunctions or program errors occur!

Ensure that dangerous states cannot occur before carrying out this function!

3.16.2 Displaying in Program Status

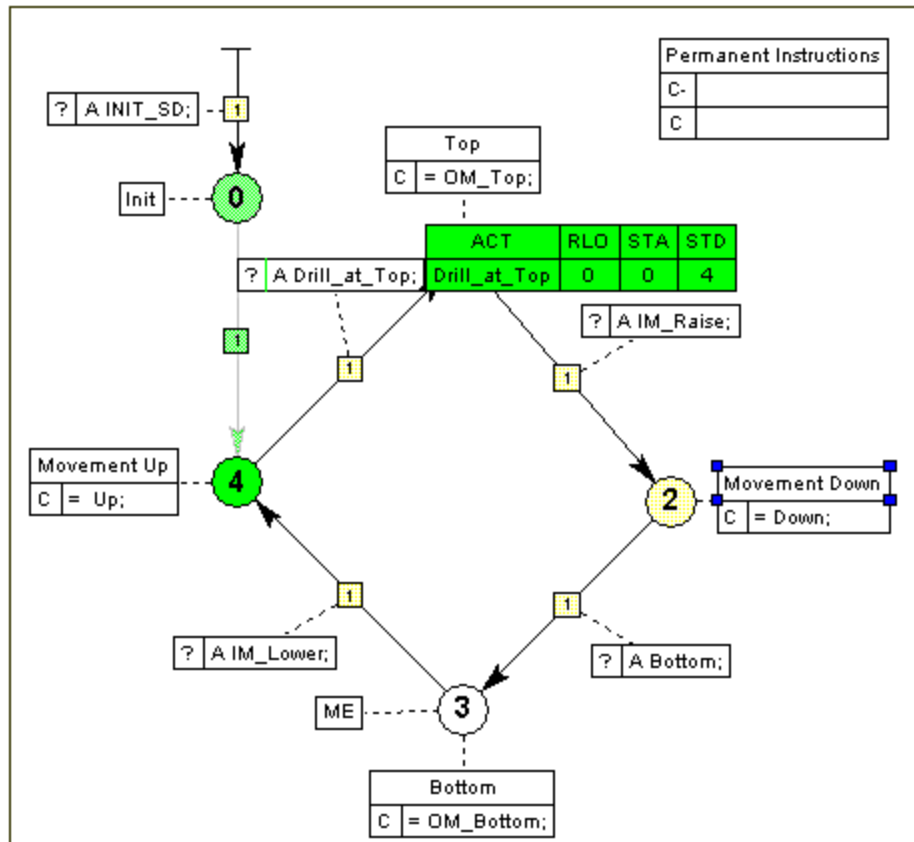
The following possibilities of monitoring are available in the various windows:

- Program status for graph groups:
The status overview is displayed here: You can see all the instances of the current graph group. The current state is displayed in each instance.
- Program status for state graphs
The following detailed status information for a selected instance is displayed here.
 - The active state is highlighted in color.
 - The transition which led to this state is marked in color.
 - The last active state can be marked optionally (in accordance with the setting in the "Status" register tab, which you can call up via the **Options > Application Settings** menu command).

- A table with status information is displayed for the transition with the highest priority leading from the active state. You can choose to display information of another instruction table if more than one is being processed.
- The table with the status information contains the following:

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.

Display of the program status



3.16.3 Prerequisites for Starting the Program Status

The following preconditions have to be fulfilled before the monitoring functions can be carried out:

- The programming device must be connected ONLINE to the CPU.
- The program must have been compiled without errors.
- The program (comprising FB, DB, and OB) must have been downloaded to the CPU.
- The CPU must be in RUN mode (read) or RUN-P mode (read and write).
- The HiGraph FC must be called from a cyclically executed block (for example, OB1).

3.16.4 Steps for Displaying the Program Status

In order to debug your user program you must first download the corresponding program parts, the FC and DB, to the CPU and call them via an organization block (for example, OB1) for cyclic processing.

Proceed as follows to start monitoring the program status:

1. Select the **Debug > Monitor** menu command while the graph group is open. The status overview for this graph group is displayed.
2. Select an instance for monitoring.
3. Select the menu command **Edit > Open Object**. The selected instance is opened ONLINE. The detailed status information is displayed.
4. The table with status information is displayed first for the transition with the highest priority leading from the active state. If required, you can select another instruction table to display its status information.
5. In order to select a different instance for monitoring change back to the status overview, select the desired instance and use again the **Edit > Open Object** menu command.
6. In order to terminate the display of the program status, deactivate the **Debug > Monitor** menu command.

Note

Depending on the capacity of the CPU used, it is possible that not all the program statuses of all the instances contained in very large graph groups can be displayed. In such cases the instances which were selected at the beginning of monitoring are taken first into consideration.

In order to display the program status of a specific instance which is not displayed initially proceed as follows:

1. Deactivate the **Debug > Monitor** menu command.
 2. Select the desired instance (or several ones resp.).
 3. Select the **Debug > Monitor** menu command once more.
-



Warning

Debugging while the plant is running can cause serious damage to persons or equipment if malfunctions or program errors occur!

Ensure that dangerous states cannot occur before carrying out this function!

3.16.5 STEP 7 Test Functions

Monitoring and controlling variables

This function allows you to:

- display current values of variables from your user program (monitoring)
- assign fixed values to the variables for test purposes (controlling)

Proceed as follows:

1. Select the **Debug > Select Variable** menu command.
2. Select the required instances and their variables in the subsequent dialog box.
3. Click on the "OK" button.
The STEP 7 variable table is opened. It already contains the selected variables. You can monitor and control the variables here.

Note

You will find more information on this function in the STEP 7 documentation.

Evaluating the diagnostic buffer

The following events can be entered in the diagnostic buffer of the S7-CPU during the program execution:

- Operating message incoming/outgoing (16#A150/16#A050)
- Fault message incoming/outgoing (16#B153/16#A053).

Requirements:

- The HiGraphErrEmitterFB (FB 20) and HiGraphMsgEmitterFC (FC 101) blocks as well as the system function block SFC 52 must exist in the "Blocks" container of the S7 program.
- Before compiling a graph group you must have selected the "Format converter diagnosis" option in the "Diagnostic" register tab by using the **Options > Settings for Status Graphs/Graph Groups** menu command in the case of graph groups in HiGraph V5 save format.

Evaluating the CPU messages

In the SIMATIC Manager you can use the **PLC > CPU Messages** menu command to trigger the display of operating and fault messages.

3.17 Printing

3.17.1 Printing a Program Documentation

After you have finished creating the program for your automation solution, you can print out all the important data by means of the print function integrated in S7-HiGraph, thus creating a program documentation.

Printable program components

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

- The state graphs in graphic or textual form
- The graph groups in graphic form
- The variable declaration with names, data types, addresses, initial values and comments for the declared variables
- A list of the current parameters with the names of the formal parameters, the symbolic names and absolute addresses of the current parameters and the comments from the symbol table.
- A list of the global addresses with their symbolic names, absolute addresses and the comments from the symbol table.

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

- The object tree (project structure)
- The symbol table
- The diagnostic buffer contents
- The reference data

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

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

3.17.2 Printing Steps

Proceed as follows to print a program documentation:

1. Open the state graph or the graph group which you want to print out.
2. Select the **File > Print** menu command.
3. Select the program components which you want to print in the "Print" dialog box.

4. In the same dialog box select further print setting, such as
 - The zoom factor,
 - The print order and
 - The graphic or textual display of the object
5. Confirm by clicking on "OK".

The file is output on the set printer. If the printout has more than one page, two periods are printed after the page number in the bottom right corner of the page. The last page does not have these periods, indicating no more pages are to follow. This makes it easy to see whether the printout is complete.

Print settings for the entire application

If you wish to output several objects with the same print settings, you have the possibility of setting the print option for the entire application. To do so, select the **Options > Application Settings** menu command and enter the options in the "Graph group settings" or "state graph" settings.

Setting the printer

Use the **File > Print Setup** menu command to select a printer and set further options which depend on the printer used.

Setting the paper format for the printout

Select the **File > Print Setup** menu command to set the paper format. You can carry out the following settings in the subsequent dialog box:

- Paper size (A4, Letter, etc.)
- Portrait or landscape format

In order to set the paper format with page margin select the **File > Page Setup** menu command and set the desired format in the subsequent dialog box (for example, A4 margin). A margin is then set at the left-hand margin of the print document which can then be used for punching and filing.

Setting headers and footers

Project-specific headers and footers can be set in the SIMATIC Manager by using the **File > Headers/Footers** menu command.

Displaying the Print Preview

Use the **File > Print Preview** menu command to check the settings used in the print preview before you send the document to the printer. You cannot edit in the print preview.

3.18 Working with Data from Older S7-HiGraph Versions

3.18.1 Converting Programs from HiGraph 2.6 / 2.7

You can continue to process both individual graph groups as well as complete projects with S7-HiGraph V5 which were originally created with HiGraph V2.6 and V2.7 as well as previous STEP 7 versions.

Programs from older HiGraph versions cannot always be converted completely.

Converting individual graph groups

Prerequisite for conversion is an existing project in STEP 7 Version 3 or higher.

1. First create a new V3 project or convert a V2 project by selecting the "Save As" menu command and activate the option "With Reorganization" in the subsequent dialog box.
2. Open the graph group either with the SIMATIC Manager or with HiGraph. S7-HiGraph recognizes that the graph group originates from a previous program version and asks whether the graph group is to be converted. Confirm the prompt.
3. In the subsequent dialog box enter a name and a filing location for the converted graph group. Note that the graph group must be saved in a STEP 7 V3 project. The graph group is now converted automatically.
4. If you used names, variables or symbols in your old program name which are no longer valid in S7-HiGraph V5, you are prompted during conversion to adapt these.

In order to suppress the prompt to modify the variable names, select the **Options > Application Settings** menu command and activate the corresponding option in the "Migration" register tab. Note that faulty state graphs may result.

5. You now only have to compile the graph group and, if necessary, generate the diagnostic data.

Converting complete projects

You can also convert a project which was created with STEP 7 Version 2 and which contains graph groups from HiGraph V2.6/2.7 completely:

1. Open the project in the SIMATIC Manager.
2. Select the **File > Save As** menu command and select the option "Reorganize before saving (slow)."
3. Enter the name and the storage location for the new project, select the file type "Project," and click the "Save" button.

If the project contains graph groups which were created with older HiGraph version, these are opened automatically and converted to S7-HiGraph V5.

4. In SIMATIC Manager select all the graph groups which are contained in the "S7-program" folder and then select the **File > Compile** menu command.
5. If necessary, generate the diagnostic data by selecting the **Options > Generate diagnostic data** menu command in one of the graph groups.

Note

The following adjustments have to be carried out after the conversion:

- As from V3 the STEP 7 basic system is no longer case-sensitive with regard to symbols. If you used the distinction between upper and lower case symbols in your old program you will be made aware of this during conversion.
 - A new block, the HiGraphErrEmitterFB (default setting FB 20), is required for the format converter diagnosis. Copy the block from the example program to the source container of your project. If a block with the number FB20 already exists in your program, assign another block number in the symbol table and recompile the HiGraphErrEmitterFB block.
-

3.18.2 Using Programs Created in S7-HiGraph Version V4.0/4.01

State graphs or graph groups which were created with S7-HiGraph V4 can be

- converted to S7-HiGraph V5 or
- processed further in V4 format.

Sources in V4 format cannot use some functions of the S7-HiGraph Version 5, such as, for example, the new standard diagnostics.

Sources which have been saved in V5 format cannot be opened in previous versions and cannot be converted downwards.

Specifying the save format of new sources

You can decide whether new sources are to be saved in V5 format or in a save format which is compatible with S7-HiGraph V4. In the latter case you cannot use the new functions of Version 5.

To do so, select the **Options > Application Settings** menu command and activate the corresponding option in the "Save format" register tab.

Leaving sources in V4 format

If you decide to leave the state graphs and graph groups in V4 format, you can activate the option "Leave in V4 format without querying" in the "Save format" register tab (application settings).

Converting sources to V5

When you open state graphs or graph groups which were created with V4, you are prompted whether you want to convert these to Version 5. You can deactivate this prompt in the "Save format" register tab (application settings).

Determining the save format currently set

In order to determine in which format the HiGraph source that is currently opened is saved select the **Options > Settings for State Graphs/Graph Groups** menu command and then the "Save format" register tab. This displays whether the current source is saved in HiGraph V4 or HiGraph V5 save format.

4 Process Error Diagnosis

4.1 Standard Diagnosis via ProTool/ProAgent

4.1.1 Standard Diagnosis via ProTool/ProAgent

It is important to recognize, find and eliminate process errors rapidly when a machine or plant is operating. The standardized diagnostic interface integrated in S7-HiGraph provides a good support.

Diagnostic functions

The following diagnostic functions are available:

- Output of messages when the system enters an error state or if a monitoring time is exceeded
- Determining the addresses causing the errors (initial-value-based criteria analysis)
- Monitoring the movements of individual units in the machine/plant as well as troubleshooting by means of guided manual operation

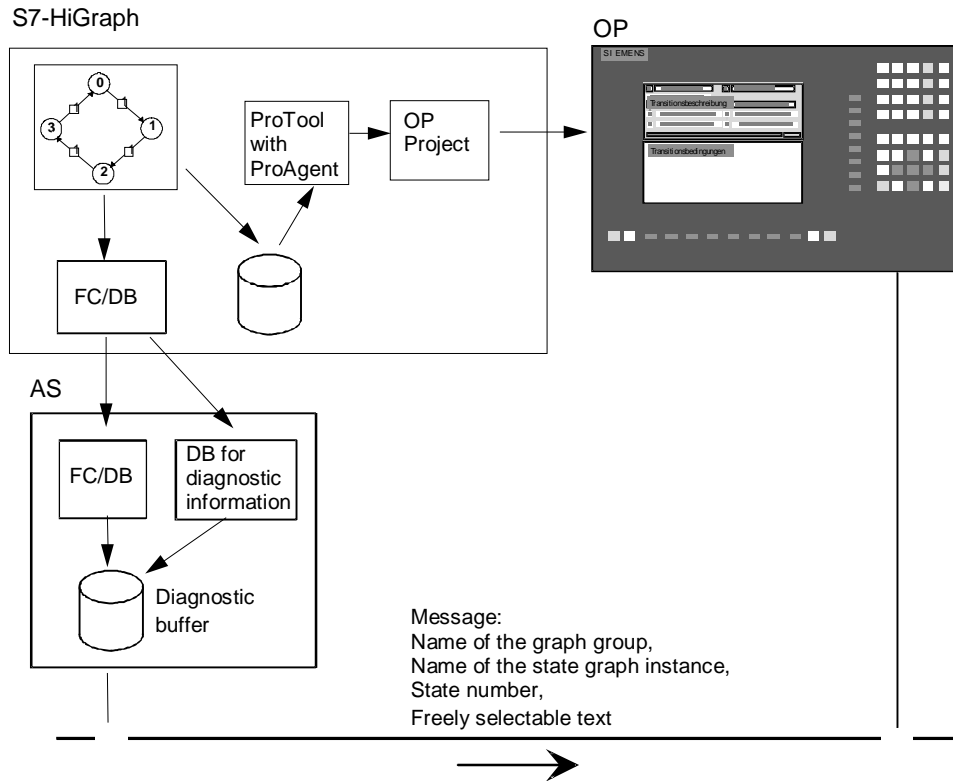
Diagnostic screens at the display device

Four standard screens for displaying the diagnostic information are defined in ProAgent. Soft keys can be used to toggle between the screens.

- The message screen
displays all the existing fault and operating messages
- The detail screen,
displays the result of the criteria analysis for a message. An analysis is carried out which signals led to the message.
- The movement screen,
displays the movements in the process
- The overview screen,
displays the defined units in the plant

4.1.2 Interaction between S7-HiGraph, the Automation System and the OP

The following figure displays the interaction between a programming device/PC with S7-HiGraph, the automation system and the operator control and monitoring system.



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, initial values
- Diagnostic data for the OP

The function and the data block of a graph group can be transferred directly from S7-HiGraph to the PLC, while the diagnostic data for all the graph groups of a PLC first have to be included via ProAgent in a ProTool project and then transferred to the OP.

During operation of the control system messages are transferred in case of errors via the MPI interface messages to the OP which contain a message number and the state in which the error occurred.

4.1.3 Prerequisites for Standard Diagnosis

The following prerequisites have to be fulfilled in order to use the standard diagnostic functions:

- ProTool/ProAgent has to be installed.
- The graph group and the state graphs contained in it have to be available in the save S7-HiGraph V5 format.
- The diagnostic information has to be configured in the user program (for example, by assigning the characteristic "Error" or "Message" to the states or by providing for monitoring times).
- A variable of the type HiGraphSUnitUDT must have been declared in the STAT declaration section of the graph group and of the state graphs.
- If you want to control movements a variable of the type HiGraphMotionUDT must have been declared in the STAT declaration section of the graph group and of the corresponding state graphs.
- The block HiGraphUnivEmitterFC (FC 102) must be contained in the "Blocks" file. It is contained in the library for S7-HiGraph. Copy the block into your project and insert the required entry into the symbol table. If the block number in your program is already occupied, you can also assign a different number in the symbol table and rename the block.
- The used CPU must support SFC17 and SFC18.

4.1.4 General Procedure for Creating Diagnostic Data (Standard Diagnosis)

Proceed as follows in order to create the diagnostic data for process diagnosis with ProTool/ProAgent:

1. While the graph group is opened select the **Options > Settings for Graph Groups/State Graphs** menu command and activate the "Standard diagnosis" option in the "Diagnosis" register tab.
2. In the same register tab enter the name of the DB in which the diagnostic information is to be saved.
3. Further diagnostic settings are required in the graph group or in the state graph in order to supply the individual screens. These settings are described in the following chapters:
 - Displaying messages in the message screen
 - Acquisition of initial values in the detail screen
 - Displaying and controlling movements in the movement screen
 - Displaying units in the overview screen
4. Compile the graph group.

5. Download all blocks in the "Blocks" folder to the automation system:
 - FC and DB which contain the user program
 - Diagnostic DB
 - HiGraphUnivEmitterFC (FC 102)

Note

Creation of diagnostic data is only possible with graph groups which were created in HiGraph V5 save format. You can set the save format in the "Save format" register tab (**Options > Application Settings** menu command).

4.1.5 Displaying Messages in the Message Screen

The message screen is used to display existing messages:

Message types

Messages are differentiated as follows:

Message type	Description
Fault messages	<p>A fault message is output</p> <ul style="list-style-type: none"> • When the program enters a state with the "Error" characteristic, or • When a monitoring time is exceeded (ST_Expired has the signal=1). <p>Fault messages do not disappear until the state causing the fault is left. In addition an obligation to acknowledgement can be defined for fault messages.</p>
Operation messages	<p>An operation message is output when the program enters a state with the "Message" characteristic. Operation messages can disappear from the display by simply acknowledging them.</p> <p>Operation messages display a status in the process. Invalid operations are often indicated by the operation messages.</p>

Message text

The displayed texts consist of:

- A text which can be selected freely,
- The name of the graph group,
- The name of the state graph instance,
- The state number

In order to set the text which can be selected, select the **Options > Application Settings** menu command and enter the desired texts in the "Message texts" register tab.

Obligation to acknowledge

You can specify the obligation to acknowledge separately for operation and fault messages.

Steps required in S7-HiGraph

In order to set the obligation to acknowledgement select the **Options > Settings for Graph Groups/State Graphs** menu command. Then select the "Standard diagnosis" register tab.

When you have opened a graph group and have selected the menu command, specify the obligation to acknowledge for the entire graph group. However, you can also define a special obligation to acknowledge for individual state graphs, if you change the settings while the state graph is opened. The settings in the individual state graphs always have a higher priority.

4.1.6 Displaying Initial Values in the Detail Screen

This figure shows the signal states of the addresses which cause a message. The signal states when the error occurs are called initial values.

The state at the moment when the message occurs is displayed. The logic can be represented in STL as well in SFC. The criteria analysis allows the display to be reduced to the signals involved in the error cause.

Steps required in S7-HiGraph

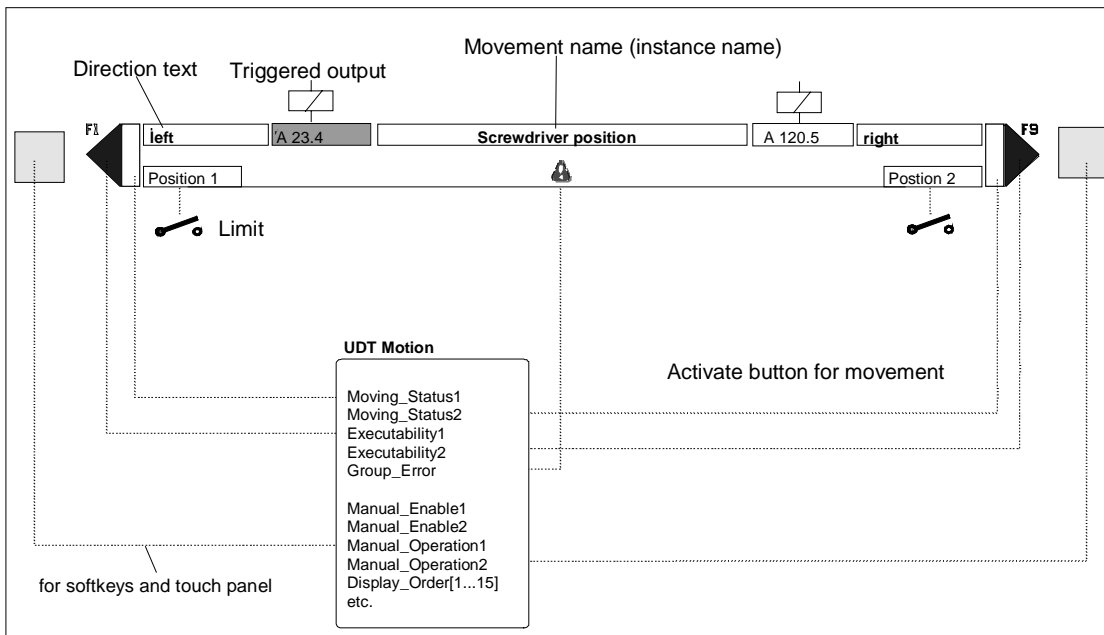
The initial values can only be displayed, if the DB which was required to acquire the initial value was specified in the graph group settings in the "Diagnostic" register tab and if this data block was downloaded to the PLC.

4.1.7 Displaying and Controlling Movements in the Movement Screen

The movement screen allows the movements of the individual units to be monitored as well as to carry out troubleshooting by means of guided manual operation.

On the one hand it shows the state of the movements of a unit (=graph group) and on the other hand it allows the units to be travelled by means of the buttons next to the display.

Every movement is represented by a "line" in the movement screen. A line contains the following information.



Supply of the displayed texts and addresses

The displayed texts and the addresses for the animation of the fields have to be made available. The following table provides an overview of the elements of a movement and their configuration:

Element	S7-HiGraph configuration
Movement name	= <Instance name>
Controlled output	Symbol or address of the output which is entered in the "Settings for state graphs" ("Standard diagnosis" register tab) .
Direction text	Text which is entered in the "Settings for state graphs" ("Standard diagnosis" register tab).
Limit	Formal parameters for the limit, meaning the name of the parameter in the declaration section "IN" which indicates that the limit has been reached. Is entered in the "Settings for state graphs" ("Standard diagnosis" register tab).

Element	S7-HiGraph configuration
Group error	The Group Error bit is supplied by S7-HiGraph automatically during code generation.
Executability	Executability<No.> bit from the HiGraphMotionUDT which has to be supplied by the user in program code. This bit indicates that the movement may be travelled in the direction displayed in manual mode.
Movement active	Moving-Status<No.> bit from the HiGraphMotionUDT which has to be supplied by the user in program code. This bit indicates that the movement is travelling in the direction displayed.
Button to activate movement	<p>The corresponding signals are to be evaluated in the conditions and actions for manual operation in accordance with the OP by the user.</p> <p>Two variants are to be differentiated depending on the OP configuration:</p> <ul style="list-style-type: none"> • Lateral keys as soft keys: The button signals are entered in the UDT elements "Manual_Operation1" or "Manual_Operation2" • Lateral keys as direct keys: There is a bit for every line in the movement screen in the "Display_Order" array. If the line which is assigned to the instance is displayed in the movement screen, the bit which is assigned to the position of the line is set in the UDT.
Sequence of the movements	At S7-HiGraph the line numbers of the movements orient themselves to the execution sequence within the graph group.

Steps required in S7-HiGraph

The following prerequisites must be fulfilled before ProAgent can display the information belonging to a movement:

- Open the "HiGraph" library in the SIMATIC Manager and copy the UDT 2 (HiGraphMotionUDT) into your S7 program.
- Before compiling open the **Options > Settings for Graph Groups/State Graphs** menu command while the state graph is opened and activate the option "state graph realizes a movement" in the "Diagnosis" register tab.
- Specify a variable name for the HiGraphMotionUDT in the "Standard Diagnosis" register tab in the "Movement UDT" input field. HiGraph then generates an entry for the HiGraphMotionUDT in the variable declaration of the state graph.
- Enter texts which describe the two directions of the movement.
- Enter the addresses or texts which are assigned to the limits of the movements. You can use symbols from the symbol table or formal parameters from the variable declaration of the state graphs as addresses.
- Supply the signals of the HiGraphMotionUDT as described in the table.

Note on programming movements

When programming the state graphs use shared addresses or addresses which you declare as variables of the state graph. However, ProAgent can only access variables of the HiGraphMotionUDT for displaying and controlling movements.

Avoid this problem by interconnecting the addresses of the state graphs to the variables of the HiGraphMotionUDT through assignment in the permanent instructions.

4.1.8 Displaying Units in the Overview Screen

The units existing at the machine or plant as well as detailed information on the units are displayed in the overview screen of ProAgent. You can make both graph groups as well as the instances of a state graph visible as a unit.

Steps required in S7-HiGraph

If you want to make graph groups visible:

- Open the graph group and select the **Options > Settings for Graph Groups/State Graphs** menu command. Activate the option "Graph group visible on display device as a unit" in the "Diagnostic" register tab.

If you want to display the instances of individual state graphs as a unit:

- Open the "HiGraph" library in the SIMATIC Manager and copy the UDT 3 (HiGraphSUnitUDT) into your S7 program.
- Open the corresponding state graph and select the **Options > Settings for Graph Groups/State Graphs** menu command. Activate the option "Instances of the state graph visible on the display device as a unit" in the "Standard diagnosis" register tab.
- Then enter a variable name in the "Variable for unit UDT" input field. HiGraph then generates an entry for the HiGraphSUnitUDT in the variable declaration of the state graph.

4.2 Diagnosis via Format Converter

4.2.1 Diagnosis via the Format Converter

Diagnostic data can also be created via the format converter. However, the diagnostic functions have certain limitations compared to the standard diagnosis. For example, only the language scope of S7-HiGraph V2 is available.

Note

You require the S7-HiGraph format converter for this type of diagnosis. This is not part of the standard package. You have to order it separately from your SIEMENS representative and install it.

Message events

The following events can trigger a message at the OP when you use the format-converter diagnostic function.

- **Error states:**
The control system has entered a state with the "Error" characteristic. An error message is output to the OP when the error state is entered. The error display disappears when the error state is left.
- **Exceeding monitoring times (timeouts):**
An error message is output at the OP if the monitoring time is exceeded (the predefined variable ST_Expired has the value 1). The error display disappears when the state in which the monitoring time was exceeded is left.
- **Outputting messages:**
A message is output at the operator control and monitoring device when the control system enters a state with the "message" feature. This message has to be acknowledged by the operator.

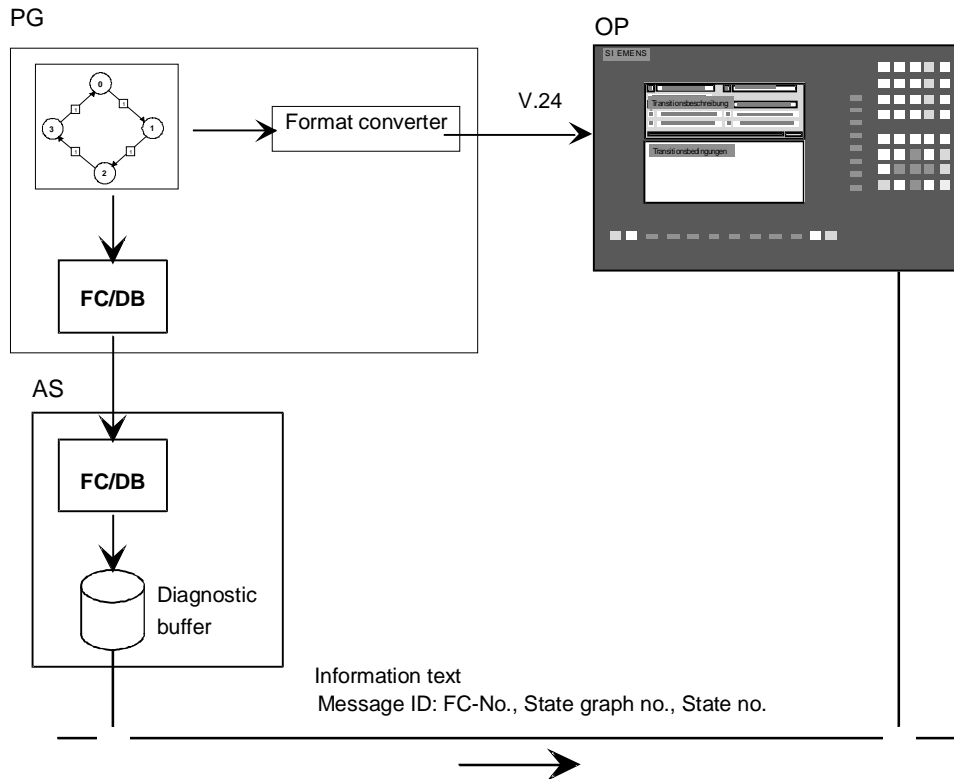
When an event occurs the following detailed information is displayed at a connected OP.

- Symbolic name of the FC or of the graph group in which the event was triggered
- Name of the corresponding instance
- Name of the state

You can localize the program point affected easily in your S7-HiGraph program.

4.2.2 Interaction between S7-HiGraph, the Automation System and the OP (Format Converter)

The following figure displays the interaction between a programming device with S7-HiGraph, the automation system and the operator control and monitoring system.



Description

After you have compiled an S7-HiGraph program, you can have the diagnostic data for the OP generated. These contain all the important data of the state graphs (states, their transitions, instructions, variables, etc. coming from the states).

On the basis of these data the OP is able to display the error messages in clear text and to carry out a criteria analysis (marking of the signals responsible for the fault).

If an error occurs in the control system, an error message is entered in the diagnostic buffer of the PLC and then transferred to the OP by means of a message. In addition to the date and time this contains an identifier for error or operation message, the number of the graph group (FC No.), the number of the instance and the state number. The OP is then able to display the error or the message as described above on the basis of this code.

4.2.3 Prerequisites for Diagnosis via the Format Converter

- The S7-HiGraph format converter must be installed on your device in order to generate the diagnostic data.
- The following blocks are required in order to output error states and messages. They are contained in the library for S7-HiGraph.
 - HiGraphErrEmitterFB (FB 20)
 - HiGraphMsgEmitterFC (FC 101)

Copy the blocks into your project and insert the required entries into the symbol table. If the block numbers in your program are already occupied, you can also assign other numbers in the symbol table and rename the block.

- The predefined variable "UsrMsgQuit" is used to acknowledge messages. Assign the desired actual value to these variables.

Note

Note that the diagnostic function does not support new functions as of HiGraph 4.0. These are:

- Variable waiting and monitoring times
 - Transition actions (!)
 - Preceding cyclic actions (C-) in the state
 - Variable current values for the variable types INT, DINT, REAL, TIME, DATE, TOD, S5TIME, CHAR
 - Other data types than BOOL, WORD or DWORD for variables in the STAT declaration section
 - Other STL instructions than those permitted in HiGraph 2.7.
 - Shared addresses in instructions
 - Instructions which are subdivided into different instruction blocks.
-

4.2.4 Generating Diagnostic Data for the Format Converter Diagnosis

After you have created a S7-HiGraph program and have assigned the desired monitoring times or characteristics, the data have to be conditioned for display on an OP:

1. While the graph group is opened select the **Options > Settings for Graph Groups/State Graphs** menu command and activate the "Format converter diagnosis" option in the "Diagnosis" register tab. This step is only required for sources in save format V5. You can skip this step if your sources are available in save format V4.
2. Then compile all the graph groups in your S7 program.
3. Now select the **Options > Format Conversion** menu command and enter the installation path of the format converter and the desired target directory for the data to be generated in the subsequent dialog box. You can select this command in any graph group. Diagnostic data are created for all the graph groups of the S7 program and are converted for further use in the OP.
4. Transfer the generated data to the OP.

5 STL Instruction Description

5.1 STL Instructions, Sorted by Instruction Families

Bit logic instructions

SIMATIC mnemonic	International mnemonic, if deviating	Description
)		Nesting closed
=		Assign
CLR		Reset RLO (=0)
FN		Edge negative
FP		Edge positive
NOT		Negate RLO
O		OR
O(OR with nesting open
ON		OR NOT
ON(OR NOT with nesting open
R		Reset
S		Set
SET		Set RLO (= 1)
A		AND
A(AND with nesting open
AN		AND NOT
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

Word logic instructions

<u>SIMATIC mnemonic</u>	<u>International mnemonic, if deviating</u>	<u>Description</u>
OD		OR double word (32 bit)
OW		OR word (16 bit)
AD		AND double word (32 bit)
AW		AND word (16 bit)
XOD		Exclusive OR double word (32 bit)
XOW		Exclusive OR word (16 bit)

Timer instructions

<u>SIMATIC mnemonic</u>	<u>International mnemonic, if deviating</u>	<u>Description</u>
FR		Enable timer (free, FR T 0 to T 255)
L		Load current timer as integer to ACCU 1
LC		Load current timer as BCD to ACCU 1
R		Reset timer
SF		Off-delay timer
SD		On-delay timer
SP		Pulse timer
SS		Retentive on-delay timer
SE		Extended pulse timer

Counter instructions

<u>SIMATIC mnemonic</u>	<u>International mnemonic, if deviating</u>	<u>Description</u>
FR		Enable counter (free, FR 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		Counter down
CU		Counter up

Load and transfer instructions

<u>SIMATIC mnemonic</u>	<u>International mnemonic, if deviating</u>	<u>Description</u>
L		Load
T		Transfer
CAR		Exchange address register 1 with address register 2
CAR1		Exchange address register 1 in ACCU 1
CAR2		Exchange address register 2 in ACCU 1

Comparison instructions

<u>SIMATIC mnemonic</u>	<u>International mnemonic, if deviating</u>	<u>Description</u>
>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 real numbers

Block calls

<u>SIMATIC mnemonic</u>	<u>International mnemonic, if deviating</u>	<u>Description</u>
CALL		<p>Block call</p> <p>At a block call enter the parameters in parentheses. The individual parameters are separated by a comma.</p> <p>Example call FC (single line): CALL FC10 (param1 :=I0.0,param2 :=I0.1);</p> <p>Example call FB (single line): CALL FB10, DB100 (para1 :=I0.0,para2 :=I0.1);</p> <p>Example call FB (several lines): CALL FB10, DB100 (para1 :=I0.0, para2 :=I0.1);</p>
CC		Conditioned block call

Fixed point arithmetic

<u>SIMATIC mnemonic</u>	<u>International mnemonic, if deviating</u>	<u>Description</u>
*D		Multiply ACCU 1 and 2 as double integer (32 bit)
*I		Multiply ACCU 1 and 2 as integer (16 bit)
/D		Divide ACCU 2 by ACCU 1 as double 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 double integer (32 bit)
+I		Add ACCU 1 and ACCU 2 as integer (16 bit)
-D		Subtract ACCU 1 from ACCU 2 as double integer (32 bit)
-I		Subtract ACCU 1 from ACCU 2 as integer (16 bit)
MOD		Division remainder double integer (32 bit)

Floating point (real number) arithmetic

SIMATIC mnemonic	International mnemonic, if deviating	Description
*R		Multiply ACCU 1 and 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)

Rotate and shift instructions

SIMATIC mnemonic	International mnemonic, if deviating	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 double integer (32 bit)
SSI		Shift with sign integer (16 bit)

Accumulator operation instructions

<u>SIMATIC mnemonic</u>	<u>International mnemonic, if deviating</u>	<u>Description</u>
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
CAD		Change sequence of bytes in ACCU 1 (32 bit)
TAK		Toggle ACCU 1 with ACCU 2
CAW		Change sequence of bytes in ACCU 1 (16 bit)

Conversion instructions

<u>SIMATIC mnemonic</u>	<u>International mnemonic, if deviating</u>	<u>Description</u>
BTD		BCD to double integer (32 bit)
BTI		BCD to integer (16 bit)
DTB		Double integer (32 bit) to BCD
DTR		Double integer (32 bit) to real number (32 bit; IEEE-FP)
INVD		1-complement double 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 double integer (32 bit)
NEGI		2-complement integer (16 bit)
NEGR		change sign of real number
RND		Round to integer
RND-		Round to lower double integer
RND+		Round to upper double integer
TRUNC		Truncate

5.2 STL Instructions, Sorted by Mnemonics

SIMATIC mnemonic	International mnemonic (if deviating)	Description
)		Nesting closed
*D		Multiply ACCU 1 and 2 as double integer (32 bit)
*I		Multiply ACCU 1 and 2 as integer (16 bit)
*R		Multiply ACCU 1 and 2 as real number (32 bit, IEEE-FP)
/D		Divide ACCU 2 by ACCU 1 as double 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 double 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 double 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 real 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)
ATAN		Arc tangent of a real number (32 bit, IEEE-FP)
BTD		BCD to double integer (32 bit)
BTI		BCD to integer (16 bit)
CALL		Block 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 (single line): CALL FC10 (param1 :=I0.0,param2 :=I0.1); Example call FB (single line): CALL FB10, DB100 (para1 :=I0.0,para2 :=I0.1); Example call FB (several lines): CALL FB10, DB100 (para1 :=I0.0, para2 :=I0.1);
CC		Conditioned block call
CLR		Reset RLO (=0)
COS		Cosine of a real number (32 bit, IEEE-FP)
DEC		Decrement ACCU 1
DTB		Double integer (32 bit) to BCD
DTR		Double integer (32 bit) to real number (32 bit, IEEE-FP)

SIMATIC mnemonic	International mnemonic (if deviating)	Description
EXP		Exponential of a real value (32 bit, IEEE-FP)
FN		Edge negative
FP		Edge positive
FR		Enable counter (free, FR C 0 to C 255)
FR		Enable timer (free, FR T 0 to T 255)
INC		Increment ACCU 1
INVD		1-complement double 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 double 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
ON(OR NOT with nesting open
OW		OR word (16 bit)
POP		CPU with two accumulators
POP		CPU with four accumulators
PUSH		CPU with two 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 lower double integer
RND+		Round to upper double integer
RRD		Rotate right double word (32 bit)
RRDA		Rotate ACCU 1 right via CC1 (32 bit)
S		Set
S		Set counter preset value
SF		Off-delay timer
SD		On-delay timer

SIMATIC mnemonic	International mnemonic (if deviating)	Description
SET		Set RLO (= 1)
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 double integer (32 bit)
SSI		Shift with sign integer (16 bit)
SE		Extended pulse timer
T		Transfer
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)
CAR		Exchange address register 1 with address register 2
CAR1		Exchange address register 1 in ACCU 1
CAR2		Exchange address register 2 in ACCU 1
CAW		Change sequence of bytes in ACCU 1 (16 bit)
TRUNC		Truncate
A		AND
A(AND with nesting open
AD		AND double word (32 bit)
AN		AND NOT
AN(AND NOT with nesting open
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		Counter down
CU		Counter up

5.3 Valid Data Types

<u>Type/Description</u>	<u>Size in bits</u>	<u>Format options</u>	<u>Range and numerical representation (lowest to highest value)</u>	<u>Example</u>
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	Binary number Hexadecimal number BCD Decimal number without preceding sign	2#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	Binary number Hexadecimal number Decimal number without preceding sign	2#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)	2#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 with preceding sign	-32768 to 32767	L 1
DINT (integer, 32 bits)	32	Decimal number with preceding sign	L#-2147483648 to L#2147483647	L L#1
REAL (floating-point number)	32	IEEE floating-point number	Upper limit: $\pm 3.402823e+38$ Lower limit: $\pm 1.175 495e-38$	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 1 ms, integer with preceding sign	T#- 24D_20H_31M_23S_648M S to T#24D_20H_31M_23S_647 MS	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

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

6 Configuration Notes

6.1 Introduction

The example of a drilling machine was used to familiarize you with S7-HiGraph programming. In reality you will have to realize far more complex automation tasks. We will therefore show you here how S7-HiGraph can be used to control complete plants.

For this purpose we have configured a transfer line in which the drilling machine already known to you occupies one machining station. Some additional functions are required in order to include the drilling process in the overall control system of the transfer line. The following functions were therefore added to the drill unit in comparison to the drilling machine:

- A state graph which controls the cooling agent supply
- Operation enables
- Operating modes (automatic, single-step and manual/setup)
- Monitoring units

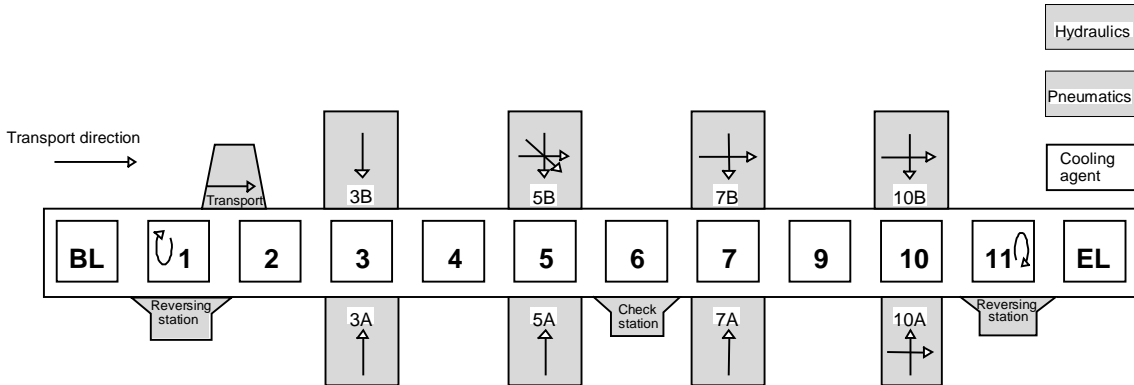
The following sections show how easy these functions are to realize with S7-HiGraph.

The "Drill unit" example is included in the scope of delivery as the project ZEn03_02_HiGraph_DrilUnit. The format converter diagnostic function was used as the example for configuring monitoring units. The last section of the chapter also contains configuration notes on the use of the standard diagnostic function.

6.2 Automation Task Transfer Line

Transfer line

The transfer line has the following structure:

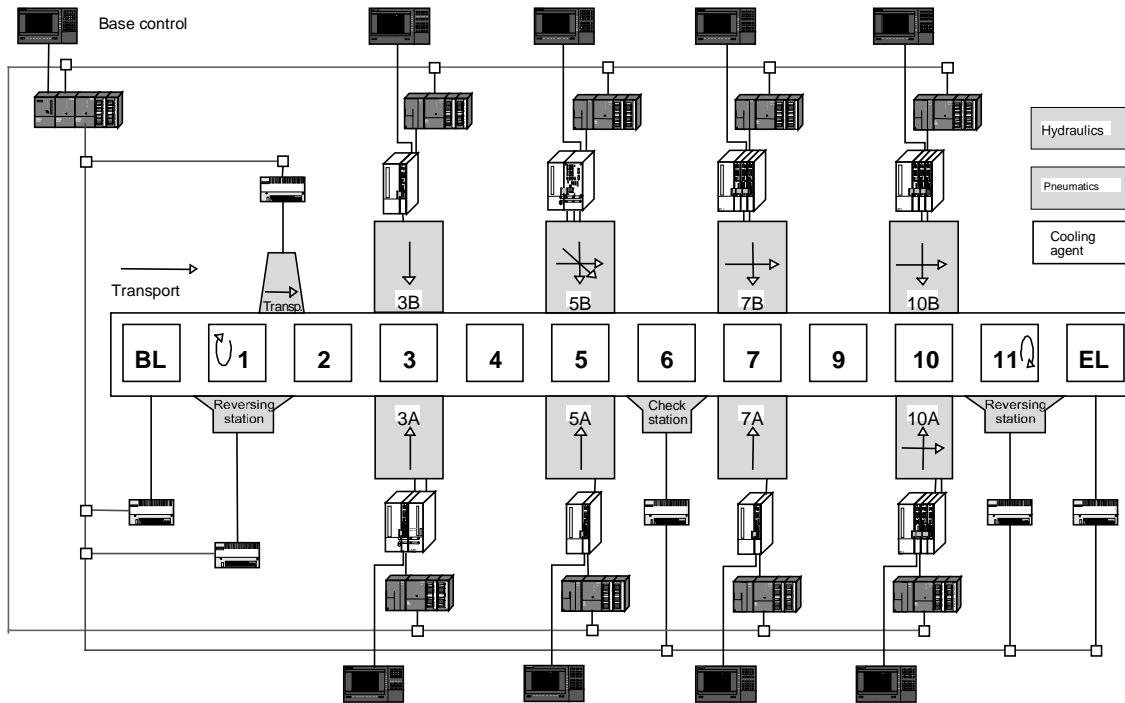


Characteristic of a transfer line is the rigid transportation system in which all the workpieces are transported simultaneously. During each transportation cycle the workpieces are each moved forward by one processing station each.

The stations 1 to 11 have processing units (3A, 3B, 5A, etc.), reversing stations (1, 11) or check stations (6) assigned to them. The loading station BL forms the beginning while the discharge station EL forms the end. Media (cooling, hydraulic and pneumatic agents) are supplied at a higher level. The drill unit occupies one of the processing units.

Device assignment

The components of the transfer line contained in the layout are assigned to the control and operating devices.



As a rule a control system with an operator panel is assigned to each processing unit. One to four controlled drives (shafts/spindles) are used per unit. One-axle modules or NCs (for example, SINUMERIK 810D/840D) are then required for these.

The higher-level base control takes over the transportation and thus the coordination of the processing units. The loading/discharge stations, reversing stations and media such as hydraulics, pneumatics and cooling agents are often assigned to the base control, since it is often not worthwhile to use an autonomous control system.

All the control systems are networked by means of Profibus DP. Profibus DP is also used to couple the decentralized I/Os to the base control system.

The base control for its part is coupled to the factory network for production data acquisition (not represented in the layout).

6.3 Determining the Functions to be Controlled

Functions of the base control system

After the assignment of the technological components to the control devices has been specified, the functions which are to be controlled by the control devices can be determined.

The table shows an example of the functions of the base control system:

Function group	Function
Transportation	The transportation consists of the movements "Raise/Lower" as well as "Forwards/Backwards". The complete transfer line is coordinated via the transportation, because processing of the workpieces is not started until transportation has been completed and all the workpieces are clamped. Further transportation is not carried out until all the units have been processed completely.
Lubrication	The lubrication of a transfer line is usually carried out centrally, with several lubrication services often being supplied. Pumps, valves, level and pressure controls are required for this purpose.
Loading/Discharging	Functions such as the intake, separation, type control, protection doors are required for the loading/discharging stations operated by the base control system.
Auxiliary units	The auxiliary units include the media: hydraulics, pneumatics and cooling agents as well as the transportation of shavings. Pumps, valves, level controls, filter monitorings, etc. are required per medium.
Reversing stations	In this example the reversing stations are also controlled by the base control system. Facilities such as "Clamp/Release", "Raise/Lower", as well as reversing devices, grippers, etc. are required.

Functions of the processing units

The following table shows an example of the functions of a simple processing unit:

Function group	Function
Machining	In the simplest case (for example, drill operations) a spindle (to drive the drill), a slide (to move the spindle with drill) and cooling agents are required for machining.
Clamping station	Every unit requires a clamping station with which the workpieces are clamped securely during machining. Up to 4 clamping cylinders can be used per unit. After machining the shavings must be rinsed away so that the next workpiece can be clamped exactly again.
Media	In the case of the central supply of a transfer line with hydraulic oil, air and cooling agents only monitoring functions for pressure and level are required additionally for the media.
Protective doors	The protective doors of a processing unit have to be locked and unlocked. Monitoring functions are required here as well.

6.4 Determining the State Graphs

The following tables show the state graphs which are required for the function units listed in the previous section. In addition, the higher-level state graphs which are required for coordination have been added.

State graphs for the base control system

Function group	Function	State graphs for
General		Operation enables, operating modes
Auxiliary units such as hydraulics, pneumatics, cooling agents, shavings transportation	Pumps, valves, level monitors, filter monitoring	Starting-up coordination, pumps, valves, level monitors, filter monitoring
Lubrication	Pumps, valves, level monitors, pressure monitors	Coordination, pumps, valves, level monitors, pressure monitors
Transportation	Raise/Lower, Forwards/Backwards	Coordination raise/lower, forwards/backwards
Loading/Discharging	Intake, separation, type control, protective doors	Coordination loading, intake, separation, type control, protective doors
Reversing stations	Clamp/Release, Raise/Lower, reversing devices, grippers, etc.	Coordination clamp/release, raise/lower, reversing devices, grippers, etc.

State graphs for a processing unit

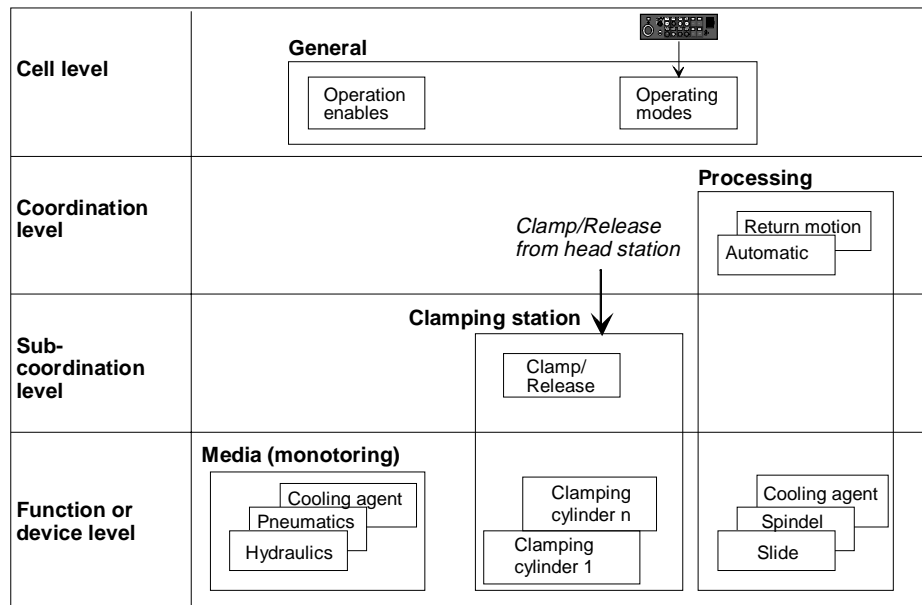
Function group	Function	State graphs for
General		Operation enables, operating modes
Media	Hydraulics, pneumatics, cooling agents	Monitoring of hydraulics, pneumatics, cooling agents
Protective doors	Locking/unlocking of protective doors	Locking/unlocking of protective doors
Clamping station	Clamping/releasing, rinsing	Coordination clamping/releasing, rinsing
Machining	Spindle, slide, cooling agent	Coordination automatic, coordination reversing, spindle, slide, cooling agent

6.5 Formation of Graph Groups

Level structure of the state graphs

The state graphs can be structured into levels which have the following meaning:

- **Cell level**
The cell level combines functions which are of central importance (operation enables and operating modes).
- **Coordination level**
The coordination level contains the coordination functions such as for automatic operation, return motion after an operation interruption, etc.
- **Sub-coordination level**
It may be advisable to combine several functions into a sub-coordination level (for example, the clamping cylinders for the function Clamp/Release).
- **Function level**
The functional unit level contains the state graphs for controlling and monitoring the individual functional units such as motors, valves, etc.



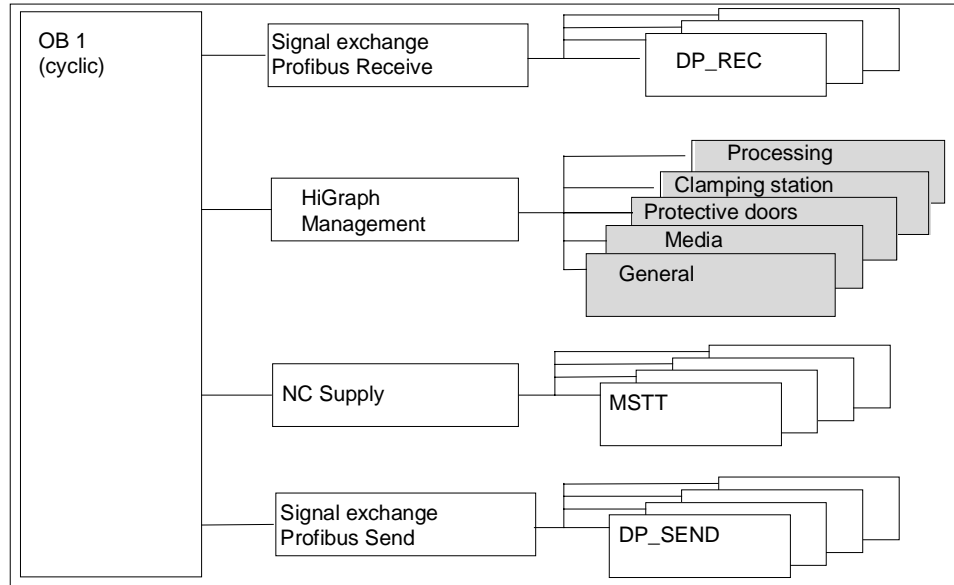
Formation of graph groups

Three typical cases are differentiated when forming graph groups (represented in the figure by frames around several state graphs):

- Graph groups with coordination function
The graph group contains state graphs of the function level as well as higher-level state graphs for their coordination. Several state graphs for coordination can also be used.
- Graph groups without coordination function
The graph group is only used to combine several state graphs (for example, hydraulics, pneumatics and cooling agents into the graph group "Auxiliary units").
- Graph group with sub-coordination function
The actual coordination by a higher-level coordinator in the base control system is carried out here. The sub-coordination shown in the example has the effect that several clamping cylinders act as one functional unit outwardly (as well as for controlling in the manual/setup mode).

6.6 Specifying the Program Structure

S7-HiGraph creates one FC and one DB each per graph group. This FC has to be called in a block (OB, FB or FC). Since as a rule other programs are also required in addition to the programs generated with S7-HiGraph, it is advisable to call all the FCs created with S7-HiGraph in a single block (FC or FB) as shown in the following example.



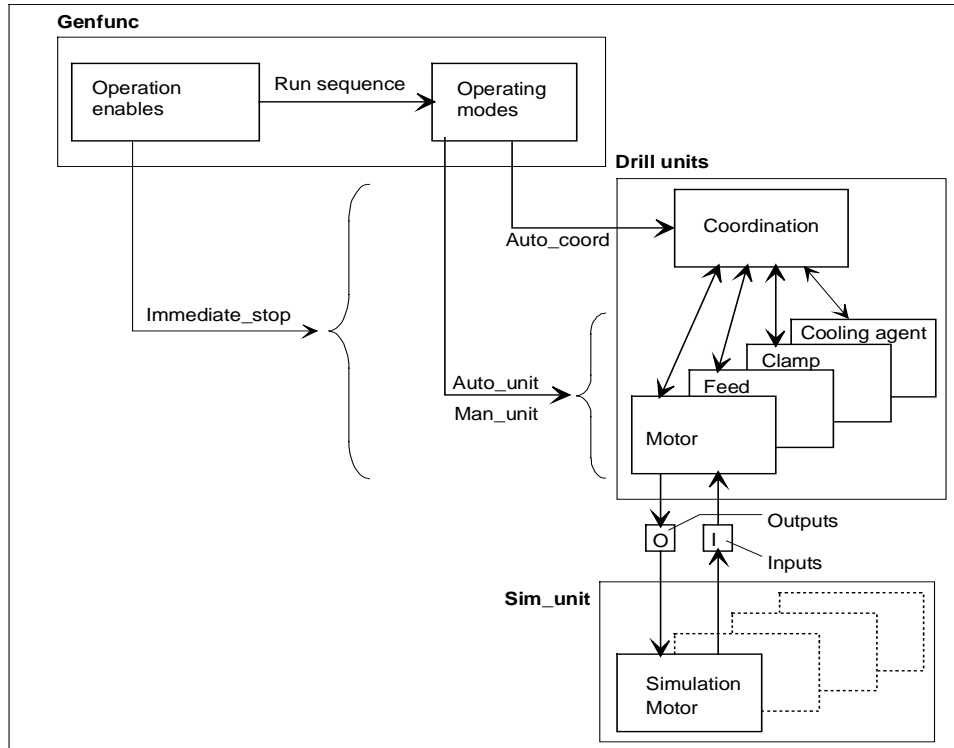
The various elements are processed cyclically in the order of their call:

- The graph groups (with gray background) are preceded by program components with which the signals which come via the Profibus from the base or transportation control system or also from the opposite processing unit are received and conditioned (Profibus Receive signal exchange).
- The various graph groups are called by a "HiGraph management" FB which is itself called in OB1.
- Those program components used to supply a connected NC (NC supply) are processed next.
- Finally, program components for sending the signals to the base control system or to the opposite processing unit (Profibus Send signal exchange).

6.7 Creating State Graphs

6.7.1 Overview: State Graphs and Graph Groups for the Drill Unit

The graph groups and state graphs interact as follows:



A graph group contains the functions for operation enables and operating modes (GenFunct). The state graph for the operation enables acts on all the state graphs via the operating modes state graph, since all the operating mode signals are removed when the execution enable ceases. In addition it supplies the signal "Immediate_stop" which is evaluated in all the relevant state graphs. If required this signal freezes all the movements.

The "Operating modes" state graph supplies the operating-mode signals to all the state graphs (both for the state graphs for coordination as well as for the state graphs with which the function units of the machine are controlled and monitored).

A second graph group contains the state graphs for operating the drill unit. These state graphs normally control actuators via output signals and evaluate sensor signals.

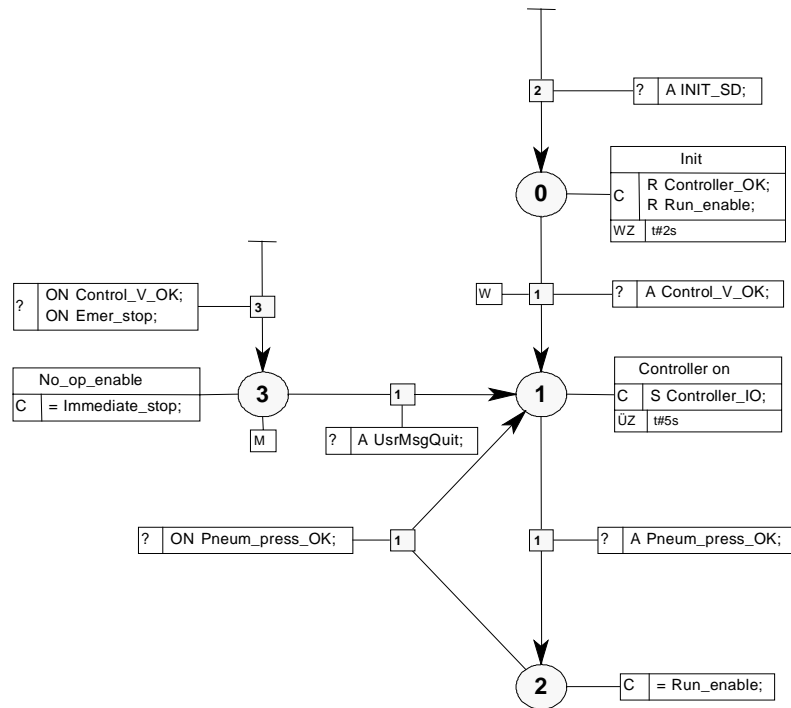
In the supplied example actuators and sensors are simulated via state graphs in order to simplify the test. The third graph group contains state graphs for this simulation.

6.7.2 State Graph for Controlling Operation Enables

In manufacturing plants media such as hydraulic and pneumatic systems from whose availability the operation of all the other functional units depends are often required. It is therefore advisable to control activation of the hydraulic and pneumatic systems from a central point after the plant has been activated and also to generate stand-by signals for the other blocks.

In the example these are the functions control voltage and pneumatic system. In addition the signal "Emer_stop" is evaluated.

A state graph as shown in the example below provides a solution for the generation of these signals.



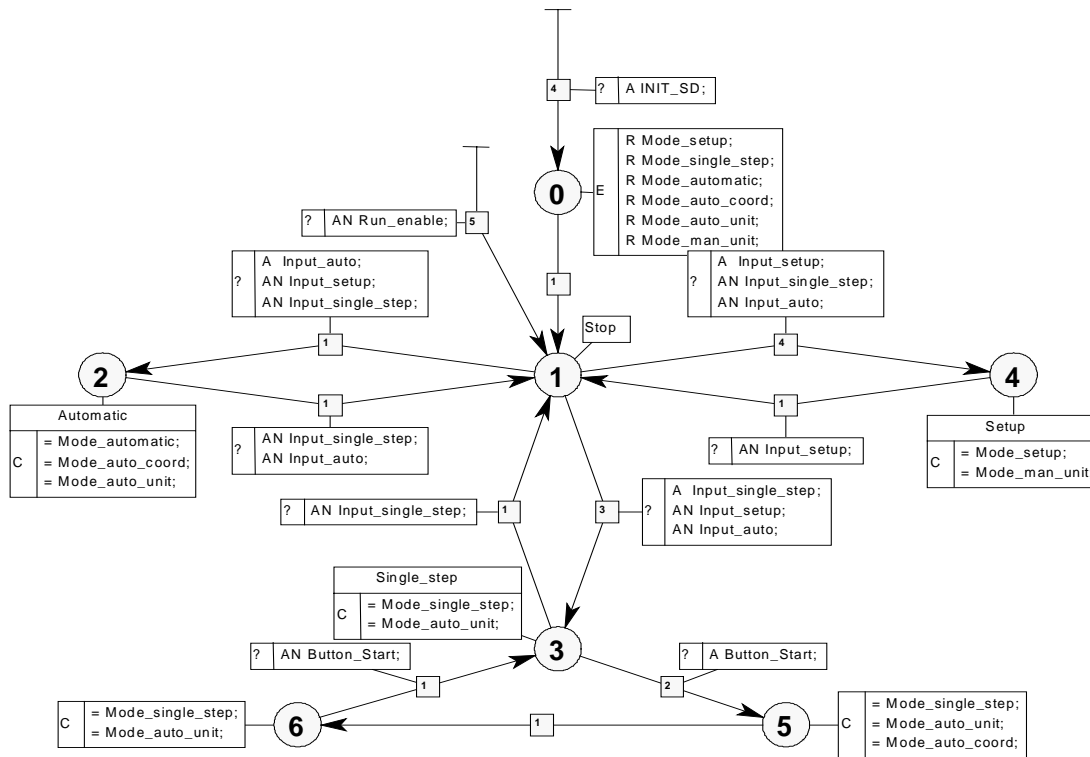
- After the control system has been activated the signal "Controller_OK" in state 1 signals that the control system has started up and that all the power supplies have been switched on. When the pneumatic pressure has built up, operation of the drill unit is enabled in state 2 (execution enable).
- When the pneumatic pressure drops, a change to state 1 removes the execution enable. The movements which have been started are completed. When the pneumatic pressure returns, the prerequisite for the execution enable is fulfilled.
- If, on the other hand, the load voltage fails or if the "Emer_off" is specified, not only the execution enable is removed, but the signal "Immediate_stop" is also output in state 3 and all movements are thus frozen. In addition, a message is output to the operator panel. When the cause of the interruption no longer exists, operation of the transfer line is enabled again by pressing the acknowledge button at the operator panel (at diagnosis via the format converter).

6.7.3 State Graphs for Controlling Operating Modes

The operating modes automatic, single-step and manual/setup are used in this example. The preselection is carried out via separate inputs (one input per operating mode).

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

Signals for indication	Mode_automatic, Mode_single_step, Mode_setup
Signals for coordination	Mode_auto_coord
Signals for functional units	Mode_auto_unit, Mode_man_unit



The state graph fulfils three functions:

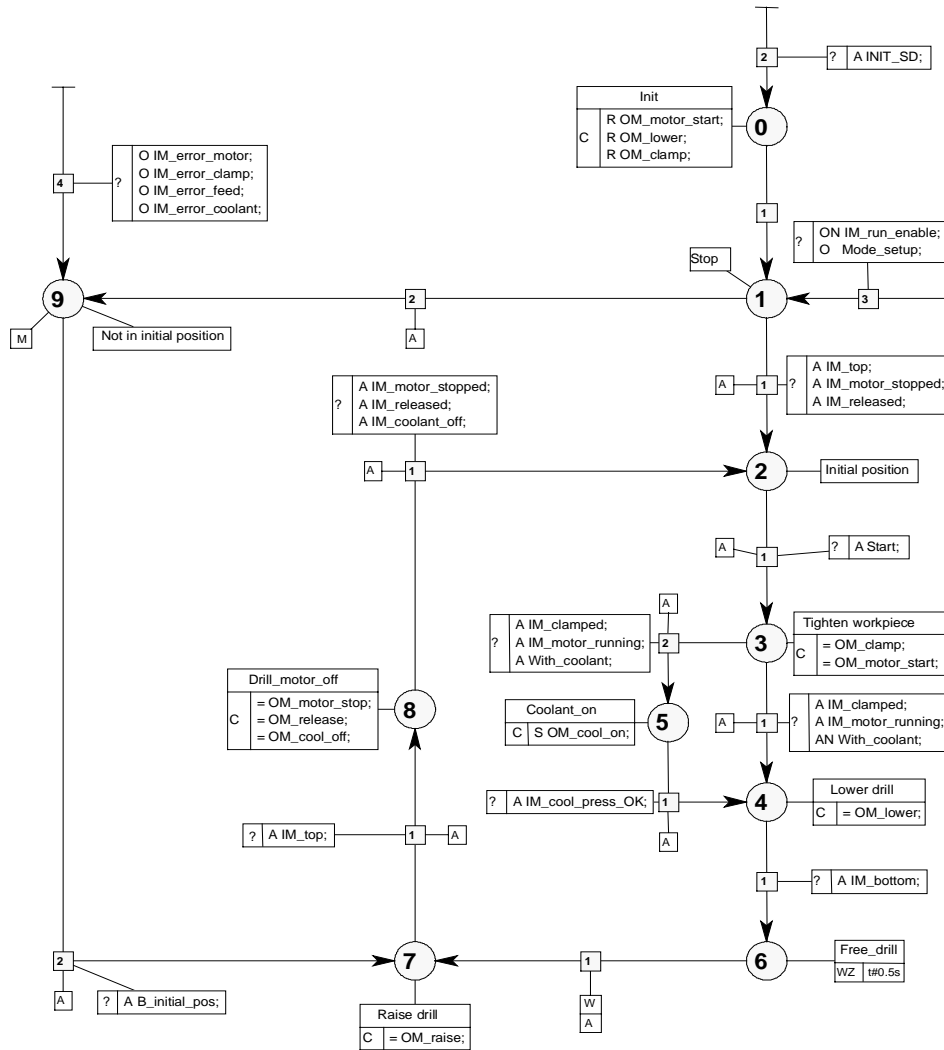
- It decodes and debounces the signals of the operating-mode selection switch. The corresponding operating mode is not activated until its signals are unique.
- It checks constantly whether the execution enable exists. Otherwise a status change to the state 1 is carried out by means of an Any transition and remaining in this state is forced.
- It generates the operating-mode signals for the state graphs of the coordination and functional units levels.

The operating mode signals are generated in accordance with the following scheme:

- The input signals Input_auto, Input_single_step and Input_setup transmit the information which operating mode the worker has preselected.
- The signals Mode_auto_unit and Mode_auto_coord are specified to the state graphs of the coordination and functional-unit level in the automatic operating mode.
- In the single-step operating mode only the automatic mode is specified for the state graph of the functional-unit level in state 3 (Mode_auto_unit signal). In the coordination state graph the lack of the signal Mode_auto_coord deactivates the transitions which have the characteristic "Automatic" (A) and thus suppress the switch to the next transition. When the Start button is pressed, the signal Mode_auto_coord is emitted for one cycle to the coordination state graphs, thus allowing the switch to the next respective state.
- In the Manual/Setup operating mode the Mode_manual_unit signal enables manual operation for the state graphs of the functional-unit level. The coordination state graph is changed to the state 1 (stop) and thus deactivated.

6.7.4 State Graph for Coordinating the Drill Unit

The coordination state graph Drilling contains the functions for the automatic and single-step operating modes. In addition the abort conditions are taken into consideration.



Initialization

After initialization the system changes immediately to state 1. In this state the state graph is held by means of an Any transition, unless an execution enable or the manual/setup operating mode exists.

If the conditions in the Any transition are not fulfilled, the initial position of the drill unit is queried via transition 1 at state 1. If it is in this position, the system changes to state 2, thus allowing the drilling process to be started. If the drill unit is not in the initial position, the system branches through transition 2 to state 9 and a message is output to the operator panel. The B_initial_pos button can be used to initiate travelling to the initial position.

Automatic and single-step operating modes

In order to implement the operating modes the "Auto" characteristic is assigned to the respective transitions (represented in the graphics by a flag with the character A). Transitions with this characteristic can only be activated, if the automatic operating mode is present or if the Start button is pressed in single-step operating mode.

The corresponding predefined "AutomaticMode" variable (in the IN area of the variable declaration) is supplied in the example via the "Mode_auto_coord" (M 10.5) bit memory which is set by the "Operating modes" state graph. In automatic operating mode this always has the signal "1". In single-step mode it is set for one cycle each after the Start button has been pressed.

Manual/Setup operating mode

The "Mode_setup" signal is used to change to state 1 (stop) via an Any transition, thus switching the state graph to inactive. The movements are then controlled directly via button signals on the operating panel. This is taken into consideration in the state graphs for the individual functional units.

Abort situations

An abort situation exists if:

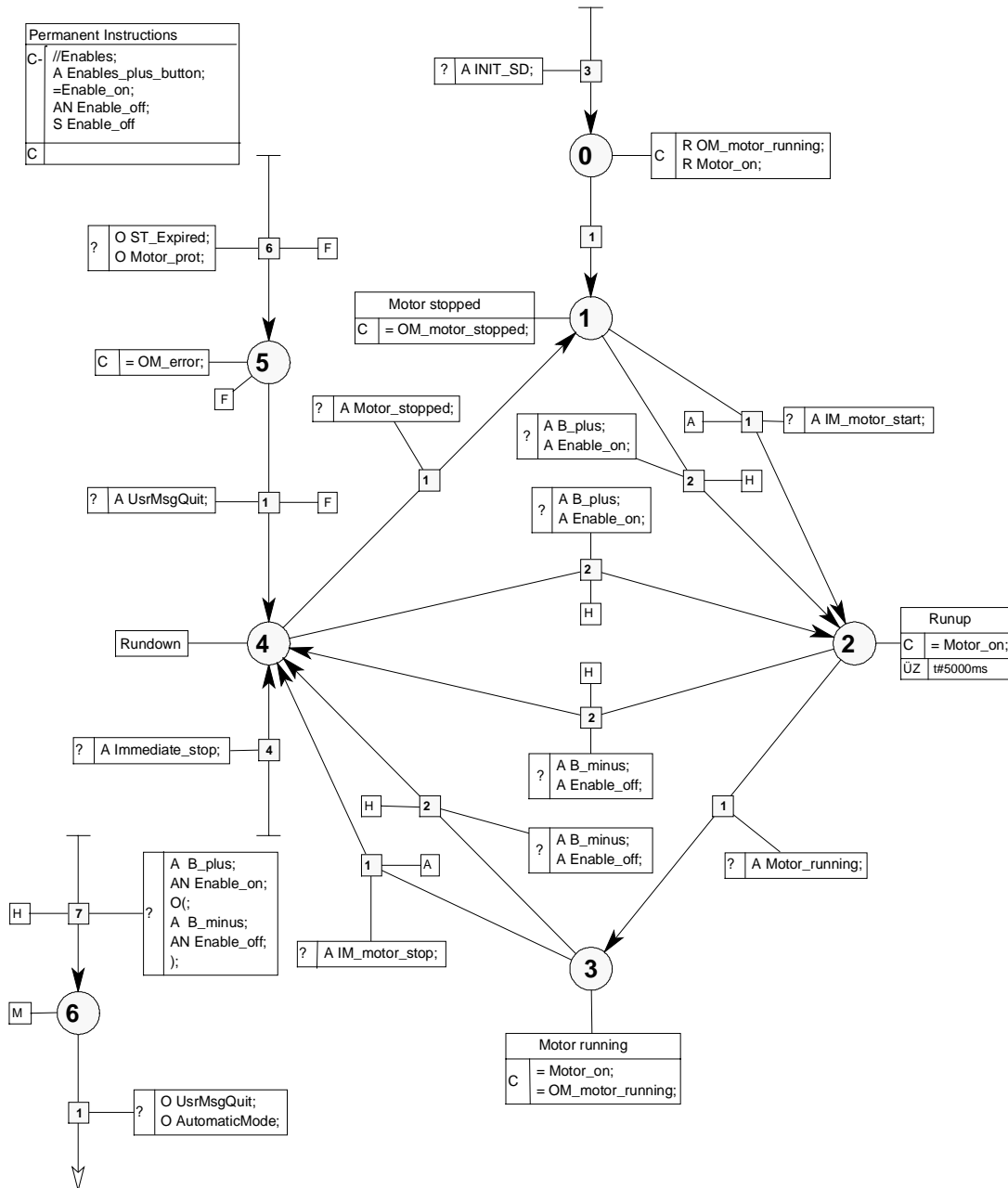
- the execution enable no longer exists (is provided by the "Operation enables" state graphs),
- there is a changeover to the manual/setup operating mode or
- there is an error in the subordinate state graphs.

In the first two cases the "Drilling" state graph branches to state 1 (stop), whereas it branches to the signal state 9 in the latter case.

At a change to the automatic operating mode a direct start is only possible, if the drill unit is in the initial position. Otherwise the button signal "B_initial_pos" must first be used to carry out travelling to the initial position.

6.7.5 Motor State Graph

The Motor state graph contains the functions for the automatic and manual/setup operating modes as well as for monitoring. In the example the state graph is used both for the drill motor and for the cooling agent, meaning that it is instanced twice.



Initialization

The motor is generally switched off during initialization.

Automatic operating mode

In order to implement this operating mode the "Auto" characteristic is assigned to the respective transition (represented in the graphics by a flag with the character A). This means that they can always be activated if the corresponding predefined variable "AutomaticMode" has the signal 1. The signal is set to the automatic and single-step operating modes by the "Operating modes" state graph. In these cases the state graph obtains its commands (messages) from the coordination graph.

Manual/Setup operating mode

The "Manual/Setup" operating mode is implemented by the transitions with the characteristic "manual" (represented in the graphics by a flag with the character H). If this operating mode is selected, the predefined ManualMode variable carries the Signal 1 and the corresponding transitions are enabled.

Controlling of the state graphs can now be carried out by using the button signals. The conditions for activating the motor manually are contained in the permanent instructions. In our example only the condition "Enable_+button" has to be fulfilled to activate the motor.

The "Enable_on" and "Enable_off" signals can be used for indicating purposes. It is possible, for example, to indicate on an operating panel which operator control is currently permitted.

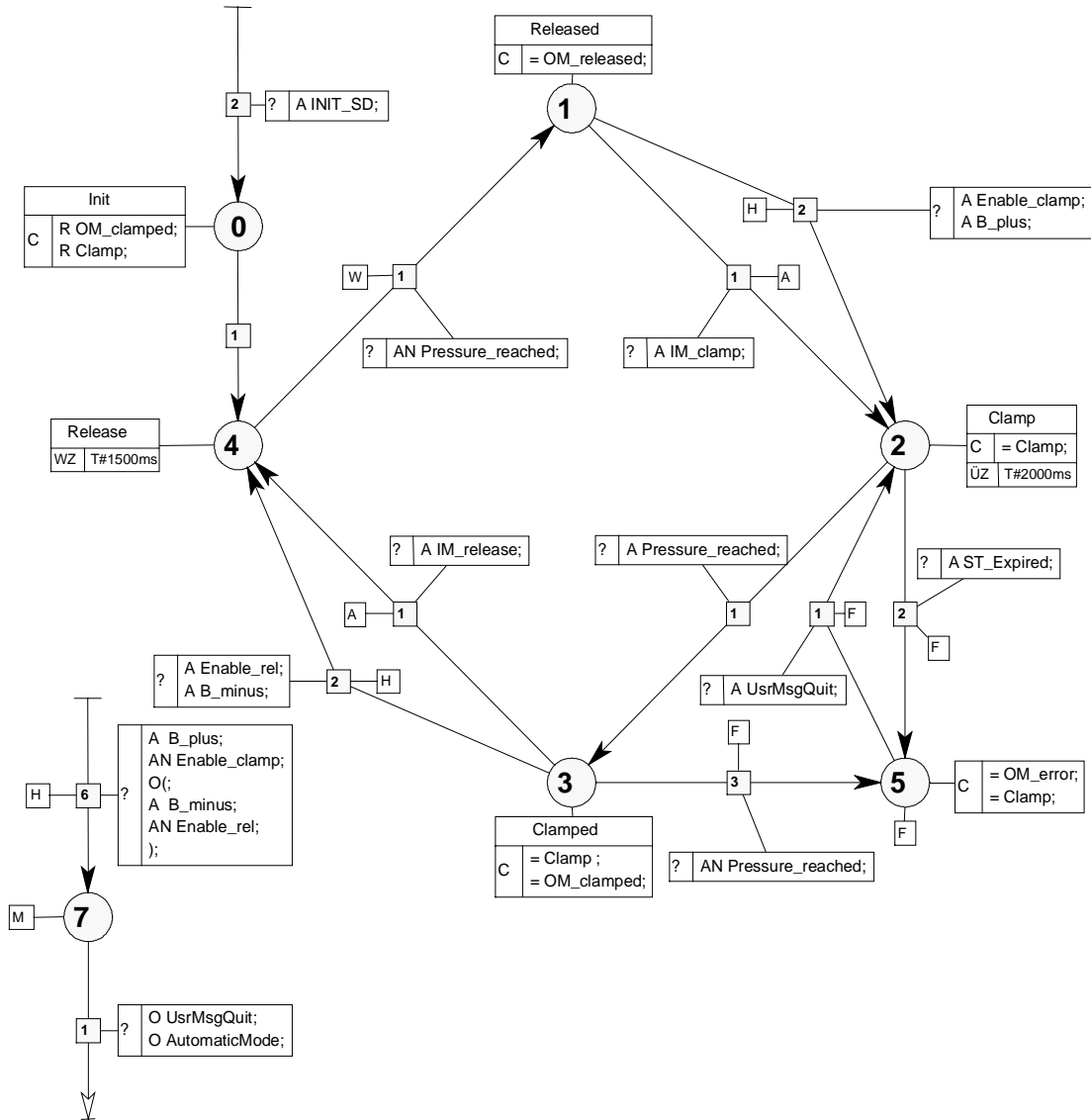
If, for example, the plus button is pressed when activation of the drill motor is not allowed, the system branches to state 6 and a message is output at the operating panel.

Monitoring units

The Motor state graph contains a time monitoring function for starting up. It is defined in the instructions for state 2 (starting up). If the specified time during starting up is exceeded, an error message is entered in the diagnostic buffer of the PLC. This message is acknowledged when you leave state 2. No further reaction to this error is programmed in this example.

6.7.6 Clamp State Graph

The Clamp state graph contains the functions for the automatic and manual/setup operating modes as well as for monitoring.



Initialization

During initialization the system branches to state 4 (release).

Automatic operating mode

In order to implement this operating mode the "Auto" characteristic is assigned to the respective transitions (represented in the graphics by a flag with the character A). This means that they are always active if the corresponding predefined variable "AutomaticMode" has the signal 1. The signal is set to the automatic and single-step operating modes by the "Operating modes" state graph. In these cases the state graph obtains its commands (messages) from the coordination graph.

Manual/Setup operating mode

The "Manual/Setup" operating mode is implemented by the transitions with the characteristic "manual" (represented in the graphics by a flag with the character H). If this operating mode is selected, the predefined ManualMode variable carries the Signal 1 and the corresponding transitions are enabled. Controlling of the state graphs can now be carried out by using the button signals.

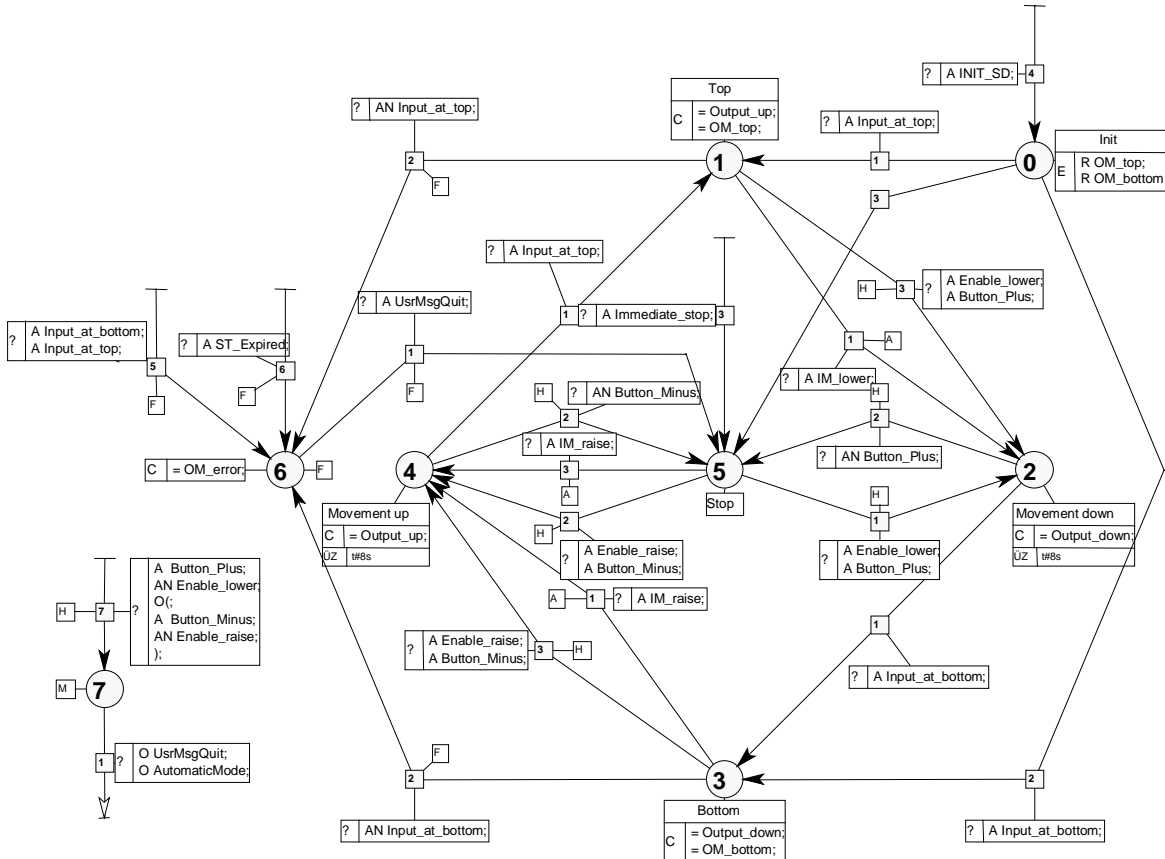
Monitoring units

The following functions are monitored:

- Duration of movement
Monitoring is carried out by specifying a monitoring time at state 2. If the permitted duration of movement is exceeded, the system branches via the Any transition 2 to state 5 by evaluating the variable "ST_Expired" (monitoring time exceeded). This has the effect that the drilling process is interrupted. This error has to be acknowledged in order to continue.
- Clamping pressure in clamping 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.

6.7.7 Valve_2E State Graph

The Valve_2E state graph contains functions for the automatic and manual/setup operating modes as well as monitoring functions and an extended initialization function.



Initialization

During initialization the "Top" and "Bottom" limits are evaluated. If the valve is in a defined limit state, the corresponding state (state 1 or state 3) is entered. If the valve is not in a defined limit state, state 5 (stop) is entered.

Automatic operating mode

In order to implement this operating mode the "Auto" characteristic is assigned to the respective transitions (represented in the graphics by a flag with the character A). This means that they are always active if the corresponding predefined variable "AutomaticMode" has the signal 1. The signal is set to the automatic and single-step operating modes by the "Operating modes" state graph. In these cases the state graph obtains its commands (messages) from the coordination graph.

Manual/Setup operating mode

The "Manual/Setup" operating mode is implemented by the transitions with the characteristic "manual" (represented in the graphics by a flag with the character H). If this operating mode is selected, the predefined ManualMode variable carries the Signal 1 and the corresponding transitions are enabled. Controlling of the state graphs can now be carried out by using the button signals.

Monitoring units

The following functions are monitored:

- Duration of movement
Monitoring is carried out by specifying monitoring times for the states 2 and 4. If the permitted duration of movement is exceeded, the system branches via the Any transition 4 to state 6 by evaluating the variable "ST_Expired" (monitoring time exceeded). This has the effect that the movement is interrupted.
- Error signals of the limit switches (double operation)
The error signals of the limit switches are monitored permanently by the Any transition 3. If both limit switches signal that both limits have been reached, the system branches to state 6. Since the "Error" characteristic is assigned here, this leads to an error message.
- Drifting away from the limit
Drifting away from the limit is monitored by querying the corresponding limit signals in the states 1 and 3 which are assigned to the limits. If the corresponding limit signal changes to "0", the system changes to the error state 6 and an error message is triggered.

Abort situations

When the "Immediate_stop" signal is specified, the system branches via an Any transition to state 5 (stop).

6.7.8 Compiler Settings

In order for the displayed state graphs to execute the desired functions, the following settings have to be carried out (the setting is made with the **Options > Settings for Graph Groups > Compile** menu command):

Graph group "General functions":

The option "Cyclic actions with RLO 0" must be selected. 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.

6.8 Standard Diagnosis Configuration

6.8.1 Information on the Supplied Example

An example of the configuration of the diagnostic inclusion is supplied in the "Drill unit" project included (program container "Drill unit with ProAgent"). It is based on the "Drill unit" example.

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 _DT: For operator panels with direct keys
- Extension _TP: For touch panels

6.8.2 State Graph for Generating the Operating-Mode Signals

Boundary conditions

- The current operating mode is displayed centrally in the overview of ProAgent
- The operating mode is selected by means of switches or buttons at the operating panel

Under these boundary conditions the state graph explained in the previous chapter can be used unchanged to generate the operating mode signals. It is only necessary to configure a corresponding field in which the set operating mode is displayed in the ProAgent overview with ProTool means.

6.8.3 State Graphs with Coordination Function

In ProAgent the state graphs for coordination of a graph group are displayed in the overview. Prerequisite is that these state graphs are available in the S7-HiGraph V5 save format and that the following settings have been carried out:

1. Select the **Options > Settings for Graph Groups / State Graphs** menu command and then select the "Standard diagnosis" register tab.
2. Activate the option "Instances of the state graph visible at the display unit as a unit".

If messages are to be generated in a state graph with coordination function the desired settings for initial value detection as well as for the display acknowledgement are to be carried out at the graph group or the state graph.

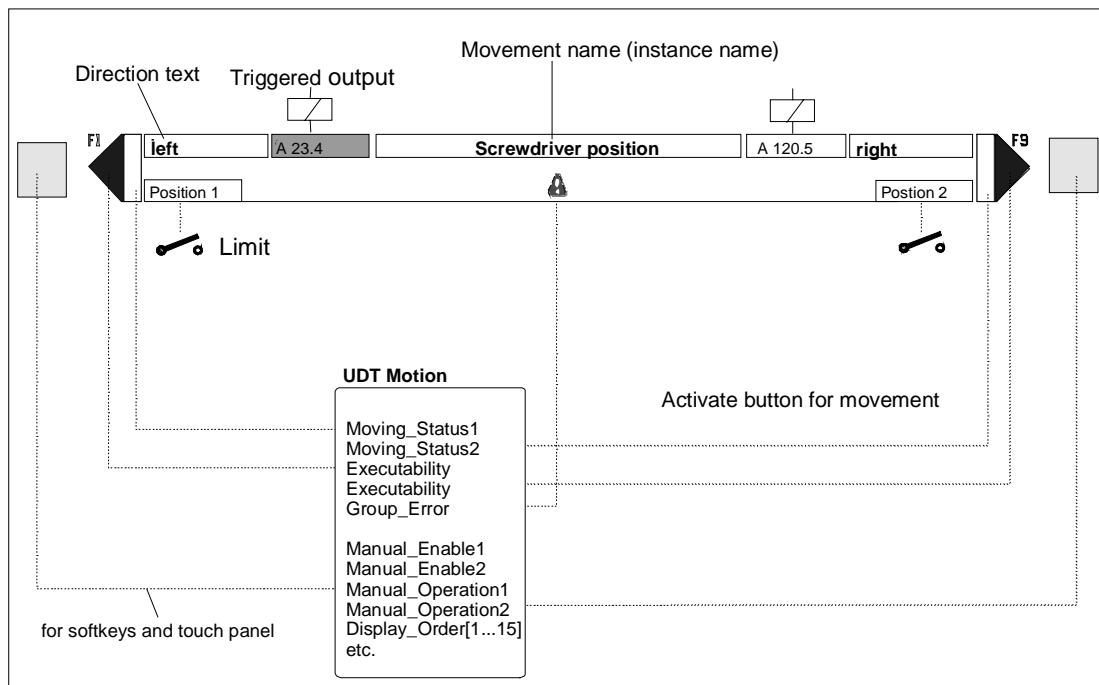
Note

Settings of the save format can be carried out by using the **Options > Application Settings > Save Format** menu command.

6.8.4 State Graphs which Realize a Movement

The movement screen of ProAgent displays "Movements", meaning the states of the function units which carry out a movement at the machine or in the process (such as valves, motors, etc). One "movement line" is reserved for each movement. The movements can be travelled manually during manual operation by using the keys which are assigned to the line.

In S7-HiGraph the display fields of a movement line have to be supplied and the key signals have to be evaluated. One structure (HiGraphMotionUDT) per movement line 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.



Supplying the display fields

- Supply of the "Moving-Status<No.>" signals of the HiGraphSUnitUDT in the movement states
- Supply of the "Executability<No.>" signals of the HiGraphSUnitUDT with the locking and enable conditions for manual operation, for example, in the preceding cyclic actions
- Depending on the design of the keys for manual operation either evaluation of the "Manual_Operation<No.>" signals or of the "Display_Order" signals interconnected with the key signals in the transitions for manual operation.
- The following entries are to be carried out under **Options > Settings for Graph Groups / State Graphs**, "Standard diagnosis" tab card:
 - Activate the option "state graph realizes a movement"
 - 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 addresses for the limits
 - If desired, carry out state-graph-specific settings for the initial-value detection as well as for the display acknowledgement
 - If desired, activate the option "Instances of the state graph visible at the display unit as a unit".

Evaluation of the key signals

The key signals are to be planned in accordance with the OP design. Three variations are to be differentiated for the OP:

Operator panel with soft keys	In the case of soft keys the bits "Manual_Enable1" or "Manual_Enable2" belonging to the movement are set as soon as the keys next to the displayed movement are operated. These signals can then be evaluated in the transitions for manual operation.
Operator panels with direct keys	<p>In the case of direct keys an assignment of the keys to the displayed movements must be carried out 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 in which the respective movement is displayed for the respective displayed movement. This means, for example, that bit 3 is set, if the movement in the 3rd line is displayed. In S7-HiGraph it is advisable to carry out the corresponding link of the direct-key signals with the "Display_Order" signals in the permanent instructions C-. This would take the following form, for example, for the "Direction movement minus" key under the assumption that up to four movements are displayed on the OP:</p> <pre>// Key signals o(u Key_Minus1; u Movement.Display_Order[1];) o(u Key_Minus2; u Movement.Display_Order[2];) o(u Key_Minus3; u Movement.Display_Order[3];) o(u Key_Minus4; u Movement.Display_Order[4];) = ButtonMinus;</pre> <p>Remark: The signals Key_Minus1 to Key_Minus4 are defined as global signals in the symbol table, whereas the signal ButtonMinus is defined as a variable in the VarStat.</p>
Touch panels	<p>Soft keys are used in touch panels. In addition the user must first select the movement which is to be operated in order to increase the operating security. 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 are to be interconnected with the soft key signals Manual_Operation1 and Manual_Operation2.</p> <p>The interconnection for the "Direction movement 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; = ButtonMinus;</pre>

6.8.5 Diagnostic Configuration in the Graph Groups

The following entries are to be carried out under **Options > Settings for Graph Groups / State Graphs > Diagnosis**:

- Activate the "Standard diagnosis" option.
- Enter the name or the number of the diagnostic DB which belongs to the graph group in the "Diagnostic DB" input field.
- If the graph group represents a technological unit which is to be displayed in the overview of ProAgent, activate the option "Graph group visible at the display unit as a unit".
- If an initial-value detection is required for the fault messages and/or operating messages, activate the corresponding fields.

Note:

The initial values which belong to a message are displayed in the detailed view of ProAgent. This means that nothing is displayed in the detailed view if initial-value detection was not activated in S7-HiGraph.

- If a display acknowledgement is required for the fault and/or operating messages, activate the corresponding fields.
- If more than one graph group is required for a technological unit, enter the DB number of the corresponding, interconnected, graph groups in the "Subordinate graph groups" field.

Note

The message texts for the fault and operation messages can be adapted under **Options > Application Settings > Message Texts**.

7 User Program Run Behavior in the PLC

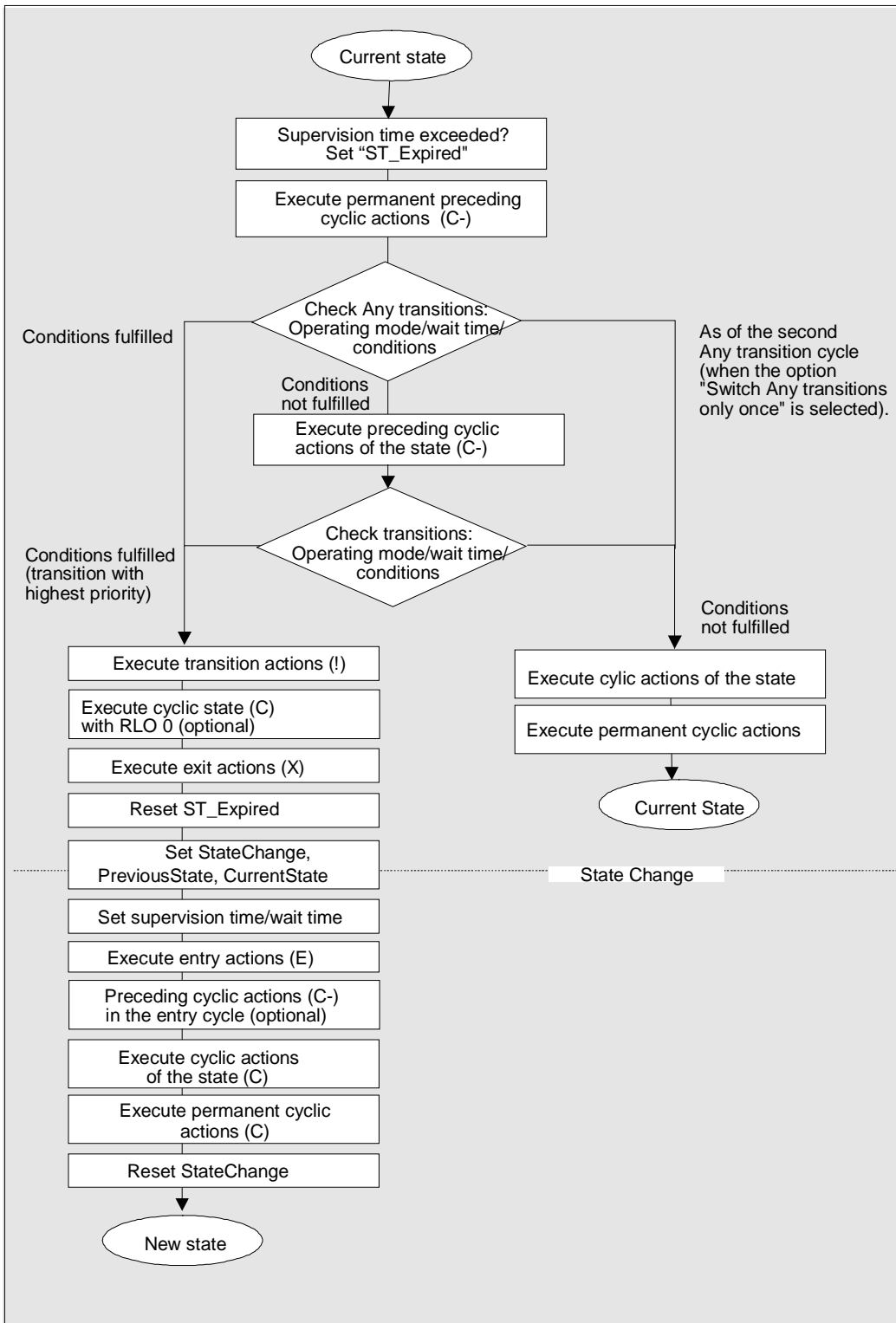
7.1 Cyclic Processing of a State in the PLC

Rules

The following rules always apply:

- Processing of an instruction table always starts with the result of logic operation RLO = 1.
- A state change occurs when:
 - All the conditions of a transition are fulfilled, meaning that the result of logic operation of the conditions of this transition is 1,
 - The correct operating mode is set if an operating mode was programmed,
 - And no waiting time is set, or the waiting time has expired.

A maximum of one state change is carried out per cycle in a state graph.



7.2 Behavior on Startup and Restart

Startup behavior

The startup behavior is realized by an automatically generated ANY transition in the initial state (startup transition") which switches in accordance with a Boolean variable "INIT_SD" specified by S7-HiGraph. The variable is a copy of the formal parameter "INIT_SD" of the FC of the graph group. It should be programmed in OB1 dependent on the system bits for startup and restart so that it is set during the first cycle following startup or restart.

Startup follows the following pattern:

	INIT_SD = 0	INIT_SD = 1
Data block empty (DB reloaded or startup of the PLC)	Resetting of the internal incoming messages Enters the initial state	Resetting of the internal incoming messages Enters the initial state
Data block with valid contents (DB already in use, restarting of the PLC)	Normal operation	Start-up transition causes a state change in the target state of the Any transition (normally initial state)

The incoming internal messages are not reset automatically during a warm restart ("buffered startup").

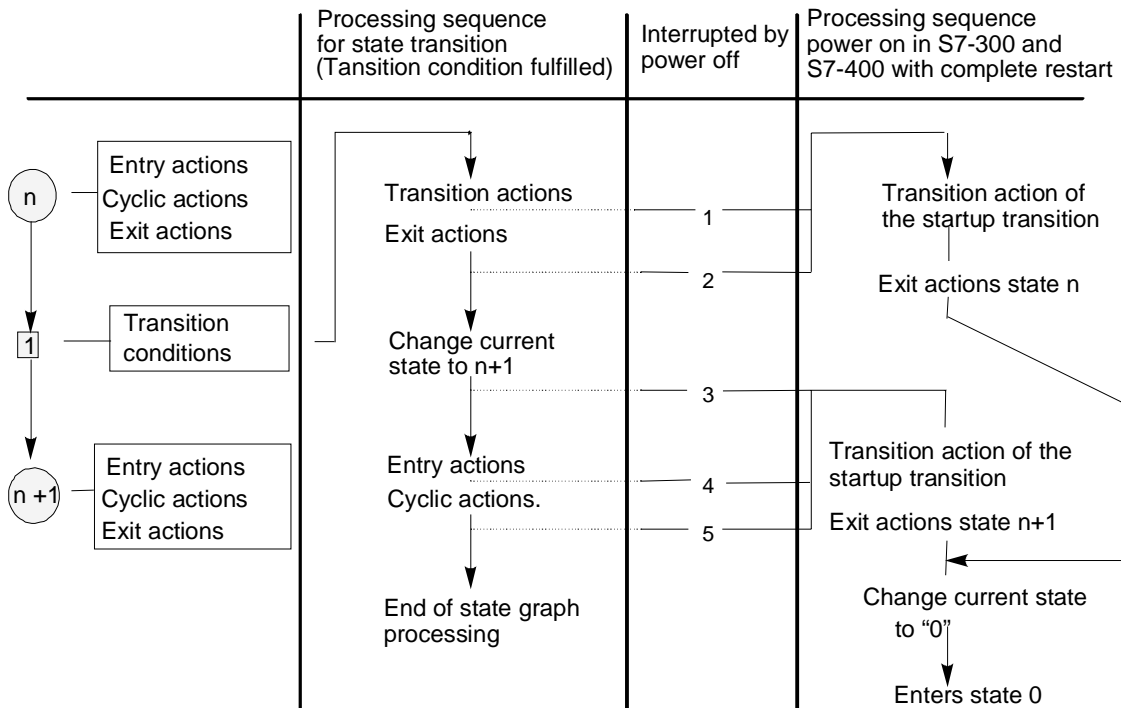
Behavior during power off/on

The behavior of S7-HiGraph at power off/on must be taken into consideration especially for PLCs which start with cold restart behavior after a power on (meaning that the point at which the program execution was interrupted at power off is not taken into consideration). This is the case for all S7-300 and S7-400 PLCs with the system setting "Startup after POWER ON: Cold restart".

- Situation at power off:
 - If no state change occurred at the "power off" point, the behavior is uncritical because a state change to the state 0 from the state reached before power off is carried out at power on (behavior as for a normal state change). If, however, a state change was taking place at the "power off" point, the behavior is no longer so easy to judge, since the program execution can be interrupted at any point in the instruction sequence belonging to the state change.
- Behavior during power on:
 - In the case of PLCs with warm startup the behavior is uncritical, because the program is continued at the interruption point after power on, so that an interrupted state change can also be completed. The prerequisite for this is that the process image update parameter is assigned ("Delete PIQ on Restart" option in Hardware Configuration is deactivated).

If, however, the PLC starts with a cold restart after a power on, the behavior of the state graph depends on the point at which processing was interrupted by a power off.

The following figure shows the behavior of a state graph at power on/off (it is assumed that the value of the formal parameter "INIT_SD" is set to "1", both at a warm restart and a cold restart, meaning that the startup transition is fulfilled):



Rules

- The cyclic actions are executed with RLO=0 between the transition actions and the exit actions if you have set the option "Execute cyclic actions with RLO=0" in the "Compile" register tab (settings).
- All the instruction blocks (for example, input, output and cyclic actions, etc.) whose processing was aborted by a power off, remain incompletely executed.
- Observe the following at an interruption by power off at point 3: After power on a complete state change to the state 0 is carried out with relation to the current state saved before power off. Result: Since the variable "CurrentState" was set to the new state before power off, but the entry action was not executed yet, it is also not executed at power on.
- In the case of PLCs with warm startup behavior an aborted state change is completed and the state n+1 becomes active. Then a state transition from state n+1 to state 0 is executed as a result of the startup transition.

Behavior of the predefined variable

If they exist, the corresponding values are supplied to the predefined variables during the runtime of the state graphs. 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 has been reloaded	Unchanged	Unchanged	Unchanged	Unchanged
After a warm restart of the PLC (buffered startup)	0	Current state before startup	Contents of the variable before startup	TRUE
After a cold restart of the PLC (unbuffered startup)	0	0	0	TRUE

7.3 Memory Requirements of the User Program

Memory space for the FC

Element		Required memory space
Graph group	Fixed length of a graph group without waiting or monitoring times	34 bytes
	Graph groups with any number of waiting and monitoring times	82 bytes
State graphs	Fixed length of a state graph	52 bytes
	State graphs with any number of waiting times	An additional 22 bytes
	State graphs with any number of monitoring times	An additional 56 bytes
	State graphs with any number of message states	An additional 60 bytes
	State graphs with permanent instructions	In addition the length of the permanent instruction. A mean value of 4.3 bytes applies per STL instruction
States	Fixed length of a state	16 bytes
	States with waiting time	An additional 8 bytes
	States with monitoring time	An additional 8 bytes
	States with entry, exit and cyclic actions	In addition the length of the entry, exit and cyclic actions. A mean value of 4.3 bytes applies per STL instruction
	Diagnosis-relevant states* at standard diagnosis	An additional 12 - 16 bytes
Transitions	Fixed length of a transition	20 bytes
	Transition with waiting time	An additional 8 bytes
	Transition with MANUAL or AUTOMATIC operating mode	An additional bytes
	Transitions with actions or conditions	An additional 4 bytes plus the size of the actions or conditions. A mean value of 4.3 bytes applies per STL
	Diagnosis-relevant transitions* at standard diagnosis	An additional 6 - 8 bytes
Any or Return transitions	Fixed length of Any or Return transitions	24 bytes
	Any or Return transitions with waiting time	An additional 12 bytes
	Any or Return transitions with MANUAL or AUTO mode	An additional 8 bytes
	Any or Return transitions with actions or conditions	An additional 4 bytes plus the size of the actions or conditions. A mean value of 4.3 bytes applies per STL

* Diagnosis-relevant states: States in which a monitoring time was defined or which have the characteristic F or M.

Diagnosis-relevant transitions: Transitions which lead out of a state with monitoring time or which lead to a state with characteristic F or M.

Note

The above listing only applies if all the states are numbered in ascending order without gaps beginning with 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 diagnosis is used)
- User-defined variables in the declaration section STAT
- Reserve memory which you have entered in the "Compile" register tab (**Options > Settings** menu command).

Memory space for the diagnostic DB (if standard diagnosis 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 diagnosis-relevant transitions are saved in the DB)	Fixed length of a transition	4 bytes per instance in which the transition is used
	STL instructions in diagnosis-relevant transitions	An additional 2-6 bytes per instruction
	At a first-up detection	An additional 1 Bool per instruction
States (only diagnosis-relevant states are saved in the DB)	Fixed length of a state	12 bytes + 3 Bools per instance in which the state is used. The Bool variables lie in 3 arrays.

8 Tips and Tricks

How can one reduce the program size?

Reducing the DB size

1. Select the **Options > Settings** menu command.
2. Activate the option "Restructure data block" in the "Compile" register tab.
3. Compile and download the program again.
(Caution: You must switch the CPU to STOP mode in order to download the program with the restructured data block.)

Reducing the FC size

- Number the state numbers in ascending order without gaps and begin from zero in each state graph.
- Use block calls (CALL) very sparingly, or at least reduce the number of their parameters.
- Use waiting and monitoring times sparingly.
- Select the **Options > Settings** menu command and deactivate the option "Switch Any transition only once" in the "Compile" register tab.
We recommend instead that you explicitly reset those conditions which caused the Any transition to be switched by using a reset command (for example, in the transition action).
- Select the **Options > Settings** menu command and deactivate the option "Execute cyclic actions with RLO=0" in the "Compile" register tab.
We recommend that you explicitly reset the signals instead (for example, in the output action).
- Deactivate the predefined variables which you do not require in your program. For this purpose select the variable in the variable declaration window and then select the **Edit > Properties** menu command. Then select the "Attributes" register tab in the dialog box and assign the value "false" to the "S7_active" attribute.

How can I find out whether an S7-HiGraph source agrees with a program running ONLINE?

During compilation the generated block is assigned the same time stamp as the source last modified (state graph or graph group). This means you can determine reliably whether an archived project version (for example, on EOD) is the same as the content of the programmable controller or not.

Proceed as follows:

1. First check in the SIMATIC Manager whether the offline block agrees with the ONLINE block (**PLC > Online/Offline Comparison** or **Options > Block comparison** menu command (as from V5)).
2. Once you have determined that the offline block is the same as the block on the programmable controller, you must establish whether the graph group has been compiled and is up-to-date. To do so, display the properties for the relevant graph group and check the time stamp of the most recent modification in the "Source files" register tab. Compare this with the modification date of the off-line block. If they are the same, the existing S7-HiGraph source agrees with a program running ONLINE.

A graph group which was compiled error-free does not run in the CPU

If the function **"Monitoring the program status"** displays that all the state graphs are in the state zero:

- Check whether you have also downloaded the block which calls the S7-HiGraph FC (for example, OB 1), to the CPU.
- Check whether you have programmed the call of the S7-HiGraph FC in it.

If the CPU displays **STOP** and/or **SF**:

- Check whether you have downloaded the S7-HiGraph FC.
- Check whether the calling block does not call the S7-HiGraph FC but rather another FC which is not downloaded.
- Check whether all the FCs called by the graph group are downloaded.
- Check whether all the DBs referenced in the graph group are downloaded.

The function **"Monitoring the program status"** cannot be selected (message **"Non-correctable error 02"**)

You have selected variables in the "Select variables" dialog box whose current parameters refer to non-existing data blocks. Delete these variables from the list.

Can S7-HiGraph sources also be printed out with printers which do not have a Postscript functionality?

When printers without Postscript functionality are used the printing image depends on the used printer/driver and on the settings used. Thus we recommend, for example, the following settings:

- HP Laserjet 4SI under WIN 95: Graphics as vector
- HP Laserjet 4mPlus under WIN NT: Text as graphics; without spooling from metafile
- HP Laserjet 5SI MX under WIN 95: Grid graphics and text
- HP Laserjet 5SI MX under WIN NT: Text as graphics; without spooling from metafile

Graph group takes a long time to open and compile

Remove unused declarations from the variable declaration windows.

No reference data are displayed in a project with S7-HiGraph blocks

Reference data cannot be generated by S7-HiGraph blocks while S7-HiGraph is running in the program status. Terminate the program status in S7-HiGraph!

A converted HiGraph V2.7 source cannot be compiled without errors

In HiGraph V2.7 it was possible to supply current parameters of the type WORD or DWORD to formal parameters of a called FC of the type INT or DINT when the FC is called from HiGraph.

As from S7-HiGraph V4.0 this is no longer possible. Such constellations result in a "type conflict" error during compilation.

Adapt the data types of the formal and current parameters to each other.

Automatic resetting of all signals which were set during a state

1. When programming the state graphs use a separate instruction block for each assignment.
2. Before compiling the graph group select the option "Execute cyclic actions with RLO=0" in the "Compile" register tab (**Options > Document Settings** menu command).
This option has the effect that the cyclic actions of a state graph are executed once more with RLO=0 when the state graph is left, and thus that all the signals which were set during the state are reset.

Inactive variables lead to the compiler message "Variable xy does not match a declaration neither is xy in the symbol table"

If the above error message is displayed although the respective variable is entered in the variable declaration, the reason may be that it is not active.

In order to activate the variable, select the "Object Properties" menu command from the pop-up menu (right-hand mouse button) and enter the value "true" for the "S7_active" attribute in the "Attributes" register tab.

Glossary

A

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.

C

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.

D

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.

F

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.

G

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 graphs, which is executed cyclically during the program execution.

I

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.

Instruction

An instruction 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.

M

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 diagnosis.

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).

O

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.

P

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 diagnosis

Localization of errors in the process (outside the PLC). For process error diagnosis 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.

R

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.

S

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.

T

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.

V

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 card 3-6
 "Instructions" tab card 3-6

)

) 5-7

*

*D 5-7
 *I 5-7
 *R 5-7

/

/D 5-7
 /I 5-7
 /R 5-7

+

+ 5-7
 +D 5-7
 +I 5-7
 +R 5-7

=

= 5-7
 ==D 5-7
 ==I 5-7
 ==R 5-7

A

A 5-7, 5-8, 5-9
 A(..... 5-7
 ABS 5-7
 Acknowledgement obligation for messages .. 4-4
 ACOS 5-7
 Actions 3-15, 3-26, 3-28, 5-7
 AD 5-7

Aligning graphics objects 3-18
 AN 5-7
 AN(..... 5-7
 Any transition 3-22
 Application example 2-2
 Arranging windows 3-6, 3-7
 Arranging working windows 3-6, 3-8
 ASIN 5-7
 Assigning characteristics 3-20
 ATAN 5-7
 Authorization 1-6, 1-7
 Authorization diskette 1-6, 1-7
 Authorization to use 1-6
 AUTHORSW.EXE 1-6
 AutomaticMode 3-13
 AW 5-7

B

Basics of programming with HiGraph 3-1, 3-2
 Behavior on startup and restart 7-3
 Block structure 1-1
 BOOL 5-10
 BTD 5-7
 BTI 5-7
 Byte 5-10

C

CAD 5-7
 CALL 5-7
 CAR 5-7
 CAW 5-7
 CD 5-7
 CHAR 5-11
 Character (CHAR) 5-10
 Characteristics 3-8, 3-20, 3-24
 CLR 5-7
 Cold restart 7-3, 7-4, 7-5
 Compatibility with previous
 HiGraph versions 3-55
 Compilation 3-42
 Conditions 3-15, 3-28, 5-7

- Conversion of programs from
 - HiGraph 2.6/2.7 3-54
 - Copying
 - State 3-21
 - Transition 3-25
 - COS 5-7
 - Counter 5-10
 - Creation of diagnostic data (steps) 4-3, 4-11
 - Criteria analysis 4-5
 - CU 5-7
 - Current parameters
 - Current parameters for messages 3-37
 - Exchanging messages between
 - state graphs 3-37
 - Graph groups
 - Programming messages 3-37
 - Interaction between variable declarations
 - and current parameter assignments 3-16
 - Messages 3-37
 - Printing 3-52
 - User interface 3-5
 - CurrentState 3-13, 3-15
 - Cutting
 - State 3-21
 - Transition 3-24
 - Cyclic actions 3-28
 - Cyclic processing of a state graph 7-1
- D**
- D 5-7
 - Data types 5-10
 - DATE 5-10, 5-11
 - Date and time (DATE_AND_TIME) 5-10
 - DATE_AND_TIME 5-10
 - Debug functions of STEP 7 3-50, 3-51
 - Debugging 3-47
 - Monitoring the program status 3-47, 3-49
 - DEC 5-7
 - Declaring variables
 - Columns in the variable
 - declaration window 3-11
 - Predefined variables 3-12, 3-13
 - Steps for entering the variable
 - declaration 3-38
 - Variable declaration window 3-5, 3-10
 - Diagnosis 4-1
 - Diagnosis via format converter 4-9, 4-10, 4-11
 - Diagnostic messages 4-4, 4-9
 - DINT 5-10
 - Displaying and controlling movements
 - in the movement screen 4-6
 - Displaying messages in the
 - message screen 4-4
 - Displaying reference data 3-40, 3-41
 - Displaying units in the overview screen 4-8
 - Double word (DWORD) 5-10
 - Downloading 3-45, 3-46
 - Downloading for the first time 3-45
 - Downloading the user program 3-45, 3-46
 - Drawing and positioning aids 3-8
 - Drilling machine example 2-2
 - DTB 5-7
 - DTR 5-7
 - DWORD 5-10
- E**
- Emergency authorization 1-6
 - Enlarging and reducing the view 3-7
 - Entering the priority 3-23
 - Entry actions 3-28
 - Example program 2-2
 - Exchanging messages between
 - state graphs 3-37
 - EXP 5-7
- F**
- Fault messages 4-4, 4-5
 - FN 5-7
 - Fonts 3-8
 - Formal parameters 3-1
 - Format converter diagnosis 4-9, 4-10, 4-11
 - FP 5-7
 - FR 5-8
- G**
- Generating and displaying reference data 3-40
 - Graph 3-4
 - Refer to state graph 3-4
 - Graph groups
 - Compilation 3-42, 3-43
 - Compiling 3-43, 3-44
 - Planning information 6-6
 - Printing 3-52, 3-53
 - Programming messages 3-37
 - Programming with absolute or symbolic
 - addresses 3-36
 - Grid 3-8
- H**
- HiGraph 1-1, 1-2, 1-3

installation	1-6
Starting.....	3-4
User interface	3-5
HiGraph 2.6 / 2.7	3-54
HiGraph Version 4.0/4.01.....	3-55
HiGraphErrEmitterFB (FB 20)	4-11
HiGraphMsgEmitterFB (FC 101)	4-11
HiGraphUnivEmitterFC (FC 102).....	4-3

I

-I	5-7
IEC date (DATE).....	5-10
IEC time (TIME).....	5-10
INC	5-7
INIT_SD	3-13, 3-45
Initial state	3-19
Initializing	
Power on/off	7-4
Inserting states	3-19
Installation	1-6
Instance	3-34
Inserting	3-34
Programming with instances.....	3-1
Instance concept	3-1
Instructions.....	3-15, 3-26, 3-27, 3-28, 3-29, 3-30, 5-7
Instructions in STL.....	5-7
Sorted by mnemonics	5-7
INT.....	5-10
Integer (INT).....	5-10
Integer 32 bits (DINT)	5-10
Interaction between S7 HiGraph	
the automation system and the OP	4-2
the automation system and the OP	
(format converter)	4-10
Interaction between variable declarations	
and current parameter assignments.....	3-16
Interaction between variable declarations	
and instructions	3-15
Internet.....	vi
Introduction	1-1
INVD	5-7
INVI	5-7
ITB.....	5-7
ITD.....	5-7

K

Knowledge	
Required.....	iii

L

L	5-7
LC	5-7
Limit.....	4-6
Linking the incoming and outgoing	
messages.....	3-38
LN	5-7

M

ManualMode	3-13
Message screen (diagnosis)	4-4
Message window.....	3-5
Messages	3-37
Migration.....	3-54
Mnemonics	3-30, 5-7
MOD.....	5-7
Monitoring and controlling variables	3-50
Monitoring the program status	3-47, 3-49
Monitoring time	3-32, 4-4, 4-9
Copying	
State	3-21
Exceeding	4-4, 4-9
Monitoring times.....	3-28
Movement screen (diagnosis)	4-6
Multiple use of state graphs	3-1

N

NEGD.....	5-7
NEGI	5-7
NEGR.....	5-7
New functions in HiGraph V5.0	1-4
NOT.....	5-8, 5-9

O

O	5-7
O(.....	5-7
OD.....	5-7
ON.....	5-7
ON(.....	5-7
Online help	iv
Opening state graphs	3-4
Operating modes	
(planning information)	6-11, 6-13
Operation enables (planning information)	6-10
Operation messages	4-4
Overview screen (diagnosis).....	4-8
OW.....	5-7

- P**
- Page numbering 3-52
 - Page settings 3-53
 - Planning the standard diagnostics with
 - a transfer line as an example 6-21, 6-22
 - Planning with a transfer line as
 - an example 6-1, 6-2
 - PLC
 - Downloading to 3-45, 3-46
 - Downloading to 3-45
 - Downloading to 3-45, 3-46
 - Pointer 5-10
 - Points of use of addresses in the program .. 3-41
 - POP 5-7
 - Positioning the working window 3-7
 - Power on/off
 - Behavior during power on/off 7-3
 - Prerequisite for program creation 3-3
 - Prerequisite for standard diagnosis 4-3
 - PreviousState 3-13, 3-15
 - Print settings 3-53
 - Applicable to the entire application 3-52
 - Graphic or textual display of objects 3-52
 - Sequence of print objects 3-52
 - Zoom factor 3-53
 - Printing 3-9, 3-52, 3-53
 - Printing steps 3-52
 - Priority class 3-23
 - Refer to "Priority" 3-23
 - ProAgent 4-1
 - Process error diagnosis 4-1, 4-2, 4-3, 4-4, 4-6, 4-8, 4-9, 4-10, 4-11
 - Program status 3-47, 3-49
 - Program structure 1-1
 - Programming a state
 - Assigning characteristics 3-20
 - States 3-19
 - Programming a state graph 3-4
 - Assigning a functional unit and a state graph (drilling machine example) .. 2-7
 - Designing the state graphs (drilling machine example) 2-7
 - Determining the program structure (drilling machine example) 2-6
 - Determining the required state graphs (drilling machine example) 2-6
 - Opening 3-4
 - Programming absolutely or symbolically 3-36
 - Programming operating modes 3-33
 - Programming state graphs
 - Basics of programming 3-1
 - Programming states
 - Inserting 3-19
 - Programming symbolically 3-36
 - Programming with symbolic addresses 3-36
 - ProTool/ProAgent 4-1
 - PUSH 5-7
- R**
- R 5-7
 - R 5-7
 - Rapid positioning at points of use
 - in the program 3-41
 - REAL 5-10
 - Real (REAL) 5-10
 - Reference to literature iv
 - Reloading 3-46
 - Reloading changes ONLINE 3-46
 - Return transition 3-22
 - RLD 5-7
 - RLDA 5-7
 - RND 5-7
 - RND- 5-7
 - RND+ 5-7
 - RRD 5-7
 - RRDA 5-7
 - Run sequence 3-35
- S**
- S 5-7
 - S5TIME (SIMATIC time) 5-10
 - S7 HiGraph 1-1
 - S7 HiGraph-specific abbreviations in the reference data 3-41
 - S7_message 3-38
 - Save format 3-55, 3-56
 - Saving 3-42
 - Saving state graphs and graph groups 3-42
 - SD 5-7
 - SE 5-7
 - SET 5-8, 5-9
 - Setting colors 3-8
 - Setting fonts in working windows 3-8
 - Setting headers and footers 3-53
 - Setting the drawing area 3-7
 - Setting the page settings 3-53
 - Setting the printer 3-53
 - Setting up a project 3-3
 - Setting up a STEP 7 project 3-3
 - Settings 3-7, 3-8, 3-43, 3-53
 - SF 5-7
 - SIMATIC time 5-10
 - SIN 5-7

- SLD..... 5-7
 - SLW..... 5-7
 - SP..... 5-7
 - SQR..... 5-7
 - SQRT..... 5-7
 - SRD..... 5-7
 - SRW..... 5-7
 - SS..... 5-7
 - SSD..... 5-7
 - SSI..... 5-7
 - ST_CurrValue..... 3-14
 - ST_Expired..... 3-13
 - ST_ExpiredPrev..... 3-13
 - ST_Stop..... 3-14, 3-15
 - ST_Valid..... 3-14
 - Standard diagnosis via
 - ProTool/ProAgent .. 4-1, 4-2, 4-3, 4-4, 4-6, 4-8
 - Starting HiGraph..... 3-4
 - Startup behavior..... 7-4
 - Startup transition..... 3-22
 - State graph..... 3-17, 7-1
 - Creating/opening..... 3-4
 - Printing..... 3-52, 3-53
 - Saving..... 3-42
 - State name
 - number and comment..... 3-20
 - StateChange..... 3-13, 3-15
 - Status..... 3-47, 3-48, 3-49
 - Steps for assigning the message type..... 3-38
 - Steps for creating a HiGraph program..... 3-2
 - Steps for programming the
 - statements for messages..... 3-38
 - STL instructions..... 5-7
 - String..... 5-11
 - Structure of a state graph..... 3-17
 - Symbol table (drilling machine example)..... 2-11
- T**
- T..... 5-7
 - TAK..... 5-7
 - TAN..... 5-7
 - Template mechanism..... 3-1
 - Refer to "Instance concept"..... 3-1
 - Test functions of STEP 7..... 3-40
 - Testing
 - Displaying reference data..... 3-40
 - Test functions of STEP 7..... 3-40
 - TIME..... 5-10, 5-11
 - Time (TIME_OF_DAY)..... 5-10
 - TIME_OF_DAY..... 5-10
 - Timer..... 5-10
 - Transition actions..... 3-28
 - Transitions..... 3-22
 - Copying Moving Deleting..... 3-24
 - Specifying priorities..... 3-23
 - Transition characteristics..... 3-24
 - Transition name..... 3-23
 - Transition properties..... 3-23
 - TRUNC..... 5-7
- U**
- UDT_Motion..... 4-3, 4-6
 - UDT_Unit..... 4-3
 - Unit overview (diagnosis)..... 4-8
 - User interface..... 3-5
 - Adapting the user interface..... 3-7, 3-8
 - Using state graphs and graph groups..... 3-1
 - UsrMsgQuit..... 4-11
- V**
- Validity of the manual..... iii
 - Variable declaration..... 3-11
 - Interaction between variable
 - declarations and instructions..... 3-15
 - Interdependency between variable
 - declarations and current parameter
 - assignments..... 3-16
 - Printing..... 3-52
 - Variable declaration window..... 3-5
 - Version 4.0/4.01..... 3-55
 - View
 - Enlarging and reducing..... 3-7
 - Volume of project data..... 3-17
- W**
- Waiting times..... 3-28
 - Warm restart..... 7-3, 7-4, 7-5
 - What has changed from V4.01 to V5.0?..... 1-4
 - WORD..... 5-10
 - Word (WORD)..... 5-10
 - WT_CurrValue..... 3-14
 - WT_Expired..... 3-14
 - WT_Valid..... 3-14

X

X 5-7
X(..... 5-7
XN..... 5-7
XN(..... 5-7

XOD..... 5-7
XOW 5-7

Z

Zooming..... 3-7

Siemens AG
A&D AS E 81
Oestliche Rheinbrueckenstr. 50
76181 Karlsruhe

From:

Your Name:.....

Your Title:

Company Name:.....

Street:

Country:

Phone:

Please check any industry that applies to you:

- | | |
|--|---|
| <input type="checkbox"/> Automotive | <input type="checkbox"/> Pharmaceutical |
| <input type="checkbox"/> Chemical | <input type="checkbox"/> Plastic |
| <input type="checkbox"/> Electrical Machinery | <input type="checkbox"/> Pulp and Paper |
| <input type="checkbox"/> Food | <input type="checkbox"/> Textiles |
| <input type="checkbox"/> Instrument and Control | <input type="checkbox"/> Transportation |
| <input type="checkbox"/> Nonelectrical Machinery | <input type="checkbox"/> Other |
| <input type="checkbox"/> Petrochemical | |

Remarks Form

Your comments and recommendations will help us to improve the quality and usefulness of our publications. Please take the first available opportunity to fill out this questionnaire and return it to Siemens.

Please give each of the following questions your own personal mark within the range from 1 (very good) to 5 (poor).

- 1. Do the contents meet your requirements?
- 2. Is the information you need easy to find?
- 3. Is the text easy to understand?
- 4. Does the level of technical detail meet your requirements?
- 5. Please rate the quality of the graphics/tables:

Additional comments:

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....