SIEMENS

SIMATIC

S7 PDIAG for S7-300 and S7-400 Configuring Process Diagnostics for LAD, STL, and FBD

Manual

Edition 07/2000
A5E00063001-01

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

The device/system may only be set up and operated in conjunction with this manual.

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

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 NET® and SIMATIC HMI® are registered 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.

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

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.

Important Notes

Purpose of the Manual

This manual and the online help for S7 PDIAG provide you with all the information you require in order to diagnose processes for the programming languages Statement List (STL), Ladder Logic (LAD), and Function Block Diagram (FBD).

This manual contains an example which shows you how to work with S7 PDIAG and describes step-by-step the procedure for configuring error definitions with S7 PDIAG.

The appendix contains information about the language elements which can be used in S7 PDIAG for programming general monitoring and motion monitoring. Both the syntax and the functions of the respective language elements which you require in order to create monitoring logic are described. This section also tells you how S7 PDIAG supports you when programming and what rules to observe.

How to Read this Manual

We recommend that you first carry out the example in Chapter 3 (Getting Started with S7 PDIAG) using the software in order to see how easy it is to start working with process diagnostics and to gain a quick overview of the procedure involved.

Another more complex example of using monitoring types in S7 PDIAG is described in Appendix C of this manual.

Use the remaining sections of the manual as and when you require them for your tasks.

Audience

This manual is intended for STEP 7 programmers and persons who are configuring, commissioning, or servicing automation systems.

General knowledge of automation technology and experience of using the STEP 7 Standard software are required.

Where is this Manual Valid?

This manual is valid for the S7 PDIAG process diagnostics software, version 4.0.

How to Use this Manual

Certain conventions have been used in this manual in order to provide easy access to information:

- References to further information on topics which appear in other chapters are denoted by (see Section x.y). References to other manuals are shown in italics.
- Each topic either describes the functions of the tool or provides information on necessary or recommended procedures.
- References to other manuals are shown using the part number of the literature between slashes /.../. Using these numbers you can find out the exact title of the manual from the literature list in the STEP 7 User Manual.

Guidelines

This manual requires knowledge of S7 programs which you can read about in the Programming Manual **/234/**. Since the S7 PDIAG process diagnostics software is based on the STEP 7 Standard software, you should also be familiar with using the Standard software which is described in the STEP 7 User Manual **/231/**.

This manual is structured as follows:

- Chapter 1 introduces you to process diagnostics with S7 PDIAG. It describes the functional capabilities and the main advantages of S7 PDIAG. It also provides you with an overview of the individual configuration steps and the general procedure when working with S7 PDIAG.
- Chapter 2 describes the operational requirements you should take into consideration and how to install the process diagnostics software and the authorization.
- Chapter 3 describes how to get started with S7 PDIAG. Using a simple example, it shows you how to monitor your process with address monitoring on your CPU. The example then takes you through the first steps with ProAgent, the configuration software for display devices.
- Chapter 4 shows you step by step how to configure address monitoring.
- Chapter 5 shows you step by step how to configure general monitoring
- Chapter 6 shows you step by step how to configure motion monitoring.
- Chapter 7 describes how to generate the monitoring blocks required for process diagnostics and how to download them to your programmable logic controller.
- Chapter 8 describes how to print out the diagnostic data created with S7 PDIAG and how you can export them, if necessary.
- Chapter 9 shows you how to create user-defined templates and modify times offline and online, and how you work with auxiliary process values and exclusion addresses when using advanced programming. In addition you learn how to group units and to search for objects in S7 PDIAG so that they can be edited. We also inform you on the reference data created by S7 PDIAG and on the new graphics motion screen as an interface to the display device.

- Appendix A provides important information on programming your user program.
- Appendix B introduces the language elements of S7 PDIAG and their syntax. With the help of these language elements, you can program your own monitoring logic for general monitoring and motion monitoring.
- Appendix C uses the example of a drill to illustrate motion monitoring with S7 PDIAG. From this example, you will become familiar with all the functions of process diagnostics.
- Appendix D contains tips and tricks to help you when working with S7 PDIAG.

Online Help

An online help is integrated in the software to complement this manual. This online help is intended to provide detailed support when using the software, and can be activated from the help menu or by pressing the F1 key.

Additional Assistance

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 S7 PDIAG 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 of the icons in the toolbar is also displayed if the cursor is positioned briefly on the icon.

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.

SIMATIC Training Center

We offer corresponding courses to help familiarize you with the SIMATIC 7 PLC. Please contact your regional training center or the central training center in:

D-90327 Nuremberg, Tel. +49 (0)911 / 895 3200.

Customer Support, Technical Support

Available worldwide at all times:

SIMATIC Hotline

Worldwide (Nuremberg) **Technical Support** (FreeContact) Local time: Mo.-Fr. 7:00 to 17:00 Phone: +49 (180) 5050-222 Fax: +49 (180) 5050-223 Email: techsupport@ad.siemens.de GMT: +1:00	**Worldwide** (Nuremberg) **Technical Support** (with costs, may only be used with SIMATIC Card) Local time: Mo.-Fr. 0:00 to 24:00 Phone: +49 (911) 895-7777 Fax: +49 (911) 895-7001 GMT: +01:00	
Europe / Africa (Nuremberg) **Authorization** Local time: Mo.-Fr. 7:00 to 17:00 Phone: +49 (911) 895-7200 Fax: +49 (911) 895-7201 Email: authorization@nbgm.siemens.de GMT: +1:00	**America** (Johnson City) **Technical Support and Authorization** Local time: Mo.-Fr. 8:00 to 19:00 Phone: +1 423 461-2522 Fax: +1 423 461-2289 Email: simatic.hotline@sea.siemens.com GMT: -5:00	**Asia / Australia** (Singapore) **Technical Support and Authorization** Local time: Mo.-Fr. 8:30 to 17:30 Phone: +65 740-7000 Fax: +65 740-7001 Email: simatic.hotline@sae.siemens.com.sg GMT: +8:00
English and German are generally used at the SIMATIC Hotlines. French, Italian and Spanish are also spoken at the Authorization hotline.		

SIMATIC Customer Support Online Services

The SIMATIC Customer Support offers you extensive additional information on the SIMATIC products via the online services:

- General current information can be obtained
  - In the **Internet** under `http://www.ad.siemens.de/simatic`
- Current product information and downloads as an additional help:
  - in the **Internet** under `http://www.ad.siemens.de/simatic-cs`
  - Via the **Bulletin Board System** (BBS) in Nuremberg (*SIMATIC Customer Support Mailbox)* under der Nummer +49 (911) 895-7100.

    To dial the mailbox select a modem with up to V.34 (28.8 kbauds) whose parameters are set as follows: 8, N, 1, ANSI, or dial in via ISDN (x.75, 64 kBit).
- Your local partner for Automation & Drives can be found by using the partner database
  - in the **Internet** under `http://www3.ad.siemens.de/partner/search.asp`

Contents

Introduction to S7 PDIAG

1

In This Chapter

This chapter introduces you to process diagnostics and will familiarize you with the terms and performance range of S7 PDIAG.

In addition, this chapter describes the standard procedure for working with process diagnostics and how S7 PDIAG also supports you in troubleshooting on the display device.

Chapter Overview

1.1 Introduction to Process Diagnostics

Introduction

System and process operators continually need to minimize production costs in order to remain internationally competitive in industry.

Down times in production systems can lead to loss of production and therefore represent an important cost factor. The aim of diagnostics is to significantly reduce this cost factor.

What is Process Diagnostics?

Process diagnostics can be used to monitor the process while a machine or plant is operating (for example, to check for errors in motions or missing initial requirements). Process diagnostics has the task of collecting information on the type, location, and cause of the error, and providing solutions to these errors.

System-Wide Interaction

Figure 1-1 below shows how all the components of process diagnostics interact throughout the system.

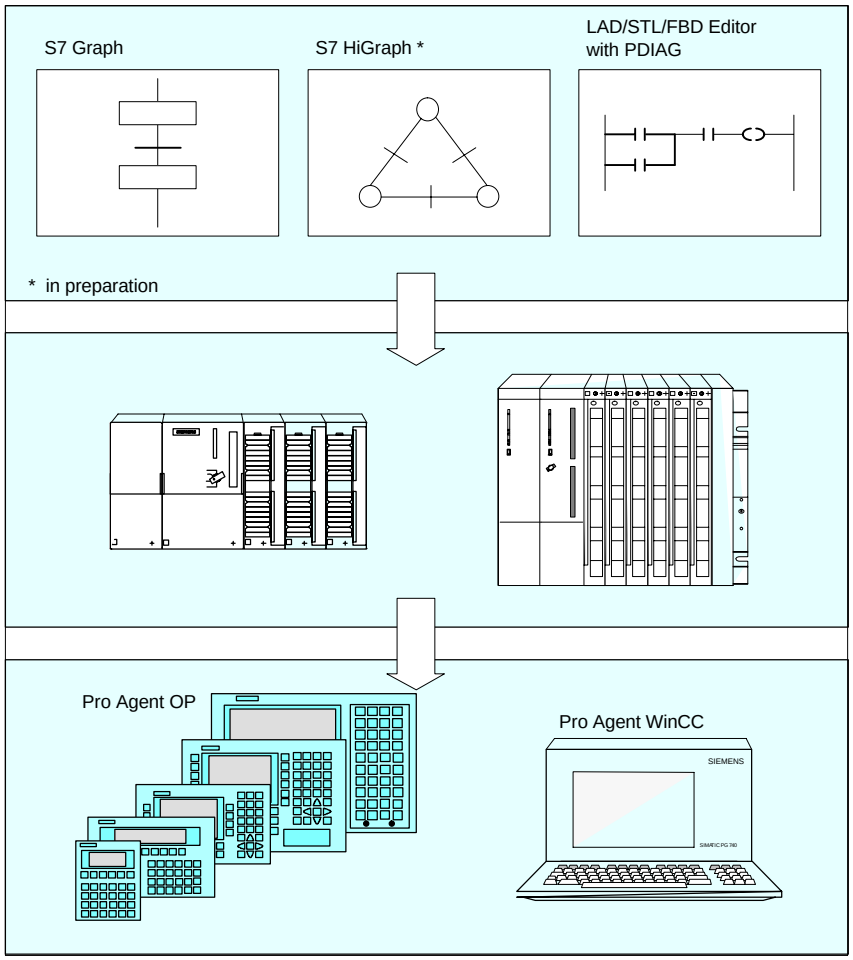


Figure 1-1 How Process Diagnostics Works

Embedding in the Entire System

Process diagnostics is completely integrated in the SIMATIC S7 configuration software. You can configure diagnostic data while you are still implementing your user program.

These data and other data relevant to process diagnostics are stored in a common database, as shown in Figure 1-2.

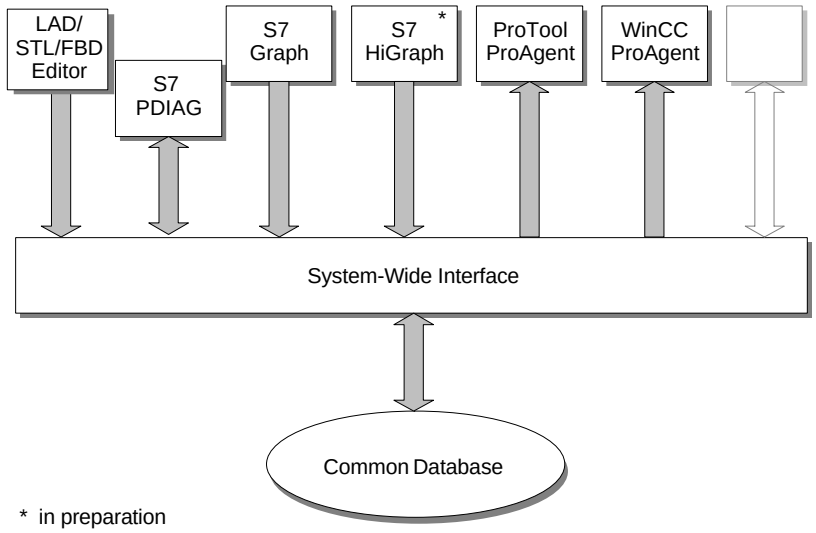


Figure 1-2 Common Database in the Entire System

By embedding process diagnostics in the entire system, you can considerably reduce the amount of work involved in configuring process diagnostics.

Advantages for the User

You, as a user of process diagnostics, have the following advantages:

- Reduction in down times and production losses when an error occurs
- Easier troubleshooting using specific information (also for machine operators within the plant itself)
- Simple configuration of the diagnostic system independently of the performance area
- Consistency between the process diagnostics and the user program

1.2 Overview of Process Diagnostics with S7 PDIAG

Introduction

If an error occurs in the process, process diagnostics performs the following functions, as shown in Figure 1-3:

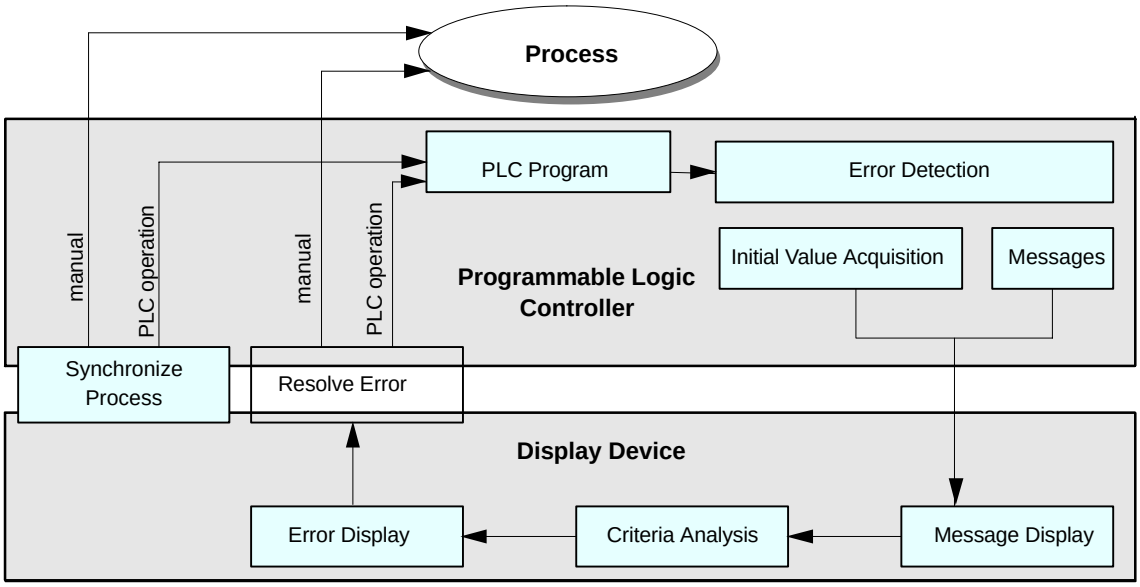


Figure 1-3 Overview of Process Diagnostics with S7 PDIAG

1. Error detection:
   When a process error occurs, it is detected by the monitoring logic configured with S7 PDIAG. At the same time, the states of the addresses which caused the error are saved. This is known as initial value acquisition.
2. Message display:
   Both incoming and outgoing process errors are detected by S7 PDIAG and represented as an incoming or outgoing message on the display device.
3. Criteria analysis:
   Using criteria analysis together with the display devices, you can determine which address caused the error (and therefore the reason for the process error in S7 PDIAG) on the basis of the initial values (with boolean program logic).
4. Resolve error:
   Errors can be resolved by manually intervening in the process and/or by operating the process using the programmable logic controller in manual operation (on the display device).
5. Synchronize process:
   Depending on the type of error to be resolved, the program will either continue processing or the plant is brought to a defined starting situation and the program then continues.

1.3 How Does S7 PDIAG Support You when Troubleshooting on Your Display Device (HMI)?

Introduction

The information supplied by S7 PDIAG also supports you when troubleshooting on display devices. Using criteria analysis, you can analyze the conditions which led to a process error on the display device. To do this, you use the initial values stored in the programmable logic controller.

The display devices are provided with the corresponding configuration data for this by the ProTool and ProAgent configuration software.

Displaying Error Messages

The diagnostic information is represented in four different screens on the display devices which you can switch to and from as required:

1. The message screen which displays all current alarm messages.
2. The diagnostic overview screen which lists all existing units in the plant.
3. The diagnostic detail screen which displays the result of criteria analysis for an alarm message. This function analyzes which signals in the user program led to the error message.
4. The motion screen which represents all the executable motions of a unit.

For more detailed information, refer to the documentation for your configuration software (for example, ProTool or ProAgent).

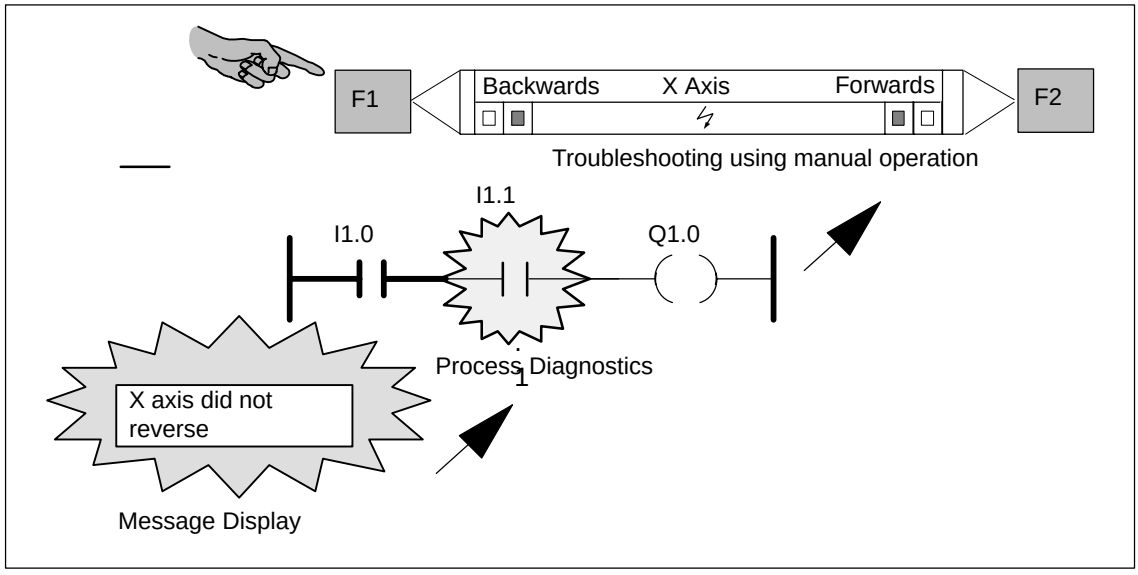


Figure 1-4 From the Message Display via Process Diagnostics to Easy Troubleshooting

1.4 Monitoring Your Process with S7 PDIAG

Introduction

S7 PDIAG expands the functional scope of the STEP 7 Standard software by offering a process diagnostics option for the programming languages Ladder Logic (LAD), Statement List (STL), and Function Block Diagram (FBD). Process diagnostics detects errors in the user process (manufacturing, distribution, processing, etc.) and provides information on:

- The type of error
- The location of the error
- The cause of the error in your process.

It also helps you when troubleshooting.

Monitoring Your Process

Using the S7 PDIAG optional package, you can monitor your process for specific errors. You may configure these errors while you are creating your user program or at a later stage. There are various types of monitoring available:

- **Address monitoring**

  You can use this type of monitoring to monitor specific, individual addresses for level or edge changes, which can be combined with a delay time. You can monitor addresses without having to modify your user program.

- **Motion monitoring**

  You can use this type of monitoring to monitor whether physical motions in your process have been executed correctly and within the specified time. The requirement for motion monitoring is that you follow the programming rules and thus adapt your user program as necessary.

- **General monitoring**

  You can use this type of monitoring to monitor process errors which are formed from the logic operations linking a number of addresses without having to modify your user program. S7 PDIAG does not trigger an error message until the logic has been fulfilled.

Error Definitions

An error definition is an address, motion, or general monitoring definition. In error definitions, you define the exact error which is to be monitored. Such error definitions can be appended to addresses in the LAD/STL/FBD Editor. You can monitor all boolean addresses with S7 PDIAG.

Monitoring Blocks

S7 PDIAG generates monitoring blocks from the configured error definitions which you can then download to your CPU in order to monitor your process.

1.5 When an Error Occurs...

Introduction

An error detected by S7 PDIAG is registered on all connected display devices while your user program is running by means of a message text which you configure. You can enter message texts while you are configuring the error definitions.

You can assign each message a priority (from 1 to 16). In this way, you can react to specific errors with different priorities in your user program.

Initial Diagnostic Address (IDA)

The initial diagnostic address is the starting point for tracing back an error if criteria analysis is to be carried out. The point of use of the address must be an assignment or one of the operations “Set” or “Reset”.

Initial Value Acquisition

If you have activated initial value acquisition, all initial values for the address being monitored are saved in the programmable logic controller during the same cycle in which the error was detected. The initial values are the binary states which have led to the result of logic operation of the address being monitored.

The information on which addresses are involved and how these addresses are interconnected is taken directly from your user program.

Criteria Analysis

Criteria analysis is carried out on the display device in order to analyze the error conditions. You can only carry out criteria analysis for boolean addresses (see Readme.wri) and analysis starts with the initial diagnostic address (IDA). Criterial analysis evaluates the initial values of all networks which determined the value of the initial diagnostic address. You can then display the states of the addresses (initial values) which caused the error (for example, limit switch at input I1.1) up to the result of logic operation in STL, LAD, and FBD directly on the display device.

Positive criteria analysis assumes the signal state “1” of the initial diagnostic address to be correct, while negative criteria analysis assumes that signal state “0” is correct.

The criteria analysis is available for all Boolean input parameters of a function block and can thus also be carried out across the block limits.

In order to carry out criteria analysis on the display device, you must activate initial value acquisition in S7 PDIAG. Criteria analysis is still possible beyond block limits without additional programming being necessary.

Exclusion Addresses

You can create a list of “exclusion addresses” for criteria analysis which are defined as “never causing an error.” Criteria analysis then hides these addresses and the network sections which contain them if they have been registered as having the value “0” (this is only possible in conjunction with ProAgent, version 5.0 or higher). In this way, you can differentiate between manual and automatic operation, for example.

Auxiliary Process Values in Message Texts

An auxiliary process value is a value (or address) which can be “added” to a message text. This value is acquired by the S7 PDIAG at the point at which the error is also recognized. The auxiliary process value is displayed by the display system at the point in the message text which you have specified. To do so insert the corresponding formal address into the message text.

This auxiliary process value can be a parameter of the type BOOL, BYTE, WORD or DWORD from the areas I, Q, M or DB. You can specify the position and the display format of the auxiliary process value in the message text.

Status / Actual Value Acquisition

Using the status/actual value acquisition of the initial value addresses, you can check on the display device whether an error state has actually been resolved (this is only possible in conjunction with ProAgent, version 5.0 or higher).

Group Error Bit ID

There is a group error bit ID in each of the user data types “Unit,” “S_UNIT,” and “Motion.” This bit is set by S7 PDIAG in all units and motions above these user data types in the hierarchy whenever an error occurs. An error in the subordinate unit “Clamp” is therefore also displayed in the unit “Drill” above it in the hierarchy.

Obligation to Acknowledge Messages

The obligation to acknowledge messages can be planned separately for each message (message-specifically). This parameter specifies whether this message has to be read and acknowledged manually at the display device or whether the message may also disappear “unread” when the error has passed.

Modifying Times in Existing Monitoring Definitions

If an existing monitoring definition contains a monitoring time (not equal to 0), you can change this using the “Modify Times” function without having to generate the monitoring blocks again. You can do this both offline and online and it has the advantage of enabling you to define the appropriate monitoring time step-by-step.

Searching For and Editing Objects

You can search for and edit the error definitions, other error definitions, units, motions and templates in S7 PDIAG. Various possibilities are available to this purpose.

1.6 Units and Motions in S7 PDIAG

Introduction

S7 PDIAG works with units and motions; these are introduced in this section.

We will then show you how units and motions are represented in the unit overview and how S7 PDIAG works with block types (function blocks) and their corresponding instances (data blocks).

What are Units?

Units structure the process view according to components which are related to one another by their technical function. If you have set up your program so that each block relates to a physical object in the process (for example, a press, a stamp, or a safety guard), the units represent an image of your process. A unit exists for each block in your program which can be diagnosed.

Units can also save data common to all other units, motions, and function blocks which lie below them in the hierarchy.

A unit may contain error definitions, motions, and other subunits.

Using units, you can combine both individual errors and motions into a functional unit. This enables you to find process errors quickly and easily.

Units are represented with other objects in a tree structure in the unit overview. Units for a data block, function, or organization block are also visible in the unit overview on the display devices.

Grouping Units

In addition to the default standard group you can group any number of units in up to 15 different groups. However, you should not group the units until the end of the structuring phase has been completed when you have already created your program hierarchy completely.

What are Motions?

Motions in a process are often defined as follows:

- They have two directions with two or more stable final positions
- They can be moved in the corresponding direction when triggered

For example, a cylinder moves from the current final position to the target final position when the hydraulic pressure is switched on.

A plant or machine may contain a large number of motions. It is therefore a good idea to combine motions with similar functions together in a subsystem, which is referred to here as a unit.

Motions are defined by the fact that the UDT_Motion is used in a block. Predefined networks are available for controlling motions easily.

Motions are sequences in the process which are monitored using error definitions. You can create several error definitions for each motion. A motion can only be contained in a unit, and represents an actual movement of a physical object in the process (for example, a clamp opening and closing).

Motions are represented with other objects in a tree structure in the unit overview. When you create your error definitions, the new motions are depicted in the unit overview, as shown in Figure 1-5 below.

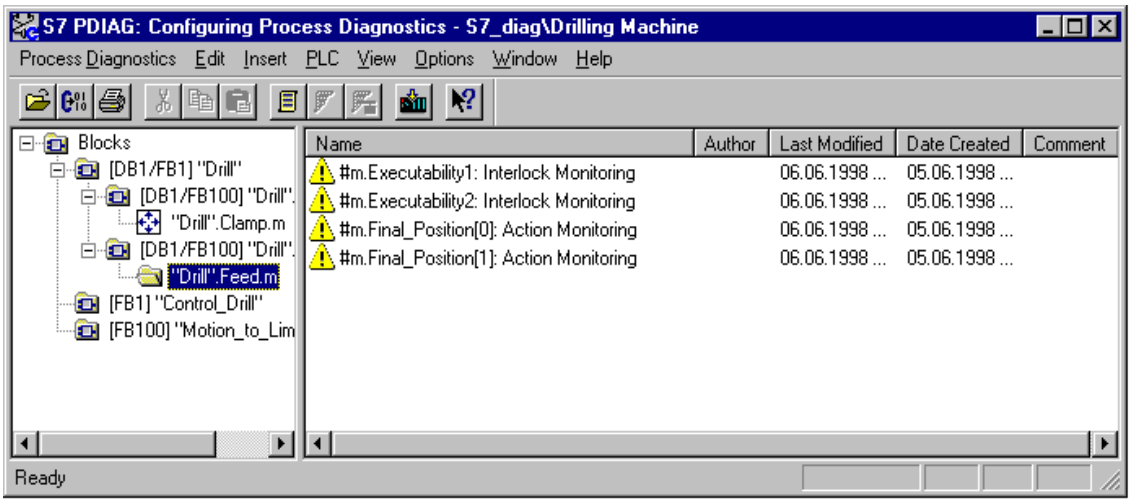


Figure 1-5 Representation of Units and Motions in the Unit Overview of S7 PDIAG

The motions represented in the unit overview are output on the display devices (using ProAgent) in the motion screens; for example, with the current final positions, and can be moved from there using manual operation, for example.

Supporting the Type / Instance Concept

S7 PDIAG supports the type / instance concept of SIMATIC S7. Error definitions can be fully configured in the block type; that is, in the function block. S7 PDIAG then automatically generates the instances for the error definitions in the same way as the instance data blocks in your user program, including the corresponding messages.

Generating Instance-Specific Message Texts

You can replace formal addresses by the name of the unit or motion in message texts.

If necessary, the final position names of motions can be automatically preset with the symbolic names configured in the block.

Diagnostic-Relevant Network Data

As from V5.0 of S7 PDIAG you can specify that the diagnostic-relevant network data are to be written to the initial-value acquisition blocks generated by S7 PDIAG (under consideration of the exclusion addresses). From there they are read by the display device (HMI) as required.

This has the advantage that you only have to generate in S7 PDIAG, depending on the change, and that the display devices (HMI) do not have to be updated every time.

1.7 Supporting Functions in S7 PDIAG

Printing

You can also print out the data you create with S7 PDIAG. The standard STEP 7 layout applies. Each page has headers and footers, and the actual content of the page consists of a leader as well as the corresponding units and error definitions.

- The block numbers for the blocks generated by S7 PDIAG are printed in the leader.
- The selected units, motions, and error definitions are then printed.

Exporting Diagnostic Data

You can export the diagnostic data generated by S7 PDIAG to an ASCII file. You can then build on these data using your own tools; for example, to create the basis for error statistics.

How S7 PDIAG Differs from S7 Graph and S7 HiGraph

In contrast to the implicit diagnostic capability available in the S7 Graph and S7 HiGraph language packages, the blocks in your user program are not affected by process diagnostics with S7 PDIAG. S7 PDIAG creates additional blocks for error detection for monitoring your process without modifying the existing blocks in your user program.

You can also monitor your entire process with S7 PDIAG, since it runs cyclically at the end of your user program (or wherever you insert it in your user program). With S7 Graph and S7 HiGraph, only the addresses within the currently active step or state are monitored.

Interfaces to the User Program

S7 PDIAG provides the following interfaces to your user program, which will help you to make use of the advantages of S7 PDIAG:

1. The supplied Ladder networks for motion programming (see Appendix A)
2. The supplied user-defined data types: “Unit,” “S_Unit,” and “Motion” (see Appendix A)
3. The supplied interface for the “user function block” (see Appendix A)

Improved Performance during Generation

In order to ensure short generation times even during large data quantities, the following extended functions were introduced in S7 PDIAG:

- Before the actual generation S7 PDIAG analyzes the changes in the user program and then only carries the generation steps which are still required.
- In addition changes are carried out in the database in order to reduce both the generation times and the handling times of blocks which can be diagnosed.

Advantages of S7 PDIAG

Of course, you can still detect errors in your process even without S7 PDIAG. With S7 PDIAG, however, this is much quicker and more effective, because you can design your own, plant-specific process diagnostics with S7 PDIAG and configure special types of monitoring definition for specific sections of the process. This results in the following advantages:

- Errors in your process can be detected early on using S7 PDIAG; this considerably reduces down times and production losses in your plant.
- Simple and fast configuring, as well as minimum programming for motion monitoring is possible.
- Troubleshooting on the display device is made easier by specific information (for example, from a criteria analysis) and is possible without additional configuration being necessary.

1.8 How Do You Start Working with S7 PDIAG?

Introduction

This section gives you an overview of how to start working with S7 PDIAG. You will find a detailed, step-by-step description of the procedures for creating the individual monitoring definitions in the following chapters. Creating address monitoring is described in Chapter 4, creating general monitoring in Chapter 5 and creating motion monitoring in Chapter 6:

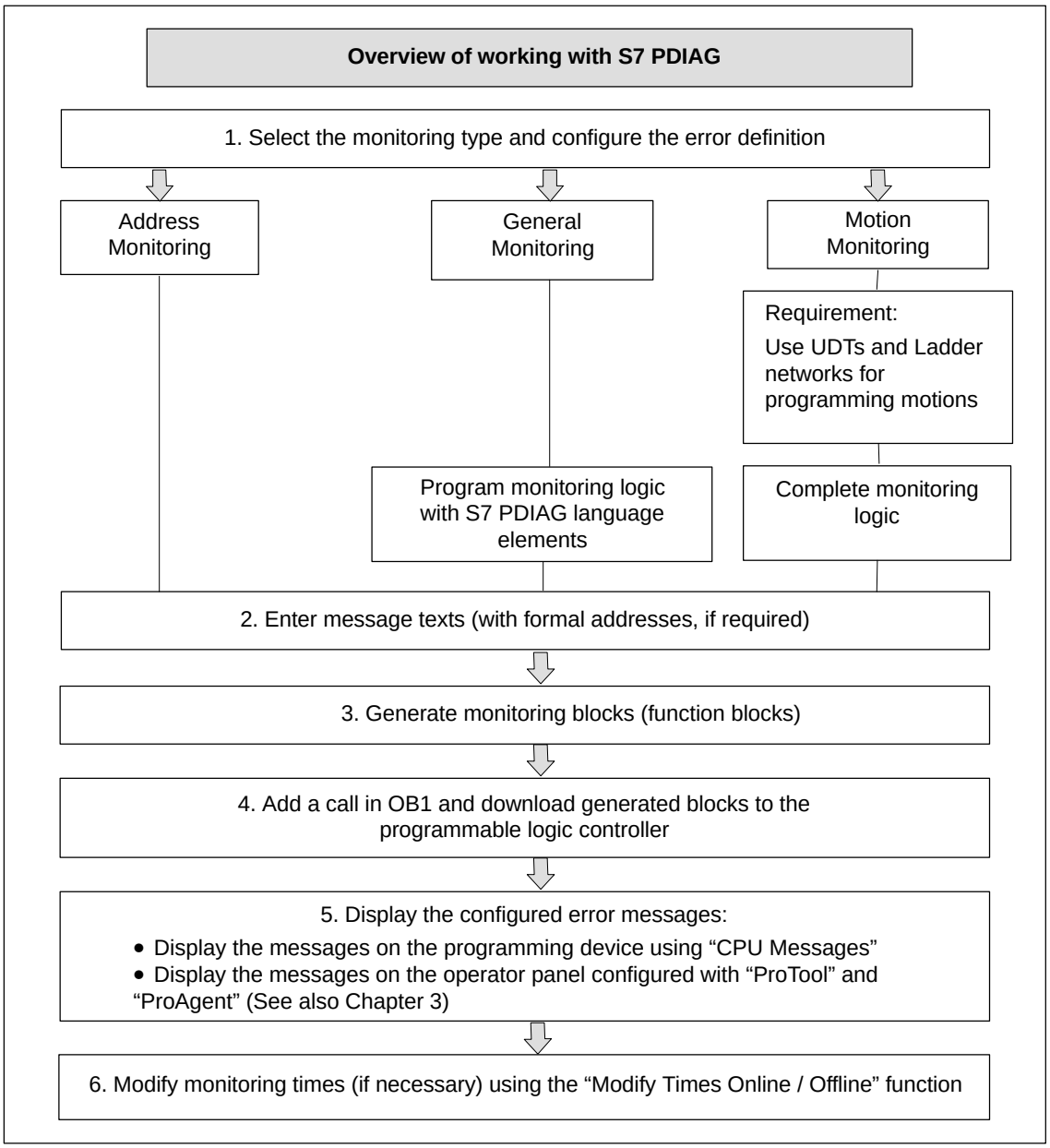


Figure 1-6 Overview of Working with S7 PDIAG

Overview of the Configuration Steps

As Figure 1-6 shows, you must carry out the following steps in order to configure process diagnostics:

- First, select a suitable type of monitoring and create an error definition in which you describe the exact error state in your process which you want to monitor.
  - With address monitoring, you select the required initial diagnostic address.
  - If you decide upon either general monitoring or motion monitoring, program or complete the monitoring logic and read the notes on programming your user program in Appendix A of this manual.
- Then configure the message texts for your error messages.
- Once you have configured all the error definitions with the corresponding message texts, you can generate the monitoring blocks which contain all the data relevant to S7 PDIAG.
- Then add a call for the error-detection blocks at the end of OB1, or at another location, and download the monitoring blocks to your programmable logic controller.

Result: If the error occurs, an error message is displayed on all the connected display devices (for example, programming devices or operator panels) with the message text you configured.

Offline Functions

You can program and configure the error definition offline on a programming device or PC.

The monitoring blocks generated are also added offline to your user program as usual.

Using the “Modify Times Offline” function, you can modify the monitoring times offline in existing monitoring definitions without having to generate the monitoring blocks again each time.

Online Functions

If the process errors you have configured occur in online operation, these are detected and registered.

Using the “Modify Times Online” function, you can modify the monitoring times online in existing monitoring definitions without having to generate the monitoring blocks again each time.

On the display device (for example, operator panel), you can carry out a criteria analysis of the process error which has just occurred. This means that the states of the addresses which led to an error are displayed here:

Example:

Address:	States:	
A I0.0	1	// First initial value
A I1.1	0	// Second initial value
A I1.2	0	// Third initial value
= Q1.0	1	// Monitors Q1.0 for level “0”

The error was triggered because the address being monitored has level “0.” Criteria analysis will cite I1.1 as the cause of the error.

1.9 How Do You Select the Right Monitoring Type?

Introduction

The following table will help you to select the most suitable monitoring type for your requirements.

You can find additional information on creating monitoring definitions and a detailed description of the procedure in the chapters given below.

Table 1-1 Selecting the Right Monitoring Type

What do you want to monitor?	Most suitable monitoring type	See:
1 address for: a defined level (0 or 1) or 1 address for: a defined edge (rising or falling), each combined with a delay time	**Address monitoring:** as level monitoring as edge monitoring	Chapter 4
Several addresses which can be combined via a freely programmable monitoring logic (for example: EP I1.0 OR I1.1 AND I1.2)	General monitoring	Chapter 5
Motion monitoring definitions which are time-dependent, for example: • Whether a motion starts within the preset startup time. This is the case if the current final position is exited. • Whether a motion is completed within the preset action time. This is the case if the target final position was reached. • Whether a target final position which has been reached remains stable; that is, if it is not left for longer than the preset reaction time. • Whether the interlock conditions necessary for executing a motion are fulfilled once the preset interlock time has expired.	**Motion monitoring:** as startup monitoring as action monitoring as reaction monitoring as interlock monitoring	Chapter 6

1.10 Planning an Auxiliary Process Value in an Error Definition

Introduction

If an error definition is active, meaning that the defined error case has occurred, a value can be acquired additionally from your process and sent with the message to the corresponding display device (HMI).

This auxiliary process value is displayed in the message text at the display device at a point specified by you. The auxiliary process value can be entered as well when creating or editing error definitions.

What is an “Auxiliary Process Value”?

An auxiliary process value is a value (or address) which can be ”added” to a message text. This value is acquired by the S7 PDIAG at the point at which the error is also recognized. The auxiliary process value is displayed by the display system at the point in the message text which you have specified. To do so insert the corresponding formal address into the message text.

This auxiliary process value can be a parameter of the type BOOL, BYTE, WORD or DWORD from the areas I, Q, M or DB.

Inserting an Auxiliary Process Value into the Message Text

You can specify the position and the display format of the auxiliary process value in the message text. To do so, create a describing block for the auxiliary process value which starts with the character “@1X” and ends with “@”. The auxiliary process value is inserted into the message text instead of this describing block.

Examples of an auxiliary value:

@1X%6d@: The auxiliary process value should be represented as a decimal number with a maximum of 6 digits.

@1X%1b@: The auxiliary process value is represented as a Boolean value “0” or “1”.

For further information on the procedure please refer to Section 9.5 or the online help for S7 PDIAG.

Installing the S7 PDIAG Optional Package 2

In This Chapter

This chapter contains important information on authorizing your process diagnostics software.

It describes how to authorize and install your software in order to work with S7 PDIAG. You will be supported by a menu-driven setup program.

Chapter Overview

2.1 Requirements for Using S7 PDIAG

Hardware Requirements

You will require a personal computer (PC) or programming device (PG) with the following system configuration:

- For Windows 95/98:
  An 80486 processor or higher, with at least 16 Mbytes RAM
  (preferably 32 Mbytes)
  For Windows NT/2000:
  A Pentium processor or higher, with at least 32 Mbytes RAM
  (preferably 64 Mbytes), and
- A monitor, keyboard, and mouse, all of which must be supported by Windows 95/98/NT/2000.
- Your S7 CPU must contain SFC17 and SFC18, otherwise the Alarm_S cannot be processed by S7 PDIAG and an error message will appear after you download the blocks.

Software Requirements

S7 PDIAG runs on a programming device or PC with the following:

- The Windows 95/98/NT or Windows 2000 operating system
- The STEP 7 Standard software, version 4.02 or higher.

Memory Capacity

The S7 PDIAG optional package requires:

- Approximately 1 MByte free memory on your hard disk for the setup
- An additional 8 to 12 MBytes free memory, depending on the extent of the installation.

Memory Required in the S7 CPU

The respective memory required by your S7 CPU consists of the following components: error detection and initial value acquisition.

Error Detection

For error detection, you require the following:

Basic memory requirement:	approximately 1368 bytes
Monitoring without time:	approximately 14 to 20 bytes
Monitoring with time:	approximately 106 to 114 bytes
Monitoring with auxiliary process value:	approximately 24 bytes

The above values are standard values which depend on the complexity of the networks being monitored and the corresponding monitoring logic used.

Initial Value Acquisition

For initial value acquisition, you require the following:

Basic memory requirement:	approximately 1470 bytes
Per planned marker word, additionally:	220 bytes
Per monitoring definition:	approximately 22 bytes
Per address:	approximately 4 bytes

Additionally for diagnostic-relevant network data in the automation system:

Basic memory requirement:	168 bytes
Per monitoring definition:	8 bytes
Per operator	1 byte

The above values are standard values which depend on the complexity of the networks being monitored and the corresponding monitoring logic used.

2.2 Authorization (License for Use)

Authorization Diskette

In order to use the S7 PDIAG software package, you require the read-only authorization diskette or the update authorization diskette included with the software package. This contains the authorization and the AUTHORSW program which enables you to display, install, and remove the authorization.

The number of authorizations you can install is determined by an authorization counter on the authorization diskette. Every time you install an authorization, the counter is decremented by one. When the counter value reaches zero, you cannot install any more authorizations using this diskette.

Caution

Note the information in the README.TXT file on the authorization diskette. If you do not adhere to these guidelines, the authorization may be irretrievably lost.

Authorizing Software during the Initial Installation

When installing your software for the first time, a message prompts you to install the authorization.

To install the authorization, proceed as follows:

1. Insert the authorization diskette when the message prompt appears.
2. Acknowledge the prompt.

The authorization is then transferred to a physical drive and your computer registers the fact that the authorization has been installed. You will find additional information and guidelines on using authorizations, as well as on installing and removing software in the STEP 7 User Manual.

If You Lose Your Authorization...

An authorization may be lost if a fault occurs on your hard disk before you are able to save the authorization.

If you lose your authorization, you can install an emergency authorization. This is also contained on the authorization diskette. The emergency authorization enables you to continue using the software package for a limited period. In this case, the remaining time before the authorization becomes invalid is displayed on startup. Within this period of time, you should obtain a replacement for the lost authorization. Contact your Siemens representative for details.

2.3 Installing the S7 PDIAG Software

Introduction

S7 PDIAG contains a setup program which executes the installation automatically. Screen prompts guide you step-by-step through the whole installation procedure.

Preparations

Before you can start installing the software, you must start Windows 95/98/NT/2000 and install the STEP 7 Standard software.

Starting the Installation Program

To start the installation program, proceed as follows:

1. Start the dialog box for installing software under Windows by double-clicking the “Add/Remove Programs” icon in the “Control Panel.”
2. Click the “Install” button.
3. Insert the data medium and click the “Continue” button. Windows 95/98/NT/2000 searches automatically for the installation program SETUP.EXE.
4. Follow the instructions displayed by the installation program.

A message will be displayed on the screen to inform you that the installation has been completed successfully.

Using Authorization

During installation, the program checks to see whether you have an authorization installed on the hard disk. If you wish, you can run the authorization program immediately or continue installing the software and execute the authorization at a later date.

Completing the Installation

A message on the screen will inform you that the installation was successful and that you can now start working with S7 PDIAG. You will find additional information and guidelines on installing and removing software in the STEP 7 User Manual.

Getting Started with S7 PDIAG and ProAgent/PC

3

In This Chapter

This chapter guides you step-by-step through the entire configuration with S7 PDIAG using address monitoring as an example.

This chapter also shows you the steps involved in creating an entire, functional process diagnostics program with a controller and a display device using ProTool and ProAgent.

Chapter Overview

3.1 Getting Started with S7 PDIAG Using Address Monitoring as an Example

Introduction

This section aims to show you how to work with S7 PDIAG using address monitoring as an example.

Overview of the Procedure

The diagram below gives you an overview of the procedure for configuring address monitoring with S7 PDIAG:

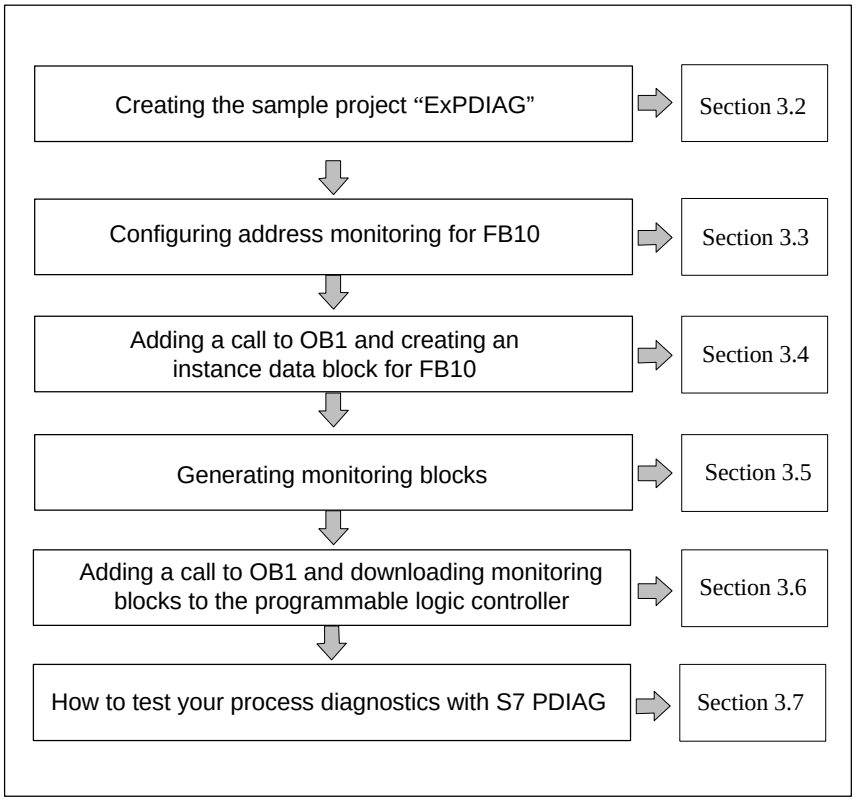


Figure 3-1 Procedure for Configuring Address Monitoring

3.2 Creating the Sample Project and Sample Program

Creating the Sample Project

First, use the STEP 7 Wizard to create a new project with the name “ExPDIAG” in the SIMATIC Manager. Insert an S7 program under your corresponding hardware configuration.

Creating the S7 Sample Program

Select the block container for your project “ExPDIAG” under the S7 program below your hardware configuration in the SIMATIC Manager and create the following function blocks using the menu command **Insert > S7 Block > Function Block**:

- FB10

You can now use the above blocks to create an address monitoring definition.

Executability

In order for the example to run on the programmable logic controller, input byte 0 and output byte 1 must be interconnected on digital modules. If you only have one CPU but no digital modules, insert OB122 (I/O access error) and monitor your parameters using the menu command **Monitor/Modify Variables**.

Programming FB10

Open FB10 in the SIMATIC Manger by double-clicking it and fill in the variable declaration table and the code section in the LAD/STL/FBD Editor as follows:

1. Enter the following in the first network:

   As the network name: Interconnection Q1.0 in FB 10

   As the program:
   ```
   U I 0.0
   U I 0.1
   U I 0.2
   U I 0.3
   = Q 1.0
   ```

2. Save the block using the menu command **File > Save**.

3.3 Configuring Address Monitoring for FB10

Introduction

Now that you have programmed the blocks for the sample program, you can create an address monitoring definition for these blocks.

Configuring Address Monitoring for FB10

1. If the FB10 is no longer opened, open it in the SIMATIC Manager by double-clicking on it. The LAD/STL/FBD Editor is opened.
2. In the example, output Q1.0 is to be monitored. Address monitoring is therefore to be added for this output. To do this, place the cursor in the instruction line "= Q1.0" and open the "Process Monitoring" dialog box using the menu command **Edit > Special Object Properties > Monitoring**.
3. Select the option "S7 PDIAG: Address Monitoring" in the "Templates" field and click "New."

   Result: The "Definition" tab in the "S7 PDIAG: Address Monitoring" dialog box is displayed. The initial diagnostic address displayed is the address in the instruction line, here Q1.0, as shown in Figure 3-2.

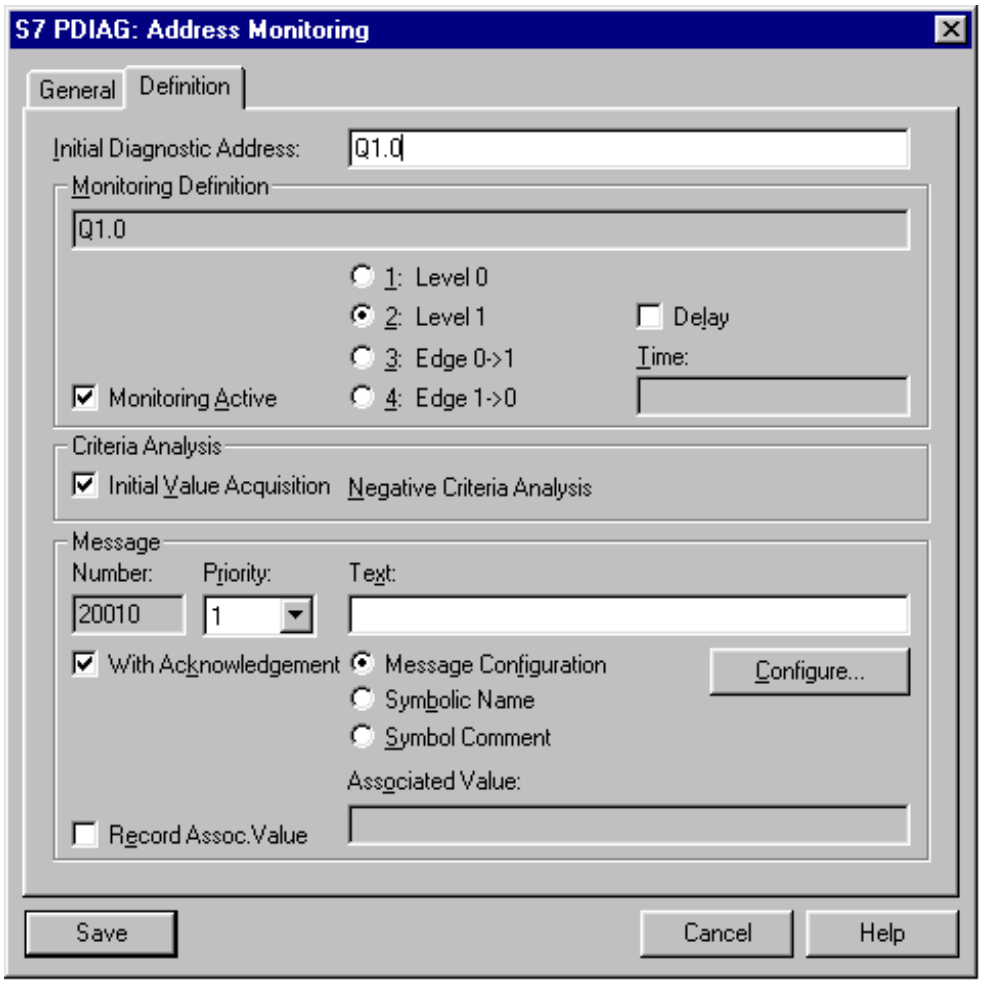


Figure 3-2 Creating Address Monitoring with S7 PDIAG

4. To assign the corresponding message text to this error message, enter the following in the “Message” group field: “Q1.0 in FB11 has level 1.”
5. Exit the tabbed sheet with “OK.” You have now configured an address monitoring definition for Q1.0 at level 1. This is now displayed in the “Process Monitoring” dialog box under “Existing Monitoring Definitions.”
6. Exit the “Process Monitoring” dialog box with “Close.”
7. Save the block using the menu command **File >Save**, so that the new error definition is saved in the block and exit the LAD/STL/FBD Editor.

3.4 Adding a Call to OB1 and Creating an Instance Data Block for FB10

Introduction

Now that you have programmed your blocks and configured address monitoring, you can add a call for these blocks in OB1 and create the instance data block for FB10 at the same time.

Procedure

Insert the following call for FB10 at the end of OB1 in the “ExPDIAG” project:

```
CALL FB10, DB10
```

Click “Yes” in the next dialog box to create the new instance data block (here: DB10) (see Figure 3-3).

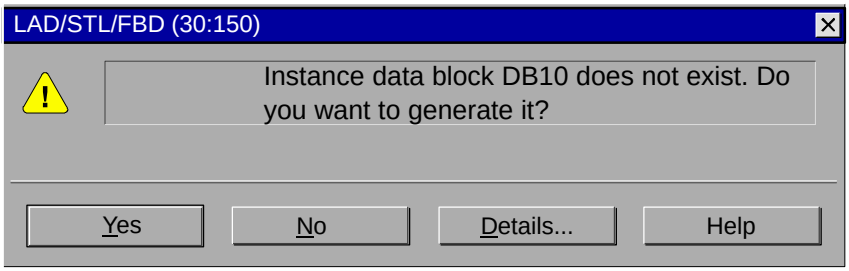


Figure 3-3 Dialog Box for Creating Instance Data Blocks

Result: DB10 is created with the data relevant to S7 PDIAG and also receives the S7 PDIAG attribute.

Save the block using the menu command **File > Save** and close the LAD/STL/FBD Editor.

3.5 *Generating Monitoring Blocks

Introduction

The following steps show you how to generate monitoring blocks from error definitions.

Procedure

To generate the monitoring blocks, proceed as follows:

1. Select the ”Blocks” container in the SIMATIC Manager and open S7 PDIAG by using the menu command **Options > Process Monitoring**.

   Result: The units relevant to PDIAG, in this case FB 10 and DB 10, are displayed in the unit overview as shown in Figure 3-4.

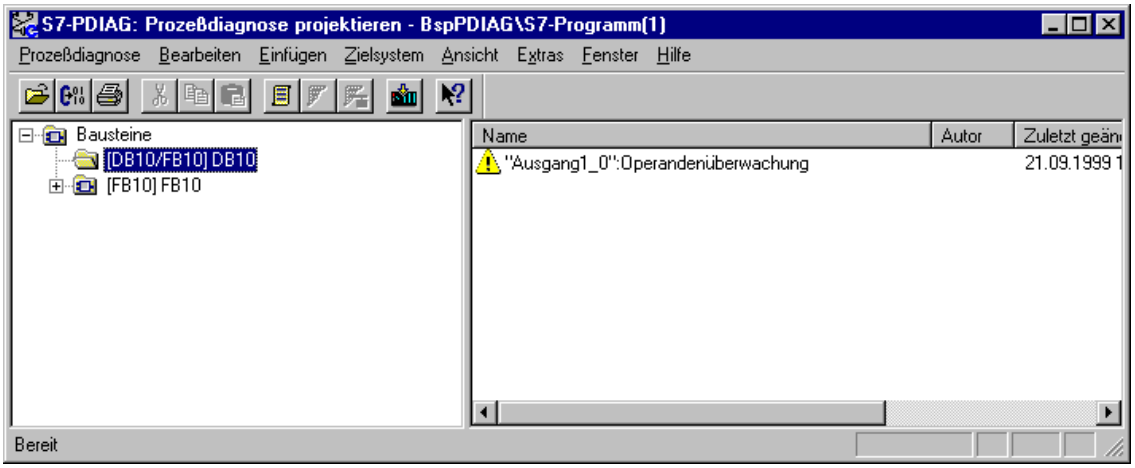


Figure 3-4 Display in the Unit Overview of S7-PDIAG

2. Select the menu command **Process Diagnostics > Compile** in S7 PDIAG. If you are compiling for the first time, you will be requested to check the compilation settings. Confirm this message with “OK.”
3. Open the “Customize” dialog box using the menu command **Options > Customize** and, in the “Default Settings” tab, enter the number “44” for the error-detection blocks to be compiled, and “45” for the initial value / status acquisition blocks, as shown in Figure 3-5.

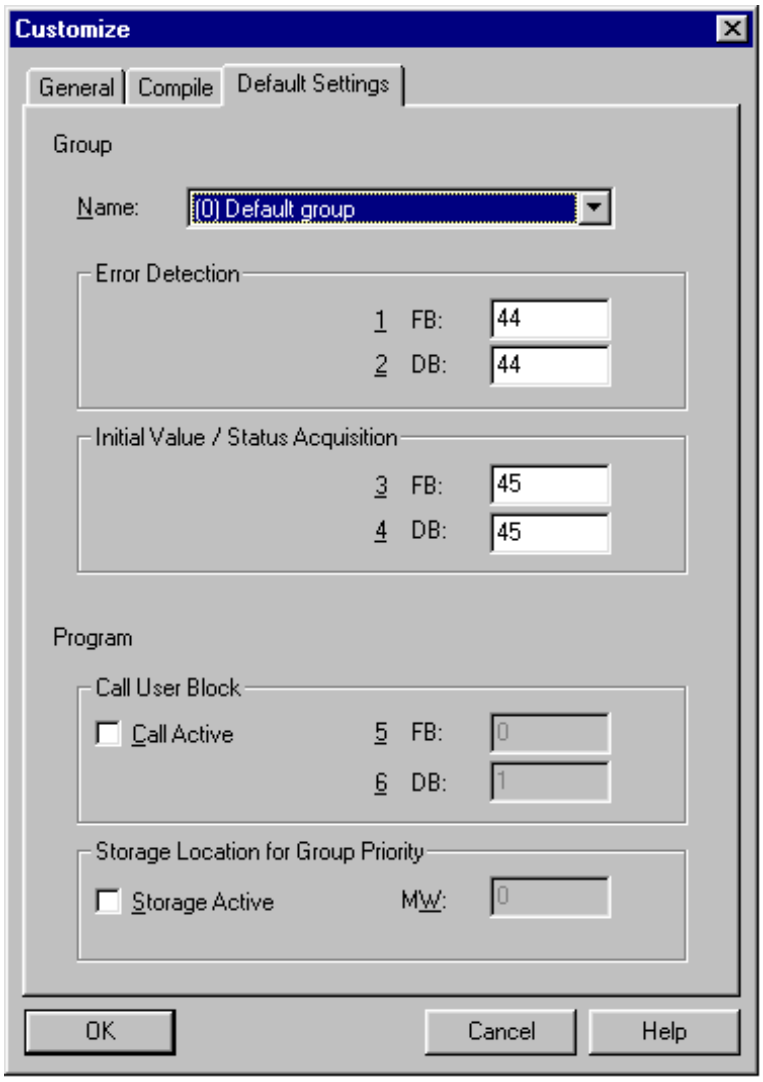


Figure 3-5 “Customize” Dialog Box

4. Exit the dialog box by clicking on “OK.” A progress bar is displayed and the monitoring blocks are generated. If an error occurs during compilation, a message will appear on the screen.

   Result: The generated monitoring blocks are displayed in the SIMATIC Manager together with the system function blocks (SFCs) required for them.

3.6 Adding a Call for Monitoring Blocks to OB1 and Downloading Monitoring Blocks to the Programmable Logic Controller

Introduction

In order for the monitoring blocks you generated to become active, you must download them to your programmable logic controller and add a call for these blocks in OB1, or at the required point in your user program.

Requirements

You have generated the monitoring blocks for your entire user program.

Adding a Call to OB1

To add the call for the generated error-detection block in OB1, proceed as follows:

1. Open OB1 in the SIMATIC Manager by double-clicking it.
2. Insert the following lines:

   CALL FB 44, DB 44

   PDIAGCycle: = OB1_SCAN_1

   Note: FB 44 contains the error detection. If an error is detected in FB 44, this block automatically calls FB 45, which is responsible for initial value/status acquisition.
3. Save the block and close the LAD/STL/FBD Editor.

Downloading the Sample Program

You can download the sample program “ExPDIAG” from the SIMATIC Manager to your programmable logic controller. Proceed as follows:

1. Select the block container in the SIMATIC Manager.
2. Download the sample program to your CPU using the menu command **PLC > Download**.

3.7 How to Test your Sample Process Diagnostics with S7 PDIAG

Introduction

Now that you have been through the whole configuration process with S7 PDIAG using the example, you can simulate a process error and display the configured messages via the CPU Messages application.

Requirements

In order to view the messages without a display device, you must start the CPU Messages application in the standard package. Proceed as follows:

1. Switch to online mode in the SIMATIC Manager.

   Result: The online project window appears.

2. Select the sample program “ExPDiag.”
3. Start the CPU Messages application using the menu command **PLC > CPU Messages...**.
4. Activate the check box under “A” in the “Customize” dialog box which appears so that you can display Alarm_S messages, and close the dialog box.

Now that you have made all the settings for displaying error messages in the CPU Messages application, you can start triggering process errors.

Triggering the Error Message in FB10

To trigger the error message configured in FB10, proceed as follows:

- Set inputs I0.0, I0.1, I0.2 and I0.3 at the same time. If you do not have any digital modules, you can do this using the STEP 7 function **Monitor/Modify Variables**.

  Result: This causes output Q1.0 in FB10 to be set to level “1.” This is recognized as an error by S7 PDIAG because of the configured error definition. An error message will appear with the message text you entered. This error message now appears in the “CPU Messages” window.

What Comes Next?

In the previous sections, you have learned step-by-step how to use S7 PDIAG to create a STEP 7 program for diagnosing processes.

You will now learn how to create a configuration for diagnosing processes on a display device (hereafter referred to as operator panel) using the ProTool configuration software and the corresponding optional package ProAgent (from the SIMATIC HMI product family).

This section will then show you how to diagnose a process yourself on the operator panel. This will familiarize you with the different diagnostic screens.

3.8 Getting Started with ProAgent

Introduction

This section shows you how to create a configuration for diagnosing processes for the previous example using ProTool and how to download this configuration to the operator panel.

Requirement

In order to configure process diagnostics in ProTool, you must have successfully generated the monitoring blocks for your user program, as described at the start of this chapter.

Display Device

The following description shows the OP25 as an example of a display device in all the figures. The procedure is identical for all display devices.

Overview of the Procedure

The diagram below gives you an overview of the procedure for diagnosing a process on your display device:

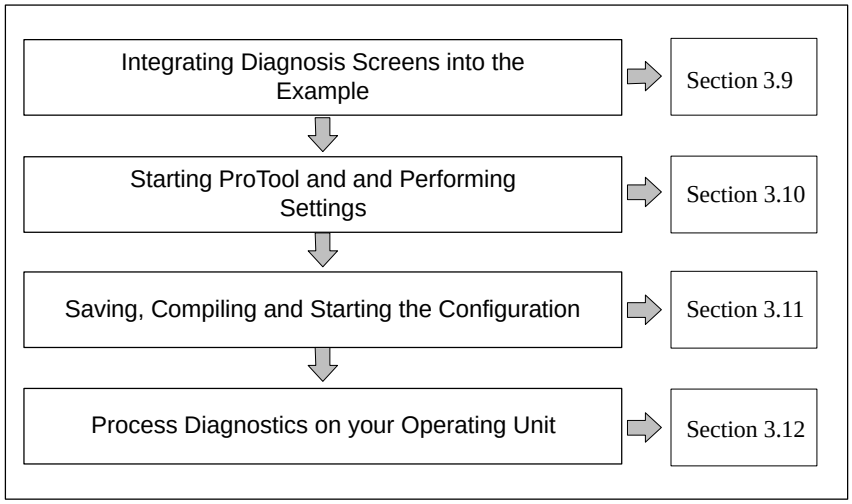


Figure 3-6 Procedure for Diagnosing a Process on your Display Device

3.9 Integrating Diagnosis Screens into the Example

Introduction

Before you can configure process diagnostics for the operating unit, you must first integrate the associated screens into your sample project.

Since only diagnostic screens are required in the example, you can apply the standard configuration directly to the example, together with the diagnostic screens. Otherwise you will have to describe, copy and insert the diagnostic screens, as described in the manual.

Procedure

To integrate diagnostic screens, proceed as follows:

1. If you have not already done so, start SIMATIC Manager and select the menu command **File > Open**.
2. In the *Open* dialog box, choose the *Projects* option and select the ”BspPDIAG” project from the list.
3. Open the ”ProAgent” project in the same way.

   Should this project not appear in the list box, click *Browse* and open the ”ProAgent” project in the ”Standard\ProAgent” subdirectory of the ProTool directory.

   The ”ProAgent” project contains standard projects for the various operating units.

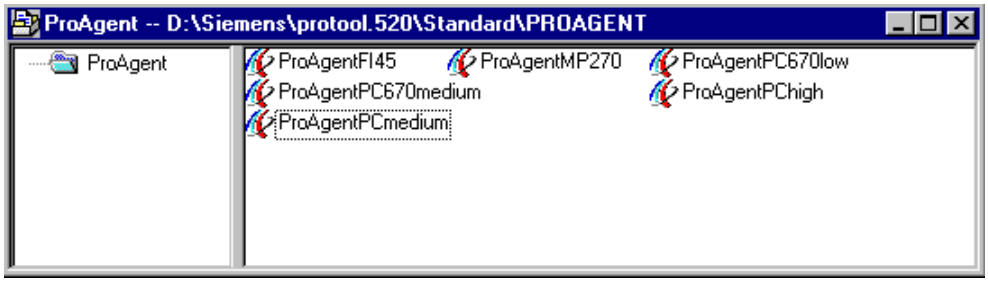


Figure 3-7 Standard Projects in the ProAgent Project

4. Use the mouse to drag and drop the ”ProAgentPCmedium” configuration into the ”BspPDIAG” project or use the **File > Save** *As* menu command to save it to the ”BspPDIAG” project.

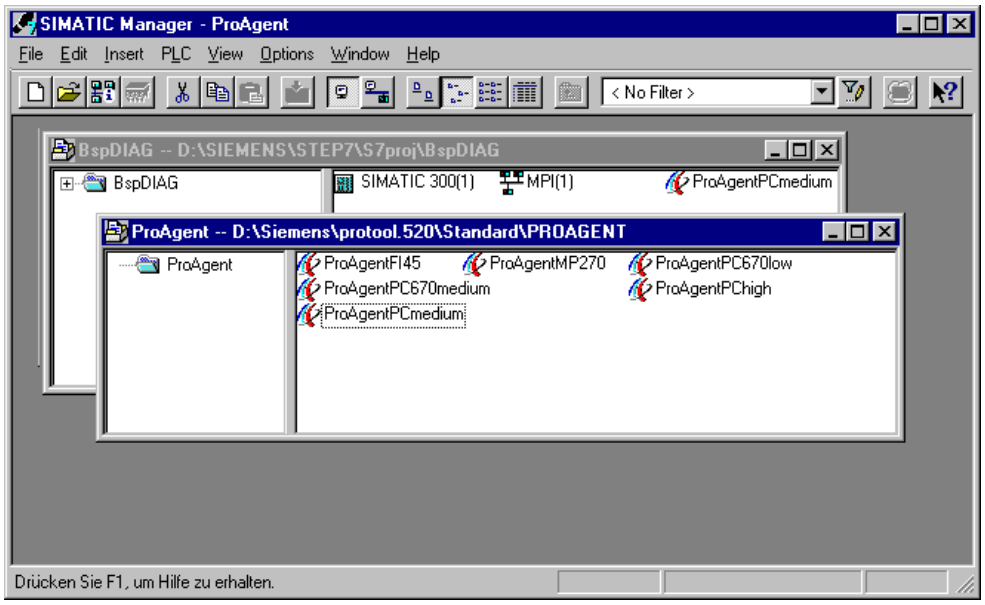


Figure 3-8 Sample Project with the ”ProAgentPCmedium” Standard Project

3.10 Starting ProTool and and Performing Settings

Introduction

The next step is to start ProTool and perform the necessary settings. Particularly important here are the network parameter, CPU and unit selections.

Selecting Network Parameters and CPU

Proceed as follows:

1. Start the ProTool CS configuration software by double-clicking on the symbol for ProAgentPCmedium.
2. From the configuration overview, select the *PLC* item.
3. Double–click on *Steuerung_1* on the right and in the *PLC* dialog box, click the *Parameters button.*
4. Select your network parameters and the networked CPU. For a CPU 316-2DP, the following parameters then result for the PLC, for example:

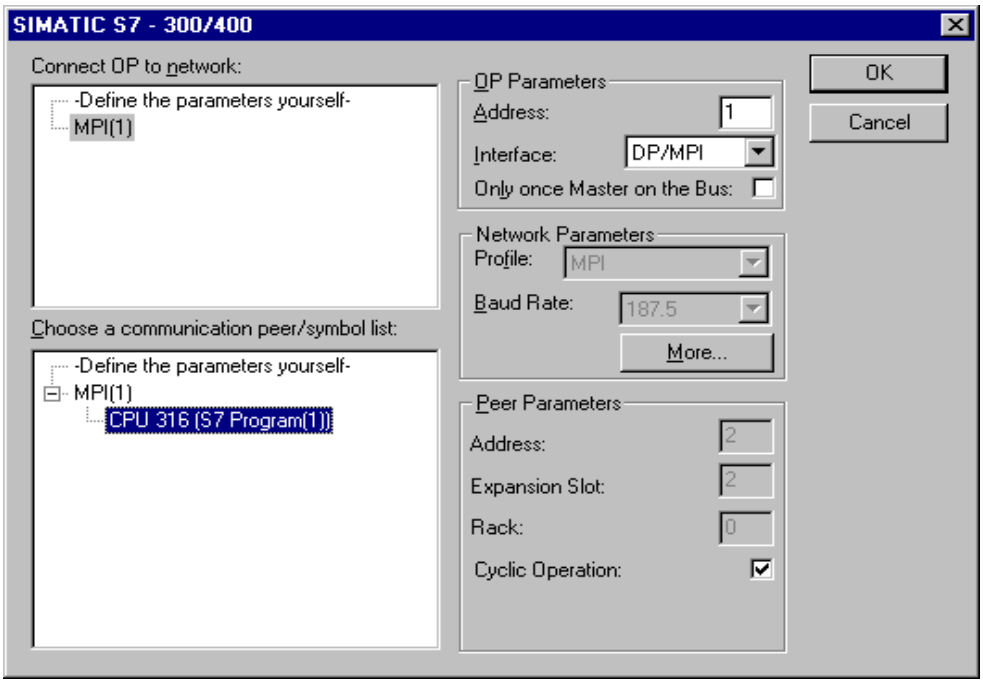


Figure 3-9 Parameters for the PLC

5. Confirm your inputs with *OK*.

Setting the Message Procedure

Then select those units for which you want to have process diagnostics performed. Proceed as follows:

1. Choose the menu command **System > ProAgent**.
2. Select the ”Steuerung_1” entry and click the >> button.

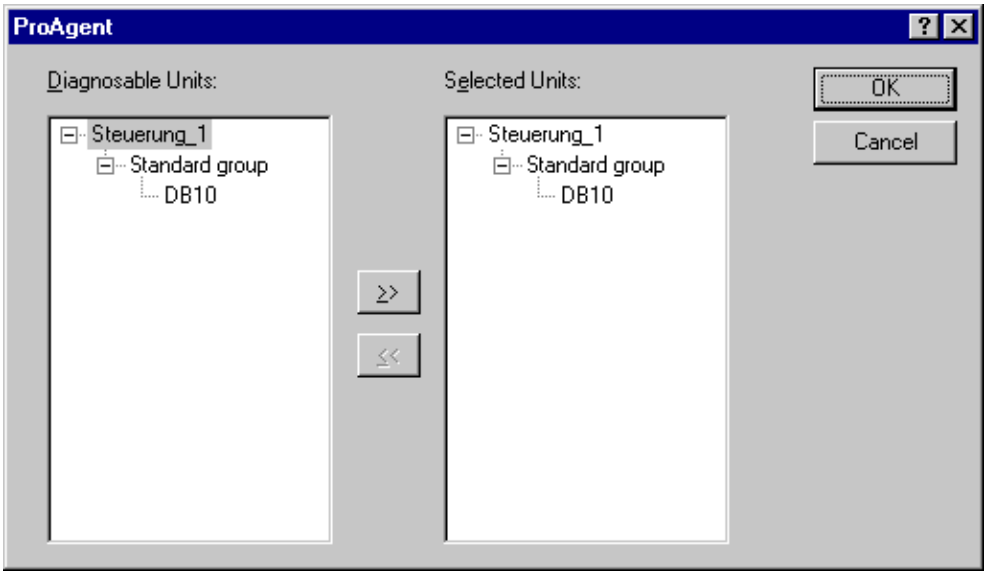


Figure 3-10 Selecting Units

The entry will be included in the list of selected units.

3. Exit the *ProAgent* dialog box with **OK**.

   Result: This makes all the units of Steuerung_1 diagnosable. All the messages for these units will be output on the operating unit.

3.11 Saving, Compiling and Starting the Configuration

Introduction

Once the configuration has been completed, you still have to save, compile and start the project. You can initiate all these steps at once by starting ProTool RT.

Note

If you do not want to use the same PC that you used for the configuration as the operating unit in this example, once the project is compiled you must first download it to the operating unit and then start it there.

Procedure

Proceed as follows:

1. Click on the symbol  for ”Start ProTool RT”.

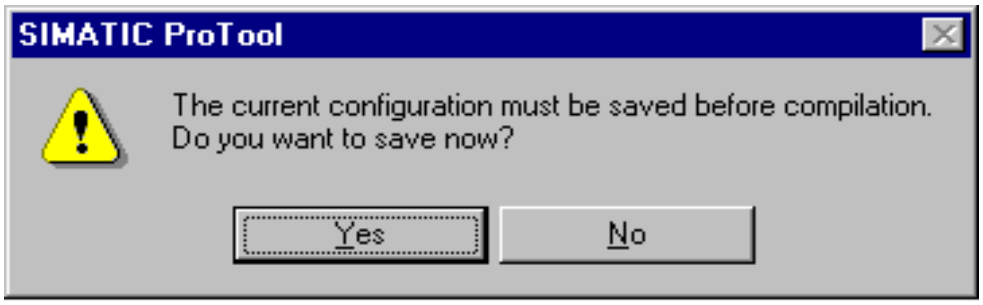


Figure 3-11 Confirmation of Saving Prior to Compilation

2. Confirm this with **Yes**.

   Result: ProTool synchronizes its data with the STEP 7 database. In doing so, the diagnostic data and the pieces of text for the ALARM_S messages are read from the database and a copy is stored in the ProTool configuration.

   The saving, compiling and downloading operations are now run. While this is happening, various messages are displayed in the status window on the *Compile* tab, for example:

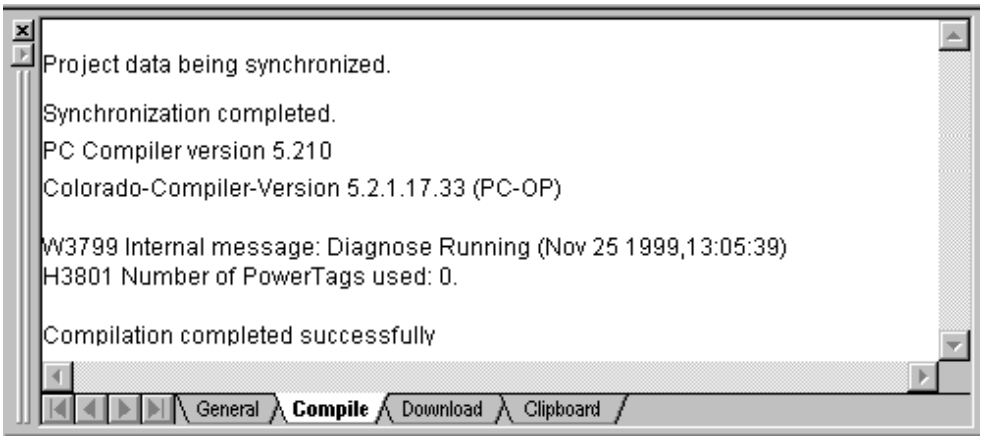


Figure 3-12 ProTool Messages during Compilation

ProTool RT starts up and you can now begin process diagnostics.

3.12 Process Diagnostics on your Operating Unit

Introduction

Now that you have created the configuration for the operating unit in the preceding sections and loaded it onto the operating unit, you can perform a process diagnosis.

Requirements

Before you can perform a process diagnosis on the operating unit, you must successfully perform all the steps described in the previous sections:

- the PLC program must be loaded on the CPU and
- the configuration must be on the operating unit.

Diagnosis start screen

Once you have started up ProTool RT on the operating unit, the diagnosis start screen appears:

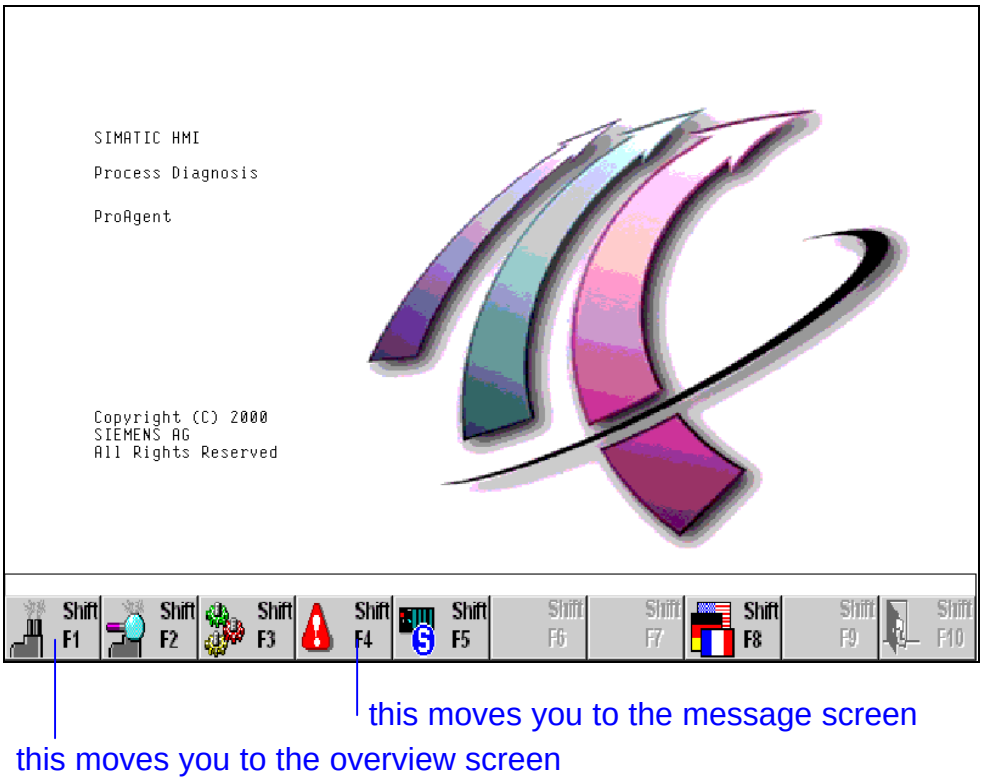


Figure 3-13 Diagnosis Start Screen on the Operating Unit

One of your options here is to change either to the overview screen or to the message screen. Change to the message screen by clicking the corresponding button.

Message screen

The message screen is blank at first because there have not been any errors so far.

1. Now simulate an error in FB 10, as you did previously in section 3.7.

   An alarm message will appear on the operating unit:

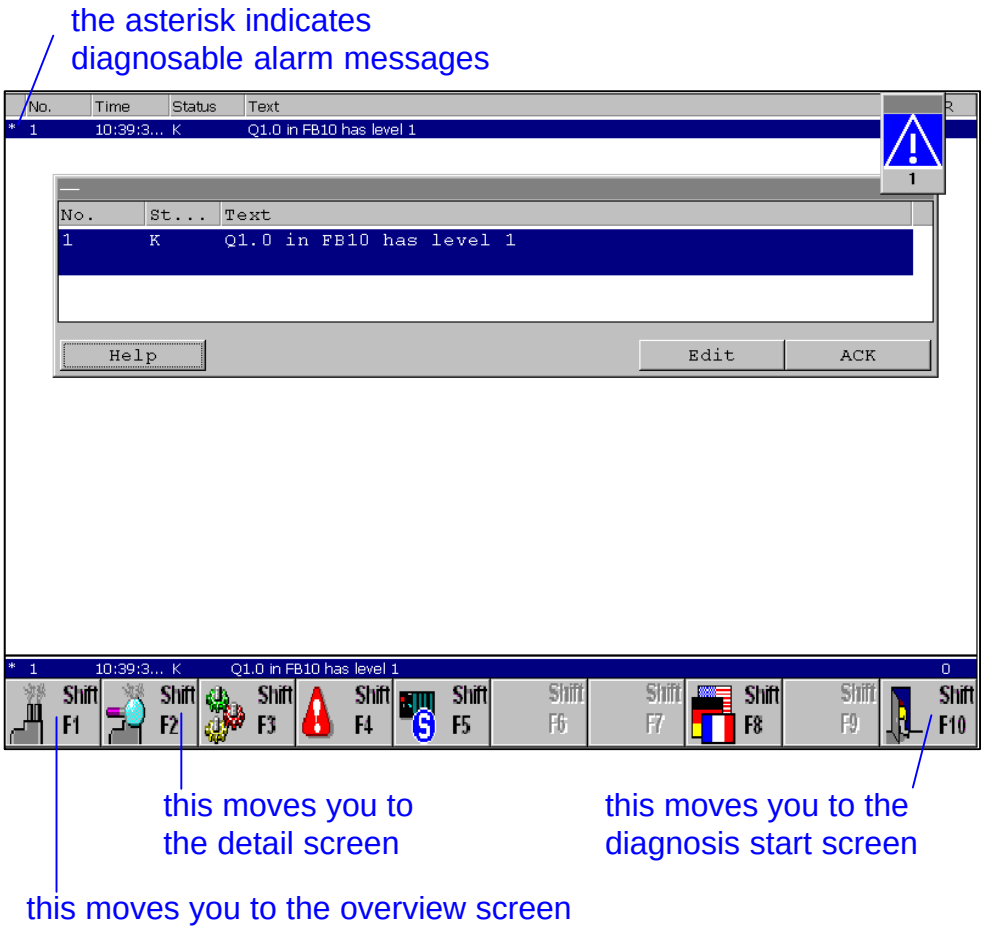


Figure 3-14 Message Screen with Message Window

2. Click on *ACK* to hide the message window.

   Of course, although you have now acknowledged the message, you still have to respond to the error itself. Until you clear the error, the little window containing the error character will keep flashing.

Explanations relating to the message screen

The asterisk to the left of the message indicates that this message is diagnosable. As only one message has been displayed so far, it has already been selected. You recognize the selection because the message line is inverted (light lettering against a dark background).

When several messages are displayed, you first have to use the cursor keys or the mouse to select the message for which you want to perform process diagnostics. Then press the corresponding button to call the overview screen.

Overview screen for "faulty unit"

In the overview screen below, you can see the faulty unit DB 10.

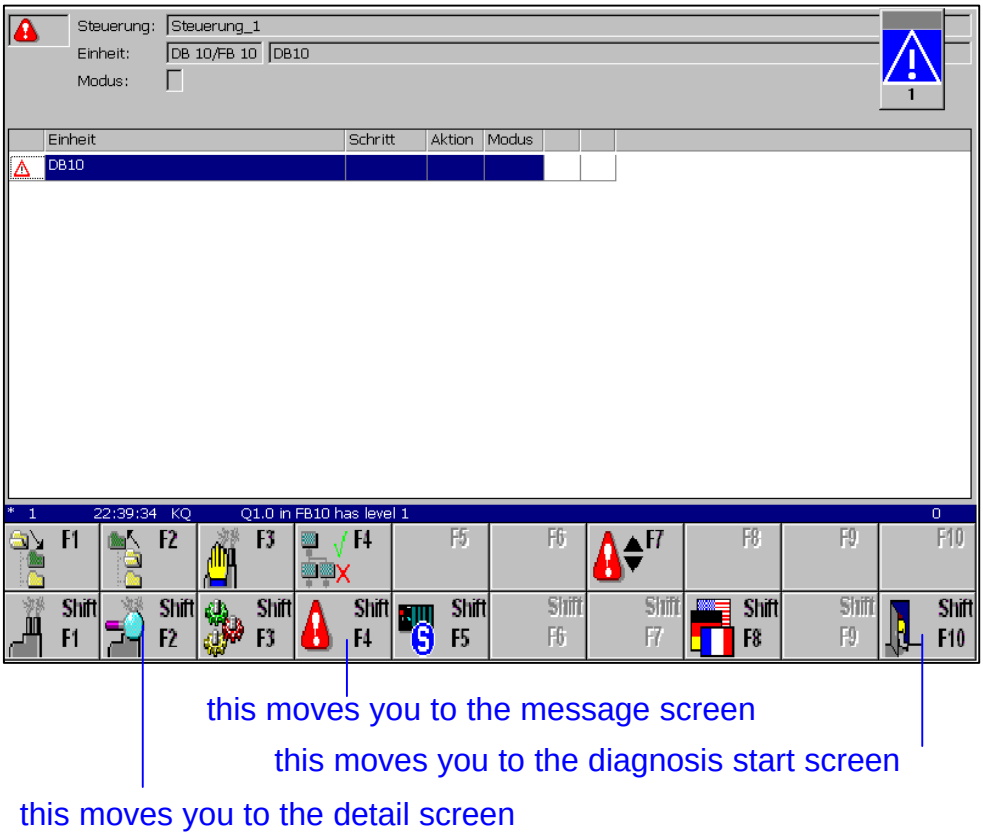


Figure 3-15 Overview Screen for Faulty Unit

Explanations relating to the overview screen for ”faulty unit”

A warning triangle identifies the faulty unit. The warning triangle is flashing, as this is the first error to occur. If the error had involved consequential errors, warning symbols would likewise appear for the affected units, but they would not flash.

You recognize where the error first occurred from the location of the warning triangle. In many cases, this is where you will find the cause of the error and its consequential errors. The faulty unit has already been selected.

Now press the corresponding button to call the detail screen.

“Higher Unit” Overview Screen

In the following overview screen, you will see the unit which is above the faulty unit in the hierarchy, in this case DB10.

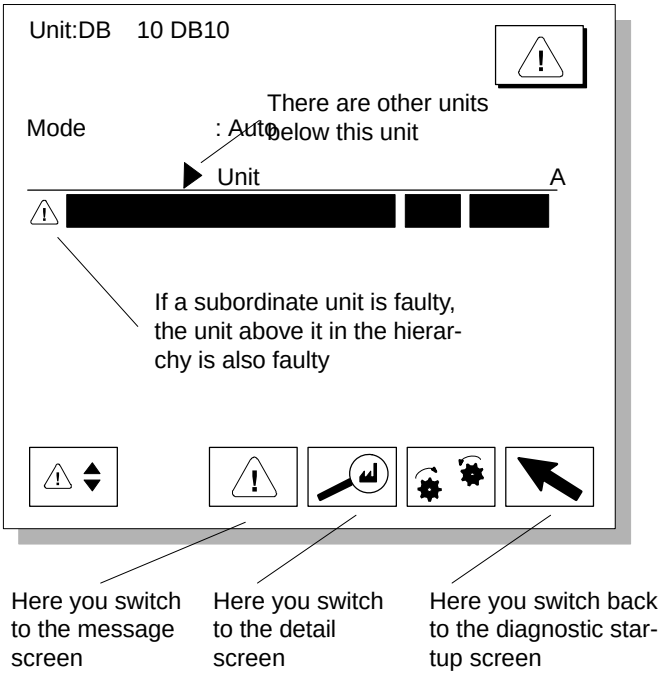


Figure 3-16 Overview Screen: Higher Unit

Explanations relating to the overview screen for "Higher Unit"

A warning triangle identifies the higher unit. The warning triangle is flashing, as this is the first error to occur. If the error had involved consequential errors, warning symbols would likewise appear for the affected units, but they would not flash.

You recognize where the error first occurred from the location of the warning triangle. In many cases, this is where you will find the cause of the error and its consequential errors. The faulty unit has already been selected.

Now press the corresponding button to call the detail screen.

Detail screen

The detail screen shows you precisely which signals triggered the alarm message.

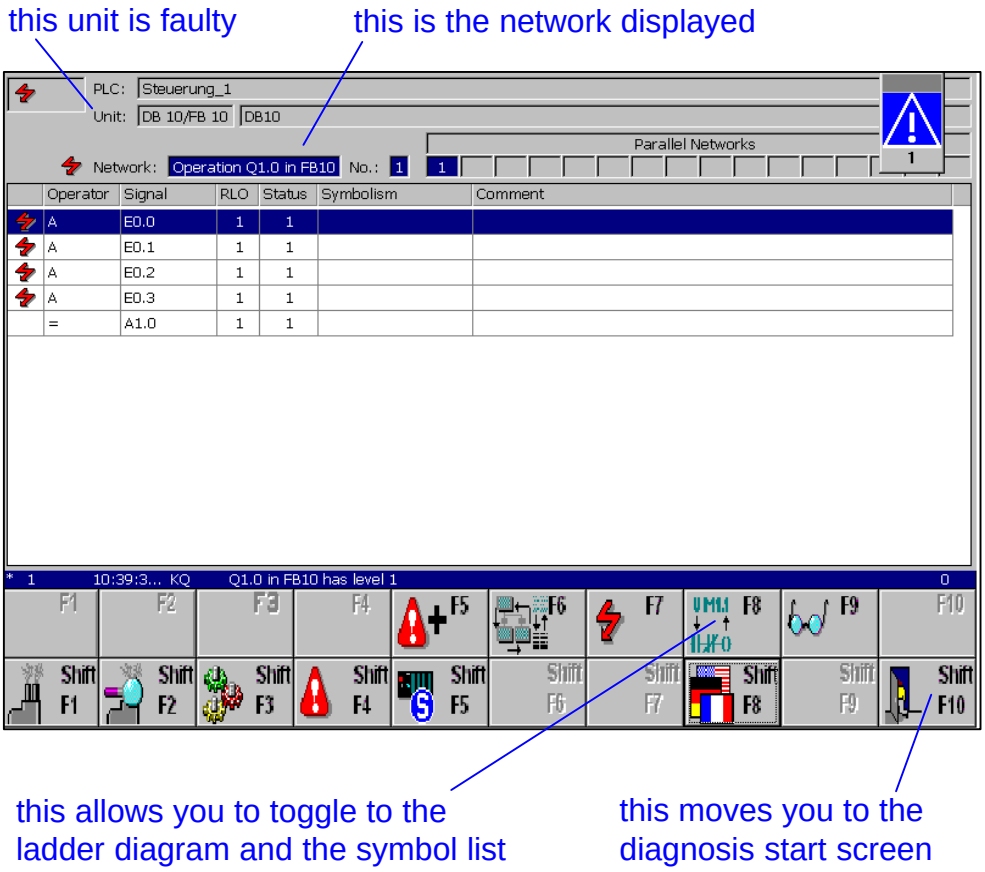


Figure 3-17 Detail Screen in STL Display

Explanations relating to the detail screen

The signals that triggered the alarm message are identified by a lightning symbol. The address monitoring defined in the *Getting Started with S7 PDIAG* chapter monitors output Q1.0. The error message is triggered when Q1.0 reaches level ”1”. This is the case in the current situation.

You can see the cause here in the central area of the detail screen:

Inputs I 0.0, I 0.1, I 0.2 and I 0.3 all have the status ”1”. In keeping with the assignment, output Q 1.0 also has the status ”1”. To clear the error, the status of at least one of the inputs must be reset to ”0”.

Display as a symbol list

For the display of program code in the central area of the detail screen, you can toggle between STL, symbol list and LAD. To do this, click on the corresponding button.

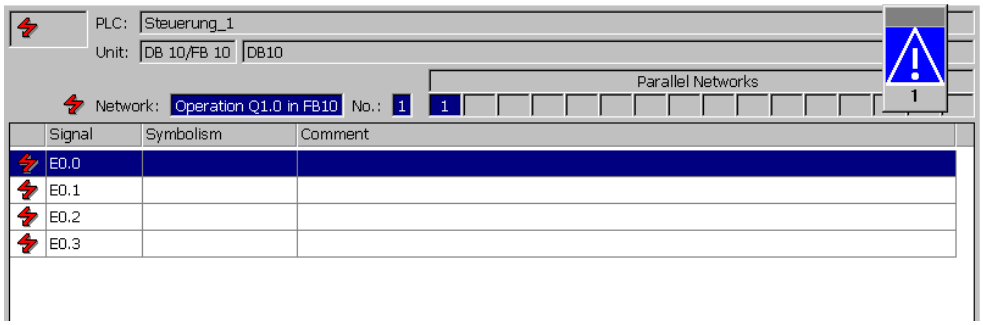


Figure 3-18 Detail Screen: Program Code as Symbol List

The display will appear as a symbol list in the central area of the detail screen.

LAD Display

Click again on the corresponding button. This will move you to the next display in the cycle:

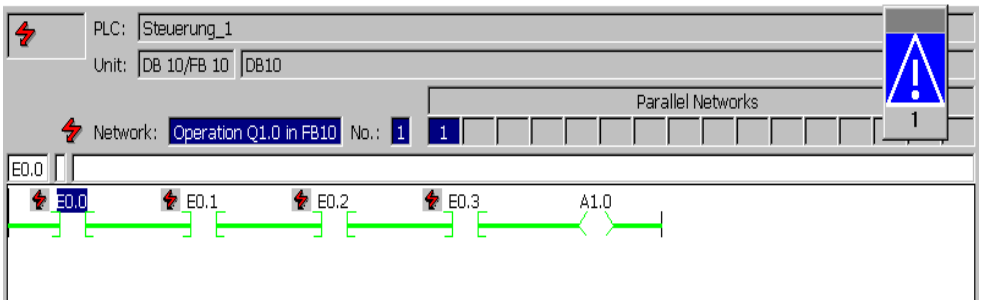


Figure 3-19 Detail Screen: Program Code as LAD Display

The display will appear as a ladder diagram in the central area of the detail screen.

Net result

You have now simulated an error, traced its alarm message on the PC and determined the cause of the error.

Additional information

In this chapter you have seen how a process diagnosis is configured in ProTool and how it is run on the operating unit. You now know the important steps and sequences.

Of course, this simple example cannot possibly demonstrate all the options ProAgent has to offer. In practice, you will normally have your own screens which you want to combine with the diagnostic screens, you may even wish to customize the diagnostic screens to suit your own particular requirements.

You will find detailed information on this and other topics in the online Help and in the ProAgent User's Guide.

Configuring Address Monitoring with S7 PDIAG

4

In This Chapter

This chapter gives you an overview of address monitoring used in process diagnostics and shows you step-by-step how to create address monitoring with S7 PDIAG.

Chapter Overview

Chapter 7 describes how to generate monitoring blocks for S7 PDIAG from your error definitions and how to download these blocks to the programmable logic controller.

4.1 Address Monitoring in S7 PDIAG

Introduction

With address monitoring, you can create error definitions which are linked to an address. This address is called the initial diagnostic address (IDA).

Address Monitoring

There are two different types of address monitoring:

1. **Level monitoring:**

   for level “0” or level “1”

2. **Edge monitoring:**

   for edge “0 >1” or “1 >0”

You can define the following parameters:

- The name of the address to be monitored
- The delay time to be set
- Whether to use initial value acquisition
- Whether to assign a message text to be displayed on the connected display devices whenever an error occurs
- The priority of the message and whether or not the message must be acknowledged.

As Level Monitoring

With level monitoring, a specific address is monitored for a defined level (0 or 1). The error state occurs when the address has the specified level for longer than the defined delay time ($\mathbf{t_{delay}}$). If the level changes during this delay time, the timer is restarted.

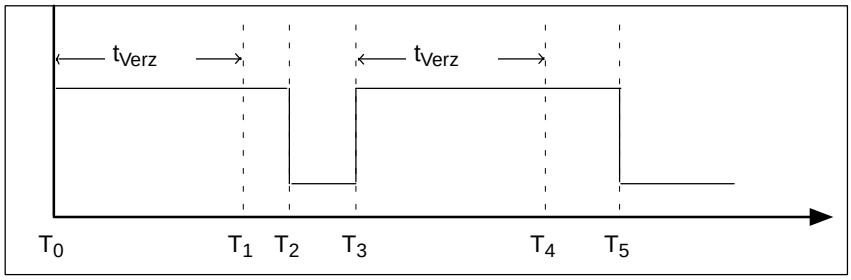


Figure 4-1 Signal States in Level Monitoring for Level “1”

Monitoring starts with the first cycle ($\mathbf{T_{o\,=}}$ startup). The delay time ($\mathbf{t_{delay}}$) is started both at $\mathbf{T_0}$ and $\mathbf{T_3}$. As soon as the defined level occurs for longer than the delay time, an error is detected and registered as “incoming” (at $\mathbf{T_1}$ and $\mathbf{T_4}$). At $\mathbf{T_2}$ and $\mathbf{T_5}$, the error is registered as “outgoing.”

As Edge Monitoring

With edge monitoring, a specific address is monitored for a defined edge (rising or falling). The error state occurs when there is an incorrect level (for example, level “1” after a rising edge) at the address after the edge change for longer than the defined delay time (**t_{delay}**). If an edge change occurs during this delay time, the timer is restarted. You can define the duration of the delay time yourself.

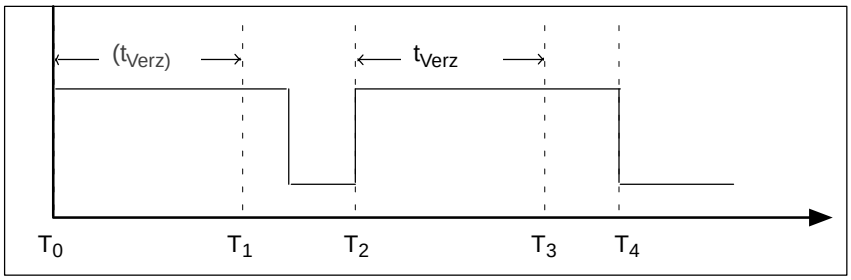


Figure 4-2 Signal States in Edge Monitoring

Edge monitoring generally behaves like level monitoring, except that with edge monitoring, the state of the address being monitored is saved at the point **T_o** (= startup); that is, the switch-on point is not interpreted as an edge.

The delay time (**t_{delay}**) does not start until the next active edge (the selected edge). This means that errors cannot be detected and registered as “incoming” until **T_3** and not at **T_1**, as would be the case with level monitoring. At **T_4**, the error is registered as “outgoing.”

Error State

With address monitoring, the error state occurs when the conditions you defined have occurred.

4.2 Overview of the Configuration Steps in Creating Address Monitoring with S7 PDIAG

Introduction

The diagram below shows you the configuration steps for creating address monitoring with S7 PDIAG.

You can find a detailed description of the procedure in the sections listed below:

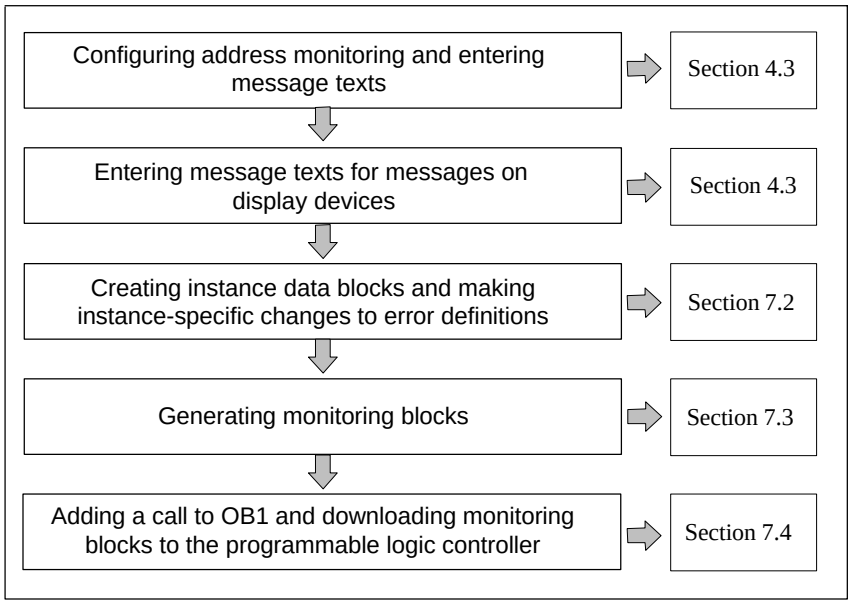


Figure 4-3 Procedure for Configuring Address Monitoring

Note : Blocks which contain preceding logic operations must also be made capable of being diagnosed (refer to Appendix A).

4.3 Configuring Address Monitoring and Entering Message Texts

Introduction

Before you can create an error definition for address monitoring, you must first select the initial diagnostic address. This is the address to which the error definition is appended. There are three ways of selecting an initial diagnostic address:

- In the incremental LAD/STL/FBD Editor
- In S7 PDIAG and
- In the symbol table (as from STEP 7 V5.0 SP3)

The various procedures are described below.

Selecting the Initial Diagnostic Address in the Editor

To select the initial diagnostic address, proceed as follows:

1. Open the block for which you want to create an error definition by double-clicking it. The LAD/STL/FBD Editor opens.
2. Position the cursor on the instruction line of the address for which you want to create an error definition and select the menu command **Edit > Special Object Properties > Monitoring**.

   Result: The “Process Monitoring” dialog box opens, as shown in Figure 4-4.

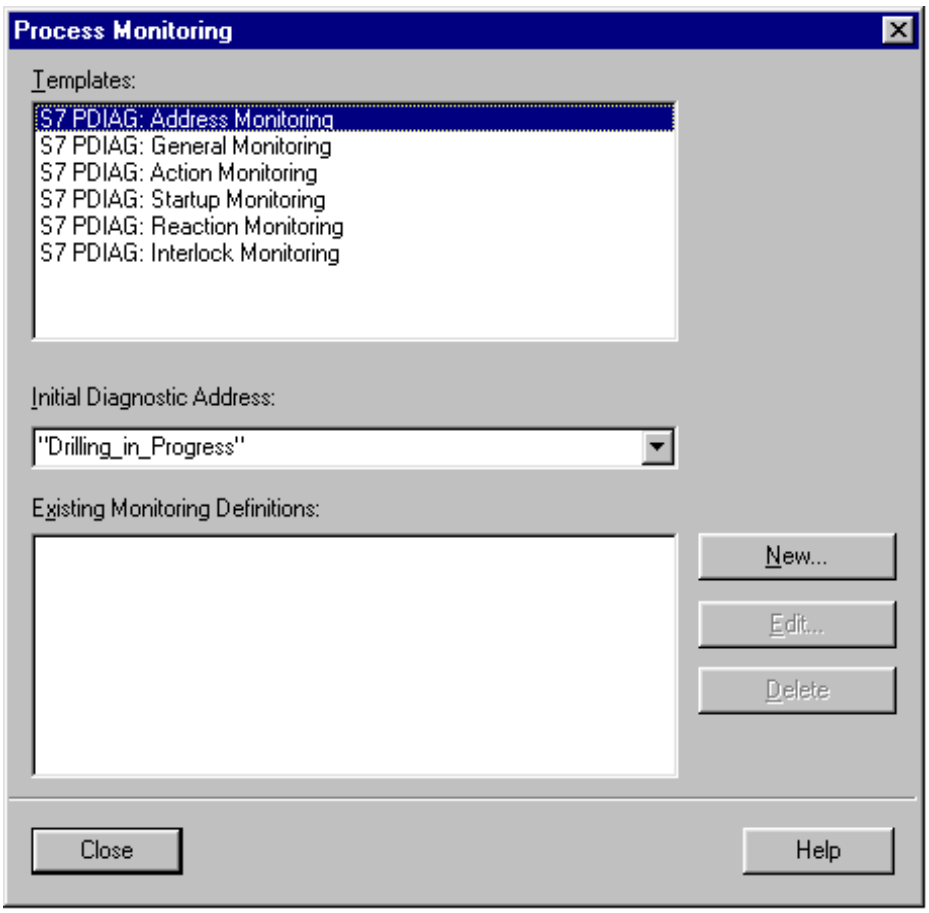


Figure 4-4 Selecting the monitoring type in the “Process Diagnostic Messages” dialog box

3. Select “S7 PDIAG: Address Monitoring” and click “New.” Fill in the “Definition” tab in the next dialog box “S7 PDIAG: Address Monitoring” according to your requirements (see Figure 4-5).

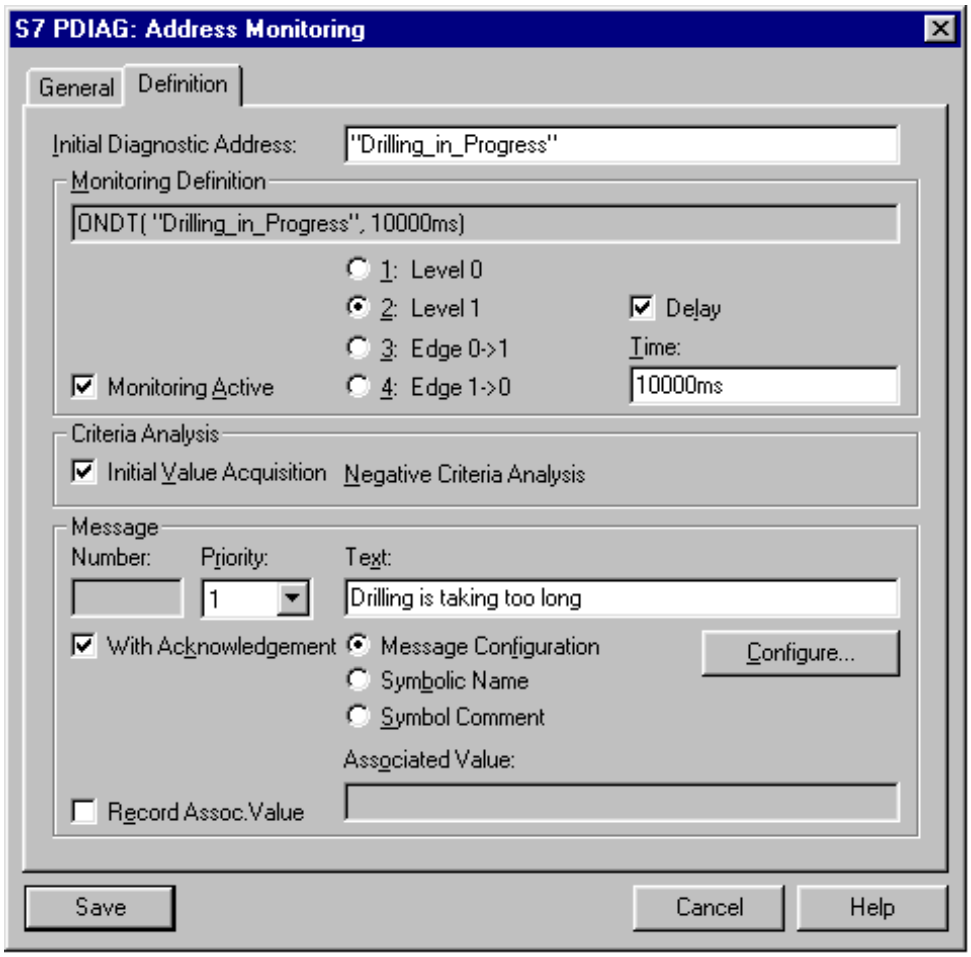


Figure 4-5 Creating Address Monitoring in S7 PDIAG

4. Enter the message text as described below.

Entering Message Texts

You can define the default setting for message texts in S7 PDIAG using the menu command **Options > Customize**.

You can enter the message texts themselves directly in the corresponding text field in the above dialog box. The following options are available:

- You can determine whether or not a message must be acknowledged by activating the check box “With Acknowledgement.”
- You can assign a priority (from 1 to 16) to any message, where “1” is the lowest priority and “16” the highest. Bits are set in a memory word corresponding to the individual priorities. You can assign the address of the memory word yourself. In this way, you can react to specific errors with different priorities in your user program. (See “User Block” and “Priority of Error Definitions” in the Glossary.)
- Here you can also configure the formal addresses for an instance-specific adaptation of the message text. The components to be replaced are marked with leading and closing characters “$$”. You will find more information on this under “Formal Addresses” in the online help for S7 PDIAG and in Chapter 7 of this manual.

If you want to configure display device-specific messages; for example, for a particular display device, proceed as follows:

1. Start the Message Configuration application by clicking the “Configure” button in the displayed dialog box.
2. Complete the tabbed sheets in Message Configuration according to your requirements. You can find further information on configuring messages in the STEP 7 User Manual and in the online help for Message Configuration.

“General” Tab

If required, you can enter the author of the error definition and a comment in the “General” tab, as well as modifying the name of the error definition.

The project path and the storage location of the monitoring definition are already entered. Exit the dialog box with “OK.”

Result: You have now configured an address monitoring definition.

Selecting the Initial Diagnostic Address in S7 PDIAG

When you select the initial diagnostic address in S7 PDIAG, you can also monitor addresses for which there are no assignments in the user program (for example, inputs). However, you cannot carry out criteria analysis on these monitoring definitions.

To select the initial diagnostic address, proceed as follows:

1. Click the block container in the SIMATIC Manager and start the S7 PDIAG application using the menu command **Options > Configure Process Monitoring**. The unit overview of S7 PDIAG is opened.
2. Select the corresponding object for which you want to create the address monitoring definition in the unit overview and select the menu command **Insert > Monitoring Definition**.
3. Continue as described earlier in the section “Selecting the Initial Diagnostic Address in the Editor” under point 3.

Selecting the Initial Diagnostic Address in the Symbol Table

Monitoring functions created by this method are contained in the unit overview of S7 PDIAG under the ”Blocks” directory.

This functionality is possible as from STEP 7 V5 SP3.

Proceed as follows:

1. Open the desired symbol table by double-clicking on it (”Symbols” object).
2. Select the symbolic name of the address for which a monitoring function is to be created.
3. Use the **Edit > Special Object Properties > Monitoring** menu command or the right-hand mouse button in the Special Object Properties > Monitoring pop-up menu to open the “**Process Monitoring**” dialog box.
4. Proceed as described under Point 3 under “Selecting the Initial Diagnostic Address in the Editor”.

Configuring General Monitoring with S7 PDIAG

5

In This Chapter

This chapter gives you an overview of general monitoring used in process diagnostics and shows you step-by-step how to create general monitoring with S7 PDIAG.

With general monitoring, you can program the monitoring logic yourself using the language elements available in S7 PDIAG. These language elements are described in Appendix B.

Chapter Overview

5.1 General Monitoring in S7 PDIAG

Introduction

General monitoring enables you to monitor several events at a time and define your own monitoring logic for these events.

General Monitoring

General monitoring provides you with a means of creating monitoring logic tailored to suit your needs using the language elements available in S7 PDIAG, and also to carry out complex error monitoring.

You can assign a message to your error definitions which will then be displayed on the registered display devices whenever an error occurs; that is, when the monitoring logic is fulfilled.

Appendix B of this manual and the online help both contain the language description for S7 PDIAG, which you will need in order to create your own monitoring logic.

Error State

With general monitoring, the error state also occurs when the conditions you defined have occured.

The following generally applies when detecting an error with a defined monitoring logic:

- If the result of logic operation is “0,” no error has been detected.
- If the result of logic operation is “1,” an error has just been detected.
- When the result of logic operation changes from “0” to “1,” an incoming error message is always generated.
- When the result of logic operation changes from “1” to “0,” an outgoing error message is always generated.

Example of General Monitoring

This section shows you an example of general monitoring. The aim is to monitor whether all three safety guards of a press are closed.

Safety guard 1: I1.0 at state 0 = safety guard open;

Safety guard 2: I3.5 at state 0 = safety guard open, and

Safety guard 3: I7.2 at state 0 = safety guard open.

Triggering: I5.0 at state 1 = press stamp down

Your monitoring logic will then look like this:

I5.0 AND NOT (I1.0 AND I3.5 AND I7.2)

Result: The error state occurs if one of the safety guards is open.

5.2 Overview of the Configuration Steps in Creating General Monitoring with S7 PDIAG

Introduction

The diagram below shows you the configuration steps for creating general monitoring with S7 PDIAG.

You can find a detailed description of the procedure in the sections listed below:

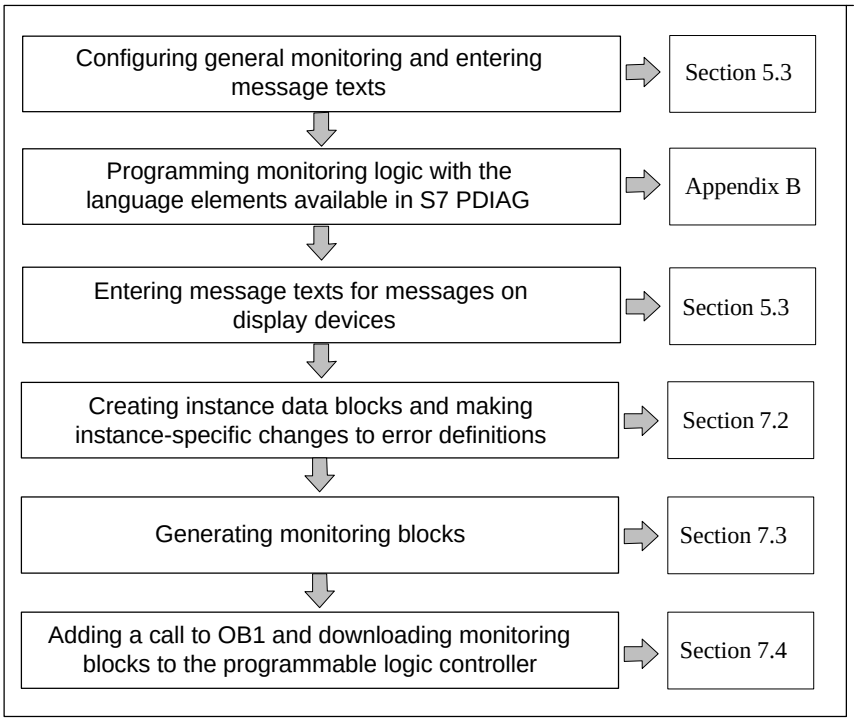


Figure 5-1 Procedure for Configuring General Monitoring

Note: You must also configure diagnostic capability for blocks with preceding logic (see Appendix A).

5.3 Configuring General Monitoring and Entering Message Texts

Introduction

Before you can create an error definition, you must first select the initial diagnostic address. This is the address to which the error definition is appended. There are three ways of selecting an initial diagnostic address:

- In the incremental LAD/STL/FBD Editor
- In S7 PDIAG
- In the symbol table (as from STEP 7 V5.0 SP3).

The various procedures are described below.

Selecting the Initial Diagnostic Address in the Editor

To select the initial diagnostic address, proceed as follows:

1. Open the block for which you want to create an error definition by double-clicking it. The LAD/STL/FBD Editor opens.
2. Position the cursor on the instruction line of the address for which you want to create an error definition and select the menu command **Edit > Special Object Properties > Monitoring**.

   Result: The “Process Monitoring” dialog box opens, as shown in Figure 5-2.

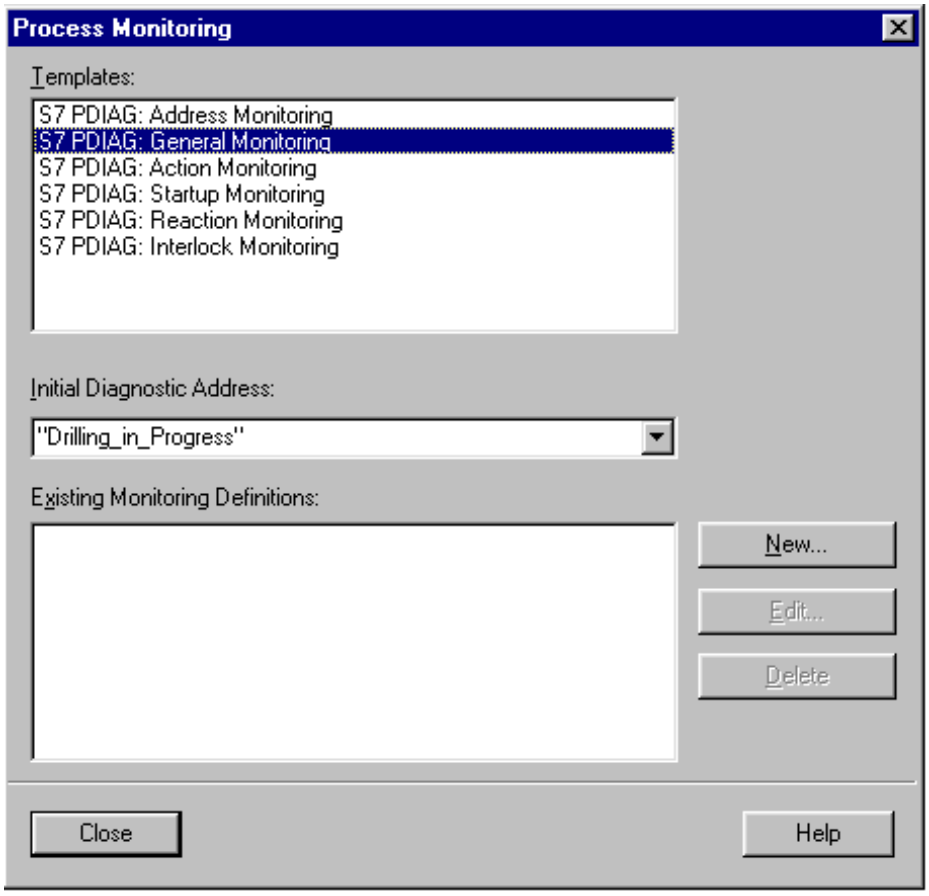


Figure 5-2 Selecting the Monitoring Type in the “Process Diagnostic Messages” Dialog Box

3. Select “S7 PDIAG: General Monitoring” and click “New.” Fill in the “Definition” tab in the next dialog box “S7 PDIAG: General Monitoring” according to your requirements (see Figure 5-3).
4. Enter your own monitoring logic in the “Monitoring Definition” field. To do this, you can use the language elements available in S7 PDIAG (see Appendix B).
   Note: The maximum number of addresses for the monitoring logic amounts to 64 addresses.

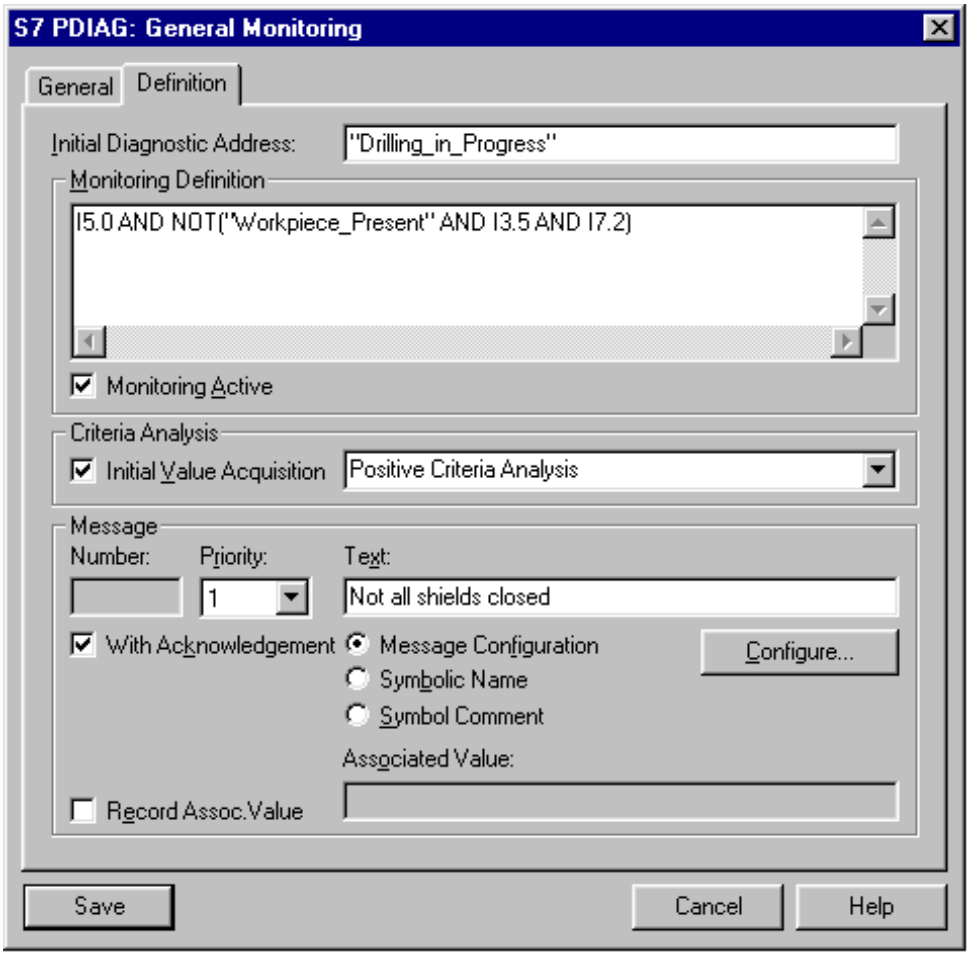


Figure 5-3 “Definition” Tab for General Monitoring

5. Enter the message text as described below.

Entering Message Texts

You can define the default setting for message texts in S7 PDIAG using the menu command **Options > Customize**.

You can enter the message texts themselves directly in the corresponding text field in the above dialog box. The following options are available:

- You can determine whether or not a message must be acknowledged by activating the check box “With Acknowledgement.”
- You can assign a priority (from 1 to 16) to any message, where “1” is the lowest priority and “16” the highest. Bits are set in a memory word corresponding to the individual priorities. You can assign the address of the memory word yourself. In this way, you can react to specific errors with different priorities in your user program. (See “User Block” and “Priority of Error Definitions” in the Glossary.)
- Here you can also configure the formal addresses for an instance-specific adaptation of the message text. The components to be replaced are marked with leading and closing characters “$$”. You will find more information on this under “Formal Addresses” in the online help for S7 PDIAG and in Chapter 7 of this manual.

If you want to configure display device-specific messages; for example, for a particular display device, proceed as follows:

1. Start the Message Configuration application by clicking the “Configure” button in the displayed dialog box.
2. Complete the tabbed sheets in Message Configuration according to your requirements. You can find further information on configuring messages in the STEP 7 User Manual and in the online help for Message Configuration.

“General” Tab

If required, you can enter the author of the error definition and a comment in the “General” tab, as well as modifying the name of the error definition.

The project path and the storage location of the monitoring definitions are already entered. Exit the dialog box with “OK.”

Result: You have now configured a general monitoring definition.

Selecting the Initial Diagnostic Address in S7 PDIAG

When you select the initial diagnostic address in S7 PDIAG, you can also monitor addresses which are not set in the user program (for example, inputs). However, you cannot carry out criteria analysis on these monitoring definitions.

To select the initial diagnostic address, proceed as follows:

1. Click the block container in the SIMATIC Manager and start the S7 PDIAG application using the menu command **Options > Configure Process Monitoring**. The unit overview of S7 PDIAG is opened.
2. Select the corresponding object for which you want to create the general monitoring definition in the unit overview and select the menu command **Insert > Monitoring Definition**.
3. Continue as described earlier in the section “Selecting the Initial Diagnostic Address in the Editor” under point 3.

Selecting the Initial Diagnostic Address in the Symbol Table

Monitoring functions created by this method are contained in the unit overview of S7 PDIAG under the ”Blocks” directory.

This functionality is possible as from STEP 7 V5 SP3.

Proceed as follows:

1. Open the desired symbol table by double-clicking on it (”Symbols” object).
2. Select the symbolic name of the address for which a monitoring function is to be created.
3. Use the **Edit > Special Object Properties > Monitoring** menu command or the right-hand mouse button in the Special Object Properties > Monitoring pop-up menu to open the “**Process Monitoring**” dialog box.
4. Proceed as described under Point 3 under “Selecting the Initial Diagnostic Address in the Editor”.

Configuring Motion Monitoring with S7 PDIAG

6

In This Chapter

First, this chapter introduces you to the concept of programming motions with S7 PDIAG.

Then, there is an overview of each type of motion monitoring which can be used in process diagnostics.

Finally, we will show you step-by-step how to create a monitoring definition in S7 PDIAG for each of the motion monitoring types listed below.

Chapter Overview

Chapter 7 describes how to generate monitoring blocks for S7 PDIAG from your error definitions and how to download these blocks to the programmable logic controller.

6.1 Advantages of Programming Motions with S7 PDIAG

Introduction

The concept of programming motions with S7 PDIAG consists of three components:

1. The supplied Ladder networks for motion programming (see Appendix A) and the corresponding data structure for the motion, the UDT_Motion (UDT2).
2. The special motion monitoring definitions with S7 PDIAG which are described in detail in the following sections.
3. The display devices which are suited to the data structure of the UDT_Motion.

Ladder Networks for Motion Programming

These networks, which you require for programming motions with S7 PDIAG, are described in detail in Appendix A of this manual. They already contain all the necessary interconnections for controlling a motion, for example:

- Checking the interlock conditions
- Controlling automatic operation
- Controlling manual operation

These networks always describe one direction in the process and are therefore required twice for each motion.

In order to make programming with these special networks easier, they have all been put together in one motion block, which can be found as FB100 in the supplied example “S7_DIAG.” These networks supply data to and remove data from the UDT_Motion (UDT2) (also supplied), which represents the interface to the display devices.

The UDT_Motion (UDT2)

The data structure of the UDT_Motion represents the interface to the display devices and is described in detail in Appendix A of this manual. The display devices recognize this data structure and in this way can access the individual data directly. The reference is determined automatically by S7 PDIAG.

The display devices can recognize from the data whether or not a motion is currently moving. The display devices can also control the motion here by setting individual bits in manual operation if this operating mode has been selected. The data structure of the UDT_Motion in FB100 is used in the supplied example “S7_DIAG.”

Data Elements of the UDT_Motion on the Display Device

You can easily resolve errors; for example, using the direct keys on the display device. Direct keys are keys which are directly linked to the digital inputs of the programmable logic controller via the digital outputs of the display device (for example, as hardware wiring or via a DP connection) and they therefore enable you to operate the motion directly.

However, the requirement for this is that you have used the UDT “Motion” in your motion programming and a standardized interface to the motion screen therefore exists. Motions are displayed on the display device in semi-graphic form. Figure 6-1 below shows the elements of the UDT_Motion represented on the display device.

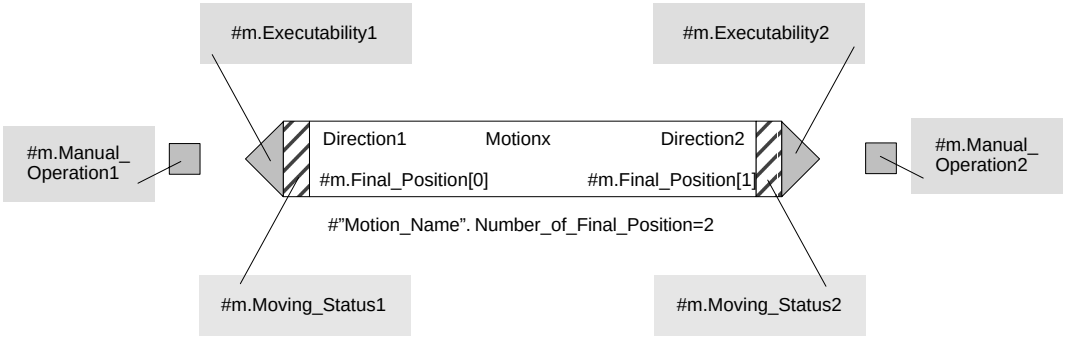


Figure 6-1 How Motions are Represented on the Display Device

Advantages of Motion Programming

If you use the concept of motion programming with S7 PDIAG, you have the following advantages:

- A motion can be monitored with error definitions and contains a group error which is formed automatically.
- A motion can be represented using the motion screens on the display device without any additional configuration being necessary.
- A motion can be visualized in all the operating modes of the machines and can also be moved using the display device (depending on the control program).
- The position and movement of the motion is displayed using up to 16 final positions and the motion status.
- The motion has two directions, and you can define the direction texts for the motion yourself. The executability can be formed for each motion direction in the user program. This shows you which motion can be moved in the current state of the machine.

6.2 Overview of Motion Monitoring in S7 PDIAG

Introduction

In processes, you often control procedures which have two stable final positions and the triggering of one final position transfers the process to the other.

For example, a cylinder can be moved from the current final position to the target final position after switching on the hydraulic pressure. In the same way, a reactor can be heated up from a current temperature to a higher value once a heating system is switched on. These procedures can be viewed as motions.

A motion is therefore part of a process which has the following characteristics:

- An object is in a current, stable final position
- A trigger is executed
- The object moves to the target final position as a result of this trigger.

Types of Motion Monitoring in S7 PDIAG

There are four predefined types of monitoring in S7 PDIAG which are specially designed for monitoring motions. You must use the Ladder networks for motion programming described in Appendix A and the UDT_Motion in order to use these definitions.

Motion monitoring definitions have a preset logic and you need only complete this. The following motion monitoring definitions are available:

1. **Action monitoring** (see Section 6.3)

   This monitors whether a motion is completed within the preset action time. This is the case when the target final position has been reached.

2. **Reaction monitoring** (see Section 6.4)

   This monitors whether a target final position which has been reached remains stable without being triggered to move in the opposite direction, or whether it is left for longer than the preset reaction time.

3. **Interlock monitoring** (see Section 6.5)

   This monitors whether the necessary interlock conditions for the motion have been met.

4. **Startup monitoring** (see Section 6.6)

   This monitors whether a motion actually starts when all the requirements have been met. This is the case when the current final position is left within the preset startup time.

The individual monitoring types are described in detail in the following sections.

Error State

With motion monitoring, the error state also occurs when the conditions you defined have occured.

Example

The following Figure 6-2 gives you an overview of how and where the individual monitoring types start in motion monitoring. The following requirements are predefined:

- There is a motion with two final positions, the current final position (CFP) and the target final position (TFP).
- Before each final position, there is a position flag (PF) and a safety guard (SG) as an interlock enable.

After the control signal, the motion is to move from the current final position (CFP) to the target final position (TFP). However, this motion may not start until the safety guard is closed.

Motion Monitoring

At the point in time T_0, the control signal is set. Interlock monitoring starts at this point. If the interlock condition is fulfilled within this time and the safety guard is closed, motion control is triggered (T_1).

The start of the motion is monitored by startup monitoring, which checks to see whether the motion leaves the current final position (CFP) within the startup time (T_2). Monitoring starts once the interlock has been enabled (T_1).

The actual motion process is monitored by the action time. This monitors whether the target final position (T_4) is reached within the action time after the interlock is enabled (T_1). The completion of the motion is detected when the target final position (T_4) has been reached.

Reaction monitoring checks whether the target final position remains stable. The reaction time starts once the position flag (T_3) is set or when the target final position has been left while the position flag is still set.

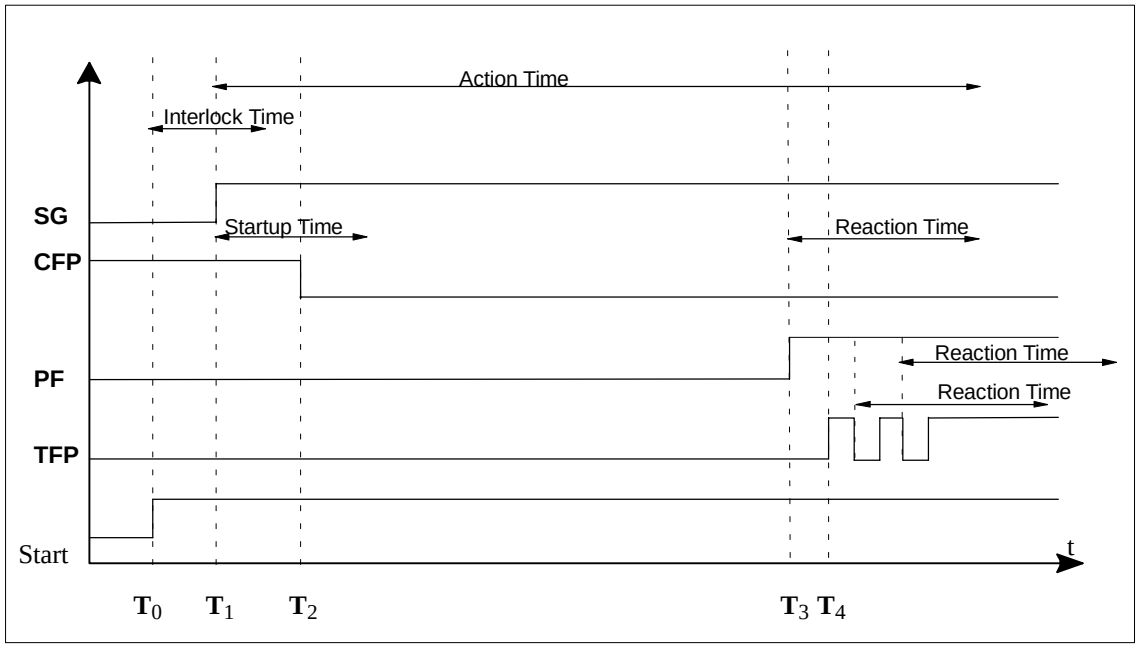


Figure 6-2 Diagram of the Individual Types of Motion Monitoring in S7 PDIAG

6.3 Motion Monitoring as Action Monitoring

Definition

Action monitoring is one of the four types of motion monitoring. Using action monitoring, you can monitor whether the target final position is reached within a specified time (action time) following a machine operation (trigger).

- The point at which the process is monitored is the target final position. This means that the execution of a motion in the process is being monitored.
- The monitoring logic is already predefined. All you need to do is add the action time and the trigger, if necessary.

Monitoring Logic

The monitoring logic for action monitoring is defined as follows:

ONDT(<Trigger>,<Action Time>)
AND
NOT <Target Final Position>

If you have used the UDT_Motion in your program, the initial diagnostic address for action monitoring is the “Final_Position[n]”. The trigger here is the enabled machine operation.

In this case, the monitoring logic is preset as follows:

ONDT(Motion_Name.Control1/2,<Action Time>)
AND
NOT Motion_Name.Final_Position[n]

“Motion_Name” is the name of the UDT_Motion in the block interface and “Control1/2” is the name of the trigger.

All you have to do here is enter the required monitoring time.

Example of Action Monitoring

A hydraulic cylinder is to transport an object from the current final position (F1) to the target final position (F2). The process leaves the current final position (F1) but is prevented from reaching the target final position (F2) within the action time despite a trigger being present.

Error Messages

The error state occurs and an error is registered as incoming when the trigger for the specified action time has been activated and the target final position (initial diagnostic address) has not become active.

The error is registered as outgoing when the final position is reached (after the action time expires) or if there is no machine operation (trigger).

6.4 Motion Monitoring as Reaction Monitoring

Introduction

Reaction monitoring is one of the four types of motion monitoring. Using reaction monitoring, you can monitor whether a final position which has been reached remains stable after a specified period of time (reaction time) has elapsed.

- The point at which the process is monitored is the target final position. In order to monitor the target final position, you will require an additional position flag which shows that reaction monitoring is active. As long as the target final position is not active while the position flag is set, the reaction time will run. If the target final position is not reached again within the reaction time, an error message is sent.
- The monitoring logic is already predefined. All you have to do is add the reaction time and the position flag, if necessary.

Monitoring Logic

The monitoring logic for reaction monitoring is defined as follows:

ONDT (<Position Flag> **AND NOT** <Target Final Position>,<Reaction Time>)

If you have used the UDT_Motion in your program, the initial diagnostic address for reaction monitoring is the “Final_Position[n]”.

In this case, the monitoring logic is preset as follows:

ONDT(Motion_Name.Position_Flag[n] **AND NOT**
Motion_Name.Final_Position[n],<Reaction Time>)

“Motion_Name” is the name of the UDT_Motion in the block interface.

All you have to do here is enter the required monitoring time.

Example of Reaction Monitoring

A hydraulic cylinder has transported an object to the target final position. However, the target final position is left, because the pressure in the hydraulic cylinder is too low.

Error Messages

The error state occurs and an error is registered as incoming when the position flag is active and the target final position (initial diagnostic address) is not active once the reaction time has expired.

The error is registered as outgoing when the target final position is reached, or if the position flag is reset.

6.5 Motion Monitoring as Interlock Monitoring

Introduction

Interlock monitoring is one of the four types of motion monitoring. Using interlock monitoring, you can monitor whether the interlock condition (executability) is fulfilled once the motion has been triggered and after a specified period of time (interlock time) has elapsed.

- The point at which the process is monitored is the executability. This monitors whether the interlock condition is fulfilled once the motion has been triggered and after the interlock time has expired.
- The monitoring logic is already predefined. All you have to do is add the interlock time and the trigger, if necessary.

Monitoring Logic

The monitoring logic for interlock monitoring is defined as follows:

ONDT(<Trigger>,<Interlock Time>)
AND
NOT <Executability>

If you have used the UDT_Motion in your program, the initial diagnostic address for interlock monitoring is the “Executability1/2”.

In this case, the monitoring logic is preset as follows:

ONDT(Motion_Name.Trigger1/2,<Interlock Time>)
AND
NOT Motion_Name.Executability1/2

“Motion_Name” is the name of the UDT_Motion in the block interface.

All you have to do here is enter the required monitoring logic.

Example of Interlock Monitoring

A car is to start moving when a button is pressed. Before the motor can start up, the brake must be released. The motor in the car is therefore locked by the brake. The trigger for “release brake” and “drive car” is given at the same time. However, the interlock does not enable the motor until the brake is released. This must be released within the specified interlock time, otherwise an error is registered.

Error Messages

The error state occurs and an error is registered as incoming when the trigger for the specified interlock time has been activated and the interlock condition (initial diagnostic address) is not yet fulfilled.

The error is registered as outgoing when the interlock condition is fulfilled or if the trigger is removed.

6.6 Motion Monitoring as Startup Monitoring

Introduction

Startup monitoring is one of the four types of motion monitoring. Using startup monitoring, you can monitor whether the current final position is left within a specified period of time (startup time) following a machine operation (trigger).

- The point at which the process is monitored is the current final position. The reaction of a motion to a trigger is therefore being monitored. This means that, in contrast to action monitoring, errors may be detected much earlier. This is particularly useful with slow processes.
- The monitoring logic is already predefined. All you have to do is add the startup time and the trigger, if necessary.

Monitoring Logic

The monitoring logic for startup monitoring is defined as follows:

ONDT(<Trigger>,<Startup Time>)
AND
<Current Final Position>

If you have used the UDT_Motion in your program, the initial diagnostic address for startup monitoring is the current final position.

In this case, the monitoring logic is preset as follows:

ONDT(Motion_Name.Control1/2,<Startup Time>)
AND
Motion_Name.Final_Position[n]

“Motion_Name” is the name of the UDT_Motion in the block interface.

All you have to do here is enter the required monitoring time.

Example of Startup Monitoring

A hydraulic cylinder is to transport an object from the current final position (F1) to the target final position (F2). Due to low pressure in the hydraulic cylinder, the current final position F1 is not left within the startup time despite a trigger being present.

Error Messages

The error state occurs and an error is registered as incoming when the trigger for the specified startup time has been activated and the current final position (initial diagnostic address) is still active.

An error is registered as outgoing when the final position being monitored is left after the startup time or if motion control is not triggered.

6.7 Overview of the Configuration Steps in Creating Motion Monitoring with S7 PDIAG

Introduction

The diagram below shows you the configuration steps for creating motion monitoring with S7 PDIAG.

You can find a detailed description of the procedure in the sections listed below:

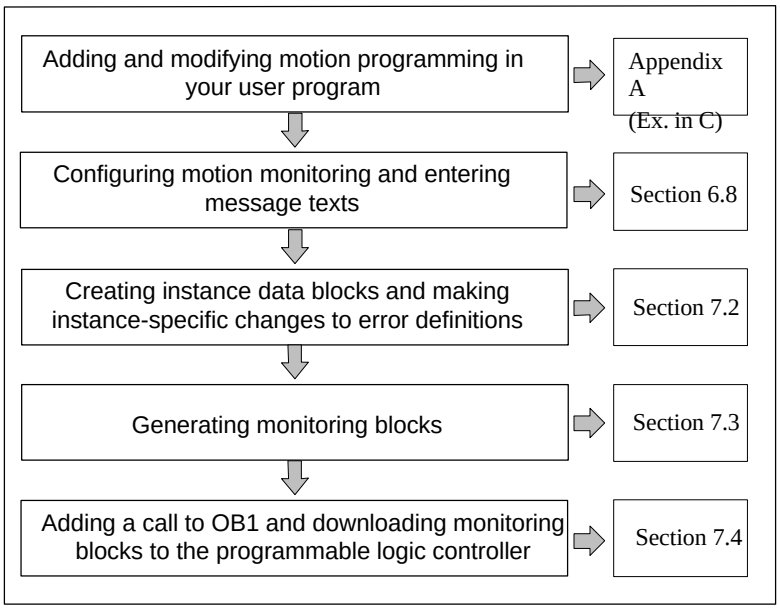


Figure 6-3 Procedure for Configuring Motion Monitoring

6.8 Configuring Motion Monitoring and Entering Message Texts

Introduction

Before you can configure an error definition for motion monitoring, you must first select the initial diagnostic address. This is the address to which the error definition is appended. There are three ways of selecting an initial diagnostic address:

- In the incremental LAD/STL/FBD Editor
- In S7 PDIAG
- In the symbol table (as from STEP 7 V5.0 SP3).

The following sections describe the various procedures.

Selecting the Initial Diagnostic Address in the Editor

To select the initial diagnostic address, proceed as follows:

1. Open the block for which you want to create an error definition by double-clicking it. The LAD/STL/FBD Editor opens.
2. Position the cursor on the instruction line of the address for which you want to create an error definition and select the menu command **Edit > Special Object Properties > Monitoring**.

   Result: The “Process Monitoring” dialog box opens, as shown in Figure 6-4.

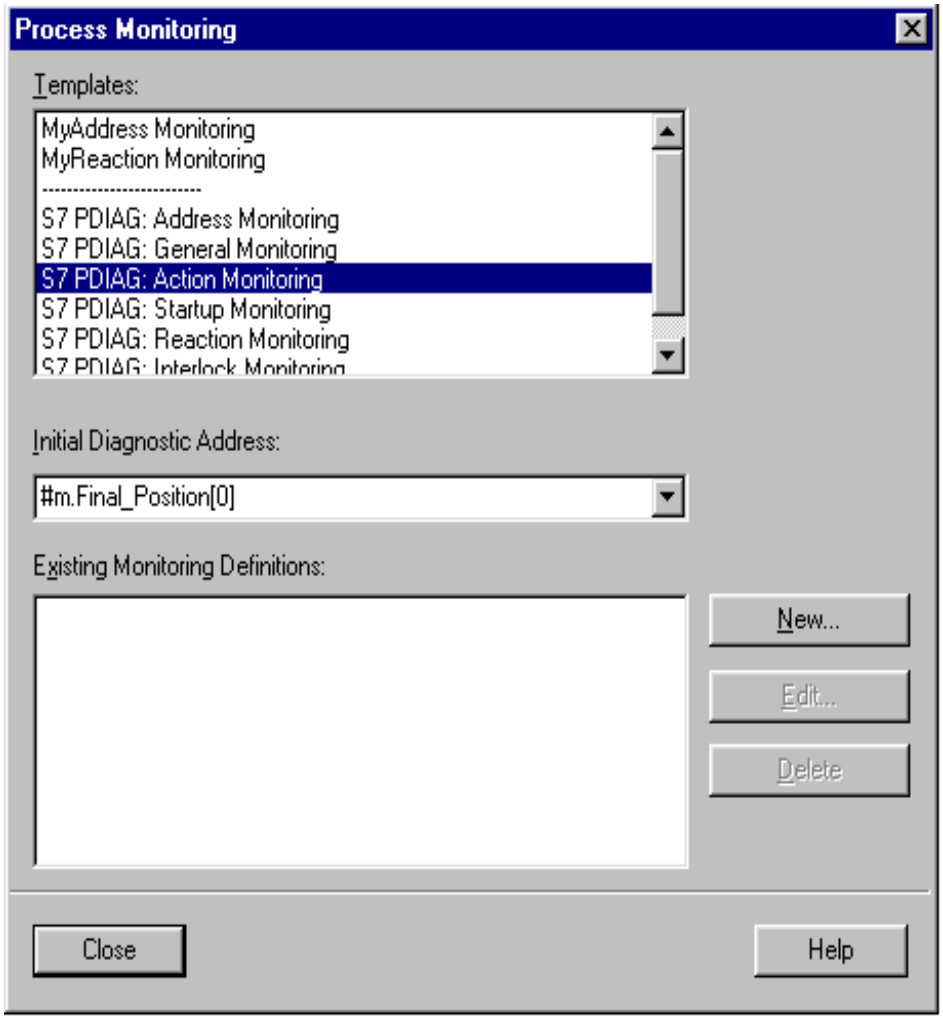


Figure 6-4 Selecting the Monitoring Type in the “Process Monitoring” Dialog Box

3. Select the required type of motion monitoring (for example, S7 PDIAG: Action Monitoring) and click “New.” Fill in the “Definition” tab in the dialog box “S7 PDIAG: Action Monitoring” according to your requirements (see Figure 6-5).
4. The monitoring logic is preset according to the monitoring type selected. All you need to do is add the required monitoring time and the trigger.

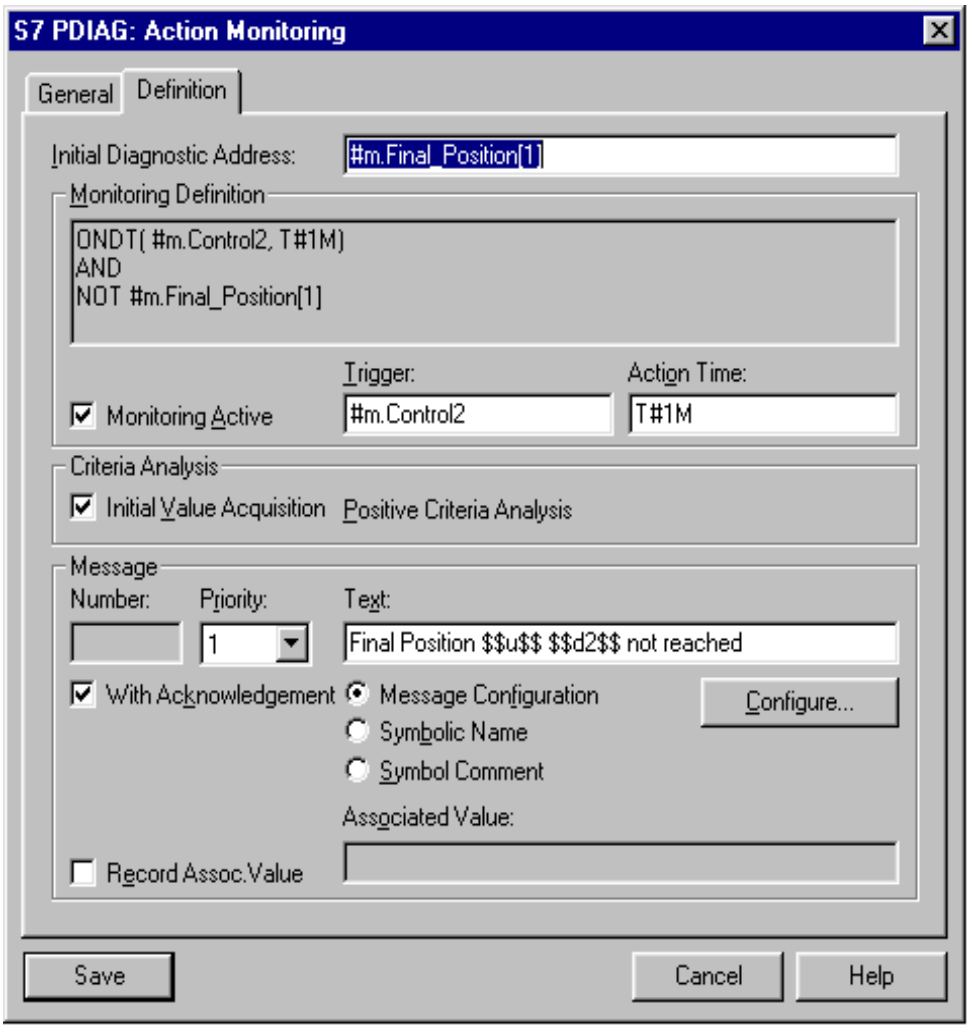


Figure 6-5 “Definition” Tab for Action Monitoring

5. Enter the message text as described below.

Entering Message Texts

You can define the default setting for message texts in S7 PDIAG using the menu command **Options > Customize**.

You can enter the message texts themselves directly in the corresponding text field in the above dialog box. The following options are available:

- You can determine whether or not a message must be acknowledged by placing a check mark next to the check box “With Acknowledgement.”
- You can assign a priority (from 1 to 16) to any message, where “1” is the lowest priority and “16” the highest. Bits are set in a memory word corresponding to the individual priorities. You can assign the address of the memory word yourself. In this way, you can react to specific errors with different priorities in your user program. (See “User Block” and “Priority of Error Definitions” in the Glossary.)
- Here you can also configure the formal addresses for an instance-specific adaptation of the message text. The components to be replaced are marked with leading and closing characters “$$”. You will find more information on this under “Formal Addresses” in the online help for S7 PDIAG and in Chapter 7 of this manual.
- You can specify that the respective message texts are to be compared automatically with the symbol table when generating instances and during compiling. To do so activate the option “Compare message texts with symbol table”.

If you want to configure display device-specific messages; for example, for a particular display device, proceed as follows:

1. Start the Message Configuration application by clicking the “Configure” button in the displayed dialog box.
2. Complete the tabbed sheets in Message Configuration according to your requirements. You can find further information on configuring messages in the STEP 7 User Manual and in the online help for Message Configuration.

“General” Tab

If required, you can enter the author of the error definition and a comment in the “General” tab, as well as modifying the name of the error definition.

The project path and the storage location of the monitoring definitions are already entered. Exit the dialog box with “OK.”

Result: You have now configured a motion monitoring definition.

Selecting the Initial Diagnostic Address in S7 PDIAG

When you select the initial diagnostic address in S7 PDIAG, you can also monitor addresses for which there are no assignments in the user program (for example, inputs). However, you cannot carry out criteria analysis on these monitoring definitions.

To select the initial diagnostic address, proceed as follows:

1. Click the block container in the SIMATIC Manager and start the S7 PDIAG application using the menu command **Options > Configure Process Monitoring**. The unit overview of S7 PDIAG is opened.
2. Select the function block which represents the motion in the unit overview and select the menu command **Insert > Monitoring Definition**.
3. Continue as described earlier in the section “Selecting the Initial Diagnostic Address in the Editor” under point 3.

Selecting the Initial Diagnostic Address in the Symbol Table

Monitoring functions created by this method are contained in the unit overview of S7 PDIAG under the ”Blocks” directory.

This functionality is possible as from STEP 7 V5 SP3.

Proceed as follows:

1. Open the desired symbol table by double-clicking on it (”Symbols” object).
2. Select the symbolic name of the address for which a monitoring function is to be created.
3. Use the **Edit > Special Object Properties > Monitoring** menu command or the right-hand mouse button in the Special Object Properties > Monitoring pop-up menu to open the “**Process Monitoring**” dialog box.
4. Proceed as described under Point 3 under “Selecting the Initial Diagnostic Address in the Editor”.

Generating and Downloading Monitoring Blocks for S7 PDIAG

7

In This Chapter

This chapter tells you how to generate monitoring blocks from the configured error definitions in S7 PDIAG and how to add them to your user program.

Chapter Overview

7.1 Overview of the Procedure

Introduction

Once you have configured the error definitions for your user program, you can then generate the S7 PDIAG monitoring blocks.

There are two steps involved in generating monitoring blocks:

- Creating instances:
  First, the error definitions and messages for the instance data blocks are generated, which means that you can still make changes to the individual instance data blocks at this point.
- Compiling:
  The program code for error detection and initial value / status acquisition for the entire user program is then generated and the monitoring blocks created.

You can then add the call for these blocks in OB1 or wherever you wish, and download the generated monitoring blocks to the programmable logic controller.

Procedure

The diagram below shows you the sequence of configuration steps:

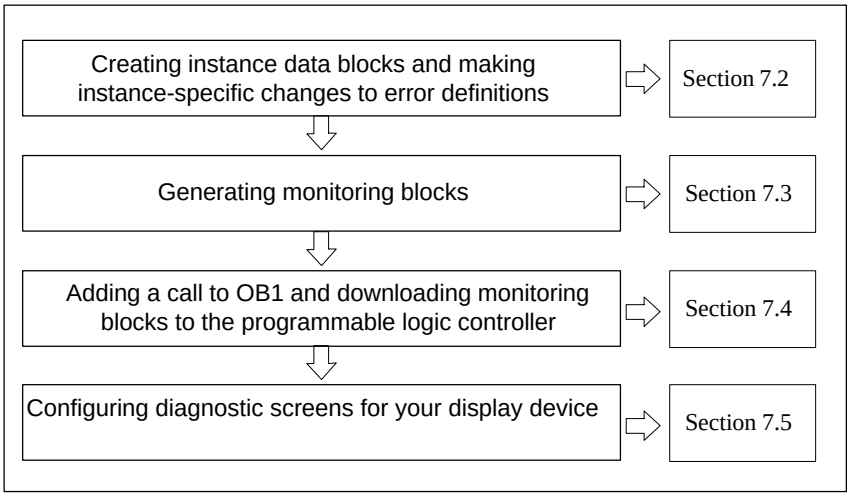


Figure 7-1 Procedure for Generating and Downloading Monitoring Blocks

7.2 Creating Instance Data Blocks and Making Instance-Specific Changes to Error Definitions

Introduction

If an identical component of a machine or plant is used several times, you can write a single function block to control this component. You can call this function block many times in your user program, each time using different data; in other words, with different instance data blocks.

However, you must be able to distinguish between these identical components on the display devices, so that an error which occurs in a specific plant component can be precisely located, for example.

This concept, also called the type instance concept, is fully supported by S7 PDIAG. Individual monitoring definitions are generated for each instance data block of a function block; these can be found in the instance data block. You can find additional information on instances of function blocks in the STEP 7 Programming manual.

Procedure

To create the instance data blocks, proceed as follows:

1. Open the block (for example, OB1) which is to call your function block with the error definitions in the SIMATIC Manager by double-clicking it. The LAD/STL/FBD Editor opens.
2. Add the call for the function block with the error definitions and the instance data block to be created with the correct name at the required location.

   Example: CALL FB30, DB30
3. Click “Yes” in the next dialog box to create the new instance data block (here: DB30) (see Figure 7-2).

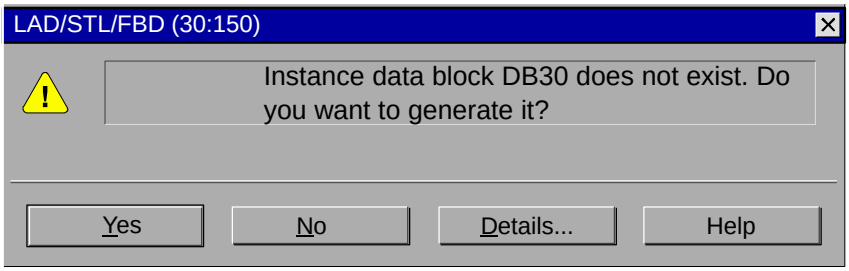


Figure 7-2 Dialog Box for Creating Instance Data Blocks

4. Save the block using the menu command **File > Save** and close the LAD/STL/FBD Editor.
5. Select the block container in the SIMATIC Manager and open S7 PDIAG using the menu command **Options > Configure Process Monitoring.**

   Result: The units relevant to S7 PDIAG are displayed in the unit overview.

6. Select the menu command **Process Diagnostics > Generate Instances**.

   Result: The error definitions assigned to the function blocks are transferred to the corresponding instance data blocks using the menu command **Generate Instances**. The message texts stored in the function blocks are added to the instances and message numbers are assigned. If you select the instance data block you created earlier (for example DB30), the instance-specific error definitions are displayed on the right-hand side of the unit overview.

   If you have positioned formal parameters in the message text, these are now replaced with data from the programmable logic controller. The final position texts for the motions are also preset. The symbolic name of the first address outside the motion is used.

7. To modify the error definitions of the individual instances, open the corresponding instance error definition on the right-hand side of the unit overview by double-clicking it.

8. Fill in the next dialog box according to your requirements and adapt the message text accordingly.

9. Exit the dialog box with “Save.”

   Note: An instance-specific, modified error definition is not overwritten again the next time instances are generated. The modified areas are marked as “blocked.”

   A deleted or unchanged error definition belonging to an instance is overwritten or created again the next time it is instantiated. Deleting an error definition which belongs to the unit of a function block also deletes the error definition for the instances of this function block.

10. If you want to overwrite all the instance-specific error definitions in a program with the data stored in the corresponding function blocks, select the menu command **Options > Customize**. Place a check mark next to the “Overwrite Instances” option in the “Compile” tab and exit the dialog box with “OK.”

    Result: All the instance-specific error definitions are now overwritten the next time instances are generated.

7.3 Generating Monitoring Blocks

Introduction

Once you have made instance-specific changes to the error definitions, you can generate the monitoring blocks. If you have grouped units, monitoring blocks are created for every group created by you during generation.

Procedure

To generate the monitoring blocks, proceed as follows:

1. Select the menu command **Process Diagnostics > Compile** in S7 PDIAG. If you are compiling for the first time, you will be requested to check the compilation settings. Confirm this message with “OK.”
2. Open the “Customize” dialog box using the menu command **Options > Customize** and, in the “Default Settings” tab, enter the settings for the blocks to be compiled for error detection and initial value / status acquisition, as shown in Figure 7-3.

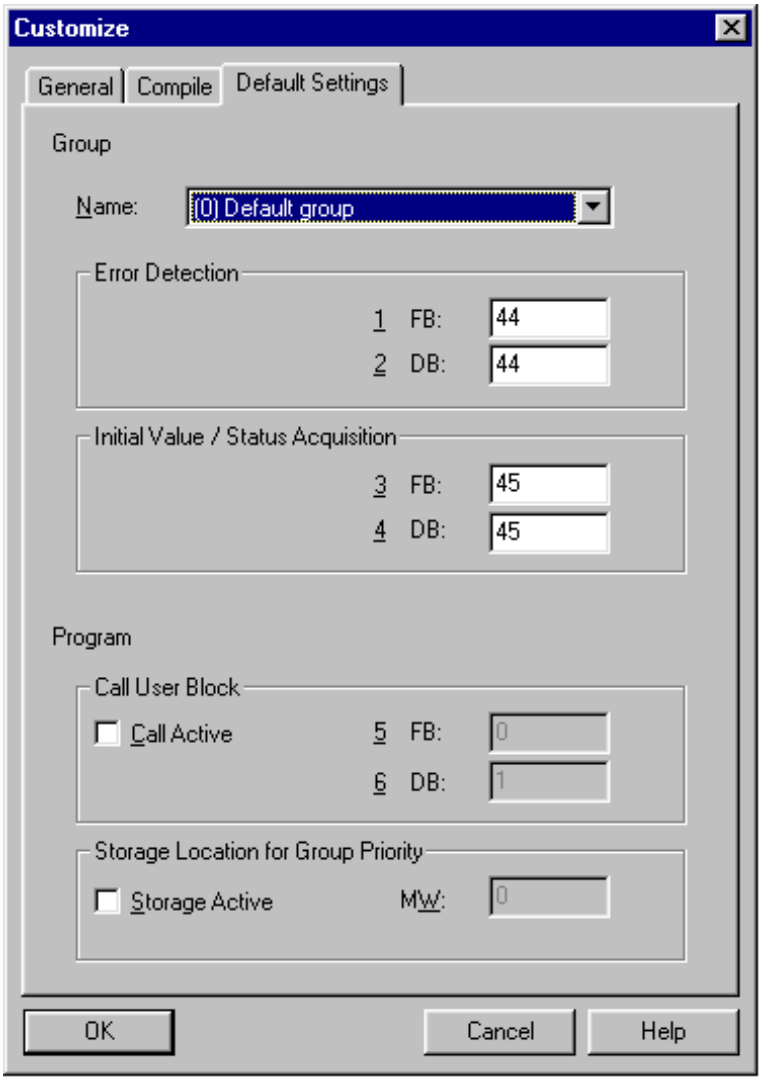


Figure 7-3 “Customize” Dialog Box

3. In the ”Name” list field select the group whose monitoring blocks you want to check or change.

Note: If you are entering different numbers for the monitoring blocks to be created, make sure that they are not already assigned to existing blocks in your user program.

4. You must also enter in the “Default Settings” tab whether you want the user block you program (see Appendix A) to be called if an error occurs and whether you want to activate group priority storage. If so, you must enter another free memory word for data storage.
5. You can define other options for generating the monitoring blocks in the “Compile” tab and set the options for presetting the message text in the “General” tab.
6. When you have finished making all your settings, click “OK” to start the compilation procedure. A progress bar appears and the monitoring blocks are generated. If an error occurs during compilation, a message will appear on the screen.

   Result: The monitoring blocks are generated with the numbers you assigned them and displayed in the SIMATIC Manager together with the system function blocks (SFC17, SFC18, and SFC64), which are also required; these are automatically included in your program.

7.4 Adding a Call to OB1 and Downloading Monitoring Blocks to the Programmable Logic Controller

Introduction

In order for the monitoring blocks you generated to become active, you must download them to your programmable logic controller and add a call for these blocks in OB1, or at the required point in your user program. This is usually at the end of the block.

Requirements

You have generated the monitoring blocks for your entire user program.

Adding a Call

To add the call for the generated error-detection block in the required block, for example, OB1, proceed as follows:

1. Open OB1 in the SIMATIC Manager by double-clicking it.
2. Insert the following lines:

   CALL FB xy, DB xy

   PDIAGCycle: = OB1_SCAN_1

   Note: You must enter the corresponding number of your error-detection block instead of “xy.” Initial value acquisition is automatically started from this block whenever an error occurs.

Downloading Monitoring Blocks and Calling Blocks

You can download the generated monitoring blocks and the calling block from the SIMATIC Manager to your programmable logic controller. Proceed as follows:

1. Select the corresponding blocks in the block container of the SIMATIC Manager.
2. Download these blocks to your CPU using the menu command **PLC > Download**.

7.5 Configuring Diagnostic Screens for Your Display Device

Introduction

Once you have finished configuring your process diagnostics with S7 PDIAG, you can now configure the diagnostic screens for your display device.

Refer to the documentation supplied with your display device for information on how to configure the screens.

Example

You will find an example of configuring the diagnostic screens with ProTool and ProAgent in Chapter 3 of this manual.

Printing and Exporting Diagnostic Data with S7 PDIAG

8

In This Chapter

In this chapter you will learn how to print out and export the diagnostic data generated by S7 PDIAG.

Chapter Overview

8.1 Printing the Diagnostic Data Generated by S7 PDIAG

Introduction

Once you have configured the error definitions for your user program and generated the corresponding monitoring blocks, you can print out the diagnostic data created with S7 PDIAG.

The standard STEP 7 layout applies. Each page has headers and footers, and the actual content of the page consists of a leader as well as the corresponding units and error definitions.

- The block numbers for the blocks generated by S7 PDIAG are printed in the leader.
- The selected units, motions, and error definitions are then printed.

Procedure

To print the data generated with S7 PDIAG, proceed as follows:

1. If you want to print all the data generated, select the uppermost object in the unit overview and select the menu command **Process Diagnostics > Print** in S7 PDIAG.
2. If you only want to print out part of the data, select the corresponding unit or monitoring definition in the unit overview of S7 PDIAG from which you want to start printing.
3. Fill in the dialog box which appears according to your requirements and confirm the print job with “OK.”

   Result: S7 PDIAG prints the data you selected.

8.2 Exporting the Diagnostic Data Generated by S7 PDIAG

Introduction

Once you have completed the process diagnostics with S7 PDIAG, you can export the diagnostic data you have generated in CSV format and use them for your own purposes. CSV format is a format which can be read with Microsoft Excel, for example, and it uses a semicolon as a separator between each of the listed elements.

You can use the exported data and a list of all the messages which occurred during the process to measure, for example, the down times and the failure frequency of your plant.

In addition, these data will help you to establish an assignment between the specified message number and the actual message.

Procedure

To export the files generated with S7 PDIAG, proceed as follows:

1. Select the menu command **Process Diagnostics > Export.** The “Export” dialog box then opens.
2. Here you enter the name of the output file or select one of the files listed. The default file type is “*.csv”.
3. Start the export procedure with “Save.”

Result: All the printable attributes of the entire program are exported.

Explanation of the Export Format

The files exported from S7 PDIAG are always displayed in the same way:

- In the first section of the file, all S7 PDIAG objects with their respective attributes are listed as a comment marked “C:”. This means that each object has its own line in the list.
- In the second section of the file, the available user data on the S7 PDIAG objects are displayed in the same way.

  Example:
  C: Block container;Name;Author;Date created;Time created; etc.
  C: Unit;Name;Author;Date created;Time created; etc.
  C: Motion;Name;Author;Date created;Time created; etc.
  C: Action monitoring; etc.

  Block Container;Blocks;;05.06.1998;07:36:42;10.06.1998;17:54:01;;
  Unit;”””Drill”””;;05.06.1998;07:38:29;05.06.1998;07:55:00;;
  Motion;”””Drill””.Clamp.m”;;05.06.1998;07:38:29;;
  Action monitoring;#m.Final_Position[0]:Action monitoring;;05.06.1998;
  07:51:54;08.06.1998;09:16:03;;;1;Yes(1);Target final position $$u$$
  $$d1$$ not reached;;#m.Final_Position[0];FB100;Yes (1);Yes (1);Positive (1);#m.Control1;T#1M;

The diagnostic data are output in the selected language. The diagnostic data can be viewed in their entirety in the export file.

Advanced Programming with S7 PDIAG

9

In This Chapter

In this chapter you will learn which additional functions are available in S7 PDIAG for advanced programming.

Chapter Overview

9.1 Creating User-Defined Templates for Monitoring Definitions with S7 PDIAG

Introduction

Using S7 PDIAG you can configure your own templates for the specific types of monitoring definition you require and then simply re-use these templates.

In general, the same procedure applies as for creating monitoring definitions.

Procedure

Proceed as follows to create a user-defined template:

1. Select the menu command **Options > Templates** in S7 PDIAG.
2. Select the appropriate monitoring definition in the following dialog box under “Templates” or “Existing Monitoring Definitions,” for example: “S7 PDIAG: Address Monitoring” and click “New.”

   Note: Existing monitoring definitions are only displayed if you have selected an initial diagnostic address.
3. The dialog box “Template (...)” opens. Complete the “General” and “Definition” tabs according to your requirements.
4. Click the “Save” button to save your template.

   Result: The template is added to the “Templates” dialog box with the name you selected (in this case, “Address Monitoring”), as shown in Figure 9-1.

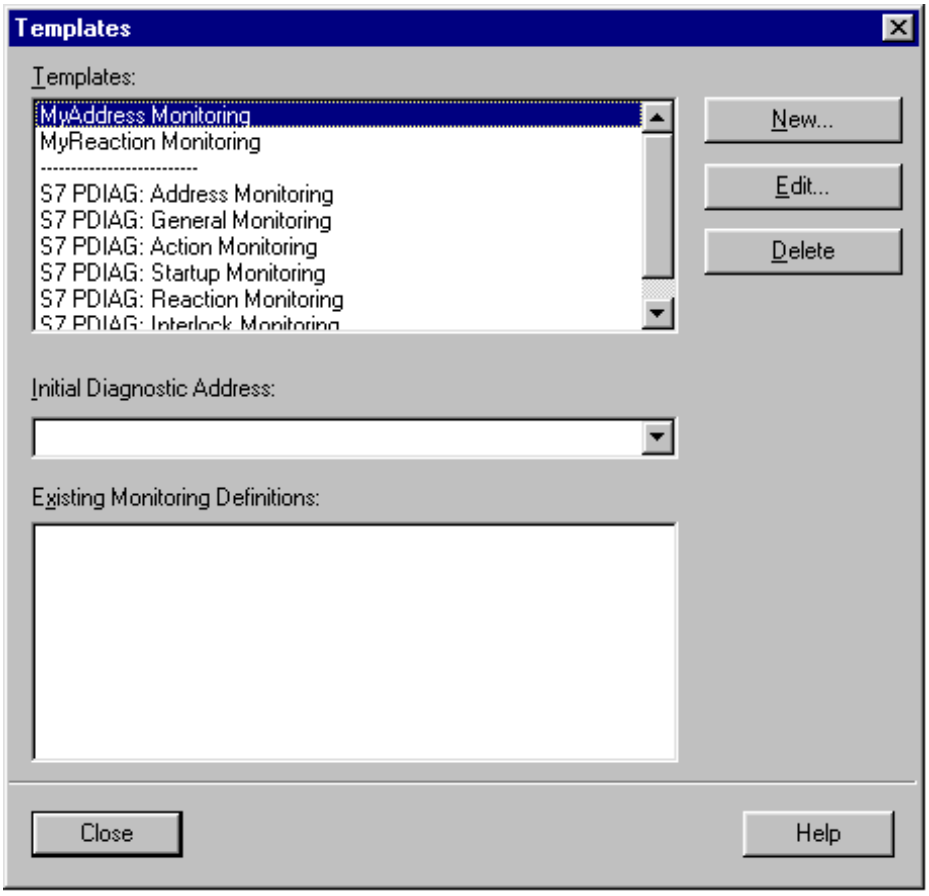


Figure 9-1 “Templates” Dialog Box

5. You can edit existing templates by selecting them in the list of templates and clicking “Edit.” You then only have to enter the changes you require.
6. Using the “Delete” button, you can delete existing templates and monitoring definitions if you have selected them.

Note: Remember that you are only creating a message template and not a real message. This is why no message number is assigned in a template.

However, you can configure the message text for your monitoring definition in the template.

9.2 Working with Standard Projects

Introduction

Standard projects are “normal” projects in which you can program everything you want to use again in other projects.

For example, you can re-use the supplied UDTs and block templates from the “S7_DIAG” standard project in your own projects.

You can also create all the special templates for error definitions in user-defined standard projects in order to use them in your projects.

The following section describes how to create a project derived from a standard project.

Procedure

To derive a user-defined project from a standard project, proceed as follows:

- Open the standard project in the SIMATIC Manager and save it under a new name with the menu command **File > Save As**.

  or:

  Create a new project and open this project together with the standard project. Now select the corresponding S7 program in the standard project which you want to copy into your project and select the menu command **File > Copy**. Paste the copied S7 program into your new project.

Modifying Standard Projects

When you modify a standard project, this does not affect any existing projects derived from this standard project.

If you have to make changes to the standard project while you are working on the derived project, you must change both projects.

9.3 Modifying Times Online / Offline

Introduction

Using the “Modify Times Online / Offline” function, you can change monitoring times both online or offline in existing monitoring definitions without having to generate the monitoring blocks again each time, because the changes you make are added to the S7 PDIAG database and the online or offline blocks.

Requirements

The monitoring definition you want to modify must meet the following requirements if you want to modify times:

- It must have already been added to the S7 PDIAG monitoring blocks in a previous compilation and the blocks must have been downloaded to the CPU
- It must be active; that is, the “Monitoring Active” check box must be set in the dialog box for the corresponding error definition
- It must contain a monitoring time which is not equal to “0.”

Procedure

In order to modify times, proceed as follows:

1. Select the compiled monitoring definition in S7 PDIAG and select the menu command **Edit > Modify Times Online / Offline** or click the corresponding button in the toolbar.

or:

1. You can also modify times online or offline from the LAD/STL/FBD Editor if you are in online or offline mode. To do this, open the block to which the monitoring definition is appended and place the cursor on the initial diagnostic address.
2. Select the menu command **Special Object Properties > Monitoring** from the pop-up menu.
3. Select the monitoring definition in which you want to modify the monitoring time in the next dialog box which appears and then click the “Modify Times” button.

4. The next dialog box displays the selected monitoring definition. However, you can only modify the monitoring time.

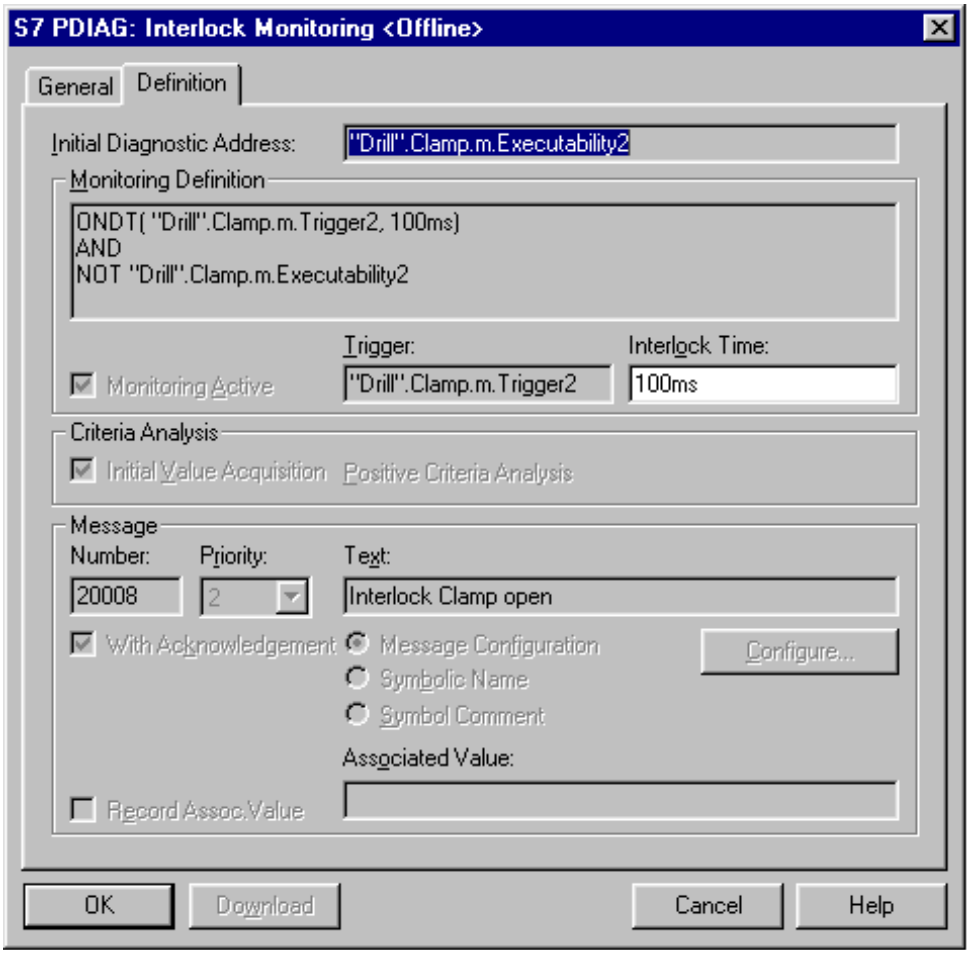


Figure 9-2 Modifying Times Offline for an Existing Interlock Monitoring Definition

– **When modifying times online...**
  Enter the new monitoring time and click the “Download” button. The modified monitoring time is then loaded to the online blocks in the CPU.

  Note: Your online data in the CPU are now no longer consistent with the offline data. If you want to make the data consistent again after modifying the monitoring time online, you must update the offline data by clicking the “Save” button.

– **When modifying times offline...**
  Enter the new monitoring time and click the “Save” button. The modified monitoring time is then added to the offline database of S7 PDIAG.

  Note: Your online data in the CPU are now no longer consistent with the offline data. If you want to make the data consistent again after modifying the monitoring time offline, you must download the modified monitoring definition to the CPU again using the menu command **PLC > Download**.

9.4 Defining Exclusion Addresses

Introduction

Criteria analysis treats all addresses in the same way when analyzing the cause of an error. This may mean that some network sections which have contributed to the error as a result of their programming logic are identified as faulty, but by using additional information, you can eliminate them as the cause of the error.

For this reason, you have the option of defining a list of “exclusion addresses” in S7 PDIAG. **These addresses and the network sections which contain them are excluded by criteria analysis if they are registered as having the value “0”** (only in conjunction with ProAgent, version 5.0 or higher).

Example of Plant Operating Modes

An example of additional information is the taking into consideration of plant operating modes. The operating mode of a plant is, by definition, coded in a number of bits in process diagnostics. Each individual bit corresponds to an operating mode. Only one bit can be set at any one time.

The example below shows the two operating modes “manual” and “automatic” and an address monitoring definition for “0”:

	STA	RLO	Faulty
A #Automatic	1	1	no
A I0.0	0	0	yes
O	1	0	yes
A #Manual	0	0	yes
A I0.1	0	0	yes
= Q1.0	0	0	yes

Viewed strictly according to the programming logic, the faulty lines have been correctly identified. However, the network section which processes the “manual” operating mode is, by definition, not the cause of the error, since the plant is already in “automatic” mode and can only be in one operating mode at a time.

It is much better for the plant operator if there are fewer erroneous lines. If you take the additional information on the operating modes into consideration, this results in a much smaller, clearer network following criteria analysis:

	STA	RLO	Faulty
A #Automatic	1	1	no
A I0.0	0	0	yes
O	1	0	no
A #Manual	0	0	no
A I0.1	0	0	no
= Q1.0	0	0	yes

Generalizing the Concept

If you generalize the concept of taking operating modes into consideration for any addresses which may be identified, you can also solve the problem of types.

In this context, the term “types” is used to refer to the difference between similar types of product which are produced in a single plant, for example:

- Car doors with electric windows
- Car doors with standard wind-down windows
- Car doors without windows, etc.

If you assign a separate address to each type, you can carry out criteria analysis in exactly the same way as with operating modes.

Identifying Addresses

All addresses which are marked for criteria analysis are called exclusion addresses. You can identify lines in your program using a list of addresses.

Procedure

In order to define exclusion addresses, proceed as follows:

1. Select the menu command **Options > Exclusion Addresses** in S7 PDIAG.
2. Enter the required addresses under “Exclusion Address” in the next dialog box which appears. You can enter both symbolic and absolute addresses.
3. Click “Insert” to add the selected address to the list of existing exclusion addresses.
4. If you want to delete any existing exclusion addresses, select the required address from the list and click “Delete.”
5. If you want to delete all the addresses in the list, click “Delete All.”
6. Click “OK” to save your entries and close the dialog box.

9.5 Working with Formal Addresses

Introduction

S7 PDIAG gives you the option of adapting message texts automatically when generating the corresponding instances and of embedding associated values in the message text. A number of formal addresses are provided for this purpose which are replaced according to the programming language used

- when compiling the error definition (generation process) or
- when displaying the message.

The components to be replaced are identified in the message text by the opening and closing characters “$$”.

9.5.1 Formal Addresses which are Replaced during Generation Process

The following formal addresses are available:

- $$u$$, $$u1$$ Name of the higher-level unit
- $$u2$$ to $$u9$$ Name of the corresponding higher-level unit
- $$ur$$ Name of the highest unit within the tree structure
- $$m$$, $$m1$$ Name of the motion
- $$o$$ Initial diagnostic address of the instance error definition
- $$d1$$ Name of the motion direction 1
- $$d2$$ Name of the motion direction 2
- $$a$$ Initial diagnostic address in absolute representation
- $$s$$ Initial diagnostic address in symbol representation
- $$c$$ Symbol comment for the initial diagnostic address
  This formal address is replaced by the symbol comment from the symbol table in the message text.

Procedure

In order to adapt message texts for a specific instance, proceed as follows:

- Enter the required formal addresses while you are configuring your message text (see “Entering Message Texts” in Chapters 4, 5, and 6).
- You can select the source of the name from the following list:
  - The names of the units
  - The name of the motion
  - The direction texts
  - The name of the instance error definition for the initial diagnostic address (IDA).

  Result: The formal addresses you use in the message texts are replaced according to the programming language used when the monitoring definitions are compiled. Depending on the formal address, the next name or the next-but-one will be used for the unit or motion, starting from the end of the name.
- The table below shows you an example:

Name	Message Text with Formal Address	Result
Drill.Clamp	Motion $$m$$ faulty	Motion Clamp faulty
Drill.Clamp	Unit $$u$$ faulty	Unit Drill faulty

9.5.2 Formal Addresses which are Replaced when Displaying Messages

Introduction

You can specify the position and the representational format of an associated value in the message text. An associated value is a value (or an address) which can accompany a message text. This value is detected by S7 PDIAG at the same time the error is detected. The associated value is shown in the message text by the display unit (HMI) at the position you have configured it. To do this, enter the corresponding formal address in the message text.

The associated value can be a parameter of the BOOL, BYTE, WORD or DWORD type from the I, Q, M or DB areas. You can specify the position and the representational format of the associated value in the message text. To do this, put together a description block for the associated value opened by the characters “@1X” and closed by “@”.

Example for an associated value:

- @1X%6d@: The associated value is to be represented as a decimal number of max 6 digits.

You can chose between the following characters of the associated value output format. The format specification is opened by the character “%”.

Format Specification	Description	Max. Representation Range
%d	Decimal with preceding sign	-2147483648 to +2147483647
%u	Decimal without preceding sign	0 to 4294967295
%X	Hexadecimal	0 to FFFFFFFF
%b	Binary	11111111111111111111111111111111
%[i]X	Hexadecimal number with i digits	
%[i]u	Decimal number without sign with i digits	
%[i]d	Decimal number with sign with i digits	
%[i]b	Binary number with i digits	

Procedure

Procede as follows to configure associated values:

- Open the monitoring definition message text in which you want to configure an associated value.
- Select the check box “Record Associated Value” and enter the address.
- Determine the position and the representational format of the associated value in the message text.
- Put together a description block for the associated value opened by the characters “@1X” and closed by “@”. The associated value is entered in the message text at the position of this description block.
- Furthermore, you can chose between the characters of the associated value output format mentioned above. The format specification is opened by the character “%”.

9.6 Grouping Units

Introduction

S7 PDIAG allows you to group units by using the menu command **Options > Group Units**.

In addition to the default standard group you can group any number of units in up to 15 different groups. Monitoring blocks are created for each group during generation. This results in smaller blocks and thus shorter generating times.

Note: You should not group the units until the end of the structuring phase has been completed when you have already created your program hierarchy completely.

Procedure

Proceed as follows in order to group units:

1. In S7 PDIAG select the **Options > Group Units** menu command.
2. The standard group is displayed in the subsequent dialog box under “Group”. The left-hand list field displays the units belonging to the standard group, as shown in the subsequent Figure 9-3:

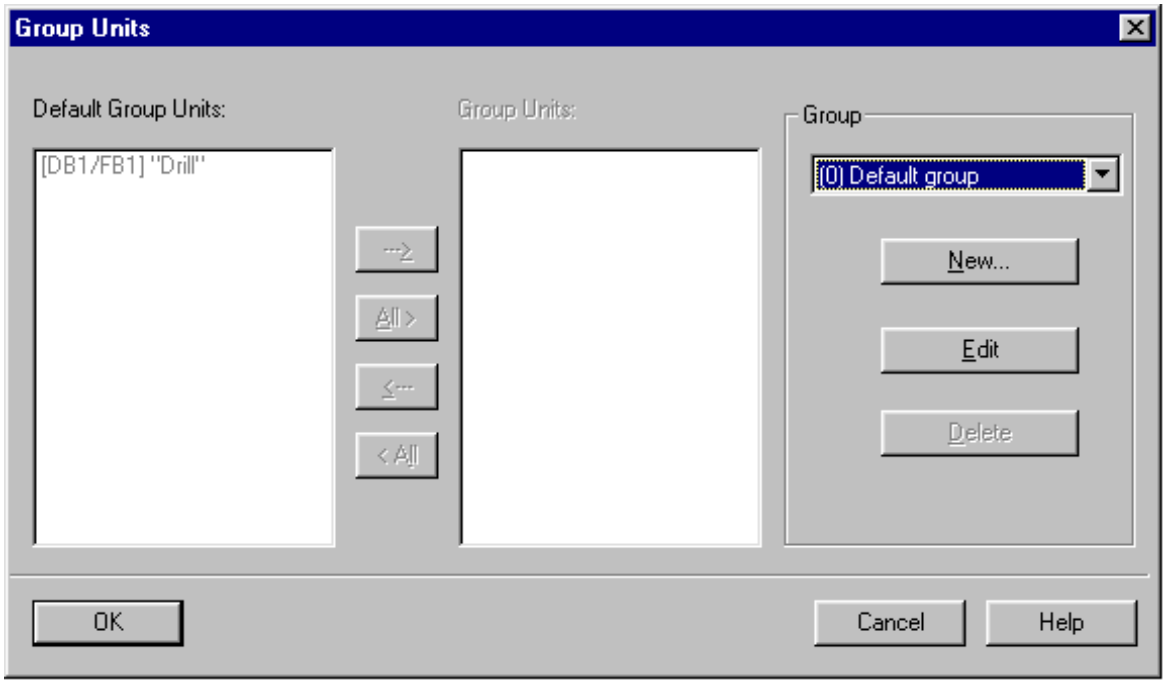


Figure 9-3 “Group Units” Dialog Box

3. Click on the “New” command button in order to create a new group. The “Group Defaults” dialog box is opened (refer to Figure 9-4):

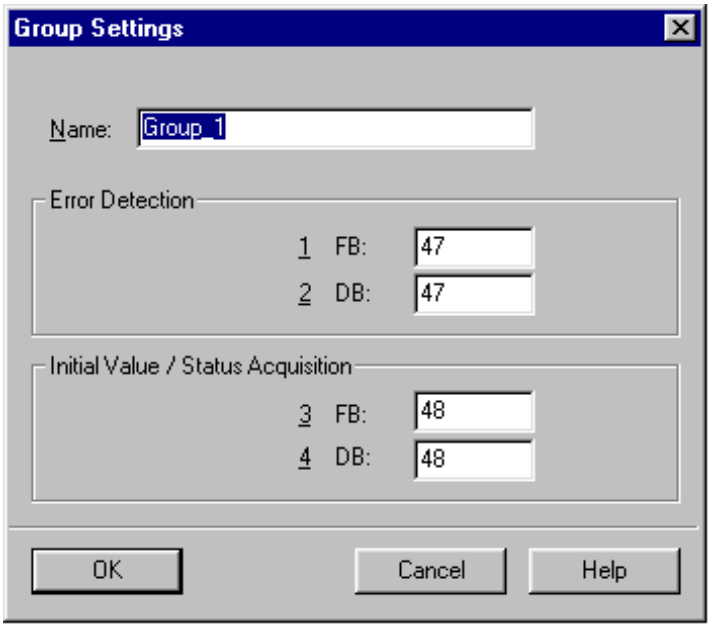


Figure 9-4 “Group Defaults” Dialog Box

4. Enter a name for the new group to be created as well as the number for the monitoring blocks for error recognition and initial-value acquisition and leave the dialog box by clicking on “**OK**”.

   Result: You have now created a new group which is displayed in the “Group Units” dialog box under “Group”.

5. In the ”Group Units” dialog box now move units from the standard group to the new group by selecting the desired units in the standard group and clicking on the corresponding command buttons (>, All >). You can also use the ”>” and ”< All” command buttons to move units back to the standard group.
6. In order to change the group defaults for the selected group again click on the “**Edit**” command button.
7. The “Group Defaults” dialog box is opened and you can change the settings. Leave the dialog box by clicking on **“OK”**.
8. You can use the “**Delete**” command button to delete the selected group, with the exception of the standard group. The units assigned to this group are moved back to the standard group.
9. After you have carried out all the settings, leave the dialog box by clicking on “**OK”**.

Result: The new group is identified in the unit overview of S7 PDIAG by a small number of a red background, as shown in the subsequent Figure 9-5:

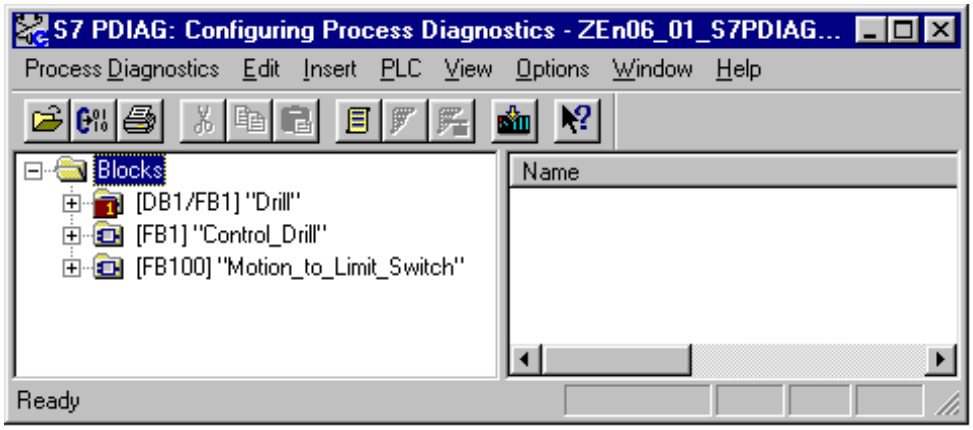


Figure 9-5 Display of “Group 1” in the unit overview of S7 PDIAG

Note: When you have created groups and copy the corresponding blocks, the group information is **not** copied as well. The group information is **not** copied either when you select all the blocks in the “Blocks” directory and then copy these.

The group information is only included in the copy when you select and copy the “Blocks” directory, since the information in contained in the directory itself.

9.7 Searching for and Editing Objects in S7 PDIAG

Introduction

You can search in S7 PDIAG for the following objects and then edit them:

- S7 PDIAG error definitions,
- Other error definitions (relevant to S7 GRAPH and S7 HiGraph),
- Units
- Motions and
- Templates

In order to edit the search result you can call up a pop-up menu with the right-hand mouse button which depends on the respective objects selected.

Procedure for Searching for Objects

Proceed as follows to search for the above objects:

1. In S7 PDIAG select the **Options > Find** menu command.
2. In the subsequent ”Search In - ...” dialog box specify what you want to search for.
   If you want to search for units or error definitions, specify whether you want to search for:
   – Types (FB),
   – Instances (DB) or
   – Other (OB, FC, Global)

   Sensible defaults are always offered for the search.
3. If subordinate objects are to be included in the search, activate the corresponding check box.
4. Click on the “**Start**” command button. The search starts from the selected object in the unit overview.

   Result: The found objects are displayed in a list.
   The list displays the name of the found objects as the first value on the left. This is followed by object-type specific information, such as: Diagnostic entry address, monitoring, initial-value acquisition, acknowledgement, priority, display class and message type for the found object.

   In the case of instance error definitions and other error definitions the respective message number for the found objects are displayed in the search result, thus facilitating the assignment to the messages at the display device considerably.

In addition the number of found and selected objects is displayed as shown in the subsequent Figure 9-6:

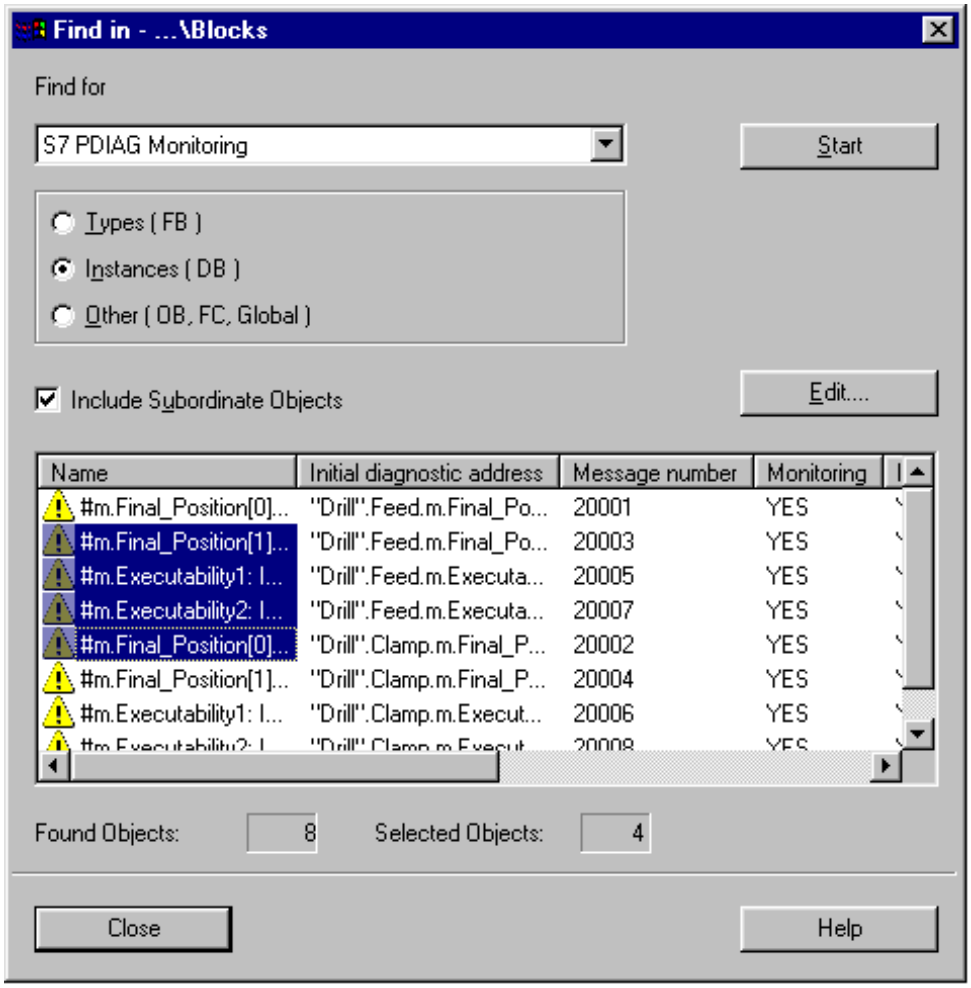


Figure 9-6 "Search In - ..." Dialog Box with Display of the Search Results

Procedure for Editing Objects

Proceed as follows to edit the found objects:

1. Select the desired objects in order to edit them. You have various possibilities:
   - Select **one or more objects** and then click on the “Edit” command button. Depending on whether you have selected a unit, a motion or an S7 PDIAG error definition, the corresponding “**Edit**” dialog box is opened. Here you can change the existing settings.
   - You can also call the “**Edit**” dialog box via the pop-up menu by using the right-hand mouse button and then change the settings.
   - If you have only selected **one object**, you can use the pop-up menu or double-click on it to have the object properties for this object displayed and edit them.
   - If you have selected an **instance error definition**, you can use the pop-up menu to have the corresponding monitoring type displayed and edit it.
2. Depending on the selected object (error definition, unit or motion) the corresponding “Edit” dialog box is opened. Further information on the respective “Edit” dialog box can be obtained in the online help of S7 PDIAG which you can call up via the F1 key or the “Help” command box.
3. Carry out the desired settings and leave the “Edit” dialog box and save your settings by clicking on “**OK**”.

9.8 Reference Data Created by S7 PDIAG

Introduction

Whenever generation has been carried out successfully, S7 PDIAG saves the generated reference data in the STEP 7 database. The corresponding STEP 7 functionality can be used to display and filter these reference data.

Generated Reference Data

The reference data created by S7 PDIAG encompass the data listed in the table below:

Reference Data:	Displayed In:
Calls of the error recognition blocks with data block	Cross-reference list and program structure
Calls of the initial-value acquisition blocks with data block	Cross-reference list and program structure
Call of the user block with data block	Cross-reference list and program structure
Assignment of the flag word (storage of the group error priority)	Cross-reference list and assignment plan

Prerequisites for Creating Reference Data

The reference data for the initial-value acquisition and error recognition blocks are created automatically during every generation process. The reference data for calling up the user blocks or for storing the group error priority are only created if you have activated these in the “Defaults” tab card of the “**Settings**” dialog box, as shown in Figure 9-7:

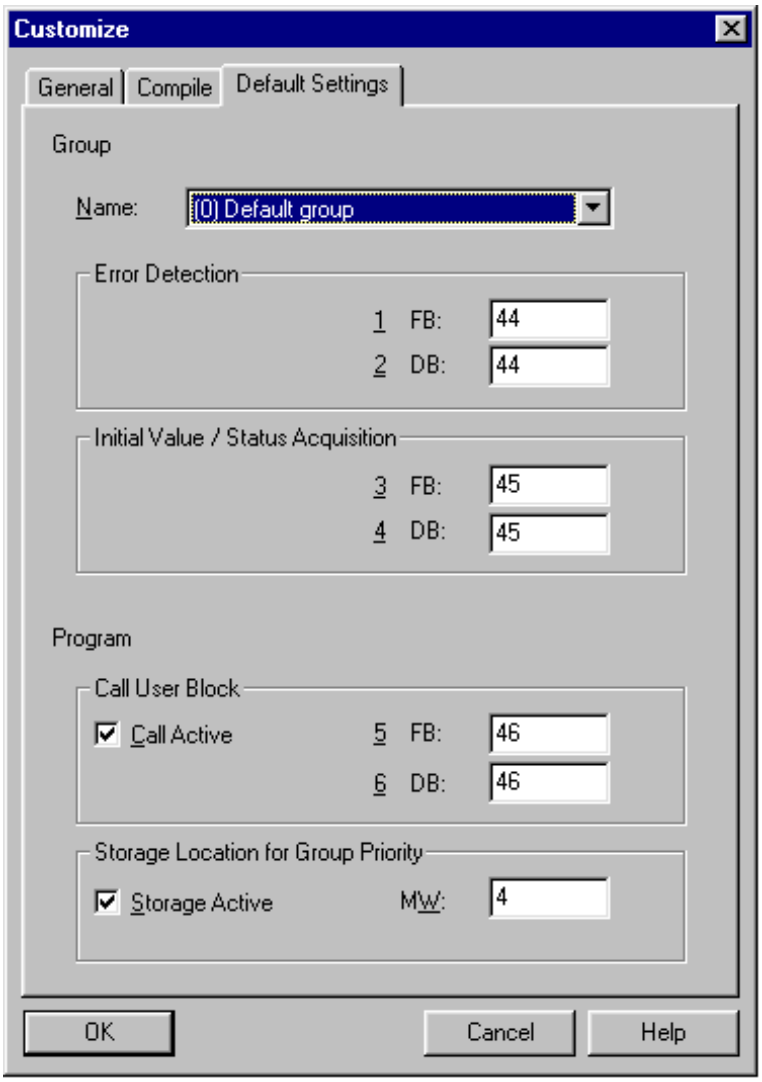


Figure 9-7 Settings for Creating Reference Data

Displaying and Filtering Reference Data

Proceed as follows in order to display or filter reference data:

1. In S7 PDIAG select the **Options > Reference Data** menu command.
2. The commands in the subsequent menu can be used to:
   - Have the reference data displayed or
   - Specify filter conditions for displaying the reference data.
3. For detailed information on the further procedure please refer to your current STEP 7 programming manual.

9.9 The Motion Screen as an Interface to the Display Device

Introduction

The properties of the selected motion are displayed in graphical form in the “Motion Screen” dialog box. The “Movement screen” dialog box orientates itself as far as possible to the representation of the movement screen at the display device.

Procedure

You can call this dialog box in S7 PDIAG by using the **Edit > Motion** menu item if you have selected a motion in the left-hand section of the S7 PDIAG unit overview.

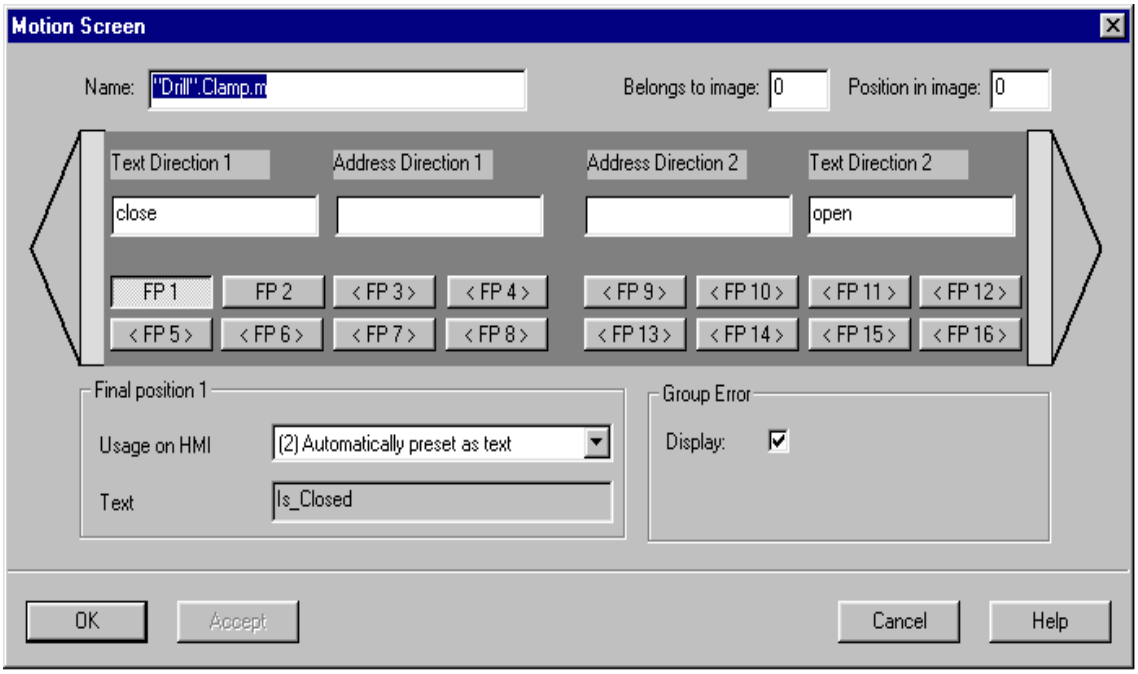


Figure 9-8 “Motion Screen” Dialog Box

Note: You can use the **Edit > Object Properties** menu command to obtain the identical information for a selected motion in non-graphical form.

9.10 Diagnostic-Relevant Network Data

Introduction

In the past the diagnostic-relevant network data were written to the data storage of S7 PDIAG. These data are then modified and transferred to the display devices depending on the respective exclusion addresses.

This meant that you also had to update the data for the display devices after every S7 PDIAG generation run.

New Functionality

As from V5.0 of S7 PDIAG you can specify that the diagnostic-relevant network data are to be written to the initial-value acquisition blocks generated by S7 PDIAG (under consideration of the exclusion addresses). From there they are read by the display device (HMI) as required.

This has the advantage that you only have to generate in S7 PDIAG, depending on the change, and that the display devices (HMI) do not have to be updated every time.

Conditions

The following conditions have to be considered:

- When you add or delete error definitions or edit existing error definitions, a message is displayed after the generation run informing you which new groups have been created. These data also have to be updated for the corresponding display devices (HMI).
- Changes to the LAD/STL/SFC Blocks:
  - If you change blocks in the LAD/STL/SFC Editor, a message is also displayed after the generation run informing you which new groups have been created. You can decide whether you want to update these data for the display devices (HMI) or not. These data do not have to be updated after changes within networks.
  - If you change the name or the number of networks, an incorrect name or an incorrect number is displayed at the display device for this network. However, the criteria analysis is correct. In this case it is up to you whether you update the diagnostic-relevant network data for the display devices.
  - If you delete networks which contain multiple assignments or if you add networks which create multiple assignments (which thus contain an address which has been assigned several times), you have to update the diagnostic-relevant network data for the display devices (HMI), since it will otherwise not be possible to carry out the criteria analysis.

- This form of saving diagnostic-relevant network data is particularly suitable for the commissioning phases. After you have completed commissioning we recommend that you save the diagnostic-relevant network data in the display device (HMI) in order to increase performance and the memory available.
- The functionality described above is not supported by ProTool/ProAGENT below V 5.3.

Notes on Programming Your User Program

A

In This Chapter

In this chapter, you will learn how S7 PDIAG supports you when programming or modifying your user program.

You will also learn how to install the UDT_Unit, the UDT_S_Unit, and the UDT_Motion as well as how to create motion monitoring using the Ladder networks in FB100 of the sample program “S7_DIAG” supplied with the software.

Chapter Overview

A.1 How Does S7 PDIAG Support You When Programming Error Definitions?

Introduction

S7 PDIAG supports you when assigning error definitions to a functional unit or motion by:

- Providing user-defined data types (UDTs)
  - The UDT_Unit (see Section A.2)
  - The UDT_S_Unit (see Section A.3)
  - The UDT_Motion (see Section A.4)
- Providing Ladder networks which describe the assignment of parameters to the UDT_Motion, for example (see Section A.5).
- Providing auxiliary networks which support you when carrying out criteria analysis if you have also programmed preceding logic operations.

What is a UDT?

A UDT is a user-defined data type which can be saved as a block. This means that you can create one UDT and then use it many times: on the one hand as a “normal” data type, and on the other hand as a template for creating blocks with the same data structure.

For additional information on UDTs, refer to the reference manual *Statement List (STL) for S7-300 and S7-400*.

The UDT_Unit

The UDT_Unit represents a functional unit with its own operating modes. The following has already been defined in the UDT_Unit:

- Group error detection
- 16 operating modes, two of which are predefined as “manual operation” and “automatic.” You can define the remaining 14 modes according to your own requirements.

You will find a detailed description of the UDT_Unit in Section A.2.

The UDT_S_Unit

The UDT_S_Unit represents a functional unit without its own operating modes. It contains the following:

- The group error address
- The group error acknowledgement.

You will find a detailed description of the UDT_S_Unit in Section A.3.

The UDT_Motion

The UDT_Motion represents a standard interface between S7 PDIAG and the display devices (OPs) and contains all the parameters for:

- Displaying motions in motion screens on the display device without the need for additional configuring
- Moving these motions manually in the motion screen on the display device.

The requirement for this is that you have used the Ladder networks supplied in FB100 for motion programming.

You will find a detailed description of the UDT_Motion in Section A.4.

Group Error Bit ID

There is a bit for group error detection in all UDTs. This is a bit which is set by S7 PDIAG in the case of an error. When S7 PDIAG recognizes an error, it sets the group error bit in all units and motions above the error in the hierarchy.

Example

A stamping machine contains the elements “Press,” “Safety Guard,” and “Punch,” where the safety guard and the punch are, in turn, components of the press.

The press is a unit in the sense of S7 PDIAG. The punch is a motion which is enabled by the safety guard.

In order to coordinate these objects with one another, you need to program a “coordination” function block which uses the UDT_Unit and the UDT_Motion.

If you now receive a group error message on your display device informing you that the unit “Press” is faulty, you can position the cursor in the motion screen on the unit “Press” and move down a level to display the three motions for the press, punch, and safety guard. Here, you will see that a group error bit is also set in the motion structure of the punch.

Result: The press is therefore faulty because the punch is faulty. The fault in the punch has caused the error definition.

Use: Using the UDT_Unit, the UDT_S_Unit, and the UDT_Motion enables you to determine which error definition was caused by means of data reduction, and to then remedy this error manually.

Preprogrammed Ladder Networks

S7 PDIAG provides you with preprogrammed Ladder networks for programming motions in FB100 of the supplied project “S7_DIAG,” which assigns the parameters to the UDT_Motion, for example.

Using these networks for motion programming reduces the amount of programming you have to do yourself to simply entering the parameters you are using in the network sections.

The advantage of using these networks, as well as convenient motion programming, is that you can actually influence motion processes manually in a motion screen on the display device, for example, and then remedy the error.

You will find a detailed description of these Ladder networks for motion programming in Section A.5.

Using Auxiliary Networks

S7 PDIAG can substitute auxiliary addresses (bit memory) in your programming logic so that they are replaced with the real networks you create. S7 PDIAG uses these networks for criteria analysis.

Example: You have used the following logic operation in a function block with diagnostic data:

I1.0 AND I1.1 = M1.0

The following then appears in another block with diagnostic data:

I1.2 AND M1.0 = Q1.1

After using the auxiliary network, the following results:

Q1.1 = (I1.2 AND (I1.0 AND I1.1))

When using auxiliary networks, brackets are always used to break down the original logic operation carried out with a memory bit.

Note: Please note that this option must have been activated under “**Options > Settings**”.

A.2 What is the UDT_Unit?

Introduction

The UDT_Unit contains all the necessary data of a process unit and can be inserted in the variable declaration table of a function block:

- In the areas: “In,” “Out,” and “Stat”

An “Array of UDT_Unit” is not permitted.

The UDT_Unit is characterized by the fact that it has the attribute “S7_pdiag_unit” and this attribute is set to “TRUE.”

Using the UDT_Unit

The UDT_Unit represents a process unit in your user program. All of the components below this process unit in the hierarchy should relate to the operating mode of this process unit.

Example: There are many individual presses in a factory. These are relatively independent of one another and may also be in different operating modes (manual, automatic). Each of these presses can be seen as a process unit which contains other units (for example, a stamp in the press, the safety guard, etc.), but it is not a good idea to control these subordinate units independently from one another in different operating modes. For this reason, the subordinate units obtain the operating mode from their respective process unit.

The UDT_Unit is is stored as “UDT1” in the supplied example.

Data Structure of the UDT_Unit

The table below shows the data structure of the UDT_Unit. You **cannot** modify this default data structure.

Address	Variable	Data Type	Initial Value	Comment
0.0		STRUCT		
+0.0	Unit_Version	WORD	W#16#0	Version of UDT
+2.0	Select_Automatic	BOOL	FALSE	1st operating mode selection
+2.1	Select_Manual	BOOL	FALSE	2nd operating mode selection
+2.2	Select_Operating_Mode2	BOOL	FALSE	3rd operating mode selection
+2.3	Select_Operating_Mode3	BOOL	FALSE	4th operating mode selection
+2.4	Select_Operating_Mode4	BOOL	FALSE	5th operating mode selection

+2.5	Select_Operating_Mode5	BOOL	FALSE	6th operating mode selection
+2.6	Select_Operating_Mode6	BOOL	FALSE	7th operating mode selection
+2.7	Select_Operating_Mode7	BOOL	FALSE	8th operating mode selection
+3.0	Select_Operating_Mode8	BOOL	FALSE	9th operating mode selection
+3.1	Select_Operating_Mode9	BOOL	FALSE	10th operating mode selection
+3.2	Select_Operating_Mode10	BOOL	FALSE	11th operating mode selection
+3.3	Select_Operating_Mode11	BOOL	FALSE	12th operating mode selection
+3.4	Select_Operating_Mode12	BOOL	FALSE	13th operating mode selection
+3.5	Select_Operating_Mode13	BOOL	FALSE	14th operating mode selection
+3.6	Select_Operating_Mode14	BOOL	FALSE	15th operating mode selection
+3.7	Select_Operating_Mode15	BOOL	FALSE	16th operating mode selection
+4.0	Automatic	BOOL	FALSE	1st operating mode of process unit
+4.1	Manual	BOOL	FALSE	2nd operating mode of process unit
+4.2	Select_Operating_Mode2	BOOL	FALSE	3rd operating mode of process unit
+4.3	Select_Operating_Mode3	BOOL	FALSE	4th operating mode of process unit
+4.4	Select_Operating_Mode4	BOOL	FALSE	5th operating mode of process unit
+4.5	Select_Operating_Mode5	BOOL	FALSE	6th operating mode of process unit
+4.6	Select_Operating_Mode6	BOOL	FALSE	7th operating mode of process unit
+4.7	Select_Operating_Mode7	BOOL	FALSE	8th operating mode of process unit
+5.0	Select_Operating_Mode8	BOOL	FALSE	9th operating mode of process unit

+5.1	Select_Operating_Mode9	BOOL	FALSE	10th operating mode of process unit
+5.2	Select_Operating_Mode10	BOOL	FALSE	11th operating mode of process unit
+5.3	Select_Operating_Mode11	BOOL	FALSE	12th operating mode of process unit
+5.4	Select_Operating_Mode12	BOOL	FALSE	13th operating mode of process unit
+5.5	Select_Operating_Mode13	BOOL	FALSE	14th operating mode of process unit
+5.6	Select_Operating_Mode14	BOOL	FALSE	15th operating mode of process unit
+5.7	Select_Operating_Mode15	BOOL	FALSE	16th operating mode of process unit
+6.0	Group_Error	BOOL	FALSE	TRUE = Unit failed
+6.1	Confirm_Units	BOOL	FALSE	TRUE: Unit is acknowledged. Set by display device (if configured) when unit is acknowledged by user. Bit must be reset by user program.
=8.0		END_STRUCT		

A.3 What is the UDT_S_Unit?

Introduction

The UDT_S_Unit contains the basic parameters necessary for a process unit and can be inserted in the variable declaration table of a function block:

- In the areas: “In,” “Out,” and “Stat”

An “Array of UDT_S_Unit” is not permitted.

The UDT_S_Unit is characterized by the fact that it has the attribute “S7_pdiag_s_unit” and this attribute is set to “TRUE.”

Using the UDT_S_Unit

The UDT_S_Unit represents a process unit in your user program. The UDT_S_Unit contains the group error address and the group error acknowledgement and therefore enables you to structure your user program without including an operating mode definition.

The UDT_S_Unit is is stored as “UDT3” in the supplied example.

Data Structure of the UDT_S_Unit

The following table shows the data structure of the UDT_S_Unit. You **cannot** modify this preset data structure.

Address	Variable	Data Type	Initial Value	Comment
0.0		STRUCT		
+0.0	Unit_Version	WORD	W#16#1	Version of UDT
+2.0	Group_Error	BOOL	FALSE	TRUE = Unit failed
+2.1	Confirm_Units	BOOL	FALSE	TRUE: Unit is acknowledged. Set by display device (if configured) when unit is acknowledged by user. Bit must be reset by user program.
=4.0		END_STRUCT		

A.4 What is the UDT_Motion?

Introduction

The UDT_Motion represents a motion in your user program. It can be inserted in the variable declaration table of a function block:

- In the areas: “In,” “Out,” and “Stat”

An “Array of UDT_Motion” is not permitted.

The UDT_Motion is characterized by the fact that it has the attribute “S7_pdiag_motion” and this attribute is set to “TRUE.”

Using the UDT_Motion

You should use the UDT_Motion each time you program a motion. The UDT_Motion is the data interface between the running user program, S7 PDIAG, and the display devices.

The UDT_Motion is stored as “UDT2” in the supplied example.

Data Structure of the UDT_Motion

The following table shows the data structure of the UDT_Motion. You **cannot** modify this preset data structure.

Address	Variable	Data Type	Initial Value	Comment
0.0		STRUCT		
+0.0	Unit_Version	WORD	B#16#1	Version number of motion structure.
+2.0	Data_Length	BYTE	B#16#0	Length of motion structure.
+3.0	Moving_Status1	BOOL	FALSE	TRUE (display device reads bit): Motion is moving in direction 1 (flashing square).
+3.1	Moving_Status2	BOOL	FALSE	TRUE (display device reads bit): Motion is moving in direction 2 (flashing square).
+3.2	Executability1	BOOL	FALSE	TRUE (display device reads bit): Motion can be moved in direction 1 when Interlock 1 is fulfilled (arrow filled).

+3.3	Executability2	BOOL	FALSE	TRUE (display device reads bit): Motion can be moved in direction 2 when Interlock 2 is fulfilled (arrow filled).
+3.4	Group_Error	BOOL	FALSE	TRUE (display device reads bit): A monitoring definition whose IDA is an element of an instance in this data structure detected an error.
+4.0	Number_of_Final_Positions	BYTE	B#16#0	Number of used final positions (display device reads byte): Display device shows the actual number of final positions contained in this motion.
+6.0	Final_Position	ARRAY (0 to 15)	FALSE	TRUE (display device reads bit field): One or more of the maximum number of final positions (16) have been reached (Final_Position_[0] is on the left).
*0.1		BOOL		
+8.0	Interlock1	BOOL	FALSE	TRUE (internal bit in PLC): Interlock conditions for motion in direction 1 are all fulfilled; motion can be moved.
+8.1	Interlock2	BOOL	FALSE	TRUE (internal bit in PLC): Interlock conditions for motion in direction 2 are all fulfilled; motion can be moved.
+8.2	Manual_Interlock1	BOOL	FALSE	TRUE (internal bit in PLC): Interlock conditions for motion in direction 1 on manual operation are all fulfilled; motion can be moved.

+8.3	Manual_Interlock2	BOOL	FALSE	TRUE (internal bit in PLC): Interlock conditions for motion in Direction 2 on manual operation are all fulfilled; motion can be moved.
+8.4	Manual_Enable1	BOOL	FALSE	TRUE (display device writes bit): Set when motion can be operated on the screen of the display device.
+8.5	Manual_Enable2	BOOL	FALSE	TRUE (display device writes bit): Set when motion can be operated on the screen of the display device.
+8.6	Manual_Operation1	BOOL	FALSE	TRUE (display device writes bit): Direction 1 key pressed on display device.
+8.7	Manual_Operation2	BOOL	FALSE	TRUE (display device writes bit): Direction 2 key pressed on display device.
+10.0	Display_Order	ARRAY (0 to 15)	FALSE	TRUE (display device writes bit field): Assignment showing which motion is displayed at which position on the screen of the display device and which direct keys are now valid (the top key corresponds to Display_Order[0]. Only 1 bit can be TRUE at any one time.
*0.1		BOOL		
+12.0	Trigger1	BOOL	FALSE	TRUE (internal bit in PLC): Trigger motion in direction 1.
+12.1	Trigger2	BOOL	FALSE	TRUE (internal bit in PLC): Trigger motion in direction 2.

+12.2	Automatic_Trigger1	BOOL	FALSE	TRUE (internal bit in PLC): Trigger motion on automatic operation in direction 1.
+12.3	Automatic_Trigger2	BOOL	FALSE	TRUE (internal bit in PLC): Trigger motion on automatic operation in direction 2.
+12.4	Control1	BOOL	FALSE	TRUE (internal bit in PLC): Trigger output of motion in direction 1.
+12.5	Control2	BOOL	FALSE	TRUE (internal bit in PLC): Trigger output of motion in direction 2.
+14.0	Position_Flag	ARRAY (0 to 15)	FALSE	(internal bit field in PLC): Position flag for edge detection for reaction monitoring. Position_Flag[0] is assigned to Final_Position[0].
*0.1		BOOL		
=16.0		END_STRUCT		

A.5 What Are the Ladder Networks for Motion Monitoring?

Introduction

The Ladder networks for motion monitoring supplied in FB100 of the sample project “S7_DIAG” create the whole motion programming, and are explained in detail in this section.

These networks also contain the program for manually operating the motion from the display device.

Data Definition in the Examples

In the examples, the following definition of the motion of a cylinder is used with the name “z” and two bits for automatic operation and manual operation of the machine.

The data structure looks like this:

z	UDT_Motion
automatic	BOOL
manual	BOOL

Note: The variables “automatic” and “manual” are typically taken from the corresponding UDT_Unit. For the sake of simplicity, this is not taken into account in the examples.

The following section shows you one direction of a motion complete with the instructions for using direct keys. In the second example, a shorter version of one direction of a motion is shown.

A.6 Complete Example for One Direction of a Motion Using Direct Keys

Introduction

The following example shows how the incoming and outgoing values of the motion structure UDT_Motion are provided. It is up to you which networks you use or modify; however, if you do change the given networks, this may restrict the functionality of the program on the display device.

Network 1

In network 1, the final position can be determined via a limit switch, a light sensor, or a combination of data. The status of the final positions is displayed in the motion screen. Here, the limit switch “Ia.b” is visualized.

Fig. A–1: Network 1: Displaying Final Position [0]

- **Startup monitoring:**
  Here the program monitors whether the current final position is actually left once the motion has been triggered.

  Monitoring logic (with time):
  ONDT (#z.Control2, ?) AND #z.Final_Position[0]
  (generally without initial value acquisition)

- **Action monitoring:**
  Here the program monitors whether the final position is actually reached once the motion has been triggered.

  Monitoring logic (with time):
  ONDT (#z.Trigger1, ?) AND NOT #z.Final_Position[0]
  (generally without initial value acquisition)

- **Reaction monitoring:**
  Here the program monitors whether the final position is left without the motion being triggered.

  Monitoring logic (without time):
  #z.Position_Flag[0] AND NOT #z.Final_Position[0]

  Monitoring logic (with time):
  #z.Position_Flag[0] AND NOT ONDT (#z.Final_Position[0], ?)

Network 2

In this network, the safety conditions (interlocks) of the motion are checked in direction 1. In the example, the opposite direction of the motion is used; in this case, the trigger for the output is negated.

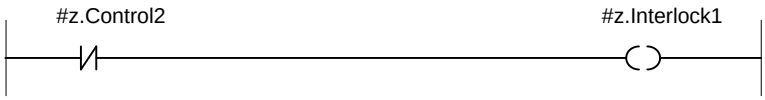


Fig. A–2: Network 2: Interlock for Direction 1

Network 3

In this network, the interlocks for manual operation of the motion are checked in direction 1.

This network may or may not be used, depending on whether there are differences in the interlock between the operating modes. Depending on the application used, network 1 can also include the interlocks for automatic operation and network 2 the interlocks for manual operation (programmed manual mode). In the example, the same interlocks apply as in network 1.

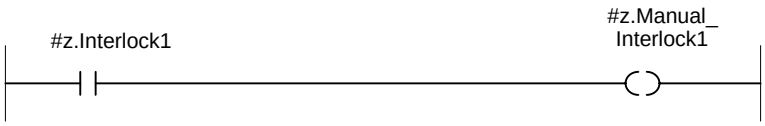


Fig. A–3: Network 3: Manual interlock for Direction 1

Network 4

In this network, the executability is formed. This supports programmed manual mode. The network shows that the motion can be moved in direction 1.

- **Interlock monitoring:**
  The interlock monitoring definition is appended to the “Executability” signal. This, together with criteria analysis, enables you to determine the missing signal which is preventing the motion from moving (either in manual operation or automatic operation).

  Monitoring logic (without time):
  #z.Trigger1 AND NOT #z.Executability1
  Monitoring logic (with time):
  ONDT (#z.Trigger1, ?) AND NOT #z.Executability1
  (generally with initial value acquisition)

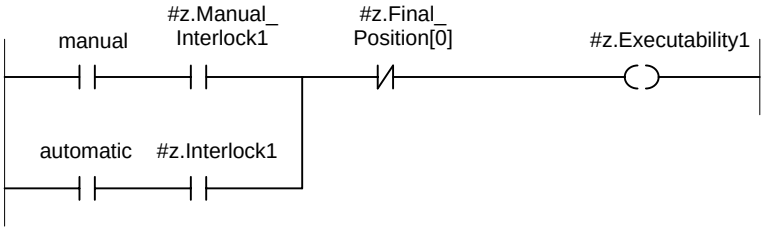


Fig. A–4: Network 4: Representing Executability for Direction 1 on the Display Device

Network 5

Here the trigger is formed which is to move the motion in direction 1.

The lowest branch in this network represents automatic operation. Due to the order in which the program is processed, the variable “#z.Automatic_Trigger1” is set at another location in the user program in order to move the motion.

The two upper branches in the network represent the operation of the motion using the direct keys via the motion screen. Here, “#z.Manual_Enable1” decides whether the motion is represented on the display device or not.

The pair “Ix1.y1” and “#z.Display_order[0]” would switch if the motion were first in the sequence and the direct key “Ix1.y1” were pressed. The same applies to all other direct keys. In this example, only two direct keys are used.

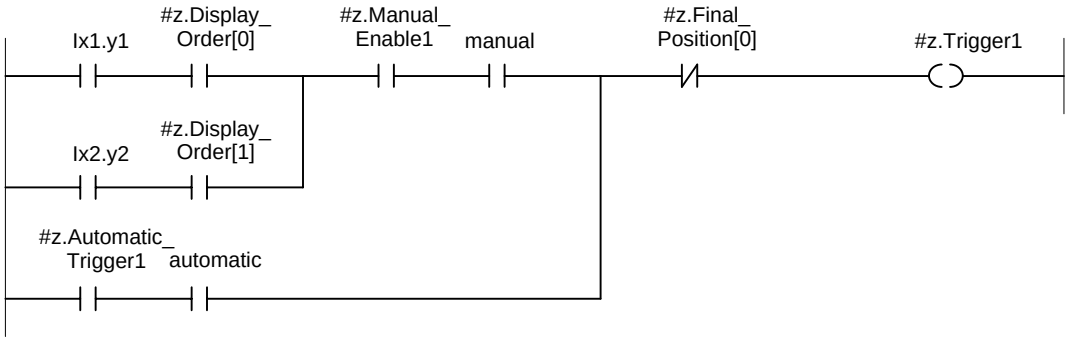


Fig. A–5: Network 5: Trigger for Direction 1

Network 6

This network is only required if you are using reaction monitoring.

The position flag is set when the motion has arrived at the final position and the trigger is still active. The trigger is then removed in the next network. As soon as the position flag is set, reaction monitoring is activated.

Note: Reaction monitoring is not activated until the motion has already been moved once, in order to avoid initialization problems.

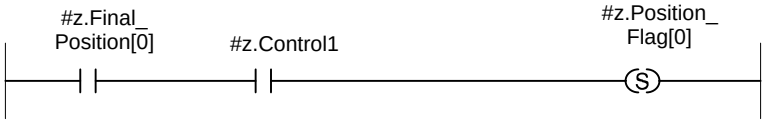


Fig. A–6: Network 6: Setting the Position Flag for Reaction Monitoring

Network 7

In this network, the trigger for the motion is formed for all operating modes.

In this example, the motion is only controlled if both the executability and the trigger for this direction are set.

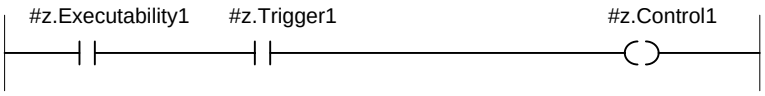


Fig. A–7: Network 7: Triggering the Motion for Direction 1

Network 8

The “Moving_Status” bit is used to represent whether the motion being triggered is actually moved on the display device.

This can be determined implicitly by triggering the output, as shown in the example, or by measuring the motion directly in the process.

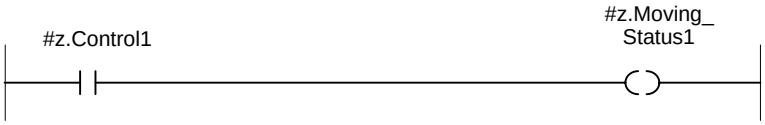


Fig. A–8: Network 8: Displaying the Motion Moving in Direction 1

Network 9

This network is only required if you are using reaction monitoring.

The position flag is reset when the motion is triggered in the opposite direction.

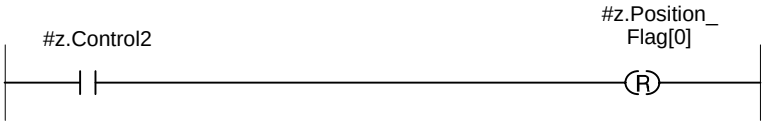


Fig. A–9: Network 9: Resetting the Position Flag

A.7 Shorter Example for One Direction of a Motion Without Using Direct Keys

Introduction

In the following example, one direction of a motion is shown in reduced form. This contains the display functions and the monitoring definitions, whilst excluding reaction monitoring

This motion cannot be moved via direct keys.

Network 1

The final position can be determined via a limit switch, a light sensor, or a combination of data. The status of the final positions is displayed in the motion screen.

In this example, the limit switch Ia.b is visualized.

- **Startup monitoring:**
  Here the program monitors whether the current final position is actually left once the motion has been triggered.

  Monitoring logic (with time):
  ONDT (#z.Control2, ?) AND #z.Final_Position[0]
  (generally without initial value acquisition)

- **Action monitoring:**
  Here the program monitors whether the final position is actually reached once the motion has been triggered.

  Monitoring logic (with time):
  ONDT (#z.Trigger1, ?) AND NOT #z.Final_Position[0]
  (generally without initial value acquisition)

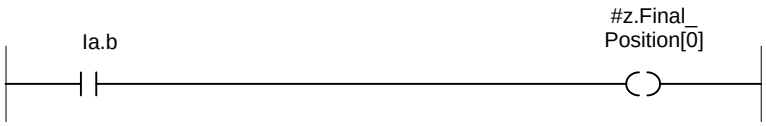


Fig. A–10: Network 1: Displaying Final Position [0]

Networks 2, 3 and 4

In these networks, the executability is formed. The networks contain the interlocks for both manual and automatic operation. This supports programmed manual mode, and the networks show that the motion can be moved in Direction 1.

You can add further interlock conditions in these networks.

- **Interlock monitoring:**
  The interlock monitoring definition is appended to the “Executability” signal. This, together with criteria analysis, enables you to determine the missing signal which is preventing the motion from moving.

  Monitoring logic (without time):
  #z.Trigger1 AND NOT #z.Executability1

  Monitoring logic (with time):
  ONDT (#z.Trigger1, ?) AND NOT #z.Executability1

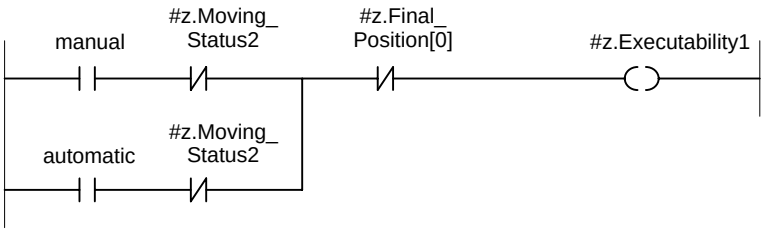


Fig. A–11: Networks 2/3/4: Displaying Executability for Direction 1 on the Display Device

Network 4

Here the trigger is formed to enable the motion to move in Direction 1.

The upper branch in the network represents the operation of the motion using the key via the motion screen. When the key is pressed, the display device sets the “Manual_Operation” bit.

The lowest branch in this network represents automatic operation. Due to the order in which the program is processed, the variable “#z.Automatic_Trigger1” is set at another location in the control program in order to move the motion.

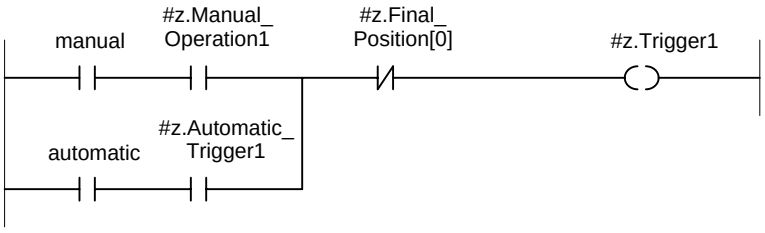


Fig. A–12 Network 4: Trigger for Direction 1

Networks 7 and 8

In these networks, the trigger for the motion is formed for all operating modes.

In this example, the motion is only controlled if both the executability and the trigger for this direction are set.

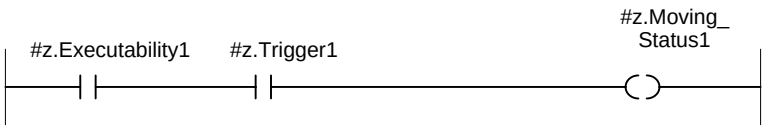


Fig. A–13 Networks 7 and 8: Triggering / Displaying the Motion Moving in Direction 1

A.8 The User Function Block as the Interface to Your User Program

Introduction

The user function block supplied with the S7 PDIAG software enables you to react to specific error messages and hence to error states in your process. It is called with different parameters, depending on whether the error is incoming or outgoing. In this way, you can obtain information on the unit, the priority, and the message number of the error, for example.

Interface of the User Function Block

The interface description for the user function block is shown below:

- FUNCTION_BLOCK FB “Message”

```
VERSION : 0.0

VAR_INPUT                 // Standardized interface
      EV_C: BOOL;         // Incoming message if TRUE
                          // Outgoing message if FALSE
      EV_ID : DWORD;      // Message number
      SD_1: ANY           // Process value V4.0: NIL
      PRIO: BYTE          // Priority of the error message
      EV_DB: WORD         // DB number of the unit sending the message
      USER_OPD: ANY       // Specific address V4.0: NIL
        ...
END_VAR                   // Standardized interface
VAR                       // Free parameters
      ...
      ...
END_VAR                   // Free parameters
VAR_TEMP                  // Free parameters
      ...
      ...
END_VAR                   // Free parameters
BEGIN
      ...                 // Any instructions for
                          // processing the input parameters
      ....
END_FUNCTION_BLOCK
```

A.9 What You Should Observe When Programming

Introduction

In order for S7 PDIAG to be able to support criteria analysis on the display device, you should observe the following notes.

Program-Oriented Work

S7 PDIAG enables program-oriented work; that is, it searches for and edits only those error definitions which you have configured in an entire program.

Using Auxiliary Networks

If a multiple address assignment is discovered while you are using auxiliary networks, replacement is aborted at this point.

Run Sequence

The run sequence of the individual networks is not taken into account when using auxiliary networks.

Example:
```
A M0.0
A I1.0
= M1.1          //M1.1 has the value of cycle n-1

A I1.1
= M 0.0         //M0.0 has the value of cycle n
```

When using the auxiliary network, the following results:

```
A I1.1
A I1.0
= M1.1          //Only cycle n is taken into account.
```

Enabling Blocks to Contain Diagnostic Data

It may also be a good idea to enable blocks where error definitions are not used to contain diagnostic data.

If the result of logic operation of an initial diagnostic address is formed from addresses which are not formed in the same block as the initial diagnostic address itself, these blocks with preceding logic operations must also be able to receive diagnostic data so that they can be included in criteria analysis.

A.10 Enabling Blocks to Contain Diagnostic Data

Introduction

In order to carry out process diagnostics, you must create the information required by S7 PDIAG for the selected block. For blocks which contain error definitions, this occurs automatically. Blocks which do not have their own error definition, but which contain preceding logic for the network to be monitored, can be given diagnostic capability as follows:

1. Activate the “Store Process Diagnostics Data” check box in the LAD/STL/FBD Editor, as described below.
2. Enable the block to which you want to append the error message to contain diagnostic data by following the instructions below.

Activating the Check Box

To activate the check box, proceed as follows:

1. Double-click the selected block in the SIMATIC Manager to open the LAD/STL/FBD Editor and select the menu command **Options > Customize**.
2. Select the “Create Block” tab in the dialog box which appears and activate the check box “Store Process Diagnostics Data.” Exit the dialog box with “OK.”

Enabling a Block to Contain Diagnostic Data

You have two means of enabling a block to contain diagnostic data:

- If you append an error definition to a block as described in Section 4.3, the block will automatically be able to receive diagnostic data.
- You can also enable a block to contain diagnostic data by assigning the following system attribute:

Attribute	Value	Assign this attribute if...	For Block
S7_pdiag	true	Information for S7 PDIAG is to be generated.	FB, FC, OB, and DB

To assign the above system attribute to a block, proceed as follows:

1. Select the menu command **File > Properties** while the block is open in the incremental LAD/STL/FBD Editor.

   Result: The dialog box containing the block properties is displayed.
2. Select the “System Attributes” tab and enter the attribute from the above table.
3. Exit the dialog box with “OK” and save the block in the Editor using the menu command **File > Save**.

   Result: You have now enabled your block to contain diagnostic data.

The Language Elements in S7 PDIAG and their Syntax

B

In This Chapter

This chapter describes the language elements available in S7 PDIAG for programming monitoring logic, and the syntax to be observed.

The chapter also informs you about the processing priorities of the individual qualifiers.

Chapter Overview

B.1 The Language Elements in S7 PDIAG

Introduction

Using the language elements in S7 PDIAG, you can program your own individual monitoring logic.

What are the Language Elements in S7 PDIAG?

The language used for programming in S7 PDIAG may contain any of the characters permitted in STEP 7 for describing addresses or timers, plus the following language elements:

- AND
- OR
- XOR
- NOT
- EN
- EP
- SRT
- ONDT
- Separators, brackets, addresses, and timers

The above language elements are described in detail in this section. You will also find examples of these elements in the online help.

AND

The AND instruction combines two logical expressions A1 and A2 to form another logical expression (A0) which can then be further combined. The result of the logic operation is TRUE when both input conditions are TRUE.

Example: I1.0 **AND** I1.1, or
MotorOn **AND** Enable, or
(MotorOn **AND** Enable) **AND** Automatic

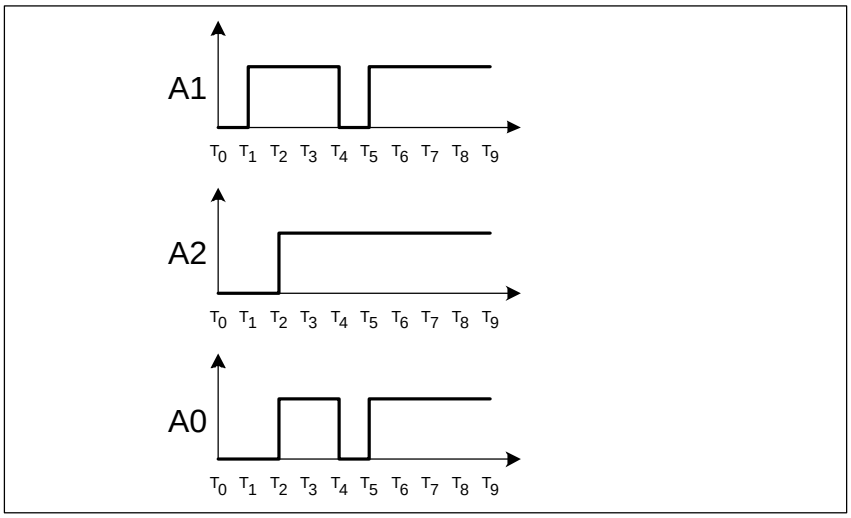


Figure B-1 Expression A1 and A2 at AND

OR

The OR instruction combines two expressions A1 and A2 to form a new expression A0 so that the expression A0 is TRUE when either A1 only or A2 only is TRUE or when both A1 and A2 are TRUE.

Example: I1.0 **OR** I1.1, or
MotorOn **OR** Enable, or
(MotorOn **OR** Enable) **OR** Automatic

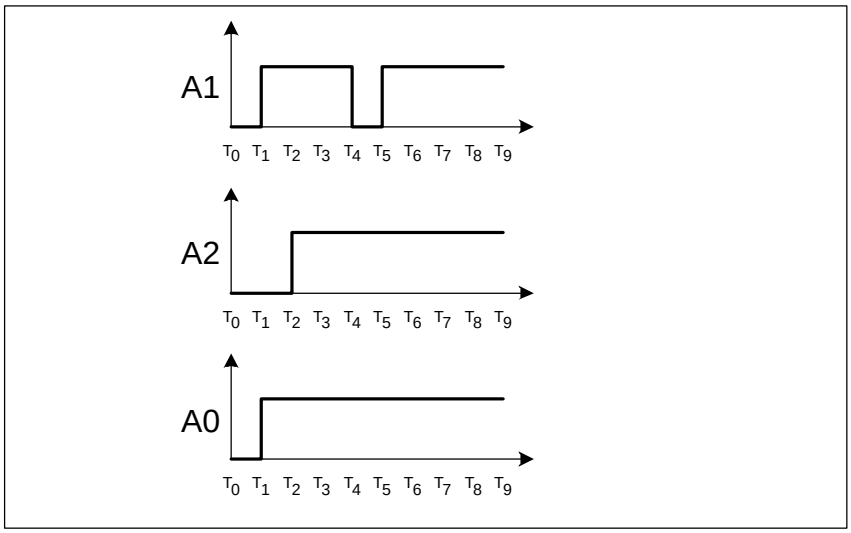


Figure B-2 Expression A1 and A2 at OR

XOR

The XOR instruction combines two logical expressions A1 and A2 to form a new expression A0 so that this expression is TRUE when either A1 only or A2 only is TRUE.

Example: I1.0 **XOR** I1.1, or
MotorOn **XOR** Enable, or
(MotorOn **XOR** Enable) **XOR** Automatic

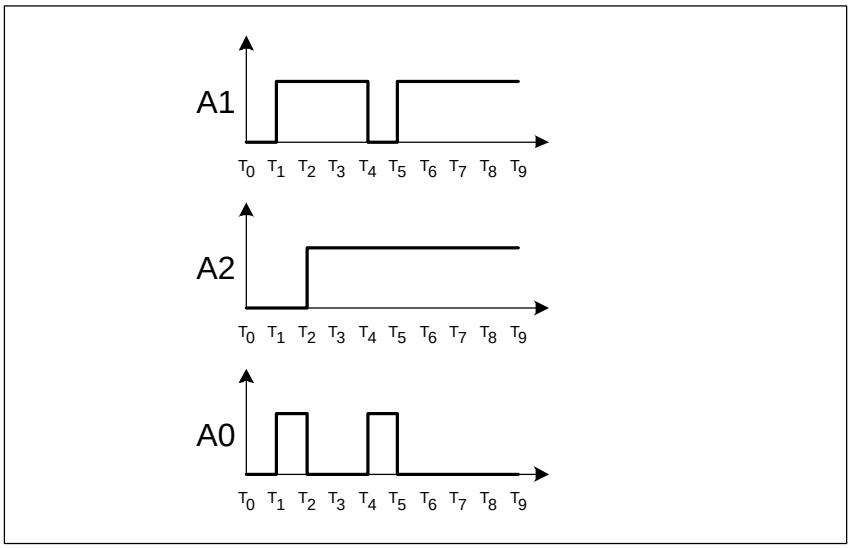


Figure B-3 Expression A1 and A2 at XOR

NOT

The NOT instruction forms the logical expression A0 by negating the expression A1. If A1 = TRUE, then A0 = FALSE. If A1 = FALSE, then A0 = TRUE. NOT therefore negates the logical result of the input expression.

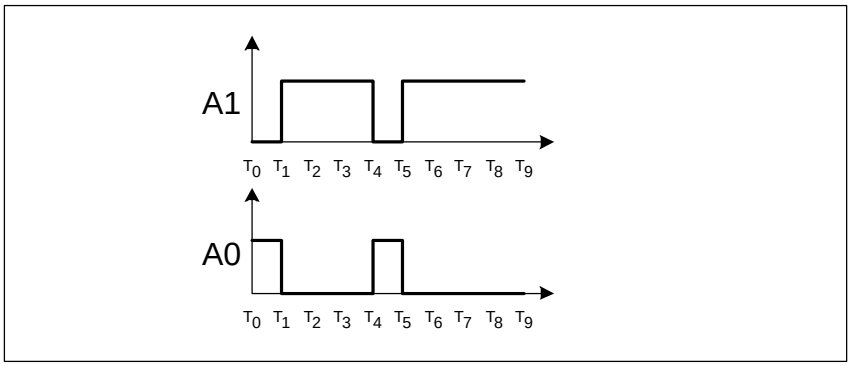


Figure B-4 Expression A1 at NOT

EN

General: A0 = EN (A1)
The EN instruction (falling edge) saves whether the last edge of the expression A1 was a rising or a falling edge. The EN instruction creates the expression A0 from the expression A1 according the following rules:

- A0 = TRUE after a falling edge (change from TRUE to FALSE) in A1.
- A0 = FALSE after a rising edge (change from FALSE to TRUE) in A1.
- A0 = FALSE until the occurrence of the first falling edge in A1.

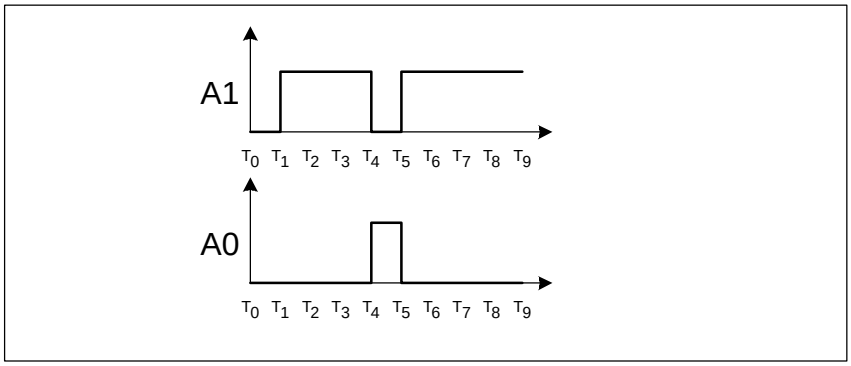


Figure B-5 Expression at EN

EN therefore evaluates whether the logic result of the input expression has caused a change from TRUE to FALSE. The result is TRUE when the change from TRUE to FALSE is recognized. The result remains TRUE until the input expression is TRUE again.

Note: The expression A1 should **not** contain the qualifiers ONDT, EN, EP, or SRT.

EP

General: A0 = EP (A1)
The EP instruction is the opposite of EN. This instruction saves whether the last edge of the expression A1 was a rising or a falling edge. The EP instruction creates the expression A0 from the expression A1 according the following rules:

- A0 = TRUE after a rising edge (change from FALSE to TRUE) in A1.
- A0 = FALSE after a falling edge (change from TRUE to FALSE) in A1.
- A0 = FALSE until the occurrence of the first rising edge in A1.

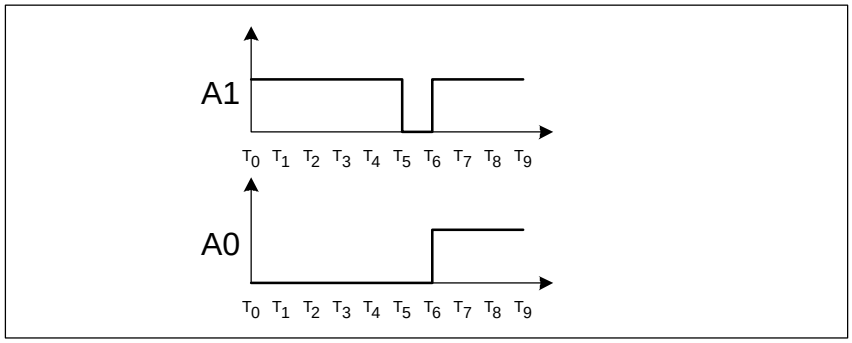


Figure B-6 Expression A1 at EP

With the EP instruction, the logic result is TRUE when the change in the input expression from FALSE to TRUE is recognized.

Note: The expression A1 should **not** contain the qualifiers ONDT, EN, EP, or SRT.

SR

General: A0 = SRT (A1, A2, T)
SRT (set / reset timer) is an on delay and monitors two expressions together. SRT works with both a set input and a reset input, both of which are pulse-driven. This means that the result is triggered by a rising edge (pulse) and not by TRUE or FALSE.

- A rising edge and therefore a positive change (from FALSE to TRUE) in A1 (set input) starts the time T, regardless of whether the timer is running. A0 is set to FALSE.
- A rising edge and therefore a positive change (from FALSE to TRUE) in A2 (reset input) stops the time T, regardless of whether the timer is running. A0 is set to FALSE. If the delay time expires before it is stopped by expression A2, the error is registered as “incoming.”
- A0 is set to TRUE if the time T has expired.
- A0 is initialized with FALSE on program start.
- If a rising edge occurs in both A1 (set input) and A2 (reset input) at the same time, A1 is ignored, since the expression A2 has higher priority, and the result is FALSE.

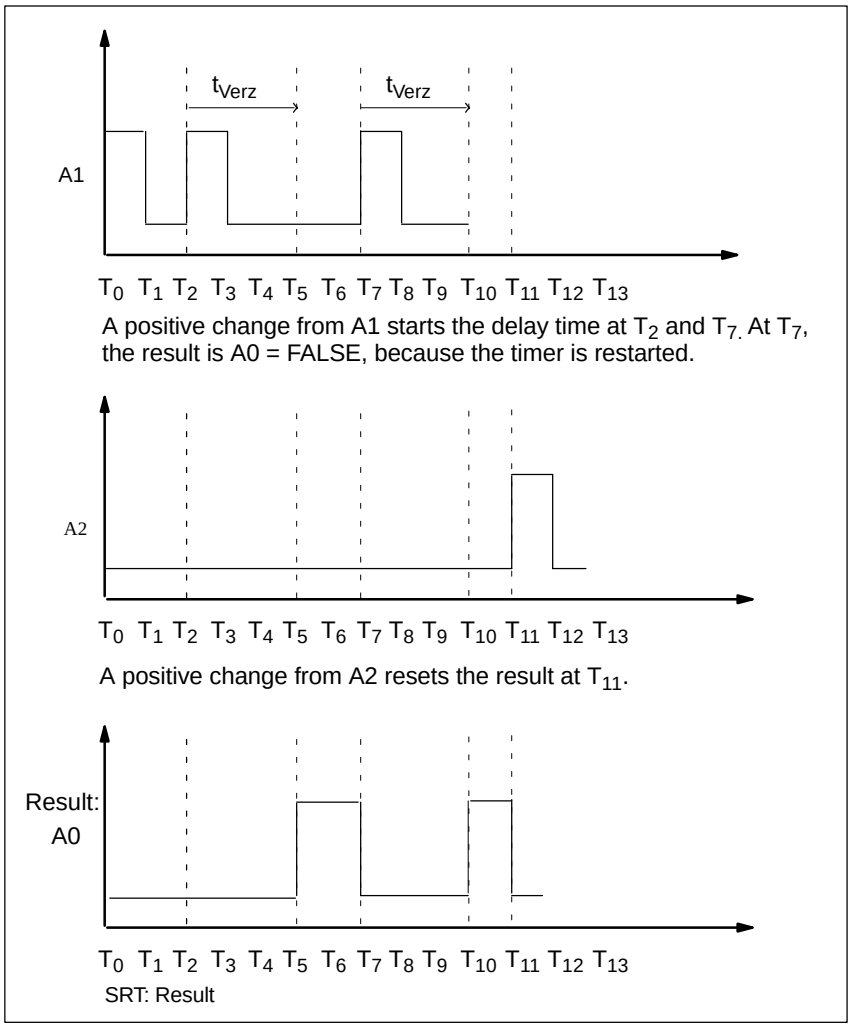


Figure B-7 Expressions A1 and A2 at SRT

Note: The expressions A1 and A2 should **not** contain the qualifiers ONDT, EN, EP, or SRT.

ONDT

General: A0 = ONDT (A1, T)
The ONDT instruction executes an on delay. Depending on the expression A1 and the time T, the ONDT instruction forms the expression A0 according to the following rules:

- If (A1 = FALSE) or (T running), then A0 = FALSE.
- If (A1 = TRUE) and (T expired), then A0 = TRUE.
- If A1 changes from FALSE to TRUE (rising edge), the timer starts again, (regardless of whether it is already running).
- If (A1 = TRUE), the timer is started (program start or complete restart).

The ONDT instruction always restarts the delay time if the result of logic operation of the expression (A1) is TRUE, see Figure LEERER MERKER:

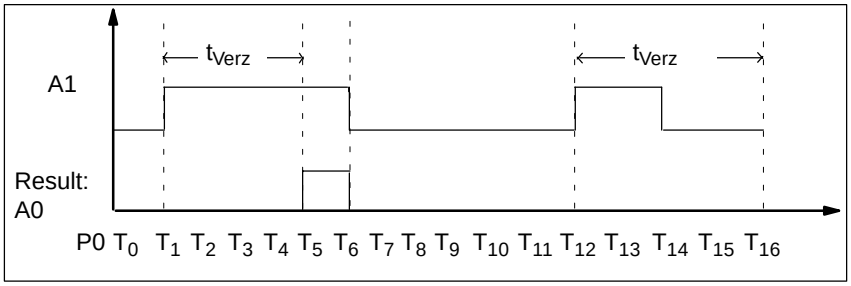


Figure B-8 On Delay with ONDT

If the expression A1 is still TRUE after the time (T) has expired, the result of ONDT is also TRUE.

Example: ONDT (I1.0, 2000 ms)

If the input I1.0 is TRUE, the result of this expression is TRUE after 2000 ms if the input remains TRUE. Here, the ONDT instruction corresponds to level monitoring at a positive level which can be further combined.

Note: The expression A1 should **not** contain the qualifiers ONDT, EN, EP, or SRT.

Separators

You must separate the individual language elements in S7 PDIAG from one another using separators.

The following characters are interpreted as separators:

- Space
- TAB
- ENTER
- Brackets (,)

Brackets

You can use brackets (or parentheses) to determine the order in which individual language elements are processed. Brackets can also be used as separators.

Addresses

You can use all binary S7 addresses in S7 PDIAG.

Timers

You can enter a time “t” in milliseconds, seconds, or hours, and in S7 time format. If you do not specify the unit of measurement, milliseconds is used as the default unit. Negative times and times = 0 are excluded.

Checking with S7 PDIAG

S7 PDIAG carries out the following checks:

1. Language element check.
2. Address range check.

You should only use language elements which belong the programming language used with S7 PDIAG, and you should arrange these language elements using the correct syntax. Otherwise, an error will be displayed both when you enter the language elements and when you try to compile the monitoring logic.

What is an Expression in S7 PDIAG?

An expression is a logic operation with binary addresses whose result then represents a binary result which can be interconnected. An expression in S7 PDIAG can consist of:

- A single address, for example **I1.0**
- Addresses which are combined with other addresses

  for example **I1.0 AND I1.1**, or even
- Addresses which are combined with other addresses and set in brackets, for example **(I1.0 AND I1.1) OR (I1.2 XOR I1.3)**

B.2 The Syntax of the Language Elements in S7 PDIAG

Introduction

The syntax of the programming language describes the relationship between the different language elements. The syntax for addresses and timers corresponds to the standard STEP 7 syntax.

This section illustrates the valid syntax for the following:

- Expressions
- Boolean expressions

The words in bold text are metawords in the language.

Expressions are always binary and can have the value TRUE or FALSE.

Syntax for Expressions

Expression: = Address

or

(Expression)

or

NOT Expression

or

Expression **OR** Expression

or

Expression **AND** Expression

or

Expression **XOR** Expression

or

EP (Boolean Expression)

or

EN (Boolean Expression)

or

ONDT (Boolean Expression, Time) o*r*
SRT (Boolean Expression, Time)

Syntax for Boolean Expressions

Boolean Expression:

Address

or

(Boolean Expression)

or

NOT Boolean Expression

or

Boolean Expression **OR** Boolean Expression

or

Boolean Expression **AND** Boolean Expression

or

Boolean Expression **XOR** Boolean Expression

B.3 The Processing Priorities of the Individual Qualifiers

Introduction

The individual qualifiers within S7 PDIAG are processed with the following priority:

Qualifier	Priority
Brackets (,)	1
NOT, EP, and EN	2
AND	3
XOR	4
OR	5
ONDT and SRT	6

With qualifiers which have the same priority, the monitoring logic is processed from left to right.

Example of Using Monitoring Types with S7 PDIAG

C

In This Chapter

This chapter uses an example of a drill to show how you can work with motion monitoring in S7 PDIAG with the help of a function block containing both the data interface for the display device and the monitoring definitions for S7 PDIAG.

To ensure that the “drill” example can be programmed and debugged as described, you require the following hardware and software components:

- Programming device / PC with the STEP 7 Standard package and S7 PDIAG optional package
- Multipoint interface connection to an S7-300 or S7-400 programmable logic controller with 16 digital inputs and 8 digital outputs, or, alternatively, with the S7 PLCSIM V4.x optional package.
- In order to familiarize yourself with the complete range of functions, you also require a display device with the process diagnostics package ProAgent.

Chapter Overview

C.1 Purpose of this Example and Tasks Involved

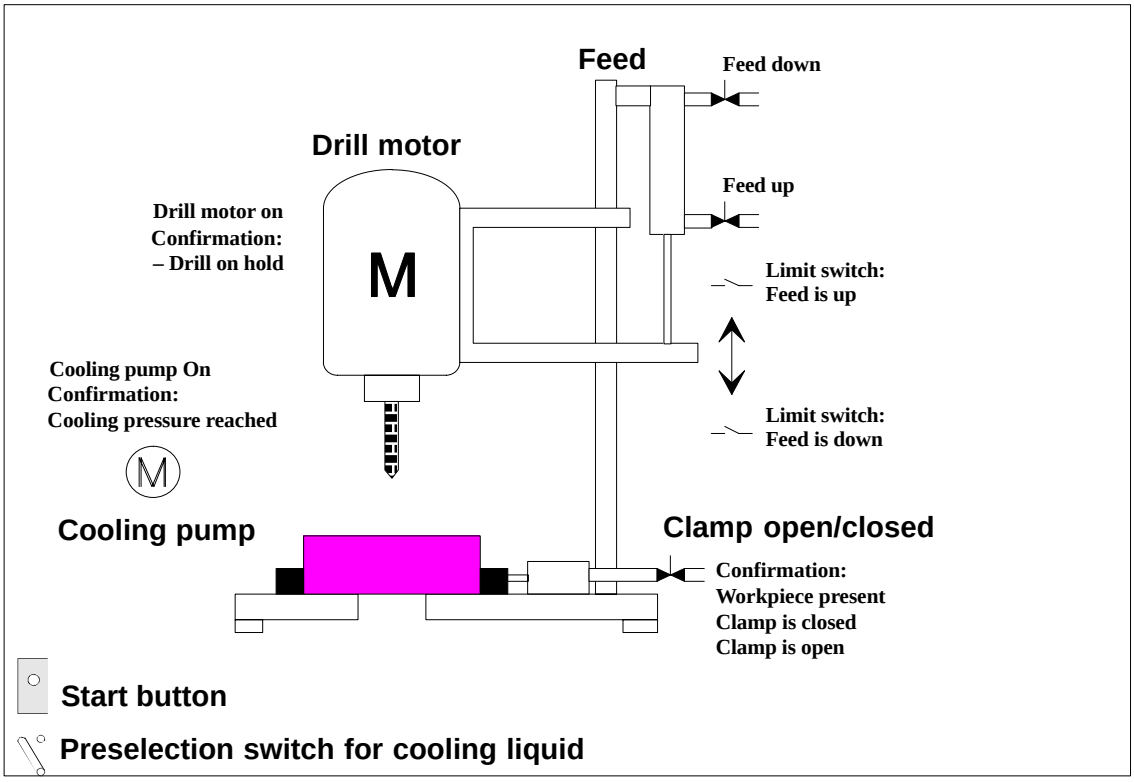
Task Definition

A monitoring definition is to be programmed to automate a drill. The structure of the drill is predefined in a diagram and the process is defined in the form of a function chart.

Structure of the Drill

The drill consists of the following elements:

- **Drill motor** with confirmations for “drill on hold”
- **Start button** and **preselection switch for cooling liquid**
- **Cooling pump** with confirmation for “cooling pressure reached”
- **Clamping device** open/closed with confirmation for clamped/released and “workpiece present”
- **Feed** drill up/down with limit switch for feed up/down



Initial State

The initial state of the drill is defined as follows:

- Drill motor and cooling pump are on hold
- Feed/drill is in “is up” position
- No workpiece has been inserted or clamped.

C.2 Function Chart and Units for the Structure of the Drilling Process

Introduction

The drilling process is divided into the following steps:

- Insert workpiece (manually)
- Press preselection switch for cooling liquid, if required (dependent on material)
- Start machine using start button
- Clamp workpiece
- Switch on cooling pump (depending on preselection)
- Lower drill using feed to lower target position (drill)
- Raise drill using feed to upper target position
- Release workpiece, switch off drill motor and cooling pump
- Remove workpiece (manually).

Function Chart

Figure C-1 below shows the corresponding function chart for the structure of the drilling process.

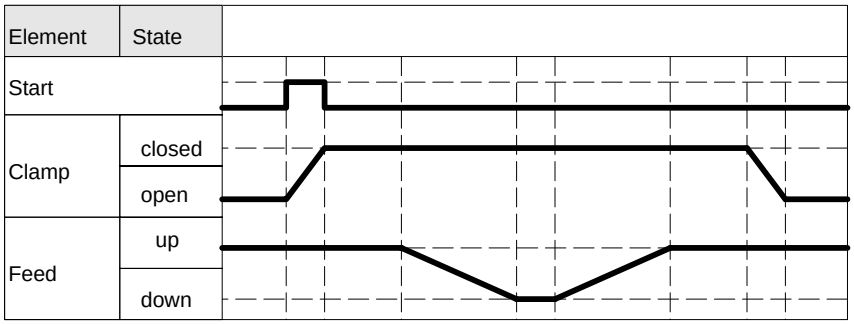


Figure C-1 Function Chart for the Drill

Determining Units

In this example there is exactly one functional unit, the drill itself.

As the drill does not have an operating mode selection switch, the UDT_S_Unit is used for this functional unit.

Determining Motions

As shown in the diagram, there are two motions:

- Feed
- Clamp

Defining the Inputs and Outputs

The relevant inputs and outputs for the drill are listed in the following table:

Address		Comment
Absolute	**Symbolic**	
Inputs in the Program (I)		
I 0.1	Drill_on_Hold	Confirmation: Drill is not active
I 1.1	Is_Open	Confirmation: Workpiece released from clamp
I 0.3	Is_Up	Limit switch “clamp in upper position”
I 0.2	Is_Down	Limit switch “feed in lower position”
I 1.2	Is_Closed	Confirmation: Workpiece clamped
I 0.6	Cooling_Pressure_OK	Confirmation: Cooling pressure reached
I 0.7	Start_Button	Start button of drill
I 0.5	Use_Cooling_Liquid	Selection switch for cooling liquid
I 1.0	Workpiece_Present	Confirmation: Workpiece in clamp
Outputs in the Program (Q)		
Q 0.0	Drill_Motor_On	Switch on drill motor
Q 0.1	Cooling_Pump_On	Switch on cooling pump (dependent on workpiece)
Q 0.5	Release_Clamp	Release workpiece from clamp
Q 0.4	Close_Clamp	Hold workpiece in clamp
Q 0.3	Feed_Up	Raise drill using feed to upper final position
Q 0.2	Feed_Down	Lower drill using feed to lower final position

C.3 Program Structure of the Drill

Overview of Blocks

The following table provides an overview of the blocks used in the user program and their functions:

Block	Description	Function
FB 1	Control drill	Control process of the drill
DB 1	Drill	Data (= process unit) of the drill
FB 100	Motion_to_Limit_Switch	Template FB for motion with two final positions
OB 1	PLC cycle	Cyclic processing of the user program
OB 100	PLC startup	Starts up the user program
FB 44	Error_Detection FB	S7 PDIAG: Error detection
DB 44	Error_Detection DB	S7 PDIAG: Error detection
FB 45	Initial_Val_Acquisition FB	S7 PDIAG: Initial value acquisition
DB 45	Initial_Val_Acquisition DB	S7 PDIAG: Initial value acquisition
UDT 1	UDT_Unit	Data structure of a unit with operating modes
UDT 2	UDT_Motion	Data structure of a motion
UDT 3	UDT_S_Unit	Data structure of a unit without operating modes

Template for Motion FB100

This block is a template for a motion function block and contains both:

- The data interface between the display device and the user program, and
- The monitoring definitions (error definitions) for the motion.

The block simply needs to be interconnected in the actual user program for the drill. This makes motions much easier to handle.

The program comment for this block provides a detailed description of the individual networks, monitoring definitions, and their function.

Using FB100

The following figure shows the call interface of the motion FB100:

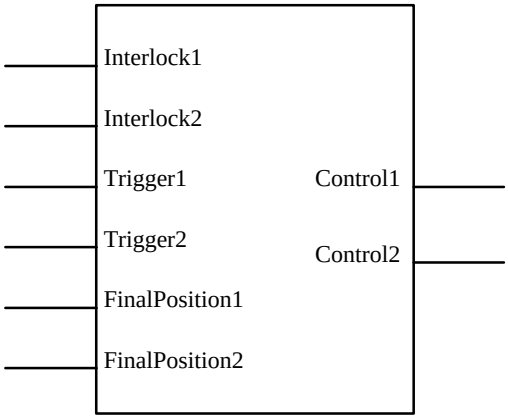


Figure C-2 Call Interface of the FB100

Using the Parameters

The following table shows how the parameters are used. The designation 1 or 2 indicates the respective direction of the motion:

- 1= on the left-hand side of the display device
- 2= on the rigt-hand side of the display device

Interlock1/2	Interlocks which, if not fulfilled, prevent the motion from moving in the respective direction. Note: The different operating modes should be taken into consideration. You can also insert the corresponding different network branches and interconnect them to this input.
Trigger1/2	Trigger which ensures that the motion can be moved in the required direction. Note: The logic for moving motions manually via the motion screens on the display device is already stored in the motion block. This means that you only need to interconnect the trigger here, for example, for automatic operation.
FinalPosition1/2	Input showing that the motion has reached or left the respective final position. Note: If you activate the option “Preset Final Positions” when compiling with S7 PDIAG, the designations of the final positions for the motion screen on the display devices are entered automatically.
Control1/2	Triggers the outputs for the respective direction of the motion.

Monitoring Definitions in FB100

In our example, the following monitoring definitions have been configured in the motion FB100:

- **Interlock monitoring definitions** (Motion should move, but is not allowed). With these monitoring definitions, initial value acquisition and therefore criteria analysis are activated. If an interlock error occurs, the criteria analysis on the display device shows which signals are missing in order for the motion to move. At the same time, the programming logic with which the interlock inputs of this function block are interconnected is analyzed.
- **Action monitoring definitions** (Motion is moving, but does not reach the target final position within a specified time).
  Initial value acquisition is not activated for this monitoring definition, as only the time taken to reach the target final position is being monitored. A default time of 1 minute has been configured here. If the requirements cease to apply while the motion is in progress, and the motion is allowed to continue, interlock monitoring is triggered.

Controlling the Drill in FB1

FB1 contains the control program for the drill.

The unit “drill” is defined in the code section for the data area of this function block. As the unit “drill” does not recognize different operating modes, the UDT_S_Unit is used as an indicator. You can use the group error bit in this UDT to determine whether or not there is an error in this unit.

The individual motions are then specified as a multiple instance for this unit. This means that you do not require a separate data block for each individual motion; instead, all the data for the motions of this unit can be combined in one block.

The functional process of the drill is controlled in the program section. The motion function block is used twice for this purpose. S7 PDIAG recognizes that it has been used twice and creates the relevant monitoring definitions for each motion. The function block for the drill has no parameters of its own and is called in the user program cycle (OB1).

Drill DB1

This data block contains all the data required to control the drill.

When you call FB1 in OB1 to control the drill, you are automatically asked whether the corresponding instance data block has been generated. The symbolic name of this data block is accepted by the display devices as the name of the unit.

S7 PDIAG Monitoring Blocks

The monitoring blocks FB 44/45 and DB 44/45 are generated by S7 PDIAG. They contain both the logic for error detection and the information required for initial value acquisition. All you have to do is call the error-detection FB44 with its corresponding instance data block DB44 at the end of the user program cycle (OB1).

C.4 Working with the Example

Preparations

In order for the example to run correctly, you require input bytes 0 and 1 and output 0 in your controller. First switch the controller to STOP.

- If you are working without a display device, you can display the messages generated by S7 PDIAG on your programming device/PC using the menu command **PLC > CPU Messages**. Place a check mark in column “A” and select “Top.”

  In this case, however, you cannot display the unit overview, the motion screen, and the criteria analysis. Instead, you can open FB1 and monitor it online.
- If you are working with a display device, create a new project and insert your CPU and the relevant display device.
- Double-click the network symbol to check that both the CPU and the display device are connected to a common network. Then copy the sample program into the program below your CPU.
- Download the sample program to the CPU and, if necessary, to the configuration for the display device.

Procedure

The error messages that occur will guide you through the example.

You will see how S7 PDIAG and ProAgent help you to “drive” the drill and how they offer you support when errors occur.

Now select the message screen on the display device, or activate the function “CPU Messages.”

Making the Initial Settings

First switch all inputs to “0” and then switch the CPU to “RUN.”

- The following message is displayed:

  “Interlock clamp open.”

This means that the clamp is not open, the motion should move in the direction “Open,” but is not permitted to do so.

Press the “Criteria Analysis” button on the display device or monitor the network for the motion “Clamp.”

- Criteria analysis displays the following result**:**

  A I0.1 Drill_on_Hold Confirmation: Drill is not active

This is the missing signal that the clamp requires in order to be able to move in the direction “Open.” Now switch the input I0.1 “Drill_on_Hold” to “1.”

You will see that the error message disappears, the output Q0.5 “Release_Clamp” is triggered, and the clamp moves in the direction “Open.”

- After a minute the following message is displayed:

  “Final Position Clamp open not reached.”

This means that the final position showing the open clamp was not reached.

You will see that the trigger for the motion is flashing in the motion screen, but the final position has not yet been reached. Now switch input I1.1 “Is_Open” to “1.” The message disappears and the motion screen displays the final position reached.

The initial settings of the drill are now complete.

Sequence of the Drilling Process

In this way you can work through the whole drilling process.

Switch input I0.7 “Start_Button” to “On.”

- The following message is displayed:

  “Interlock clamp close.”

- Criteria analysis displays the following result:

  A I1.0 Workpiece_Present Confirmation: Workpiece in clamp

As there is no workpiece in the clamping device, clamping cannot take place. Simulate the insertion of a workpiece by activating input I1.0 “Workpiece_Present.”

The clamp now closes while ouput Q0.4 “Close_Clamp” is triggered. At the same time, the drill is switched on, Q0.0 “Drill_Motor_On.”

- When the specified minute has elapsed, the following error message is displayed:

  “Final Position Clamp close not reached.”

First leave final position I1.1 “Is_Open” and then set final position I1.2 “Is_Closed.”

- The following message is now displayed:

  “Interlock Feed down.”

- Criteria analysis displays the following result:

  AN I0.1 Drill_on_Hold Confirmation: Drill is not active

Now simulate the drill motor turning by resetting input I0.1 “Drill_on_Hold.” You must note that because of the “AN” instruction, negative logic is displayed.

The feed now proceeds downwards and drilling takes place. This can be seen at the set output Q0.2 “Feed_Down.”

- When the specified monitoring time of one minute has elapsed, the following message is displayed:

  “Final Position Feed down not reached.”

First leave the final position I0.3 “Is_Up” and then set the final position I0.2 “Is_Down.”

Now the feed travels upwards again. This can be observed at the set output Q0.3 “Feed_Up.”

- When the configured monitoring time of one minute has elapsed, the following message is displayed:

  “Final position Feed up not reached.”

First leave the final position I0.2 “Is_Down” and then set the final position I0.3 “Is_Up.”

All that remains is for the clamp to be opened again.

- However, first the following message appears:

  “Interlock clamp open.”

- Criteria analysis for this displays the following result:

  A I0.1 Drill_on_Hold Confirmation: Drill is not active

Now acknowledge this message again.

Output Q0.5 “Release_Clamp” is now triggered.

- When the configured monitoring time of one minute has elapsed, the following error message is displayed:

  “Final position Clamp open not reached.”

First leave the final position I1.2 “Is_Closed” and then set the final position I1.1 “Is_Open.”

The drilling process is now complete and you can remove the workpiece by resetting input I1.0 “Workpiece_Present.” You can see in the motion screen that the left-hand triangle of the “Clamp” motion is now empty. This shows that this motion can now no longer be moved. As soon as you insert a workpiece, the clamp can be used again.

Other Drilling Processes

You can now work through the same process and include, for example, the use of the cooling liquid.

Summary

With a relatively simple configuration, S7 PDIAG and ProAgent provide both automatically generated motion screens and detailed information on any errors that occur. This enables you to reduce the down times of your automation solution and contributes to increased productivity.

Tips and Tricks for Working with S7 PDIAG D

In This Chapter

The following table contains information which will help you when working with S7 PDIAG.

Chapter Overview

Section	Description	Page
D.1	Support for Working with S7 PDIAG	D-2

D.1 Support for Working with S7 PDIAG

Introduction

The following notes may be of use when you are working with S7 PDIAG.

Renaming Instance Data Blocks

When you copy or rename an instance data block containing diagnostic data, the diagnostic data are deleted.

Remedy:
If necessary, remove the monitoring disable functions at the corresponding instance DB (**Options > Find... menu command**) and use the **Process Diagnostics > Compile** menu command to generate the instance-specific error definitions again in S7 PDIAG.

Which Addresses are Monitored?

You are searching for the error definitions or the monitored addresses for a particular block.

Remedy:
Open the block in the LAD/STL/FBD Editor and select any assignment. Then call up the “Initial Diagnostic Address” list using the menu command **Edit > Special Object Properties > Monitoring**. The list displays details of all the addresses which are monitored in this block.

Where are the Error Definitions?

You have created an error definition in the LAD/STL/FBD Editor but it does not appear in S7 PDIAG.

Remedy:
Save the block in the editor; the error definitions are now displayed in S7 PDIAG.

Where is the Relevant Instance Data Block?

You have added your error definition to a function block, but the instance data blocks are not visible in the unit overview.

Remedy:
Generate the relevant instance data block again. This instance data block is then designated as relevant for diagnostic purposes and is visible in S7 PDIAG.

Why is there no Message Number?

There is no message number in the function block, as you are only defining the template for your error definition in the function block.

Remedy:
Generate the instances in S7 PDIAG and a message number will be assigned.

Renaming User-Defined Data Types

When you copy user-defined data types (UDTs) from the supplied FB100 and add them to your user program and modify the name of a motion in the function block using the LAD/STL/FBD Editor, this name is not automatically changed in S7 PDIAG.

Remedy:
Open the data block in the LAD/STL/FBD Editor and save it again.

Diagnostic Data are not Accepted

Although you have set the attribute S7_pdiag = TRUE, the diagnostic data for this block are not accepted.

Remedy:
Open the block in the LAD/STL/FBD Editor and save it again.

Assigning Block Numbers

You must make sure that you do not assign the same block numbers in S7 PDIAG as you have in your user program. Otherwise the S7 PDIAG blocks are generated and will overwrite the blocks in your user program.

Remedy:
Before generating the blocks with S7 PDIAG, check that there are no double assignments of block numbers.

No Criteria Analysis Possible

You are unable to carry out criteria analysis, although it should be possible because the requirements have been met. The requirements are that:

- Blocks containing assignments possess diagnostic capabilities, and
- The initial value acquisition of the monitoring definition is activated.

Remedy:
Check that you have activated the check box for storing the process diagnostics data in the LAD/STL/FBD Editor. You can activate this option by placing a check mark in the box.

S7 PDIAG Templates with STEP 7 V3.2

If you are using S7 PDIAG in conjunction with STEP 7 V3.2, you can only create a template on the basis of a function block (the type of an error definition).

In this configuration, it is not possible to base the template on a data block (the instance of an error definition), nor can you use error definitions in functions, organization blocks, or the block container.

The Message Number Changes when a Block is Copied

You should note that when a block is copied within a program, the message number is changed to ensure the message numbers remain unique.

Limitations in Case of a Multiple Assignment on an Address

In the case of multiple assignments to an address which is used as an initial diagnostic address, a criteria analysis is in principle possible.

In the case of multiple assignments to an address which is used as an initial diagnostic address, a criteria analysis is in principle **not** possible.

Example 1: Address is used as an IDA:

```
Network1:
        U       M0.0
        =       M3.0        (IDA)
Network 2:
        U       M1.0
        =       M3.0        (IDA)
```

Result: In case of a fault both networks are displayed on the display device.

Example 2: Address is not used as an IDA:

```
Network 1:
        U       M0.0
        =       M2.0
Network 2:
        U       M1.0
        =       M2.0
Network 3:
        U       M2.0
        =       M3.0 (IDA)
```

Result: In case of an error the address M2.0 is displayed on the display device as the causing factor. However, an extended criteria analysis on the address M2.0 is not possible, since it has been assigned both in Network 1 and in Network 2.

Copying Projects and Blocks

Observe the following points if you want to copy projects or blocks with S7 PDIAG information in the SIMATIC Manager :

- When you have created groups and copy the corresponding blocks, the group information is not copied as well. The group information is not copied either when you select all the blocks in the “Blocks” directory and then copy these.
- The group information is only contained in the copy if you select and copy the “Blocks” directory or the higher-level directories.
- Global error definitions which were created directly at the “Blocks” directory in S7 PDIAG are only copied if the “Blocks” directory or the higher-level directories are copied in the SIMATIC Manager.

Searching For and Editing Error Definitions

If you use the **Options > Find** menu command in S7 PDIAG to search for instance error definitions or other error definitions, the respective message numbers for the found objects are displayed.

You can sort these by double-clicking on the header. This facilitates the assignment to the messages at the display device (HMI).

If you select an instance error definition, you can click with the right-hand mouse button to call up the ”Monitoring Type” pop-up menu and change, for example the message texts, directly in the corresponding type of the instance error definition.

Glossary

Absolute Address

An absolute address includes the address identifier and the physical memory location where the address is stored. Examples: Input I 12.1; Memory Word MW25; Data Block DB3.

Action Monitoring

Action monitoring is a type of motion monitoring which monitors whether a motion is completed within a specific action time. This is the case when the target final position is reached.

Action Time

The action time is the time within which a motion must be completed.

Actual Final Position

The actual final position is the final position in which the motion is currently located. It is defined as “Final_Position(n)” when using the UDT_Motion.

Address

An address is part of a STEP 7 statement and specifies what the processor should execute the instruction on. Addresses can be absolute or symbolic.

Address Monitoring

Address monitoring allows you to monitor specific individual addresses for level or edge changes; this type of monitoring can be combined with a time delay. Address monitoring is directly linked to an address, called the initial diagnostic address.

Auxiliary Process Value

An auxiliary process value is a value (or address) which can be “added” to a message text. This value is acquired by the S7 PDIAG at the point at which the error is also recognized. The auxiliary process value is displayed by the display system at the point in the message text which you have specified. To do so insert the corresponding formal address into the message text.

This auxiliary process value can be a parameter of the type BOOL, BYTE, WORD or DWORD from the areas I, Q, M or DB.

You can specify the position and the display format of the auxiliary process value in the message text. To do so, create a describing block for the auxiliary process value which starts with the character “@1X” and ends with “@”. The auxiliary process value is inserted into the message text instead of this describing block.

Bit Memory (M)

A memory area in the system memory of a SIMATIC S7 CPU. This area can be accessed using write or read access (bit, byte, word, and double word). The bit memory area can be used by the user to store interim results.

Block

Blocks are part of the user program and can be distinguished by their function, their structure, or their purpose. STEP 7 provides the following types of blocks:

- Logic blocks (FB, FC, OB, SFB, SFC)
- Data blocks (DB, SDB)
- User-defined data types (UDT)

Central Processing Unit (CPU)

The CPU is the central module in the programmable controller in which the user program is stored and processed. It consists of an operating system, processing unit, and communication interfaces.

Compiling

This process creates an executable user program from a source file.

Criteria Analysis

Using criteria analysis, you can analyze error conditions. You can then display the states of the addresses (initial values) which caused the error (for example, limit switch at input I1.1) up to the result of logic operation in STL, LAD, and FBD directly on the display device.

In order to be able to carry out criteria analysis on the display device, you must activate initial value acquisition in S7 PDIAG.

Data Block (DB)

Data blocks are areas in the user program which contain user data. There are shared data blocks which can be accessed by all logic blocks, and there are instance data blocks which are associated with a particular function block (FB) call.

Direct Keys

Direct keys are keys which are linked directly to the digital inputs of the controller (for example, as hardware wiring or via a DP interface) via the digital outputs of the display device. They enable you to operate the motion directly in the motion screen on the display device.

Edge Monitoring

Edge monitoring monitors a specific address for a defined edge (rising or falling). The error state occurs after an edge change if the incorrect level is at the address for longer than the defined delay time (for example, level “1” after a rising edge).

Error Definition

In error definitions, you can define the exact error which is to be monitored. Such error definitions can be appended to addresses in the LAD/STL/FBD Editor or created with S7 PDIAG.

Error Message

The errors detected by S7 PDIAG are registered on all connected display devices using the message text you configured. You can enter message texts for error messages while you are configuring the error definitions.

Exclusion Addresses

Exclusion addresses are addresses which you define in a list as “never causing an error.” Criteria analysis hides these addresses and the subnets which contain them if they are registered as having the value “0” (only in conjunction with ProAgent, version 5.0 or higher).

Executability

Executability means that the executability of the motion is enabled. The executability is defined as “Executability1/Executability2” when using the UDT_Motion.

Formal Addresses in Message Texts

S7 PDIAG enables you to adapt message texts automatically to the corresponding instances during the generation run. A number of formal addresses have been provided for this purpose, which are then replaced when the error definition is compiled or when the message is replaced.

Function Block (FB)

According to the International Electrotechnical Commission’s IEC 1131-3 standard, function blocks are logic blocks that reference an instance data block, meaning they have static data. A function block allows you to pass parameters in the user program, which means they are suitable for programming complex functions that are required frequently, for example, closed-loop control, operating mode selection.

Function Block Diagram (FBD)

Function Block Diagram is a graphic representation of the STEP 7 programming language. FBD uses the logic boxes from the familiar Boolean algebra to represent logic.

General Monitoring

With general monitoring, you can specify your own monitoring logic as a sequence of logical expressions. You create a monitoring logic using the language elements available, which enables you to carry out complex error monitoring. The error state occurs when the defined conditions are fulfilled.

Group Error

A group error is an error which is passed from a lower-level unit causing an error to the highest unit in the unit overview of S7 PDIAG and is displayed at the display device.

Initial Value

Initial values are the binary states which have led to the result of logic operation for the address being monitored.

Initial Value Acquisition

Initial value acquisition in S7 PDIAG causes all initial values for the address being monitored to be saved in the programmable logic controller during the same cycle in which the error was detected.

Instance Data Block

An instance data block stores the formal parameters and static data for function blocks. An instance data block can be associated with a function block call or a call hierarchy of function blocks.

Interlock Monitoring

Interlock monitoring is one of the four types of motion monitoring. With interlock monitoring, you monitor whether the interlock condition (executability) is fulfilled once the motion has been triggered and after a specified period of time (interlock time) has elapsed.

Interlock Time

The interlock time is the time within which the interlock conditions must be fulfilled.

Ladder Logic (LAD)

Ladder Logic is a graphic representation of the STEP 7 programming language. Its syntax corresponds to the representation of a circuit diagram.

Level Monitoring

With level monitoring, a specific address is monitored for a defined level (0 or 1). The error state occurs when the address has had the specified level for longer than the defined delay time.

Library

A library is a container for blocks, source files, and charts.

Modifying Times Online / Offline

If an existing monitoring definition contains a monitoring time (not equal to 0), you can modify this using the “Modify Times” function without having to generate the monitoring blocks again. You can modify times both offline and online, and it has the advantage of enabling you to determine the monitoring time step-by-step.

Monitoring Blocks

Monitoring blocks are the blocks for error detection as well as for initial value and status acquisition, which are generated from the error definitions you create in S7 PDIAG. After you have downloaded these monitoring blocks to your user program, you can carry out process diagnostics.

Monitoring Definitions

S7 PDIAG offers the following types of monitoring:

- Address monitoring
- General monitoring
- Motion monitoring

Monitoring Logic

The monitoring logic is the logic defined with the S7 PDIAG language elements which is used to monitor your process. With address monitoring and motion monitoring, the monitoring logic is already predefined and you only need complete the logic.

With general monitoring, you can program your own logic using the language elements in S7 PDIAG.

Monitoring Type

This term is used for the type of an error definition, in contrast to its instance.

Motions

Motions in a process are often defined as follows:

- They have two directions with two or more stable final positions
- They can be moved in the corresponding direction when triggered

For example, a cylinder moves from the current final position to the target final position when the hydraulic pressure is switched on.

A system or machine may contain a large number of motions. It is therefore a good idea to combine motions with similar functions together in a subsystem, which is referred to here as a unit.

Motions are sequences in the process which are monitored using error definitions. You can create several error definitions for each motion. A motion can only be contained in a unit, and represents an actual movement of a physical object in the process (for example, a punch moving up and down).

Motions are defined by the fact that the UDT_Motion is used in a block. Predefined Ladder networks are available for controlling motions easily.

Motion Monitoring

With motion monitoring, you can monitor whether physical motions have been carried out correctly and quickly enough in your process.

S7 PDIAG supports you by providing the UDT_Motion, which has a preprogrammed data structure for controlling and monitoring the motion.

There are four different ways of monitoring motions in S7 PDIAG, each of which monitors specific parameters:

- Action monitoring
- Reaction monitoring
- Interlock monitoring
- Startup monitoring

Online / Offline

When online, a data link between the programming device and the programmable logic controller exists; when offline, no connection exists.

Organization Block (OB)

Organization blocks form the interface between the CPU operating system and the user program. The sequence in which the user program should be processed is laid down in the organization blocks.

Other Error Definitions

This term is used for the S7 GRAPH and S7 HiGraph error definitions.

Position Flag

This is defined as “Position_Flag(n)” when using the UDT_Motion (see network 6 in Appendix A.6).

Priority of Error Definitions

You can assign a priority between 1 and 16 to error definitions. Each priority is assigned to a bit in a memory word. This bit is set when an error with the corresponding priority occurs and not reset until the last error with this priority has been removed.

The priority weighting is carried out in the same way as for the display devices. The assignment of priorities to the bits in the memory word is as follows:

- Priority 1: = Bit 0 in the low byte of the memory word
- Priority 16: = Bit 7 in the high byte of the memory word

Note: In projects which have been created with S7 PDIAG, version 3.0, all monitoring definitions implicitly receive the priority “1.”

ProAgent

ProAgent is an optional package in ProTool and also a configuration software package for your display device (OP).

Programmable Logic Control System (PLC)

A programmable logic control system is a programmable controller or part of a controller on which the user program runs. Programmable logic control systems can be, for example: SIMATIC S7, M7, and C7 devices.

Project

A project is a container for all objects in an automation solution, independent of the number of stations, modules, and how they are connected in a network.

ProTool

ProTool is the configuration software for your display device (OP).

Reaction Monitoring

Reaction monitoring is one of the four types of motion monitoring. With reaction monitoring, you monitor whether a final position which has been reached remains stable after a specified period of time (reaction time) has expired.

Reaction Time

The reaction time is the time by which the target final position must have been reached and stabilized.

S7 Graph

The S7 Graph programming language extends the functional scope of STEP 7 to include a graphical means of programming sequential control systems. S7 Graph is an optional package for STEP 7.

S7 HiGraph

The S7 HiGraph programming language for S7-300/400 devices extends the functional scope of STEP 7 to include a state graph programming system. S7 HiGraph is a programming tool for creating control programs on the basis of state graphs and is an optional package for STEP 7.

S7 PDIAG

Using the S7 PDIAG optional package, you can configure monitoring definitions which monitor your process for specific errors. You may configure these errors while you are creating your user program. The following types of monitoring are available: address monitoring, general monitoring, and motion monitoring.

S7 Program

An S7 program is a container for blocks, source files, and charts for S7 programmable modules which also contains the symbol table.

Scan Cycle Monitoring Time

If the processing time for the user program exceeds the set scan cycle monitoring time, the operating system produces an error message and the CPU goes into STOP mode.

Scan Cycle Time

The scan cycle time is the time the CPU takes to process the user program once only.

SIMATIC Manager

The SIMATIC Manager is the graphic user interface for SIMATIC users under Windows 95.

Standard Group

The standard group is created by default. From the standard group you can move units to other groups defined by you (max. of 15).

If a group created by you is deleted, the units contained in it are reintegrated automatically into the standard group.

Startup Monitoring

Startup monitoring is one of the four types of motion monitoring. With startup monitoring, you monitor whether the current final position (actual final position) is left following a machine operation (trigger) within a specified period of time (startup time).

Startup Time

The startup time is the time within which a motion must have started.

Statement List (STL)

Statement List is a textual representation of the STEP 7 programming language, similar to machine code.

Syntax Check

In incremental input mode for STEP 7 programs, a syntax check is run after each line has been completed. This means that the software checks whether, for example, a STEP 7 statement has been entered correctly. In free-edit mode, the syntax check is run during compilation.

System Data

“System Data” is an object containing the configuration data and parameters of a station.

System Function (SFC)

A system function (SFC) is a function integrated in the CPU operating system which can be called in the user program when required.

Target Final Position

The target final position is the the final position which is to be reached by the current motion. It is defined as “Final_Position(n)” when using the UDT_Motion.

Trigger

The trigger starts the motion. It is defined as “Control1/Control2” when using the UDT_Motion.

UDTs in S7 PDIAG

The user-defined data types supplied with S7 PDIAG can be saved as blocks. In this way, you can re-use any UDT which you have created many times:

- As a normal data type, or
- As a template for creating blocks with the same data structure.

The following UDTs are available in S7 PDIAG:

- The UDT_Unit
- The UDT_S_Unit
- The UDT_Motion

UDT_Unit

The UDT_Unit incorporates the information required in order for the display device (operator panel) to assign an alarm message to the faulty program location.

The UDT_Unit contains the following:

- Group error detection and group error acknowledgement
- 16 operating modes, of which two are predefined as “manual operation” and “automatic operation.” You can define the remaining 14 operating modes according to your own individual requirements.

UDT_Motion

The UDT_Motion represents a standardized interface between S7 PDIAG and the display devices (operator panels) and contains all the parameters for the following:

- For displaying motions in motion screens on the display device without the need for additional configuration, and
- For manually moving these motions in the motion screen on the display device.

Note: The requirement for this is that you use the Ladder networks for motion programming supplied in FB100 of the sample project “S7_DIAG”.

UDT_S_Unit

The UDT_S_Unit contains the group error address and the group error acknowedgement. This saves memory space and means that the program no longer has to run through the operating mode of the process unit in all subunits.

Unit

Units structure the process view according to components which are related to one another by their technical function. If you have set up your project so that each block relates to a physical object in the process (for example, a press, a punch, or a safety guard), the units represent an image of your process. A unit exists for each block in your program which can be diagnosed.

Units can also save data common to all other units, motions, and function blocks which lie below them in the hierarchy.

A unit may contain error definitions, motions, and other subunits.

Using units, you can combine both individual errors and motions into a technological unit. This enables you to find process errors quickly and easily.

Units are represented with other objects in a tree structure in the unit overview. Units for a data block, function, or organization block are also visible in the unit overview on the display devices.

User Block

The user block is a function block with a predefined interface which is supplied with S7 PDIAG. This block enables you to react to specific errors in your user program without substantial programming being necessary. It is called with different parameters, depending on whether the error is incoming or outgoing. For example, it provides you with information on the unit, the priority, and the message number of the error.

User-Defined Templates

As well as the predefined monitoring definitions supplied with S7 PDIAG, you can also create your own templates for specific monitoring definitions. Using templates makes configuring much easier and involves less time and effort. You can also store incomplete monitoring logic in your own templates.

Note that you can only store a message template within a template. This is why no message number is assigned in a template. However, you can still configure the message text for your monitoring definition in the template.

User Program

The user program contains all the statements and declarations and the data required for signal processing to control a plant or a process. The program is linked to a programmable module (for example, CPU, FM) and can be structured in the form of smaller units (blocks in S7 and tasks in M7).

Variable

A variable defines an item of data with variable content which can be used in the STEP 7 user program. A variable consists of an address (for example, M3.1) and a data type (for example, BOOL), and can be identified by means of a symbolic name (for example, BELT_ON).

Variable Table (VAT)

The variable table is used to collect together the variables that you want to monitor and modify and set their relevant formats.

Index

F

G

I

L

M

N

O

P

R

S

T

U

W

Siemens AG
A&D AS E 81

Oestliche Rheinbrueckenstr. 50
D-76181 Karlsruhe
Federal Republic of Germany

From:
Your Name: _
Your Title: _
Company Name: _
Street: _
City, Zip Code _
Country: _
Phone: _

Please check any industry that applies to you:

- ❐ Automotive
- ❐ Chemical
- ❐ Electrical Machinery
- ❐ Food
- ❐ Instrument and Control
- ❐ Nonelectrical Machinery
- ❐ Petrochemical
- ❐ Pharmaceutical
- ❐ Plastic
- ❐ Pulp and Paper
- ❐ Textiles
- ❐ Transportation
- ❐ Other _ _ _ _ _ _ _ _ _ _ _

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:

_ _
_ _
_ _
_ _
_ _
_ _
_ _
_ _
_ _
_ _
_ _
_ _