SIEMENS

SIMATIC

ProC/C++ for M7-300 and M7-400 Writing C Programs

User Manual

This manual is part of the documentation package with the order number:

6ES7812-0CA01-8BA0

C79000-G7076-C519-01

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® and SINEC® 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
Automation Group
Industrial Automation Systems
Postfach 4848, D-90327 Nürnberg

Siemens Aktiengesellschaft

C79000-G7076-C519

Preface

Purpose

The information in this manual makes it possible for you to:

- Install and operate the M7 ProC/C++ optional package.
- Import symbols in C/C++ programs and assign them to C variables.
- Write C/C++ programs for the SIMATIC M7 programmable controller with the help of Borland C++.

Contents of this Manual

This user manual describes the M7 ProC/C++ optional package.

The manual provides you with information about the following subject areas:

- Installing M7 ProC/C++
- Working with C/C++ programs
- Operating the integrated symbol import editor
- Writing a sample application

Audience

This manual is intended for STEP 7 users who write C and C++ application programs for SIMATIC M7.

The following knowledge is prerequisite to writing M7 applications with the aid of textual high-level language programming in C/C++:

- Using STEP 7

  Basic knowledge of the STEP 7 Standard software is imperative for the entire project management, symbol processing, and transfer of generated programs from the programming device to the M7 programmable controller.

  Refer to the STEP 7 User Manual for the relevant information.

- C and C++ programming languages

  Knowledge of the C and C++ programming languages is necessary for creating applications. In addition, you should be familiar with using the integrated Borland C/C++ development environment.

  You will find the necessary information in the documentation supplied with Borland C/C++.

What's New?

This manual provides the following new topics on changes and functional extensions supported from version V3.1 of M7 ProC/C++:

Topic	Chapter
Borland C++ V5.01 support	3
Direct symbol access	4

Additional Manuals Which You Require

The following table lists the manuals you need in addition to this user manual.

Title	Contents
Standard Software for S7 and M7 STEP 7 User Manual	STEP 7 installation, function, and operation
System Software for M7-300 and M7-400 Program Design Programming Manual	Introduction to writing M7 applications in C/C++
System Software for M7-300 and M7-400 System and Standard Functions, Reference Manual	Descriptions of the RMOS system call-ups M7-API library, and standard CRUN library
System Software for M7-300 and M7-400 Installation and Operation User Manual	Installation, startup, and operation of M7 programmable controllers
ProC/C++ for M7-300 and M7-400 Debugging C programs User Manual	Operating the Organon XDB386 debugger
Borland C++ Documentation on CD-ROM	• C/C++ programming • Operating the integrated development environment and the tools

How to Use This Manual

The manual is structured as follows to make it easier for you to access special information quickly:

- There is a complete table of contents at the beginning of this manual containing a list of all the figures and tables contained in the entire manual.
- The left column on each chapter page contains information providing you with an overview of the contents in that particular section.
- There is an extensive keyword index at the end of the manual providing you with fast access to the information you want.

Up-To-Date Information

You can find up-to-date information on SIMATIC products from the following sources:

- On the Internet under `http://www.aut.siemens.de/simatic`
- Via the fax polling number 08765-93 00 96 50

In addition, the SIMATIC Customer Support service provides up-to-date information and download facilities for users of SIMATIC products:

- On the Internet under `http://www.aut.siemens.de/simatic_cs`
- Via the SIMATIC Customer Support BBS (Bulletin Board System) under the following number +49 (911) 895-7100

  For dialing into the BBS, use a modem of up to V.34 (28.8 Kbps) and set the following parameters: 8, N, 1, ANSI, or alternatively use ISDN (x.75, 64 Kbps).

The phone and fax numbers of the SIMATIC Customer Support service are:
Tel. +49 (911) 895-7000
Fax. +49 (911) 895-7002.
You can also send queries directly using E-mail on the Internet or via the above-mentioned BBS.

Additional Assistance

If you have any questions regarding the software described in this electronic manual and cannot find an answer here or in the online help, please contact the Siemens representative in your area. You will find a list of addresses in the “Siemens Worldwide” Appendix of the STEP 7 User Manual, for example.

If you have any questions or comments on this electronic manual, please fill out the remarks form at the end of the manual and return it to the address shown on the form. We would be grateful if you could also take the time to answer the questions giving your personal opinion of this electronic manual.

Siemens also offers a number of training courses to introduce you to the SIMATIC S7 automation system. Please contact your regional training center or the central training center in Nuremberg, Germany for details:

D-90327 Nuremberg, Tel. (+49) (0911) 895 3154.

Contents

1 Product Overview

Overview

The M7 ProC/C++ optional software package makes it possible to integrate the development environment of Borland C++ into STEP 7. This creates the best possible situation for you when you write C/C++ programs for your SIMATIC M7 automation computer.

M7 Optional Packages

In addition to STEP 7, you need both M7 system software as well as a development environment for C programs when you write M7 application programs in the C and C++ high-level languages. These software components are located in the M7 optional packages.

This Chapter

This chapter informs you:

- Which functions M7 ProC/C++ offers
- Which M7 optional packages exist
- Which tools are available for writing C/C++ programs

Chapter Overview

1.1 Overview

Introduction

M7 ProC/C++ permits free programming in the C and C++ high-level languages with integrated code generation and an integrated debug functionality.

The example in Figure 1-1 shows you the environment in which you can write your M7 applications and where your applications run.

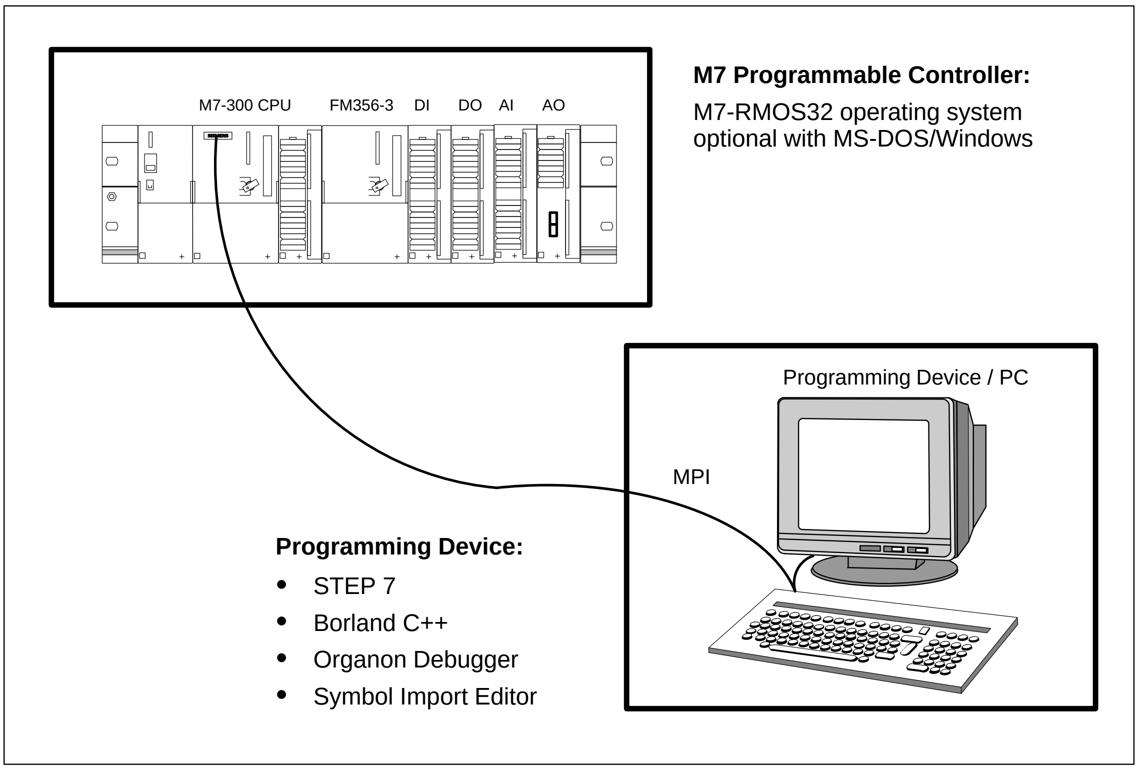


Figure 1-1 System Environment for an M7 RMOS32 Application

Programming Device

Additional tools for programming and writing applications for M7 programmable controllers can be integrated in STEP 7. It is also possible to incorporate both the integrated development environment of Borland C++ for textual, high-level language programming and the Organon XDB386 debugger.

The programming device is connected to the M7 programmable controller via a multipoint interface (MPI).

M7 Applications

The M7 RMOS32 API interface in the form of C libraries is offered to you for accessing M7 system resources, from accessing I/O modules up to managing tasks.

M7 Programmable Controller

The CPU and FM modules of the SIMATIC M7-300 and 400 automation computer family are the hardware platform for M7 applications. The modules are all AT compatible and offer almost all the system components of an AT-compatible PC.

The 32-bit real-time multitasking operating system, M7 RMOS32 (Real-Time Multitasking Operating System), which can be used with the MS-DOS and MS Windows operating systems, forms the operating system basis for the modules of the SIMATIC M7 automation computer family. The integrated M7 server connects additional S7 components.

M7 ProC/C++ Function Range

The M7 ProC/C++ optional package offers you the following functionalities:

- Incorporating the integrated development environment of Borland C++ (Borland IDE = Integrated Development Environment) into STEP 7
- Importing symbol information in C/C++ programs
- Remote testing of C/C++ programs with the Organon XDB386 integrated debugger
- Online help for the functions named above

1.2 M7 Optional Packages

Overview

In addition to STEP 7, you will need one or more M7 optional packages to write M7 application programs.

Table 1-1 Optional Packages for M7 Programming

Package	Contents
M7 SYS	• M7 RMOS32 operating system • M7 API system library and RMOS libraries • Header files for M7 API and M7 RMOS32 • MPI utilities
CFC	Programming software for CFC applications
M7 ProC/C++	• Integration of the Borland development environment into STEP 7 • Symbol Import Editor and Generator • Organon XDB386 debugger
Borland C++	Borland C++ development environment

Interdependencies

The following figure illustrates the interdependencies between the M7 optional packages:

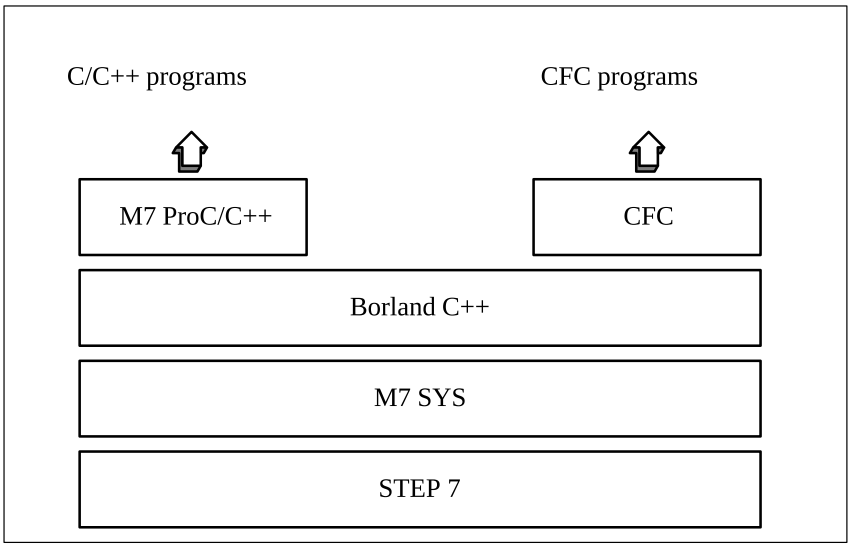


Figure 1-2 Interdependencies of the M7 Optional Packages for M7 Programming

Table 1-2 Summary

For Writing...	You Need the Following M7 Optional Package...
C/C++ programs	1. M7-SYS 2. M7-ProC/C++ 3. Borland C++
CFC programs	1. M7-SYS 2. CFC 3. Borland C++

In addition, you need STEP 7.

Where You Will Find Support

The specific tools for writing M7 applications are partially integrated into STEP 7 and are partially integrated into the M7 optional packages. The following table illustrates the support you find in the individual packages.

Table 1-3 Support for Writing M7 Application Programs

Package ...	Supports You ...
STEP 7	• When you install the M7 operating system • When you manage the M7 programmable controller • When you transfer, start, and delete the M7 application programs • When you call up status data and diagnostic data • When you perform a CPU memory reset
M7-SYS	Through the M7 operating system utilities and system software for: • Sequential control systems • Memory and resource management • Access to computer hardware and SIMATIC hardware • Interrupt management and diagnostics • Status monitoring • Communication
Borland C++	• For writing C and C++ programs
M7-ProC/C++	• Through integrated code generation (integrating the C development environment) • Through integrating project symbolics in the source code • Through the integrated debug functionality
CFC	• For writing, testing, and debugging CFC programs • For starting and running CFC programs

1.3 Overview of the Tools for Writing and Testing Programs

STEP 7

STEP 7 offers you the basic functionality you need to:

- Create and manage projects
- Configure and set parameters for the M7 programmable controller hardware
- Configure networks and connections
- Manage symbol data
- Transfer data to the M7 programmable controller via MPI
- Query information about the M7 programmable controller
- Carry out certain settings on the M7 programmable controller and to perform an overall reset on the M7 programmable controller.

This functionality is independent of whether your programmable controller is the SIMATIC S7 or the SIMATIC M7. You will not notice any differences in the S7 and M7 programmable controllers until you write programs. The programmable controllers have different operating systems and execution software. To write C/C++ application programs for M7, you need the following software:

- C/C++ development environment Borland IDE
- Symbol import and symbol generator for C variables (contained in M7 ProC/C++)
- Remote debugger for Borland programs (contained in M7 ProC/C++)
- System software for M7-300 and M7-400 (contained in M7 SYS)

Borland IDE Development Environment

M7 ProC/C++ integrates the Borland IDE C/C++ development environment into the SIMATIC Manager. This means it is called up automatically as soon as you edit an M7 C/C++ program. For C/C++ programs, you can comfortably edit, compile, link, and simultaneously access all project-specific configuration data and symbol data. In addition, you have all the Borland C++ tools available: AppExpert, ClassExpert, project management, Resource Workshop, command line tools, etc.

Symbol Import Editor and Symbol Generator

The integration of symbols in the source code makes it easier to control inputs and outputs.

With the Symbol Import Editor, you can select symbol entries for your current C/C++ program from the program-specific symbol table and connect them with C variables of your C/C++ program. The generator creates several header files that are integrated in your C/C++ program. This makes it possible to access inputs and outputs of signal modules using C variables. The header files for the application are generated automatically upon program compilation.

The Symbol Import Editor and the Symbol Generator are components of the M7 optional package, M7 ProC/C++. You can call up:

- The Symbol Import Editor directly from the Borland IDE.
- The Symbol Generator from both the Borland IDE and the Symbol Import Editor.

Organon Debugger

The Organon XDB386 debugger is integrated in M7 ProC/C++. The Organon XDB386 debugger makes the remote debugging of C/C++ applications possible on the M7 programmable controller.

System Software for the M7-300 and M7-400

The system software for the M7-300 and M7-400 contains:

- The operating system and the M7 system server for the M7-300 and M7 400
- Libraries and header files for creating the program
- Software for communication between the programming device and the programmable controller, consisting primarily of:
  - Drivers for MPI communication
  - The remote terminal interface for remote operation of the M7 programmable controllers

These software components are contained in the M7 SYS optional package.

Note

The standard functions and system functions for RMOS API, M7 API, and the CRUN library are not implemented as C++ class libraries.

Installing M7 ProC/C++ 2

Overview

This chapter provides you with the information you need to install the M7 ProC/C++ software package successfully. This chapter also includes information about the Symbol Import Editor, Symbol Generator, and Organon XDB386 debugger tools.

Chapter Overview

2.1 Prerequisites for Using M7 ProC/C++

Hardware Prerequisites

To generate applications for an M7 automation computer with the help of STEP 7 and the Borland C/C++ development environment, you need the following hardware:

- A programming device (for example, the PG 740) or an AT-compatible PC with an 80486 processor (or higher) and at least 16 MB RAM, and a VGA monitor.
- An MPI connection (MPI = Multi Point Interface) between the programming device and the M7 automation computer. To establish an MPI connection, you need:
  - An MPI card with a cable for the PC, order number: 6ES7793-2AA00-0AA0 that must be installed in your PC or programming device (this MPI accessory is already installed in some programming devices), or
  - An MPI cable for the PC, order number 6ES7901-0BF00-0AA0 that is connected to the MPI card for your PC or programming device
- Approximately 5 Mbytes free memory on the hard disk drive to install M7 ProC/C++.
- Free memory on the hard disk drive to install
  - STEP 7
  - M7 SYS system software
  - Borland C++

  The free memory required for these components can be determined during installation.
- An available memory of at least 1 MB on drive C: for the setup program. After successful installation, the temporary setup files are deleted again.
- We recommend using a mouse supported by Windows 95.

Software Prerequisites

The following prerequisites must be met for the installation of M7 ProC/C++:

On the PC or programming device:

- The **Windows**®**95** operating system must be installed on the PC or programming device.
- The **STEP 7 V3.1** Standard software must be installed on the PC or programming device. The STEP 7 user manual describes how to install STEP 7.
- **Borland C++ V5.01**: You must have the Borland integrated development environment installed on your PC or programming device.

  The Borland documentation describes the installation of the integrated development environment.
- The **M7 SYS** optional package V1.2 or higher must be installed on your PC or programming device. The M7 SYS user manual describes how to install this software.

Note

The optional software packages M7 SYS and Borland C++ can also be installed at a later date, since they are only necessary for the operation of M7 ProC/C++.

On the M7 automation computer:

For the execution and remote debugging of M7 application programs, a complete M7 RMOS32 operating system (optional with MS-DOS or MS Windows) must be installed on the M7 programmable controller.

2.2 Installation

Overview

M7 ProC/C++ contains a setup program that carries out the installation automatically. Screen prompts guide you step-by-step through the entire installation procedure.

Procedure

You must start Windows 95 before you begin the installation.

1. Insert the data medium into the relevant PC or programming device drive (usually designated as drive A: or drive B:).
2. Open the Explorer and select the relevant drive. The Explorer displays the contents of diskette 1.
3. Select the **Setup.exe** file. The file is highlighted.
4. Double-click Setup.exe or press the enter key to start the installation.
5. The installation is carried out automatically.

   The Organon XDB386 debugger is automatically installed as well.
6. At the end of the installation, a message box is displayed. The message shows you the installation was successful.

Log File

The S7CM7LOG.TXT log file is created during the installation.
This file contains information significant for the Hotline service.

2.3 Settings and Options

Compiler and Linker Options

The following paths important for the compiler and linker calls are preset in the Borland development environment:

- For header files (*.h):

  `c:\siemens\step7\m7sys2.00\inc`

- For library files (*.lib):

  `c:\siemens\step7\m7sys2.00\lib`

If you set directories other than **c:\siemens\step7** during the STEP 7 installation, you must provide the Borland development environment with this information.

Setting Paths

To set other paths, you must proceed as follows before using M7 ProC/C++ for the first time:

1. Start the *regedit.exe* Registration Editor.
2. Obtain the <PATHS> identifier from the registry area HKEY_LOCAL_MACHINE\SOFTWARE\SIEMENS\AUTSW\M7SYS2.00
3. Start the Borland IDE outside of the SIMATIC Manager.
4. Open the Borland C default project. The project path is:

   `<PATHS>\s7cm7\c__tmplb\defaultp.ide`

5. Call up the **Options** ▸ **Project** ▸ **Directories** menu command and adapt the directories in the “source directories” field.

   Include: **`<PATHS>\m7sys2.00\inc`** for *.h files

   Library: **`<PATHS>\m7sys2.00\lib`** for *.lib files

6. Generate a make file for the project with the **Project** ▸ **Generate make file** menu command.
7. Use the **Project** ▸ **Close** menu command to close the project and save your changes.
8. Open the Borland C++ default project. The project path is:

   `<PATHS>\s7cm7\cpptmplb\defaultp.ide`

9. Repeat Steps 4 through 8 for the C++ project.

Result: The changed paths are entered into the default C and C++ projects.

Generating with Debugger Options

The debugger options for the compiler and linker are normally activated in the Borland project file (*.ide). To optimize the generated code after troubleshooting has taken place, you can turn off the debugger options. (See Section 5.2.)

Help Files Paths

The M7 SYS optional package contains a help file you can integrate in the Borland help search area. (See Section 3.1.)

The path name is: **`c:\siemens\step7\s7bin\m7rrmosb.hlp`** or **`s7rcm71b.hlp`**

Additional help files with reference information are located in this path.

Using Multiple Versions of M7 SYS

The current version of M7 ProC/C++ supports the simultaneous operation of more than one version of the system software M7 SYS on the PC or programming device. If you are using M7 SYS V2.0, the default paths stated above are valid.

You need to adapt this path if:

- You are using more than one version of M7 SYS.
- You are working with projects created with an earlier version of M7 ProC/C++.

3 Working with C/C++ Programs

Overview

This chapter describes how you use M7 ProC/C++ to do the following:

- Write C/C++ programs
- Modify existing programs
- Document programs
- Use online help

C/C++ Programs for M7-300/400

You use the integrated development environment from Borland (Borland IDE) to write C/C++ programs. You can write the following two program types for M7-300/400 systems in the C and C++ programming languages:

- M7 RMOS32 programs
- MS-DOS or MS Windows programs

Borland Projects

In the Borland IDE, the C/C++ programs have a project structure containing all software components needed to write a C/C++ application program. A project in the sense of the Borland IDE is therefore a component of the entire project of your automation task and is always located in an M7 program object.

To avoid misunderstandings, the projects you process with Borland IDE will always be referred to as C/C++ programs in the following.

Borland Documentation

The Borland documentation provided with the software on diskettes / CD-ROM contains a complete description of the functionality and operation of the Borland development environment.

Chapter Overview

3.1 Main Procedure

Calling Up the Borland IDE

You have two means of calling the integrated Borland development environment:

1. From the SIMATIC Manager in the context of your current project, if you want to process or modify existing C/C++ programs. Symbol import and the Organon debugger are supported. This procedure is described in this user manual.
2. From the Borland C++ program group, if you double-click the relevant icon. In this case, there is no support for symbol import and the Organon debugger. This procedure is not handled to any other extent in this manual.

Note

The SIMATIC Manager can process only directories named with the “8.3” **xxxxxxxx[.yyy]** notation. Long directory names or file names are not supported.

Procedure

You usually proceed as follows when you process C/C++ programs:

1. Select an **M7 program** object in the SIMATIC Manager.
2. Create a C/C++ program.
3. Open the C/C++ program to process it with the Borland IDE, from editing the source code up to generating the executable code.

If necessary, you can also carry out the following actions in any order:

- Modify the program properties, such as name or call parameter.
- Determine which software components from your program you want to transfer to the M7 programmable controller.
- Document your program.

These actions are described in Section 3.4.

Using Help

You have the following help when writing C programs:

- Help for the Borland IDE
- Help for the system function calls and standard function calls for M7-300 and M7-400, and functions for symbol import.

  Insert the following help files from the M7 system software into the search area of the OpenHelp help function in the Borland IDE:

File	Contents
m7rrmosb.hlp	M7-300 and M7-400 system software
s7rcm71b.hlp	Functions for symbol import

Proceed as follows:

1. Open an MS-DOS window and call the “OpenHelp” command. The “Configuring OpenHelp” dialog box opens. This dialog box usually contains the “Borland C++ IDE Help Files” help search area in the upper field.
2. Select the “Borland C++ IDE Help Files” help search area.
3. Click the “Add” button. A file selection dialog opens up.
4. Select the help files named above. Section 2.3 contains the paths. The selected files are displayed in the “Available Help Files” list field.
5. Select the help files in the “Available Help Files” list field. Transfer the files into the search range by clicking the “>” button.

   Result: The help files paths also appear in the help search area window and are available for context-sensitive access from the Borland Editor.
6. Click “OK” to close the dialog box .

3.2 Writing and Opening C/C++ Programs

Application

Within an M7 program container, you have the possibility of writing new C/C++ programs or modifying existing C/C++ programs.

Writing New C/C++ Programs

To write a new C or C++ program, proceed as follows:

1. Insert an M7 program object in the opened project window using the following menu command:
   Insert ▸ Program ▸ M7 Program
2. Select the M7 program.
3. Select one of the following menu commands to insert the desired program type:

Table 3-1 C for M7 Program Types

Menu Command	Program Type
Insert ▸ M7 software ▸ DOS/Windows Program	C or C++ program for MS-DOS/Windows
Insert ▸ M7 Software ▸ C Program	C program for M7 RMOS32
Insert ▸ M7 Software ▸ C++ Program	C++ program for M7RMOS32

Result:

A C or C++ program object is created and the symbol for the created object appears in the project window. The created object corresponds to a Borland IDE project.

C and C++ program objects contain a name with consecutive numbering as standard, for example, “C program 1” or “C++ program 4.” If you want to change the program name, you can subsequently carry out the change in the “C Program Properties” dialog box . (See Section 3.4.)

Opening C/ C++ Programs

To process a C/C++ program, you must either

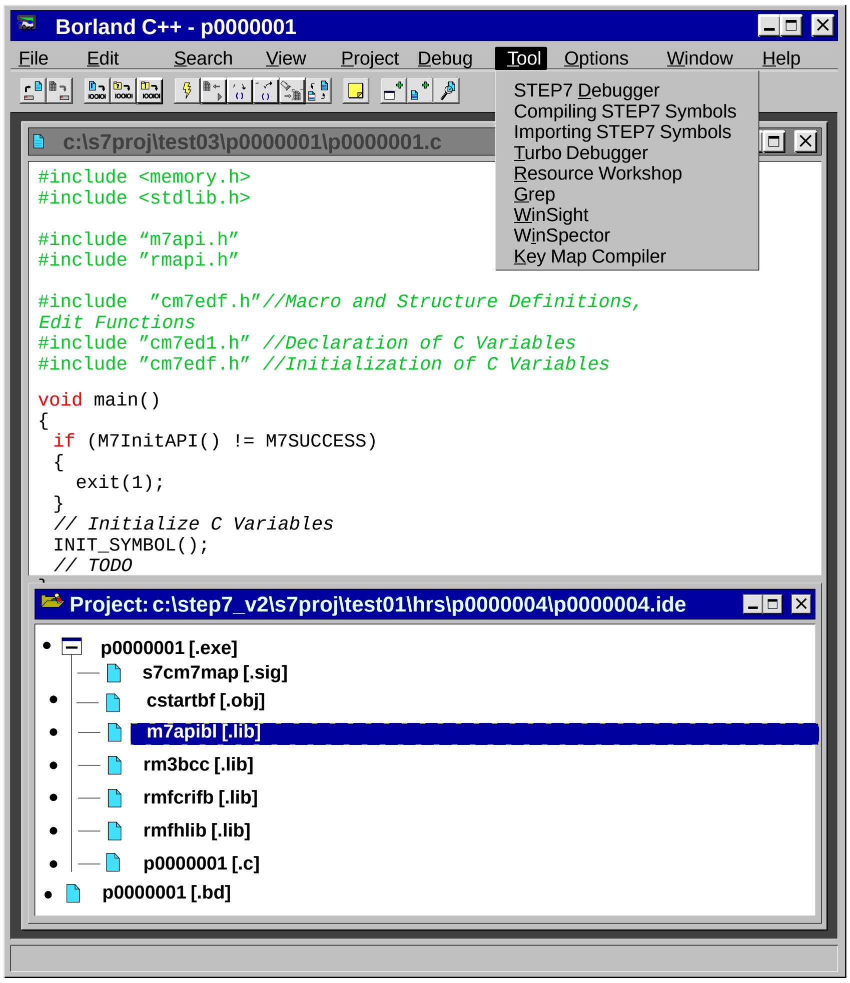
- Double-click the C/C++ program object, or
- Select the C/C++ program project and call up the **Edit ▸ Open Object** menu command.

This opens the main window in the Borland development environment (IDE).

The Borland IDE Main Window

Multiple windows can be opened in the Borland IDE main window, for example:

- The editor window for the source code to be processed, and
- The project window.



```
#include <memory.h>
#include <stdlib.h>

#include "m7api.h"
#include "rmapi.h"

#include  "cm7edf.h"//Macro and Structure Definitions,
Edit Functions
#include "cm7ed1.h" //Declaration of C Variables
#include "cm7edf.h" //Initialization of C Variables

void main()
{
  if (M7InitAPI() != M7SUCCESS)
  {
    exit(1);
  }
  // Initialize C Variables
  INIT_SYMBOL();
  // TODO
```

Project: c:\step7_v2\s7proj\test01\hrs\p0000004\p0000004.ide

```
p0000001 [.exe]
    s7cm7map [.sig]
    cstartbf [.obj]
    m7apibl [.lib]
    rm3bcc [.lib]
    rmfcrifb [.lib]
    rmfhlib [.lib]
    p0000001 [.c]
p0000001 [.bd]
```

Figure 3-1   C Program in the Borland Window

Editor Window Contents

C/C++ Programs for M7RMOS32:

After you have opened a new C program or C++ program for M7 RMOS32, you can edit the source code, for example, by double-clicking the source code node. The editor window shows you a sample of a source file. Include directives for integrating the following header files are already inserted in this source file:

- Header files for the function calls of the RMOS, M7 API, and CRUN functions
- Header file for the symbol table for symbolic addressing

You can now start editing your program here.

DOS and Windows Programs:

After you have opened a new DOS or Windows program, you must then select the **Project ▸ New project** menu command in the Borland IDE. The *New Project* dialog box is displayed through the call. You must carry out all necessary settings in this dialog box .

- Use the default path and name specified by Borland.

  A note that this project already exists appears in the Borland dialog box . You are asked if you want to overwrite this project. Respond with “Yes.”

- The Borland user documentation contains instructions for additional settings.

Project Window Contents

The dependency tree structure of the individual files for generating your program is displayed in the project window. In this tree, the following files already take the dependencies of your RMOS program into consideration (see Table 3-2):

- Startup code for Borland/RMOS: cstartbf.obj for C programs and cstrtbfp.obj for C++ programs
- RMOS libraries: rm3bcc.lib, rmfcrif.lib, and rmhlib.lib
- M7 API library: m7apibl.lib

Generating the Symbol Header Files

Each time you change symbols and before the compiler is started, the symbol generator is called up in the Borland IDE to generate new symbol header files.

When you compile from the SIMATIC Manager, the symbol generator is also called up automatically.

Compiling the Programs

You have several means of compiling C and C++ programs, for example:

- In the Borland IDE with the **Project** commands
- In the SIMATIC Manager without starting the Borland IDE (the make file must have been updated first). When the program source code is complete, select the C program object or C++ program object in your project window. Call up the **Edit ▸ Compile** menu command from the menu bar or from the context menu (right mouse button).

  Result: The C/C++ program is compiled so that you can then transfer the executable program with the **PLC ▸ Manage M7 System** menu command to the programmable controller.

Working with the Borland IDE

Refer to the Borland user documentation for more detailed information about working with the Borland IDE.

Opening “old” C/C++ Programs

C/C++ programs created with earlier versions, for example, version 1.1, and programs created with the current version 3.1 of M7 ProC/++ differ in the following:

- The import table S7CM7MAP.SIG was not integrated in the Borland project tree in earlier programs.
- The symbol generator was not called up automatically when the C/C++ program was compiled.

To adapt “old” programs, proceed as follows:

1. Open the program with the menu command **Edit** ▸ **Open Object**.
2. Define the symbol generator as the compiler by doing the following:
   - Select the menu command **Options** ▸ **Tools** from the main window of the Borland IDE.
   - Select the symbol generator in the “Tools” dialog box and click the “Edit” button and then the “Other” button.
   - Set the “Tool type” to “Compiler”.
   - Exit the opened dialog boxes by clicking “OK”.
3. Integrate the import table S7CM7MAP.SIG into the Borland project tree. Proceed as follows:
   - Select the executable .EXE file for the program in the project window.
   - Select the menu command **Add Node** with the right mouse button.
   - Select and open the S7CM7MAP.SIG file from the dialog box “Add to Project List.” The file is added to the project tree. It must be the first item in the project tree.
4. Adapt the paths if needed (see Section 2.3 and the “README” file for M7 ProC/C++).

3.3 C/C++ Program Structure

Project Structure

The following table provides a list of the files generated when a new C/C++ program is generated for M7 RMOS32, copied into the respective Borland project directory, or linked to the compiled code.

Eight-character long default names are assigned to the project-specific files (for example, p0000001.ide).

Table 3-2 RMOS C/C++ Program Structure (example **motor_22**)

File	Description	What is stored?	When are the data entered?
engine_22[.ide]	Borland project file	Project-specific settings for the development environment	When a Borland project is written
engine_22[.exe]	File executable on the M7	Instructions for the processor, executable program	During compiling or linking the project files
engine_22[.cpp] engine_22[.c]	C++ or C source module	Files and instructions for program processing	When editing the source text
cstartbf[.obj]	Startup code for C programs	Code executed first when a C program is started	During the installation of the M7 SYS package
cstrtbfp[.obj]	Startup code for C++ programs	Code executed first when a C++ program is started	
rm3bcc[.lib]	RMOS library	Interface to the Borland compiler	
rmfcrif[.lib]	RMOS library	Interface to CRUN functions	
rmfhlib[.lib]	RMOS library	Interface to RMOS system functions, RMOS API	
m7apibl[.lib]	M7 API library	Interface for the M7 server utilities, M7 API	
motor_22[.bd]	Bound file	Information for the Organon xdb386 remote debugger	After writing the executable program
s7cm7map[.sig]	Import table	Symbol information assigned to C variables	When the import table is saved by the Symbol Import Editor

3.4 Modifying Program Properties

Program Properties

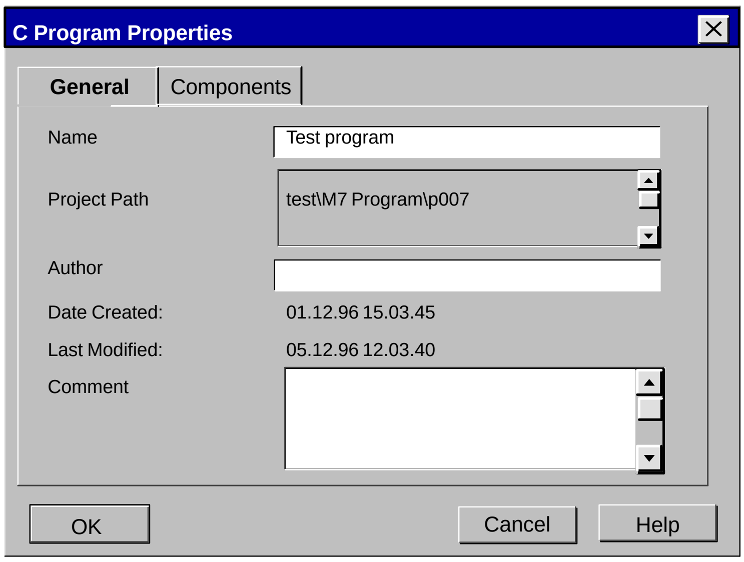
You can change the following properties for M7 C/C++ programs:

- Program names
- Source file
- Call parameters
- Settings for transferring software components from the programming device to the programmable controller
- Documentation data

Procedure

You use the following steps to modify an existing C/C++ program:

1. Select the C, C++, or DOS/Windows program you want to modify in the open project window of the SIMATIC Manager.
2. Select the **Edit ▸ Object Properties** menu command. The “C Program Properties” dialog box is opened.
3. Open the “General” tab if you want to edit the program name or the author (Figure 3-2).
4. Open the “Components” tab if you want to change properties of program components (Figure 3-3).
5. Edit the properties you want to change in the dialog box.



Name	Test program
Project Path	test\M7 Program\p007
Author	
Date Created:	01.12.96 15.03.45
Last Modified:	05.12.96 12.03.40
Comment	

Figure 3-2 “C Program Properties” Dialog Box, “General” Tab

Name

The current program name visible next to the object symbol in the project window of the SIMATIC Manager is displayed in this text field. A default name is assigned when the object is created. You change the program name by editing directly in this field. The program name can be a maximum of 24 characters long.

Project Path

The directory under which the project is stored on the programming device is displayed in this field. You can not change this path.

Author

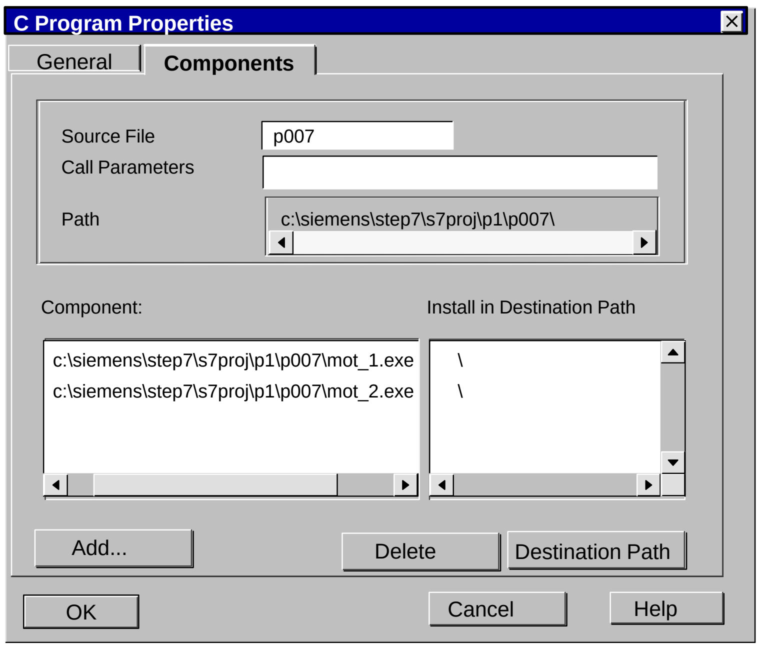
Enter your own name (max. 40 characters).

Date Created, Last Modified

These fields display the date and time at which the C/C++ program was created or when the properties of the C/C++ program were last changed.

Comment

Enter comments on the C/C++ program here (max. 254 characters). Entering a comment is optional.



Source File: p007
Call Parameters:
Path: c:\siemens\step7\s7proj\p1\p007\

Component:	Install in Destination Path
c:\siemens\step7\s7proj\p1\p007\mot_1.exe	\
c:\siemens\step7\s7proj\p1\p007\mot_2.exe	\

Figure 3-3 “C Program Properties” Dialog Box, “Components” Tab

Source File

The file name (the first eight characters) of the Borland project is displayed in this field. You can change this name if you need to. The name must always be eight characters long. If applicable, the name is automatically increased to eight characters when you end the properties dialog with the “_” underline character. All of the C/C++ program files are renamed to the new name.

Changing the name has a direct effect on the “Component” field (for files with the same name in the same directory). If necessary, the names are adapted here.

Call Parameters

You can enter call parameters with which your executable program is started on the programmable controller in this input field. M7 ProC/C++ generates a batch file in which the program call is saved with the call parameters. This file is transferred to the programmable controller.

Path

Displays the complete path under which the selected C/C++ program is stored as a Borland software project. If the path is too long to be displayed, you can shift the display.

Components, Install in Destination Path

The source path and destination path are displayed in these two list fields for each component to be transferred from the programming device to the programmable controller.

Add...

You can add a software component in the program with the help of the **Add...** button. For example, a Windows project could be expanded by an existing DLL in this manner.

When you select the **Add...** button, a file selection box appears. This box enables you to select the desired software component. The selected software component then appears in the “Component” field.

Note

In DOS or Windows programs, you must select all the components to be transferred to the M7 programmable controller by using the “Add” button, and then specify the required destination path. If the “Component” and “Install in Destination Path” fields remain empty, nothing will be transferred to the M7 programmable controller.

Delete

If you select a software component with the mouse and then click the “Delete” button, the affected software component is removed from the “Component” field.

Destination Path

After you select a software component and click the “Destination Path...” button, a dialog box appears.

The input field shows you the path under which the software component is saved on the M7 automation computer. The “\” root directory is preset. You can change this path by entering an absolute directory name without specifying a drive, for example: “\program1\motor2.”

Importing Symbols 4

Overview

This chapter describes how you generate, manage, and integrate C variables from the symbol table entries of an M7 program into your application program.

Import Table

The import table makes it easier to control peripheral inputs and outputs in your application program. You assign symbolic names of input signals and output signals to the C variables you use in your C/C++ program.

Direct Symbol Access

From the current product version onwards, direct symbol access is supported. This means that you can access symbols directly from the C program by calling the M7 API M7Loadxxx, M7LoadDirectxxx, M7Readxxx, M7Storexxx, and M7Writexxx functions.

Direct symbol access can be used both with new C programs created with the current version as as well as with old programs which were created with ProC/C++ version 1.1 and which have to be edited.

The previous symbol access via the C functions for symbol import continues to be supported for reasons of compatibility.

Chapter Overview

4.1 Opening the Import Table

Overview

The integration of the Borland C/C++ development environment in STEP 7 makes it convenient for you to generate and manage C variables that can be generated from the peripheral inputs and outputs of the symbol table of an M7 program.

You are supported in the following tasks:

- Selecting the symbols from the program-specific symbol table and converting the symbols into corresponding C variables.

  The selected symbols, as well as the relevant C variables are stored in the program-specific import table (file name: S7CM7MAP.SIG) in the current project path.
- Editing the C variables of the import table, as well as checking them for correct syntax and clarity.
- Generating header files with variable declarations, macros, and functions with which the C variables can be addressed from the C program.
- Automatic consistency checking of the import table and symbol table during every program conversion from the SIMATIC Manager menu.

  New header files are generated automatically and are integrated into the project when there are inconsistencies. This ensures that the current symbol parameters are always used.

Procedure

In contrast to other STEP 7 objects, the import table is not called up directly from the SIMATIC Manager, but is rather integrated into the C/C++ development environment from Borland. Proceed as follows to open the import table of your C/C++ program:

1. Open the C/C++ program you require.
2. Open the import table. Use the following menu command from the Borland development environment:

   Tool ▸ Importing STEP 7 Symbols

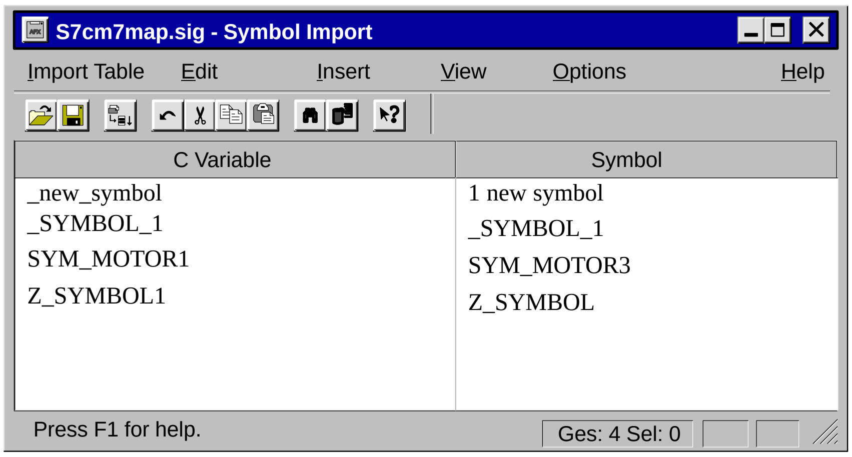
   Result: The window for the import table opens. (See Figure 4-1).

   If an import table with symbols already exists for a C/C++ project you are currently processing, the contents of the import table are displayed.

Supported Address Types

The symbol generator supports the following address types only:

- Process-image input
- Process-image output
- Peripheral input
- Peripheral output


S7cm7map.sig - Symbol Import

C Variable	Symbol
_new_symbol	1 new symbol
_SYMBOL_1	_SYMBOL_1
SYM_MOTOR1	SYM_MOTOR3
Z_SYMBOL1	Z_SYMBOL

Press F1 for help.

Ges: 4 Sel: 0

Figure 4-1 *Import Table*

User Interface

The window below the menu line contains a toolbar with symbols for frequently used menu commands. The lower window edge shows the status line displaying information about the selected menu command.

The import table primarily consists of two list boxes in which the selected symbols or the respective C variables are listed. You can change the column width of the individual boxes when you use the slider.

Symbol Column

The right list box in the import table displays the symbols stored in the symbol import file.

With the help of the symbol table dialog box (see Section 4.2), you can select the symbols you want to use in your C/C++ program from the symbol table of the relevant project. The selected symbols appear in the right list box of the import table.

C Variables Column

The appropriate C variable names are displayed in the left list box of the import table. Double C variable names are shown in color (red) in the import table, if this was activated under **Options** ▸ **Settings** (see Section 4.2). You can edit the C variable name in the list box.

4.2 Importing Symbols and Processing C Variables

Main Procedure

Proceed as follows to insert the desired symbols from the symbol table of your M7 program into the import table:

1. Open the import table for your program from the Borland IDE by using the **Tool ▸ Importing STEP 7 Symbols** menu command**.**
2. Set the type of symbol access (see below).
3. Call up the **Insert ▸ C variable** menu command in the import table window.

   Result: The selection dialog box opens up. You see the symbol names displayed, as well as the corresponding addresses and the data type.
4. Select the lines you want to transfer in the selection list.
5. Click the “Add” button. The selected symbols are transferred to the import table and appear in the symbol column. Automatically generated variable names which you can change if necessary appear in the C variable column.
6. Edit the C variables, if necessary.
7. Save the import table.
8. Generate the header files.

Setting Symbol Access

You must set the type of symbol access to be used before generating the header files with the symbol generator. You have two possibilities:

- Direct symbol access set: In this case you must use the M7 API function calls (see Table 4-2)
- Direct symbol access not set (as in earlier M7 ProC/C++ versions): In this case you must use the C functions for symbol import (see Appendix A).

Note

Please note that direct symbol access is not compatible with the previously used C functions for symbol import and vice versa. This means that the generated header files can be used only with the specific program you have set them for.

Setting symbol access incorrectly leads to errors when compiling the C program.

To set the type of symbol access, proceed as follows:

1. Select the menu command **Options** ▸ **Customize**. The “Customize” dialog box is opened.
2. Select the required option:
   – Activate direct symbol access if you want to access symbols quickly from the C program by calling the M7 API M7Loadxxx, M7LoadDirectxxx, M7Readxxx, M7Storexxx, and M7Writexxx functions or
   – Leave this option inactive if you want to continue using the C functions for symbol import in your C program (see Appendix A)
3. Confirm with “OK”.

Editing C Variables

Default C variables are generated automatically from the symbolic names before the symbols are inserted into the import table. The default C variable names satisfy the C syntax.

After you enter the symbols into the import table and have generated default C variable names, you can edit the variable names.

1. Select a symbol in the symbol column and press **F2**, or

   double-click the selected symbol, or

   select the **Edit** ▸ **Rename** menu command.
2. Edit the variable name directly in the *C Variable* column.

   The edited C variable names are then checked for correct syntax. A validity check then follows, if you chose this option with the **Options** ▸ **Customize** menu command.

Variable Names with Double Assignments

You can use the **Options** ▸ **Customize** menu command to determine what should happen when variable names with double assignments are found during the consistency check. There are two possibilities:

- Add a numerical index to make the variable name unique
- Identify double entries using a different color (red)

Variable Length

The maximum variable length is fixed at 128 characters.

If direct symbol access is activated, the maximum variable length is fixed at 53 characters.

Saving the Import Table

The import table can be saved after the C variable names have been entered into the desired form. You have the following possibilities:

- Select the **Import Table ▸ Save** menu command, click the relevant symbol in the toolbar, or
- Select the **Import Table ▸ Compile** menu command.

In both instances, the import table is saved under the S7CM7MAP.SIG file name in the current project path.

Generating Header Files

The symbol import editor generates several header files with the respective variable declarations, macros, and functions for the C variables saved in the import table.

To generate header files, select the **Import Table ▸ Compile** menu command. During the first generation, the following three header files are generated in the project directory under the preset names:

– CM7DEF.H – contains definitions of the data types used, the macros, and functions

– CM7DECL.H – declaration of all C variables from the import table

– CM7INIT.H – initialization of all C variables from the import table

The files are updated for all further compilation procedures.

These header files are integrated into the default C and C++ project as standard.

Processing Symbols

In addition to the possibility of selecting symbols from the symbol table and transferring the symbols to the import table, you can also:

- Process the symbols of the current symbol table, that is, you can change the following data: symbol name, address, data type, and comment.
- Create new symbols.
- Create links in the Symbol Editor for processing the entire symbol table.

To process symbols, select a symbol in the symbol column in the import table and select the **Edit ▸ Object Properties** menu command. The online help contains additional information.

4.3 Using Symbolic C Variables in the Program

Why Have Symbolic Names?

Using symbolic addresses makes your C application program more transparent as regards process signal processing and makes programming easier. In addition, when you change a slot assignment, you need only change the new address in the Symbol Editor and then compile the C program from the SIMATIC Manager.

Procedure

Proceed as follows to use symbolic names:

1. Use the Symbol Import Editor to help select the symbols you want to address in your C program from the symbol list. If necessary, change the preset C variable name.
2. Generate the relevant header files with the **Import Table ▶ Compile** menu command. This step is optional since the import table is compiled automatically together with the C program.
3. **Without direct symbol access:**
   Use the relevant C functions from the CM7DEF.H header file in your C application program to initialize, read, and write symbols. (See Table 4-1).

   With direct symbol access (see below).

C Functions for Symbol Import without Direct Symbol Access

The following functions are available for symbolic C variables:

Table 4-1 Functions for Symbolic C Variables

	Function	Description
1.	INIT_SYMBOL	Initializes all C variables from the import table
2.	LOAD-SYMBOL	Loads the value of a symbol from the process I/O or from the process image
3.	GET_SYMBOL	Provides the internal value of a C variable
4.	SET_SYMBOL	Sets the internal value of a C variable
5.	STORE_SYMBOL	Saves the value of a symbol to the process I/O or in the process image

Descriptions of the functions for handling symbolic C variables can be found in Appendix A.

Example

Chapter 6 explains the procedure using a sample program.

M7 API Functions for Direct Symbol Access

If you have activated direct symbol access, the symbols are converted into defines during compilation, so you can pass the symbol names directly as arguments to the M7 API functions.

Table 4-2 shows the arguments of the M7 API functions to which the imported symbol names can be assigned.

Table 4-2 Symbol Arguments of M7 API Functions

M7 API Function	Description	Symbol Argument
M7LoadBit	Load bit from process image	ByteOffset, BitOffset
M7LoadByte/Word/DWord	Load byte/word/double word from process image	ByteOffset
M7LoadDirect	Read I/O area directly	Addr
M7LoadDirectByte/Word/DWord	Read byte/word/double word directly from I/O	Addr
M7Read	Read S7 data area	Addr
M7ReadBit	Read bit from S7 object	ByteOffset, BitOffset
M7ReadByte/Word/DWord	Read byte/word/double word from S7 object	ByteOffset
M7StoreBit	Overwrite bit in process image	ByteOffset, BitOffset
M7StoreByte/Word/DWord	Overwrite byte/word/double word in process image	ByteOffset
M7StoreDirect	Write data directly to I/O area	Addr
M7StoreDirectByte/Word/DWord	Write byte/word/double word directly to I/O	Addr
M7Write	Write user data to S7 data area	Addr
M7WriteBit	Overwrite bit in S7 object	ByteOffset, BitOffset
M7WriteByte/Word/DWord	Overwrite byte/word/double word in S7 object	ByteOffset

For further information on these functions, refer to the Programming Manual and the Reference Manual “System Software for M7-300 and M7-400”.

Caution

When using direct symbol access, you must ensure that the M7 API functions match the address types used and that the correct arguments are passed to the functions, since no check is carried out.

In case of error, corrupted data may be written to the process image and to the peripheral outputs!

Examples:

Example 1:
Suppose that in STEP 7 the input bit 5.3 has been assigned the symbolic name “MOTOR” and that it has been imported to your C program. In this case, the Symbol Generator creates the following statement:

```
#define MOTOR 5,3
```

A call to ***M7LoadBit*** would be coded as follows:

```
BOOL test;
long ErrorCode;
test = M7LoadBit(M7IO_IN, MOTOR, &ErrorCode);
```

Example 2:
Suppose that in STEP 7 the output word 12 has been assigned the symbolic name “OUT_12” and that it has been imported to your C program. In this case, the Symbol Generator creates the following statement:

```
#define OUT_12 12
```

A call to ***M7StoreDirectWord*** would be coded as follows:

```
WORD value;
long ErrorCode;
value = 1;
ErrorCode = M7StoreDirectWord(OUT_12,value);
```

Testing Programs 5

Overview

You can use the Organon XDB386 debugger to test your C/C++ programs online on the M7 programmable controller.

What is Described in This Chapter?

This chapter describes how you call up the XDB386 debugger and which settings are necessary for M7 programs.

Debugger Documentation

The “Debugger for C Programs” manual contains a complete description of the functionality and operation of the XDB386 debugger.

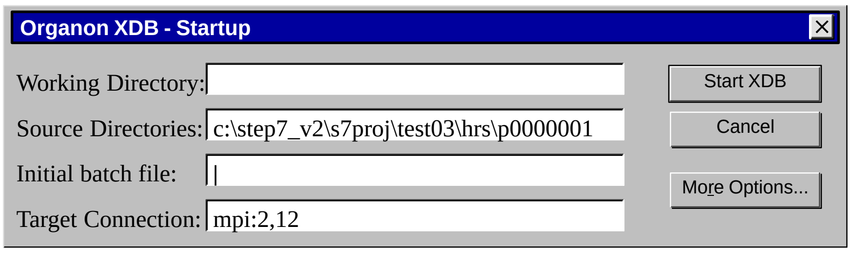
Chapter Overview

Section	Contents	Page
5.1	Starting the Debugger	5-2
5.2	Working with the Debugger	5-3

5.1 Starting the Debugger

Calling Up the Debugger

You call up the Organon XDB386 debugger from the main window of the Borland IDE with the **Tool** ▸ **STEP7 Debugger** menu command. The startup dialog box for the debugger appears.


Organon XDB - Startup

Working Directory:

Source Directories: c:\step7_v2\s7proj\test03\hrs\p0000001

Initial batch file: |

Target Connection: mpi:2,12

Setting the Start Options

The necessary settings in the startup dialog box are usually generated automatically. The fields have the following significance:

Table 5-1 Startup Options for the XDB386

Field	Entry
Working Directory	This field can remain empty. If you enter a directory here, you can enter the relative path to the working directory under **Source Directories**. The working directory is always searched to locate the source files and no longer needs to be specified under **Source Directories**.
Source Directories	The paths under which the source files are located are displayed here. You can expand the list, if necessary. A comma or space separates the individual entries.
Initial batch file	In addition to the startup.xdb initialization file that is always carried out automatically, you can also enter an additional startup file if necessary, such as with project-specific instructions.
Target connection	Communications connection to the M7 programmable controller, for example: MPI: mpi:2,12 or Adapt the entry to your type of communication as required.

These settings are applicable for all additional XDB386 calls from this C/C++ program and can be changed subsequently.

Starting the Debugger

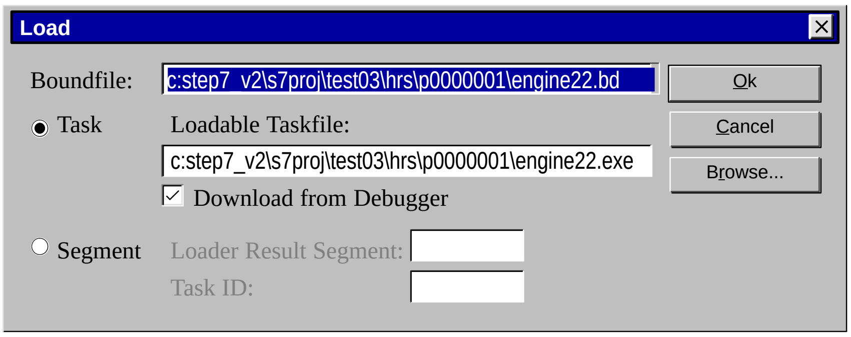
Select the “Start XDB” button in the startup dialog box to start the debugger. This starts the XDB386 debugger on the PC or programming device and the main window of the debugger opens.

5.2 Working with the Debugger

Loading the Programs

To load the programs to be tested, select the **File** ▶ **Load** command in the main window of the debugger. The "Load" dialog box opens. You then have two possibilities:

- To test a program which already exists on the M7 programmable controller, or
- To load a program directly from the programming device to the to the M7 programmable controller (download from debugger).



Load

Boundfile: c:step7_v2\s7proj\test03\hrs\p0000001\engine22.bd

[✓ Task] Loadable Taskfile: c:step7_v2\s7proj\test03\hrs\p0000001\engine22.exe

✓ Download from Debugger

Segment Loader Result Segment:

Task ID:

Figure 5-1 "Load" Dialog Box

Additional Procedures

After you load the program, you normally proceed as follows in the main window of the debugger:

1. Register the loaded task with the Debugger by activating the "set_task" button.
2. Let the task run through up to the "main" function by selecting the "go_main" button.

Refer to the "Debugger for C Programs" manual for additional information on working with the XDB386 debugger.

Activating / Deactivating the Debug Option

When you generate a C or C++ program for M7 RMOS32, the debug option is preset in the ***.ide** Borland project file. This means that debug information is stored in the executable program.

To turn the debug options on or off:

1. Call up the following menu command in the Borland window:

   Options ▸ Project ▸ Compiler ▸ Debugging

2. Activate or deactivate the following options:

   – Debug information in OBJ files and

   – Symbol display information in OBJ files:

3. Call up the **Options ▸ Project ▸ Linker ▸ General** menu command from the Borland IDE.

4. Activate or deactivate the “Include debug information” option.

After the development work has been concluded, you must compile the program without debug information to generate an efficient code.

Generating a Sample Application 6

Overview

This chapter explains the individual steps based on a simple example. This example presumes a simple configuration of a M7-300 system. Therefore, this example can be executed on almost every M7 programmable logic control system.

The aim of this chapter is to generate a sample program that reads eight input signals, outputs them to digital outputs, and lets the digital outputs flash at second intervals.

This chapter shows you the main procedure to use when creating applications for an automation computer with M7 ProC/C++.

Note

The example is without direct symbol access. It shows the use of the C functions for symbol import.

If you wish to use direct symbol access, you must modify the program accordingly using M7 API functions (see the “System Software for M7-300 and M7-400” Reference Manual and Programming Manual).

Chapter Overview

6.1 Overview

Introduction

For generating a C/C++ application on a M7 automation computer, we recommend individual, self-contained steps you execute in a specific order. Depending on the result of the various steps, it is necessary to repeat the one or the other step (recursive procedure).

Step Sequence

You should carry out the steps in the following order:

1. Create a project.
2. Configure and set parameters for the HW configuration.
3. Create an M7 program object.
4. Create symbols.
5. Create a C program for M7 RMOS32.
6. Select the required symbols and generate the respective C variables.
7. Edit, compile, and connect the application program.
8. Test the application on the M7 programmable controller with the debugger.
9. Transfer the application program to the M7 programmable controller.
10. Start the application program on the M7 programmable controller.

Step 2 is optional. Step 9 can also be carried out before Step 8. Steps 1 through 4 are described extensively in the *STEP 7 User Manual*. The steps are combined in the following under “Project Prerequisites.”

6.2 Prerequisites

Hardware Prerequisites

To carry out the individual steps of the exercise example, you will need the following minimum configuration for your M7 programmable controller. The M7-300 system must contain the following modules as a minimum:

- A power supply module, for example, the PS 307; 2A
- An M7-300 CPU 388-4 with the MSM 378 mass storage module
- A digital input/output module, for example, the SM 323; DI16/DO16 x 24 V DC

In addition, your programming device must be connected to the M7 programmable controller via MPI. The *STEP 7 User Manual* describes how to establish a connection and set the MPI addresses.

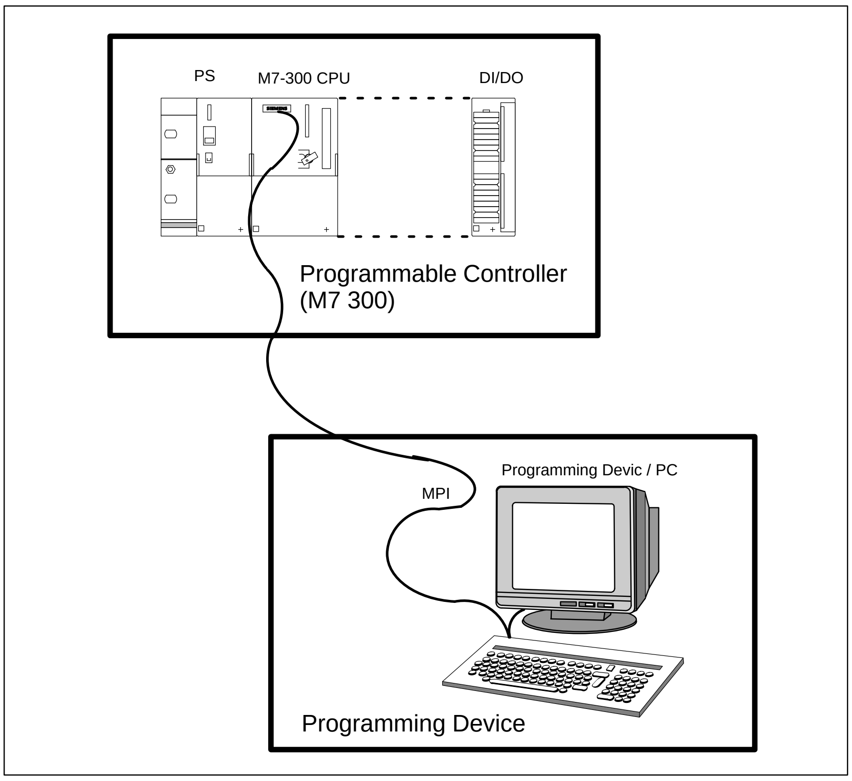


Figure 6-1 System Configuration for the Exercise Example

Software Prerequisites

You need the following software components on your programming device:

- STEP 7 Standard
- M7 SYS system software
- M7 ProC/C++
- Borland C++ development environment

Project Prerequisites

A project fulfilling the following conditions must be created on your programming device in the SIMATIC Manager:

- The following objects are created:
  - SIMATIC S7-300 station
  - Relevant programmable module (M7 CPU)
  - M7 program
- Configuring and assigning parameters for the hardware configuration are carried out. The start address for the digital input/output module is set to logical address 12.
- The symbols for your project are defined. The following entries are present in the symbol table for the sample program:
  - Test signal = PQB 12 with the following symbol data:

Input Field	**Entry**
Symbol name	Test Signal
Address	PQB 12 (peripheral output)
Data type:	BYTE

  - Input = PIB 12 with the following symbol data:

Input Field	**Entry**
Symbol name	Input
Address	PIB 12 (peripheral input)
Data type:	BYTE

The M7 RMOS32 (optional with MS-DOS/Windows) must be installed and must be able to run on your M7 programmable controller.

6.3 Creating an M7 C Program

Application

You assign a C program for the M7 programmable controller to the “M7 program” software object you have created.

Procedure

You proceed as follows to create a C program for the M7-300 CPU:

1. Select the M7 program object symbol in the project and select the **Insert** ▸ **M7 Software** ▸ **C Program** menu command.

   Result: A C program object is created that is visible in your project structure. The C program corresponds to a Borland C project. You can now process this object further with the Borland development environment.

2. Select the C program object. Select the **Edit** ▸ **Object Properties** menu command. The properties dialog box is opened.

3. Change the name of the software component in the *test program*. Exit the dialog box using the “OK” button.

   Result: The C program object is displayed in your project structure under the name of *test program*. All necessary header files are already integrated.

6.4 Importing Symbols into the C Program

Application

You import the symbols for the digital inputs and outputs, that is, you assign each of them a C variable to conveniently be able to access the process I/Os from your program.

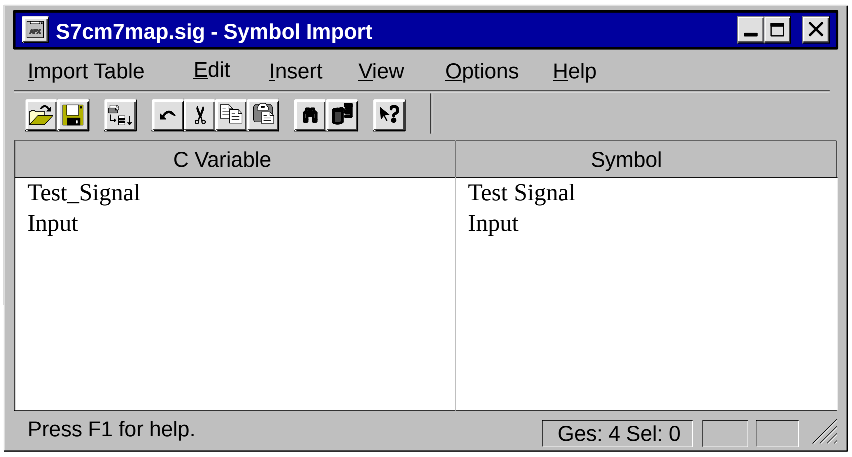
Defining Symbols

To import symbols into your C program:

1. Mark the *test program* object and select the **Edit ▸ Open Object** menu command. You can also simply double-click the software component. The Borland window for the selected software component opens up. (See Figure 3-1).
2. Select the **Tool ▸ Importing STEP 7 Symbols** menu command. The symbol import editor window opens up.
3. Select the **Options ▸ Customize** menu command and deactivate the "Direct Symbol Access" check box.
4. Select the **Insert ▸ C Variable** menu command as your next step.

   Result: A selection dialog box opens up. You see the symbols, the relevant addresses, and data type displayed.
5. Select the "Test Signal" and "input" entries in the selection list. Select the "Add" button. The selected symbol information is transferred to the import table.

   The *Test_Signal and input* C variable names automatically generated from the symbol names are displayed in the C variable column.



C Variable	Symbol
Test_Signal	Test Signal
Input	Input

Figure 6-2 Import Table for the Sample Application

6. Select the **Import Table ▸ Save** menu command. The import table with the symbol name or C variable name is stored.

6.5 Editing, Compiling, and Linking the C Program

Editing the C Program

You can edit the C program in the next step.

1. First close the import table so that only the Borland window of the generated C program component is open.
2. Supplement your source code based on the example in Figure 6-3. The include instructions are already present.

```
#include <memory.h>
#include <stdlib.h>

#include “m7api.h”
#include “rmapi.h”

#include “cm7def.h”    // Macro definition and structure definitions, processing
                          functions
#include “cm7decl.h”   // C variable declaration
#include “cm7init.h”   // C variable initialization

void main()
{
  BYTE byinput;
  if(M7InitAPI() != M7SUCCESS)
  {
         exit(1);
  }
  // Initialize C variable
  INIT_SYMBOL();
  while (1)
  {
         LOAD_SYMBOL(&input);
         byinput = (BYTE)GET_SYMBOL(input);
         SET_SYMBOL(&Test_Signal, byinput);
         STORE_SYMBOL(Test_Signal);
         RmPauseTask(1000);
         SET_SYMBOL(&Test_Signal, 0);
         STORE_SYMBOL(Test_Signal);
         RmPauseTask(1000);
  }
}
```

Figure 6-3 M7 RMOS32 Sample Program in the Borland Editor

Compiling and Linking the C Program

After you edit the above program, you can compile and link the program. You have the following possibilities available:

- Select the **Project ▸ Build all** menu command. This command recompiles and links the entire Borland C project.
- Select the **Project ▸ Make all** menu command. The last modification times of the individual files are analyzed and the relevant compilation and link sequences are carried out.

6.6 Debugging the Application Program

Application

With the M7 debugger for C programs, you can load and carry out a remote test of your sample program via Download in the M7 programmable controller.

Procedure

To load and test the sample program on the M7 programmable controller:

1. Call up the Borland IDE in the main window with the **Tool ▸ STEP 7 Debugger** menu command. A startup dialog box for entering the source directory and the target connection opens up.
2. Accept the preset specifications.
3. To start the debugger, select the “Start XDB” button in the startup dialog box. This starts the XDB386 debugger on the M7 programmable controller and the debugger main window opens up.
4. Select with the **File ▸ Load** menu command in the main window of the debugger. The “Load” dialog box opens up.
5. Accept the preset task name and bound file name. Activate the “Download” option to load the program directly from the programming device to the M7 programmable controller. Select the “OK” button.
6. Generate a task by selecting the “set_task” button.
7. Let the task run up to the “main” function by selecting the “go_main” button.

The relevant commands can be used to set breakpoints and display variable contents. (Refer to the *Description of the M7 Debugger* manual.)

Note

After you conclude the development work, you should compile the program without debug information to generate an efficient code. (See Section 5.2.)

During debugging, the program is loaded into the main memory of the M7 programmable controller. However, the program is not transferred to the hard disk drive. To transfer the sample program to the hard disk drive on the M7 programmable controller, carry out the procedure described in Section 6.7.

6.7 Transferring the Application Program to the M7 Programmable Controller and Starting the Application Program

Application

After you have generated the executable program for your M7 CPU, you can transfer it to the file system on the M7 programmable controller.

Prerequisite: The M7 programmable controller must be started and there must be a functioning MPI connection between the M7 programmable controller and the programming device to transfer the program.

Procedure

To transfer the application program to the M7 programmable controller:

1. Select the “M7 program” object in your project window. Select the **PLC ▸ Manage M7 System** menu command. The tabbed dialog box for selecting operating systems and programs appears.
2. Open the “Program” tab.
3. Select:
   - “MPI/RFS” as the target medium
   - The first free drive in the list, for example N: as the local drive. Select “C” as the partner drive, which is the hard disk drive of the M7-300.
4. Select your C program from the selection field and click on the “Install” button.

   The program is transferred to the M7 programmable controller and is entered automatically in the \etc\inittab file so that the program starts automatically during the next system power up.
5. Then carry out a reset and boot the operating system on the M7 programmable controller.

   Result: The application starts. The I12 input signals are displayed on the Q12 outputs and flash at second intervals.

C Functions for Symbol Import

A

Overview

This chapter provides descriptions of the C functions for processing symbolic C variables.

Note

When using these functions in your C program, direct symbol access must be deactivated with the menu command **Options** ▸ **Customize** (see Section 4.2).

Functions

The following functions are available for symbol import:

Section	Function	Description	Page
A.1	GET_SYMBOL	Returns the internal value of a C variable.	A-2
A.2	INIT_SYMBOL	Initializes all C variables from the import table	A-3
A.3	LOAD_SYMBOL	Loads the value of a symbol from the process I/O or from the process image	A-4
A.4	SET_SYMBOL	Sets the internal value for a C variable	A-5
A.5	STORE_SYMBOL	Saves the value of a symbol to the process I/O or in the process image	A-6

A.1 GET_SYMBOL

Function

Provides the internal value of a C variable

Syntax

#include <memory.h>
#include <cm7def.h>

DWORD GET_SYMBOL (**SYMBOL** *SymName*);

Parameter

Parameter Name	Significance
SymName	C variable name

Description

This function returns the value of the C variable structure as a return value. The return value must be converted to the relevant data type, for example:

```
int   iinput
iinput = (int) GET_SYMBOL (Cvariable) ;
```

Call up the function after the LOAD_SYMBOL function is called up.

Return Value

C variable structure value

See Also

LOAD_SYMBOL, SET_SYMBOL

A.2 INIT_SYMBOL

Function	**Initializes all C variables from the import table**

Syntax	**#include <cm7def.h>**
	#include <cm7init.h>
	void INIT_SYMBOL();

Description	This function initializes all C variables from the import table. You must call up this function in the application program before you use the symbolic C variables.
See Also	STORE_SYMBOL, SET_SYMBOL, LOAD_SYMBOL, GET_SYMBOL

A.3 LOAD_SYMBOL

Function

Loads the symbol value from the process I/O or process image

Syntax

#include <m7api.h>
#include <cm7def.h>

M7ERR_CODE LOAD_SYMBOL (SYMBOL ***pSymbol*****);**

Parameter

Parameter Name	Significance
pSymbol	C variable address

Description

This function loads the I/O signal assigned to the C variable from the process image or from the process I/O. To read the value of the C variable structure, you then call up the GET_SYMBOL function.

Return Value

=M7SUCCESS	The function was carried out successfully.
<M7SUCCESS	An error has occurred.

Error Codes

Error Code	Significance
M7E_BSY	Local bus is busy, only for direct reading from process I/O
M7E_CMD	Local bus command error, only for direct reading from process I/O
M7E_HWFAULT	General hardware fault, only for direct reading from process I/O
M7E_PAR	Addressed module does not exist (process I/O), a parameter error (process image) occured or the symbol references an output
M7E_PARITY	Local bus parity error, only for direct reading from process I/O
M7E_QVZ	Local bus timeout, only for direct reading from process I/O
M7E_DP_SLAVE_STATE	Device not ready for data communication, only for direct reading from process I/O

See Also

STORE_SYMBOL, GET_SYMBOL

A.4 SET_SYMBOL

Function **Sets the internal value of a C variable**

Syntax

#include <memory.h>
#include <cm7def.h>

void SET_SYMBOL (
    SYMBOL **pSymName*,
    DWORD *value*);

Parameters

Parameter Name	Significance
pSymName	C variable address
value	C variable value to be written

Description This function writes the new value in the specified C variable structure.

See Also STORE_SYMBOL, GET_SYMBOL

A.5 STORE_SYMBOL

Function

Saves a symbol's state in the process I/O or process image

Syntax

#include <m7api.h>
#include <cm7def.h>

M7ERR_CODE STORE_SYMBOL (**SYMBOL** *symbol*);

Parameter

Parameter Name	Significance
symbol	C variable name

Description

This function saves the symbol assigned to the C variable in the process image or on the process I/O. You must first set the internal value of the relevant C variable with the SET_SYMBOL function.

Return Value

=M7SUCCESS	The function was carried out successfully.
<M7SUCCESS	An error has occurred.

Error Codes

Error Code	Significance
M7E_BSY	Local bus is busy, only for direct writing into process I/O
M7E_CMD	Local bus command error, only for direct writing into process I/O
M7E_HWFAULT	General hardware fault, only for direct writing into process I/O
M7E_PAR	Parameter error or the symbol references an input
M7E_PARITY	Local bus parity error on the local bus, only for direct writing into process I/O
M7E_QVZ	Local bus timeout, only for direct writing into process I/O
M7E_DP_SLAVE_STATE	Device not ready for data communication, only for direct writing into process I/O

See Also

LOAD_SYMBOL, SET_SYMBOL

Index

Siemens AG
AUT E 146

Östliche Rheinbrückenstr. 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	❐ Pharmaceutical
❐ Chemical	❐ Plastic
❐ Electrical Machinery	❐ Pulp and Paper
❐ Food	❐ Textiles
❐ Instrument and Control	❐ Transportation
❐ Nonelectrical Machinery	❐ Other _ _ _ _ _ _ _ _ _ _ _
❐ Petrochemical	

Remarks Form

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

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

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

Additional comments:

_ _
_ _
_ _
_ _
_ _
_ _
_ _
_ _
_ _
_ _
_ _
_ _