

SIMATIC

S7-200 Programmable Controller, CPU 210

System Manual

Preface, Contents

Installing the CPU 210

1

Installing and Using the
STEP 7-Micro/WIN Version 2.0
Software

2

Getting Started with a Sample
Program

3

Basic Concepts for
Programming the CPU 210

4

Instruction Set

5

Appendix

CPU 210 Data Sheets

A

Special Memory (SM)

B

Error Handling and Error Codes

C

Converting STEP 7-Micro/DOS
Files to STEP 7-Micro/WIN Files

D

Execution Times for STL
Instructions

E

CPU 210 Order Numbers

F

Index

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.

Copyright © Siemens SE&A 1997 All rights reserved

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

Siemens Energy & Automation, Inc.
3333 Old Milton Parkway
Alpharetta, GA 30202

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.

Technical data subject to change.
© Siemens SE&A 1997

Preface

Purpose

The CPU 210 is an addition to the S7-200 series micro-programmable logic controllers (Micro PLCs). Its compact design, low cost, and powerful instruction set make the CPU 210 a perfect solution for controlling small applications. The selection of voltage options provides you with the flexibility you need to solve your automation problems.

The *SIMATIC S7-200 CPU 210 Programmable Controller System Manual* provides information about installing and programming the CPU 210 and the program development station (PDS 210). This manual also includes descriptions and examples for the programming instructions, typical execution times for the instructions, and the data sheets for the CPU 210 and related equipment.

Audience

This manual is designed for engineers, programmers, installers, and electricians who have a general knowledge of programmable logic controllers.

Scope of the Manual

The information contained in this manual pertains in particular to the following products:

- CPU 210 and the PDS 210
- STEP 7-Micro/WIN version 2.0 programming software

How to Use This Manual

If you are a first-time (novice) user of S7-200 Micro PLCs, you should read the entire manual. If you are an experienced user, refer to the table of contents or index to find specific information.

The manual is organized in the following topics:

- “Installing the CPU 210” (Chapter 1) provides an overview of some of the features of the equipment and the procedures, dimensions, and basic guidelines for installing the CPU 210.
- “Installing and Using the STEP 7-Micro/WIN Version 2.0 Software” (Chapter 2) describes how to install the programming software. It also provides a basic explanation about the features of the software.
- “Getting Started with a Sample Program” (Chapter 3) helps you enter a sample program, using the STEP 7-Micro/WIN software.
- “Basic Concepts for Programming the CPU 210” (Chapter 4) provides information about how the CPU 210 processes data and executes a program.
- “Instruction Set” (Chapter 5) provides explanations and examples of the programming instructions used by the CPU 210.

Additional information (such as the equipment data sheets, error code descriptions and execution times) are provided in the appendices.

Additional Assistance

For assistance in answering technical questions, for training on this product, or for ordering, contact your Siemens distributor or sales office.

Contents

1	Installing the S7-200 CPU 210	
1.1	Product Overview	1-2
	Equipment Requirements	1-2
	Features of the CPU 210	1-3
1.2	Pre-installation Considerations	1-4
	Installation Configuration	1-4
	Clearance Requirements for Installing a CPU 210	1-4
	DIN Rail Requirements	1-5
	Panel-Mounting Dimensions	1-5
1.3	Installing a CPU 210	1-6
	Mounting a CPU 210 on a Panel	1-6
	Installing a CPU 210 on a DIN Rail	1-6
	Installing a CPU 210 in a Panel Box	1-7
1.4	Installing the Field Wiring	1-8
	General Guidelines	1-8
	Grounding and Circuit Referencing Guidelines for Using Isolated Circuits	1-9
	Using the Optional Field Wiring Connector	1-10
	Guidelines for AC Installation	1-10
	Guidelines for DC Installation	1-10
1.5	Using Suppression Circuits	1-12
	Protecting DC Transistors	1-12
	Protecting Relays Controlling DC Power	1-12
2	Installing and Using the STEP 7-Micro/WIN Version 2.0 Software	
2.1	Installing the STEP 7-Micro/WIN Version 2.0 Software	2-2
	Pre-installation Instructions	2-2
	Installation Instructions for Windows 3.1	2-2
	Installation Instructions for Windows 95	2-2
	Troubleshooting the Installation	2-2
2.2	Establishing Communication with the PDS 210	2-3
	Connecting Your Computer to the PDS 210 for PPI Communications	2-3
	Setting Up the Communications Parameters	2-4
2.3	Configuring the Preferences for STEP 7-Micro/WIN	2-5
2.4	Creating and Saving a Project	2-6
	Creating a New Project	2-6
	Saving a Project	2-6

2.5	Creating a Program	2-7
	Entering Your Program in Ladder	2-7
	Entering Your Program in Statement List	2-8
	Compiling the Program	2-8
	Viewing a Program in Ladder or Statement List	2-9
2.6	Downloading A Program	2-10
	Downloading the Program to the PDS 210	2-10
	Copying Your Program to the Memory Cartridge	2-11
	Transporting the Program to the CPU 210	2-11
2.7	Using Symbolic Addressing	2-13
	Guidelines for Entering Symbolic Addresses	2-13
	Starting the Symbol Table Editor	2-13
	Editing Functions within the Symbol Table	2-14
	Sorting Table Entries	2-14
	Displaying the Symbolic Addresses	2-14
2.8	Using the Status Chart	2-15
	Reading and Writing Variables with the Status Chart	2-15
	Editing Addresses	2-15
2.9	Debugging and Monitoring Your Program	2-16
	Using Single/Multiple Scans to Monitor Your Program	2-16
	Displaying the Status of the Program in Ladder Logic	2-16
2.10	Error Handling for the PDS 210	2-17
	Responding to Fatal Errors	2-17
	Responding to Non-Fatal Errors	2-18
3	Getting Started with a Sample Program	
3.1	Defining the Requirements for the Application Example	3-2
	Defining the Inputs and Outputs for the Application	3-2
	Creating Symbolic Names for the Elements of the Program	3-2
3.2	Designing the Control Logic	3-4
	Defining the Operation of the Program	3-4
	Designing the Control Logic for Arming and Disarming the System	3-6
	Designing the Control Logic for Turning On the Low-Level Alert Notification	3-7
	Designing the Control Logic for Turning On the Alarm and Modem Dialer	3-8
3.3	Putting the Control Logic into a Program	3-9
3.4	Creating a Project with STEP 7-Micro/WIN	3-13
3.5	Creating a Symbol Table	3-14
	Entering the Symbol Names	3-14
3.6	Entering the Program	3-15
	Programming with Symbolic Addresses	3-15
	Using the Ladder Editor to Enter the Program	3-15
	Compiling the Program	3-21
	Saving the Sample Program	3-21
3.7	Creating a Status Chart	3-22
	Building a Status Chart	3-22

3.8	Downloading and Monitoring the Sample Program	3-23
	Downloading the Project to the PDS 210	3-23
	Using the Ladder Editor to Monitor the Status of the Program	3-23
	Using the Status Chart to Monitor and Modify the Current Values of the Program	3-24
3.9	Modifying the Sample Program	3-25
	Creating the Blink Patterns for the LED	3-25
	Turning the LED On and Off	3-26
4	Basic Concepts for Programming the CPU 210	
4.1	Guidelines for Designing a Micro PLC System	4-2
	Partitioning Your Process or Machine	4-2
	Creating the Functional Specifications	4-2
	Designing the Safety Circuits	4-3
	Specifying the Operator Stations	4-3
	Creating the PLC Configuration Drawings	4-3
	Creating a List of Symbolic Names	4-3
4.2	Concepts for Creating a Program	4-4
	Relating the Program to Inputs and Outputs	4-4
	Accessing Data in the Memory Areas	4-4
	Organizing the Program	4-5
4.3	Understanding the Scan Cycle of the CPU 210	4-6
	Understanding the Basic Scan Cycle of the CPU 210	4-6
	Understanding the Basic Scan Cycle of the PDS 210	4-7
	Using the Debug Option to Specify the Number of Scans	4-8
4.4	Understanding the Programming Languages	4-9
	Understanding the Basic Elements of Ladder	4-9
	Understanding the Statement List Instructions	4-10
4.5	Understanding the Addresses of the Memory Areas	4-11
	Using the Memory Address to Access Data	4-11
	Addressing the Input Image Register (I)	4-12
	Addressing the Outputs (Q)	4-12
	Addressing the Bit Memory (M) Area	4-12
	Addressing the Special Memory (SM) Bits	4-12
	Addressing the Timer (T) Memory Area	4-13
	Addressing the Counter (C) Memory Area	4-13
	Using Constant Values	4-13
4.6	Sample Program Using an Interrupt Routine	4-14
4.7	Using the Analog Adjustment Potentiometer	4-16
5	Instruction Set	
5.1	Valid Ranges for the CPU 210 and PDS 210	5-2
	Valid Operand Ranges	5-2
5.2	Contact Instructions	5-3
	Standard Contacts	5-3
	Not	5-3
	Positive, Negative Transition	5-3
	Compare Word Integer	5-4
	Contact Examples	5-4
	Output	5-5
	Set, Reset	5-5

	Output Example	5-5
5.4	Timer Instructions	5-6
	On-Delay Timer	5-6
	Understanding How the CPU 210 Updates the Timers	5-6
	Timer Example	5-7
5.5	Counter Instructions	5-8
	Up/Down Counter	5-8
	Counter Example	5-8
5.6	Increment and Decrement Instructions	5-9
	Increment Word, Decrement Word	5-9
	Increment, Decrement Example	5-9
5.7	Move Instruction	5-10
	Move Word	5-10
	Move Examples	5-10
5.8	Program Control Instructions	5-11
	END	5-11
	Watchdog Reset	5-11
	Considerations for Using the WDR Instruction to Reset the Watchdog Timer	5-11
	END and WDR Example	5-12
	Jump to Label, Label	5-12
	Jump to Label Example	5-12
5.9	Logic Stack Instructions	5-13
	And Load	5-13
	Or Load	5-13
	Logic Stack Example	5-13
5.10	Interrupt Instructions	5-14
	Interrupt Routine, Return from Interrupt Routine	5-14
	Enable Interrupt, Disable Interrupt	5-14
	Guidelines and Restrictions for Using the Interrupt Routine	5-15
	Sharing Data Between the Main Program and the Interrupt Routine	5-15
	Interrupt Example	5-16
A	CPU 210 Data Sheets	
A.1	General Technical Specifications	A-2
A.2	CPU 210 DC Power Supply, 24 VDC Inputs, 24 VDC Outputs	A-4
A.3	CPU 210 AC Power Supply, 24 VDC Inputs, Relay Outputs	A-6
A.4	CPU 210 AC Power Supply, AC Inputs, Relay Outputs	A-8
A.5	PDS 210 AC Power Supply, DC Inputs, Relay Outputs	A-10
A.6	Memory Cartridge 8K x 8	A-12
A.7	Memory Cartridge 16K x 8	A-13
A.8	PC/PPI Cable	A-14
A.9	DC Input Simulator	A-15

- B Special Memory (SM)**
- C Error Handling and Error Codes**
- D Converting STEP 7-Micro/DOS Files to STEP 7-Micro/WIN Files**
- E Execution Times for STL Instructions**
- F CPU 210 Order Numbers**

Index

Installing the S7-200 CPU 210

The S7-200 CPU 210 is one of the S7-200 series of micro-programmable logic controllers (Micro PLCs) that can control a variety of automation applications. Figure 1-1 shows an S7-200 CPU 210. The compact design and low cost of the CPU 210 make a perfect solution for controlling small applications. In addition, the variety of input and output voltages provides you with the flexibility you need to solve your automation problems with the maintenance-free operation of the CPU 210.

The CPU 210 is easy to install. You can use the mounting holes to attach the module to a panel, or you can use the built-in DIN clips to mount the module onto a DIN rail. The small size of the CPU 210 allows you to make efficient use of space.

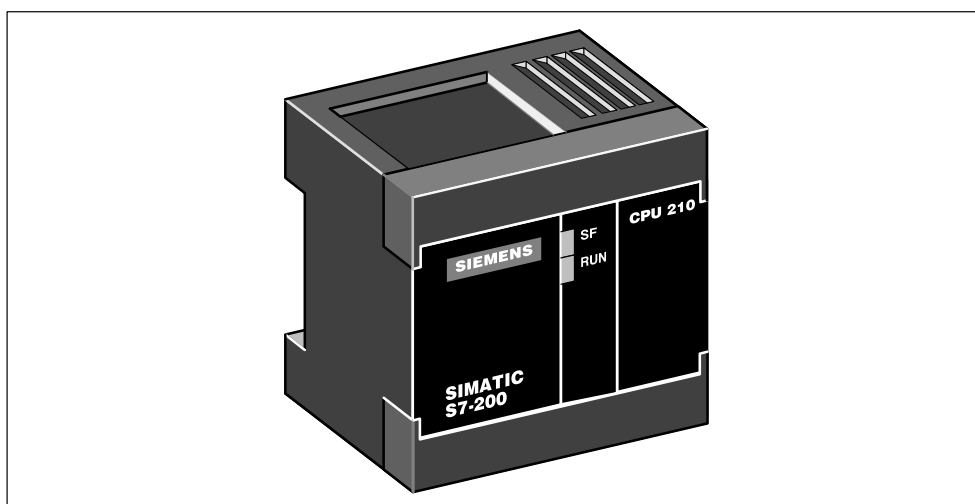


Figure 1-1 S7-200 CPU 210

Chapter Overview

Section	Description	Page
1.1	Product Overview	1-2
1.2	Pre-installation Considerations	1-4
1.3	Installing a CPU 210	1-6
1.4	Installing the Field Wiring	1-8
1.5	Using Suppression Circuits	1-12

1.1 Product Overview

The CPU 210 combines a central processing unit (CPU), power supply, and discrete I/O points into a compact, stand-alone device.

- The CPU executes the program and stores the data for controlling the automation task or process.
- The inputs and outputs are the system control points: the inputs monitor the signals from the field devices (such as sensors and switches), and the outputs control pumps, motors, or other devices in your process.
- Status lights provide visual information about the CPU mode (RUN) or whether a system fault (SF) has been detected.

Equipment Requirements

As shown in Figure 1-2, you use the STEP 7-Micro/WIN programming software with a program development station (the PDS 210) to create and to test your program. The final program is then loaded onto a memory cartridge, which is then installed in the CPU 210. You need the following equipment to create programs for the CPU 210:

- Personal computer (PC) running the STEP 7-Micro/WIN programming software. Refer to Chapter 2 for the requirements for installing the STEP 7-Micro/WIN software.
- Program development station (PDS 210).
- PC/PPI communications cable.
- Memory cartridge for transferring the program to the CPU 210.

Refer to the data sheets in Appendix A for order numbers and other specifications of this equipment.

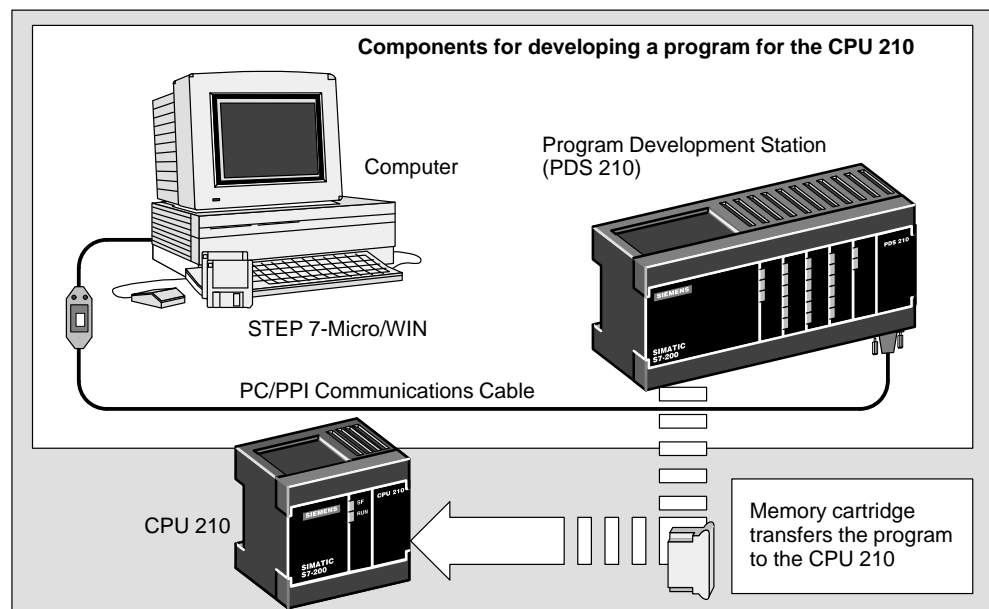


Figure 1-2 Components of a CPU 210 Micro PLC System

Features of the CPU 210

The CPU 210 is an integral part of the S7-200 family of Micro PLCs. Table 1-1 provides a summary of the major features of the CPU 210.

Table 1-1 Features of the CPU 210

Feature		CPU 210
Physical Size (length x width x depth)		90 x 80 x 62 mm
Memory cartridge for downloading the program		Yes
Memory	Program size	256 words
	Storage type	EEPROM
	Internal memory	48 bits (3 words)
Inputs/Outputs (I/O)	Local inputs	4 digital inputs
	Local outputs	4 digital outputs
	Expansion I/O	No
	DC Input delay filter	15 ms
	AC Input delay filter	55 ms
	Sink/Source Inputs (DC)	Yes
Instructions (36 total)	Boolean execution speed	95 μ s/instruction
	On-Delay Timers	4
	Resolution	100 ms
	Up/Down Counters	4
	Current value saved on power down	Yes
	Jump / Label	Yes
Additional Features	Analog adjustment potentiometers	1
	Hardware Input Interrupts	1
	Interrupt response	20 μ s on, 40 μ s off

1.2 Pre-installation Considerations

Installation Configuration

As shown in Figure 1-3, you can install a CPU 210 either on a panel or on a DIN rail. You can mount the CPU 210 either horizontally or vertically.

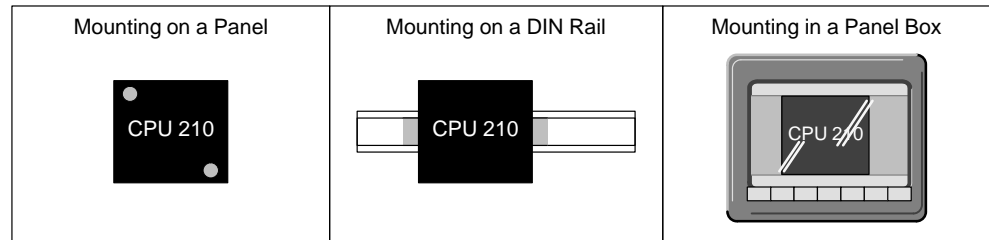


Figure 1-3 Mounting Configurations

Clearance Requirements for Installing a CPU 210

Use the following guidelines as you plan your installation:

- The CPU 210 is designed for natural convection cooling. You must provide a clearance of at least 25 mm (1 inch), both above and below the units, for proper cooling. See Figure 1-4. Continuous operation of all electronic products at maximum ambient temperature and load will reduce their life.
- If you are installing a CPU 210 on a panel, you must allow 75 mm (2.9 inches) for the minimum panel depth. See Figure 1-4.
- Be sure to allow enough space in your mounting design to accommodate the I/O wiring connections.

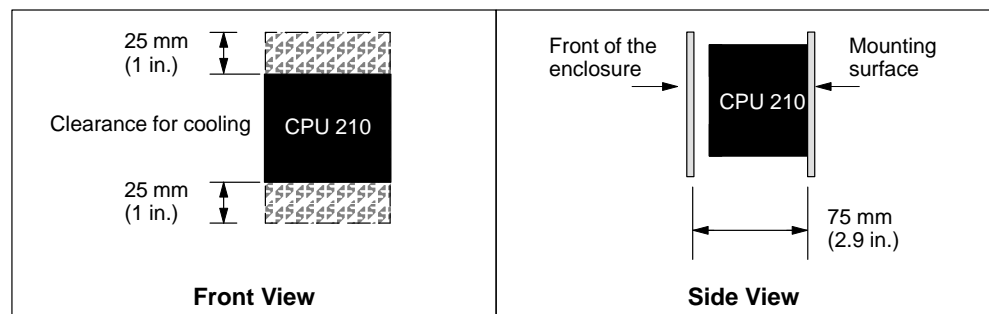


Figure 1-4 Clearance Requirements for Installing a CPU 210

DIN Rail Requirements

The CPU 210 can be installed on a standard DIN rail (DIN EN 50 022). Figure 1-5 shows the dimensions for this DIN rail.

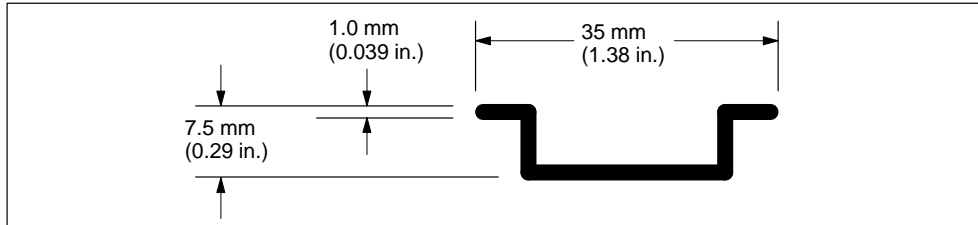


Figure 1-5 DIN Rail Dimensions

Panel-Mounting Dimensions

The CPU 210 and the PDS 210 include mounting holes to facilitate installation on panels. Figure 1-6 provides the mounting dimensions.

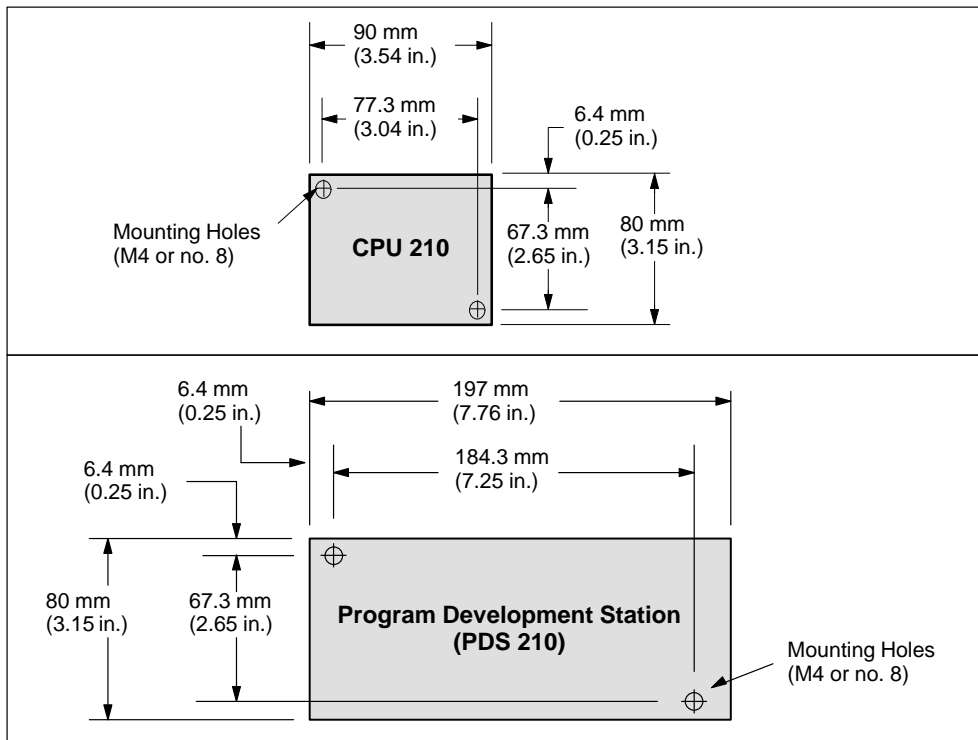


Figure 1-6 Mounting Dimensions for the CPU 210 and PDS 210

1.3 Installing a CPU 210



Warning

Failure to disable all power to the CPU 210 and related equipment during installation or removal procedures may result in death or serious personal injury, and/or damage to equipment.

Disable all power to the CPU 210 and related equipment before installation or removal.

Always follow appropriate safety precautions and ensure that power to the CPU 210 is disabled before installation.

Mounting a CPU 210 on a Panel

To install a CPU 210 on a panel, follow these steps:

1. Locate, drill, and tap the mounting holes for DIN M4 or American Standard number 8 screws. Refer to Section 1.2 for mounting dimensions and other considerations.
2. Secure the CPU 210 onto the panel, using DIN M4 or American Standard number 8 screws.

Installing a CPU 210 on a DIN Rail

To install a CPU 210 on a DIN rail (as shown in Figure 1-7), follow these steps:

1. Secure the DIN rail every 75 mm (approximately 3 inches) to the mounting panel.
2. Snap open the DIN clip (located on the bottom of the CPU 210) and hook the back of the module onto the DIN rail.
3. Snap the DIN clip closed, carefully checking to ensure that the DIN clip fastened the module securely onto the rail.

Note

Modules in an environment with high vibration potential or modules that have been installed in a vertical position may require DIN Rail Stops.

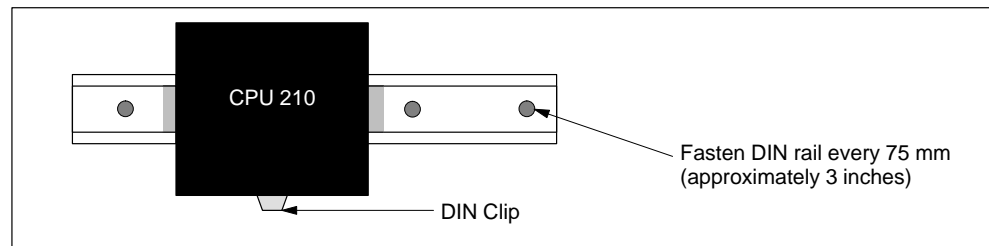


Figure 1-7 Installing a CPU 210 on a DIN Rail

Installing a CPU 210 in a Panel Box

To install a CPU 210 in a panel box, follow these steps:

1. Open one of the I/O access covers on the CPU 210. As shown in Figure 1-8, remove the access cover by gently pressing against the access cover until the hinges spring free. Repeat this procedure for the other access cover.

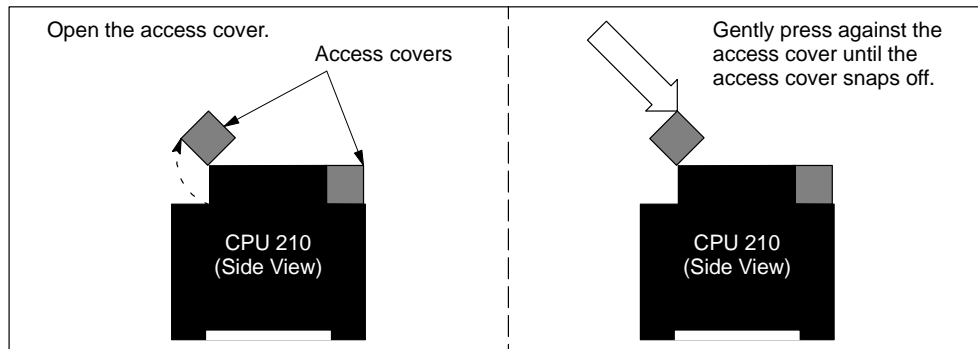


Figure 1-8 Removing the Access Covers from the CPU 210

2. Snap open the DIN clip (located on the bottom of the module).
3. Open the panel box and hook the back of the module onto the DIN rail. See Figure 1-9.
4. Snap the DIN clip closed, carefully checking to ensure that the DIN clip fastened the module securely onto the rail.

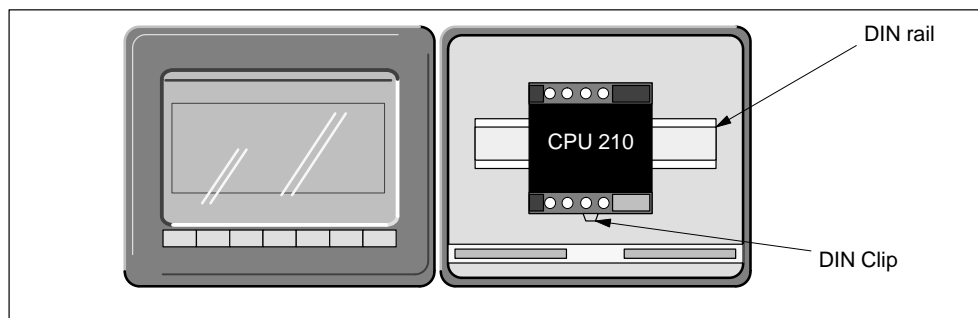


Figure 1-9 Installing the CPU 210 in a Panel Box

1.4 Installing the Field Wiring



Warning

Failure to disable all power to the CPU 210 and related equipment during installation or removal procedures may result in death or serious personal injury, and/or damage to equipment.

Disable all power to the CPU 210 and related equipment before installing or removing field wiring.

Always follow appropriate safety precautions and ensure that power to the CPU 210 is disabled before installing field wiring.

General Guidelines

The following items are general guidelines for designing the installation and wiring of your S7-200 CPU 210:

- Ensure that you follow all applicable electrical codes when wiring the CPU 210. Install and operate all equipment according to all applicable national and local standards. Contact your local authorities to determine which codes and standards apply to your specific case.
- Always use the proper wire size that will carry the required current. The CPU 210 accepts wire sizes from 1.50 to 0.50 mm² (14 to 22 AWG).
- Ensure that you do not over-tighten the connector screws. The maximum torque is 0.56 N-m (5 inches-pounds).
- Always use the shortest wire possible (maximum 500 meters shielded, 300 meters unshielded). Wiring should be run in pairs, with a neutral or common wire paired with a hot or signal-carrying wire.
- Separate AC wiring and high-energy, rapidly switched DC wiring from low-energy signal wiring.
- Properly identify and route the wiring to the CPU 210, using strain relief for the wiring as required. For more information about identifying the terminals, see the data sheets in Appendix A.
- Install appropriate surge suppression devices for wiring that is subject to lightning surges.
- External power should not be applied to an output load in parallel with a DC output point. This may cause reverse current through the output, unless a diode or other barrier is provided in the installation.



Warning

Control devices can fail in an unsafe condition, resulting in unexpected operation of controlled equipment.

Such unexpected action could result in death or serious personal injury, and/or equipment damage.

Consider using an emergency stop function, electromechanical overrides, or other redundant safeguards that are independent of the programmable controller.

Grounding and Circuit Referencing Guidelines for Using Isolated Circuits

The following items are grounding and circuit guidelines for using isolated circuits:

- You should identify the reference point (0 voltage reference) for each circuit in the installation, and the points at which circuits with possible different references can connect together. Such connections can result in unwanted current flows that can cause logic errors or damage circuits. A common cause of different reference potentials is grounds which are physically separated by long distances. When devices with widely separated grounds are connected with a sensor cable, unexpected currents can flow through the circuit created by the cable and the ground. Even over short distances, load currents of heavy machinery can cause differences in ground potential or directly induce unwanted currents by electromagnetic induction. Power supplies that are improperly referenced with respect to each other can cause damaging currents to flow between their associated circuits.
- The CPU 210 includes isolation boundaries at certain points to help prevent unwanted current flows in your installation. When you plan your installation, you should consider where these isolation boundaries are, and where they are not provided. You should also consider the isolation boundaries in associated power supplies and other equipment, and where all associated power supplies have their reference points.
- You should choose your ground reference points and use the isolation boundaries provided to interrupt unneeded circuit loops that could allow unwanted currents to flow. Remember to consider temporary connections which may introduce a new circuit reference, such as the connection of a programming device to the CPU.
- When locating grounds, you must also consider safety grounding requirements and the proper operation of protective interrupting devices.

The following descriptions are an introduction to general isolation characteristics of the CPU 210, but some features may be different on specific products. Consult the data sheet in Appendix A for your product for specifications of which circuits include isolation boundaries and the ratings of the boundaries. Isolation boundaries rated less than 1500 VAC are designed as functional isolation only and should not be depended on as safety boundaries.

- CPU logic reference is the same as DC Sensor Supply M.
- CPU logic reference is the same as the input power supply M on a CPU with DC power supply.
- CPU logic is isolated from ground to 100 VDC.
- DC digital inputs and outputs are isolated from CPU logic to 500 VAC.
- Relay outputs and AC inputs are isolated from CPU logic to 1500 VAC.
- Relay output groups are isolated from each other by 1500 VAC.
- AC power supply Line and Neutral are isolated from ground, the CPU logic, and all I/O to 1500 VAC.

Using the Optional Field Wiring Connector

The optional field wiring fan-out connector (Figure 1-10) allows for field wiring connections to remain fixed when you remove and re-install the CPU 210. Refer to Appendix F for the order number.

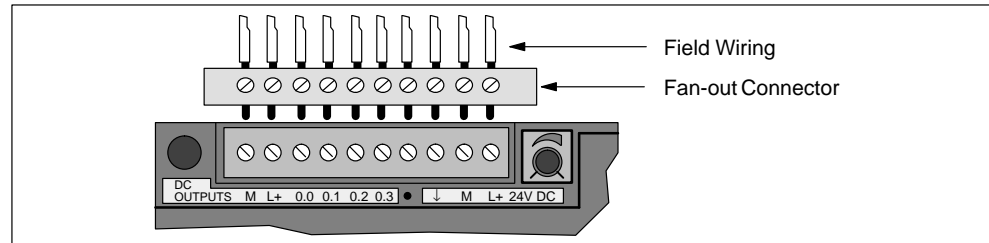


Figure 1-10 Optional Field Wiring Connector

Guidelines for AC Installation

The following items are general wiring guidelines for AC installations. Refer to Figure 1-11.

- Provide a single disconnect switch (**A**) that removes power from the CPU, all input circuits, and all output (load) circuits.
- Provide overcurrent devices (**B**) to protect the CPU power supply, the output points, and the input points. You can also fuse each output point individually for greater protection. External overcurrent protection for input points is not required when you use the 24 VDC sensor supply (**C**) from the CPU 210. This sensor supply is short-circuit protected.
- Connect all CPU 210 ground terminals to the closest available earth ground (**D**) to provide the highest level of noise immunity. It is recommended that all ground terminals be connected to a single electrical point. Use 14 AWG or 1.5 mm² wire for this connection.

If required, you can use a DC Sensor Supply from the CPU 210 to supply power for the inputs (**E**). Refer to the guidelines for DC installation, especially in regard to connecting and external power supply in parallel with the power supply of the CPU 210.

Guidelines for DC Installation

The following items are general wiring guidelines for isolated DC installations. Refer to Figure 1-11.

- Provide a single disconnect switch (**1**) that removes power from the CPU, all input circuits, and all output (load) circuits.
- Provide overcurrent devices to protect the CPU power supply (**2**), the output points (**3**), and the input points (**4**). You can also fuse each output point individually for greater protection. External overcurrent protection for input points is not required when you use the 24 VDC sensor supply from the CPU 210. This sensor supply is internally current limited.
- Ensure that the DC power supply has sufficient surge capacity to maintain voltage during sudden load changes. External capacitance (**5**) may be required.

- Install or equip ungrounded DC power supplies with a resistor and a capacitor in parallel (6) from the power source common to protective earth ground. The resistor provides a leakage path to prevent static charge accumulations, and the capacitor provides a drain for high frequency noise. Typical values are 1M Ω and 4700 pf. You can also create a grounded DC system by connecting the DC power supply to ground (7).
- Connect all CPU 210 ground terminals to the closest available earth ground (8) to provide the highest level of noise immunity. It is recommended that all ground terminals be connected to a single electrical point. Use 14 AWG or 1.5 mm² wire for this connection.
- Always supply 24 VDC circuits from a source that provides safe electrical separation from 120/230 VAC power and similar hazards. Refer to the following documents for standard definitions of "safe separation": PELV (protected extra low voltage) according to EN60204-1, and Class 2 or Limited Voltage/Current Circuit according to UL 508.



Warning

Connecting an external 24 VDC power supply in parallel with the DC sensor supply of the CPU 210 can result in a conflict between the two supplies as each seeks to establish its own preferred output voltage level. The result of this conflict can be shortened lifetime or immediate failure of one or both power supplies, with consequent unpredictable operation of the PLC system. Unpredictable operation could result in death or serious injury to personnel, and/or damage to equipment and property.

The CPU 210 DC Sensor Supply and any external power supply should provide power to different points, with at most one connection between the two supplies.

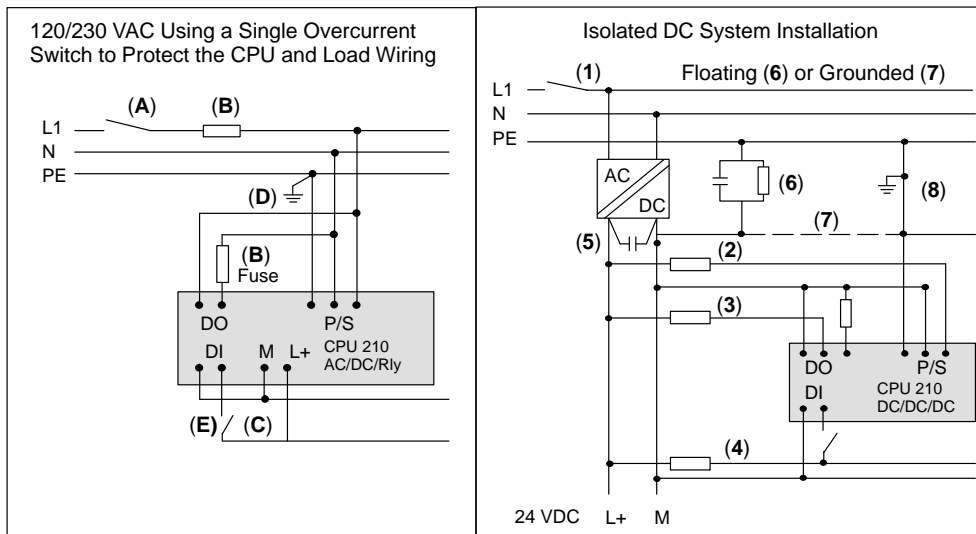


Figure 1-11 Wiring Guidelines for AC and DC Installation

1.5 Using Suppression Circuits

Install or equip inductive loads with suppression circuits that limit voltage rise on loss of power. Use the following guidelines to design adequate suppression. The effectiveness of a given design is dependent on the application, and you must verify it for a particular use. Be sure all components are rated for use in the application.

Protecting DC Transistors

The DC transistor outputs of the CPU 210 contain zener diodes that are adequate for many installations. Use external suppression diodes for either large or frequently switched inductive loads to prevent overpowering the internal diodes. Figure 1-12 shows typical applications for DC transistor outputs.

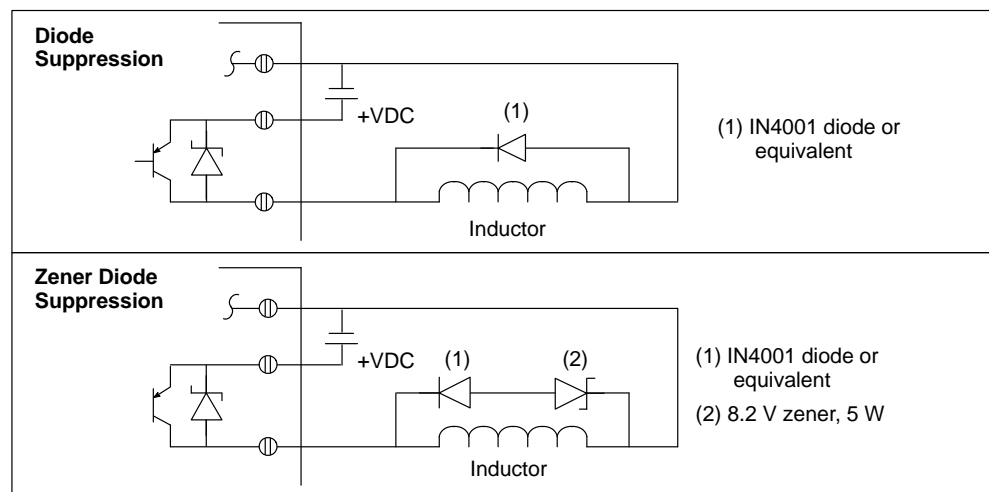


Figure 1-12 Diode Suppression and Zener Diode Suppression

Protecting Relays Controlling DC Power

Resistor/capacitor networks, as shown in Figure 1-13, can be used for low voltage (30 V) DC relay applications. Connect the network across the load. You can also use diode suppression, as shown in Figure 1-12, for DC relay applications. A threshold voltage of up to 36 V is allowed if you use a reverse zener diode.

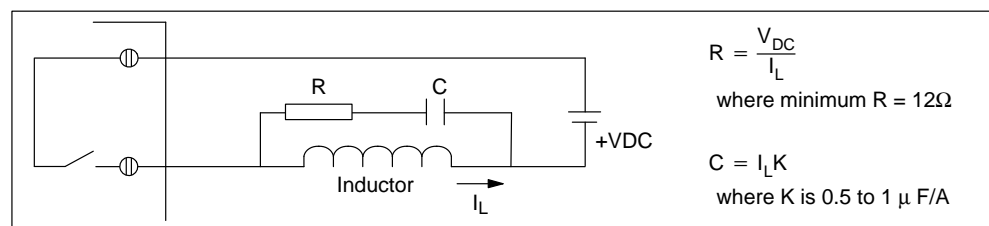


Figure 1-13 Resistor/Capacitor Network on Relay-Driven DC Load

Installing and Using the STEP 7-Micro/WIN Version 2.0 Software

2

This manual describes Version 2.0 of STEP 7-Micro/WIN. Previous versions of the software may operate differently.

STEP 7-Micro/WIN is a Windows-based software application used for programming the S7-200 Micro PLC (programmable logic controller). The STEP 7-Micro/WIN programming software package provides a set of tools required to program the S7-210 in either statement list (STL) or ladder logic (LAD) programming language.

In order to use STEP 7-Micro/WIN, you must have the following equipment:

- Recommended: a personal computer (PC) with an 80486 or greater processor and 8 Mbyte of RAM or a Siemens programming device (such as a PG 740); minimum computer requirement: 80386 with 8 Mbyte of RAM
- A PC/PPI cable connected to your communications port (COM)
- A program development station (PDS 210)
- VGA monitor, or any monitor supported by Microsoft Windows
- At least 35 Mbyte of free hard disk space (recommended)
- Microsoft Windows 3.1, Windows for Workgroups 3.11, Windows 95, or Windows NT 3.51 or greater
- Optional but recommended: any mouse supported by Microsoft Windows

STEP 7-Micro/WIN provides extensive online help. Use the **Help** menu command or press **F1** to obtain the most current information.

Chapter Overview

Section	Description	Page
2.1	Installing the STEP 7-Micro/WIN Version 2.0 Software	2-2
2.2	Establishing Communication with the PDS 210	2-3
2.3	Configuring the Preferences for STEP 7-Micro/WIN	2-5
2.4	Creating and Saving a Project	2-6
2.5	Creating a Program	2-7
2.6	Downloading a Program	2-10
2.7	Using Symbolic Addressing	2-13
2.8	Using the Status Chart	2-15
2.9	Debugging and Monitoring Your Program	2-16
2.10	Error Handling for the PDS 210	2-17

2.1 Installing the STEP 7-Micro/WIN Version 2.0 Software

Pre-installation Instructions

Before running the setup procedure, do the following:

- If a previous version of STEP 7-Micro/WIN is installed, back up all application programs to diskette.
- Make sure all applications are closed, including the Microsoft Office toolbar.

Installation may require that you restart your computer.

Installation Instructions for Windows 3.1

If you have Windows 3.1 (Windows for Workgroups 3.11 or Windows NT) on your machine, use the following procedure to install the STEP 7-Micro/WIN software:

1. Start by inserting Disk 1 in the disk drive of your computer (usually designated drive A: or drive B:).
2. From the Program Manager, select the menu command **File ► Run...**
3. In the Run dialog box, type **a:\setup** and click on the "OK" button. This starts the setup procedure.
4. Follow the online setup procedure to complete the installation.

Installation Instructions for Windows 95

If you have Windows 95 on your machine, you can use the following procedure to install the STEP 7-Micro/WIN software:

1. Start by inserting Disk 1 in the disk drive of your computer (usually designated drive A: or drive B:).
2. Click once on the **Start** button to open the Windows 95 menu.
3. Click on the **Run...** menu item.
4. In the Run dialog box, type **a:\setup** and click on the "OK" button. This starts the setup procedure.
5. Follow the online setup procedure to complete the installation.

Troubleshooting the Installation

The following situations can cause the installation to fail:

- Not enough memory: you need to have at least 35 Mbyte of free space on your hard disk.
- Bad diskette: verify that the diskette is bad, then call your salesman or distributor.
- Operator error: start over and read the instructions carefully.
- Failure to close any open applications, including the Microsoft Office toolbar.

Note

Review the README.txt file included on your diskettes for the most recent information about STEP 7-Micro/WIN. (In the x position, the letter A = German, B = English, C = French, D = Spanish, E = Italian.)

2.2 Establishing Communication with the PDS 210

Connecting Your Computer to the PDS 210 for PPI Communications

Figure 2-1 shows a typical configuration for connecting your personal computer to your PDS 210 with the PC/PPI cable. To establish proper communications between the components, follow these steps:

1. Set the dipswitches on the PC/PPI cable for the baud rate of 9600 baud.
2. Connect the RS-232 end of the PC/PPI cable labeled PC to the communications port of your computer, either COM1 or COM2, and tighten the connecting screws.
3. Connect the other end (RS-485) of the PC/PPI cable to the communications port of the PDS 210, and tighten the connecting screws.

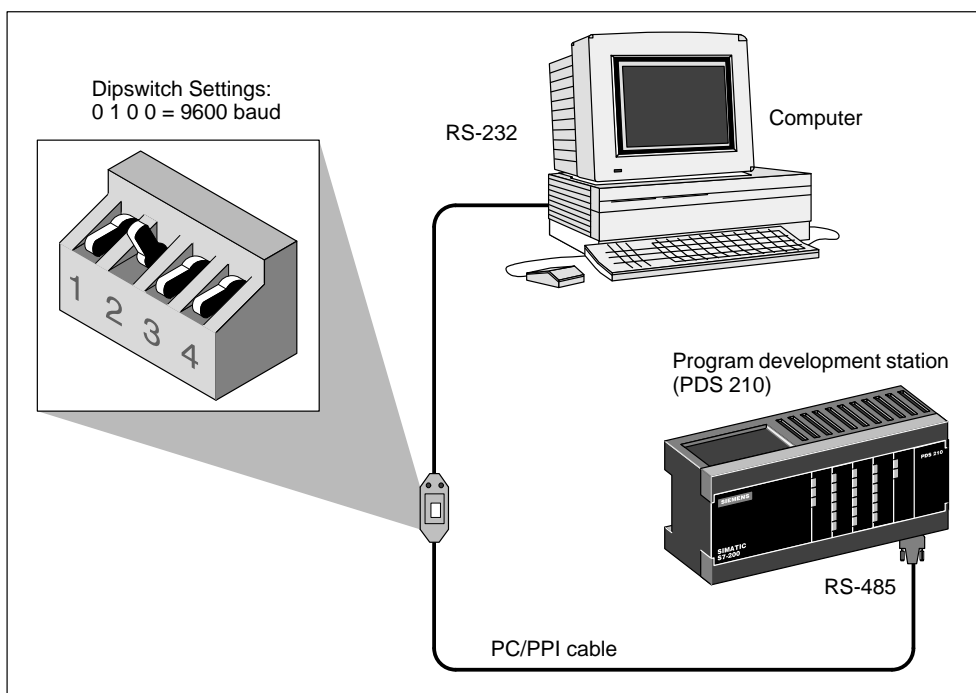


Figure 2-1 Communicating with a PDS 210 in PPI Mode

Setting Up the Communications Parameters

Figure 2-2 shows the Setup Communications dialog box. The first two port options are for PC communication ports. The address for the PDS 210 is 2 and cannot be changed. To set up the communication parameters, follow these steps:

1. Select the menu command **Setup ► Communications...**
2. Verify that the information in the dialog box is correct for your configuration. Remember that the CPU address for the PDS 210 is always 2, and that the baud rate is always 9600.
3. Confirm your selections by clicking the "OK" button.

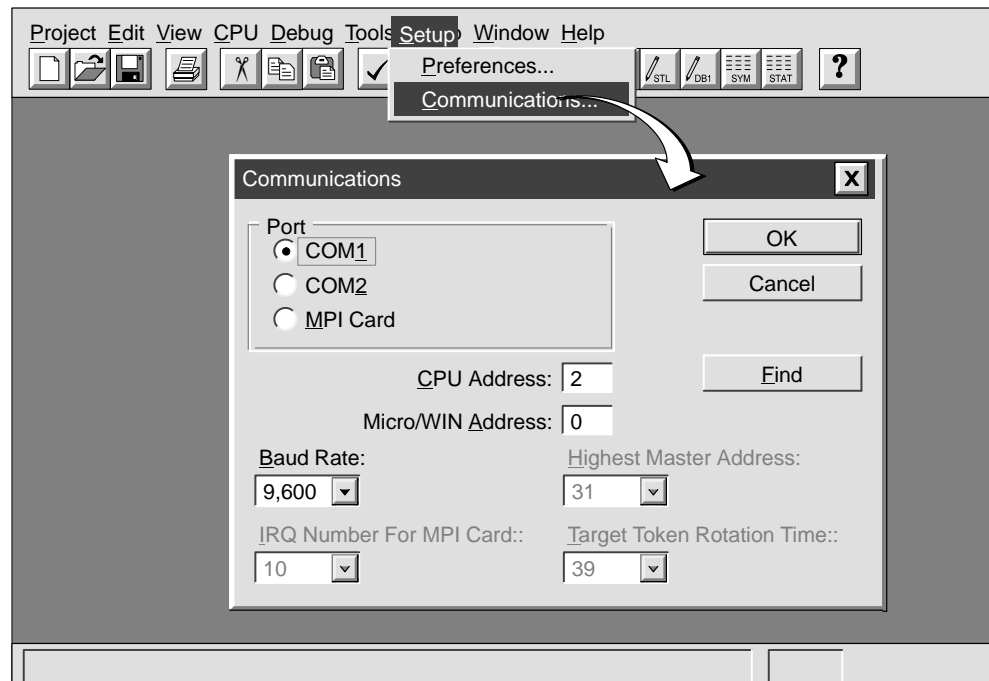


Figure 2-2 Setting Up Communications with the PDS 210

2.3 Configuring the Preferences for STEP 7-Micro/WIN

Before creating a new project, specify the preferences for your programming environment. To select your preferences, follow these steps:

1. Select the menu command **Setup ► Preferences...** as shown in Figure 2-3.
2. Select your programming preferences in the dialog box that appears.
3. Confirm your choices by clicking the “OK” button.

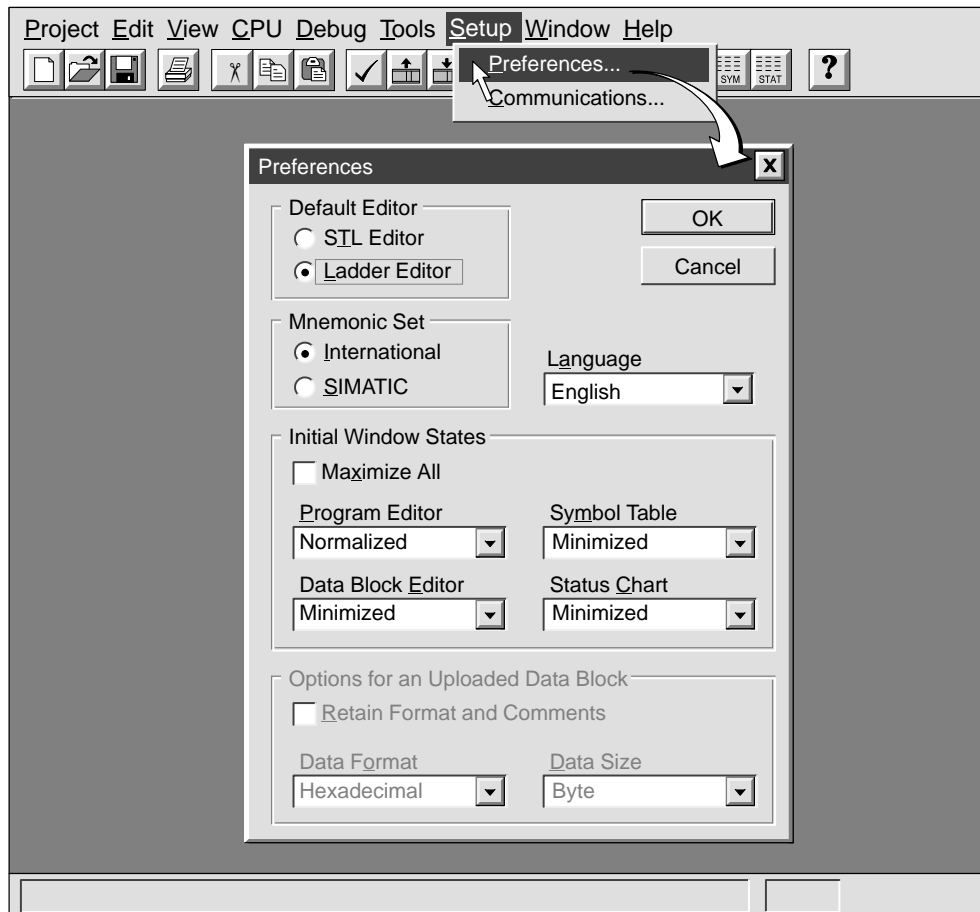


Figure 2-3 Selecting Your Programming Preferences

2.4 Creating and Saving a Project

Before you create a program, you must create or open a project. When you create a new project, STEP 7-Micro/WIN opens the following editors:

- Ladder Editor or Statement List Editor (depending on your selected preference)
- Data Block Editor (not applicable for the PDS 210)
- Status Chart
- Symbol Table

Creating a New Project

The Project menu command allows you to create a new project, as shown in Figure 2-4. Select the menu command **Project ► New....** The CPU Type dialog box is displayed. If you select the CPU type from the drop-down list box, the software displays only those options which are available for your CPU. If you select "None," no CPU-specific restrictions are placed on your program. When you download the program, the CPU notifies you if you have used options that are not available. For example, if your program uses an instruction that is not supported by your CPU, the program is rejected.

Note

STEP 7-Micro/WIN does not range-check parameters. For example, you can enter MW999 as a parameter to a ladder instruction even though it is an invalid parameter. This error would be identified when you attempt to download the program.

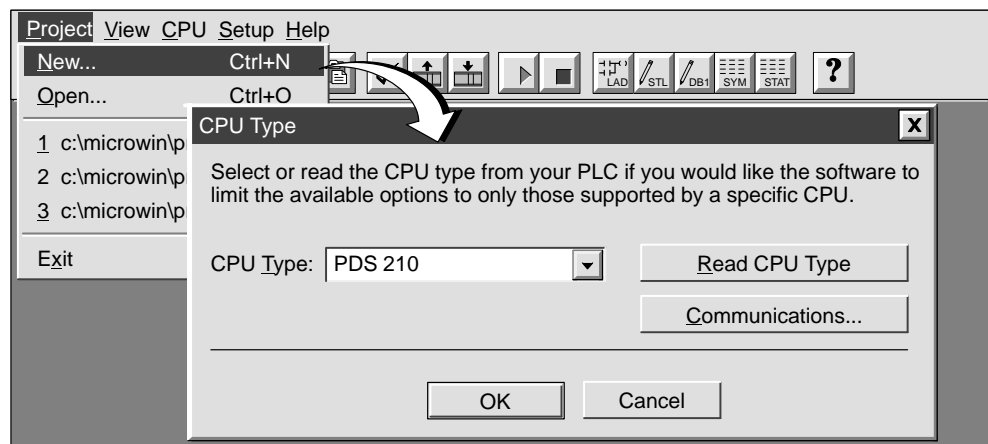



Figure 2-4 Creating a New Project

Saving a Project

You can save a copy of the active project to a different name or location by selecting the menu command **Project ► Save As...** You can save all of the components of your project by selecting the menu command **Project ► Save All** or by clicking the Save button: 

2.5 Creating a Program

STEP 7-Micro/WIN allows you to create the user program (OB1) with either the Ladder Editor or the Statement List Editor.

Entering Your Program in Ladder

The Ladder Editor window allows you to write a program using graphical symbols. See Figure 2-5. The toolbar includes some of the more common ladder elements used to enter your program. The first (left) drop-down list box contains instruction categories. You can access these categories by clicking or pressing F2. After a category is selected, the second drop-down list contains the instructions specific to that category. To display a list of all instructions in alphabetic order, press F9 or select the All Instructions category.

Each network allows two types of comments:

- Single-line network title comments are always visible in the ladder display. You can access the network editor by double-clicking anywhere in the network title region.
- Multi-line network comments are only visible through a dialog box, but can be printed (if that option has been selected through the Page Setup dialog). You can access the network comment editor by double-clicking anywhere in the network title region.

To start entering your program, follow these steps:

1. To enter a program title, select the menu command **Edit ► Program Title**.
2. To enter ladder elements, select the type of element you want by clicking the corresponding icon button or selecting from the instruction list.
3. Type the address or parameter in each text field and press ENTER.

To change or replace one of the elements, move the cursor to that element and select the new element. You can also cut, copy, or paste elements at the cursor location.

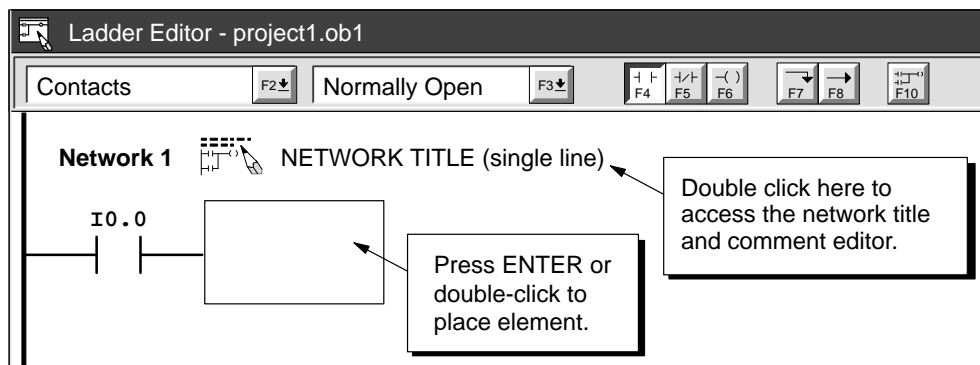


Figure 2-5 Ladder Editor Window

Entering Your Program in Statement List

The Statement List (STL) Editor is a free-form text editor which allows a certain degree of flexibility in the way you choose to enter program instructions. Figure 2-6 shows an example of a statement list program.

You can cut, copy, and paste in the STL Editor. STEP 7-Micro/WIN also includes search-and-replace functions.

```

STL Editor - project1.ob1
// Program for a Home Security System

NETWORK 1          //Sound the alarm!
LD      I0.3        // If (the panic alarm has been t
LDW>=  T0, +600    // or (if the alert timer is >= 6
A       I0.2        //      and the system is armed)
OLD     // then
S       M0.1, 1     // set the high-level alarm bit
S       Q0.3, 1     // set the modem dialer bit
R       M0.2, 1     // reset the low-level alarm bit

Network 2          //Evaluate the system status.
LDN     I0.0        // If zone 1 is open
ON      I0.1        // or if Zone 2 is open
    
```

Figure 2-6 STL Editor Window with Sample Program

To enter an STL program, follow these guidelines:

- Start each comment with a double slash (//). Each additional comment line must also begin with a double slash.
- End each line with a carriage return.
- Separate each instruction from its address or parameter with a space or tab.
- Do not use a space between the operand type and the address (for example, enter I0.0, not I 0.0).
- Separate each operand within an instruction with a comma, space, or tab.
- Use quotation marks when entering symbol names. For example, if your symbol table contains the symbol name Start1 for the address I0.0, enter the instruction as follows:

```
LD "Start1"
```

To be able to view an STL program in ladder, you must divide segments of code into separate networks by entering the keyword NETWORK. (Network numbers are generated automatically after you compile or upload the program.)

Compiling the Program

After completing a network or series of networks, you can check the syntax of your code by selecting the menu command **CPU ► Compile** or by clicking the Compile button:

Viewing a Program in Ladder or Statement List

You can view a program in either ladder or STL by selecting the menu command **View ► STL** or **View ► Ladder**, as shown in Figure 2-7.

When you change the view from STL to ladder and back again to STL, you may notice changes in the presentation of the STL program, such as:

- Instructions and addresses are changed from lower case to upper case.
- Spaces between instructions and addresses are replaced with tabs.

You can accomplish the same formatting of the STL instructions by selecting the menu command **CPU ► Compile** while the STL Editor is active.

Note

Certain combinations of statement list instructions cannot successfully be converted to ladder view. In that case, the message "Illegal Network" marks the section of code that cannot be represented in ladder. You can view the STL instructions for the "illegal" network by clicking on the network title. Use the STL Editor to modify an illegal network so that it can be viewed in ladder.

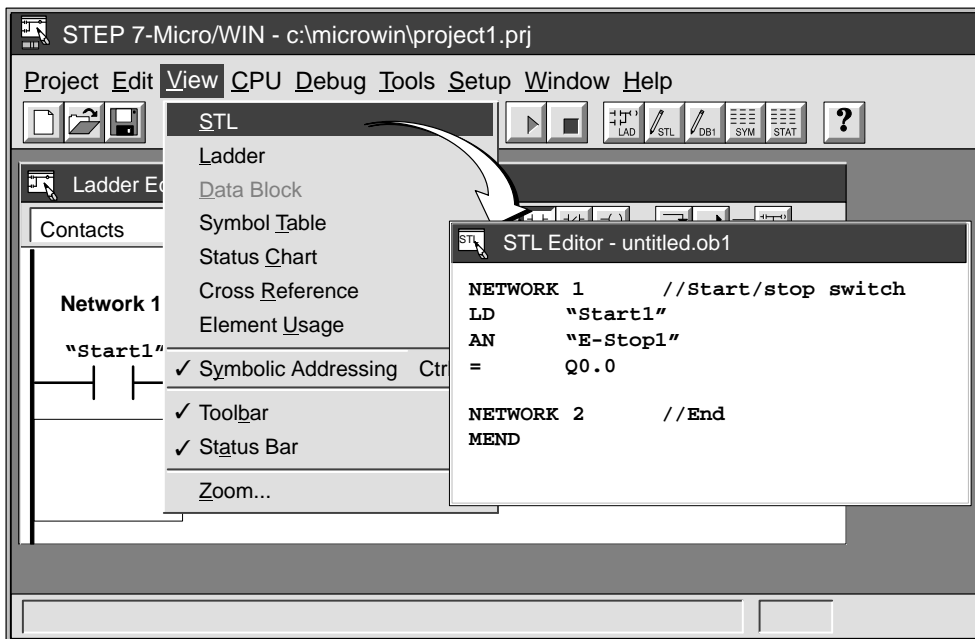


Figure 2-7 Changing the Program View from Ladder to Statement List

2.6 Downloading A Program

After developing and testing your program on the PDS 210, you must transfer the program to the CPU 210 using the memory cartridge. In the same manner as you could use a diskette to transfer files from one computer to another, you use a memory cartridge to transfer your program from the PDS 210 to the CPU 210.

Downloading the Program to the PDS 210

After completing your program, you can download the project to the PDS 210. To download your program, select the menu command **Project ► Download...** or click the Download button in the main window. 

The Download dialog box that appears allows you to specify the project components you want to download, as shown in Figure 2-8. Select only "Program Code Block" for the PDS 210: the data block and the CPU configuration are not used by the CPU 210.

Click on the "OK" button to confirm your choices and to execute the download operation.

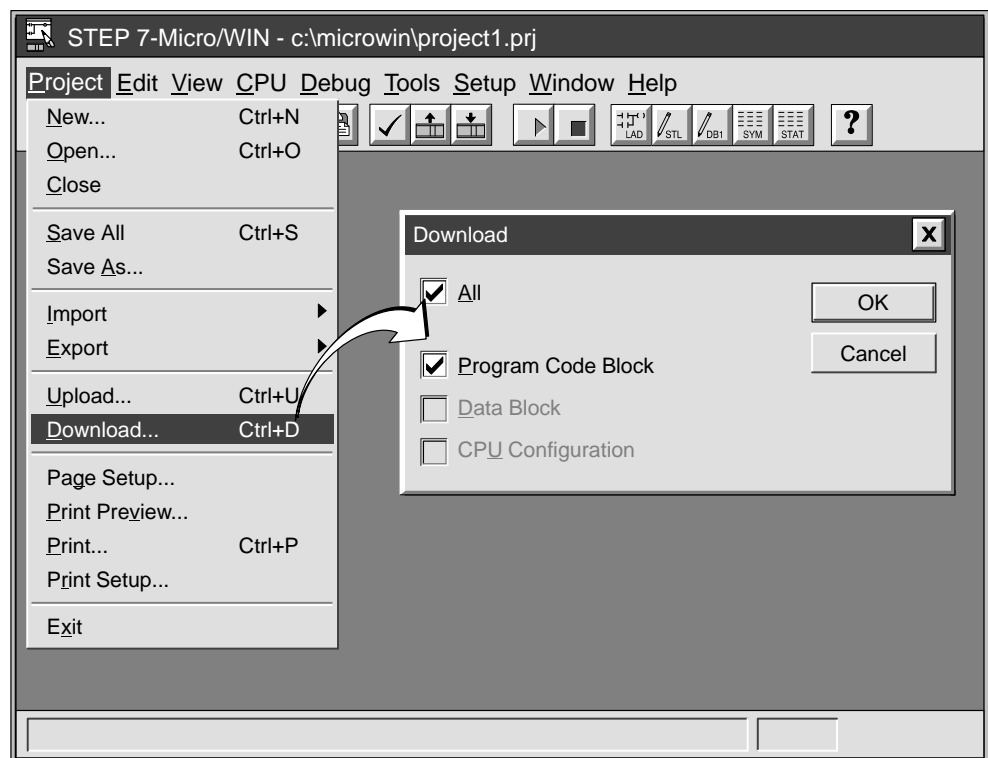


Figure 2-8 Downloading Project Components to the CPU

Copying Your Program to the Memory Cartridge

You can copy your program to the memory cartridge only when the PDS 210 is powered up and the memory cartridge is installed. (You can install or remove the memory cartridge while the PDS 210 is powered up.)



Caution

Electrostatic discharge can damage the memory cartridge or the receptacle on the PDS 210 or CPU 210.

You should make contact with a grounded conductive pad and/or wear a grounded wrist strap when you handle the cartridge. You should store the cartridge in a conductive container.

To install the memory cartridge, remove the protective tape from the memory cartridge receptacle and insert the memory cartridge into the receptacle located under an access cover of the PDS 210. (The memory cartridge is keyed for proper installation.) After the memory cartridge is installed, use the following procedure to copy the program:

1. If the program has not already been downloaded to the PDS 210, use the menu command **Project ► Download...** to download the program. (See Figure 2-8.)
2. Use the menu command **CPU ► Program Memory Cartridge** to copy the program to the memory cartridge. See Figure 2-9.
3. Remove the memory cartridge from the PDS 210.

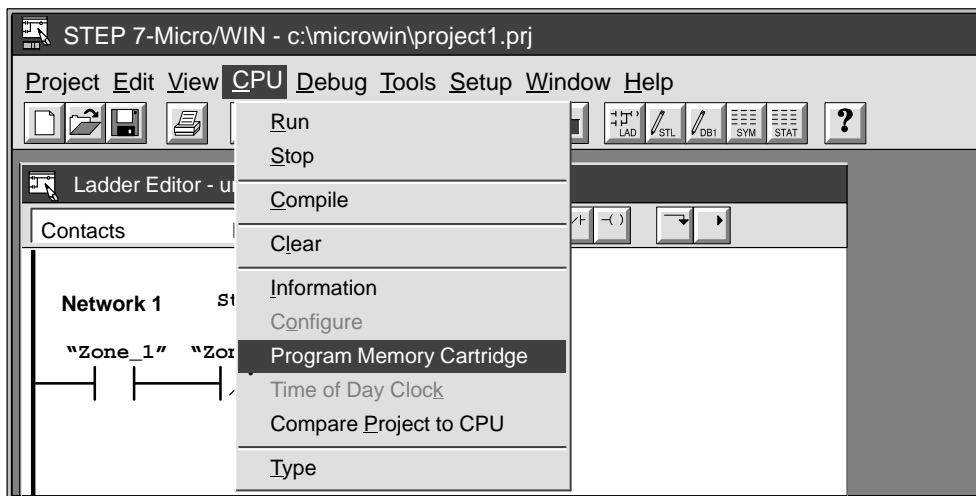


Figure 2-9 Copying the Program to the Memory Cartridge

Transferring the Program to the CPU 210

To transfer the program from the memory cartridge to the CPU 210, follow these steps:

1. Turn off the power to the CPU 210.
2. Insert the memory cartridge in the CPU 210. (The memory cartridge is keyed for proper installation.)
3. Turn on the power to the CPU 210.
4. After the RUN LED turns on, remove the memory cartridge from the CPU 210.

As shown in Figure 2-10, the CPU 210 performs the following tasks after you turn the power on when a memory cartridge is installed in the CPU 210:

- The M, T, and Q areas of memory are cleared.
- The current values for the counters (which are stored in the permanent memory) are cleared. (The current values for the counters are erased only when the memory cartridge is installed in the CPU 210. If a memory cartridge is not installed, the current values are retained.)
- The user program is copied from the memory cartridge to the permanent EEPROM memory.

Always remove the memory cartridge from the CPU 210 after the program has been installed.

Note

Turning the power on with a blank memory cartridge in the CPU 210 causes an error and lights the error LED. Any program stored in the permanent EEPROM is not affected or overwritten. To correct the error condition, remove the memory cartridge and cycle the power again.

When a valid program is installed, the CPU 210 automatically goes to RUN mode when power is applied.

As your program runs, the CPU 210 updates the values stored in the RAM memory (the values stored in M memory, the current values for the four counters, and the current values for the four timers).

When you turn the power off, the CPU 210 saves the current values of the four counters to the permanent EEPROM memory. The other values stored in RAM (such as M memory, current values for the timers, and the copy of the user program) are cleared.

Unless a memory cartridge is installed in the CPU 210, the current values for the counters are retentive. The current values for the counters are automatically restored to the RAM memory when you turn power on for the CPU 210 (with no memory cartridge installed).

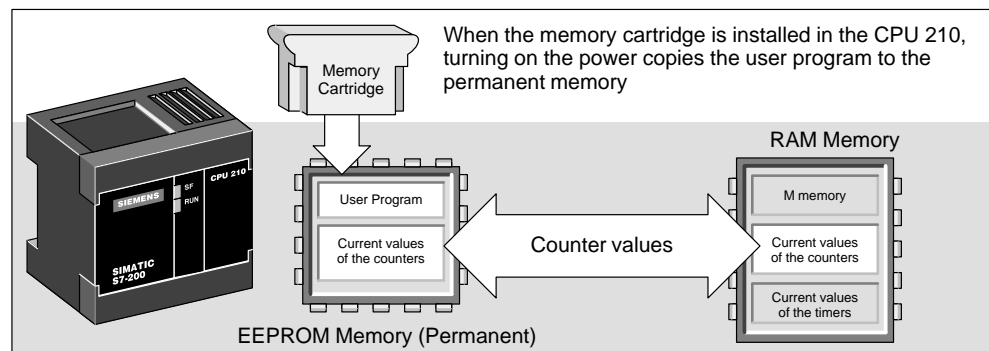


Figure 2-10 Loading a Program with the Memory Cartridge

2.7 Using Symbolic Addressing

The Symbol Table allows you to give symbolic names to inputs, outputs, and internal memory locations. See Figure 2-11. You can use the symbols you have assigned to these addresses in the Ladder Editor, STL Editor, and Status Chart of STEP 7-Micro/WIN.

Guidelines for Entering Symbolic Addresses

The first column of the Symbol Table is used to select a row. The other columns are for the symbol name, address, and comment. For each row, you assign a symbolic name to the absolute address of a discrete input, output, memory location, special memory bit, or other element. A comment for each assigned symbol is optional. Follow these guidelines when creating a Symbol Table:

- You can enter symbol names and absolute addresses in any order.
- You can use up to 23 characters in the Symbol Name field; however, depending on the font size of your Windows environment, you may not see the full name displayed in the Ladder Editor.
- You can define up to 500 symbols.
- The Symbol Table is case-sensitive: for example, "Low_Alert" is considered a different symbol from "low_alert".
- All leading and trailing spaces will be removed from the symbol name. All adjacent internal spaces will be converted to a single underscore. For example, if you type "Zone 1" and press ENTER, the symbol name appears as: "Zone_1".
- Duplicate symbol names and/or addresses will be marked by blue italics, will not be compiled, and cannot be used in the program. Overlapping addresses are not flagged as duplicates; for example, MW0 and MW1 overlap in memory but are not flagged as duplicates.

Starting the Symbol Table Editor

The Symbol Table editor appears by default as a minimized window icon at the bottom of the main window. To access the Symbol Table, double-click the icon, or click the Restore or Maximize button on the icon (in Windows 95).

	Symbol Name	Address	Comment
	Zone_1	I0.0	
	Zone_2	I0.1	
	Armed	I0.2	
	Panic_Alarm	I0.3	Turns on the siren
	LED	Q0.0	
	Alarm	Q0.1	
	Low_Alert	Q0.2	
	<i>LED_Bit</i>	M0.0	
	<i>LED_Bit</i>	M0.1	

Figure 2-11 Example of a Symbol Table

Editing Functions within the Symbol Table

The Symbol Table provides the following editing functions:

- **Edit ► Cut / Copy / Paste** within a cell or from one cell to another.
- **Edit ► Cut / Copy / Paste** one or several adjacent rows.
- **Edit ► Insert Row(s)** above the row containing the cursor. You can also use the INSERT or INS key for this function.
- **Edit ► Delete Row(s)** for one or several highlighted adjacent rows. You can also use the DELETE or DEL key for this function.
- To edit any cell containing data, use the arrow keys or mouse to select the cell you want to edit. If you begin typing, the field clears and the new characters are entered. If you double-click the mouse or press F2, the field becomes highlighted, and you can use the arrow keys to move the editing cursor to the place you want to edit.
- Clicking the right mouse button displays a menu of editing functions which are available with the Symbol Table editor.

Sorting Table Entries

After entering symbol names and their associated absolute addresses, you can sort the Symbol Table alphabetically by symbol names or numerically by addresses in the following ways:

- Select the menu command **View ► Sort Symbol Name** to sort the symbol names in alphabetical order.
- Select the menu command **View ► Sort Symbol Address** to sort the absolute addresses numerically in the following order for memory types: I, Q, M, C, T, and SM.

Displaying the Symbolic Addresses

After you create the Symbol Table for your program, you can use the menu command **View ► Symbolic Addressing** to enable or disable the use of symbolic addressing with the Program Editor (ladder or STL) and the Status Chart. See Figure 2-12.

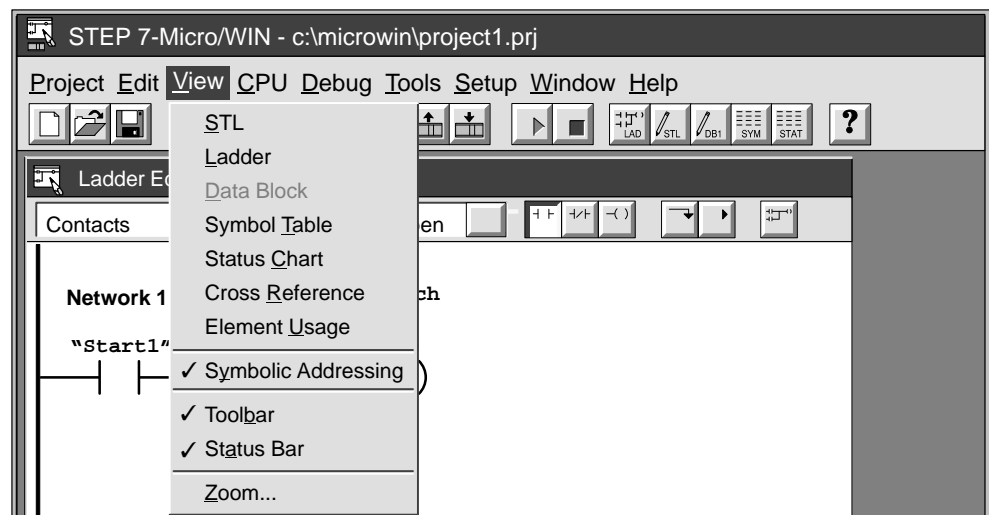


Figure 2-12 Displaying the Symbolic Addresses



2.8 Using the Status Chart

The Status Chart editor appears by default as a minimized window icon at the bottom of the main window. To access the Status Chart, double-click the icon, or click the Restore or Maximize button on the icon (in Windows 95).


You can use the Status Chart to read or write variables in your program. You cannot force values in the PDS 210.

Reading and Writing Variables with the Status Chart

Figure 2-13 shows an example of a Status Chart. To read or write variables using the Status Chart, follow these steps:

1. In the first cell in the Address column, enter the address or the symbol name of an element from your program that you want to read or write, and press ENTER. Repeat this step for all additional elements you want in the chart.
2. If the element is a bit (I, Q, or M, for example), the format is set as bit in the Format column. If the element is a word, select the cell in the Format column and double-click or press the SPACEBAR to cycle through the valid formats.
3. To view the current PLC value of the elements in your chart, click the Single Read button  or the Continuous Read button  on the Status Chart.

You can click the Stop Read button  to stop the updating of status.

4. To change a value, enter the new value in the Change Value to column and click the Write button  to write the value to the PDS 210.

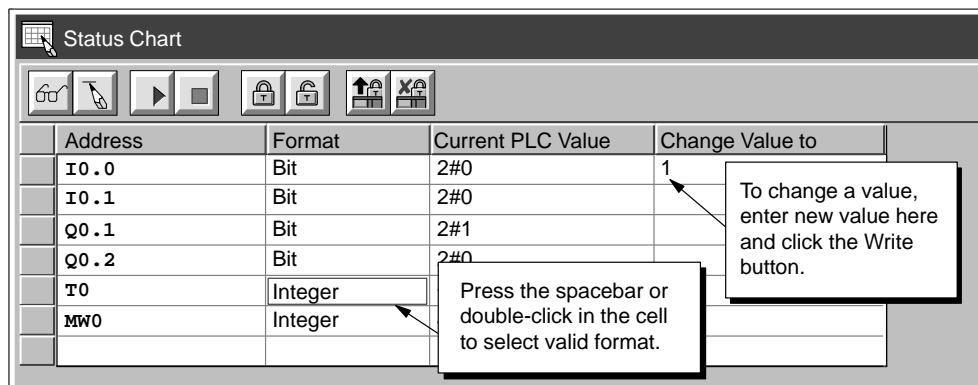


Figure 2-13 Example of a Status Chart

Editing Addresses

To edit an address cell, use the arrow keys or mouse to select the cell you want to edit.

- If you begin typing, the field clears and the new characters are entered.
- If you double-click the mouse or press F2, the field becomes highlighted and you can use the arrow keys to move the editing cursor to the place you want to edit.
- Clicking the right mouse button displays a menu of editing functions which are available with the Status Chart editor.

2.9 Debugging and Monitoring Your Program

Using Single/Multiple Scans to Monitor Your Program

You can specify that the PDS 210 execute your program for a limited number of scans (from 1 scan to 65,535 scans). By selecting the number of scans for the PDS 210 to run, you can monitor the program as it changes the process variables. Use the menu command **Debug ► Execute Scans...** to specify the number of scans to be executed. Figure 2-14 shows the dialog box for entering the number of scans for the CPU to execute.

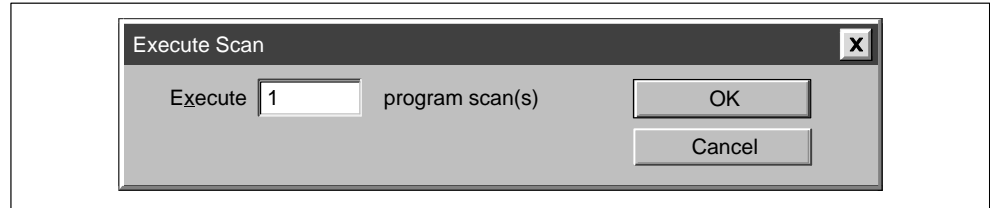


Figure 2-14 Executing Your Program for a Specific Number of Scans

Displaying the Status of the Program in Ladder Logic

As shown in Figure 2-15, the program editor of STEP 7-Micro/WIN allows you to monitor the status of the online program. (The program must be displaying ladder logic.) This allows you to monitor the status of the instructions in the program as they are executed by the CPU.

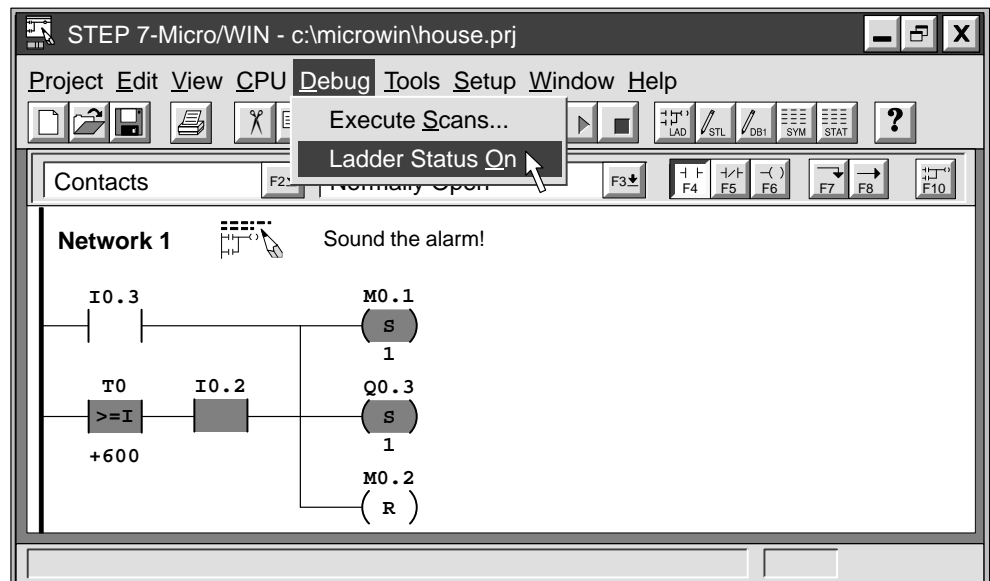


Figure 2-15 Displaying the Status of a Program in Ladder Logic

2.10 Error Handling for the PDS 210

The PDS 210 classifies errors as either fatal errors or non-fatal errors. You can use STEP 7-Micro/WIN to view the error codes that were generated by the error. Figure 2-16 shows the dialog box that displays the error code and the description of the error. Refer to Appendix C for a complete listing of the error codes.

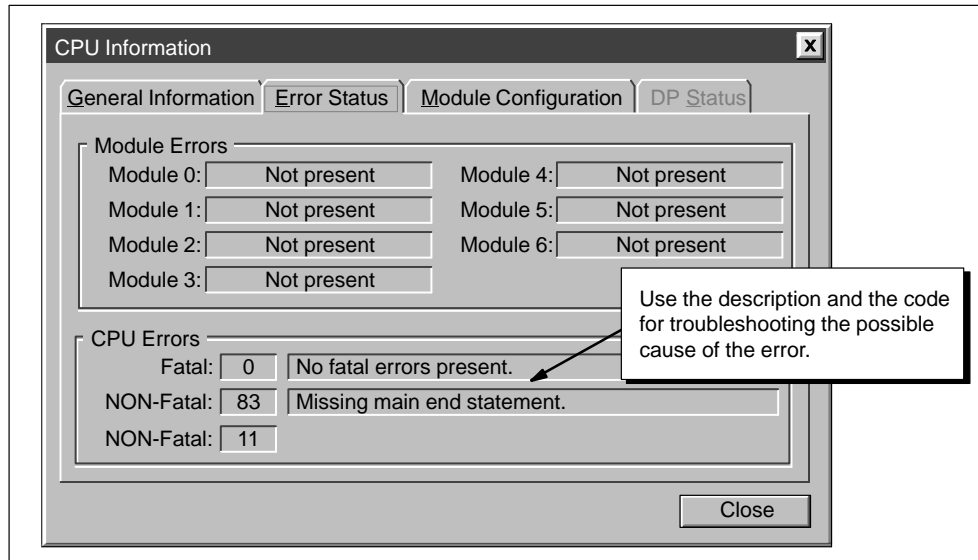


Figure 2-16 CPU Information Dialog: Error Status Tab

Responding to Fatal Errors

Fatal errors cause the PDS 210 to stop the execution of your program. Depending upon the severity of the fatal error, it can render the PDS 210 incapable of performing any or all functions. The objective for handling fatal errors is to bring the PDS 210 to a safe state from which the PDS 210 can respond to interrogations about the existing error conditions. When a fatal error is detected by the PDS 210, the PDS 210 changes to the STOP mode, turns on the System Fault LED and the STOP LED, and turns off the outputs. The PDS 210 remains in this condition until the fatal error condition is corrected.

Once you have made the changes to correct the fatal error condition, you must restart the PDS 210. You can restart the PDS 210 by cycling power. Restarting the PDS 210 clears the fatal error condition and performs power-up diagnostic testing to verify that the fatal error has been corrected. If another fatal error condition is found, the PDS 210 again sets the fault LED indicating that an error still exists. Otherwise, the PDS 210 begins normal operation.

There are several possible error conditions that can render the PDS 210 incapable of communication. In these cases, you cannot view the error code from the PDS 210. These errors indicate hardware failures that require the PDS 210 module to be repaired; these conditions cannot be fixed by changes to the program or clearing the PDS 210 memory.

Responding to Non-Fatal Errors

Non-fatal errors can degrade some aspect of the PDS 210 performance, but they do not render the PDS 210 incapable of executing your program or from updating the I/O. As shown in Figure 2-16, you can use STEP 7-Micro/WIN to view the error codes that were generated by the non-fatal error. For the PDS 210, there are two basic categories of non-fatal errors:

- **Run-time errors.** All non-fatal errors detected in RUN mode are reflected in special memory (SM) bits. Your program can monitor and evaluate these bits. Refer to Appendix B for more information about the SM bits used for reporting non-fatal run-time errors.
- **Program-compile errors.** The PDS 210 compiles the program as it downloads. If the PDS 210 detects that the program violates a compilation rule, the download is aborted and an error code is generated. (A program that was already downloaded to the PDS 210 would still exist in the EEPROM and would not be lost.) After you correct your program, you can download it again.

The PDS 210 does not change to STOP mode when it detects a non-fatal error.

Getting Started with a Sample Program

3

You can enter the program for the sample application on a computer running STEP 7-Micro/WIN. To download the program, you must have the equipment shown in Figure 3-1. The size of the sample program is 155 bytes.

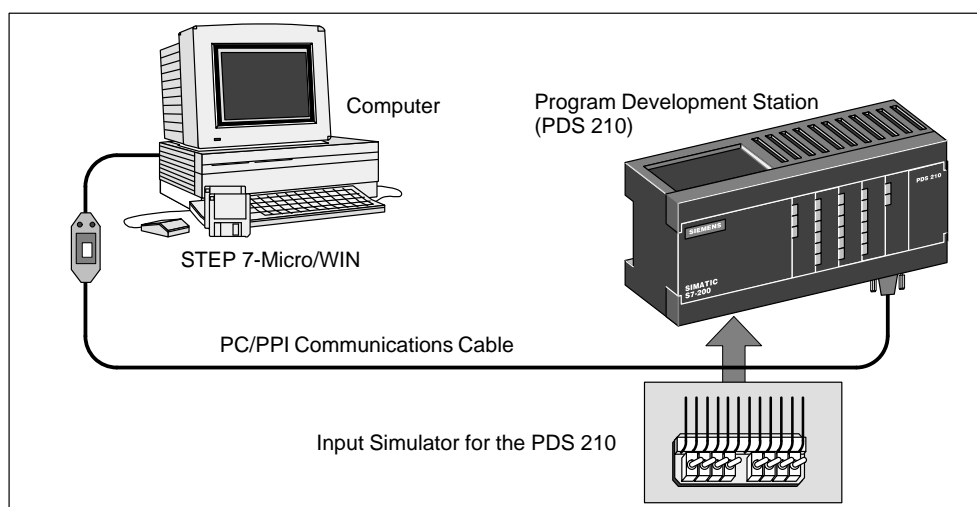


Figure 3-1 Requirements to Run the Sample Program

Chapter Overview

Section	Description	Page
3.1	Defining the Requirements for the Application Example	3-2
3.2	Designing the Control Logic	3-4
3.3	Putting the Control Logic into a Program	3-9
3.4	Creating a Project with STEP 7-Micro/WIN	3-13
3.5	Creating a Symbol Table	3-14
3.6	Creating the Program	3-15
3.7	Creating a Status Chart	3-22
3.8	Downloading and Monitoring the Sample Program	3-23
3.9	Modifying the Sample Program	3-25

3.1 Defining the Requirements for the Application Example

Defining the Inputs and Outputs for the Application

This chapter describes a sample program for a home security system. As shown in Figure 3-2, the program monitors two zones. Any breach of security results in an alarm being sounded. The sample program uses the following inputs:

- Input 1 (I0.0) monitors zone 1 (entrance, living room, kitchen, and bedroom 3).
- Input 2 (I0.1) monitors zone 2 (bedroom 1, bedroom 2, bathroom, and rear entrance).
- Input 3 (I0.2) provides the arm/disarm switch for the security system.
- Input 4 (I0.3) provides a “panic button” to immediately turn on the alarm siren.

In addition to the inputs, the program uses the following outputs.

- Output 1 (Q0.0) controls the LED on the security system.
- Output 2 (Q0.1) turns on the siren to sound an alarm.
- Output 3 (Q0.2) turns on a low-level notification alert to signify that the alarm will be turned on in a predetermined number of seconds.
- Output 4 (Q0.3) turns on an external interface relay (perhaps to an automatic dialing machine).

Figure 3-3 shows a wiring diagram for the home security application.

Creating Symbolic Names for the Elements of the Program

Symbolic names help to document or define the specific memory locations or I/O used by your program. Table 3-1 lists the symbolic names used by the program for the sample application. The sample program also uses SM0.5 to generate an on/off (blinking) pattern for the LED.

Table 3-1 Symbolic Names for the Application Example

Element	Address	Symbolic Name	Description
Inputs	I0.0	Zone_1	Normally Closed input for Zone 1
	I0.1	Zone_2	Normally Closed input for Zone 2
	I0.2	Armed	Armed = closed, and disarmed = open
	I0.3	Panic_Alarm	Normally Open input for panic alarm
Outputs	Q0.0	LED	System LED (on if armed, or flashing if disarmed and zone 1 or zone 2 open)
	Q0.1	Alarm	High-level alarm (siren)
	Q0.2	Low_Alert	Low-level alert to disarm system
	Q0.3	Modem	Relay to start the modem dialer unit
Internal Memory	M0.0	LED_Bit	Stores the status for the LED
	M0.1	Alarm_Bit	Stores the status for the alarm
	M0.2	Low_Bit	Stores the status for the low-level alert
Timers	T0	Alert_Timer	Provides a delay before the alarm turns on
	T2	Exit_Timer	Delay time after arming the system

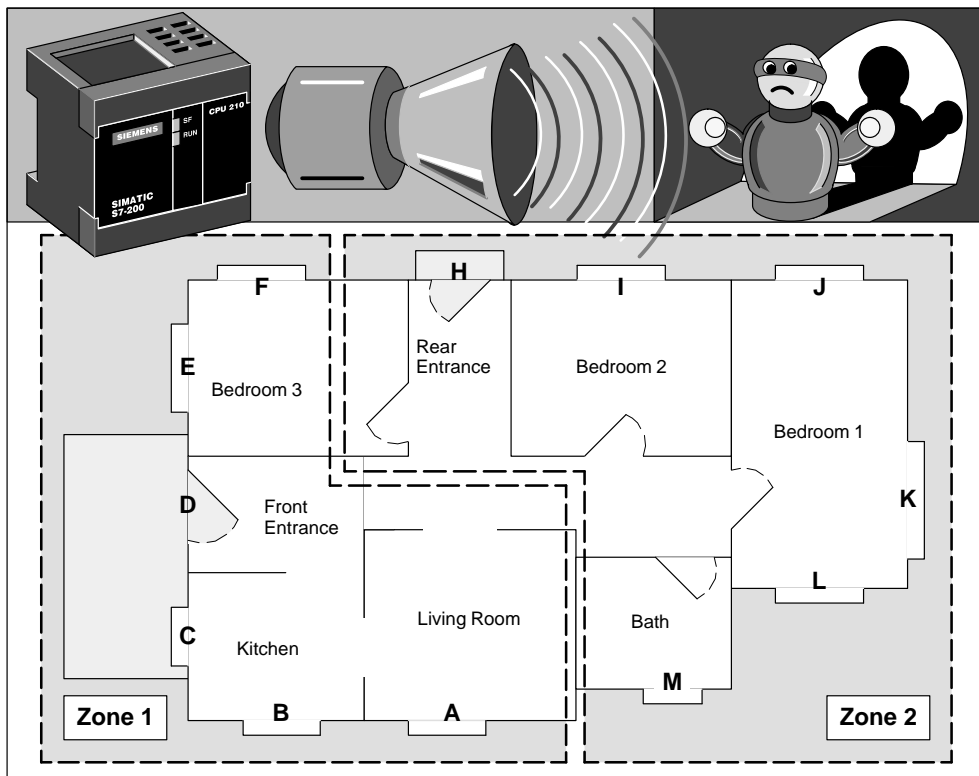


Figure 3-2 Sample Application for a Home Security System

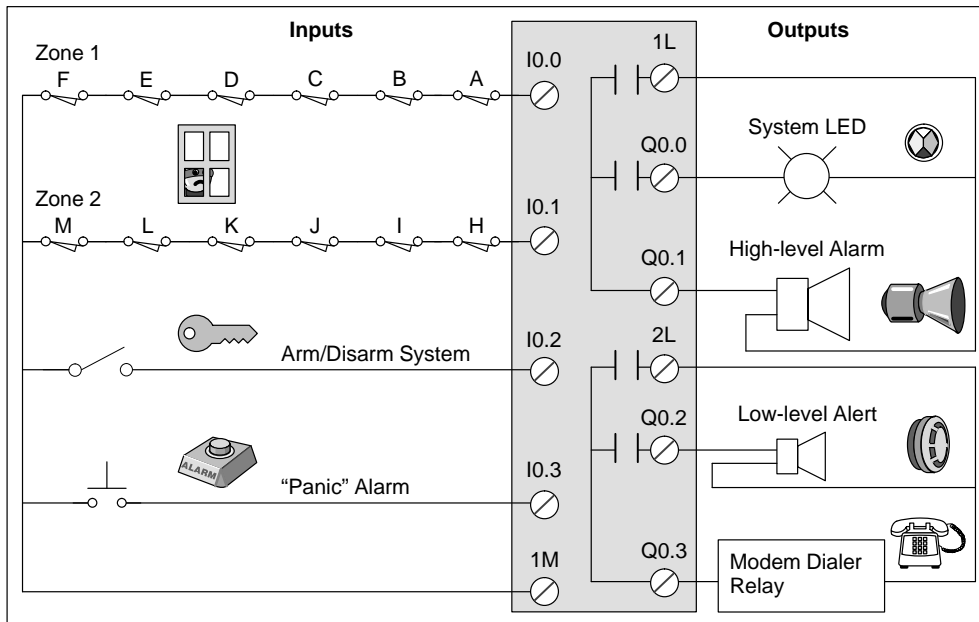


Figure 3-3 Wiring Diagram for the Sample Home Security Application

3.2 Designing the Control Logic

Creating a program is more involved than just entering the instructions into a file. Individual elements or tasks make up the control logic contained within the program. These elements relate to the various instructions, which are then arranged into networks.

This section provides an insight into how the sample program was structured.

Defining the Operation of the Program

Before entering the instructions into a program, you must plan the tasks that the program is to perform. For the home security system described in Section 3.1, the program must evaluate the status of the four inputs and respond by turning on or off the four outputs. As shown in Figure 3-4, the control logic of the program must perform the following tasks:

- If the system is not armed, the program flashes the LED (Q0.0) on and off when either Zone 1 (I0.0) or Zone 2 (I0.1) is open.
- When the system is armed (by turning the key to the “on” or “arm” position, which turns on I0.2), the program must start a delay timer which allows 90 seconds for the owner to exit the house. During this delay time, the program does not respond when either zone (I0.0 or I0.1) opens.
- If the system has been armed and the delay time for exiting the house has been achieved, the program evaluates the status of both zones. If either zone (I0.0 or I0.1) opens, the program starts a notification sequence that turns on the low-level alert buzzer (Q0.2) and starts a timer. This allows a reminder (and time) for the owner to disarm the system after returning home.
- Once the notification sequence has started, the program has two possible actions:
 - If the system is disarmed (by turning the key to the “off” or “disarm” position, which turns off I0.2), the program turns off the outputs (Q0.0 and Q0.2) and resets the timers.
 - If the system has not been disarmed within 60 seconds, the program turns on the alarm and the modem dialer (Q0.1 and Q0.3).
- If the panic alarm (I0.3) is turned on, the program turns on the alarm and the modem dialer (Q0.1 and Q0.3). The program performs this task regardless of the state of the arm/disarm switch (I0.2) and does not perform the notification sequence that provides a delay time for disarming the system.
- If the system is disarmed (by turning the key to the “off” or “disarm” position, which turns off I0.2) after the alarm (Q0.1) has been turned on, the program turns off the outputs (Q0.1 and Q0.3) and resets the timers.

Each of these tasks can be expressed as a sequence of instructions: the conditions of the logic determine the action to be taken.

Because the CPU 210 provides immediate outputs, the program uses the internal memory bits (M memory) to store the interim states of the logic relating to the physical outputs. After evaluating the control logic, the program uses the states of these memory bits to turn the outputs on or off.

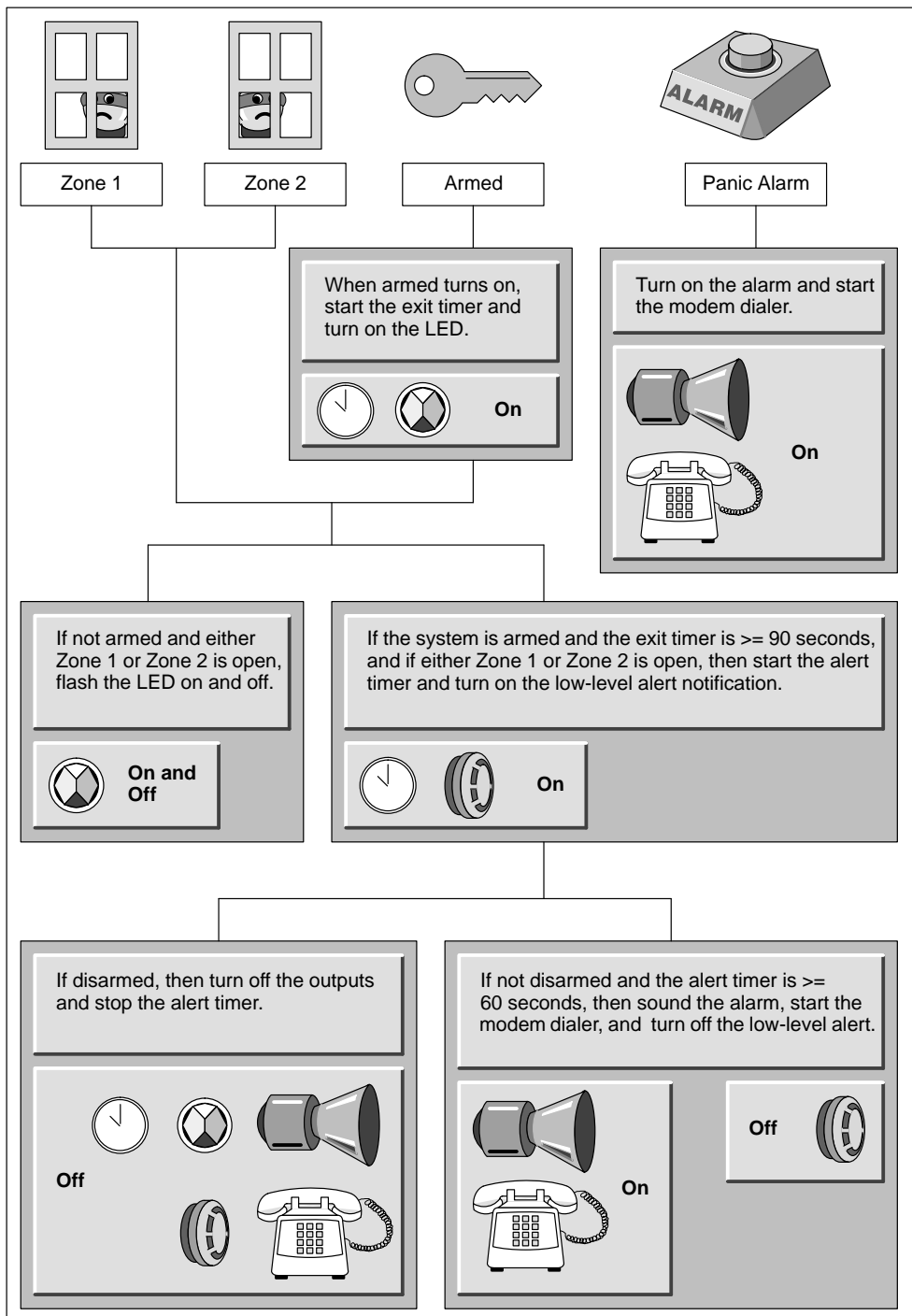


Figure 3-4 Basic Tasks for the Home Security Application Program

Designing the Control Logic for Arming and Disarming the System

Figures 3-5 and 3-6 shows the control logic for arming and disarming the security system.

- As shown in Figure 3-5, arming the security system activates (enables) the bits of M memory that control the outputs (alarm siren and modem dialer). The control logic for arming the system also provides a delay between the turning on of the arm/disarm switch and the activation of the security system. This allows time for the owner to arm the system and exit the house. (There is a different timer that controls a low-level alert notification that allows the owner to disarm the system.)
- As shown in Figure 3-6, disarming the security system stops the notification and alarm sequence.

Before the security system has been armed, the LED flashes on and off if one of the zones is open. Figure 3-7 shows the control logic for using one of the SM bits (SM0.5) to generate the on/off pulse for the LED.

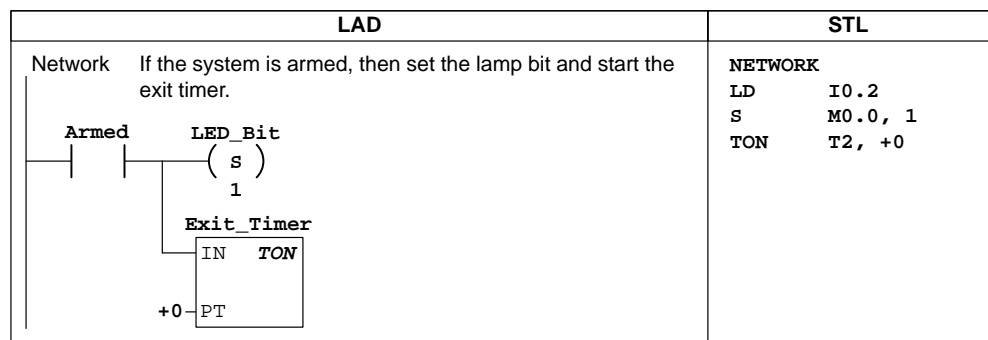


Figure 3-5 Control Logic for Arming the Security System

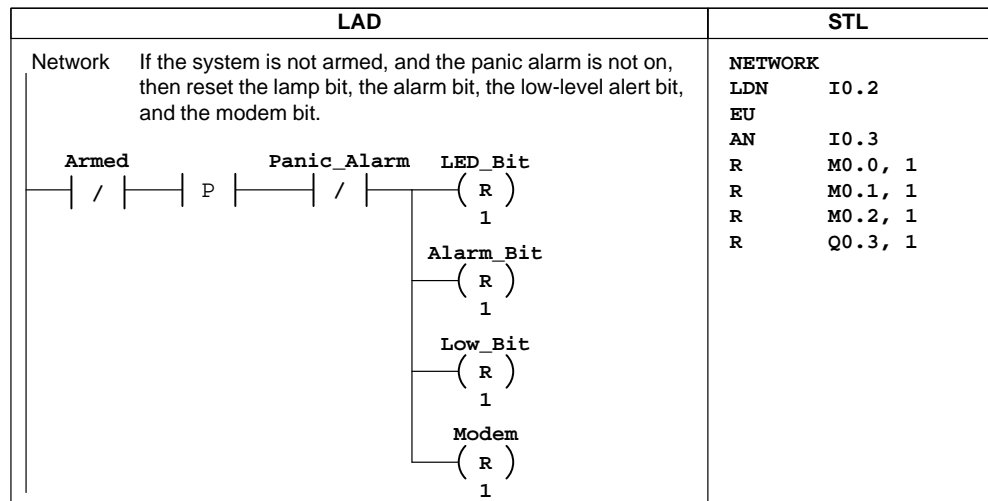


Figure 3-6 Control Logic for Disarming the Security System

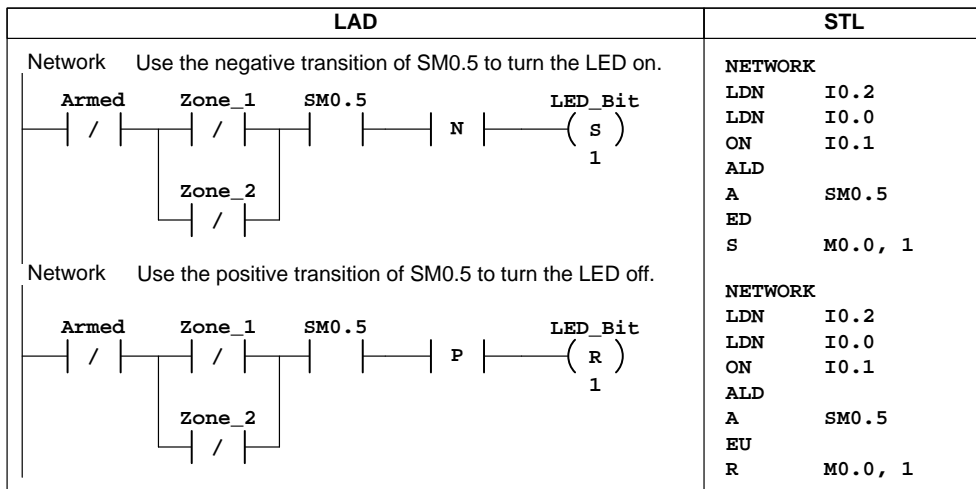


Figure 3-7 Control Logic for Flashing the LED On and Off

Designing the Control Logic for Turning On the Low-Level Alert Notification

On a breach of security (created when either Zone 1 or Zone 2 opens after the security system has been armed), the program turns on the low-level alert notification. This allows the owner a specified time to disarm the system (such as when re-entering the house). As shown in Figure 3-8, the program monitors the states of both zones and the arm/disarm switch. It also allows for the exit time (90 seconds).

After a breach of security has been identified, the program starts the timer for the low-level alert notification.

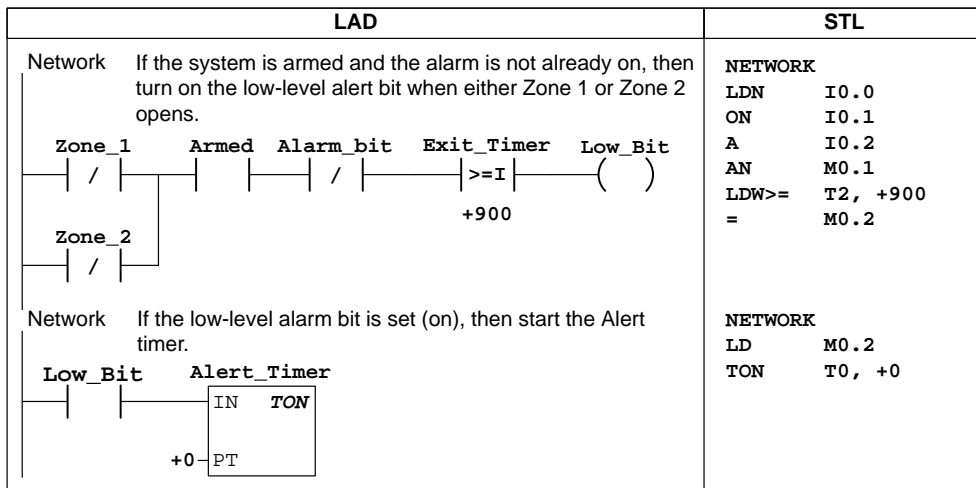


Figure 3-8 Control Logic for Turning On the Low-Level Alert Notification

Designing the Control Logic for Turning On the Alarm and Modem Dialer

Because the outputs turn on immediately, the program uses memory bits (M) for storing the results of the control logic. See Figure 3-9. At the end of the program, these bits turn on (or off) the outputs.

LAD	STL
<p>Network If the LED bit is on, turn on the output for the system LED.</p> <pre> graph LR LED_Bit[LED_Bit] --- LED[LED] style LED fill:none,stroke:none </pre>	<pre> NETWORK LD M0.0 = Q0.0 </pre>
<p>Network If the Alarm bit is on, turn on the output for the alarm.</p> <pre> graph LR Alarm_Bit[Alarm_Bit] --- Alarm[Alarm] style Alarm fill:none,stroke:none </pre>	<pre> NETWORK LD M0.1 = Q0.1 </pre>
<p>Network If the low-level alert bit is on, turn on the output for the low level alert.</p> <pre> graph LR Low_Bit[Low_Bit] --- Low_Alert[Low_Alert] style Low_Alert fill:none,stroke:none </pre>	<pre> NETWORK LD M0.2 = Q0.2 </pre>

Figure 3-9 Control Logic for Turning On the Outputs

As shown in Figure 3-10, the memory bits for the alarm siren and the modem dialer are turned on by either of two conditions:

- Someone pushes the “panic button” (regardless of the arm/disarm state of the system and without providing the low-level alert notification).
- The system has not been disarmed during the 60 seconds that the low-level alert notification has been on.

Turning on the alarm also resets the low-level alert notification.

LAD	STL
<p>Network</p> <pre> graph LR Panic_Alarm[Panic_Alarm] --- Alarm_Bit[Alarm_Bit] Alert_Timer[Alert_Timer] --- Armed[Armed] Alert_Timer --- Modem[Modem] Alert_Timer --- Low_Bit[Low_Bit] Armed --- Alarm_Bit Armed --- Modem Armed --- Low_Bit Alarm_Bit --- Alarm_Bit Modem --- Modem Low_Bit --- Low_Bit style Alarm_Bit fill:none,stroke:none style Modem fill:none,stroke:none style Low_Bit fill:none,stroke:none </pre>	<pre> NETWORK LD I0.3 LDW>= T0, +600 A I0.2 OLD S M0.1, 1 S Q0.3, 1 R M0.2, 1 </pre>

Figure 3-10 Control Logic for Enabling the Alarm and Modem Bits

3.3 Putting the Control Logic into a Program

After you have designed the control logic for your application, you arrange the instructions into a program. You can choose either STL or ladder for your program.

Figure 3-11 provides a listing of the ladder program for the sample program. This program includes the control logic from Section 3.2. The program ends with the End instruction.

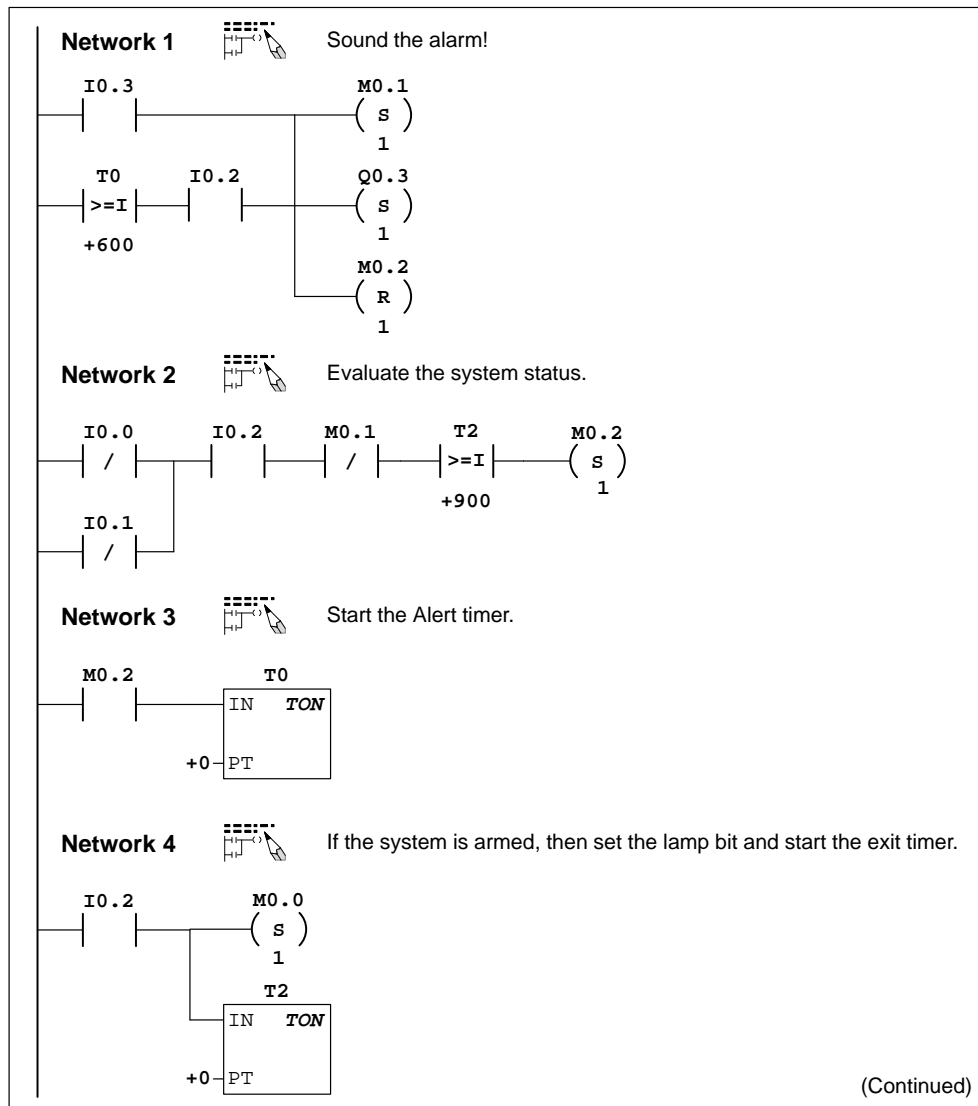


Figure 3-11 Home Security Program Example in Ladder

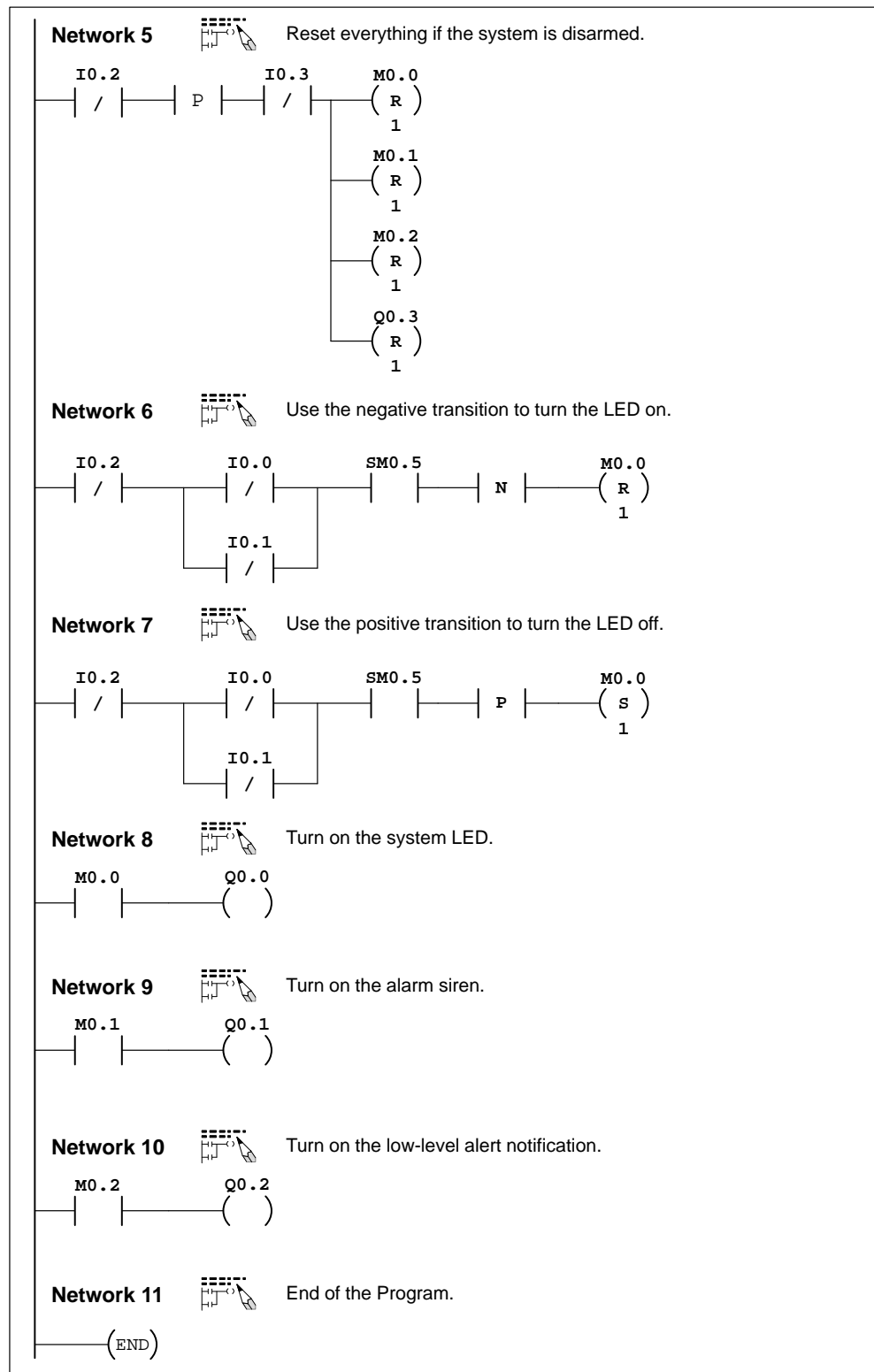


Figure 3-11 Home Security Program Example in Ladder, continued

Table 3-2 provides a listing of the STL program for the sample program. This program includes the control logic from Section 3.2. The program ends with the End instruction (MEND).

Table 3-2 Sample Program in Statement List

STL	Description
NETWORK 1	
LD I0.3	// if (the panic alarm has been turned on)
LDW>= T0, +600	// or (if the alert timer is >= 60 seconds
A I0.2	// and the system is armed)
OLD	// then
S M0.1, 1	// set High Level Alarm bit
S Q0.3, 1	// set Modem Dialer bit
R M0.2, 1	// reset Low Level Alarm bit
NETWORK 2	
LDN I0.0	// if zone 1 is open
ON I0.1	// or if zone 2 is open
A I0.2	// and the system is armed
AN M0.1	// and the high-level alarm bit is not set
AW>= T2, +900	// and the exit timer is less than 90 seconds
S M0.2, 1	// then set the low-level alert bit
NETWORK 3	
LD M0.2	// if the low-level alert bit has been set,
TON T0, +0	// then start the alert timer
NETWORK 4	
LD I0.2	// when the system is armed
S M0.0, 1	// set the LED bit
TON T2, +0	// and start the exit timer
NETWORK 5	
LDN I0.2	// if the system is not armed
EU	// and the panic alarm is not on
AN I0.3	// then
R M0.0, 1	// reset LED bit
R M0.1, 1	// reset High Level Alarm bit
R M0.2, 1	// reset Low Level Alarm bit
R Q0.3, 1	// reset Modem Dialer
NETWORK 6	
LDN I0.2	// if the system is not armed
LDN I0.0	// and if zone 1 is open
ON I0.1	// or zone 2 is open
ALD	// and
A SM0.5	// using the 1/2 second counter SM bit
ED	// on the negative edge
R M0.0, 1	// reset the LED bit
NETWORK 7	
LDN I0.2	// if the system is not armed
LDN I0.0	// and if zone 1 is open
ON I0.1	// or zone 2 is open
ALD	// and
A SM0.5	// using the 1/2 second counter SM bit
EU	// on the positive edge
S M0.0, 1	// set the LED bit
NETWORK 8	
LD M0.0	// if the LED bit has been set
= Q0.0	// turn on the LED output

Table 3-2 Sample Program in Statement List, continued

NETWORK 9		
LD	M0.1	// if the high-level alarm bit has been set
=	Q0.1	// turn on the high-level alarm output
NETWORK 10		
LD	M0.2	// if the low-level alert bit has been set
=	Q0.2	// turn on the low-level alert output
NETWORK 11		
MEND		// end of the main program

3.4 Creating a Project with STEP 7-Micro/WIN

To create a new project, select the menu command **Project ► New...**, as shown in Figure 3-12. The CPU Type dialog box is displayed. Select “PDS 210” from the drop-down list box.

You can name your project at any time; for this example, refer to Figure 3-13 and follow these steps to name the project:

1. Select the menu command **Project ► Save As...**
2. In the File Name field, type the following: `house.prj`
3. Click on the “OK” button.

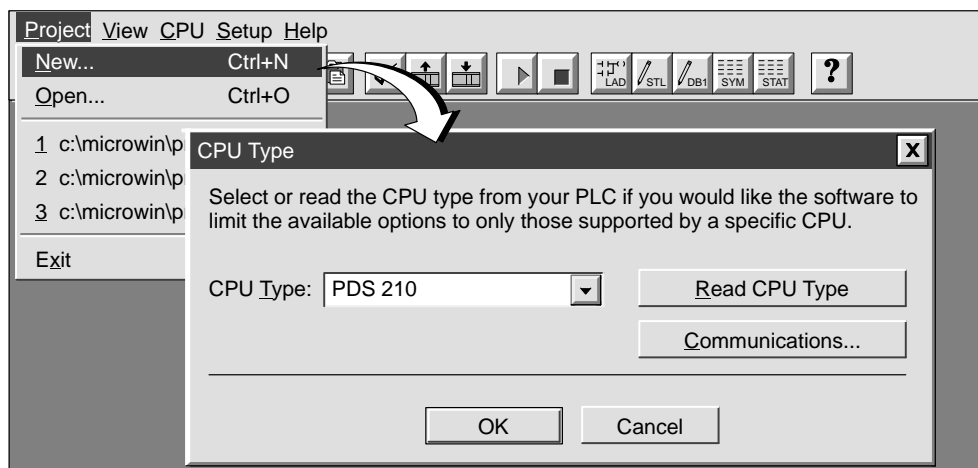


Figure 3-12 Creating a New Project and Selecting the CPU Type

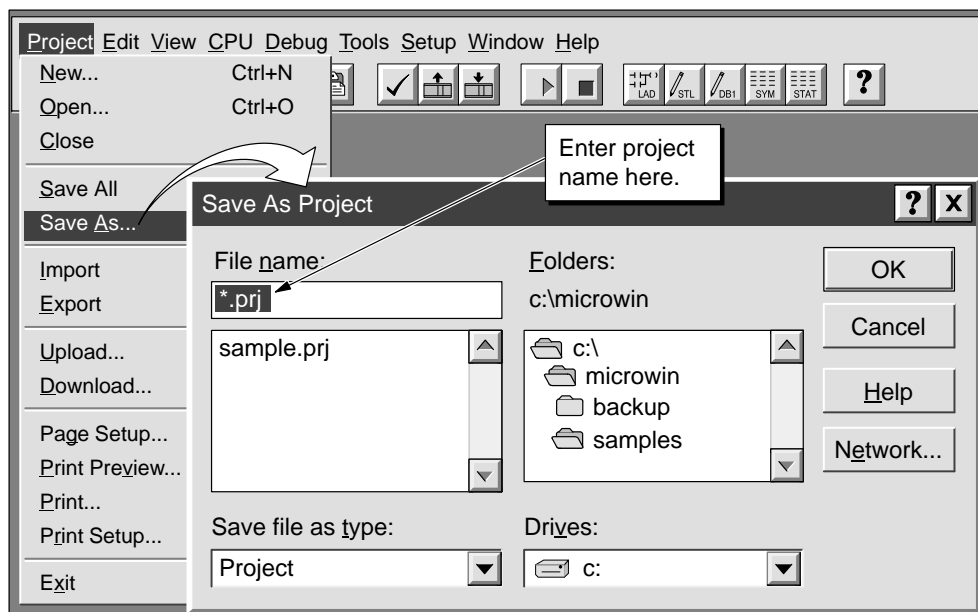


Figure 3-13 Naming the Sample Project

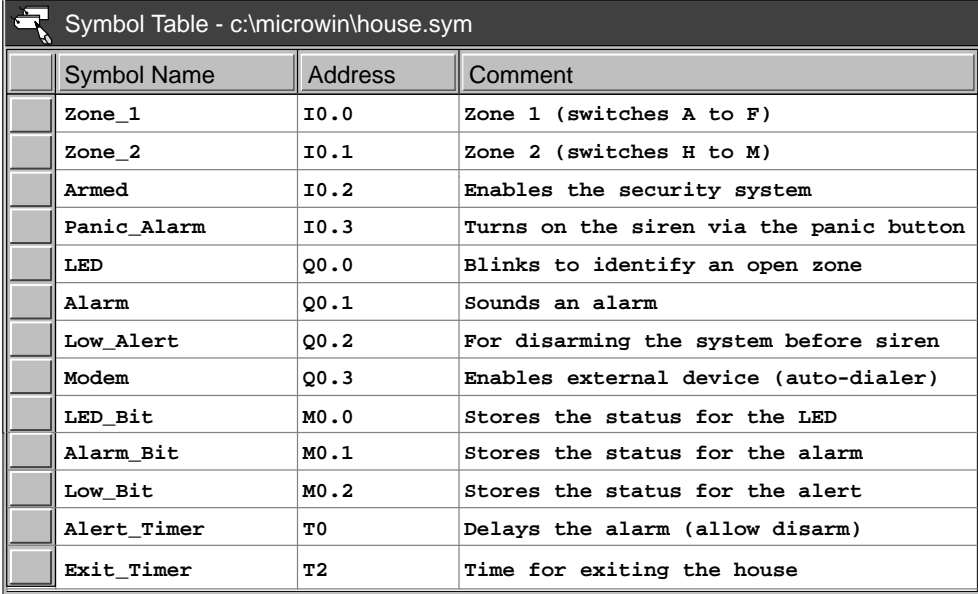
3.5 Creating a Symbol Table

To make programming easier, you can define symbolic names (or symbols) for memory addresses. You use the Symbol Table to define the set of symbols used to represent the absolute addresses in the sample program. To open the Symbol Table editor, double-click the icon, or click the restore or maximize button on the icon (in Windows 95). You can also select the menu command **View ► Symbol Table...**

Entering the Symbol Names

Figure 3-14 shows the list of symbol names and the corresponding addresses for the sample program. To enter the symbol names, follow these steps:

1. Select the first cell in the Symbol Name column, and type the following: `Zone_1`
2. Press ENTER to move the focus to the first cell in the Address column. Type the address `I0.0` and press ENTER. The focus moves to the cell in the Comment column. (Comments are optional, but they are a useful way to document the elements in your program.)
3. Press ENTER to start the next symbol row, and repeat these steps for each of the remaining symbol names and addresses.
4. Use the menu command **Project ► Save All** to save your Symbol Table.



Symbol Name	Address	Comment
Zone_1	I0.0	Zone 1 (switches A to F)
Zone_2	I0.1	Zone 2 (switches H to M)
Armed	I0.2	Enables the security system
Panic_Alarm	I0.3	Turns on the siren via the panic button
LED	Q0.0	Blinks to identify an open zone
Alarm	Q0.1	Sounds an alarm
Low_Alert	Q0.2	For disarming the system before siren
Modem	Q0.3	Enables external device (auto-dialer)
LED_Bit	M0.0	Stores the status for the LED
Alarm_Bit	M0.1	Stores the status for the alarm
Low_Bit	M0.2	Stores the status for the alert
Alert_Timer	T0	Delays the alarm (allow disarm)
Exit_Timer	T2	Time for exiting the house

Figure 3-14 Symbol Table for the Sample Program

3.6 Entering the Program

You can enter the program in either statement list (STL) or ladder. You can also use either absolute addressing or symbolic addressing.

To enter the program in STL, you open the STL Editor window and start typing the instructions. (You use the **View ▶ STL** menu command to change the editor from ladder to STL.) Remember to put double slashes (//) before any comments and to end each line by pressing ENTER.

To enter the program in STL, type the commands (with or without the comments) listed in Table 3-2. You can cut, copy, and paste in the STL Editor. STEP 7-Micro/WIN also includes search-and-replace functions.

Programming with Symbolic Addresses

Before you start entering your program, make sure the ladder view is set for symbolic addressing. Use the menu command **View ▶ Symbolic Addressing** and look for a check mark next to the menu item, which indicates that symbolic addressing is enabled.

Note

Symbol names are case-sensitive. The name you enter must match exactly the uppercase and lowercase characters entered in the symbol table. If there is any mismatch, the cursor stays on the element and displays the "Add Symbol" dialog. You can then add the new symbol to the Symbol Table, or cancel and correct the entry.

Using the Ladder Editor to Enter the Program

To access the Ladder Editor, double-click the icon at the bottom of the main window. (You use the **View ▶ Ladder** menu command to change the editor from STL to ladder.) Figure 3-15 shows some of the basic tools you will use in the Ladder Editor.

Refer to Figure 3-11 for the program listing in ladder. Entering the comments is optional.

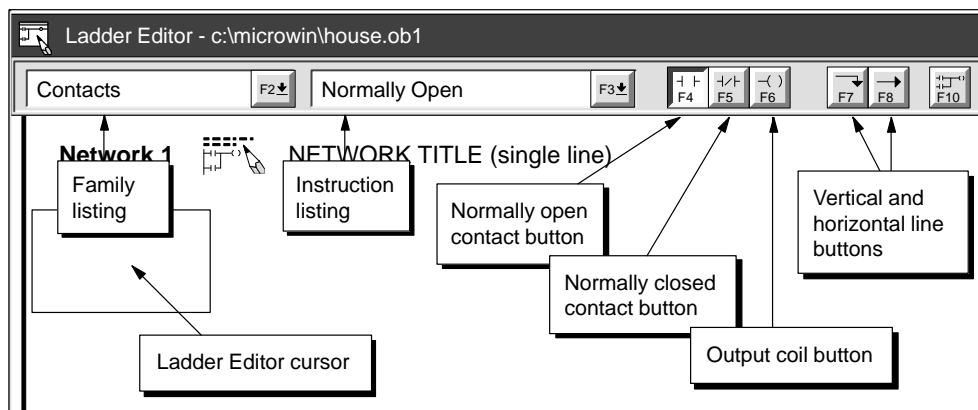


Figure 3-15 Some of the Basic Ladder Editor Tools

Refer to Figure 3-16 and follow these steps to enter the first network of the sample program:

1. Click the mouse cursor in the left-most position below the network title. Enter a normally open contact by clicking the F4 toolbar button or by selecting "Contacts" from the family listing and then selecting "Normally Open" from the instruction listing. A normally open contact appears with the name "Zone_1" highlighted above it. (Every time you enter a contact, the software displays the default address of I0.0, which in this example is defined as "Zone_1" in the Symbol Table.)
2. "Panic_Alarm" is the first element required for Network 1. While "Zone_1" is highlighted, type either the symbol name **Panic_Alarm** or the absolute address I0.3 (the software accepts entry of either form).
3. Press ENTER to confirm the first element. The symbol name "Panic_Alarm" is displayed. The ladder cursor moves to the second column position.
4. Click the F8 toolbar button to insert a horizontal line. (You can also select "Lines" from the family listing and then select "Horizontal" from the instruction listing.)

To change or replace one of the elements, move the cursor to that element and select the new element. You can also cut, copy, or paste elements at the cursor location.

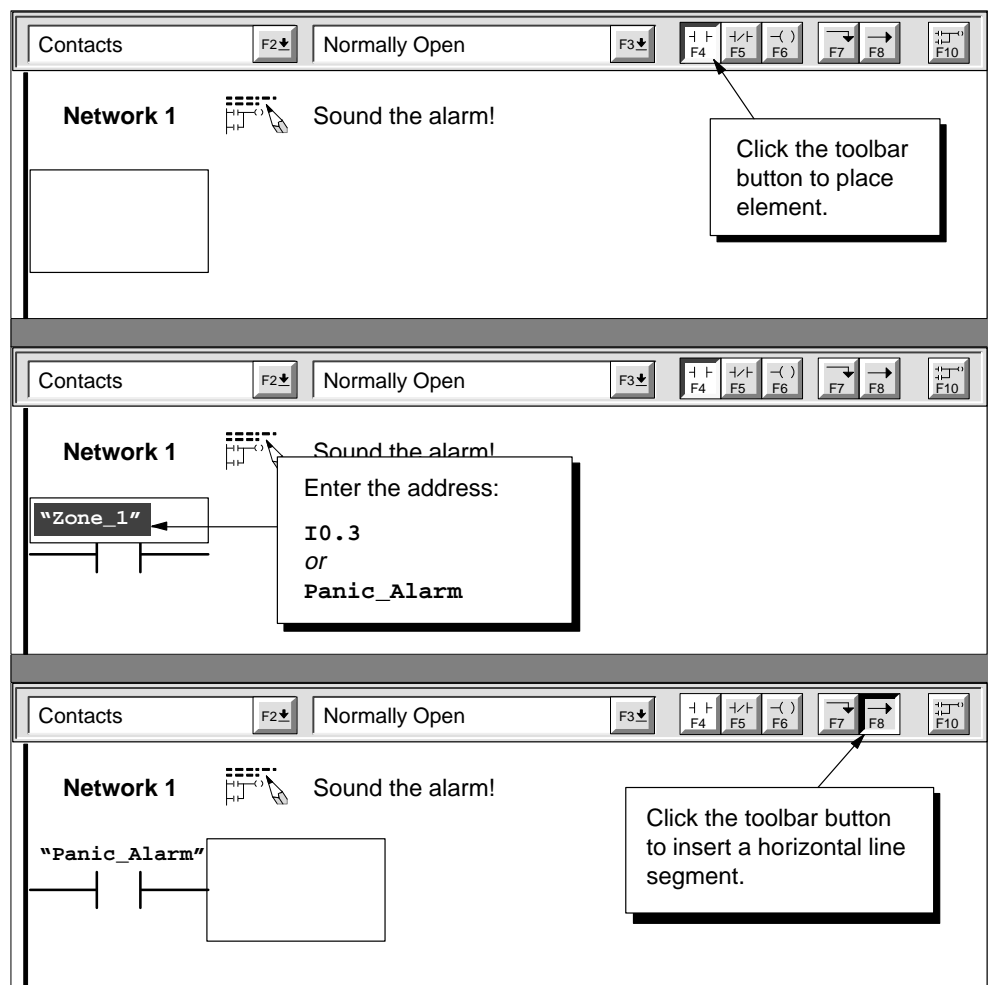


Figure 3-16 Entering the First Contact of the First Network

Refer to Figure 3-17 and follow these steps to enter the output coil that ends the first rung of the first network:

1. Select "Output Coils" from the family listing and select "Set" from the instruction listing.
2. Type either the symbolic name `Alarm_Bit` or the absolute address `M0.1` in the highlighted area.
3. Pressing ENTER highlights the "number of points" field (located underneath the output coil). Press ENTER to accept the default value of 1. (The CPU 210 allows only one point to be set or reset by any one Set or Reset instruction.)
4. Move the cursor to the position below the first contact.

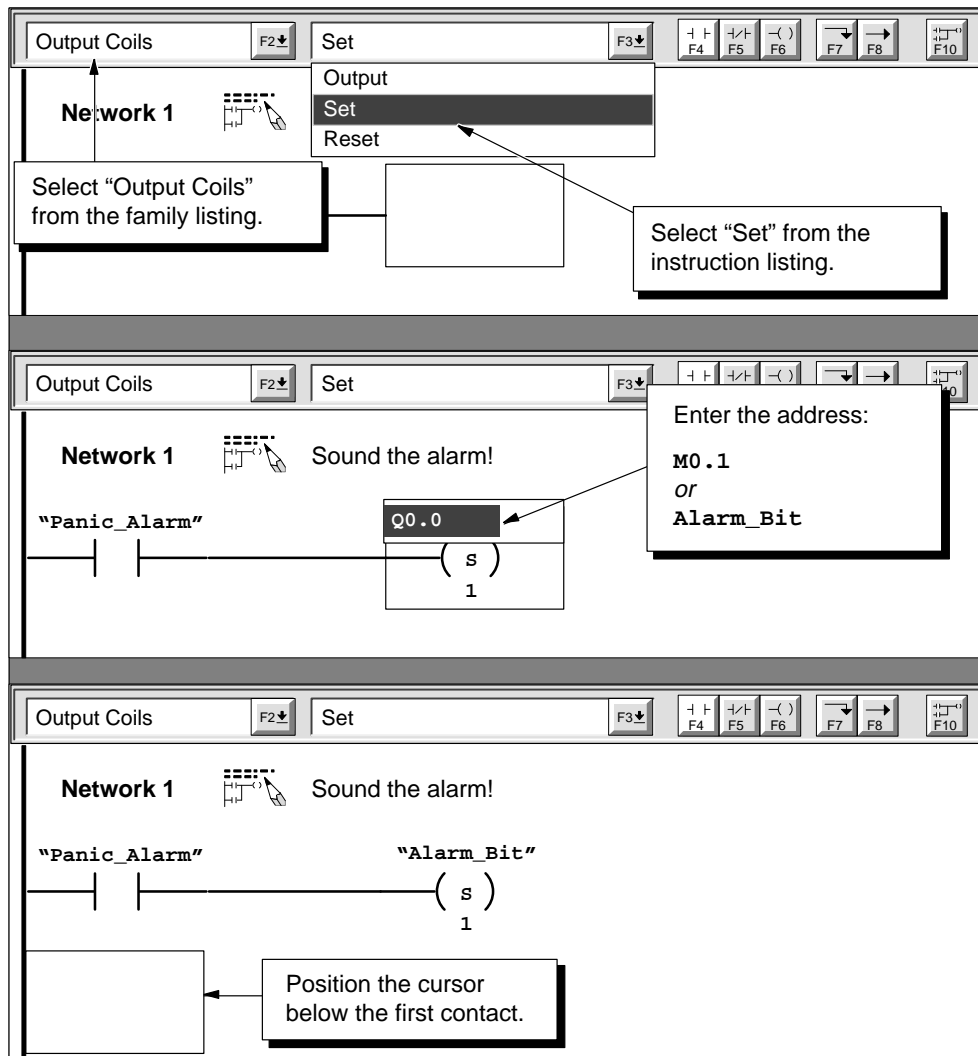


Figure 3-17 Entering the Output Coil

Refer to Figure 3-18 and follow these steps to enter the two contacts on the second rung of the first network:

1. Select "Contacts" from the family listing and select ">= Integer" from the instruction listing. This inserts a comparison instruction at the cursor position. This instruction compares the value of the notification timer (Alert_Timer) with the time value.
2. Type either the symbolic name **Alert_Timer** or the absolute address **T0** in the highlighted area. Pressing ENTER highlights the second value for the comparison.
3. Type **600** and press ENTER. This instruction becomes true (and turns on) when the timer is greater than or equal to 600, which equals 60 seconds.
4. Click the F4 toolbar button to create a normally open contact. Type **Armed** (or **I0.2**) and press ENTER.

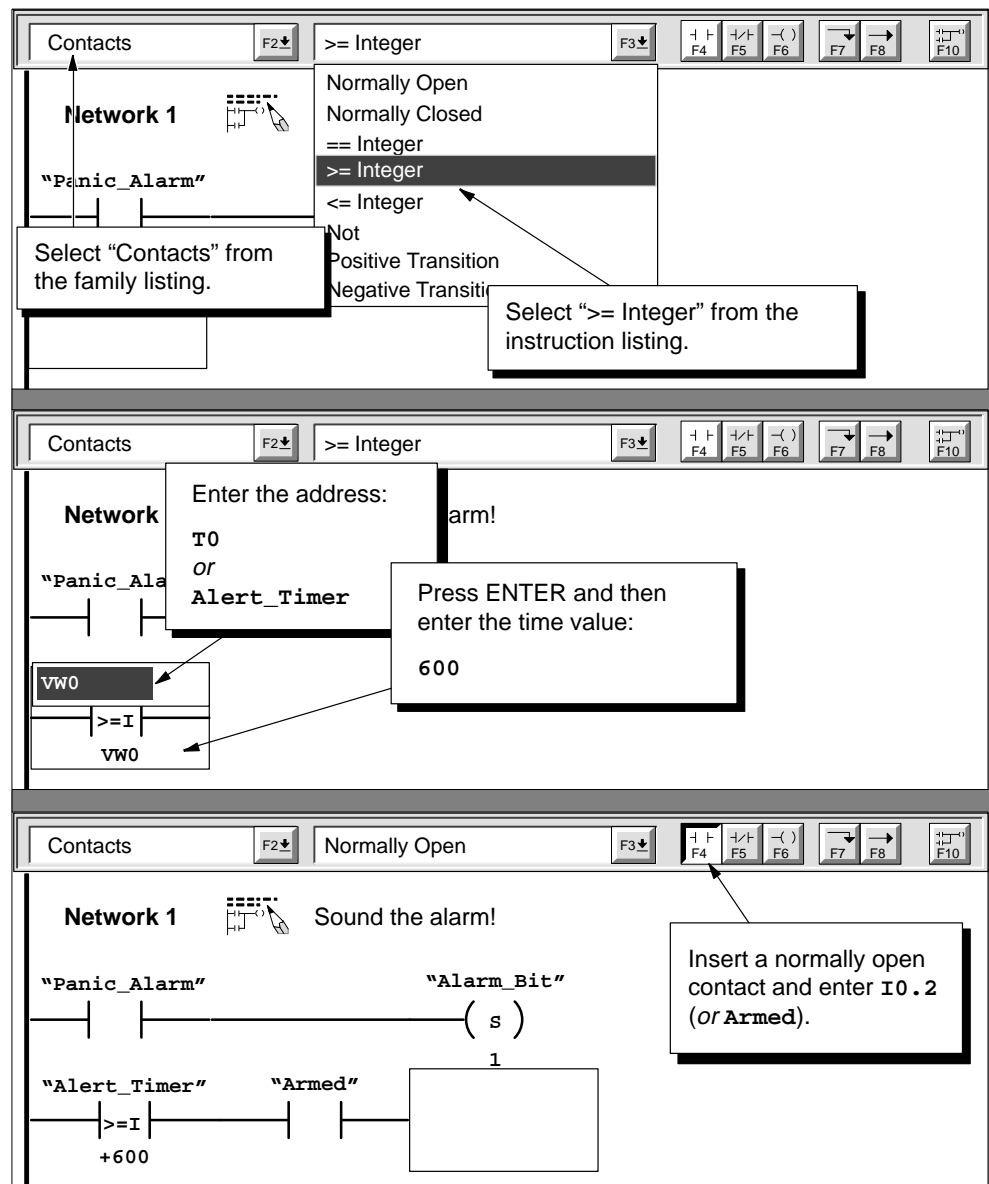


Figure 3-18 Entering the Comparison Instruction and Next Contact

Refer to Figure 3-19 and follow these steps to enter a vertical line and to copy the output coil from the first rung:

1. Move the cursor to the horizontal line above the contact for "Armed" (or I0.2). Click the F7 toolbar button to insert a vertical line that connects the first rung with the second rung.
2. Move the cursor to the output coil on the first rung. Use the menu command **Edit ▶ Copy** to copy the output coil to the clipboard.
3. Move the cursor down and use the menu command **Edit ▶ Paste** to paste the output coil. Type **Modem** (or **Q0.3**) in the highlighted field and press ENTER. Press ENTER again to accept the default value of 1.

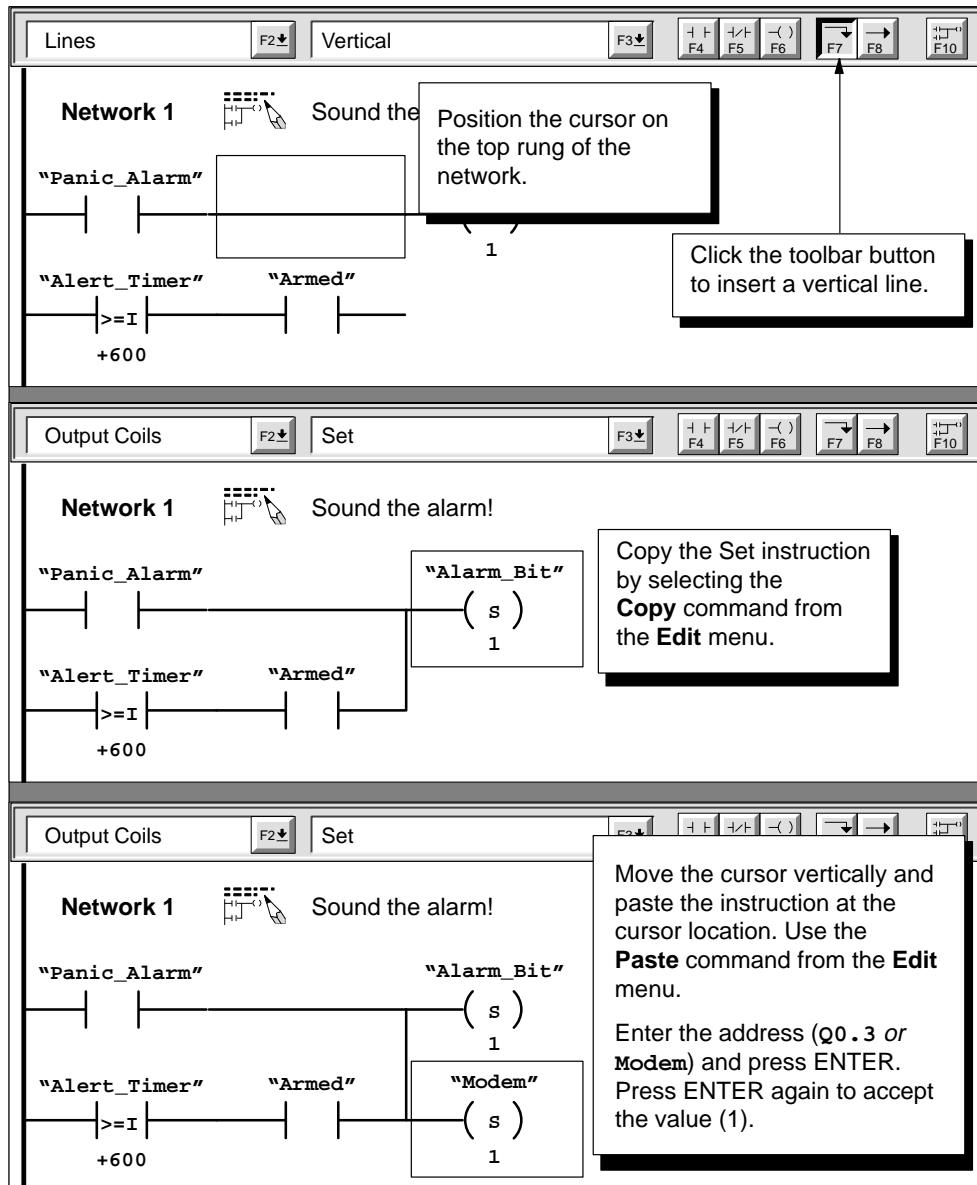


Figure 3-19 Entering a Vertical Line and Copying the Output Coil

Figure 3-20 shows the remaining steps for finishing the first network. After you have entered the first network, move the cursor to the second network. Refer to Figure 3-11 and enter the remaining networks of the sample program.

The figure consists of three vertically stacked screenshots of a PLC programming software interface, illustrating the steps to complete Network 1.


Top Screenshot: The 'Lines' menu is open, and 'Vertical' is selected. A callout box says: "Position the cursor over the contact for 'Armed' (or I0.2)." Another callout points to the F7 toolbar button, saying: "Click the toolbar button to insert a vertical line." The ladder logic shows Network 1 with contacts for "Panic_Alarm", "Alert_Timer" (with a timer value of +600), and "Armed". The output coil is "Alarm" (S) with address 1.

Middle Screenshot: The 'Output Coils' menu is open, and 'Reset' is selected. A callout box says: "Select 'Output Coils' from the family listing." Another callout points to the 'Reset' option in the instruction listing, saying: "Select 'Reset' from the instruction listing." The ladder logic shows the same inputs, but the output coil is now "Alarm Bit" (S) with address 1.

Bottom Screenshot: The network is complete. A callout box says: "Enter the address (M0.2 or Low_Bit) and the value (1). The first network is now complete." The ladder logic shows the same inputs, but the output coil is now "Low_Bit" (R) with address 1.


Figure 3-20 Completing the First Network

Compiling the Program

After completing the sample program, check the syntax by selecting the menu command **CPU ► Compile** or by clicking the Compile button: 

If you have entered all the networks correctly as shown in the sample program, you will get a "Compile Successful" message that also includes information on the number of networks and the amount of memory used by the program. Otherwise, the Compile message will indicate which networks contain errors.

Saving the Sample Program

Saving the project saves all the components of your sample project. You save your project by selecting the menu command **Project ► Save All** or by clicking the Save button: 

3.7 Creating a Status Chart

To monitor the status of selected elements in the sample program, you create a Status Chart containing the elements you want to monitor while running the program. You can use the Status Chart to monitor and modify the program as it runs on your PDS 210; however, you cannot monitor the status of a program running on a CPU 210.

STEP 7-Micro/WIN provides an easy method for creating a Status Chart: simply copy any or all of the elements in the Symbol Table and paste them into a Status Chart.

Building a Status Chart

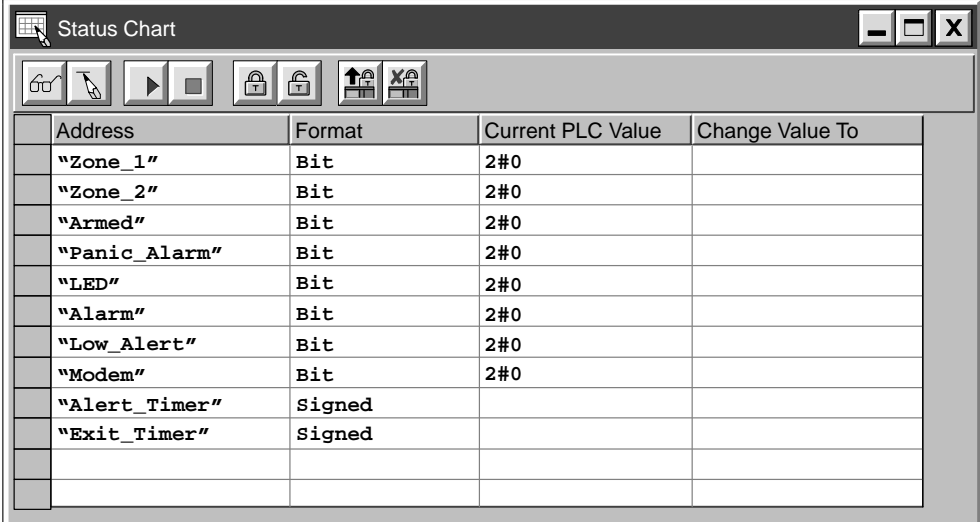
To access the Status Chart editor, double-click the icon at the bottom of the main window. Then enter the elements for the sample program by following these steps:

1. Select the first cell in the Address column, and type the following: `zone_1`
Press ENTER to confirm your entry. This element type can only be displayed in bit format (either 1 or 0) so you cannot change the format type.

2. Select the next row, and repeat these steps for each of the remaining elements, as shown in Figure 3-21.


You can use the menu command **Edit ► Insert Row(s)** (or the INSERT or INS key) to insert a blank row above the row containing the cursor.

Timers and counters can be displayed in other formats. With the focus in the Format column cell, press the SPACEBAR to cycle through the formats that are valid for these element types. For this example, select Signed for the timers.



Address	Format	Current PLC Value	Change Value To
"Zone_1"	Bit	2#0	
"Zone_2"	Bit	2#0	
"Armed"	Bit	2#0	
"Panic_Alarm"	Bit	2#0	
"LED"	Bit	2#0	
"Alarm"	Bit	2#0	
"Low_Alert"	Bit	2#0	
"Modem"	Bit	2#0	
"Alert_Timer"	Signed		
"Exit_Timer"	Signed		

Figure 3-21 Status Chart for the Sample Program

Save your Status Chart by selecting the menu command **Project ► Save All** or by clicking the Save button: 

3.8 Downloading and Monitoring the Sample Program

Once you have downloaded your program to the PDS 210, you can use the Debug features to monitor or debug the operation of your program.

Downloading the Project to the PDS 210

The PDS 210 must be in STOP mode for you to download a program. To download your program, select the menu command **Project ► Download...**. An information message tells you whether or not the download operation was successful.

Note

STEP 7-Micro/WIN does not verify that your program uses memory or I/O addresses that are valid for the PDS 210 or the CPU 210. If you attempt to download a program that uses invalid addresses or program instructions that are not supported by the PDS 210, the PDS 210 rejects the attempt to download the program and displays an error message.

You must ensure that all memory locations, I/O addresses, and instructions used by your program are valid for the PDS 210 and CPU 210.

Using the Ladder Editor to Monitor the Status of the Program

Ladder status shows the current state of events in your program. Reopen the Ladder Editor window, if necessary, and select the menu command **Debug ► Ladder Status On**.

If you have an input simulator connected to the input terminals on your CPU, you can turn on switches to see power flow and logic execution. For example, if you turn on switch **I0.2**, the power flow for Network 1 will be complete when timer **T0** is greater than or equal to 600. The network will look like the one shown in Figure 3-22: **M0.1** and **Q0.3** are set to 1, and **M0.2** is reset to 0.

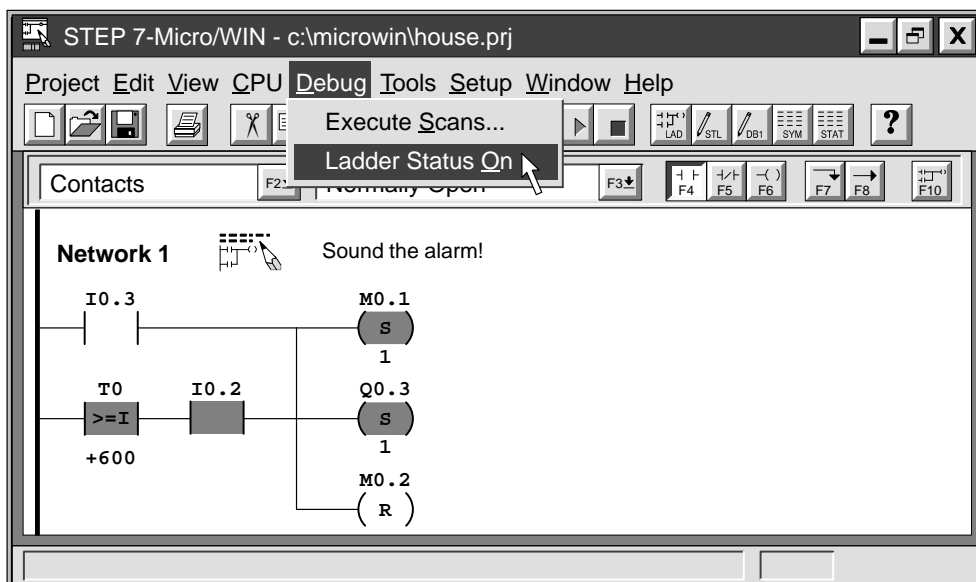





Figure 3-22 Monitoring Status of the First Network

Using the Status Chart to Monitor and Modify the Current Values of the Program

You can use the Status Chart to monitor or modify the current values of any I/O points or memory locations. Reopen the Status Chart window, if necessary, and select the menu command **Debug ► Chart Status On**, as shown in Figure 3-23. As you switch inputs on or off with the CPU in RUN mode, the Status Chart shows the current status of each element.

- To view the current value of the elements in your program, click the Single Read button  or the Continuous Read button  in the Status Chart window.
- To stop the reading of status, click the Stop button  in the Status Chart window.

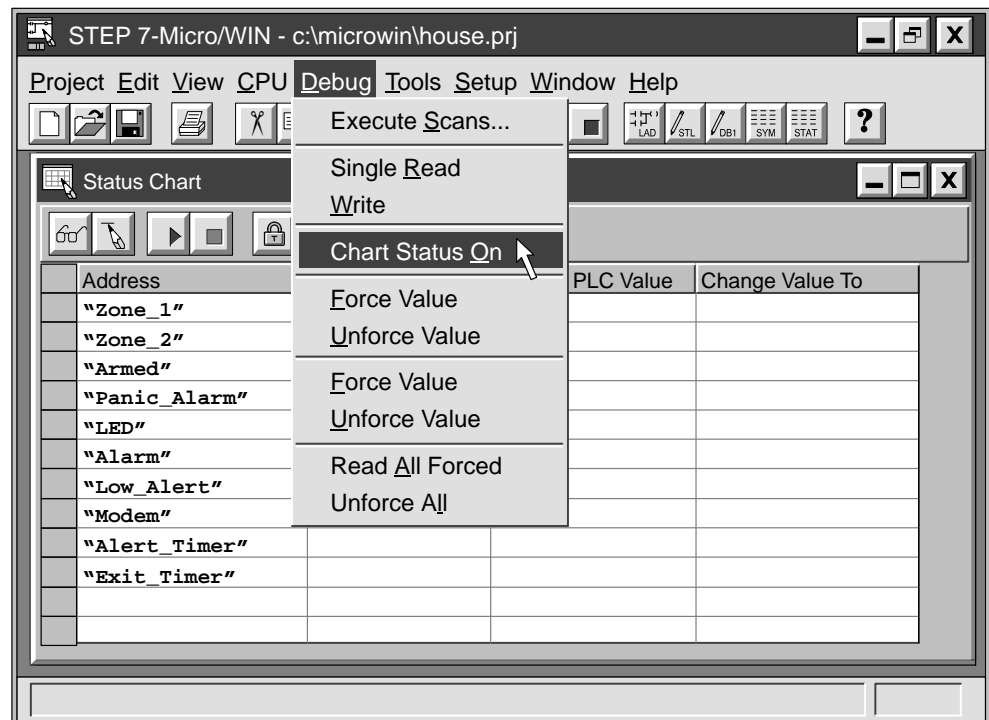


Figure 3-23 Monitoring the Status Chart of the Sample Program

3.9 Modifying the Sample Program

You can use the following networks of control logic to modify the sample program. These networks provide the following enhancements to the sample program:

- If Zone 1 is open, the LED flashes one time.
- If Zone 2 is open, the LED flashes two times.
- If both zones are open, the LED flashes three times (one short flash, followed by a pause and then two short flashes).

The modified program uses the memory addresses shown in Table 3-3. If you used symbolic addressing with your program, add these symbolic names and addresses to the Symbol Table.

Table 3-3 Memory Locations Used to Modify the Sample Program

Element	Address	Symbolic Name	Description
Internal Memory	M0.7	Blink_Bit	Stores the status for the LED
	MW1	Step_Counter	Tracks the blinking of the LED
	MW3	Blink_Pattern	Stores the pattern for flashing the LED on and off
Timers	T1	Blink_Timer	Increments the step counter

Creating the Blink Patterns for the LED

The program uses different bit patterns as the basis for the logic that turns the LED on and off. Based on the condition, the program loads a value into the word that stores the blink pattern. Figure 3-24 shows the networks that move the bit patterns into MW3. Use STEP 7-Micro/WIN to enter the networks into the program.

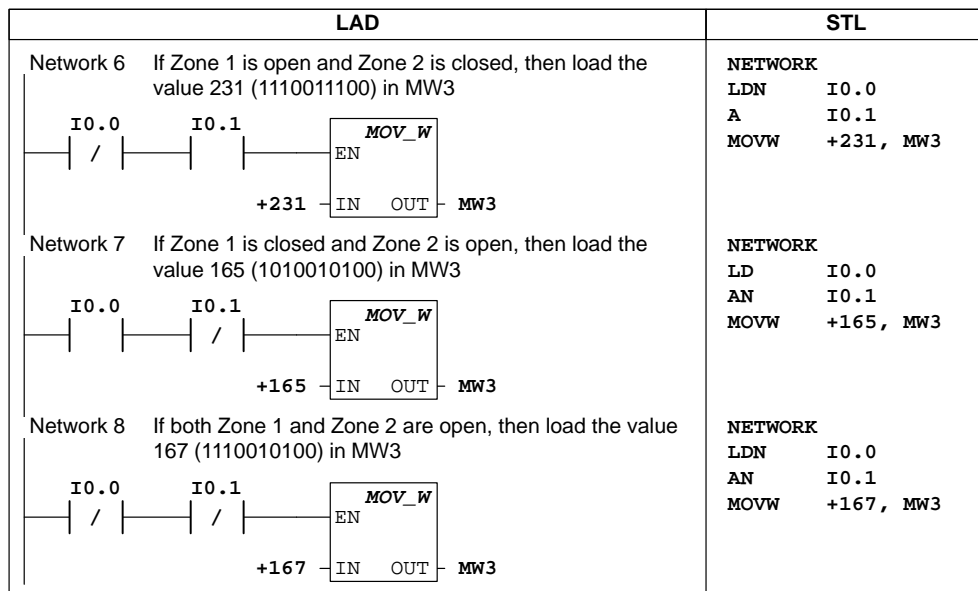


Figure 3-24 Control Logic for Lamp Operations

Turning the LED On and Off

The program uses a timer (T1) and the pattern stored in MW3 to turn the LED on and off. The program increments MW1 to count the number of passes through the control logic for flashing the lights; after 10 passes, the MW1 is reset to 0.

Figure 3-25 shows the control logic for starting the timer. The timer starts when the system is armed, the light-blink timer bit is not set, and either Zone 1 or Zone 2 opens.

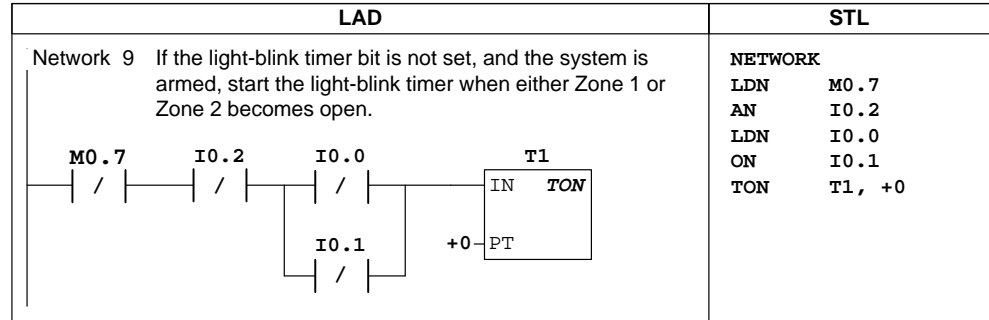


Figure 3-25 Control Logic for Starting the Light-Blink Timer

Figure 3-26 shows the control logic for incrementing the count for the number of times that the light-blink logic has been performed.

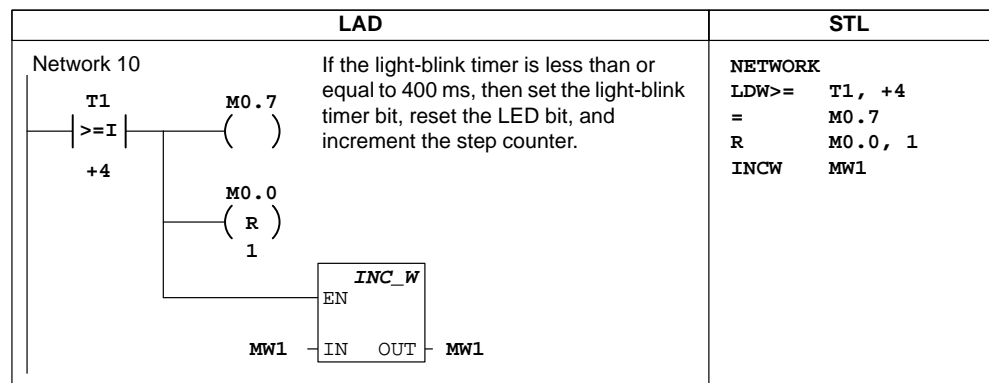


Figure 3-26 Control Logic for Setting the Timer Bit and Incrementing the Counter

Figure 3-27 shows the control logic for turning the LED on and off. Each pass through the light-blink logic evaluates a different bit of MW3 (M4.0 to M4.7). Based on the pattern loaded (see Figure 3-24), the LED turns on or off.

Figure 3-28 shows the control logic for resetting the count.

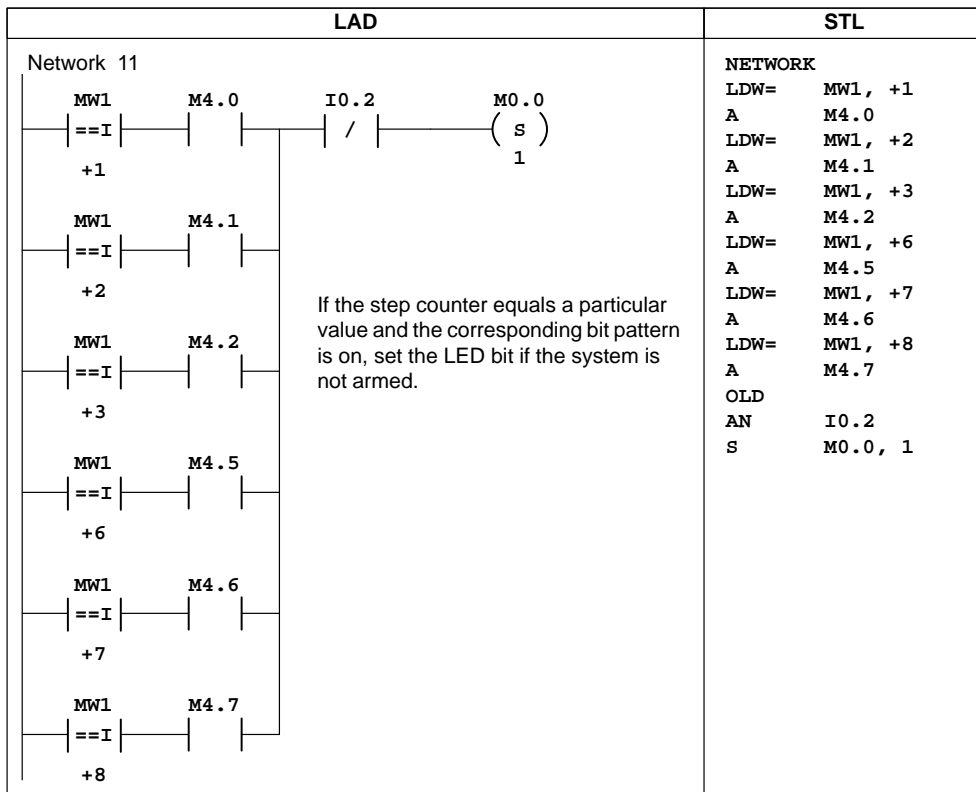


Figure 3-27 Control Logic for Controlling the Blink Pattern

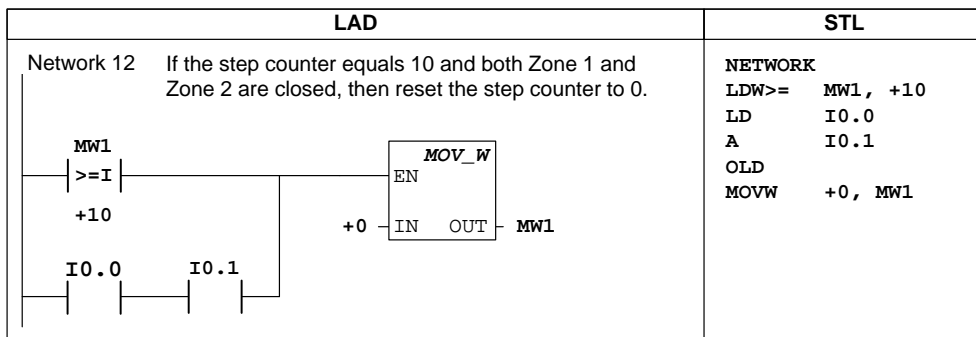


Figure 3-28 Control Logic for Resetting the Counter

Basic Concepts for Programming the CPU 210

4

Before you start to program your application using the CPU 210, you should become familiar with some of its basic operational features.

Chapter Overview

Section	Description	Page
4.1	Guidelines for Designing a Micro PLC System	4-2
4.2	Concepts for Creating a Program	4-4
4.3	Understanding the Scan Cycle of the CPU 210	4-6
4.4	Understanding the Programming Languages	4-9
4.5	Understanding the Addresses of the Memory Areas	4-11
4.6	Sample Program Using an Interrupt Routine	4-14
4.7	Using the Analog Adjustment Potentiometer	4-16

4.1 Guidelines for Designing a Micro PLC System

There are many methods for designing a Micro PLC system. This section provides some general guidelines that can apply to many design projects. Of course, you must follow the directives of your own company's procedures and of the accepted practices of your own training and location. Figure 4-1 shows some of the basic steps in the design process.

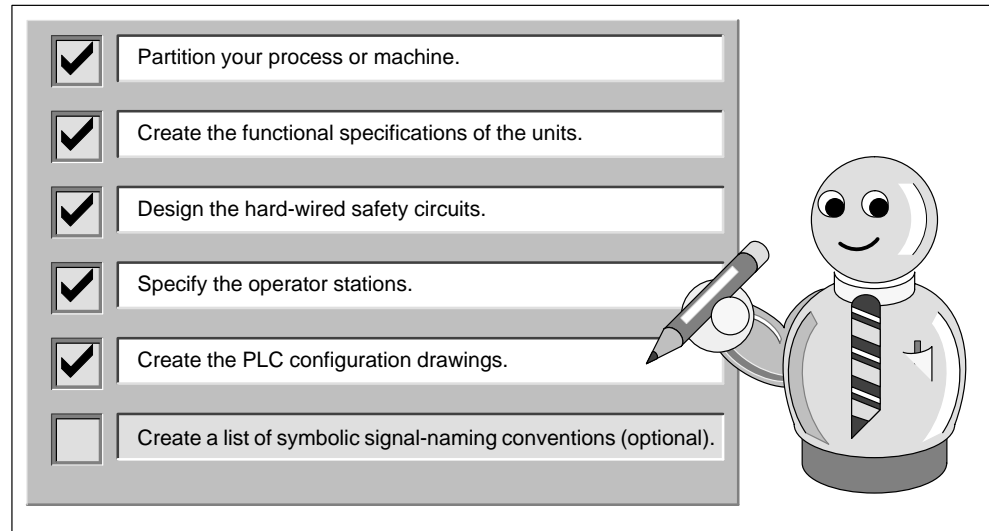


Figure 4-1 Basic Steps for Planning a PLC System

Partitioning Your Process or Machine

Divide your process or machine into sections that have a level of independence from each other. These partitions will determine the boundaries between controllers and will influence the functional description specifications and the assignment of resources.

Creating the Functional Specifications

Write the descriptions of operation for each section of the process or machine. Include the following topics:

- Input/output (I/O) points
- Functional description of the operation
- Permissives (states that must be achieved before allowing action) for each actuator (solenoids, motors, drives, etc.)
- Description of the operator interface
- Interfaces with other sections of the process or machine

Designing the Safety Circuits

Identify equipment requiring hard-wired logic for safety. Control devices can fail in an unsafe manner, producing unexpected startup or change in the operation of machinery. Where unexpected or incorrect operation of the machinery could result in physical injury to people or significant property damage, consideration should be given to the use of electro-mechanical overrides which operate independently of the CPU 210 to prevent unsafe operations.

The following tasks should be included in the design of safety circuits:

- Identify improper or unexpected operation of actuators that could be hazardous.
- Identify the conditions that would assure the operation is not hazardous, and determine how to detect these conditions independently of the CPU 210.
- Identify how the CPU 210 and its I/O will affect the process when power is applied and removed, and on detected errors. This information should only be used for designing for the normal and expected off-normal operation, and should not be relied on for safety purposes.
- Design manual or electro-mechanical safety overrides that will block the hazardous operation independent of the CPU.
- Provide appropriate status information from the independent circuits to the CPU 210 so that the program and any operator interfaces have necessary information.
- Identify any other safety-related requirements for safe operation of the process.

Specifying the Operator Stations

Based on the requirements of the functional specifications, create drawings of the operator station. Include the following items:

- Overview showing the location of each operator station in relation to the process or machine
- Mechanical layout of the devices (display, switches, lights, etc.) for the operator station
- Electrical drawings with the associated I/O of the CPU 210

Creating the PLC Configuration Drawings

Based on the requirements of the functional specification, create configuration drawings of the control equipment. Include the following items:

- Overview showing the location of the CPU in relation to the process or machine
- Mechanical layout of the CPU 210 (including cabinets and other equipment)
- Electrical drawings for each CPU 210 (including the device model numbers and I/O addresses)

Creating a List of Symbolic Names

If you choose to use symbolic names for addressing, create a list of symbolic names for the absolute addresses. Include not only the physical I/O signals, but also the other elements that will be used in your program.

4.2 Concepts for Creating a Program

Relating the Program to Inputs and Outputs

Figure 4-2 shows a simple diagram of how an electrical relay diagram relates to the CPU 210. In this example, the state of the operator panel switch for opening the drain is added to the states of other inputs. The calculations of these states then determine the state for the output that goes to the solenoid that closes the drain. The CPU continuously cycles through the program, reading and writing data.

The execution of the program follows a simplified flow of information: the state of the physical input is copied to the memory area → the CPU 210 executes the program → whenever the program modifies an output, the CPU 210 immediately updates the physical output. Each memory area is assigned a mnemonic identifier (such as “I” for input) that is used for accessing the data stored in that area of memory.

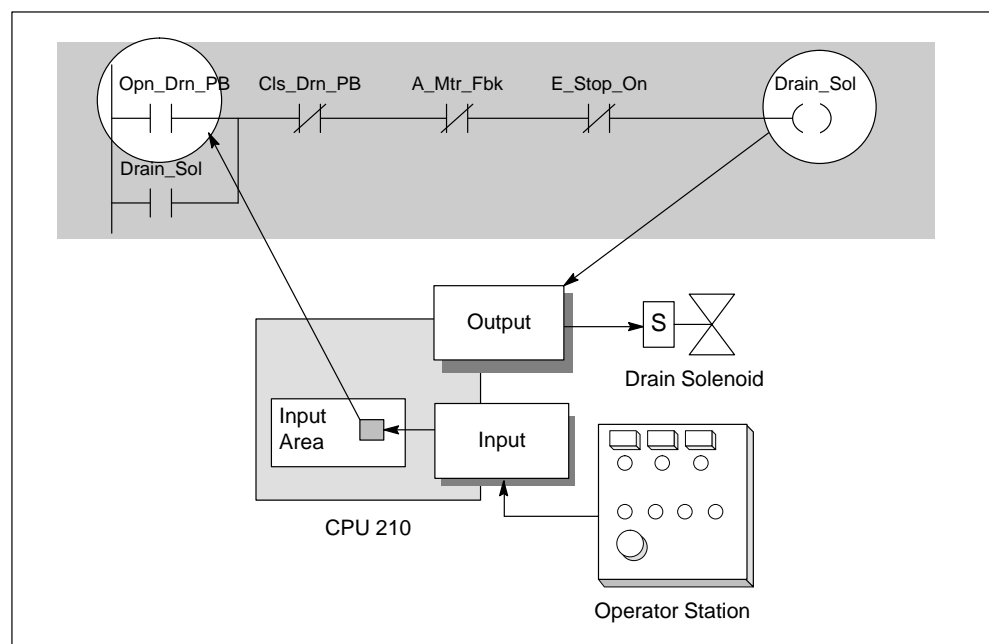


Figure 4-2 Relating the Program to Inputs and Outputs

Accessing Data in the Memory Areas

All of the memory areas of the CPU 210 have “absolute” addresses. You access a specific location by entering an address (such as $I0.0$ for the first input point). An absolute address for a memory area includes not only the area identifier (such as “M”), but also the size of data to be accessed: W (word: two bytes). (The CPU 210 provides 3 words or 48 bits for the M memory area.) The absolute address also includes a numeric value: either the number of bytes from the beginning of the memory area (offset) or the device number. (This value is dependent on the area identifier. See Section 4.5.)

Organizing the Program

As shown in Figure 4-3, a program for the CPU 210 is structured into the following organizational elements: the main program and an optional hardware interrupt routine.

- The main program stores the instructions that control your application. The instructions in the main program are executed sequentially once per scan of the CPU. To terminate the main program, use an Unconditional End coil in ladder or a main program end instruction (MEND) in STL.
- The CPU 210 also allows one optional hardware interrupt routine. If you use the interrupt routine in your program, the CPU executes these instructions on a specific hardware event (the rising edge when $\text{I}0.0$ turns on). Place the interrupt routine after the end of the main program (following the Unconditional End (MEND) instruction). Use a Return From Interrupt (RETI) instruction to terminate the interrupt routine.

Section 4.6 provides an example of a program using an interrupt routine. The interrupt routine is not executed as part of the normal scan cycle, but is executed when the interrupt event occurs (which may be at any point in the scan cycle).

For additional information about designing and entering a program, see the sample application in Chapter 3.

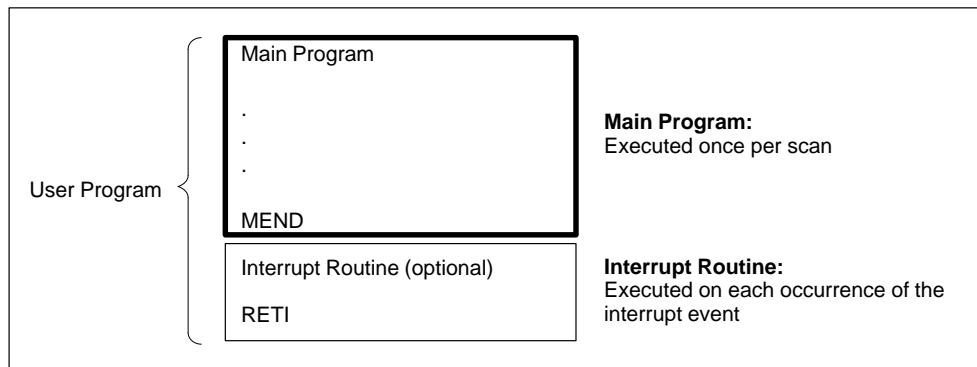


Figure 4-3 Program Structure for the CPU 210

4.3 Understanding the Scan Cycle of the CPU 210

The basic operation of the CPU 210 is very simple:

- The CPU reads the status of the inputs.
- The program that is stored in the CPU uses the inputs to evaluate the control logic. As the program runs, the CPU writes the data to the outputs.

The CPU 210 continuously executes your program. If your program uses the optional hardware interrupt routine, the interrupt routine can run anytime after the CPU executes the Enable Interrupt (ENI) instruction in the main program segment.

Understanding the Basic Scan Cycle of the CPU 210

The CPU 210 executes your program in a continuous, cyclical series of tasks called a scan. As shown in Figure 4-4, the CPU 210 performs the following tasks during the scan cycle:

1. During the first scan (after power is turned on) only, the CPU 210 clears the outputs (Q), bit memory (M) area, and the current values for the timers (T). The CPU 210 clears these elements only during the first scan.

If a memory cartridge is not installed, the CPU 210 restores the current values for the four counters.

2. The CPU 210 filters the inputs and updates the value of the analog adjustment potentiometer (stored in SMW2). This delays the scan by approximately 15 ms.
3. The CPU 210 executes the user program. As the program writes values to the outputs, the CPU 210 immediately updates the outputs.
4. The CPU 210 updates the time base for the 100 ms timers.

The interrupt routine is not executed as part of the normal scan cycle, but is executed when the interrupt event occurs (which may be at any point in the scan cycle). After the Enable Interrupt (ENI) instruction in the main program segment has been executed, the CPU 210 executes the interrupt routine on the rising edge of $\text{I}0.0$. The CPU 210 can run the interrupt routine any time within the scan cycle.

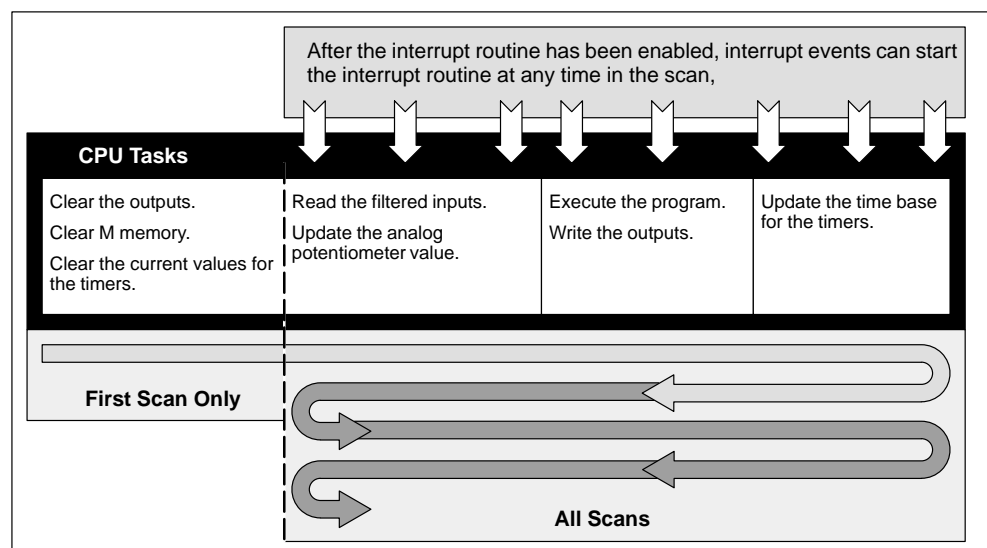


Figure 4-4 Scan Cycle for the CPU 210

Understanding the Basic Scan Cycle of the PDS 210

The scan cycle for the PDS 210 is similar to the scan cycle for the CPU 210. Because the PDS 210 communicates with STEP 7-Micro/WIN, it must process any communication requests. The PDS 210 also updates the timers before executing the program, which is different from the CPU 210.

As shown in Figure 4-5, the PDS 210 performs the following tasks during the scan cycle:

1. During the first scan (after power is turned on) only, the PDS 210 clears the outputs (Q), bit memory (M) area, and the current values for the timers (T). The PDS 210 clears these elements only during the first scan.
2. The PDS 210 filters the inputs and updates the value of the analog adjustment potentiometer (stored in SMW2). This delays the scan by approximately 15 ms.
3. The PDS 210 updates the time base for the 100 ms timers. (Notice that the PDS 210 updates the timers before executing the program.)
4. The PDS 210 executes the user program. As the program writes values to the outputs, the PDS 210 immediately updates the outputs.
5. The PDS 210 processes any communication requests from STEP 7-Micro/WIN.

The interrupt routine is not executed as part of the normal scan cycle, but is executed when the interrupt event occurs (which may be at any point in the scan cycle). After the Enable Interrupt (ENI) instruction in the main program segment has been executed, the PDS 210 executes the interrupt routine on the rising edge of $\text{I}0.0$. The PDS 210 can run the interrupt routine any time within the scan cycle.

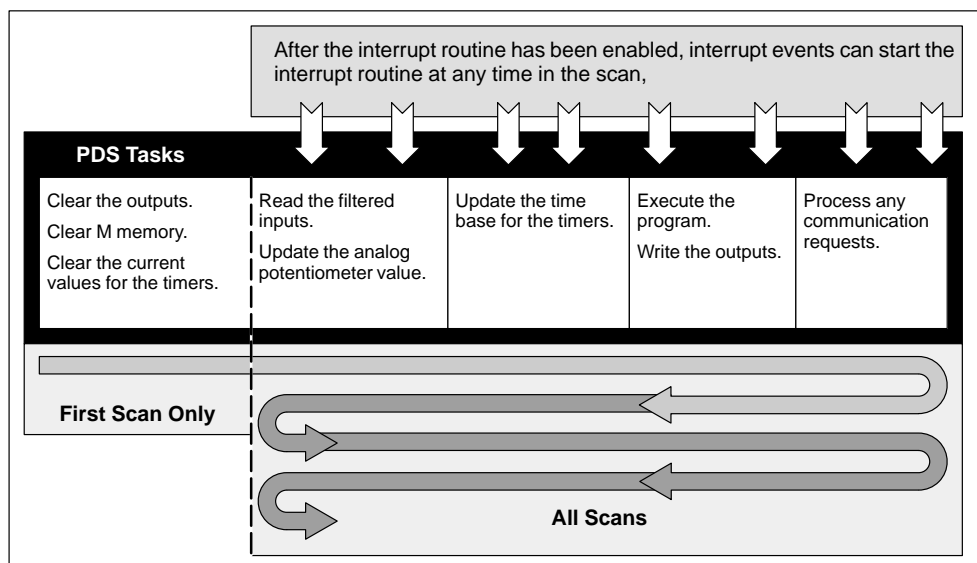


Figure 4-5 Scan Cycle for the PDS 210

Using the Debug Option to Specify the Number of Scans

STEP 7-Micro/WIN allows you to debug your program by specifying a number of scans to be run on the PDS 210 before stopping. (See Section 2.9.) You can specify to run one scan or several. The PDS 210 executes the first scan as if power were just turned on. At the end of the specified number of scans, all of the outputs are cleared.

You can only use the debug option with the PDS 210.

As shown in Figure 4-6, the PDS 210 performs the following tasks when debugging a program:

1. During the first scan only, the PDS 210 clears the outputs (Q), bit memory (M) area, and the current values for the timers (T).
2. The PDS 210 filters the inputs and updates the value of the analog adjustment potentiometer (stored in SMW2).
3. The PDS 210 updates the time base for the 100 ms timers.
4. The PDS 210 executes the user program. As the program writes values to the outputs, the PDS 210 immediately updates the outputs.
5. The PDS 210 processes any communication requests from STEP 7-Micro/WIN.
6. If you specified more than 1 scan, the PDS 210 starts the next scan, starting with Step 2.
7. After the specified number of scans have been run, the PDS 210 clears all outputs and disables the hardware interrupt.

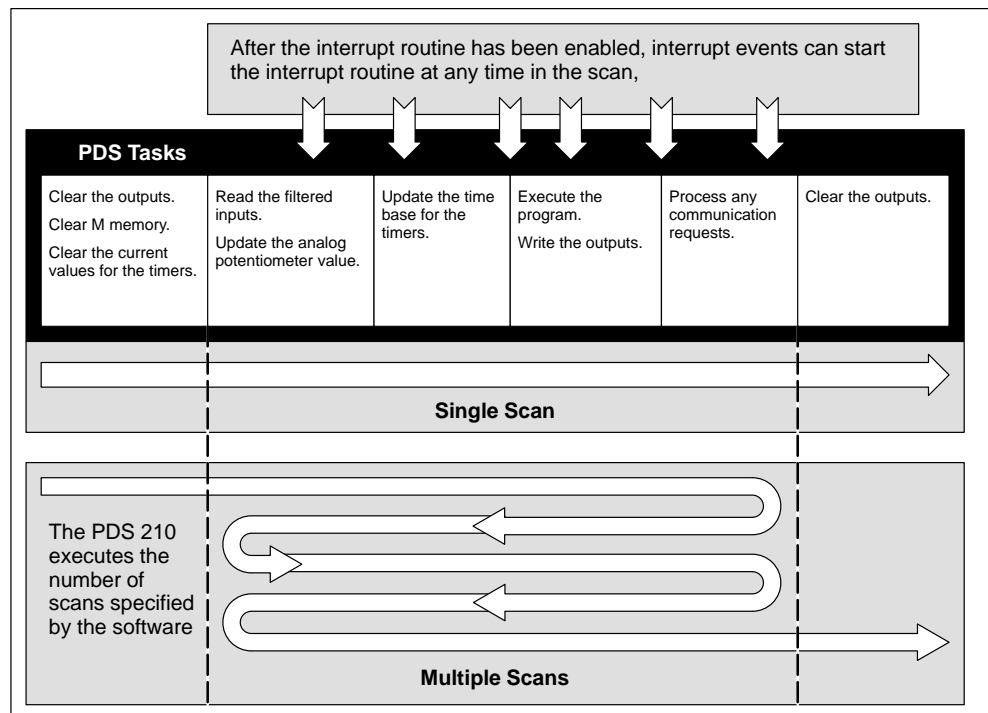


Figure 4-6 Scan Cycle for the Debug Option

4.4 Understanding the Programming Languages

The CPU 210 and STEP 7-Micro/WIN support the following programming languages:

- Statement list (STL) is a set of mnemonic instructions that represent functions of the CPU.
- Ladder is a graphical language that resembles the electrical relay diagrams for the equipment.

STEP 7-Micro/WIN also provides two representations for displaying the addresses and the programming instructions in the program: International and SIMATIC. Both the international and SIMATIC representations refer to the same CPU 210 instruction set. There is a direct correspondence between the International and the SIMATIC representation; both representations have the same functionality.

Understanding the Basic Elements of Ladder

When you write a program in ladder, you create and arrange the graphical components to form a network of logic. As shown in Figure 4-7, the following types of elements are available for creating your program:

- Contacts: each of these elements represents a switch through which power can flow when a switch is closed.
- Coils: each of these elements represents a relay that is energized by power flowing to that relay.
- Boxes: each of these elements represents a function that is executed when power flows to the box.
- Networks: this element forms a complete circuit. Power flows from the left power rail through the closed contacts to energize the coils or boxes.

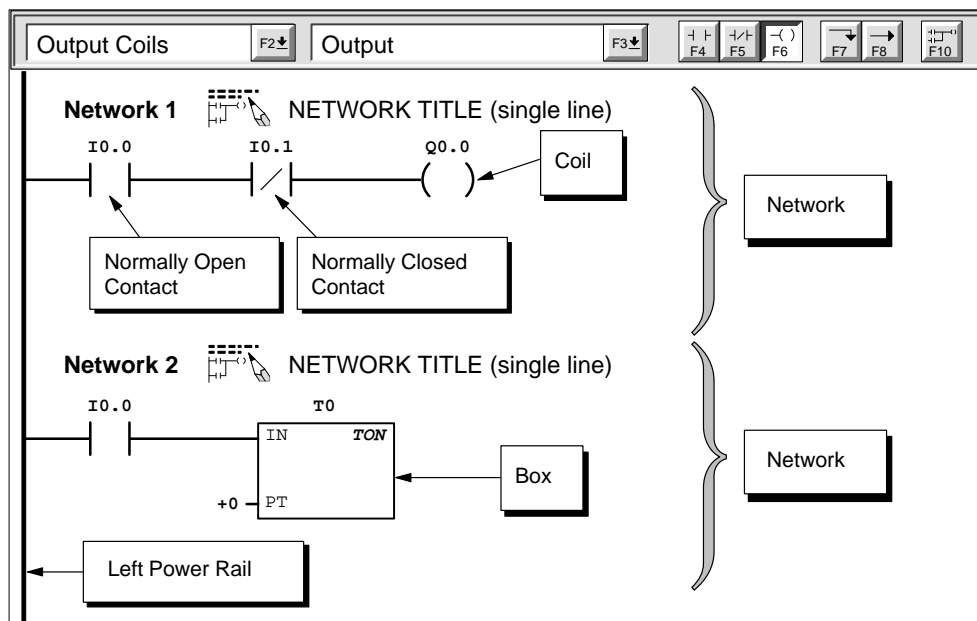


Figure 4-7 Basic Elements of Ladder

Understanding the Statement List Instructions

Statement list (STL) is a programming language in which each statement in your program includes an instruction that uses a mnemonic abbreviation to represent a function of the CPU. You combine these instructions into a program to produce the control logic for your application. Figure 4-8 shows the basic elements of a statement list program.

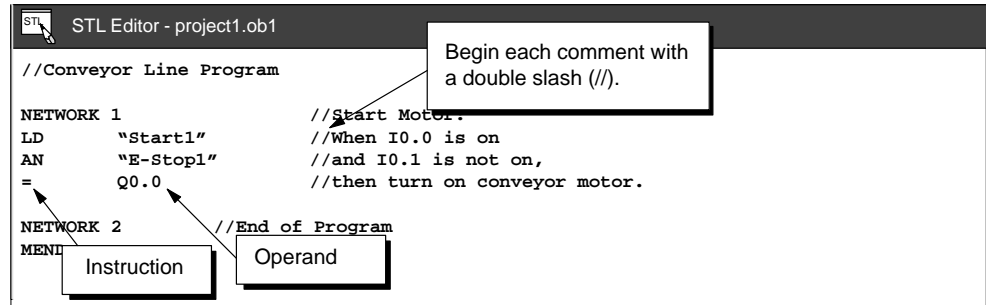


Figure 4-8 STL Editor Window with Sample Program

The STL instructions use a logic stack for solving your control logic. As shown in Figure 4-9, this logic stack is eight bits deep by one bit wide. Most of the STL instructions work either with the first bit or with the first and the second bits of the logic stack. New values can be “pushed” (or added) onto the stack; when the top two bits of the stack are combined, the stack is “popped” (reduced by one bit).

While most STL instructions only read the values in the logic stack, many STL instructions also modify the values stored in the logic stack. Figure 4-9 shows three examples of how three instructions use the logic stack.

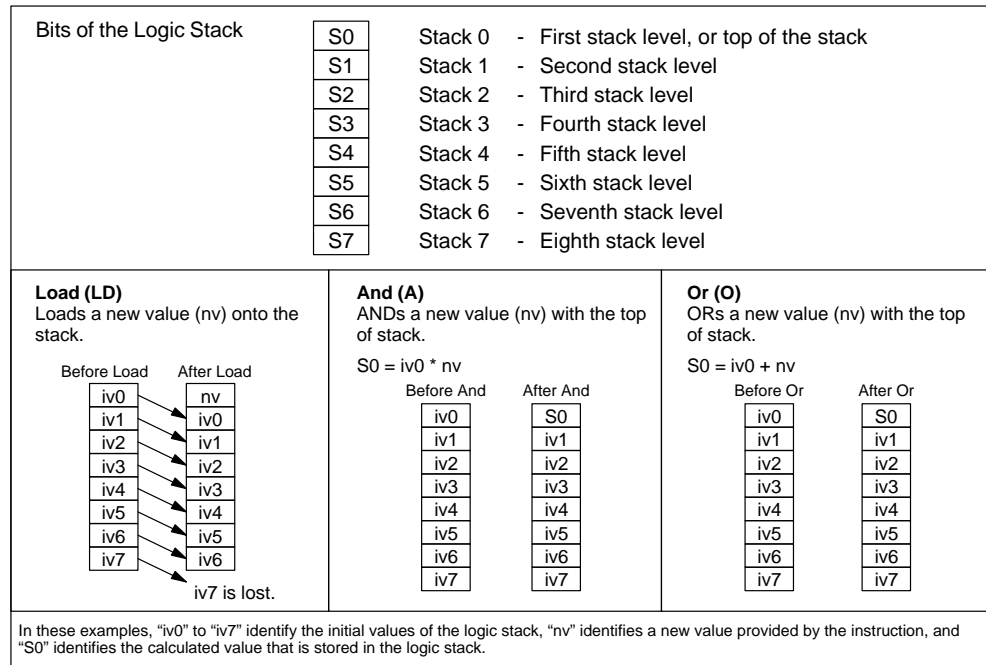


Figure 4-9 Logic Stack of the CPU 210

4.5 Understanding the Addresses of the Memory Areas

The CPU 210 provides 4 digital input points and 4 digital output points. In addition to the I/O, the CPU provides memory areas for storing information. These memory locations have unique addresses that can be accessed by your program. Figure 4-10 shows the memory areas and the range of addresses for the CPU 210.

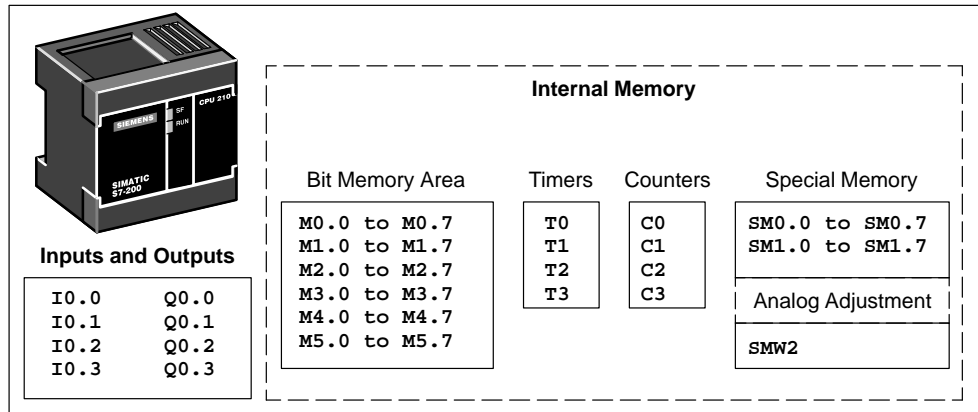


Figure 4-10 Memory Addresses for the CPU 210

Accessing the data in words (16-bit units) allows the following ranges of integer values:

- Unsigned Integer: 0 to 65,535 (decimal)
0 to FFFF (hexadecimal)
- Signed Integer: -32,768 to +32,767 (decimal)
8000 to 7FFF (hexadecimal)

Using the Memory Address to Access Data

To access a bit in a memory area, you specify the address, which includes the memory area identifier, the byte address, and the bit number. Figure 4-11 shows an example of addressing a bit (which is also called "byte.bit" addressing). In this example, the memory area and byte address (M=bit memory area, and 3=byte 3) are followed by a period (".") to separate the bit address (bit 4).

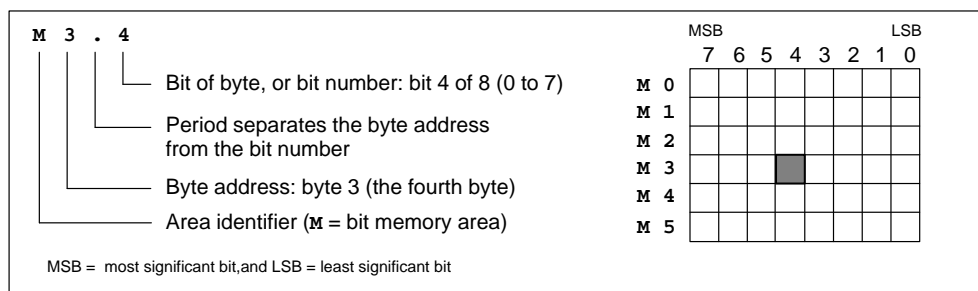


Figure 4-11 Accessing a Bit of Data in the CPU Memory (Byte.Bit Addressing)

You can access data in many CPU memory areas (T, C, M, and SM) as words. To access a word of data in the CPU memory, you must specify the address in a similar way to the address for a bit. This includes an area identifier, data size designation, and the starting byte address of the value, as shown in Figure 4-12. Timer (T) and counter (C) data are accessed by using an address format that includes an area identifier and a device number.

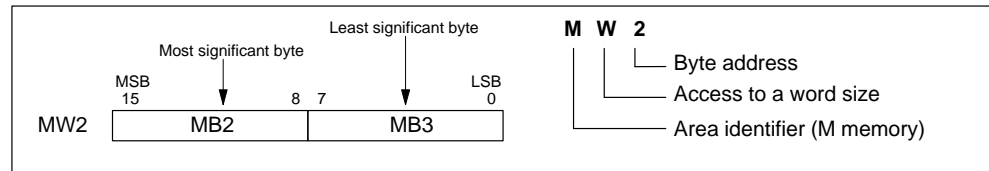


Figure 4-12 Accessing a Word of Data in the CPU Memory

Addressing the Input Image Register (I)

As described in Section 4.2, the CPU samples the physical input points at the beginning of each scan cycle and writes the filtered values to the input image register.

Format: Bit `I0.[bit address]` `I0.1`

Addressing the Outputs (Q)

When the control logic of the program turns an output coil on, the CPU immediately turns that output on.

Format: Bit `Q0.[bit address]` `Q0.0`

Addressing the Bit Memory (M) Area

You can use the internal memory bits (M memory) as control relays to store the intermediate status of an operation or other control information.

Format: Bit `M[byte address].[bit address]` `M2.7`
 Word `M[size][starting byte address]` `MW0`

Addressing the Special Memory (SM) Bits

The SM bits provide a means for communicating information between the CPU and your program. You can use these bits to select and control some of the special functions of the CPU 210, such as a bit that turns on for the first scan, bits that toggle at fixed rates, or a word that stores the value of the analog adjustment potentiometer.

For more information about the SM bits, see Appendix B. While the SM area is based on bits, you can access the data in this area as bits or (for the analog adjustment) a word.

Format: Bit `SM[byte address].[bit address]` `SM0.1`
 Word `SM[size][starting byte address]` `SMW2`

Addressing the Timer (T) Memory Area

Timers are devices that count increments of time. The four timers (T0 to T3) have resolutions (time-base increments) of 100 ms. The current value of each timer is stored as a 16-bit (word) signed integer. You access the current value by using the timer address (T + timer number).

Format: `T[timer number]` T0

Addressing the Counter (C) Memory Area

Counters are devices that count each low-to-high transition event on the counter input(s). The four counters (C0 to C3) provided by the CPU 210 count both up and down. The current value of each counter stores the accumulated count as a 16-bit (word) signed integer.

Format: `C[counter number]` C2

Using Constant Values

Many of the programming instructions for the CPU 210 allow you to use constant values. These constants can only be word-length, signed integers. The CPU stores all constants as binary numbers, which can then be represented in decimal, hexadecimal, or ASCII formats.

Decimal Format: `[decimal value]`
 Hexadecimal Format: `16#[hexadecimal value]`
 ASCII Format: `'[ASCII text]`

The CPU 210 does not support “data typing” or data checking (such as specifying that the constant is stored as an integer or a signed integer) and does not check for a certain data type. For example, an LDW>= instruction can use the value in MW2 as a signed integer value, while a MOVW instruction can use the same value in MW2 as an unsigned binary value.

The following examples show constants for decimal, hexadecimal, and ASCII format:

- Decimal constant: `20047`
- Hexadecimal constant: `16#4E4F`
- ASCII constant: `'AD'` (ASCII text goes between the apostrophes—also known as “single-quote marks.”)

4.6 Sample Program Using an Interrupt Routine

You can use the hardware interrupt of the CPU 210 to control tasks that require high-speed counting functions. For example, you can use the CPU 210 to count a pulse train from an RTD sensor instrument and control a resistance heater. Figure 4-13 shows a sample application for the following tasks:

1. An instrument (such as a PT100 RTD sensor) measures the temperature, and an RTD module generates an output pulse train that is proportional to the temperature.
2. Using the hardware interrupt event (the rising edge of $\text{I}0.0$), the CPU 210 counts the pulses that it receives over a period of time (5 seconds). Based on the count (which relates to the temperature), the CPU 210 turns a digital output on or off.
3. The power contactor turns the resistance heater on or off, based on the state of the output of the CPU 210.

Figure 4-14 shows a sample program for this application example. This example counts up to 3 kHz.

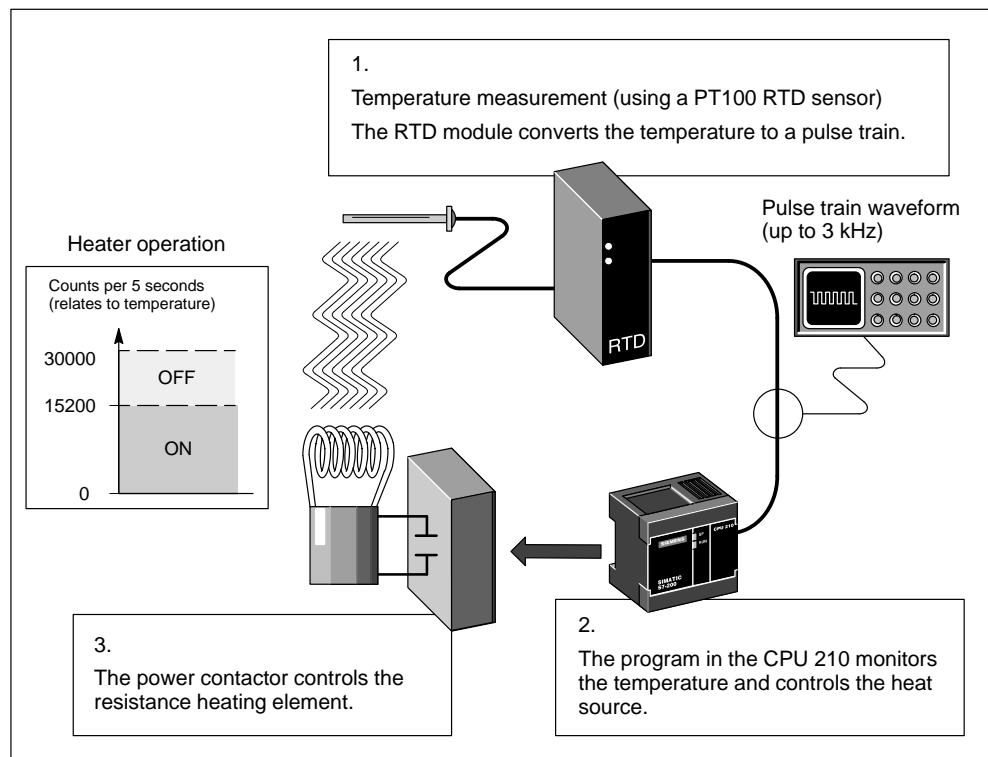


Figure 4-13 Sample Application Using the Hardware Interrupt

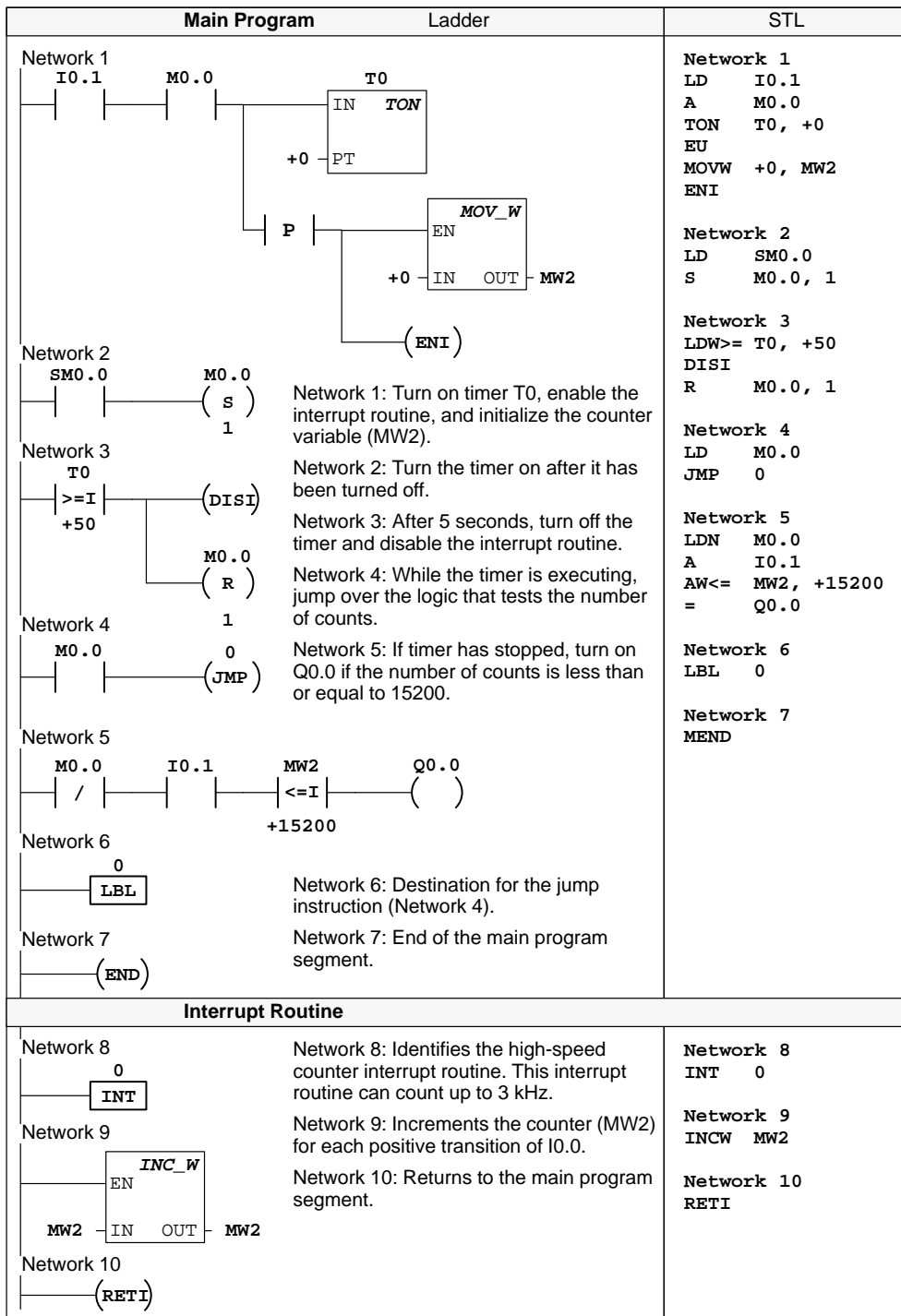


Figure 4-14 Using an Interrupt Routine to Provide a High-Speed Counter

4.7 Using the Analog Adjustment Potentiometer

As shown in Figure 4-15, your CPU 210 provides one analog adjustment potentiometer (located under the access cover of the module). You can adjust this potentiometer to increase or decrease values which are stored in bytes of Special Memory (SMW2). Your program can use this read-only value for a variety of functions, such as updating the current value for a timer or a counter, entering or changing the preset values, or setting limits.

You use a small screwdriver to make the adjustments: turn the potentiometer clockwise (to the right) to increase the value, and counter-clockwise (to the left) to decrease the value.

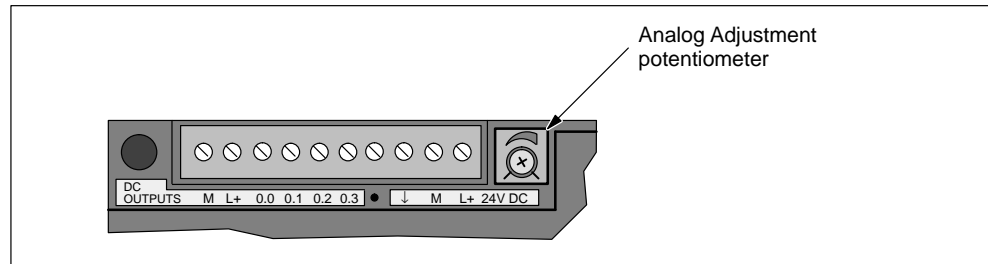


Figure 4-15 Analog Adjustment Potentiometer

SMW2 stores the digital value that represents the position of the analog adjustment potentiometer.

The CPU 210 samples the analog adjustment potentiometer at least three times a second and has a range from 0 to 255. The new value of the analog adjustment potentiometer is written to SMW2 at the beginning of the next scan.

The analog adjustment potentiometer on the PDS 210 has a nominal range of 0 to 255, with a guaranteed range of 10 to 200.

Figure 4-16 shows a sample program that uses the value entered with the analog adjustment potentiometer.

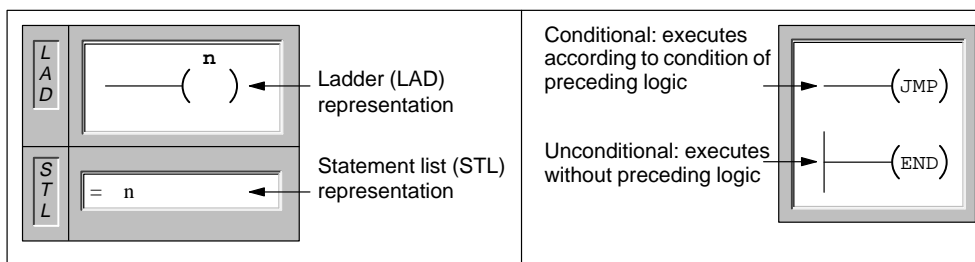
LAD		STL
<p>Network 1</p> <p>Read the analog adjustment value and store that value in MW0.</p>	<p>Network 1</p> <pre>LD I0.0 MOVW SMW2, MW0</pre>	
<p>Network 2</p> <p>Start timer T0.</p>	<p>Network 2</p> <pre>LDN M2.0 TON T0, 0</pre>	
<p>Network 3</p> <p>Turn on M2.0 when T0 reaches the value entered with the analog adjustment.</p>	<p>Network 3</p> <pre>LDW >= T0, MW0 = M2.0</pre>	

Figure 4-16 Sample Program for Using the Analog Adjustment

Instruction Set

5

The following conventions are used in this chapter to illustrate the equivalent ladder and statement list instructions:



Chapter Overview

Section	Description	Page
5.1	Valid Ranges for the CPU 210 and PDS 210	5-2
5.2	Contact Instructions	5-3
5.3	Output Instructions	5-5
5.4	Timer Instructions	5-6
5.5	Counter Instructions	5-8
5.6	Increment and Decrement Instructions	5-9
5.7	Move Instruction	5-10
5.8	Program Control Instructions	5-11
5.9	Logic Stack Instructions	5-13
5.10	Interrupt Instructions	5-14

5.1 Valid Ranges for the CPU 210 and PDS 210

Valid Operand Ranges

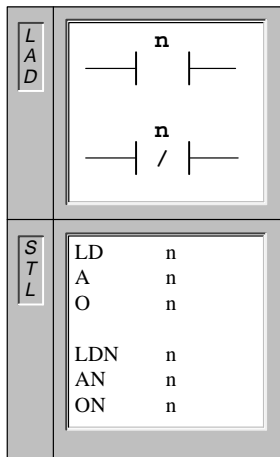
Table 5-1 provides the valid ranges of the operands used to access data in the different memory areas. These ranges vary according to the size of the data being accessed.

Table 5-1 Operand Ranges

Access Method	CPU 210 and PDS 210	
Bit Access (Byte.bit)	I	0.0 to 0.3
	Q	0.0 to 0.3
	M	0.0 to 5.7
	SM	0.0 to 1.7
Word Access	T	0 to 3
	C	0 to 3
	MW	0 to 4
	SMW	0 to 2
	Constant	

5.2 Contact Instructions

Standard Contacts



The **Normally Open** contact is closed (on) when the bit value of address $n = 1$.

In STL, the normally open contact is represented by the **Load**, **And**, and **Or** instructions. These instructions Load, AND, or OR the bit value of address n to the top of the stack.

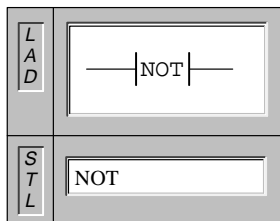
The **Normally Closed** contact is closed (on) when the bit value of address $n = 0$.

In STL, the normally closed contact is represented by the **Load Not**, **And Not**, and **Or Not** instructions. These instructions Load, AND, or OR the logical Not of the bit value of address n to the top of the stack.

Operands: n : I, M, SM

These instructions obtain the referenced value from the image register, which is updated at the beginning of each CPU scan.

Not

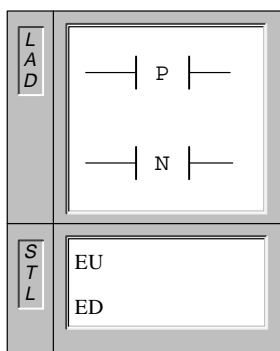


The **Not** contact changes the state of power flow. When power flow reaches the Not contact, it stops. When power flow does not reach the Not contact, it sources power flow.

In STL, the **Not** instruction changes the value on the top of the stack from 0 to 1, or from 1 to 0.

Operands: none

Positive, Negative Transition



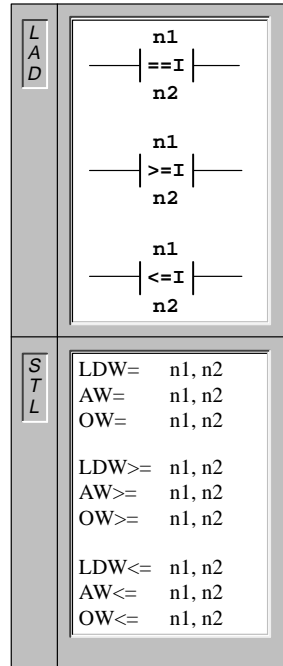
The **Positive Transition** contact allows power to flow for one scan for each off-to-on transition. In STL, the Positive Transition contact is represented by the **Edge Up** instruction. Upon detection of a 0-to-1 transition in the value on the top of the stack, the top of the stack value is set to 1; otherwise, it is set to 0.

The **Negative Transition** contact allows power to flow for one scan, for each on-to-off transition. In STL, the Negative Transition contact is represented by the **Edge Down** instruction. Upon detection of a 1-to-0 transition in the value on the top of the stack, the top of the stack value is set to 1; otherwise, it is set to 0.

Operands: none

You can have a total of 32 transition instructions in a program. These can be any combination of Positive Transition (EU) and Negative Transition (ED) instructions.

Compare Word Integer



The **Compare Word Integer** instruction is used to compare two values: $n1 = n2$, $n1 \geq n2$, or $n1 \leq n2$.

Operands: n1: T, C, MW, SMW
 n2: T, C, MW, SMW, Constant

In ladder, the contact is on when the comparison is true.

In STL, the instructions Load, AND, or OR a 1 with the top of stack when the comparison is true.

Word comparisons are signed ($16\#7FFF > 16\#8000$).

You can create a $<>$, $<$, or $>$ comparison by using the Not instruction with the $=$, \geq , or \leq compare instruction. The following sequence is equivalent to a $<>$ comparison of MW0 to 50:

```
LDW=  MW0, 50
NOT
```

Contact Examples

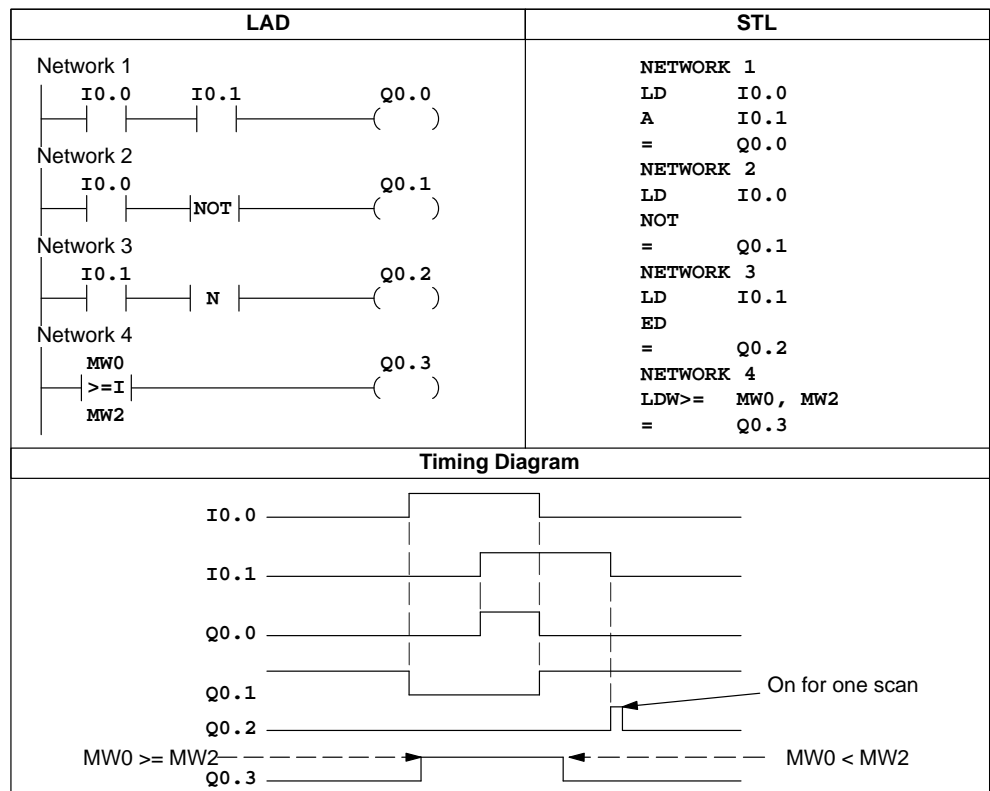
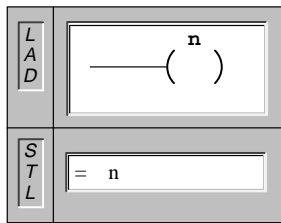


Figure 5-1 Example of Boolean Contact Instructions

5.3 Output Instructions

Output

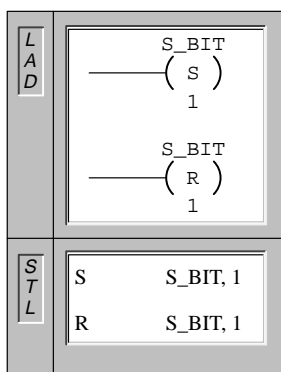


When the **Output** instruction is executed, the specified parameter (n) is turned on.

In STL, the output instruction copies the top of the stack to the specified parameter (n).

Operands: n: Q, M

Set, Reset



When the **Set** and **Reset** instructions are executed, the point specified by S_BIT is set (turned on) or reset (turned off).

Operands: S_BIT: Q, M

Output Example

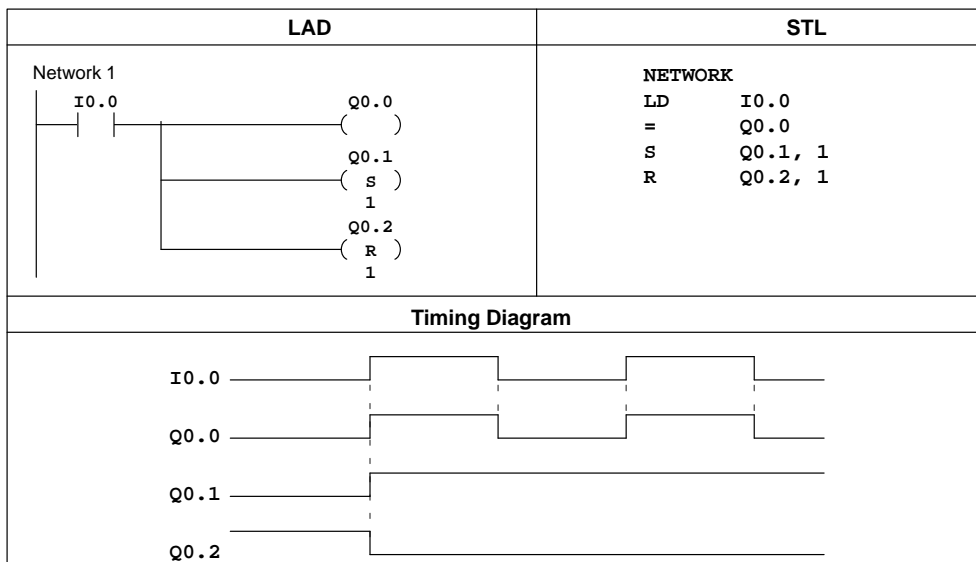
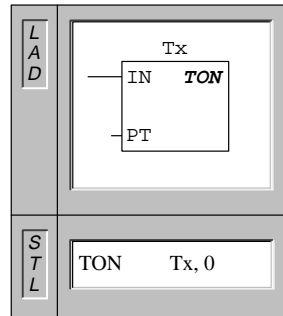


Figure 5-2 Example of Output Instructions

5.4 Timer Instructions

You can use the timer to implement time-based counting functions. The timer times up while the enabling input is on. When the enabling input is off, the timer automatically resets. The timer is best used when you are timing a single interval.

On-Delay Timer



The **On-Delay Timer** instruction times up to the maximum value when enabled. The timer resets when disabled, and stops timing when it reaches the maximum value (3276.7 seconds).

Operands: Tx: T0 to T3
 PT: 0 (preset value is not used)

Timers in the CPU 210 do not use the preset value. The timer counts time for as long as it is enabled. Use a Compare Word instruction to detect the timer value.

Each timer has a 100-ms resolution, with a maximum value of 3276.7 seconds. Once enabled, the timer counts up to the maximum value and stops, unless it is disabled prior to reaching the maximum value. Disabling the timer resets the timer value to zero (0).

Understanding How the CPU 210 Updates the Timers

The timers in the CPU 210 have a 100-ms resolution: each timer counts the number of 100-ms intervals that have elapsed since the timer was last updated. The timer is updated by adding the accumulated number of 100-ms intervals (since the beginning of the previous scan) to the current value (for that timer) when the timer instruction is executed.

The update of the timers is not automatic, since the current value of a timer is updated only if the timer instruction is executed. Consequently, if a timer is enabled but the timer instruction is not executed each scan, the current value for that timer will not be updated, and it will lose time. Likewise, if the same 100-ms timer instruction is executed multiple times in a single scan, the number of 100-ms intervals will be added to the timer's current value multiple times, and it will gain time. For this reason, you should use timers only where the timer instruction will be executed exactly once per scan.

Note

The process of accumulating 100-ms intervals is performed independently of the enabling and disabling of timers, so a given 100-ms timer will be enabled at a point somewhere within the current 100-ms interval. This means that a timed interval for a given 100-ms timer can be up to 100 ms short. Set the parameter for the Compare Word instruction for a value of one greater than the minimum desired timed interval. For example, to guarantee a timed interval of at least 2100 ms, set the value of the Compare Word instruction to 22 (2100 ms equals 21 100-ms units, plus 1 100-ms unit equals 22 100-ms units).

Timer Example

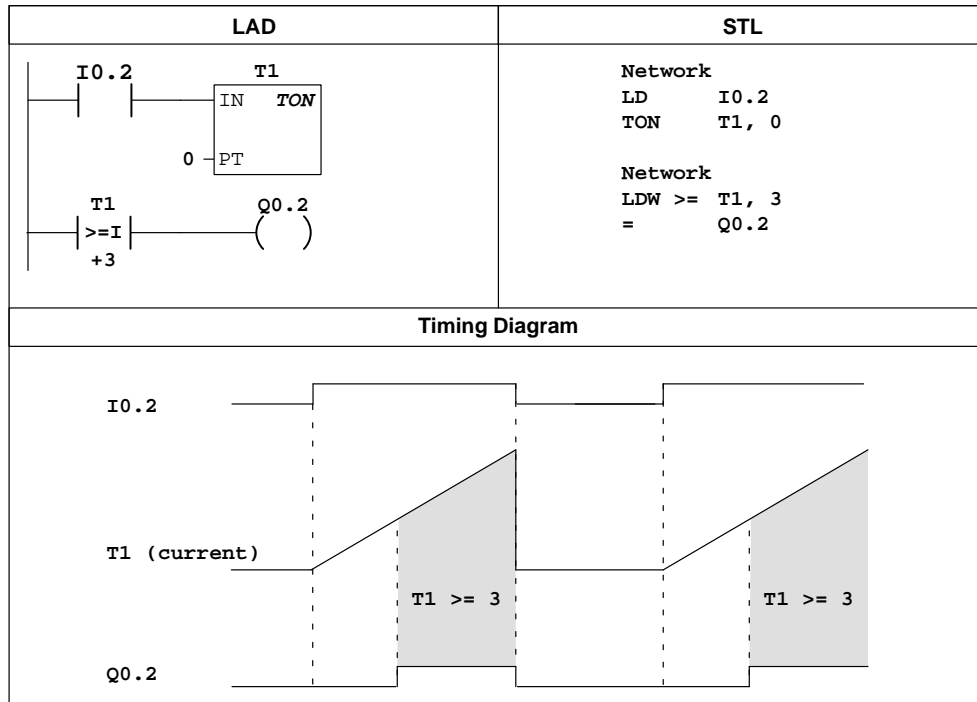


Figure 5-3 Example of the Timer Instruction

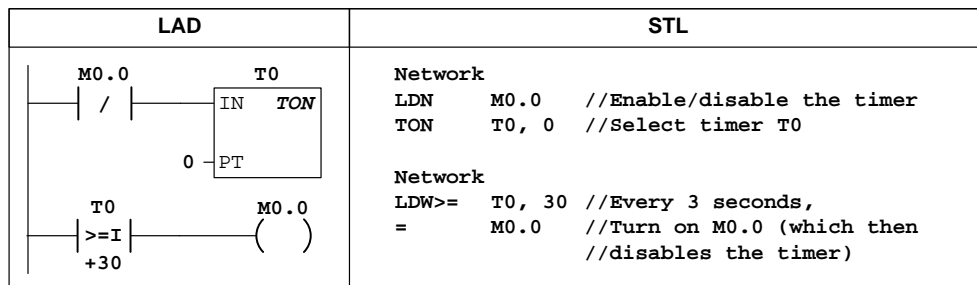


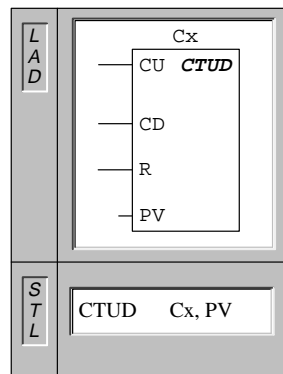
Figure 5-4 Example of an Automatically Retriggered One-Shot Timer

5.5 Counter Instructions

The Up/Down Counter counts up each time the count-up input transitions from off to on, and counts down each time the count-down input transitions from off to on. The counter resets when the reset input turns on. Upon reaching maximum value (32,767), the next rising edge at the count-up input will cause the current count to wrap around to the minimum value (-32,768). Likewise on reaching the minimum value (-32,768), the next rising edge at the count-down input will cause the current count to wrap around to the maximum value (32,767).

The Up/Down counter has a current value that maintains the current count. Use the counter number to reference the current value. Since there is one current value for each counter, do not assign the same number to more than one counter.

Up/Down Counter



The **Up/Down Counter** instruction counts up on rising edges of the Count Up (CU) input. It counts down on the rising edges of the Count Down (CD) input. The counter resets when the Reset (R) input turns on.

In STL, the Reset input is the top of the stack value, the Count Down input is the value loaded in the second stack location, and the Count Up input is the value loaded in the third stack location.

Operands: Cx: C0 to C3
PV: 0 (preset value is not used)

Counter Example

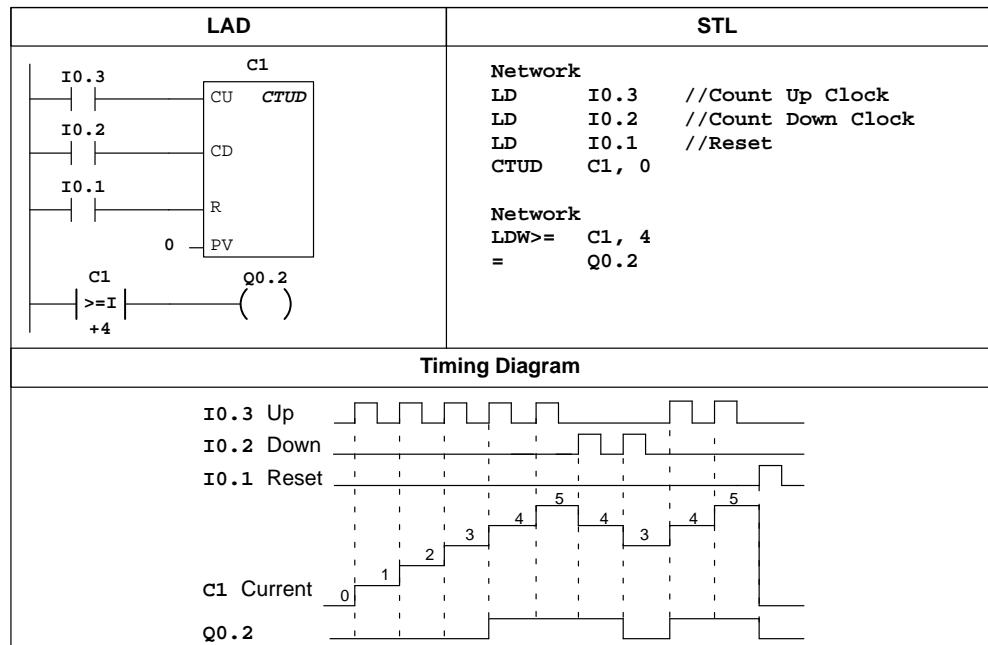
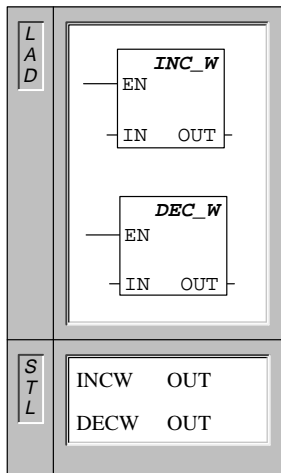


Figure 5-5 Example of the Counter Instruction

5.6 Increment and Decrement Instructions

Increment Word, Decrement Word



The **Increment Word** and **Decrement Word** instructions add or subtract 1 to or from the input word.

Operands: IN: T, C, MW
 OUT: T, C, MW

In ladder: IN + 1 = OUT
 IN - 1 = OUT

In STL: OUT + 1 = OUT
 OUT - 1 = OUT

Increment and decrement word operations are signed (16#7FFF > 16#8000).

When programming in ladder, if you specify the address for IN to be the same address as OUT, you can reduce the amount of memory required.

These instructions affect the following Special Memory bits:
 SM1.0 (zero); SM1.1 (overflow); SM1.2 (negative)

Increment, Decrement Example

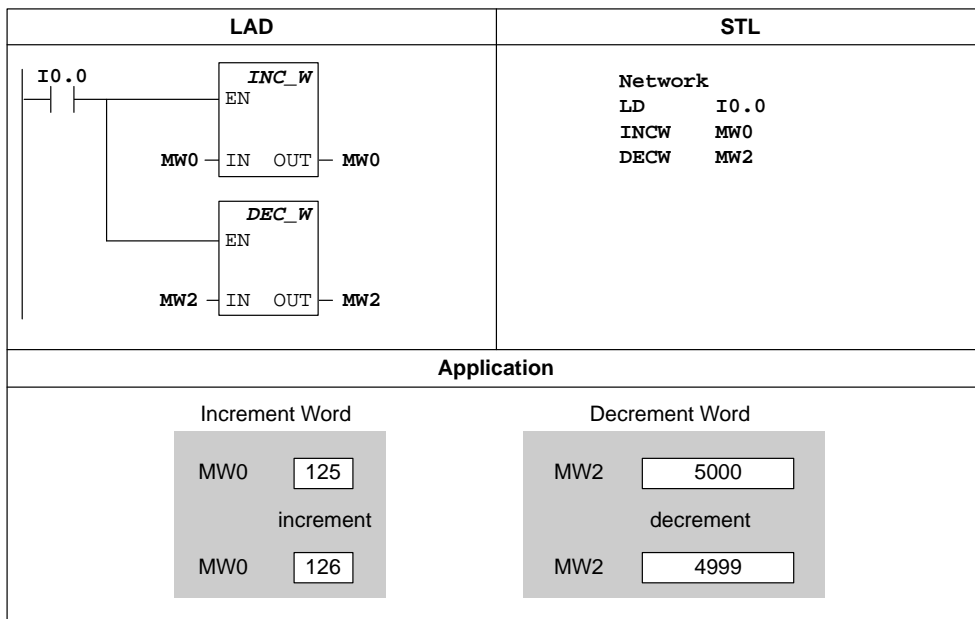
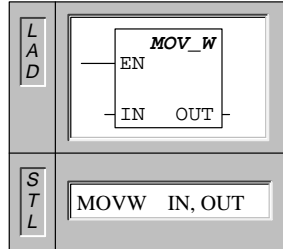


Figure 5-6 Example of Increment/Decrement Instructions

5.7 Move Instruction

Move Word



The **Move Word** instruction moves the input word (IN) to the output word (OUT). The input word is not altered by the move.

Operands: IN: T, C, MW, SMW, Constant
 OUT: T, C, MW

Move Examples

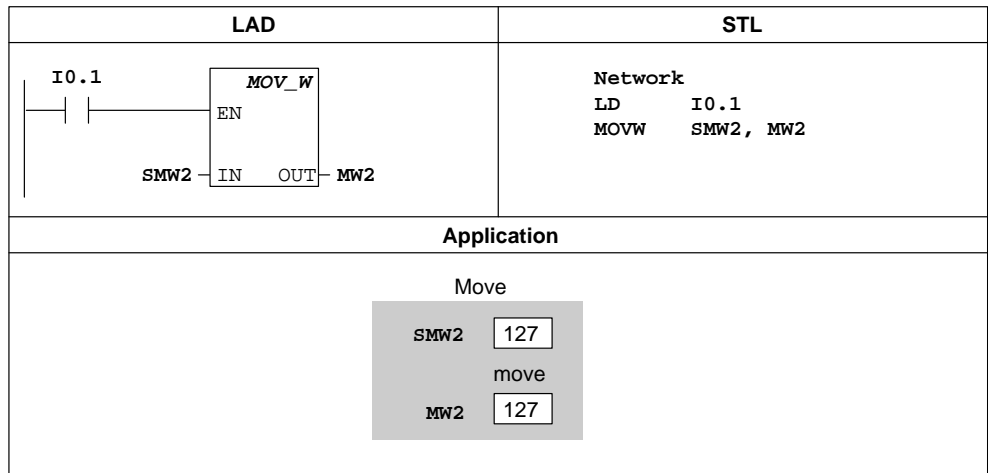
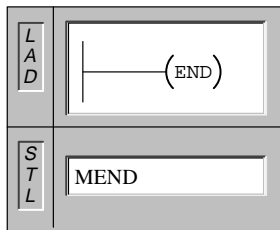


Figure 5-7 Example of Move Instruction

5.8 Program Control Instructions

END

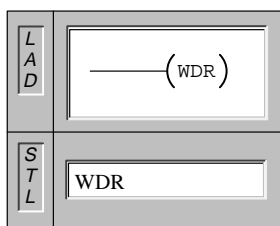


The **Unconditional END** coil is the last instruction in (or, terminates) the main user program. In STL, the unconditional END operation is represented by the **MEND** instruction.

Operands: None

You must terminate the main program with an unconditional END (MEND) instruction.

Watchdog Reset



The **Watchdog Reset** instruction allows the CPU system watchdog timer to be retriggered. This extends the time the scan is allowed to take without getting a watchdog error.

Operands: None

Considerations for Using the WDR Instruction to Reset the Watchdog Timer

You should use the Watchdog Reset instruction carefully. If you use looping instructions to either prevent scan completion, or to excessively delay the completion of the scan, the following processes are inhibited until the scan cycle is terminated:

- Inputs will not update
- Special memory (SM) will not update (SM0, SMW2)
- Timers will not properly accumulate time for scans exceeding 25 seconds

Note

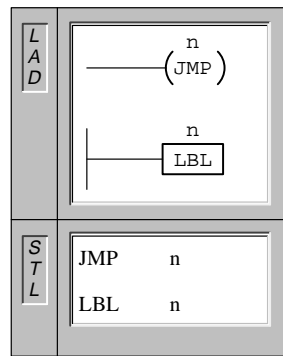
If you expect your scan time to exceed 300 ms, or if you expect a burst of interrupt activity that may prevent completion of the main scan for more than 300 ms, you should use the WDR instruction to re-trigger the watchdog timer.

END and WDR Example

LAD		STL
Network 15		Network LD M0.1 WDR . . Network MEND
	When M0.1 is on, retrigger the Watchdog Reset (WDR) to allow the scan time to be extended.	
Network 78		
	Terminate the main program.	

Figure 5-8 Example of END and WDR Instructions

Jump to Label, Label



The **Jump to Label** instruction performs a branch to the specified label (n) within the program. When a jump is taken, the top of stack value is always a logical 1.

The **Label** instruction marks the location of the jump destination (n).

Operands: n: 0 to 63

Both the Jump and corresponding Label must be in the main program or in the interrupt routine. You cannot jump from the main program to a label in the interrupt routine. Likewise, you cannot jump from the interrupt routine to a label outside the interrupt routine.

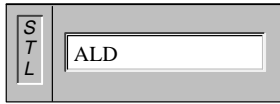
Jump to Label Example

LAD		STL
Network 14		Network LDN SM0.1 JMP 4 . . Network LBL 4
	If this is not the first scan, jump to LBL 4.	
Network 33		
	You can use the JMP to LBL instruction in the main program or in the interrupt routine. The JMP and its corresponding LBL must always be located within the same segment of code (either the main program, or the interrupt routine).	

Figure 5-9 Example of Jump to Label and Label Instructions

5.9 Logic Stack Instructions

And Load



The **And Load** instruction combines the values in the first and second levels of the stack using a logical And operation. The result is loaded in the top of stack. After the ALD executes, the stack depth is one less.

Operands: none

Or Load



The **Or Load** instruction combines the values in the first and second levels of the stack, using a logical Or operation. The result is loaded in the top of stack. After the OLD executes, the stack depth is one less.

Operands: none

Figure 5-10 illustrates the operation of the And Load and Or Load instructions.

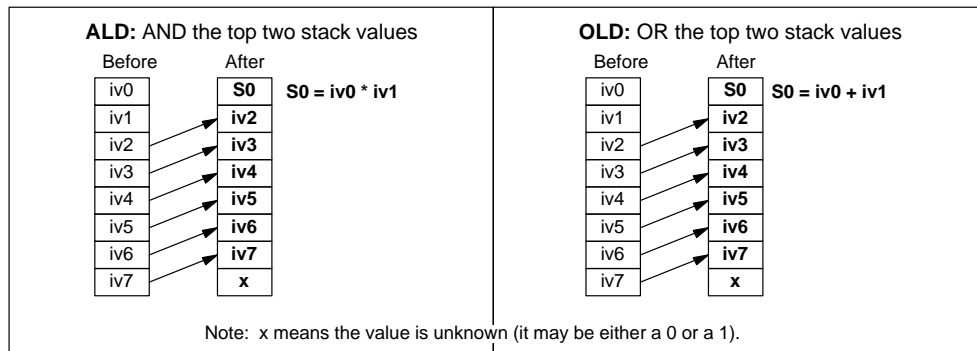


Figure 5-10 And Load and Or Load Instructions

Logic Stack Example

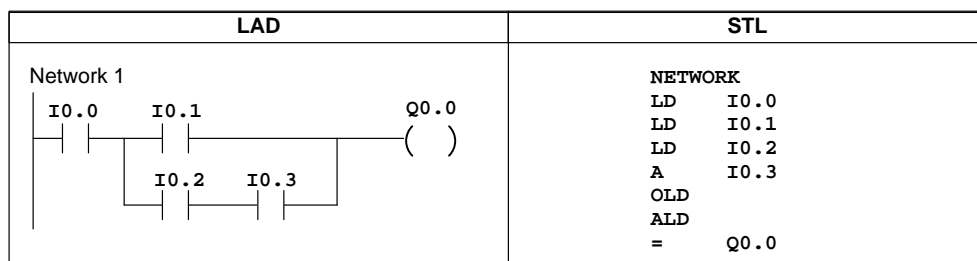
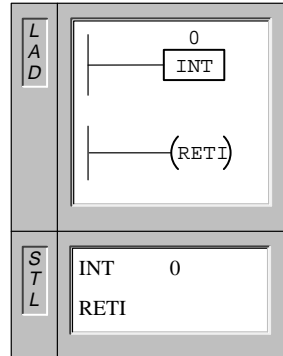


Figure 5-11 Example of Logic Stack Instructions

5.10 Interrupt Instructions

The CPU 210 has one interrupt event (rising edge of $I0.0$). To access this event, you must program an interrupt routine (INT 0), and enable the interrupt. The ENI (enable interrupt) instruction must be executed. An interrupt will occur on the next rising edge of $I0.0$ (after the ENI instruction is executed). It is only necessary to execute the ENI instruction once each time the CPU 210 is powered up unless the DISI (disable interrupt) instruction is executed.

Interrupt Routine, Return from Interrupt Routine



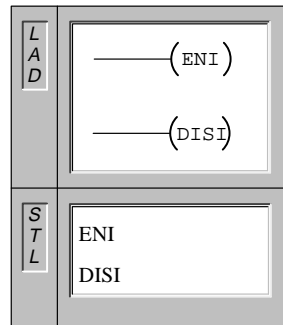
The **Interrupt Routine** instruction marks the beginning of the interrupt routine.

The **Unconditional Return from Interrupt** coil must be used to terminate the interrupt routine.

Operands: n: 0

You can identify the interrupt routine by the interrupt routine label that marks the entry point into the routine. The routine consists of all your instructions between the interrupt label and the unconditional return from interrupt instruction. The interrupt routine executes in response to the rising edge of $I0.0$. You must end the routine (thereby returning control to the main program) by executing the Return from Interrupt instruction (RETI).

Enable Interrupt, Disable Interrupt



The **Enable Interrupt** instruction enables processing of the interrupt events.

The **Disable Interrupt** instruction globally disables processing of the interrupt events. While interrupts are disabled, interrupt events are ignored.

Operands: None

Guidelines and Restrictions for Using the Interrupt Routine

Interrupt processing provides quick reaction to an I/O event. You should optimize the interrupt routine to perform a specific task, and then return control to the main routine. By keeping the interrupt routine short, execution is quick, and other processes are not deferred for long periods of time. If this is not done, unexpected conditions can cause abnormal operation of equipment controlled by the main program.

The following restrictions apply to the use of the interrupt routine:

- You must put the interrupt routine after the end of the main program.
- You cannot use the Disable Interrupt (DISI), Enable Interrupt (ENI), or the End (MEND) instructions in the interrupt routine.
- You must terminate the interrupt routine by a Return from Interrupt instruction (RETI).

Contact and coil logic may be affected by interrupts. To avoid the disruption of the main program segment (caused by branching to and from the interrupt routine), the operating system of the CPU saves and reloads the logic stack and the special memory (SM) bits that indicate the status of instruction operations.

Sharing Data Between the Main Program and the Interrupt Routine

You can share data between the main program and the interrupt routine. For example, a part of your main program may provide data to be used by an interrupt routine, or vice versa. If your program is sharing data, you must also consider the effect of the asynchronous nature of interrupt events, which can occur at any point during the execution of your main program. Problems with the consistency of shared data can result due to the actions of the interrupt routine when the execution of instructions in your main program is interrupted by an interrupt event.

There are a number of programming techniques you can use to ensure that data is correctly shared between your main program and interrupt routine. These techniques either restrict the way access is made to shared memory locations, or make instruction sequences using shared memory locations unable to be interrupted.

- For a ladder program that is sharing a single variable: Use the Move (MOV_W) instruction to access a shared variable. While many ladder instructions are composed of interruptible sequences of STL instructions, the Move instruction in ladder is composed of a single STL instruction whose execution cannot be affected by interrupt events.
- For an STL or a ladder program that is sharing multiple variables: If the shared data is composed of a number of related words, you can use the interrupt disable/enable instructions (DISI and ENI) to control the execution of the interrupt routine. At the point in your main program where operations on shared memory locations are to begin, disable the interrupt. Once all actions affecting the shared locations are complete, re-enable the interrupt. During the time that interrupts are disabled, the interrupt routine cannot execute and therefore cannot access the shared memory locations; however, this approach can cause interrupt events to be missed.
- If the interrupt routine and the main program are sharing a bit in a byte, then the remaining seven bits cannot be used for any purpose. For example: if **M1.0** is being used for coordination between the interrupt routine and the main program, then **M1.1** through **M1.7** cannot be used for any purpose.

Interrupt Example

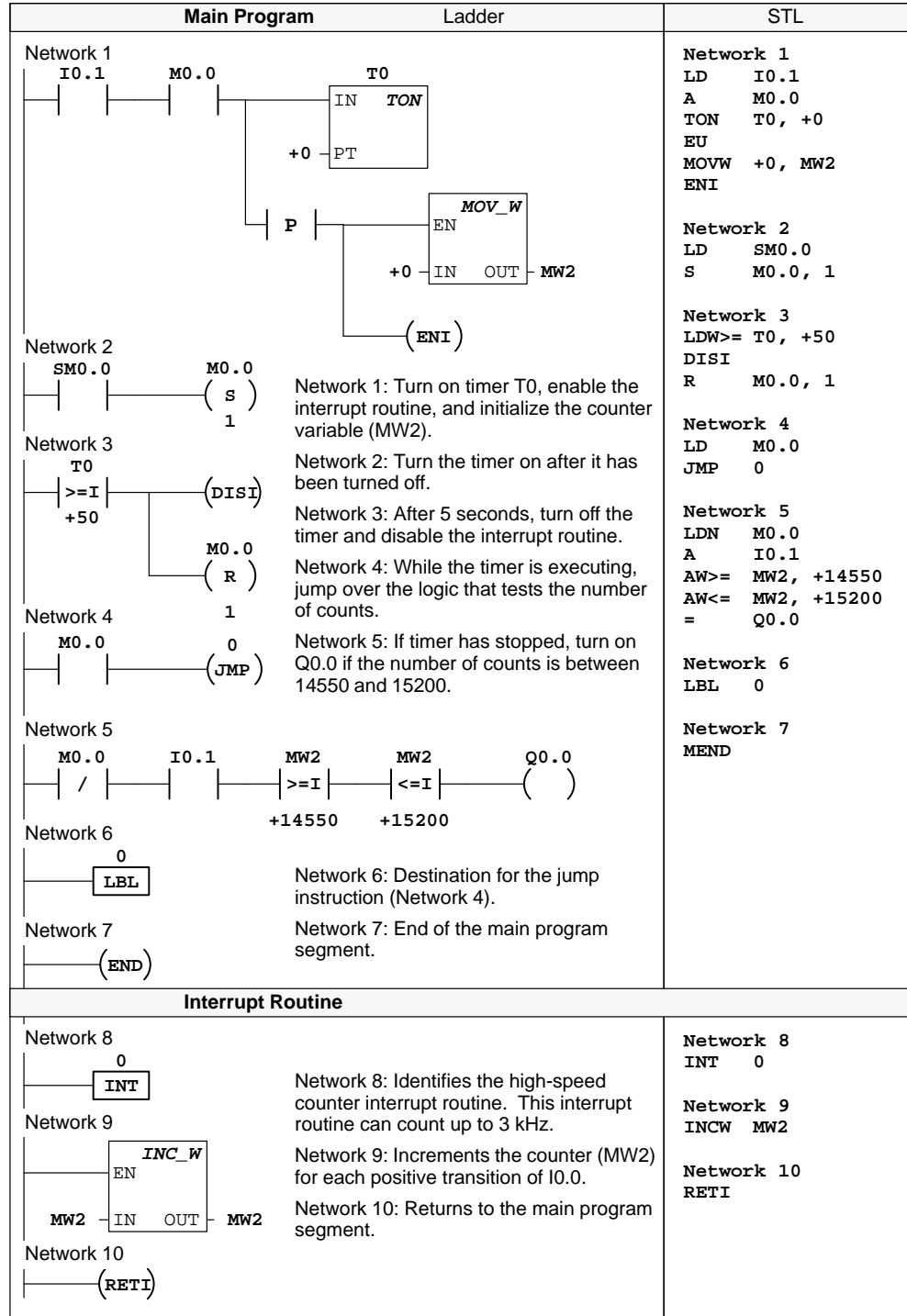


Figure 5-12 Using an Interrupt Routine to Provide a High-Speed Counter

CPU 210 Data Sheets

A

Chapter Overview

Section	Description	Page
A.1	General Technical Specifications	A-2
A.2	CPU 210 DC Power Supply, 24 VDC Inputs, 24 VDC Outputs	A-4
A.3	CPU 210 AC Power Supply, 24 VDC Inputs, Relay Outputs	A-6
A.4	CPU 210 AC Power Supply, AC Inputs, Relay Outputs	A-8
A.5	PDS 210 AC Power Supply, DC Inputs, Relay Outputs	A-10
A.6	Memory Cartridge 8K x 8	A-12
A.7	Memory Cartridge 16K x 8	A-13
A.8	PC/PPI Cable	A-14
A.9	DC Input Simulator	A-15

efesotomasyon.com

A.1 General Technical Specifications

National and International Standards

The national and international standards listed below were used to determine appropriate performance specifications and testing for the S7-200 family of products. Table A-1 defines the specific adherence to these standards.

- Underwriters Laboratories, Inc.: UL 508 Listed (Industrial Control Equipment)
- Canadian Standards Association: CSA C22.2 Number 142 Certified (Process Control Equipment)
- Factory Mutual Research: FM Class I, Division 2, Groups A, B, C, & D Hazardous Locations, T4A
- VDE 0160: Electronic equipment for use in electrical power installations
- European Community (CE) Low Voltage Directive 73/23/EEC
EN 61131-2: Programmable controllers - Equipment requirements
- European Community (CE) EMC Directive 89/336/EEC

Electromagnetic emission standards:
EN 50081-1: residential, commercial, and light industry
EN 50081-2: industrial environment

Electromagnetic immunity standards:
EN 50082-2: industrial environment

Technical Specifications

The S7-200 CPUs and all S7-200 expansion modules conform to the technical specifications listed in Table A-1.

Table A-1 Technical Specifications for the S7-200 Family

Environmental Conditions — Transport and Storage	
IEC 68-2-2, Test Bb, Dry heat & IEC 68-2-1, Test Ab, Cold	-40° C to +70° C
IEC 68-2-30, Test Db, Damp heat	25° C to 55° C, 95% humidity
IEC 68-2-31, Toppling	100 mm, 4 drops, unpacked
IEC 68-2-32, Free fall	1 m, 5 times, packed for shipment
Environmental Conditions — Operating	
Functional range ¹	0° C to 55° C, 95% maximum non-condensing humidity
IEC 68-2-14, Test Nb	5° C to 55° C, 3° C/minute
IEC 68-2-27 Mechanical shock	15 G, 11 ms pulse, 6 shocks in each of 3 axis
IEC 68-2-6 Sinusoidal vibration	0.35 mm peak-to-peak 10 to 57 Hz; 2 G panel mount, 1G din rail mount, 57 to 150 Hz; 10 sweeps each axis, 1 octave/minute
EN 60529, IP20 Mechanical protection	Protects against finger contact with high voltage as tested by standard probes. External protection is required for dust, dirt, water, and foreign objects of less than 12.5 mm in diameter.

Table A-1 Technical Specifications for the S7-200 Family, continued

Electromagnetic Compatibility — Immunity²	
IEC 801-2 Electrostatic Discharge	8 kV air discharge to all surfaces and communication port
IEC 801-3 Radiated electromagnetic field	26 MHz to 1 GHz 10 V/m, 80% modulation with 1 kHz signal 900 MHz \pm 5 MHz, 10 V/m, 50% duty cycle, 200 Hz repetition frequency
IEC 801-4 Fast transient bursts	2 kV, 5 kHz with coupling network to AC and DC system power 2 kV, 5 kHz with coupling clamp to digital I/O and communications
IEC 801-5 Surge immunity	2 kV asymmetrical, 1 kV symmetrical 5 positive / 5 negative pulses 0°, +90°, -90° phase angle (24 VDC circuits require external surge protection)
VDE 0160 Non-periodic overvoltage	at 85 VAC line, 90° phase angle, apply 390 V peak, 1.3 ms pulse at 180 VAC line, 90° phase angle, apply 750 V peak, 1.3 ms pulse
Electromagnetic Compatibility — Conducted and Radiated Emissions³	
EN 55011, Class A, Group 1, conducted ² 0.15 to 0.5 MHz 0.5 to 5 MHz 5 to 30 MHz	< 79 dB (μ V) Quasi-peak; < 66 dB (μ V) Average < 73 dB (μ V) Quasi-peak; < 60 dB (μ V) Average < 73 dB (μ V) Quasi-peak; < 60 dB (μ V) Average
EN 55011, Class A, Group 1, radiated ² 30 MHz to 230 kHz 230 MHz to 1 GHz	30 dB (μ V/m) Quasi-peak; measured at 30 meters 37 dB (μ V/m) Quasi-peak; measured at 30 meters
EN 55011, Class B, Group 1, conducted ⁴ 0.15 to 0.5 MHz 0.5 to 5 MHz 5 to 30 MHz	< 66 dB (μ V) Quasi-peak decreasing with log frequency to 56 dB (μ V) < 56 dB (μ V) Average decreasing with log frequency to 46 dB (μ V) < 56 dB (μ V) Quasi-peak; < 46 dB (μ V) Average < 60 dB (μ V) Quasi-peak; < 50 dB (μ V) Average
EN 55011, Class B, Group 1, radiated ⁴ 30 MHz to 230 kHz 230 MHz to 1 GHz	30 dB (μ V/m) Quasi-peak; measured at 10 meters 37 dB (μ V/m) Quasi-peak; measured at 10 meters
High Potential Isolation Test	
24 V / 5 V nominal circuits 115/230 V circuits to ground 115/230 V circuits to 115/230 V circuits 230 V circuits to 24 V / 5 V circuits 115 V circuits to 24 V / 5 V circuits	500 VAC (optical isolation boundaries) 1500 VAC 1500 VAC 1500 VAC 1500 VAC

- 1 Operating temperatures are based on the immediate surrounding air at the device.
- 2 Unit must be mounted on a grounded metallic frame with the S7-200 ground connection made directly to the mounting metal. Cables are routed along metallic supports.
- 3 Applicable for all devices bearing the CE (European Community) mark.
- 4 Unit must be mounted in a grounded metal enclosure. AC input power line must be equipped with a Schaffner FN 680-2.5/06 filter or equivalent, 25. cm maximum wire length from filters to the S7-200. The 24 VDC supply and sensor supply wiring must be shielded.

A.2 CPU 210 DC Power Supply, 24 VDC Inputs, 24 VDC Outputs

Model Number: 6ES7 210-0AA00-0XB0

General Features		Input Points	
Physical Size (L x W x D)	90 x 80 x 62 mm (3.54 x 3.15 x 2.44 in.)	Input Type (IEC 1131-2)	Sink/Source IEC 1131 Type 1 in Sink mode
Weight	0.18 kg (0.4 lbs.)	ON State Range	15-30 VDC, 4 mA minimum 35 VDC, 500 ms surge
Power Dissipation	10 W maximum, with maximum output loading	ON State Nominal	24 VDC, 7 mA
User Program Size	256 words (EEPROM)	OFF State Maximum	5 VDC, 1 mA
I/O	4 Inputs / 4 Outputs	Response Time	
Boolean Execution Speed	95 µs per instruction	I0.0 to I0.3	15 ms maximum
Internal Memory Bits	48 bits	Interrupt I0.0	20 µs on, 40 µs off
Timers	4	Optical Isolation	500 VAC, 1 minute
Counters	4 (retentive)	Power Supply	
Analog Adjustments	1	Voltage Range	20.4 to 28.8 VDC
Standards Compliance	UL 508 CSA C22.2 142 FM Class I, Division 2 VDE 0160 compliant CE compliant	Input Current	25 mA at 24 V typical 150 mA at 24 V maximum load
Output Points		UL / CSA Rating	5 VA
Output Type	Sourcing Transistor	Hold Up Time	20 ms minimum from loss of 24 VDC power
Voltage Range	20.4 to 28.8 VDC	Inrush Current	10 A peak at 28.8 VDC
Maximum Load Current*	<u>0° to 40° C</u> <u>55° C</u> per single point 0.75 A 0.50 A all points total 2.25 A 1.75 A	5 VDC Current	100 mA
Linear derate 40° to 55° C		Isolated	No
Inductive Load Clamping	(per common) single pulse 2A L/R = 10 ms 1A L/R = 100 ms	DC Sensor Supply	
repetitive	1 W energy dissipation (1/2 Li ² x switch rate < 1 W)	Voltage Range	16.4 to 28.8 VDC
Leakage Current	100 µA	Ripple/Noise (<10Mhz)	Same as supplied voltage
Switching Delay	25 µs on, 120 µs off	24 VDC Available Current	100 mA
Surge Current	4 A, 100 ms	Short Circuit Current Limit	< 120 mA
Voltage Drop	1.8 V maximum at maximum current	Isolated	No
Optical Isolation	500 VAC, 1 minute		
Short Circuit Protection	None		

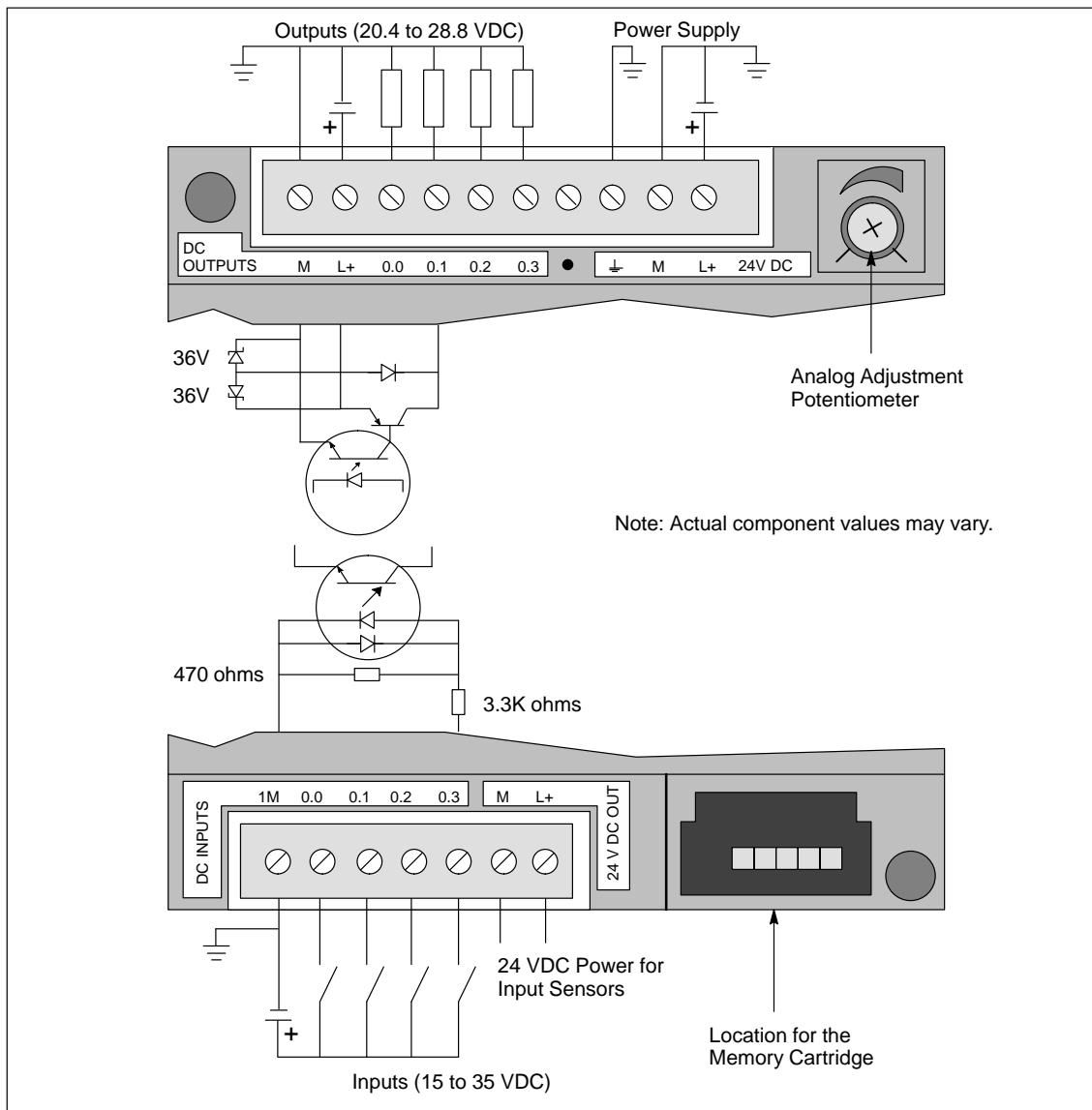


Figure A-1 Connector Terminal Identification for CPU 210 DC/DC/DC

A.3 CPU 210 AC Power Supply, 24 VDC Inputs, Relay Outputs

Model Number: 6ES7 210-0BA00-0XB0

General Features		Input Points	
Physical Size (L x W x D)	90 x 80 x 62 mm (3.54 x 3.15 x 2.44 in.)	Input Type (IEC 1131-2)	Sink/Source IEC 1131 Type 1 in Sink mode
Weight	0.23 kg (0.51 lbs.)	ON State Range	15-35 VDC, 4 mA minimum 35 VDC, 500 ms surge
Power Dissipation	5.0 W maximum, with maximum output loading	ON State Nominal	24 VDC, 7 mA
User Program Size	256 words (EEPROM)	OFF State Maximum	5 VDC, 1 mA
I/O	4 Inputs / 4 Outputs	Response Time	
Boolean Execution Speed	95 μ s per instruction	I0.0 to I0.3	15 ms maximum
Internal Memory Bits	48 bits	Interrupt I0.0	20 μ s on, 40 μ s off
Timers	4	Optical Isolation	500 VAC, 1 minute
Counters	4 (retentive)	Power Supply	
Analog Adjustments	1	Voltage / Frequency Range	85 to 264 VAC at 47 to 63 Hz
Standards Compliance	UL 508 CSA C22.2 142 FM Class I, Division 2 VDE 0160 compliant CE compliant	Input Current	1.75 VA typical, no load 4.75 VA typical, maximum load
Output Points		Hold Up Time	20 ms minimum from loss of AC power
Output Type	Relay, dry contact	Inrush Current	20 A peak at 264 VAC
Voltage Range	5 to 30 VDC / 250 VAC	Fusing (non-replaceable)	2 A, 250 V, Slow Blow
Maximum Load Current		Isolated	Yes. Transformer, 1500 VAC, 1 minute
per point	2 A	DC Sensor Supply	
per common	4 A	Voltage Range	20.4 to 30.0 VDC
Overcurrent Surge	7 A with contacts closed	Ripple/Noise (<10Mhz)	1 V peak-to-peak maximum
Isolation Resistance	100 M Ω minimum (new)	24 VDC Available Current	50 mA
Switching Delay	10 ms maximum	Short Circuit Protection	Yes
Lifetime	10,000,000 Mechanical 100,000 with Rated Load	Isolation	
Contact Resistance	200 m Ω maximum (new)	to logic	No
Isolation		to AC power	Yes
coil to contact	1500 VAC, 1 minute		
contact to contact	1000 VAC, 1 minute		
Short Circuit Protection	None		

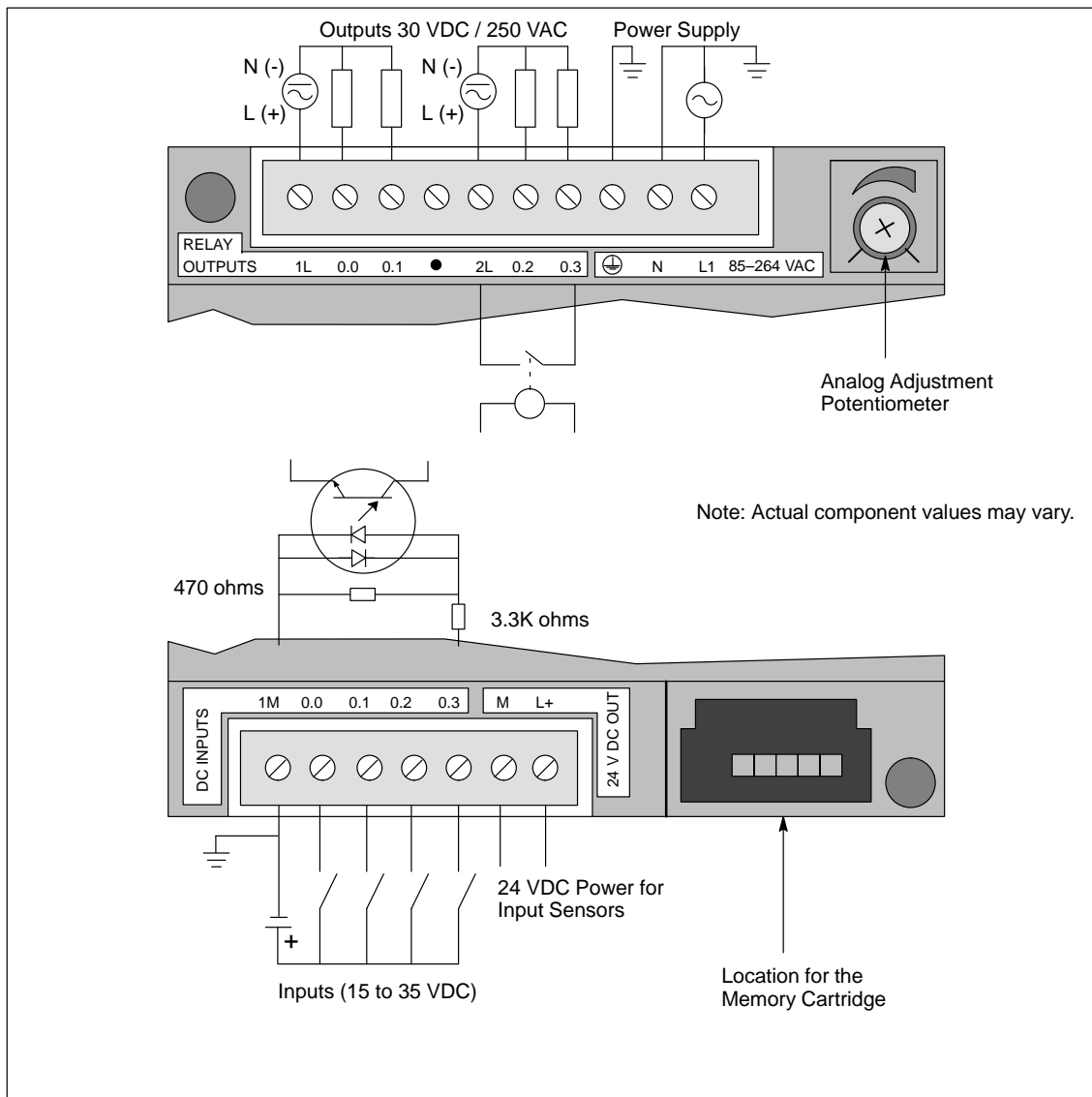


Figure A-2 Connector Terminal Identification for CPU 210 AC/DC/Relay

A.4 CPU 210 AC Power Supply, AC Inputs, Relay Outputs

Model Number: 6ES7 210-0CA00-0XB0

General Features		Input Points	
Physical Size (L x W x D)	90 x 80 x 62 mm (3.54 x 3.15 x 2.44 in.)	Input Type (IEC 1131-2)	Type 1 Sinking
Weight	0.23 kg (0.51 lbs.)	ON State Range	164 to 265 VAC, 47 to 63 Hz, 4 mA minimum
Power Dissipation	5.0 W maximum, with maximum output loading	ON State Nominal	230 VAC, 50 Hz, 7 mA
User Program Size	256 words (EEPROM)	OFF State Maximum	40 VAC, 1 mA
I/O	4 Inputs / 4 Outputs	Response Time	40 ms typical 55 ms maximum
Boolean Execution Speed	95 μ s per instruction	Optical Isolation	1500 VAC, 1 minute
Internal Memory Bits	48 bits	Power Supply	
Timers	4	Voltage / Frequency Range	85 to 264 VAC at 47 to 63 Hz
Counters	4 (retentive)	Input Current	1.74 VA typical, no load 4.75 VA typical, maximum load
Analog Adjustments	1	Hold Up Time	20 ms minimum from loss of AC power
Standards Compliance	UL 508 CSA C22.2 142 FM Class I, Division 2 CE compliant	Inrush Current	10 A peak at 265 VAC
Output Points		Fusing (non-replaceable)	2 A, 250 V, Slow Blow
Output Type	Relay, dry contact	Isolated	Yes. Transformer, 1500 VAC, 1 minute
Voltage Range	5 to 30 VDC / 250 VAC	DC Sensor Supply	
Maximum Load Current per point	2 A	Not applicable	
per common	4 A		
Overcurrent Surge	7 A with contacts closed		
Isolation Resistance	100 M Ω minimum (new)		
Switching Delay	10 ms maximum		
Lifetime	10,000,000 Mechanical 100,000 with Rated Load		
Contact Resistance	200 m Ω maximum (new)		
Isolation			
coil to contact	1500 VAC, 1 minute		
contact to contact	1000 VAC, 1 minute		
Short Circuit Protection	None		

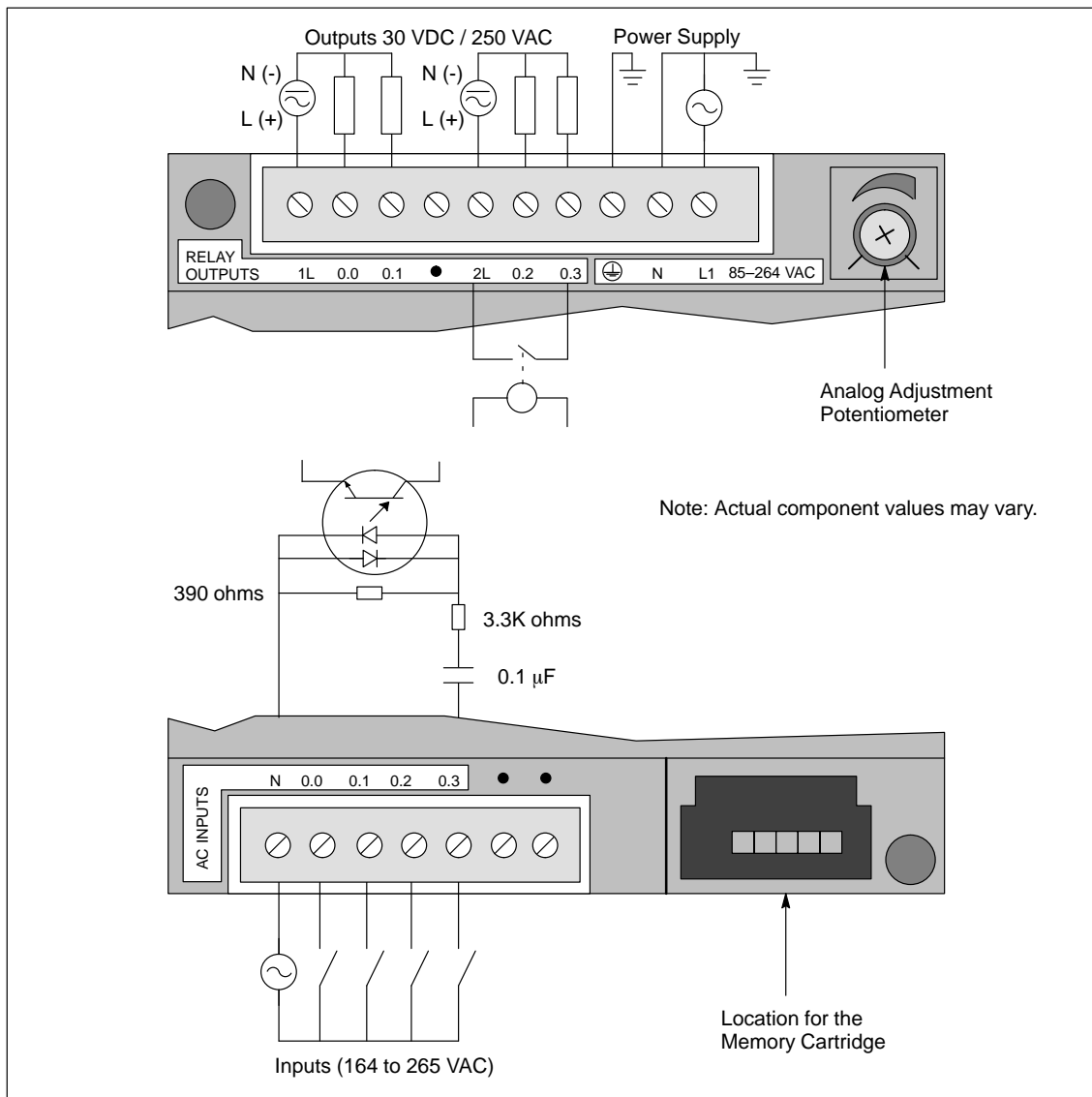


Figure A-3 Connector Terminal Identification for CPU 210 AC/AC/Relay

A.5 PDS 210 AC Power Supply, DC Inputs, Relay Outputs

Model Number: 6ES7 210-8XX00-6AA0

General Features		Input Points	
Physical Size (L x W x D)	197 x 80 x 62 mm (7.76 x 3.15 x 2.44 in.)	Input Type (IEC 1131-2)	Type 1 Sinking
Weight	0.5 kg (1.1 lbs.)	ON State Range	15-30 VDC, 4 mA minimum 35 VDC, 500 ms surge
Power Dissipation	9 W maximum, with maximum output loading	ON State Nominal	24 VDC, 7 mA
User Program Size / Storage	256 words (RAM)	OFF State Maximum	5 VDC, 1 mA
I/O	4 Inputs / 4 Outputs	Response Time	
Boolean Execution Speed	95 μ s	I0.0 to I0.3	15 ms maximum
Internal Memory Bits	48 bits	Interrupt I0.0	210 μ s on, 70 μ s off
Timers	4	Optical Isolation	1500 VAC, 1 minute
Counters	4 (retentive)	Power Supply	
Analog Adjustments	1	Voltage / Frequency Range	85 to 264 VAC at 47 to 63 Hz
Standards Compliance	UL 508 CSA C22.2 142 FM Class I, Division 2 VDE 0160 compliant CE compliant	Input Current	4.5 VA typical, CPU only 50 VA max. load
Output Points		Hold Up Time	20 ms min. from 110VAC
Output Type	Relay, dry contact	Inrush Current	20 A peak at 264 VAC
Voltage Range	5 to 30 VDC / 250 VAC	Fusing (non-replaceable)	2 A, 250 V, Slow Blow
Maximum Load Current		Isolated	Yes. Transformer, 1500 VAC, 1 minute
per point	2 A	DC Sensor Supply	
per common	4 A	Voltage Range	20.4 to 28.8 VDC
Overcurrent Surge	7 A with contacts closed	Ripple/Noise (<10Mhz)	1 V peak-to-peak maximum
Isolation Resistance	100 M Ω minimum (new)	24 VDC Available Current	280 mA
Switching Delay	10 ms maximum	Short Circuit Current Limit	< 600 mA
Lifetime	10,000,000 Mechanical 100,000 with Rated Load	Isolated	No
Contact Resistance	200 m Ω maximum (new)		
Isolation			
coil to contact	1500 VAC, 1 minute		
contact to contact	1000 VAC, 1 minute		
Short Circuit Protection	None		

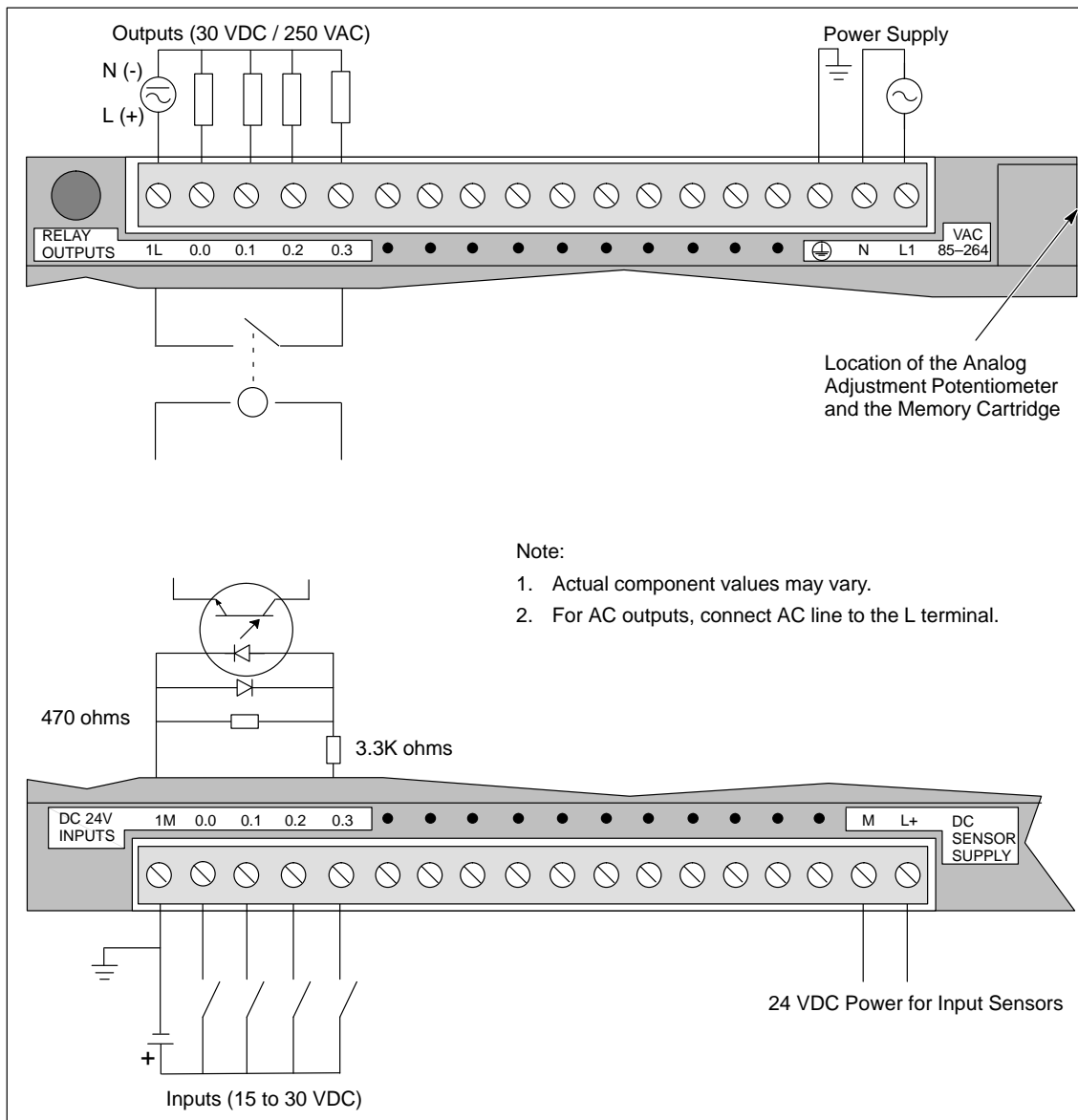


Figure A-4 Connector Terminal Identification for PDS 210 AC/DC/Relay

A.6 Memory Cartridge 8K x 8

Model Number: 6ES7 291-8GC00-0XA0

General Features	
Physical Size (L x W x D)	28 x 10 x 16 mm (1.1 x 0.4 x 0.6 in.)
Weight	3.6 g (0.01 lbs.)
Power Dissipation	0.5 mW
Memory Type	EEPROM
User Storage	8,192 bytes
Standards Compliance	UL 508 CSA C22.2 142 FM Class I, Division 2 VDE 0160 compliant CE compliant

Memory Cartridge Dimensions

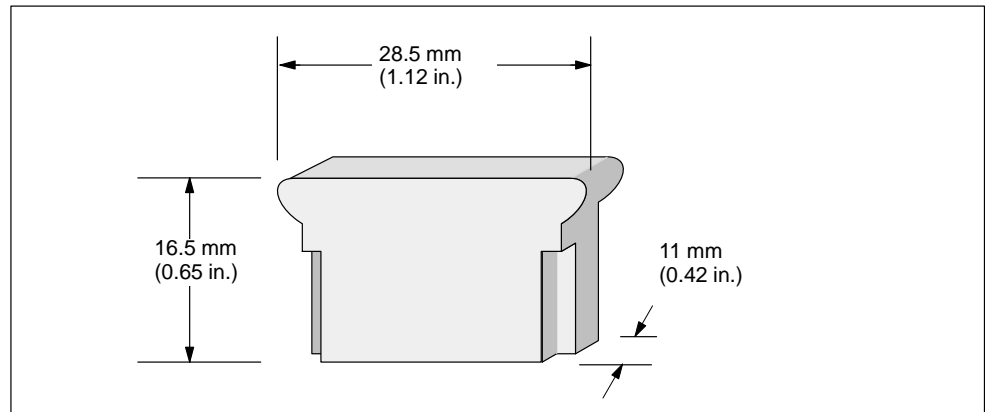


Figure A-5 Memory Cartridge Dimensions - 8K x 8

A.7 Memory Cartridge 16K x 8

Model Number: 6ES7 291-8GD00-0XA0

General Features	
Physical Size (L x W x D)	28 x 10 x 16 mm (1.1 x 0.4 x 0.6 in.)
Weight	3.6 g (0.01 lbs.)
Power Dissipation	0.5 mW
Memory Type	EEPROM
User Storage	16,384 Bytes
Standards Compliance	UL 508 CSA C22.2 142 FM Class I, Division 2 VDE 0160 compliant CE compliant

Memory Cartridge Dimensions

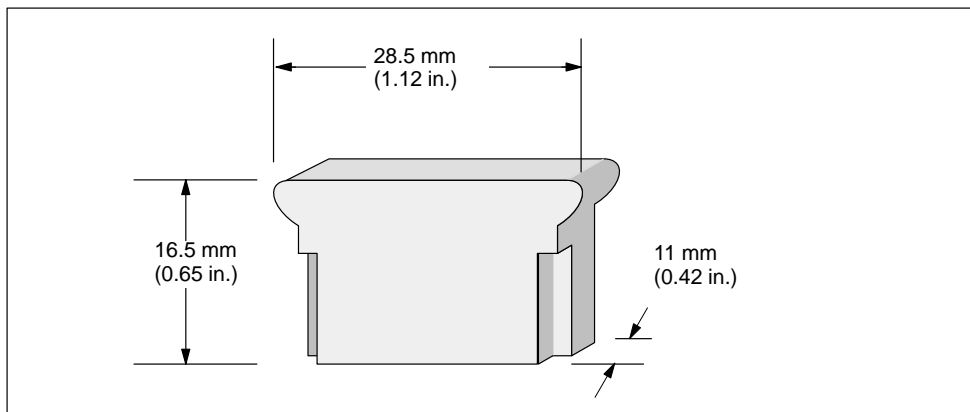


Figure A-6 Memory Cartridge Dimensions - 16K x 8

A.8 PC/PPI Cable

Model Number: 6ES7 901-3BF00-0XA0

General Features	
Cable Length	5 meters (197 in.)
Weight	0.3 kg (0.7 lbs.)
Power Dissipation	0.5 W
Connector Type	PC PLC
	9 pin Sub D (socket) 9 pin Sub D (pins)
Cable Type	RS-232 to RS-485 non-isolated
Cable Receive/Transmit Turn-around Time	2 character times
Baud Rate Supported (selected by dip switch)	<u>Switch</u> 38.4 k 0000 19.2 k 0010 9.6 k 0100 2.4 k 1000 1.2 k 1010 600 1100
Standards Compliance	UL 508 CSA C22.2 142 FM Class I, Division 2 VDE 0160 compliant; CE compliant

Table A-2 PC/PPI Cable Pin-Out

RS-232 Pin	Function Computer End	RS-485 Pin	Function PDS 210 End
2	Received Data (PC listens)	8	Signal A
3	Transmitted Data (PC sends)	3	Signal B
5	Signal Common	7	+24 V
		2	+24 V Return (PLC logic common)
		1	Shield (PLC logic common)

PC/PPI Cable Dimensions

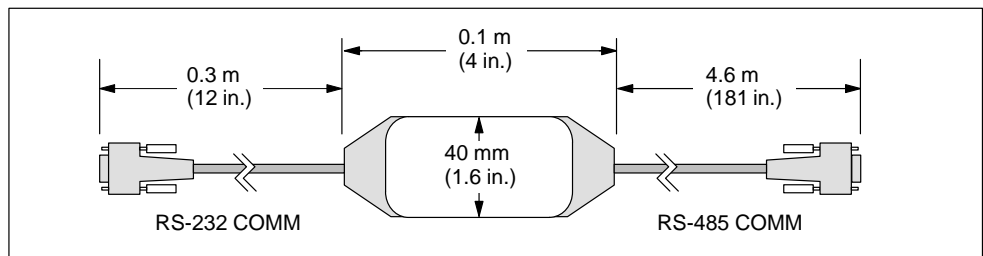


Figure A-7 PC/PPI Cable Dimensions

A.9 DC Input Simulator

Model Number: 6ES7 274-1XH00-0XA0

General Features	
Physical Size (L x W x D)	91 x 36 x 22 mm (3.6 x 1.4 x 0.85 in.)
Weight	0.03 kg (0.06 lb.)
Points	14

Note

The PDS 210 supports only 4 of the 14 simulator points.

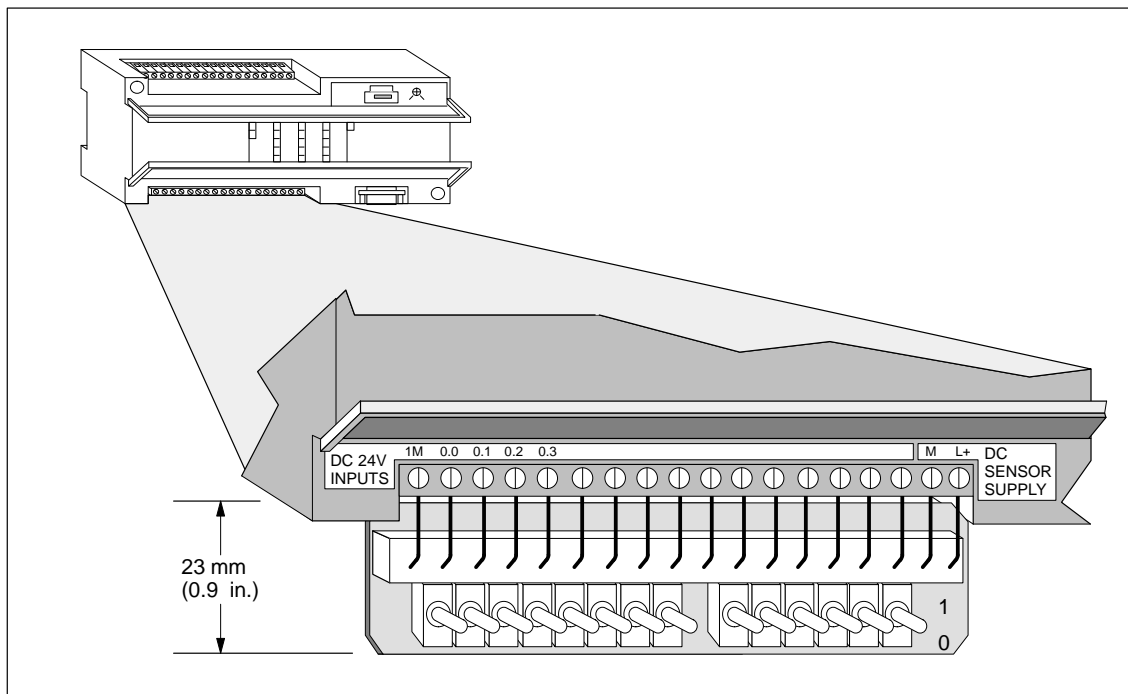


Figure A-8 Installation of the DC Input Simulator

B

Special Memory (SM)

The bits of the SM memory not only provide a variety of status and control functions, but also serve as a means of communicating information between the CPU 210 and your program. You can read the special memory as bits or words. The program development station (PDS 210) provides additional words of special memory for diagnostic capabilities.

SMW0: Status Bits

As described in Table B-1, the bits of SMW0 contain status bits that are updated by the CPU 210 at the end of each scan cycle.

Table B-1 Special Memory Bits SM0.0 to SM1.7

SM Bits	Description
SM0.0	This bit is always on.
SM0.1	This bit is on for the first scan.
SM0.2	Reserved
SM0.3	Reserved
SM0.4	This bit provides a clock pulse that is off for 30 seconds and on for 30 seconds, for a cycle time of 1 minute. It provides an easy-to-use delay, or a 1-minute clock pulse.
SM0.5	This bit provides a clock pulse that is off for 0.5 seconds and then on for 0.5 seconds for a cycle time of 1 second. It provides either an easy-to-use delay or a 1-second clock pulse.
SM0.6	This bit is a scan clock which is off for one scan and then on for the next scan. This bit can be used as a scan counter input.
SM0.7	Reserved
SM1.0	This bit is turned on by the execution of the Increment Word or the Decrement Word instructions when the result of the operation is zero.
SM1.1	This bit is turned on by the execution of the Increment Word or the Decrement Word instructions when an overflow results.
SM1.2	This bit is turned on by the execution of the Increment Word or the Decrement Word instructions when the operation performed produces a negative result.
SM1.3	Reserved
SM1.4	Reserved
SM1.5	Reserved
SM1.6	Reserved
SM1.7	Reserved

SMW2: Analog Adjustment

SMW2 stores the digital value that represents the position of the analog adjustment potentiometer. You access this value as a word.

The CPU 210 samples the analog adjustment potentiometer at least three times a second and has a range of integers from 0 to 255. The new value of the analog adjustment potentiometer is written to SMW2 at the beginning of the next scan.

The analog adjustment potentiometer on the PDS 210 has a nominal range of integers from 0 to 255, with a guaranteed range of 10 to 200.

Table B-2 Special Memory Word SMW2

SM Word	Description
SMW2	This word stores the value entered with the analog adjustment potentiometer. This value is updated once per scan.

SMW4 through SMW20: Reserved

SMW4 through SMW20 are reserved for future use.

SMW22 to SMW26: Scan Times (only for the PDS 210)

SMW22 to SMW26 are available only through communications with the PDS 210. As described in Table B-3, SMW22, SMW24, and SMW26 provide scan time information: minimum scan time, maximum scan time, and last scan time in milliseconds.

Table B-3 Special Memory Words SMW22 to SMW26

SM Word	Description
SMW22	This word provides the scan time of the last scan.
SMW24	This word provides the minimum scan time recorded since entering the RUN mode.
SMW26	This word provides the maximum scan time recorded since entering the RUN mode.

C

Error Handling and Error Codes

Non-Fatal Errors (Compile Rule Violations)

When you download a program, the PDS 210 compiles the program. If the PDS 210 detects that the program violates a compile rule (such as an illegal instruction), the PDS 210 aborts the download and generates a non-fatal, compile-rule error code. Table C-1 describes the error codes that are generated by violations of the compile rules.

Table C-1 Compile Rule Violations

Error Code	Non-Fatal Errors (Compile Rule Violations)
0080	Program too big to compile; you must reduce the program size.
0081	Stack underflow; too many And Load (ALD) or Or Load (OLD) instructions in one network.
0082	Illegal instruction; check instruction mnemonics.
0083	Missing MEND or instruction not allowed in main program: add a MEND instruction, or remove incorrect instruction.
0087	Missing Label (LBL or INT); add the appropriate label.
0089	Missing RETI or instruction not allowed in an interrupt routine: add RETI to the end of the interrupt routine or remove incorrect instruction.
008C	Duplicate Label (LBL or INT); rename one of the labels.
008D	Illegal Label (LBL or INT); ensure the number of labels allowed was not exceeded.
0090	Illegal parameter; verify the allowed parameters for the instruction.
0091	Range error (with address information); check the operand ranges.
0092	Error in the count field of an instruction (with count information); verify the maximum count size.

Fatal Error Codes and Messages for the PDS 210

You cannot access the error codes for the CPU 210. The information about error codes is provided to help you identify problems with your PDS 210 program development station.

Fatal errors cause the PDS 210 to stop the execution of your program. Depending on the severity of the error, a fatal error can render the PDS 210 incapable of performing any or all functions. The PDS 210 attempts to perform the following tasks when a fatal error is detected:

- Changes the mode of operation to the STOP mode
- Turns on the System Fault LED
- Turns off the outputs

The PDS 210 remains in this condition until the fatal error is corrected. Table C-2 provides a list with descriptions for the fatal error codes that can be read from the PDS 210.

Table C-2 Fatal Error Codes and Messages Read from the PDS 210

Error Code	Description
0000	No fatal errors present
0001	User program checksum error
0002	Compiled ladder program checksum error
0003	Scan watchdog time-out error
000A	Memory cartridge failed
000B	Memory cartridge checksum error on user program
0010	Internal software error
0013	Memory cartridge is blank or the program is not understood by the PDS 210 (or the CPU 210)

Handling of Fatal Errors for the CPU 210

Any error detected by the CPU 210 is treated as a fatal error. Depending on the severity of the error, a fatal error can render the CPU 210 incapable of performing any or all functions.

When the CPU 210 detects a fatal error, it turns on the fault indicator and clears the outputs. It remains in this condition until the error has been corrected and the power has been cycled. The CPU 210 can detect the following error conditions:

- Error during the power-up diagnostics: these errors can indicate faulty hardware, but more often are caused by turning power on with an invalid memory cartridge installed.

For a memory cartridge that does not contain a program or that contains a program for a different S7-200 CPU (not a CPU 210), remove the memory cartridge (and install a memory cartridge with a valid CPU 210 program) and then cycle the power to the CPU 210.
- Watchdog time-out errors: these indicate that the user program is using Jump instructions without resetting the watchdog timer. Refer to Section 5.8 for information about the Watchdog Reset (WDR) instruction.

Converting STEP 7-Micro/DOS Files to STEP 7-Micro/WIN Files

D

STEP 7-Micro/WIN allows you to import programs created in the STEP 7-Micro/DOS programming software into Micro/WIN projects.

Importing a STEP 7-Micro/DOS Program

To import a STEP 7-Micro/DOS program into a STEP 7-Micro/WIN project, follow these steps:

1. Select the menu command **Project ► Open**.
2. In the List Files of Type box, select Micro/DOS Project (*.vpu) files.
3. Use the directory browser to select the STEP 7-Micro/DOS directory that contains the program you want to import. Double-click to display the contents in the list box on the left, as shown in Figure D-1.
4. Select the program in the list box, or type the program name in the File Name field.
5. Click the “OK” button. The Micro/DOS program and associated files open as an untitled project.

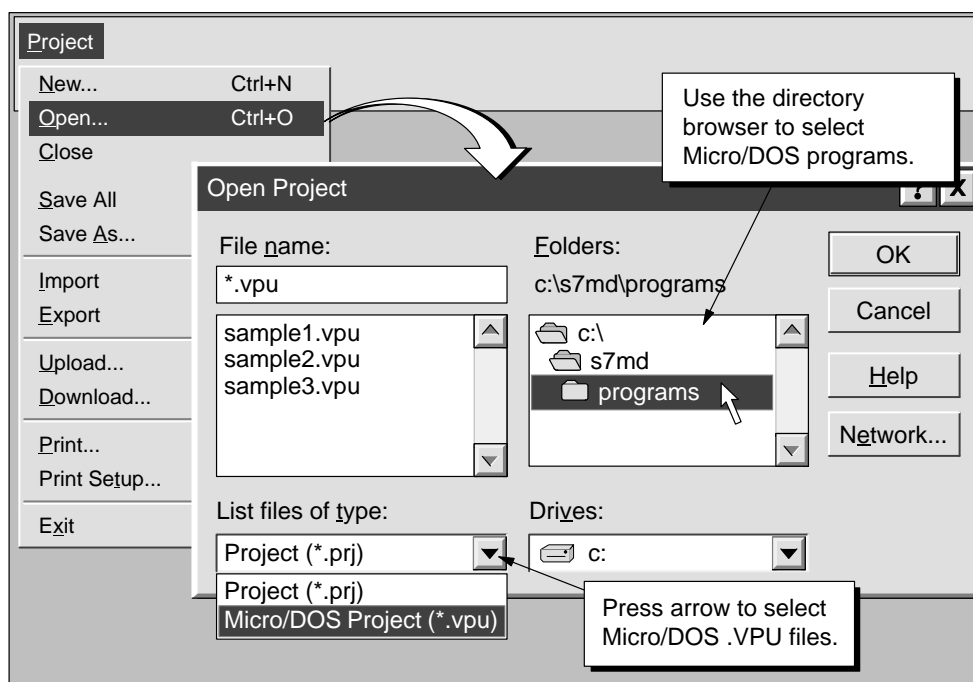


Figure D-1 Converting STEP 7-Micro/DOS Programs to STEP 7-Micro/WIN

Import Guidelines and Limitations

When you import a Micro/DOS .VPU program file, a copy of the following Micro/DOS files are converted to the Micro/WIN format after you save them:

- Program files
- V memory and data (not applicable for the CPU 210 and PDS 210)
- Synonyms and descriptors
- Status chart that has the same name as the project

The following actions occur when you import a Micro/DOS program into a Micro/WIN project:

- Constants that were defined in V memory are maintained. (not applicable for the CPU 210 and PDS 210)
- Micro/DOS synonyms are converted to Micro/WIN symbols, but truncated, if necessary, to fit the 23-character limit. The synonym descriptors, which can be up to 144 characters, are truncated to the 79-character limit for symbol comments in Micro/WIN.
- Micro/DOS network comments (up to 16 lines of 60 characters) are preserved in the STL and LAD editors.
- A Micro/DOS status chart that has the same name as the Micro/DOS program is converted to a Micro/WIN status chart. For example, if you have a program named TEST.VPU that has status charts TEST.CH2 and TEST2.CH2, the status chart named TEST is imported, but not the status chart named TEST2.
- The network address, password, privilege level, output table, and retentive ranges are set based upon the Micro/DOS files. You can find these parameters with the menu command **CPU ► Configure...** (not applicable for the CPU 210 and PDS 210)

Saving the Converted Program

To add the imported program to the same directory as your other current STEP 7-Micro/WIN projects, follow these steps:

1. Select the menu command **Project ► Save As...** and use the directory browser to select your current STEP 7-Micro/WIN directory.
2. In the File Name box, type the name you want to assign to the imported program files, using the .PRJ extension.
3. Click the "OK" button.

Note

Once saved or modified, the program imported into STEP 7-Micro/WIN cannot be exported back into the STEP 7-Micro/DOS format. The original Micro/DOS files, however, are not changed. You can still use the original files within STEP 7-Micro/DOS.

Execution Times for STL Instructions

E

Table E-1 lists the basic execution times of the STL instructions for the CPU 210. The calculation of the basic execution time for an STL instruction shows the time required for executing the logic, or function, of the instruction when power flow is present (where the top-of-stack value is ON or 1).

For some instructions, the execution of that function is conditional upon the presence of power flow: the CPU performs the function only when power flow is present to the instruction (when the top-of-stack value is ON or 1).

The overhead for the CPU 210 is 140 μs per scan. There is also an additional overhead that is associated with the system clock: add 60 μs for every 1 ms counted by the system clock.

Table E-1 Execution Times for the STL Instructions (in μs)

Instruction	Description	On (μs)	Off (μs)
=	Output valid for Q, M	120	120
A	And valid for I, M, SM	110	110
ALD	And Load	60	60
AN	And Not valid for I, M, SM	80	80
AW < =	And Word if less than or equal	300	300
AW=	And Word if equal	300	300
AW > =	And Word if greater than or equal	300	300
CTUD	Up/Down Counter	110	100
DECW	Decrement Word	140	70
DISI	Disable Interrupts	60	60
ED	Edge Down (Negative Transition)	120	120
ENI	Enable Interrupts	60	60
EU	Edge Up (Positive Transition)	110	110
INCW	Increment Word	140	70
INT	Interrupt Routine Add 110 μs if the interrupt routine uses any of the following instructions: <ul style="list-style-type: none"> • MOVW • LDW<=, LDW>=, LDW= • OW<=, ,OW>=, OW= • AW<=, AW>=, AW= 	30	Not applicable
JMP	Jump to Label	70	70
LBL	Label	0	0

Table E-1 Execution Times for the STL Instructions (in μs), continued

Instruction	Description	On (μs)	Off (μs)
LD	Load valid for I, M, SM	70	70
LDN	Load Not valid for I, M, SM	110	110
LDW <=	Load Word if less than or equal	230	230
LDW =	Load Word if equal	230	230
LDW >=	Load Word if greater than or equal	230	230
MEND	Main Program End	50	Not applicable
MOVW	Move Word	210	170
NOT	Not	60	60
O	Or valid for I, M, SM	110	110
OLD	Or Load	60	60
ON	Or Not valid for I, M, SM	110	110
OW < =	Or Word is less than or equal	300	300
OW =	Or Word if equal	300	300
OW > =	Or Word if greater than or equal	300	300
R	Reset	120	70
RETI	Return from Interrupt Add 100 μs if the interrupt routine uses any of the following instructions: <ul style="list-style-type: none"> • MOVW • LDW<=, LDW>=, LDW= • OW<=, OW>=, OW= • AW<=, AW>=, AW= 	70	Not applicable
S	Set	120	70
TON	Non-retentive Timer	110	90
WDR	Watchdog Reset	60	60

CPU 210 Order Numbers

F

CPU and Program Development Station	Order Number
CPU 210 DC Power Supply, 24 VDC Inputs, 24 VDC Outputs	6ES7 210-0AA00-0XB0
CPU 210 AC Power Supply, 24 VDC Inputs, Relay Outputs	6ES7 210-0BA00-0XB0
CPU 210 AC Power Supply, AC Inputs, Relay Outputs	6ES7 210-0CA00-0XB0
PDS 210 AC Power Supply, 24 VDC Inputs, Relay Outputs	6ES7 210-8XX00-6AA0

General	Order Number
Memory Cartridge 8K x 8	6ES7 291-8GC00-0XA0
Memory Cartridge 16K x 8	6ES7 291-8GD00-0XA0
PC/PPI Cable	6ES7 901-3BF00-0XA0
DC Input Simulator	6ES7 274-1XH00-0XA0
10-Position Fan-Out Connector <i>(package of 10 connectors)</i>	6ES7 290-2AA00-0XA0

Programming Software	Order Number
STEP 7-Micro/WIN Individual License	6ES7 810-2AA00-0YX0
STEP 7-Micro/WIN Copy License	6ES7 810-2AA00-0YX1
STEP 7-Micro/WIN Update	6ES7 810-2AA00-0YX3

Manuals	Order Number	
S7-200 Programmable Controller CPU 210 System Manual	German	6ES7 298-8EA00-8AH0
	English	6ES7 298-8EA00-8BH0
	French	6ES7 298-8EA00-8CH0
	Spanish	6ES7 298-8EA00-8DH0
	Italian	6ES7 298-8EA00-8EH0

efesotomasyon.com

Index

A

- Absolute addressing, 4-4
- AC installation, guidelines, 1-10
- Access cover
 - location of the analog adjustment potentiometer, 4-16
 - removing, 1-7
- Accessing data
 - analog adjustment value in special memory (SM), B-2
 - memory areas, 4-4
 - operand ranges, 4-11, 5-2
- Accessing the Symbol Table, 2-13
- Addressing
 - analog adjustment value in special memory (SM), B-2
 - bit memory (M) area, 4-12
 - counter memory area, 4-13
 - input image register, 4-12
 - memory areas, 4-11
 - outputs, 4-12
 - special memory (SM), 4-12
 - timer, 4-13
 - using symbolic names, 2-13, 3-14
- Agency approvals, A-2
- Analog adjustment
 - addressing special memory (SM), 4-12
 - changing the value, 4-16
 - current value stored in SMW2, B-2
 - location of the potentiometer, 4-16
 - nominal range, 4-16
 - sample program, 4-16
- And (A) instruction, 5-3
 - effect on the logic stack, 4-10, 5-3
- And Load (ALD) instruction, 5-13
 - effect on the logic stack, 5-13
- And Not (AN) instruction, 5-3
- Area identifier with device number, accessing timers and counters, 4-12

B

- Baud rate, 2-3
- bit memory. *See* special memory (SM) bits
- Bit memory (M), 4-11
 - addressing, 4-12
- Bit.byte addressing, 4-11
- Boxes, represented in ladder, 4-9

C

- Cables
 - baud rate, 2-3
 - order number, F-1
 - pin assignment, A-14
 - specifications (PC/PPI), A-14
- Case-sensitive symbols, 2-13, 3-15
- CE certification, A-3
- Changing elements in the program
 - ladder, 2-7, 3-16–3-20
 - STL, 2-8, 3-15
- Changing view between ladder and STL
 - creating networks in STL, 2-8
 - menu command, 2-9
- Clearance requirements, 1-4
- Clock, effect on scan time, E-1
- Clock pulse bits, B-1
- Coils, represented in ladder, 4-9, 5-5
- Comments, in statement list, 4-10
- Communications
 - baud rate, 2-3
 - connecting to the PDS 210, 2-3
 - pin assignment, A-14
 - point-to-point (PPI) interface, 2-3
 - setting up parameters, 2-4
- Compare Word instructions, 5-4
- Compiling
 - errors, rule violations, C-1
 - sample program, 3-21
 - STEP 7-Micro/WIN program, 2-8
- Computer requirements, 2-1
- Connector (optional), 1-10
 - order number, F-1

- Connector terminal
 - CPU 210 AC/AC/Relay, A-9
 - CPU 210 AC/DC/Relay, A-7
 - CPU 210 DC/DC/DC, A-5
 - PDS 210, A-11
- Considerations
 - AC installation, 1-10
 - converting files to STEP 7-Micro/WIN, D-2
 - creating the functional specifications, 4-2
 - DC installation, 1-10
 - designing safety circuits, 4-3
 - field wiring, 1-8
 - grounding and isolated circuits, 1-9
 - guidelines for designing a PLC system, 4-2–4-3
 - installation
 - clearance requirements, 1-4
 - using DIN rail stops, 1-6
 - suppression circuits, 1-12
 - using the interrupt routine, 5-15
 - using the Watchdog Reset (WDR) instruction, 5-11
- Constants, 4-13
- Contact instructions, 5-3–5-4
 - Compare Word Integer, 5-4
 - example, 5-4
 - Negative Transition (ED), 5-3
 - NOT, 5-3
 - Positive Transition (EU), 5-3
 - standard contacts, 5-3
- Contacts, represented in ladder, 4-9, 5-3
- Continuous read (Status Chart option), 2-15
 - See also Single read; Status Chart; Write
- Control logic, sample application, 3-4–3-8
- Converting files, STEP7-Micro/DOS to STEP 7-Micro/WIN, D-1
- Converting files from STEP 7-Micro/DOS
 - guidelines and limitations, D-2
 - saving a program, D-2
- Cooling, clearance requirements, 1-4
- Copy, cut, and paste
 - in a Status Chart, 2-15
 - in a Symbol Table, 2-14
- Copying a program from the PDS 210, 2-11
- Counter instructions, 5-8
 - accessing the current value, 4-13
 - addressing, 4-13
 - current value, 4-13, 5-8
 - current values saved in the CPU 210, 2-12
 - device number, 4-12
 - example, 5-8
 - up/down counter, 5-8
- CPU 210
 - agency approvals, A-2
 - analog adjustment
 - location of the potentiometer, 4-16
 - value stored in special memory (SM), B-2
 - basic operation, 4-4
 - compile rule violations, C-1
 - current values for counters saved on power down, 2-12
 - data block not used, 2-10
 - dimensions, 1-5
 - electromagnetic specifications, A-3
 - environmental specifications, A-2
 - equipment requirements, 1-2
 - error handling, C-2
 - execution times, E-1
 - fatal errors, C-1
 - general technical specifications, A-2
 - high potential isolation test, A-3
 - installation
 - in a panel box, 1-7
 - on a DIN rail, 1-6
 - on a panel, 1-6
 - screw size, 1-6
 - interrupt routine, 4-14–4-16, 5-14–5-16
 - guidelines and restrictions, 5-15
 - memory areas, 4-11–4-13
 - memory cartridge location
 - CPU 210 AC/AC/Relay, A-9
 - CPU 210 AC/DC/Relay, A-7
 - CPU 210 DC/DC/DC, A-5
 - order numbers, F-1
 - organizing the program, 4-5
 - product overview, 1-1–1-4
 - program loading, 2-11
 - programming, 1-2
 - restoring program after power down, 2-12
 - scan cycle, 4-6–4-8
 - effect of Watchdog Reset (WDR) instruction, 5-11
 - special memory (SM), B-1
 - summary of features, 1-2
 - transporting a program to, 1-2, 2-11
- CPU 210 AC/AC/Relay
 - order number, F-1
 - specifications, A-8
- CPU 210 AC/DC/Relay
 - order number, F-1
 - specifications, A-6
- CPU 210 DC/DC/DC
 - order number, F-1
 - specifications, A-4

- Creating a program, 2-7–2-9, 3-1–3-25
 - creating a project, 2-6
 - interrupt routine, 4-14
 - high-speed counter example, 4-14–4-16, 5-16
- Creating networks, 2-7, 2-8
- Current value
 - counters, 5-8
 - current value saved at power down, 2-12
 - timers, 5-6
- Cut, copy, and paste
 - in a Status Chart, 2-15
 - in a Symbol Table, 2-14
- D**
- Data checking, not supported, 4-13
- Data sheets
 - CPU 210 AC/AC/Relay, A-8
 - CPU 210 AC/DC/Relay, A-6
 - CPU 210 DC/DC/DC, A-4
 - DC input simulator, A-15
 - memory cartridge, A-12, A-13
 - PC/PPI cable, A-14
 - PDS 210, A-10
- Data typing, not supported, 4-13
- DC input simulator
 - order number, F-1
 - specifications, A-15
- DC installation, guidelines, 1-10
- DC relay, 1-12
- DC transistor, protecting, 1-12
- Debugging the program, 2-16
 - scan cycle, 4-8
- Decrement Word (DECW) instructions, 5-9
 - example, 5-9
- Deleting and inserting rows
 - in a Status Chart, 2-15
 - in a Symbol Table, 2-14
- Designing control logic, example with a sample application, 3-4–3-8
- Device number, timers and counters, 4-12
- Digital inputs
 - addressing, 4-11–4-13
 - reading, 4-6–4-9
- Digital outputs
 - addressing, 4-11–4-13
 - writing to, 4-6–4-9
- Dimensions
 - CPU 210, 1-5
 - DIN rail, 1-5
 - PDS 210, 1-5
- DIN rail
 - dimensions, 1-5
 - installing on a, 1-6
 - using DIN rail stops, 1-6
- Disable Interrupt (DISI) instruction, 5-14–5-17
 - disabling and enabling interrupts, 5-14
- Displaying status in ladder, 2-16
- Downloading a program
 - CPU 210, 2-11
 - PDS 210, 2-10–2-12
 - sample program, 3-23
- Duplicate symbol names, 2-13
- E**
- ED (Negative Transition) instruction, 5-3
- Editing a cell in a Status Chart, 2-15
- Editing a cell in the Symbol Table, 2-14
- Editing a program, 3-15–3-20
- Editing functions, using right mouse button
 - Status Chart, 2-15
 - Symbol Table, 2-14
- Editing tools
 - Status Chart, 2-15
 - Symbol Table, 2-14
- Electromagnetic specifications, A-3
- Elements of an address, 4-11
- Elements of an STL instruction, 4-10
- Enable Interrupt (ENI) instruction, 5-14–5-17
 - enabling and disabling interrupts, 5-14
- Enclosure
 - See also* Panel box
 - clearance requirements, 1-4
 - installing in a panel box, 1-7
- End instruction, 5-11
- Ending the main program segment, 4-5, 5-11
- Entering a program
 - in ladder, 3-15–3-20
 - in STL, 3-15
- Entering a symbolic name in STL, 2-8
- Entering comments, in STL, 2-8, 3-15
- Entering symbols, 2-13–2-15
- Entries
 - duplicate symbol names, 2-13
 - number of characters per symbol, 2-13
 - number of symbols allowed in a Symbol Table, 2-13
- Environmental specifications, A-2
- Equipment requirements, 1-2
 - STEP 7-Micro/WIN, 2-1
- Error codes
 - compile rule violations, C-1
 - error handling for the CPU 210, C-2
 - fatal errors, C-1

Error handling

- fatal errors, 2-17–2-19, C-1
- non-fatal errors, C-1
- non-fatal errors, 2-17–2-19
- responding to errors, 2-17–2-19, C-1
- restarting the CPU after a fatal error, 2-17–2-19

EU (Positive Transition) instruction, 5-3

European Community (CE) certification, A-3

Examples

- contact instructions, 5-4
- counter, 5-8
- End (MEND), 5-12–5-14
- high-speed counter, 4-14–4-16, 5-16
- increment/decrement word, 5-9
- interrupt routine, 4-14–4-16
- Interrupt Routine instructions, 5-16
- jump to label, 5-12–5-14
- ladder elements, 4-9
- logic stack, 5-13–5-15
- move word, 5-10–5-12
- output instructions, 5-5
- Status Chart, 2-15
- Symbol Table, 2-13
- timer, 5-7
- using the analog adjustment, 4-16
- watchdog reset, 5-12–5-14

Execution times, E-1

- affected by power flow, E-1
- statement list instructions, E-1

F

Fan-out connector, order number, F-1

Fatal errors, 2-17–2-19, C-1

Features, 1-2

Field wiring

- installation procedure, 1-8
- optional connector, 1-10
- order number, F-1
- wire sizes, 1-8

First scan bit, B-1

G

Grounding and isolation, wiring guidelines, 1-9

Guidelines

- AC installation, 1-10
- case-sensitive symbols, 2-13
- creating a program with STL, 2-8
- DC installation, 1-10
- designing a PLC system, 4-2–4-3
- entering symbolic addresses, 2-13
- grounding and isolation, 1-9
- number of characters per symbol, 2-13
- number of symbols allowed, 2-13
- overlapping memory addresses in symbol names, 2-13
- suppression circuits, 1-12
- DC relay, 1-12
- wiring, 1-8

Guidelines for converting files, D-2

H

Hardware interrupt, 4-5

- affecting the scan cycle, 4-6–4-9
- effect on the Watchdog Reset (WDR) instruction, 5-11
- enabling and disabling, 5-14
- example, 4-14–4-16, 5-16
- guidelines and restrictions, 5-15
- high-speed counter example, 4-14–4-16, 5-16
- interrupt instructions, 5-14–5-16
- returning from the interrupt routine, 5-14
- sample interrupt routine, 4-14–4-16, 5-16
- sharing data between main program and interrupt, 5-15

Help. *See* Online help

High potential isolation test, A-3

High-speed counter, using the hardware interrupt, 4-14–4-16, 5-16

I

Importing files from STEP 7-Micro/DOS, D-1

- guidelines and limitations, D-2
- saving a program, D-2

Increment Word (INCW) instructions, 5-9

- example, 5-9

Input image register, addressing, 4-12

- Input simulator
 - order number, F-1
 - specifications, A-15
 - Inputs, basic operation, 4-4
 - Inserting an instruction, 3-16–3-20
 - Inserting and deleting rows
 - in a Status Chart, 2-15
 - in a Symbol Table, 2-14
 - Installation
 - agency certifications and specifications, A-3
 - clearance requirements, 1-4
 - considerations for installing a CPU 210
 - clearance requirements, 1-4
 - using DIN rail stops, 1-6
 - dimensions
 - CPU 210, 1-5
 - DIN rail, 1-5
 - PDS 210, 1-5
 - in a panel box, 1-7
 - on a DIN rail, 1-6
 - on a panel, 1-6
 - screw size, 1-6
 - setting the baud rate (PDS 210), 2-3
 - STEP 7-Micro/WIN
 - Windows 3.1, 2-2
 - Windows 95, 2-2
 - Instructions
 - And (A) / And Not (AN), 5-3
 - effect on the logic stack, 4-10, 5-3
 - And Load (ALD), 5-13
 - effect on logic stack, 5-13
 - Compare Word, 5-4
 - Contacts, 5-3–5-4
 - Counter (CTUD), 5-8
 - Decrement Word (DECW), 5-9
 - Disable Interrupt (DISI), 5-14–5-17
 - Enable Interrupt (ENI), 5-14–5-17
 - End, 5-11
 - End (MEND), 4-5, 5-11
 - execution times, E-1
 - Increment Word (INCW), 5-9
 - Interrupt, 5-14–5-16
 - Interrupt Routine (INT), 5-14–5-17
 - Jump to Label, 5-12
 - Load (LD) / Load Not (LDN), 5-3
 - effect on the logic stack, 4-10, 5-3
 - Logic Stack, 5-13
 - Move Word (MOVW), 5-10
 - Negative Transition (ED), 5-3
 - Normally Open / Normally Closed contact, 5-3
 - NOT, 5-3
 - On-Delay Timer, 5-6
 - Or (O) / Or Not (ON), 5-3
 - effect on the logic stack, 4-10, 5-3
 - Or Load (OLD), 5-13
 - effect on logic stack, 5-13
 - Outputs, 5-5
 - Positive Transition (EU), 5-3
 - Program Control, 5-11–5-12
 - Reset (R), 5-5
 - Return from Interrupt (RETI), 4-5, 5-14–5-17
 - Set (S), 5-5
 - standard contacts, 5-3
 - Timer, 5-6–5-7
 - Unconditional End (MEND), 4-5
 - Up/Down Counter (CTUD), 5-8
 - Watchdog Reset (WDR), 5-11–5-13
 - Interrupt instructions, 5-14–5-16
 - Disable Interrupt (DISI), 5-14–5-17
 - Enable Interrupt (ENI), 5-14–5-17
 - example, 5-16
 - Interrupt Routine (INT), 5-14–5-17
 - Return from Interrupt (RETI), 5-14–5-17
 - Interrupt Routine (INT) instruction, 5-14–5-17
 - Interrupts
 - effect on the Watchdog Reset (WDR)
 - instruction, 5-11
 - enabling and disabling, 5-14
 - example, 4-14–4-16, 5-16
 - guidelines and restrictions, 5-15
 - hardware interrupt, 4-5
 - affecting the scan cycle, 4-6–4-9
 - high-speed counter, 4-14–4-16, 5-16
 - instructions, 5-14
 - Interrupt instructions, 5-14–5-16
 - response time, 1-3
 - CPU 210 AC/AC/Relay, A-8
 - CPU 210 AC/DC/Relay, A-6
 - CPU 210 DC/DC/DC, A-4
 - PDS 210, A-10
 - sample interrupt routine, 4-14–4-16, 5-16
 - sharing data between main program and
 - interrupt, 5-15
 - system support, 5-15
 - Isolated DC wiring guidelines, 1-10
- J**
- Jump to Label instruction, 5-12
- K**
- Keyword, “network”, 2-8

L

LAD. See Ladder or Program

Ladder

- basic elements, 4-9
 - boxes, 4-9
 - changing to STL, 2-9
 - changing elements in a program, 2-7, 3-16–3-20
 - coils, 4-9
 - contacts, 4-9
 - displaying status, 2-16
 - editor, 2-7
 - entering a program, 2-7, 3-15–3-20
 - inserting an instruction, 3-16–3-20
 - networks, 2-7, 4-9
 - program examples
 - contact instructions, 5-4
 - counter, 5-8
 - End (MEND), 5-12
 - high-speed counter, 4-14–4-16, 5-16
 - increment/decrement word, 5-9
 - interrupt, 4-14–4-16, 5-16
 - jump to label, 5-12
 - logic stack (ALD and OLD), 5-13
 - move word, 5-10
 - output instructions, 5-5
 - timer, 5-7
 - watchdog reset, 5-12
 - sample program, 3-1–3-25
 - sharing data between main program and interrupt, 5-15
 - tools of the Ladder Editor, 3-15
 - using the Ladder Editor, 2-7, 3-15–3-20
 - viewing a program, 2-9
- Last scan time, stored in special memory (SM), B-2
- Load (LD) / Load Not (LDN), effect on the logic stack, 5-3
- Load (LD) instruction, 5-3
 - effect on the logic stack, 4-10
- Load Not (LDN) instruction, 5-3
- Loading a program
 - CPU 210, 2-11
 - PDS 210, 2-10–2-12
- Location of the analog adjustment potentiometer, 4-16
- Logic stack, 4-10
 - affected by the interrupt routine, 5-15
 - effect of Or (O) / And (A) / Load (LD), 4-10
 - effect of Or Load (OLD) / And Load (ALD), 5-13

- Logic Stack instructions, 5-13
 - And Load (ALD), 5-13
 - effect on logic stack, 5-13
 - example, 5-13–5-15
 - operation, 5-13
 - Or Load (OLD), 5-13
 - effect on logic stack, 5-13

M

- Main Program End (MEND) instruction, 4-5, 5-11
- Main program segment, 4-5
- Manuals, order number, F-1
- Maximum scan time, stored in special memory (SM), B-2
- Memory areas, 4-4
 - accessing data, 4-11
 - addressing bit memory (M), 4-12
 - addressing special memory (SM), 4-12
 - addressing the inputs, 4-12
 - addressing the outputs, 4-12
 - bit memory (M), 4-11
 - CPU 210, 4-11–4-13
 - PDS 210, 4-11–4-13
 - saved in the CPU 210, 2-12
- Memory cartridge
 - copying a program from the PDS 210, 2-11
 - location
 - CPU 210 AC/AC/Relay, A-9
 - CPU 210 AC/DC/Relay, A-7
 - CPU 210 DC/DC/DC, A-5
 - PDS 210, A-11
 - order number, F-1
 - power up with an empty memory cartridge, 2-12
 - scan cycle, 4-6–4-8
 - specifications, A-12, A-13
 - transporting the program to the CPU 210, 2-10–2-12
- MEND instruction, 5-11
- Menu of editing functions, right mouse button
 - Status Chart, 2-15
 - Symbol Table, 2-14
- Minimum scan time, stored in special memory (SM), B-2
- Monitoring
 - program, 2-16
 - sample program, 3-23

Mounting

- agency certifications and specifications, A-3
- clearance requirements, 1-4
- dimensions
 - CPU 210, 1-5
 - DIN rail, 1-5
 - PDS 210, 1-5
- in a panel box, 1-7
- on a DIN rail, 1-6
- on a panel, 1-6
- screw size, 1-6

Move instruction, Move Word (MOVW), 5-10

Move Word (MOVW) instruction, 5-10

Move Word (MOVW) instructions, example, 5-10–5-12

N

Negative Transition (ED) instruction, 5-3

Networks

- in ladder, 2-7
- in STL, 2-8
- keyword “network”, 2-8
- represented in ladder, 4-9

Non-fatal errors, responding to, 2-18

Normally Closed Contact instruction, 5-3

Normally Open Contact instruction, 5-3

NOT instruction, 5-3

Number of characters per symbol, 2-13

Number of symbols allowed, 2-13

Numbers, representation of, 4-11

O

OB1. *See* Program

On-Delay Timer instruction, 5-6

Online help, STEP 7-Micro/WIN, 2-1

Operand, 4-10

Or (O) / Or Not (ON), effect on the logic stack, 5-3

Or (O) instruction, 5-3

- effect on the logic stack, 4-10

Or Load (OLD) instruction, 5-13

- effect on the logic stack, 5-13

Or Not (ON) instruction, 5-3

Order numbers, F-1

Output (coil) instruction, 5-5

- represented in ladder, 4-9

Output instructions, 5-5

- coil, 5-5
- example, 5-5
- represented in ladder, 4-9
- Reset (R), 5-5
- Set (S), 5-5

Outputs

- addressing, 4-12
- basic operation, 4-4
- represented in ladder, 4-9
- writing to, 4-6–4-9

Overview

- CPU 210, 1-1–1-4
- PDS 210, 1-1–1-4

P**Panel box**

- See also* Enclosure
- installing in a, 1-7

Paste, copy, and cut

- in a Status Chart, 2-15
- in a Symbol Table, 2-14

PC/PPI cable

- baud rate, 2-3
- order number, F-1
- pin assignments, A-14
- specifications, A-14

PDS 210

- agency approvals, A-2
- analog adjustment
 - location of the potentiometer, 4-16
 - value stored in special memory (SM), B-2
- basic operation, 4-4
- baud rate, 2-3
- compile rule violations, C-1
- debug option, scan cycle, 4-8
- dimensions, 1-5
- downloading a program, 2-10–2-12
- electromagnetic specifications, A-3
- environmental specifications, A-2
- equipment requirements, 1-2
- execution times, E-1
- fatal errors, C-1
- general technical specifications, A-2
- high potential isolation test, A-3
- input simulator, order number, F-1
- interrupt routine, guidelines and restrictions, 5-15
- logic stack, 4-10
- memory areas, 4-11–4-13
- memory cartridge location, A-11
- order numbers, F-1
- organizing the program, 4-5
- product overview, 1-1–1-4
- scan cycle, 4-6–4-9
 - debug option, 4-8
 - effect of Watchdog Reset (WDR) instruction, 5-11

- scan times in special memory (SM), B-2
 - special memory (SM), B-1
 - specifications, A-10
 - DC input simulator, A-15
 - summary of features, 1-2
 - transporting a program to CPU 210, 2-10–2-12
 - Pin assignments, PC/PPI cable, A-14
 - Positive Transition (EU) instruction, 5-3
 - Potentiometer
 - analog adjustment
 - accessing the analog value, 4-12
 - changing the value, 4-16
 - nominal range, 4-16
 - sample program, 4-16
 - value stored in special memory (SM), B-2
 - location, 4-16
 - value of analog adjustment, B-2
 - Power flow, effect on execution times, E-1
 - Powering up with an empty memory cartridge, 2-12
 - PPI (point-to-point interface), communications, 2-3
 - Preferences, setting in STEP 7-Micro/WIN, 2-5
 - Product overview
 - CPU 210, 1-1–1-4
 - PDS 210, 1-1–1-4
 - Program
 - addressing bit memory (M), 4-12
 - addressing special memory (SM), 4-12
 - addressing the inputs, 4-12
 - addressing the outputs, 4-12
 - boxes, 4-9
 - coils, 4-9
 - compiling, 2-8
 - concepts, 4-4
 - contacts, 4-9
 - creating, 2-7–2-11
 - creating a high-speed counter, 4-14–4-16, 5-16
 - debugging, 2-16
 - downloading, 2-10
 - ending the interrupt routine, 4-5
 - ending the main program segment, 4-5
 - examples
 - analog adjustment, 4-16
 - contact instructions, 5-4
 - counter, 5-8
 - high-speed counter, 4-14–4-16, 5-16
 - increment/decrement word, 5-9
 - interrupt routine, 4-14–4-16, 5-16
 - jump to label, 5-12
 - logic stack instructions, 5-13
 - main program end, 5-12
 - move word, 5-10
 - output instructions, 5-5
 - timer, 5-7
 - watchdog reset, 5-12
 - executing, 4-6–4-9
 - guidelines and limitations for converting files, D-2
 - high-speed counter example, 4-14–4-16, 5-16
 - importing files from STEP 7-Micro/DOS, D-1
 - languages, 4-9–4-11
 - monitoring, 2-16
 - networks, 4-9
 - sample program, 3-1–3-25
 - system requirements, 3-1
 - scan cycle, 4-6–4-9
 - PDS 210, 4-7–4-9
 - setting preferences, 2-5
 - structure, 4-5
 - viewing, 2-9
 - Program Control instructions, 5-11–5-12
 - End (MEND), 5-11
 - example
 - End (MEND), 5-12–5-14
 - jump to label, 5-12–5-14
 - watchdog reset, 5-12–5-14
 - Jump to Label, 5-12
 - Watchdog Reset (WDR), 5-11–5-13
 - Program development station. *See* PDS 210
 - Programming concepts, 4-4
 - Programming languages, 4-9–4-11
 - Programming software, order number, F-1
 - Project
 - creating, 2-6
 - sample program, 3-13
 - saving, 2-6
- ## R
- Ranges
 - analog adjustment value, 4-16, B-2
 - integer values, 4-11
 - Reading the inputs, 4-6–4-9
 - Status chart, 2-15
 - Reading values in a Status Chart
 - continuous read, 2-15
 - single read, 2-15
 - stop read option, 2-15
 - Removing the access cover, 1-7
 - Replacing elements in a program
 - ladder, 2-7, 3-16–3-20
 - STL, 2-8, 3-15
 - Reset (R) instruction, 5-5
 - Resetting the counter, 5-8
 - Resetting the timer, 5-6
 - Resolution of the timers, 5-6
 - Restarting the CPU, after a fatal error, 2-17–2-19
 - Restoring memory in the CPU 210, 2-12
 - Return from Interrupt (RETI) instruction, 4-5, 5-14–5-17

- Right mouse button
 - Status Chart editor, 2-15
 - Symbol Table, 2-14
- Rungs of logic. *See* Networks

- S**
- S7-200, technical specifications, A-2
- Sample program
 - compiling, 3-21
 - control logic, 3-4–3-8
 - creating a project, 3-13
 - creating a Status Chart, 3-22
 - creating symbol table, 3-14
 - designing the control logic, 3-4–3-8
 - download, 3-23
 - entering program in ladder, 3-15–3-21
 - inputs and outputs, 3-2
 - ladder, 3-1–3-25
 - ladder program, 3-9
 - monitoring, 3-23
 - program examples
 - contact instructions, 5-4
 - counter, 5-8
 - End (MEND), 5-12
 - high-speed counter, 4-14–4-16, 5-16
 - increment/decrement word, 5-9
 - interrupt, 4-14–4-16, 5-16
 - jump to label, 5-12
 - logic stack (ALD and OLD), 5-13
 - move word, 5-10
 - output instructions, 5-5
 - timer, 5-7
 - watchdog reset, 5-12
 - saving, 3-21
 - statement list, 3-1–3-25
 - STL program, 3-11
 - symbolic names, 3-2
 - system requirements, 3-1
- Saving the logic stack during interrupt routine, 5-15
- Saving your program, 2-6, 3-21
 - after converting files to STEP 7-Micro/WIN, D-2
 - saving a project, 2-6
- Scan cycle, 4-6–4-9
 - debug option, 4-8
 - effect of system clock, E-1
 - effect of Watchdog Reset (WDR) instruction, 5-11
 - execution times, E-1
 - interrupting, 4-5, 4-6–4-9
 - PDS 210, 4-7–4-9
 - scan times stored in special memory (SM), B-2
- Screw size for installation, 1-6
- Set (S) instruction, 5-5
- Setting the baud rate, 2-3
- Setting up
 - communication parameters, 2-4
 - selecting preferences in STEP 7-Micro/WIN, 2-5
- Single read (Status Chart option), 2-15
 - See also* Continuous read; Status Chart; Write
- Single-phase wiring guidelines, 1-10
- SMW0 status bits, B-1
- SMW2 analog adjustment, current value, B-2
- SMW22 to SMW26 scan times, B-2
- Special memory (SM)
 - addressing, 4-12
 - analog adjustment, accessing the analog value, 4-12
 - clock pulse bits, B-1
 - first scan bit, B-1
 - saved during interrupt routine, 5-15
 - SMW2 analog adjustment, B-2
 - SMW22 to SMW26 scan times, B-2
 - status bits, B-1
 - storing the analog adjustment value, 4-16
 - supporting the interrupt routine, 5-15
- Specifications
 - CPU 210 AC/AC/Relay, A-8
 - CPU 210 AC/DC/Relay, A-6
 - CPU 210 DC/DC/DC, A-4
 - DC input simulator, A-15
 - electromagnetic, A-3
 - environmental, A-2
 - general, 1-3, A-2
 - high potential isolation test, A-3
 - memory cartridge, A-12, A-13
 - PC/PPI cable, A-14
 - PDS 210, A-10
 - S7-200 family, A-2
- Standard contact instructions, 5-3
- Standards, national and international, A-2
- Statement list, 4-9–4-11
 - basic elements, 4-10
 - changing elements in a program, 2-8, 3-15
 - changing to ladder, 2-9
 - creating networks, 2-8
 - editor, 2-8
 - entering a symbolic name, 2-8, 3-14
 - entering an instruction, 2-8
 - entering comments, 2-8
 - execution times for instructions, E-1
 - guidelines for creating a program, 2-8, 4-2–4-3
 - program, entering in STEP 7-Micro/WIN, 2-8

- program examples
 - contact instructions, 5-4
 - counter, 5-8
 - End (MEND), 5-12
 - high-speed counter, 4-14-4-16, 5-16
 - increment/decrement word, 5-9
 - interrupt, 4-14-4-16, 5-16
 - jump to label, 5-12
 - logic stack (ALD and OLD), 5-13
 - move word, 5-10
 - output instructions, 5-5
 - timer, 5-7
 - watchdog reset, 5-12
- sample program, 3-1-3-25
- sharing data between main program and interrupt, 5-15
- viewing a program, 2-9
- Status, displaying in ladder, 2-16
- Status Chart
 - building for sample program, 3-22
 - continuous read option, 2-15
 - editing addresses, 2-15
 - reading and writing variables, 2-15
 - sample program, 3-22
 - single read option, 2-15
 - STEP 7-Micro/WIN, 2-15
 - stop read option, 2-15
 - write option, 2-15
- STEP 7-Micro/DOS, converting files to STEP 7-Micro/WIN, D-1
- STEP 7-Micro/WIN
 - changing elements in a program, 2-8-2-13, 3-15-3-20
 - compiling a program, 2-8, 3-21
 - converting files from STEP 7-Micro/DOS, D-1
 - creating a program, 2-7-2-11, 3-15-3-21
 - creating a project, 2-6, 3-13
 - debug option, scan cycle, 4-8
 - debugging and monitoring the program, 2-16, 3-23-3-25
 - displaying status in ladder, 2-16, 3-23
 - downloading a program, 2-10, 3-23
 - editing a program, 2-8-2-13, 3-15-3-20
 - entering a sample program, 3-1-3-25
 - entering instructions in the program, 3-15-3-20
 - equipment requirements, 2-1
 - installing, 2-2
 - Ladder Editor, 2-7
 - Ladder editor, 2-16
 - online help, 2-1
 - program examples
 - contact instructions, 5-4
 - counter, 5-8
 - End (MEND), 5-12
 - high-speed counter, 4-14-4-16, 5-16
 - increment/decrement word, 5-9
 - interrupt, 4-14-4-16, 5-16
 - jump to label, 5-12
 - logic stack (ALD and OLD), 5-13
 - move word, 5-10
 - output instructions, 5-5
 - timer, 5-7
 - watchdog reset, 5-12
 - programming preferences, 2-5
 - replacing elements in a program, 2-8-2-13, 3-15-3-20
 - sample program (and entering), 3-1-3-25
 - saving a project, 2-6, 3-21
 - Status Chart, 2-15, 3-22
 - STL editor, 2-8
 - Symbol Table, 2-13, 3-14
 - troubleshooting installation, 2-2
 - viewing a program, 2-9
- STL. *See* Statement list or Program
- Stop read (Status Chart option), 2-15
 - See also* Continuous read; Single read; Status Chart; Write
- Storing a program
 - CPU 210, 2-11
 - PDS 210, 2-10-2-12
- Storing the value of the analog adjustment, B-2
- Summary of features, 1-2
- Suppression circuits, guidelines
 - DC relay, 1-12
 - DC transistor, 1-12
- Symbol Table
 - duplicate symbol names, 2-13
 - number of characters per symbol, 2-13
 - number of entries allowed, 2-13
 - sample program, 3-14
 - STEP 7-Micro/WIN, 2-13-2-15
- Symbolic addressing, 2-13-2-15, 3-14, 3-15
 - duplicate symbol names, 2-13
 - entering a symbolic name in STL, 2-8
 - number of characters per symbol, 2-13
 - number of symbols allowed, 2-13
- System clock, effect on scan time, E-1

T

- Terminating the main program segment, 5-11
- Time of last scan, stored in special memory (SM), B-2
- Timer instructions, 5-6–5-7
 - accessing the current value, 4-13
 - addressing, 4-13
 - current value, 4-13, 5-6
 - device number, 4-12
 - example, 5-7
 - On-Delay Timer, 5-6
 - resetting, 5-6
 - resolution, 5-6
 - updating the time value, 5-6
- Transporting a program to the CPU 210, 2-10–2-12
- Troubleshooting
 - compile errors, C-1
 - error handling, 2-17–2-19, C-2
 - error handling for the CPU 210, C-2
 - fatal errors, 2-17–2-19, C-1
 - non-fatal errors, 2-18, C-1
 - powering up with an empty memory cartridge, 2-12
 - STEP 7-Micro/WIN installation, 2-2

U

- Unconditional End (MEND) instruction, 4-5, 5-11
- Up/Down Counter (CTUD) instruction, 5-8

- User manual, order number, F-1
- Using the Ladder Editor, 3-15–3-20

V

- Valid ranges for CPU 210, 1-3, 4-11–4-13, 5-2

W

- Watchdog Reset (WDR) instruction, 5-11–5-13
 - considerations, 5-11
- Windows 3.1, installing STEP 7-Micro/WIN, 2-2
- Windows 95, installing STEP 7-Micro/WIN, 2-2
- Wiring
 - guidelines, 1-8–1-13
 - AC installation, 1-10
 - DC installation, 1-10
 - optional field wiring connector, 1-10
 - order number, F-1
 - suppression circuits, 1-12–1-13
- Wiring diagram
 - CPU 210 AC/AC/Relay, A-9
 - CPU 210 AC/DC/Relay, A-7
 - CPU 210 DC/DC/DC, A-5
 - PDS 210, A-11
- Write (Status Chart option), 2-15
 - See also* Continuous read; Single read; Status Chart
- Writing to the outputs, 4-6–4-9
- Writing values in a Status Chart, 2-15

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:

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



Remarks Form

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

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

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

Additional comments:

