

NEAR EAST UNIVERSITY

Faculty of Engineering

**Department of Electrical and Electronic
Engineering**

**CONTROL ELECTRIC COUNTERS WITH PLC
DEVICE**

**Graduation Project
EE-400**

Student: Buğra Tansu (20000275)

Supervisor: Mr. Özgür Özerdem

Lefkoşa-2001

ACKNOWLEDGMENTS

First I want to thank Mr. Özgür Özerdem to be my advisor. Under his guidance, I successfully overcome many difficulties and learn a lot about PLC's. In each discussion, he explained my question patiently, and I left my quick progress from his advises. He always help me a lot either in my study. I asked him many questions in PLC's and he always answered my question quickly and in detail.

Special thanks to Cemal. With he kind help, I could use Step7-Microwin16, which is called Simatic successfully to perform computational problem. Thanks to faculty Engineering for having such a good computational environment.

I also want to thank my friends in NEU: Bora, Mertsan, Türkay and Cüneyt.

Finally, I want to thank my family, especially my parents. Without their endless support and love for me, I would never achieve my current position.

ABSTRACT

My project aim is control the electric counter with PLC device and CPU 212. First step sensor read the red point on the electric counter and to work out the units that are connected to the counter when they reach the value that we determine.

I can do PLC device with STEP 7-Micro/WIN programming. STEP 7-Micro/WIN is a programming software application for the S7-200 family of programmable logic controllers.

When programming in statement list (STL), in order to ensure that your user program will also display, compile, and run correctly in ladder (LAD).

STEP 7-Micro/WIN automatically compiles a project when you perform a project download. Components that fail to compile will not be downloaded.

The STEP 7-Micro/WIN STL compiler checks all lines for proper comment syntax, makes sure the program contains valid instruction names with the correct number of parameters, and verifies that correct address identifiers are used.

In LAD programs, the basic elements of logic are represented with contacts, coils, and boxes. A set of interconnected elements that make a complete circuit is called a network.

STL program elements are represented by a set of instructions for performing the desired functions.

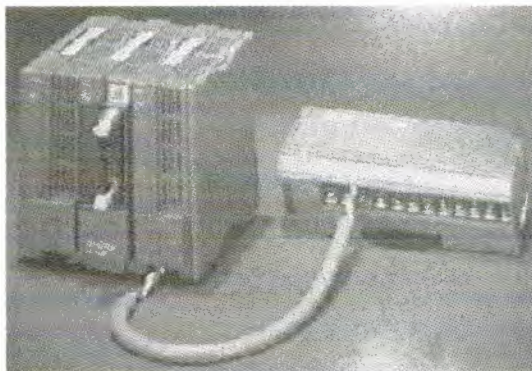
S7-200 programs consist of a main user program that may be followed by subroutines and/or interrupt routines. The main program is terminated by an unconditional END (MEND in STL).

When writing your program, you can use either of two modes of addressing instruction operands; direct or indirect.

Direct addressing specifies the memory area, size, and location, you can address indirectly the data types Q, M, T, C, V, and I.

The user memory in the S7-200 CPUs consists of three blocks; program, data, and configurable parameters.

INTRODUCTION



A PLC (i.e. Programmable Logic Controller) is a device that was invented to replace the necessary sequential relay circuits for machine control. The PLC works by looking at its inputs and depending upon their state, turning on/off its outputs. The user enters a program, usually via software, that gives the desired results.

The first chapter represent, history of PLC and communication began to appear in approximately 1973. The first such system was Modicon's Modbus.

In chapter two, the PLC mainly consists of a CPU, memory areas, and appropriate circuits to receive input/output data.

Chapter three represents, a PLC works by continually scanning a program.

The fourth chapter explain ladder diagram with give examples and explain how we can understand truth table.

Chapter five present, the CPU 212 is the low-cost entry into the SIMATIC S7-200 family; explain functions and how we can use programming.

In chapter six, the S7-200 series is a line of micro-programmable logic controllers that can control a variety of automation applications and explain S7-200 Programming Language.

Chapter seven present direct addressing, counter and timer.

Chapter eight present of the project.

Chapter nine explains part of processing unit and advantages of PLC.

In TRNC PLC devices are used, in local newspaper, medicine factory and washing machine room in a hotel.

PLC is very expensive, but they occupy small area and we can change their program.

PLC is used in communication are widely.

Nowadays, PLC, PC and relay systems are used in the control of Industrial Machines.

From two type of PLCs, although the compact PLC cheaper than the modular PLC, modular PLC is used widely rather than compact PLC.

TABLE OF CONTENTS

| | |
|---|-----|
| ACKNOWLEDGMENTS | i |
| ABSTRACT | ii |
| INTRODUCTION | iii |
| CHAPTER 1 | |
| PLC HISTORY | 1 |
| 1.1 First Introduced | 1 |
| 1.2 Control System | 1 |
| 1.3 Mid70's the dominant PLC technologies | 1 |
| 1.4 Communications | 2 |
| CHAPTER 2 | |
| PLC MAINLY CONSIST | 3 |
| 2.1 What does each part do? | 3 |
| 2.1.1 Input Relays (contacts) | 3 |
| 2.1.2 Internal Utility Relays (contacts) | 3 |
| 2.1.3 Counters | 3 |
| 2.1.4 Timers | 4 |
| 2.1.5 Output Relays (coils) | 5 |
| 2.1.6 Data Storage | 5 |
| CHAPTER 3 | |
| PLC OPERATION | 6 |
| 3.1 PLC Scanning | 6 |
| 3.1.1 Check Input Status (step1) | 6 |
| 3.1.2 Execute Program (step 2) | 6 |
| 3.1.3 Update Output Status (step 3) | 6 |
| 3.2 Response Time Concerns | 7 |
| 3.2.1 Pulse Stretch Function | 8 |
| 3.2.2 Interrupt Function | 8 |

CHAPTER 4

| | |
|--------------------------------------|----|
| PLC REGISTERS | 10 |
| 4.1 Ladder Diagram and PLC Registers | 10 |
| 4.2 Truth Table | 11 |
| 4.3 The Program Scan | 12 |

CHAPTER 5

| | |
|-------------------------|----|
| CPU 212 | 15 |
| 5.1 Overview | 15 |
| 5.2 Area of application | 15 |
| 5.3 Design | 15 |
| 5.4 Functions | 16 |
| 5.5 Programming | 18 |

CHAPTER 6

| | |
|--|----|
| S7-200 MICRO PLC | 19 |
| 6.1 Introducing the S7-200 Micro PLC | 19 |
| 6.2 Comparing the Features of the S7-200 Micro PLCs | 19 |
| 6.2.1 Equipment Requirements | 19 |
| 6.2.2 Capabilities of the S7-200 CPUs | 20 |
| 6.3 Major Components of the S7-200 Micro PLC | 21 |
| 6.3.1 <i>S7-200 CPU Module</i> | 21 |
| 6.4 Installing and Using the STEP 7-Micro/WIN Software | 22 |
| 6.5 Installing the Step 7-micro/win software | 22 |
| 6.5.1 Pre-Installation Instruction | 22 |
| 6.5.2 Installation Instruction for windows 3.1 | 23 |
| 6.5.3 Installation instruction for Windows 95 or windows NT 4.0 | 23 |
| 6.5.4 Troubleshooting the Installation | 24 |
| 6.6 Using STEP7-Micro/WIN to set up to the Communications Hardware | 24 |
| 6.6.1 General Information for Installing or Removing the Communications Hardware | 24 |
| 6.7 Special Hardware Installation Information for Windows NT Users | 25 |

| | |
|---|----|
| 6.8 Establishing Communication with the S7-200 CPU | 26 |
| 6.8.1 Connection Your Computer to the S7-200 CPU | |
| Using the PC/PPI Cable | 26 |
| 6.8.2 Connecting your computer to the S7-200 | |
| Using the MPI or CP Card | 27 |
| 6.8.3 From what point do I set up Communications? | 29 |
| 6.9 Setting Up Communication within STEP 7-Micro/WIN | 29 |
| 6.10 Setting up communication from the Windows Control Panel | 30 |
| 6.11 Setting up communication during installation | 31 |
| 6.12 Selecting the correct module parameter set and setting it up | 31 |
| 6.13 Concept of an S7-200 Program | 32 |
| 6.13.1 Relating the program to inputs and outputs | 32 |
| 6.14 Concepts of the S7-200 Programming Languages | 33 |
| 6.14.1 Understanding the basic elements of ladder logic | 33 |
| 6.14.2 Understanding the statement list instruction | 34 |
| 6.15 Basic Elements for Constructing a Program | 35 |
| 6.15.1 Organizing a Program | 35 |
| 6.15.2 Example Program using subroutines and interrupts | 37 |
| 6.16 Selecting the Mode of Operation for the CPU | 37 |
| 6.16.1 Changing the operating mode with the mode switch | 38 |
| 6.16.2 Changing the operating mode with STEP7-Micro/WIN | 38 |
| 6.16.3 Changing the operating mode from the program | 38 |
| CHAPTER 7 | |
| DIRECT ADDRESSING | 39 |
| 7.1 Direct Addressing of the CPU Memory Areas | 39 |
| 7.1.1 Using the Memory Address to Access Data | 39 |
| 7.2 Timers and Counter | 40 |
| 7.2.1 Addressing the Timer Memory Area | 40 |
| 7.2.2 Addressing the Counter Memory Area | 41 |
| 7.2.3 On-Delay Timer, Retentive On-Delay Timer | 42 |
| 7.2.4 Understanding the S7-200 Timer instruction | 43 |

| | |
|---|----|
| 7.2.5 Updating Timers with 1ms Resolution | 44 |
| 7.2.6 Updating Timers with 10ms Resolution | 45 |
| 7.2.7 Updating Timers with 100ms Resolution | 45 |
| 7.2.8 Updating the Timer Current Value | 46 |
| 7.2.9 Count Up Counter, Count Up/Down Counter | 48 |
| 7.2.10 Understanding the High-Speed Counter Instruction | 49 |
| 7.3 Addressing a local and expansion I/O | 50 |
| 7.3.1 Examples of local and expansion I/O | 51 |
| 7.4 Using the Selectable input filter to provide noise rejection | 53 |
| 7.5 Using the output table to configure the states of the outputs | 53 |
| 7.6 Analog Adjustments | 54 |
| CHAPTER 8 | |
| GRADUATION PROJECT | 56 |
| 8.1 Explanation of the Project | 56 |
| 8.2 Program of the Project | 57 |
| 8.3 PLC Programmable Logic controller | 60 |
| 8.3.1 Compact PLC's | 60 |
| 8.3.2 Modular PLC's | 60 |
| 8.3.3 Input Unit | 60 |
| 8.3.4 Output Unit | 60 |
| 8.3.5 Programming Technique | 61 |
| CHAPTER 9 | |
| PROCESSING UNIT | 62 |
| 9.1 System Memory | 62 |
| 9.2 CPU | 62 |
| 9.3 Program Memory | 62 |
| 9.4 Data Bus | 62 |
| 9.5 Image Register | 62 |
| 9.6 PLC Operating System | 63 |
| 9.7 User Program Operating | 63 |
| 9.7.1 When the PLC is in RUN mode | 63 |

| | |
|------------------------------------|----|
| 9.8 Accessing Data Memory | 64 |
| 9.8.1 Bit Access | 64 |
| 9.8.2 Byte word double word access | 64 |
| 9.9 Advantage PLC | 64 |
| <hr/> | |
| CONCLUSION | 65 |
| REFERENCES | 66 |
| APENDIX | 67 |
| CPU 212 Product Image | 67 |
| CPU 212 Technical data | 70 |

CHAPTER 1

PLC HISTORY

1.1 First Introduced

In the late 1960's PLCs were first introduced. The primary reason for designing such a device was eliminating the large cost involved in replacing the complicated relay based machine control systems. Bedford Associates (Bedford, MA) proposed something called a Modular Digital Controller (MODICON) to a major US car manufacturer. Other companies at the time proposed computer based schemes, one of which was based upon the PDP-8. The MODICON 084 brought the world's first PLC into commercial production.

1.2 Control System

When production requirements changed so did the control system. This becomes very expensive when the change is frequent. Since relays are mechanical devices they also have a limited lifetime, which required strict adherence to maintenance schedules.

Troubleshooting was also quite tedious when so many relays are involved. Now picture a machine control panel that included many, possibly hundreds or thousands, of individual relays. The size could be mind-boggling. How about the complicated initial wiring of so many individual devices! These relays would be individually wired together in a manner that would yield the desired outcome.

These "new controllers" also had to be easily programmed by maintenance and plant engineers. The lifetime had to be long and programming changes easily performed. They also had to survive the harsh industrial environment. That's a lot to ask! The answers were to use a programming technique most people were already familiar with and replace mechanical parts with solid-state ones.

1.3 Mid70's the dominant PLC technologies

In the mid70's the dominant PLC technologies were sequencer state-machines and the bit-slice based CPU. The AMD 2901 and 2903 were quite popular in Modicon and A-B PLCs. Conventional microprocessors lacked the power to quickly solve PLC logic in all but the smallest PLCs. As conventional microprocessors evolved, larger and larger PLCs were

being based upon them. However, even today some are still based upon the 2903.(ref A-B's PLC-3) Modicon has yet to build a faster PLC than their 984A/B/X which was based upon the 2901.

1.4 Communications

Communications abilities began to appear in approximately 1973. The first such system was Modicon's Modbus. The PLC could now talk to other PLCs and they could be far away from the actual machine they were controlling. They could also now be used to send and receive varying voltages to allow them to enter the analog world. Unfortunately, the lack of standardization coupled with continually changing technology has made PLC communications a nightmare of incompatible protocols and physical networks. Still, it was a great decade for the PLC!

The 80's saw an attempt to standardize communications with General Motor's manufacturing automation protocol (MAP). It was also a time for reducing the size of the PLC and making them software programmable through symbolic programming on personal computers instead of dedicated programming terminals or handheld programmers. Today the world's smallest PLC is about the size of a single control relay!

The 90's have seen a gradual reduction in the introduction of new protocols, and the modernization of the physical layers of some of the more popular protocols that survived the 1980's. The latest standard (IEC 1131-3) has tried to merge plc-programming languages under one international standard. We now have PLCs that are programmable in function block diagrams, instruction lists, C and structured text all at the same time! PC's are also being used to replace PLCs in some applications. The original company who commissioned the MODICON 084 has actually switched to a PC based control system.

CHAPTER 2

PLC MAINLY CONSISTS

The PLC mainly consists of a CPU, memory areas, and appropriate circuits to receive input/output data. We can actually consider the PLC to be a box full of hundreds or thousands of separate relays, counters, timers and data storage locations. Do these counters, timers, etc. really exist? No, they don't "physically" exist but rather they are simulated and can be considered software counters, timers, etc. These internal relays are simulated through bit locations in registers (Figure 2.1)

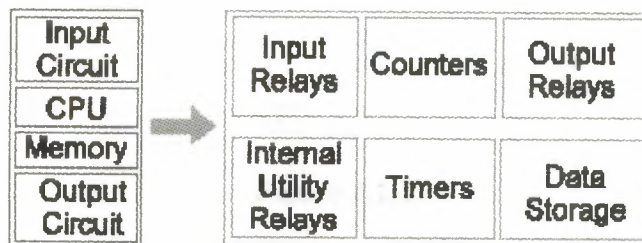


Figure 2.1 PLC mainly consist

2.1 What does each part do?

2.1.1 Input Relays (contacts)

These are connected to the outside world. They physically exist and receive signals from switches, sensors, etc. Typically they are not relays but rather they are transistors.

2.1.2 Internal Utility Relays (contacts)

These do not receive signals from the outside world nor do they physically exist. They are simulated relays and are what enables a PLC to eliminate external relays. There are also some special relays that are dedicated to performing only one task. Some are always on while some are always off. Some are on only once during power-on and are typically used for initializing data that was stored.

2.1.3 Counters

These again do not physically exist. They are simulated counters and they can be programmed to count pulses. Typically these counters can count up, down or both up and

down. Since they are simulated they are limited in their counting speed. Some manufacturers also include high-speed counters that are hardware based. We can think of these as physically existing. Most times these counters can count up, down or up and down. (Symbol shows Figure 2.1.1). The Count Up/Down (CTUD) box counts up on rising edges of the Count Up (CU) input. It counts down on the rising edges of the Count Down (CD) input. It resets when the Reset (R) input turns on.

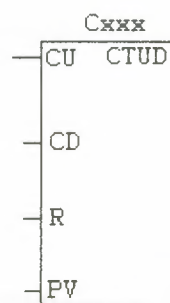


Figure 2.1.1

2.1.4 Timers

These also do not physically exist. They come in many varieties and increments. The most common type is an on-delay type. Others include off-delay and both retentive and non-retentive types. Increments vary from 1ms through 1s. The On-Delay Timer (TON) box times up to the maximum value when the enabling Input (IN) comes on. When the current value (Txxx) is \geq the Preset Time (PT), the timer bit turns on. It resets when the enabling input goes off. Timing stops upon reaching the maximum value.

In the status chart, you can display timer and counter values as either bits or words. If you display a timer or counter value as a bit, the output status is displayed (output on or off). If you display a timer or counter value as a word, the current value is used.

The On-Delay Timers time in one of three resolutions, depending on the timer number you use. Each increment of the current value is a multiple of the time base. For example, a preset of 20 for a 10-millisecond timer represents 200 milliseconds. (Symbol shows in Figure 2.1.2)

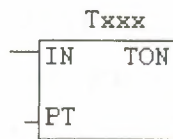


Figure 2.1.2

2.1.5 Output Relays (coils)

These are connected to the outside world. They physically exist and send on/off signals to solenoids, lights, etc. They can be transistors, relays, or triacs depending upon the model chosen.

2.1.6 Data Storage

Typically there are registers assigned to simply store data. They are usually used as temporary storage for math or data manipulation. They can also typically be used to store data when power is removed from the PLC. Upon power-up they will still have the same contents as before power was removed and very convenient and necessary.

CHAPTER 3

PLC OPERATION

3.1 PLC Scanning

A PLC works by continually scanning a program (Figure 3.1). We can think of this scan cycle as consisting of 3 important steps. There are typically more than 3 but we can focus on the important parts and not worry about the others. Typically the others are checking the system and updating the current internal counter and timer values.

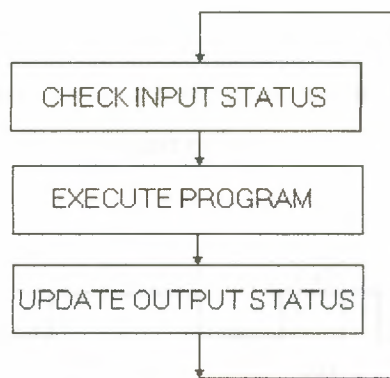


Figure 3.1 Scanning a program

3.1.2 Execute Program (step 2)

Next the PLC executes your program one instruction at a time. Maybe your program said that if the first input was on then it should turn on the first output. Since it already knows which inputs are on/off from the previous step it will be able to decide whether the first output should be turned on based on the state of the first input. It will store the execution results for use later during the next step.

3.1.3 Update Output Status (step 3)

Finally the PLC updates the status of the outputs. It updates the outputs based on which inputs were on during the first step and the results of executing your program during the second step. Based on the example in step 2 it would now turn on the first output because the first input was on and your program said to turn on the first output when this condition

is true.

After the third step the PLC goes back to step one and repeats the steps continuously.

3.2 Response Time Concerns

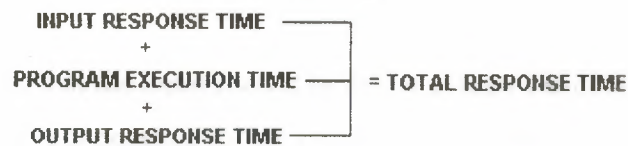


Figure 3.2 Response Time Concerns

Now that we know about response time, here's what it really means to the application. The PLC can only see an input turn on/off when it's looking. In other words, it only looks at its inputs during the check input status part of the scan.

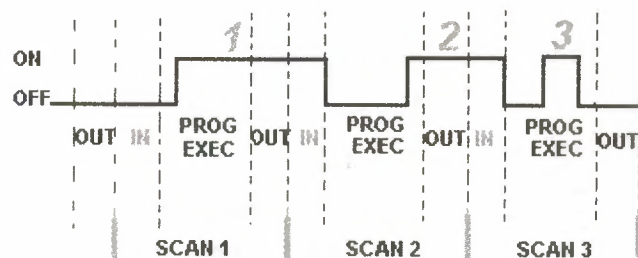


Figure 3.3

In the Figure 3.3, input 1 is not seen until scan 2. This is because when input 1 turned on, scan 1 had already finished looking at the inputs.

Input 2 is not seen until scan 3. This is also because when the input turned on scans 2 had already finished looking at the inputs.

Input 3 is never seen. This is because when scan 3 was looking at the inputs, signal 3 was not on yet. It turns off before scan 4 looks at the inputs. Therefore signal 3 is never seen by the PLC.

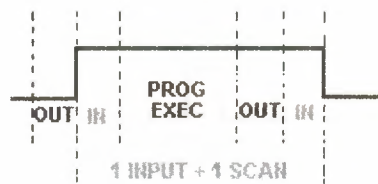


Figure 3.4

To avoid this we say that the input should be on for at least 1 input delay time + one scan time. But what if it was not possible for the input to be on this long? Then the PLC doesn't see the input turn on. Therefore it becomes a paperweight. Not true, of course there must be a way to get around this. Actually there are 2 ways.

3.2.1 Pulse Stretch Function

This function extends the length of the input signal until the PLC looks at the inputs (Figure 3.5) during the next scan (i.e. it stretches the duration of the pulse.)

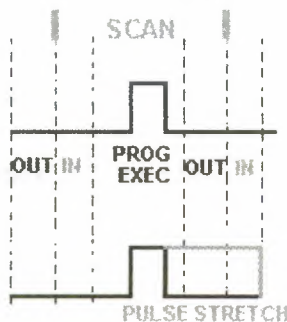


Figure 3.5

3.2.2 Interrupt Function

This function interrupts the scan to process a special routine that you have written (Figure 3.6) i.e. As soon as the input turns on, regardless of where the scan currently is, the PLC immediately stops what its doing and executes an interrupt routine. (A routine can be thought of as a mini program outside of the main program.) After its done executing the interrupt routine, it goes back to the point it left off at and continues on with the normal scan process.

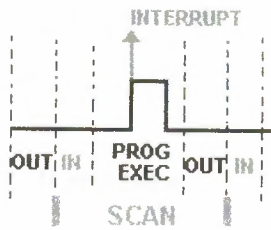


Figure 3.6

Now let's consider the longest time for an output to actually turn on. Let's assume that when a switch turns on we need to turn on a load connected to the PLC output.

The Figure 3.7 below shows the longest delay (worst case because the input is not seen until scan 2) for the output to turn on after the input has turned on.

The maximum delay is thus 2 scan cycles - 1 input delay time.

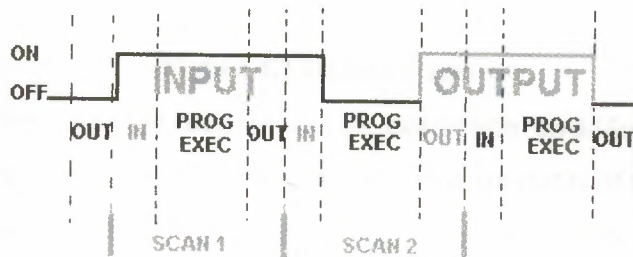


Figure 3.7 The longest delay

CHAPTER 4

PLC REGISTERS

4.1 Ladder Diagram and PLC Registers

We'll now change switch 2 (SW2) to a normally closed symbol (load bar instruction). SW1 will be physically OFF and SW2 will be physically ON initially. The ladder diagram shows in the Figure 4.1.

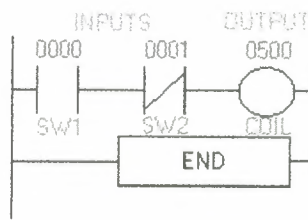


Figure 4.1 Ladder diagram

Notice also that we now gave each symbol (or instruction) an address. This address sets aside a certain storage area in the PLC's data files so that the status of the instruction (i.e. true/false) can be stored. Many PLCs use 16 slot or bit storage locations. In the example above (Table 4.1) we are using two different storage locations or registers.

Table 4.1 Two different storage locations or registers

| REGISTER 00 | | | | | | | | | | | | | | | |
|-------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 15 | 14 | 13 | 12 | 11 | 10 | 09 | 08 | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |
| | | | | | | | | | | | | | | 1 | 0 |
| REGISTER 05 | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 09 | 08 | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |
| | | | | | | | | | | | | | | | 0 |

In the table 4.2 above we can see that in register 00, bit 00 (i.e. input 0000) was logic 0 and bit 01 (i.e. input 0001) was logic 1. Register 05 shows that bit 00 (i.e. output 0500) was logic 0. The logic 0 or 1 indicates whether an instruction is False or True.

Table 4.2 The register

| LOGICAL CONDITION OF SYMBOL | | | |
|-----------------------------|-------|-------|-------|
| LOGIC BITS | LD | LDB | OUT |
| Logic 0 | False | True | False |
| Logic 1 | True | False | True |

The PLC will only energize an output when all conditions on the rung are TRUE. So, looking at the table above, we see that in the previous example SW1 has to be logic 1 and SW2 must be logic 0. Then and only then will the coil be true (i.e. energized). If any of the instructions on the rung before the output (coil) are false then the output (coil) will be false (not energized).

In LAD programs, the basic elements of logic are represented with contacts, coils, and boxes. A set of interconnected elements that make a complete circuit is called a network.

A hard-wired input is represented by a symbol called a contact. A normally open contact enables power flow when closed. A contact can also be normally closed. In this case, power flow occurs when the contact is opened.

4.2 Truth Table

Let's now look at a truth table 4.3 of our previous program to further illustrate this important point. Our truth table will show all possible combinations of the status of the two inputs.

Table 4.3 Truth Table

| Inputs | | Outputs | Register Logic Bits | | |
|---------|----------|-----------|---------------------|----------|-----------|
| SW1(LD) | SW2(LDB) | COIL(OUT) | SW1(LD) | SW2(LDB) | COIL(OUT) |
| False | True | False | 0 | 0 | 0 |
| False | False | False | 0 | 1 | 0 |
| True | True | True | 1 | 0 | 1 |
| True | False | False | 1 | 1 | 0 |

Notice from the chart that as the inputs change their states over time, so will the outputs. The output is only true (energized) when all preceding instructions on the rung are true.

4.3 The Program Scan

Let's watch what happens in this program scan by scan with Figure 4.2, Figure 4.3, Figure 4.4 and Figure 4.5.

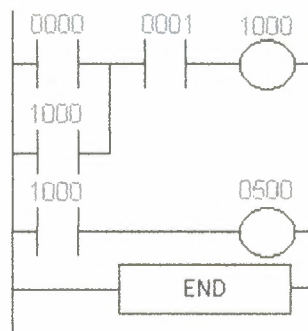


Figure 4.2

Initially the tank is empty. Therefore, input 0000 is TRUE and input 0001 is also TRUE.

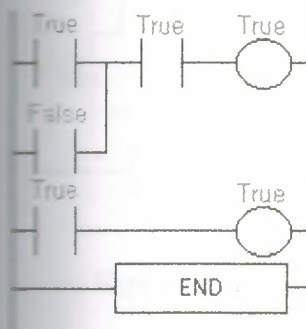


Figure 4.3

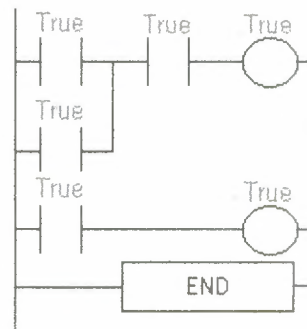


Figure 4.4

After 100 scans the oil level rises above the low level sensor and it becomes open. (i.e. FALSE)

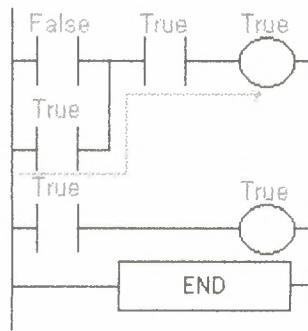


Figure 4.5

Notice that even when the low level sensor is false there is still a path of true logic from left to right. This is why we used an internal relay. Relay 1000 is latching the output (500) on. It will stay this way until there is no true logic path from left to right. (i.e. when 0001 becomes false) After 1000 scans the oil level rises above the high level sensor at it also becomes open (i.e. false) (Figure 4.6 and Figure 4.7)

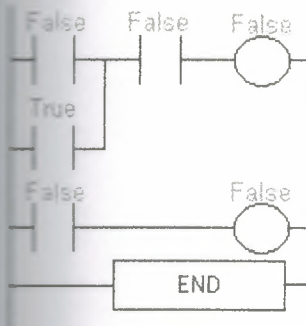


Figure 4.6

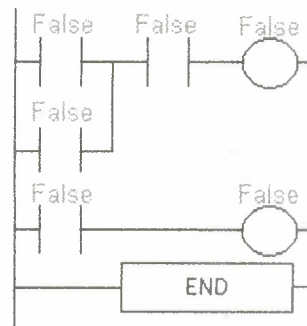


Figure 4.7

Since there is no more true logic path, output 500 is no longer energized (true) and therefore the motor turns off.

After 1050 scans the oil level falls below the high level sensor and it will become true again (Figure 4.8)

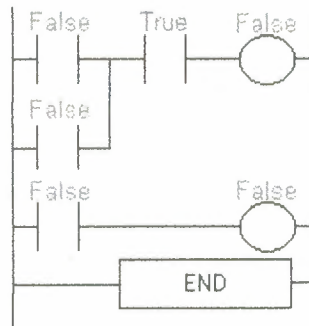


Figure 4.8

CHAPTER 5

CPU 212

5.1 Overview

- Low-cost entry into the SIMATIC S7-21x Series
- All round talent with a wide spectrum of connectable expansion modules
- With analog value processing

5.2 Area of application

The CPU 212 is the low-cost entry into the SIMATIC S7-200 family. A wide range of connectable expansion modules not only opens up the world of analog value processing but also makes the CPU a real all round talent.

5.3 Design

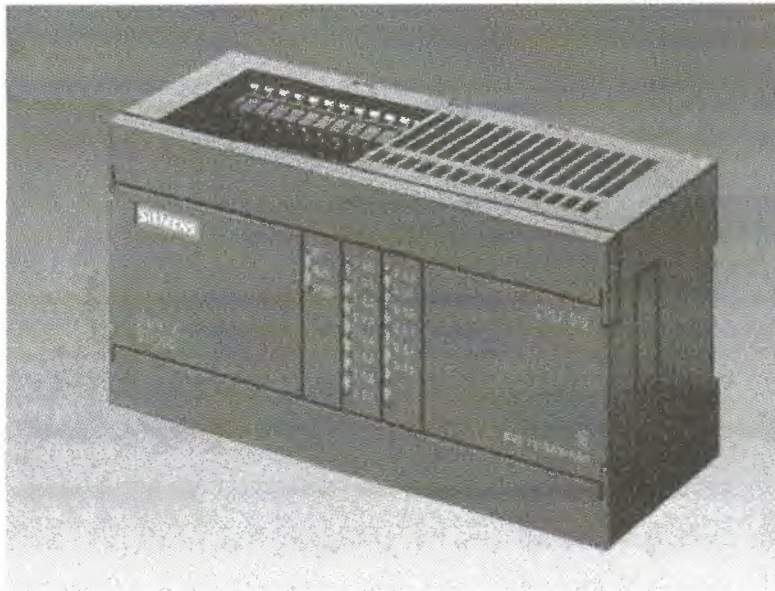


Figure 5.1

The CPU 212 (Figure 5.1) features:

- Integrated 24 V transmitter and load power supply;
for direct connection of sensors and transmitters. With its 180mA output current it can also be used to supply loads.
- 7 variants;
with different supply voltages and control voltages.
- Integrated digital inputs and outputs;
8 inputs and 6 outputs.
- Interrupt inputs;
for extremely rapid response to rising or falling edges of process signals.
- High-speed counter;
1 high-speed counter (2 kHz), for implementation as an up or down counter.
- Problem-free expansion with digital and analog expansion modules (EM, optional).
- Simulator (optional);
for simulating the integrated inputs and testing the user program.
- Analog potentiometer;
1 analog potentiometer, easy-to-use in everyday operation as a set point adjuster, e.g. for setting timers.

5.4 Functions

- Comprehensive instruction set;
numerous basic operations such as binary logic operations, result assignment, storing, counting, setting up timers, loading, transferring, comparing, shifting, rotating, generating complements, calling subroutines, integrated communication instructions (e.g. RECEIVE Freeport) and user-friendly functions such as pulse-width modulation, pulse sequence function, arithmetic functions, jump functions, loop functions and code conversions aid programming.
- Counting;
user-friendly counter functions in conjunction with the integrated counters open up new applications for the user.

- Interrupt handling;
- Edge-triggered interrupts (activated by rising or falling edges of process signals on interrupt inputs) support a rapid response to process events.
- Timed interrupts can be set from 5 ms to 255 ms at intervals of 1 ms.
- Counter interrupts can be triggered when a set point is reached or when the direction of counting changes.
- Communication interrupts support the fast and easy exchange of information with I/O devices, e.g. printers or barcode readers.
- Direct scanning and control of inputs and outputs;
inputs and outputs can also be directly scanned and set independently of the cycle. The controller is then able to respond quickly to process events (e.g. direct resetting of outputs on the occurrence of an interrupt).
- Password protection;
the three-level password protection concept provides effective protection for company expertise. The protection concept features the following modes of access to the user program:
 - Complete access: The program can be changed as required.
 - Read-only: The program is protected against unauthorized modification. Testing, setting system parameters and copying the program are all possible.
 - Complete protection: The program is protected against modifications and unauthorized reading and copying. Parameters can be set.
- Test and diagnosis functions;
user-friendly functions support test and diagnosis: The complete program is executed over a number of cycles that can be specified and analyzed. Internal parameters, such as bit-memories, timers or counters are logged over up to 124 cycles.
- "Forcing" of inputs and outputs in test and diagnosis mode;
inputs and outputs can be set independent of the cycle and therefore permanently, for the purpose of testing the user program for example.

5.5 Programming

The program packages STEP 7-Micro/DOS V1.3, STEP 7-Micro/WIN16 V2.6 or STEP 7-Micro/WIN32 V3.0 are available for programming the CPU 212.

Every function of the CPU can be programmed using these packages. If programming is performed via the serial interface of the programming device or PC, a PC/PPI cable will also be necessary.

If the programming software STEP 7-Micro/WIN32 V3.0 is used, it is also possible to program the CPU via the SIMATIC CPs CP 5511 or CP 5611. In this manner, communication rates of up to 187 kbit/s are possible.

CHAPTER 6

S7-200 MICRO PLC

6.1 Introducing the S7-200 Micro PLC

The S7-200 series is a line of micro-programmable logic controllers (Micro PLCs) that can control a variety of automation applications. Figure 6-1 shows an S7-200 Micro PLC. The compact design, expandability, low cost, and powerful instruction set of the S7-200 Micro PLC make a perfect solution for controlling small applications. In addition, the wide variety of CPU sizes and voltages provides you with the flexibility you need to solve your automation problems.

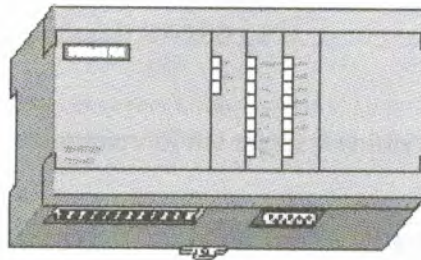


Figure 6.1 S7-200 Micro PLC

6.2 Comparing the Features of the S7-200 Micro PLCs

6.2.1 Equipment Requirements

Figure 6.2 shows the basic S7-200 Micro PLC system, which includes an S7-200 CPU module, a personal computer, STEP 7-Micro/WIN programming software, and a communications cable. In order to use a personal computer (PC), you must have one of the following sets of equipment:

- A PC/PPI cable
- A communications processor (CP) card and multipoint interface (MPI) cable
- A multipoint interface (MPI) card. A communications cable is provided with the MPI card.

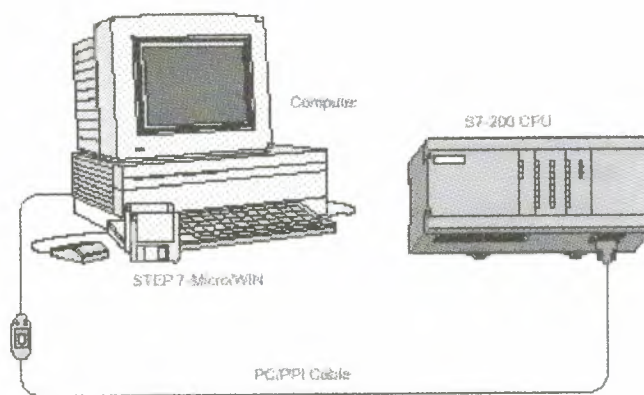


Figure 6.2 The basic S7-200 Micro PLC system

6.2.2 Capabilities of the S7-200 CPUs

The S7-200 family includes a wide variety of CPUs. This variety provides a range of features to aid in designing a cost-effective automation solution. Table 6.1 provides a summary of the major features of each S7-200 CPU.

Table 6.1 Provides a summary of the major features of each S7-200 CPU.

| Features | CPU 212 | CPU 214 | CPU 215 | CPU 216 |
|----------------------------|-------------------------|-------------------------|-------------------------|-------------------------|
| Physical Size of Unit | 160 mm x 80 mm x62mm | 197 mm x 80 mm x62mm | 218 mm x 80 mm x62mm | 218 mm x 80 mm x62mm |
| Memory | | | | |
| Program (EEPROM) | 512 words | 2kwords | 4kwords | 4kwords |
| User data | 512 words | 2kwords | 2.5kwords | 2.5kwords |
| Internal memory bits | 128 | 256 | 256 | 256 |
| Memory cartridge | none | Yes(EEPROM) | Yes(EEPROM) | Yes(EEPROM) |
| Optional battery cartridge | none | 200 days typical | 200 days typical | 200 days typical |
| Backup upper capac | 50 hours typical | 190 hours typical | 190 hours typical | 190 hours typical |
| Inputs/Outputs (I/O) | | | | |
| Local I/O | 8 DI/6 DQ | 14 DI/10 DQ | 14 DI/10 DQ | 24 DI/16 DQ |
| Expansion modules (max.) | 2 modules | 7 modules | 7 modules | 7 modules |
| Process-image I/O | 64 DI/64 DQ | 64 DI/64 DQ | 64 DI/64 DQ | 64 DI/64 DQ |
| Analog I/O (expansion) | 16 AL/16 AQ | 16 AL/16 AQ | 16 AL/16 AQ | 16 AL/16 AQ |
| Selectable input filters | No | yes | yes | yes |
| Instructions | | | | |
| Boolean execution speed | 1.2µs/instruction | 0.8µs/instruction | 0.8µs/instruction | 0.8µs/instruction |
| Counters/timers | 64/64 | 128/128 | 256/256 | 256/256 |
| For/next loops | No | Yes | Yes | Yes |
| Integer math | Yes | Yes | Yes | Yes |
| Real math | No | Yes | Yes | Yes |
| PID | No | No | Yes | Yes |

| Additional Features | | | | |
|---------------------------------------|---------------------------|---------------------------|-------------------------------|--|
| High speed counter | 1 S/W | 1 S/W, 2H/W | 1 S/W, 2H/W | 1 S/W, 2H/W |
| Analog adjustments | 1 | 2 | 2 | 2 |
| Pulse outputs | None | 2 | 2 | 2 |
| Communication interrupt events | 1 transmit/ 1 receiver | 1 transmit/ 1 receiver | 1 transmit/ 2 receiver | 2 transmit/ 4 receiver |
| Timed interrupts | 1 | 2 | 2 | 2 |
| Hardware input interrupts | 1 | 4 | 4 | 4 |
| Real time clock | None | Yes | Yes | Yes |
| Communications | | | | |
| Number of common ports | 1 (RS-485) | 1 (RS-485) | 2 (RS-485) | 2 (RS-485) |
| Protocol supported Port 0: Port 1: | PPI, Freeport N/A | PPI, Freeport N/A | PPI, Freeport, MPI DP, MPI | PPI, Freeport, MPI PPI, Freeport, MPI |
| Peer to peer | Slave only | Yes | Yes | Yes |

6.3 Major Components of the S7-200 Micro PLC

An S7-200 Micro PLC consists of an S7-200 CPU module alone or with a variety of optional expansion modules.

6.3.1 S7-200 CPU Module

The S7-200 CPU module 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 power supply provides electrical power for the base unit and for any expansion module that is connected.
- 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.
- The communications port allows you to connect the CPU to a programming device or to other devices. Some S7-200 CPUs have two communications ports.
- Status lights provide visual information about the CPU mode (RUN or STOP), the current state of the local I/O, and whether a system fault has been detected.

6.4 Installing and Using the STEP 7-Micro/WIN Software

STEP 7-Micro/WIN is a Windows-based software application that supports both the 16-bit Windows 3.1 environment (STEP 7-Micro/WIN 16) and the 32-bit Windows 95 and Windows NT environments (STEP 7-Micro/WIN 32). In order to use STEP 7-Micro/WIN, the following equipment is recommended:

- Recommended: a personal computer (PC) with an 80586 or greater processor and 16 Mbytes of RAM, or a Siemens programming device (such as a PG 740);
minimum computer requirement: 80486 processor with 8 Mbytes
- One of the following sets of equipment:
- A PC/PPI cable connected to your communications port (PC COM1 or COM2)
- A communications processor (CP) card and multipoint interface (MPI) cable
- A multipoint interface (MPI) card (A communications cable comes with the MPI card.)
- VGA monitor, or any monitor supported by Microsoft Windows
- At least 50 Mbytes of free hard disk space
- Microsoft Windows 3.1, Windows for Workgroups 3.11, Windows 95, or Windows NT 4.0 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.

6.5 Installing the STEP 7-Micro/WIN Software

6.5.1 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 STEP 7-Micro/WIN projects to diskette.
- Make sure all applications are closed, including the Microsoft Office toolbar. Installation may require that you restart your computer.

6.5.2 Installation Instructions for Windows 3.1

If you have Windows 3.1 (Windows for Workgroups 3.11) on your machine, use the following procedure to install the STEP 7-Micro/WIN 16 software:

1. Start by inserting Disk 1 in the disk drive of your computer (usually 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 "OK" or press ENTER. This starts the setup procedure.
4. Follow the online setup procedure to complete the installation.

6.5.3 Installation Instructions for Windows 95 or Windows NT 4.0

If you have Windows 95 or Windows NT 4.0 on your machine, use the following procedure to install the STEP 7-Micro/WIN 32 software:

1. Start by inserting Disk 1 in the disk drive of your computer (usually 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 "OK" or press ENTER. This starts the setup procedure.
5. Follow the online setup procedure to complete the installation.
6. At the end of the installation, the Install/Remove Modules dialog box appears automatically. See Figure 6.3. You can install the hardware for your machine to communicate now, or you can wait until later.

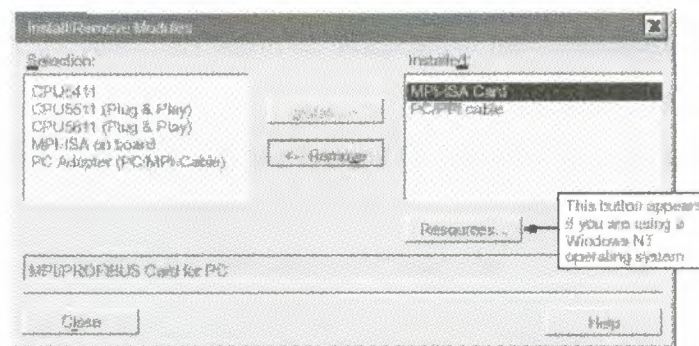


Figure 6.3 Install/Remove Modules Dialog Box

6.5.4 Troubleshooting the Installation

The following situations can cause the installation to fail:

- Not enough memory: at least 50 Mbytes of free space are required 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

Review the README x.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.)

6.6 Using STEP 7-Micro/WIN to Set Up the Communications Hardware

6.6.1 General Information for Installing or Removing the Communications Hardware

If you are using Windows 95 or Windows NT 4.0, the Install/Remove Modules dialog box appears automatically at the end of your software installation. See Figure 6.3. If you are using Windows 3.1, follow these steps:

1. Select the menu command Setup-Communications. The Communications dialog box appears.
2. Click the "PG/PC Interface..." button. The Setting the PG/PC Interface dialog box appears.
3. Click the "Install..." button. The Install/Remove Modules dialog box appears. See Figure 3-1.

You will need to base your installation of communications hardware on the following criteria:

- The operating system that you are using (Windows 3.1, Windows 95, or Windows NT 4.0)
- The type of hardware you are using, for example:
- PC with PC/PPI cable
- PC or SIMATIC programming device with multipoint interface (MPI) or communications processor (CP) card.
- CPU 212, CPU 214, CPU 215, CPU 216

- Modem
- The baud rate you are using

Table 6.2 shows the possible hardware configurations and baud rates that STEP 7-Micro/WIN support, depending on the type of CPU that you are using.

Table 6.2 Hardware Configurations Supported by STEP 7- Micro/WIN

| Type of CPU | STEP 7-Micro/WIN Version | Hardware Supported | Baud Rates Supported | Operating System | Type of Parameter Set |
|---|--------------------------|--|-------------------------|---|-----------------------|
| CPU 212, CPU 214, CPU 216 CPU 215 port 0 | Micro/WIN 1.6 | PC/PP1 cable, MPI-ISA card | 9.6 kbaud or 19.2 kbaud | Windows 3.1 | PP1, PP1 multi-master |
| | | | | Windows 95 or Windows NT | PP1 |
| | Micro/WIN 3.2 | PC/PP1 cable, MPI-ISA card, MPI-ISA card on board, CP 5411, CP 5511, CP 5611 | 9.6 kbaud or 19.2 kbaud | Windows 95 or Windows NT | PP1, PP1 multi-master |
| CPU 215 port 1 (DP port) | Micro/WIN 1.6 | Not supported | Not supported | Windows 3.1 Windows 95 or Windows NT | Not supported |
| | Micro/WIN 3.2 | MPI-ISA card, MPI-ISA card on board, CP 5411, CP 5511, CP 5611 | 9.6 kbaud to 12 Mbaud | Windows 95 or Windows NT | MPI |

6.7 Special Hardware Installation Information for Windows NT Users

Installing hardware modules under the Windows NT operating system is slightly different from installing hardware modules under Windows 95. Although the hardware modules are the same for either operating system, installation under Windows NT requires more knowledge of the hardware that you want to install. Windows 95 tries automatically to set up system resources for you; however, Windows NT does not. Windows NT provides you with default values only. These values may or may not match the hardware configuration.

However, these parameters can be modified easily to match the required system settings.

When you have installed a piece of hardware, select it from the Installed list box and click the "Resources" button. The Resources dialog box appears. See Figure 6.4. The Resources dialog box allows you to modify the system settings for the actual piece of hardware that you installed. If this button is unavailable (gray), you do not need to do anything more.

At this point you may need to refer to your hardware manual to determine the setting for

each of the parameters listed in the dialog box, depending on your hardware settings. You may need to try several different interrupts in order to establish communication correctly.

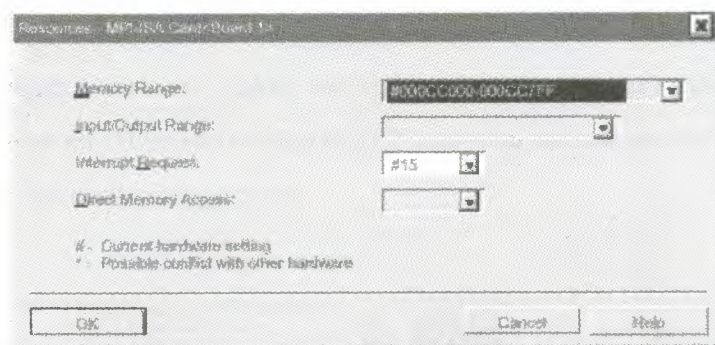


Figure 6.4 Resources Dialog Box for Windows NT

6.3 Establishing Communication with the S7-200 CPU

You can arrange the S7-200 CPUs in a variety of configurations to support network communications. You can install the STEP 7-Micro/WIN software on a personal computer (PC) that has a Windows 3.1x, Windows 95, or Windows NT operating system, or you can install it on a SIMATIC programming device (such as a PG 740). You can use the PC or the programming device as a master device in any of the following communications configurations:

- A single master device is connected to one or more slave devices. See Figure 6.5.
- A single master device is connected to one or more slave devices and one or more master devices. See Figure 6.6 and Figure 6.7.
- A CPU 215 functions as a remote I/O module owned by an S7-300 or S7-400 programmable logic controller or by another PROFIBUS master. See Figure 6.8.
- A single master device is connected to one or more slave devices. This master device is connected by means of 11-bit modems to either one S7-200 CPU functioning as a slave device or else to a network of S7-200 CPUs functioning as slave devices. See Figure 6.9.

6.3.1 Connecting Your Computer to the S7-200 CPU Using the PC/PPI Cable

Figure 6.5 shows a typical configuration for connecting your personal computer to your

CPU with the PC/PPI cable. To establish proper communications between the components, follow these steps:

1. Set the DIP switches on the PC/PPI cable for the baud rate.
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 CPU, and tighten the connecting screws.

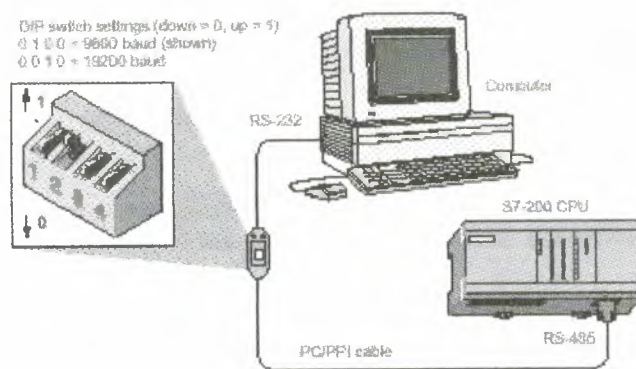


Figure 6.5 Communicating with a CPU in PPI Mode

Figure 6.6 shows a configuration with a personal computer connected to several S7-200 CPU modules. STEP 7-Micro/WIN is designed to communicate with one S7-200 CPU at a time; however, you can access any CPU on the network. The CPU modules in Figure 6.6 could be either slave or master devices. The TD 200 is a master device.

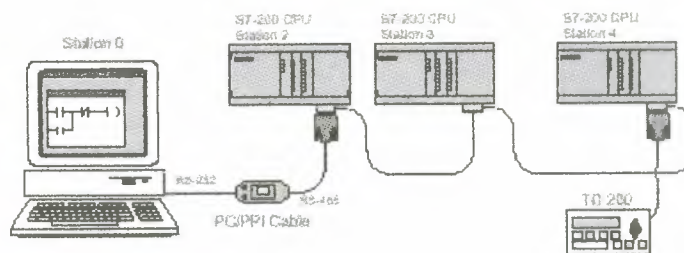


Figure 6.6 Using a PC/PPI Cable for Communicating with Several S7-200 CPU Modules

6.8.2 Connecting Your Computer to the S7-200 CPU Using the MPI or CP Card

You can use STEP 7-Micro/WIN with a multipoint interface (MPI) or communications

processor (CP) card. Either card provides a single RS-485 port for connection to the network using an MPI cable. STEP 7 Micro/WIN 32 (the 32-bit version) supports the MPI parameter set for an MPI network; STEP 7-Micro/WIN 16 (the 16-bit version) does not. After establishing MPI communications, you can connect STEP 7-Micro/WIN on a network that contains other master devices. Each master must have a unique address. Figure 6.7 shows a sample network with master and slave devices.

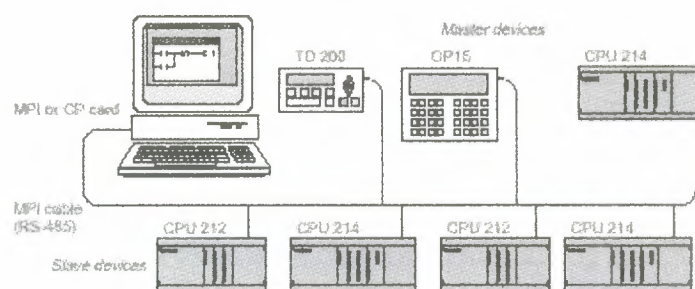


Figure 6.7 Example of an MPI or CP Card with Master and Slave Devices

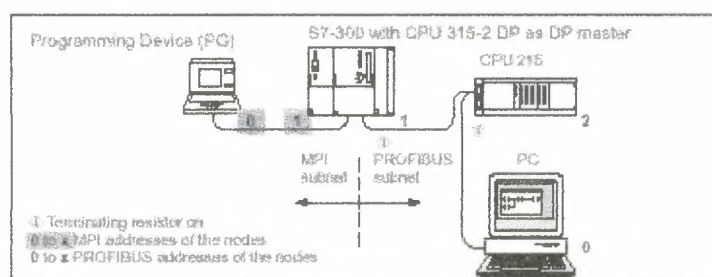


Figure 6.8 CPU 215 on a PROFIBUS Subnetwork, with MPI Subnetwork

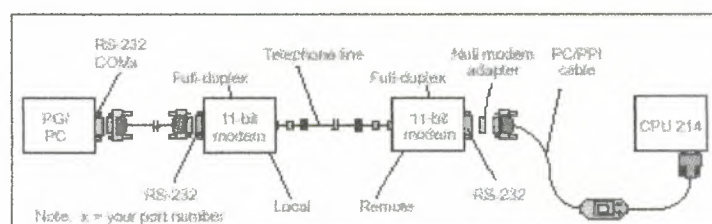


Figure 6.9 S7-200 Data Communications Using an 11-Bit Modem

6.3 From What Point Do I Set Up Communications?

Depending on the operating system that you are using, you can set up communications from any of the following points:

- Under Windows 3.1
 - Within STEP 7-Micro/WIN 16 only
- Under Windows 95 or Windows NT 4.0
- During the final step of the installation (see Section 3.1)
- From the Setting the PG/PC Interface icon, found in the Windows Control Panel
- Within STEP 7-Micro/WIN 32

6.3 Setting Up Communications within STEP 7-Micro/WIN

Within STEP 7-Micro/WIN there is a Communications dialog box that you can use to configure your communications setup. See Figure 6.10. You can use one of the following ways to find this dialog box:

- Select the menu command Setup-Communications....
- Create a new project and click the “Communications...” button in the CPU Type dialog box.
- If you have a project open, select the menu command CPU-Type... and click the “Communications” button in the CPU Type dialog box.

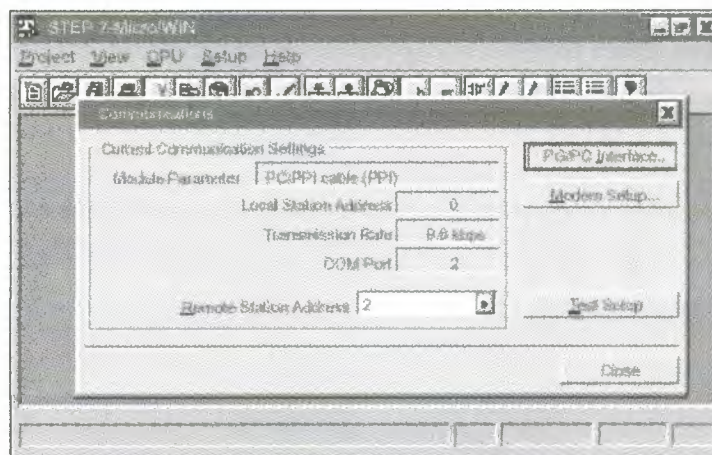


Figure 6.10 Setting Up the Communications between Programming Device or PC and the CPU

After you have called up the Communications dialog box, click the “PG/PC Interface” button. The Setting the PG/PC Interface dialog box appears. See Figure 6.11.

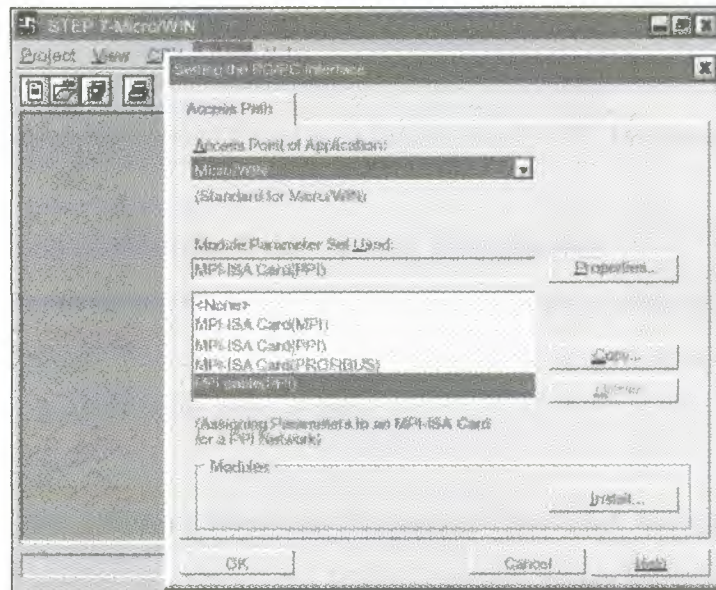


Figure 6.11 Setting the PG/PC Interface Dialog Box

6.10 Setting Up Communications from the Windows Control Panel

If you are using the Windows 95 or Windows NT 4.0 operating system, you can set up the communications configuration by means of the Control Panel. From the Control Panel, select the Setting the PG/PC Interface icon. See Figure 6.12.

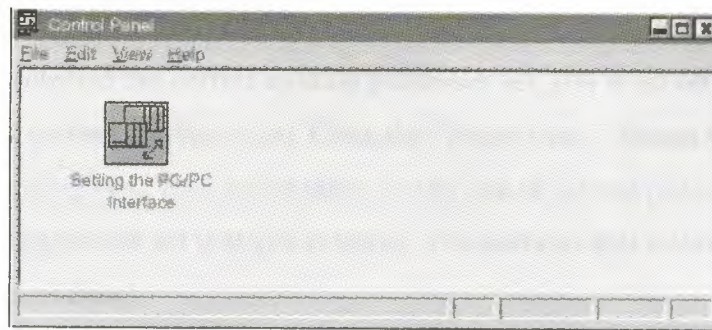


Figure 6.12 Control Panel with Setting the PG/PC Interface Icon

6.11 Setting Up Communications during Installation

Under the Windows 95 or Windows NT 4.0 operating system, at the end of the STEP 7-Micro/WIN installation, the Communications dialog box appears automatically. You can set up your configuration at that time, or later.

6.12 Selecting the Correct Module Parameter Set and Setting It Up

When you have reached the Setting the PG/PC Interface dialog box (see Figure 6.11), you must select “Micro/WIN” in the Access Point of Application list box in the Access Path tab. This dialog box is common to several different applications, such as STEP 7 and WinCC, so you must tell the program the application for which you are setting parameters.

When you have selected “Micro/WIN” and have installed your hardware, you need to set the actual properties for communicating with your hardware. The first step is to determine the protocol that you want to use on your network. See Table 6.2. In most cases, you will use the PPI protocol for all of your CPU modules, except for the high-speed port (DP port) on the CPU 215. This port uses the MPI protocol.

When you have decided what protocol you want to use, you can choose the correct setup from the Module Parameter Set Used list box in the Setting the PG/PC Interface dialog box.

This box lists each hardware type that you have installed, along with the protocol type in parentheses. For example, a simple setup might require you to use the PC/PPI cable to communicate with a CPU 214. In this case, you select “PC/PPI cable (PPI).” Another example is a setup that requires communicating with a CPU 215 through its high-speed port (DP port) by means of a plain MPI-ISA card that you have installed in your computer. In

this case, you select "MPI-ISA Card (MPI)."

After you have selected the correct module parameter set, you must set up the individual parameters for the current configuration. Click the "Properties..." button in the Setting the PG/PC Interface dialog box. This action takes you to one of several possible dialog boxes, depending on the parameter set that you selected. The sections that follow describe each of these dialog boxes in detail.

In summary, to select a module parameter set, follow these steps:

- 1 In the Setting the PG/PC Interface dialog box (see Figure 6.11), select "Micro/WIN" in the Access Point of Application list box in the Access Path tab.
- 2 Ensure that your hardware is installed.
- 3 Determine the protocol that you want to use.
- 4 Select the correct setup from the Module Parameter Set Used list box in the PG/PC Interface dialog box.
- 5 Click the "Properties..." button in the Setting the PG/PC Interface dialog box.

6.13 Concepts of an S7-200 Program

6.13.1 Relating the Program to Inputs and Outputs

The basic operation of the S7-200 CPU is very simple:

- The CPU reads the status of the inputs.
- The program that is stored in the CPU uses these inputs to evaluate the control logic. As

the program runs, the CPU updates the data.

- The CPU writes the data to the outputs.

Figure 6.13 shows a simple diagram of how an electrical relay diagram relates to the S7-200 CPU. 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.

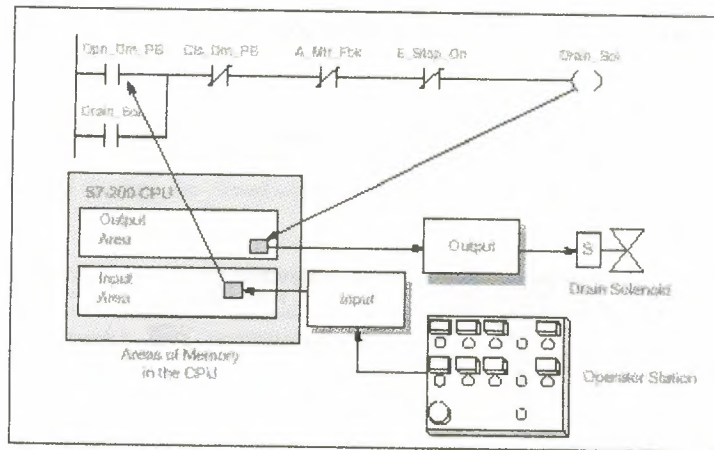


Figure 6.14 Relating the Program to Inputs and Outputs

6.14 Concepts of the S7-200 Programming Languages

The S7-200 CPU (and STEP 7-Micro/WIN) supports the following programming languages:

- Statement list (STL) is a set of mnemonic instructions that represent functions of the CPU.
- Ladder logic (LAD) 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 S7-200 instruction set. There is a direct correspondence between the international and the SIMATIC representation; both representations have the same functionality.

6.14.1 Understanding the Basic Elements of Ladder Logic

When you write a program in ladder, you create and arrange the graphical components to form a network of logic. As shown in Figure 6.15, 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: each of these elements forms a complete circuit. Power flows from the left power rail through the closed contacts to energize the coils or boxes.

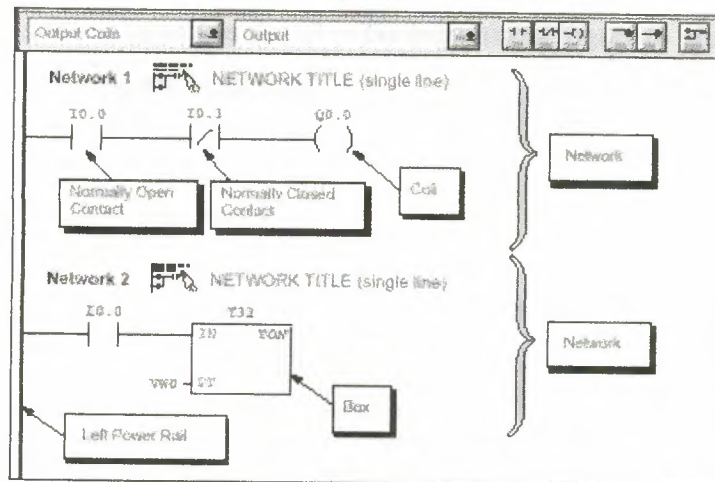


Figure 6.15 Basic Elements of Ladder Logic

6.14.2 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 6.16 shows the basic elements of a statement list program.

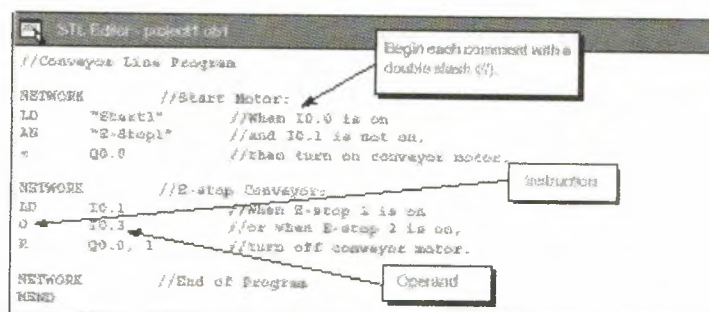


Figure 6-17 STL Editor Window with Sample Program

The STL instructions use a logic stack in the CPU for solving your control logic. As

shown in Figure 6-18, this logic stack is nine 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 6-18 shows examples of how three instructions use the logic stack.

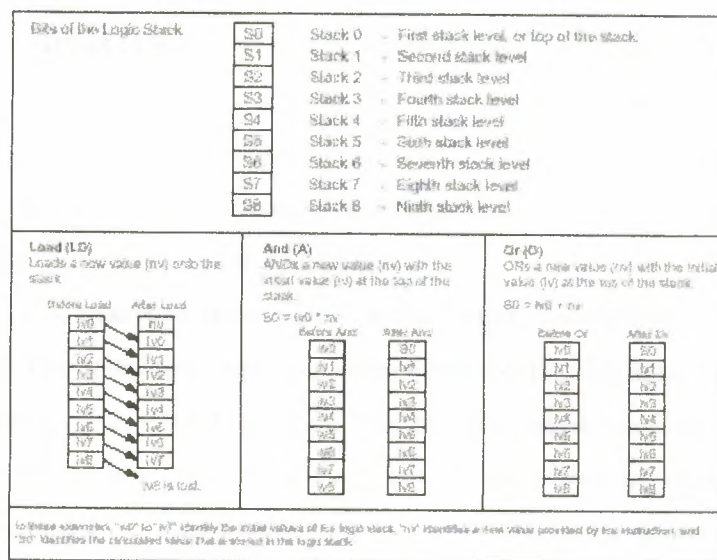


Figure 6-5 Logic Stack of the S7-200 CPU

6.15 Basic Elements for Constructing a Program

The S7-200 CPU continuously executes your program to control a task or process. You create this program with STEP 7-Micro/WIN and download it to the CPU. From the main program, you can call different subroutines or interrupt routines.

6.15.1 Organizing the Program

Programs for an S7-200 CPU are constructed from three basic elements: the main program, subroutines (optional), and interrupt routines (optional). As shown in Figure 6.19, an S7-200 program is structured into the following organizational elements:

- **Main program:** The main body of the program is where you place 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. See in Figure 6.19.

- Subroutines: These optional elements of your program are executed only when called from the main program. Place the subroutines after the end of the main program (following the Unconditional End coil in ladder logic or the MEND instruction in STL). Use a Return (RET) instruction to terminate each subroutine. See in Figure 6.19.
- Interrupt routines: These optional elements of your program are executed on each occurrence of the interrupt event. Place the interrupt routines after the end of the main program (following the Unconditional End coil in ladder logic or the MEND instruction in STL). Use a Return From Interrupt (RETI) instruction to terminate each interrupt routine. See in Figure 6.19.

Subroutines and interrupt routines follow the Unconditional End coil or MEND instruction of the main program; there is no other requirement for locating the subroutines and interrupt routines within your program. You can mix subroutines and interrupt routines following the main program; however, in order to provide a program structure that is easy to read and understand, consider grouping all of the subroutines together after the main program, and then group all of the interrupt routines together after the subroutines.

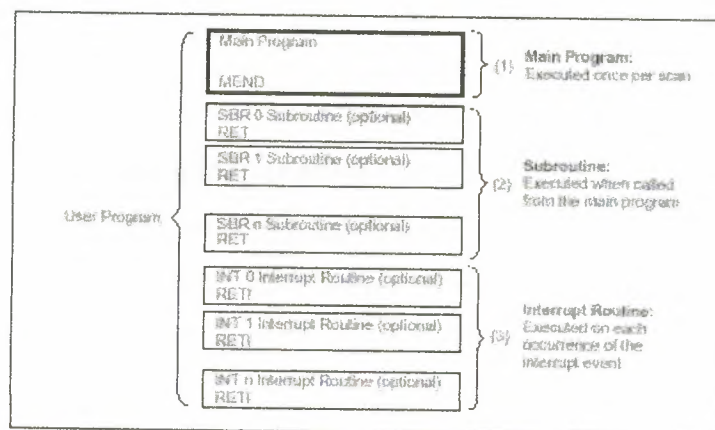


Figure 6.19 Program Structure for an S7-200 CPU

6.15.2 Example Program Using Subroutines and Interrupts

Figure 6.20 shows a sample program for a timed interrupt, which can be used for applications such as reading the value of an analog input. In this example, the sample rate of the analog input is set to 100 ms.

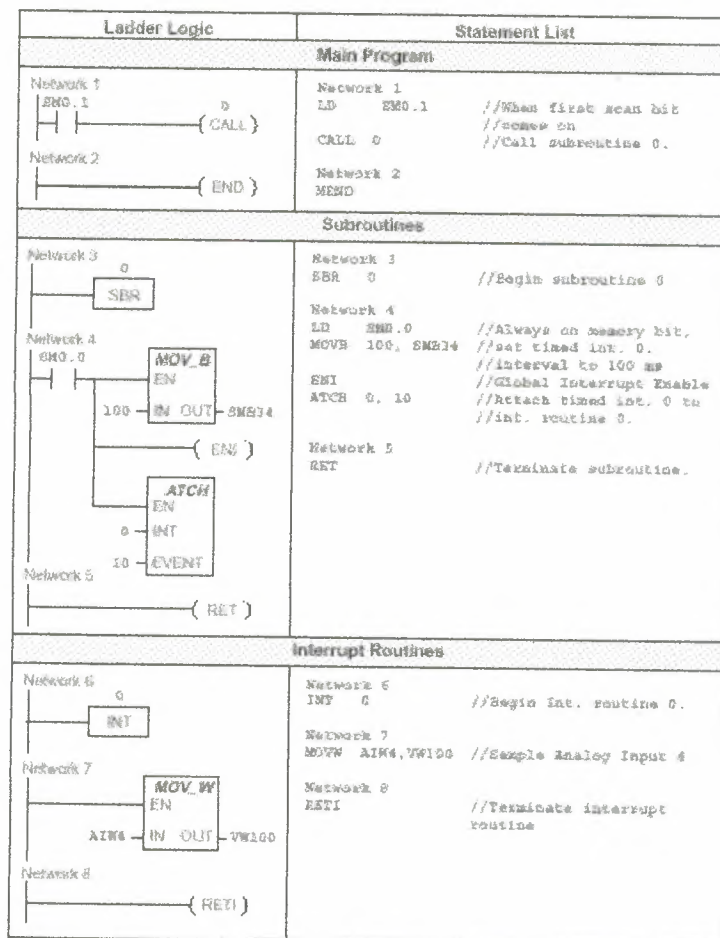


Figure 6.20 Sample Program for Using a Subroutine and an Interrupt Routine

6.16 Selecting the Mode of Operation for the CPU

The S7-200 CPU has two modes of operation:

- **STOP:** The CPU is not executing the program. You can download a program or configure the CPU when the CPU is in STOP mode.
- **RUN:** The CPU is running the program. When the CPU is in RUN mode, you cannot download a program or configure the CPU.

- The status LED on the front of the CPU indicates the current mode of operation. You must place the CPU in the STOP mode to load the program into program memory.

6.16.1 Changing the Operating Mode with the Mode Switch

You can use the mode switch (located under the access door of the CPU module) to select the operating mode for the CPU manually:

- Setting the mode switch to STOP mode stops the execution of the program.
- Setting the mode switch to RUN mode starts the execution of the program.
- Setting the mode switch to TERM (terminal) mode does not change the CPU operating mode, but it does allow the programming software (STEP 7-Micro/WIN) to change the CPU operating mode.

If a power cycle occurs when the mode switch is set to STOP or TERM, the CPU goes automatically to STOP mode when power is restored. If a power cycle occurs when the mode switch is set to RUN, the CPU goes to RUN mode when power is restored.

6.16.2 Changing the Operating Mode with STEP 7-Micro/WIN

As shown in Figure 6.21, you can use STEP 7-Micro/WIN to change the operating mode of the CPU. To enable the software to change the operating mode, you must set the mode switch on the CPU to either TERM or RUN.

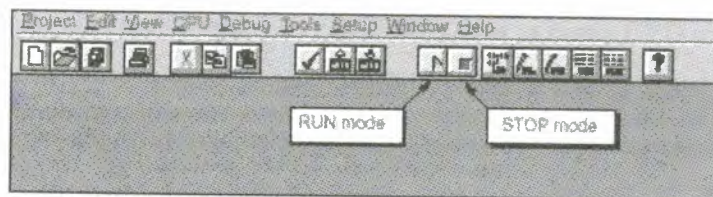


Figure 6-9 Using STEP 7-Micro/WIN to Change the Operating Mode of the CPU

6.16.3 Changing the Operating Mode from the Program

You can insert the STOP instruction in your program to change the CPU to STOP mode. This allows you to halt the execution of your program based on the program logic.

CHAPTER 7

DIRECT ADDRESSING

7.1 Direct Addressing of the CPU Memory Areas

The S7-200 CPU stores information in different memory locations that have unique addresses. You can explicitly identify the memory address that you want to access. This allows your program to have direct access to the information.

7.1.1 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 7.1 shows an example of accessing a bit (which is also called “byte. bit” addressing). In this example, the memory area and byte address (I=input, and 3=byte 3) are followed by a period (“.”) to separate the bit address (bit 4).

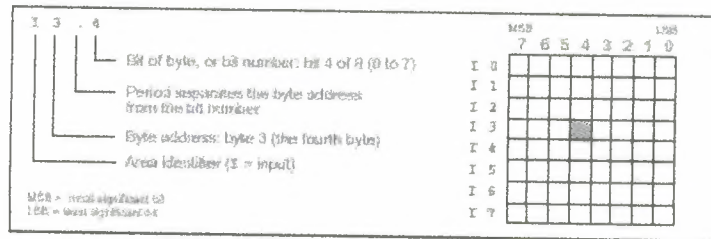


Figure 7.1 Example of accessing bit

By using the byte address format, you can access data in many CPU memory areas (V, I, Q, M, and SM) as bytes, words, or double words. To access a byte, word, or double word of data in the CPU memory, you must specify the address in a way similar to specifying the address for a bit. This includes an area identifier, data size designation, and the starting byte address of the byte, word, or double word value, as shown in Figure 7.2. Data in other CPU memory areas (such as T, C, HC, and the accumulators) are accessed by using an address format that includes an area identifier and a device number.

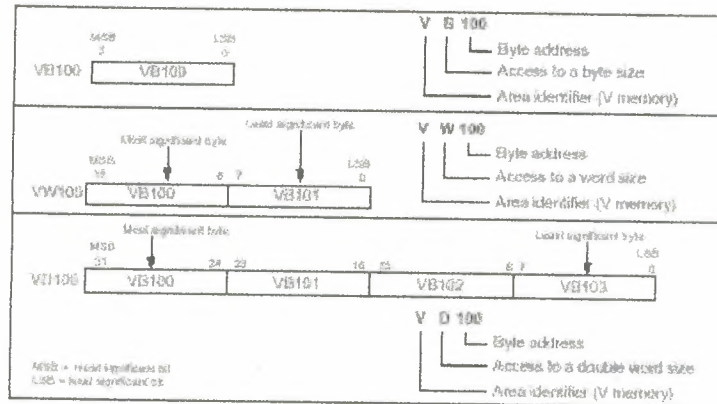


Figure 7.2 Data size designation

7.2 Timers and Counter

7.2.1 Addressing the Timer (T) Memory Area

In the S7-200 CPU, timers are devices that count increments of time. The S7-200 timers have resolutions (time-base increments) of 1 ms, 10 ms, or 100 ms. There are two variables that are associated with a timer:

- **Current value:** this 16-bit signed integer stores the amount of time counted by the timer.
- **Timer bit:** this bit turns on (is set to 1) when the current value of the timer is greater than or equal to the preset value. (The preset value is entered as part of the timer instruction.)

You access both of these variables by using the timer address (T + timer number).

Access to either the timer bit or the current value is dependent on the instruction used: instructions with bit operands access the timer bit, while instructions with word operands access the current value. As shown in Figure 7.3, the Normally Open Contact instruction accesses the timer bit, while the Move Word (MOV_W) instruction accesses the current value of the timer.

Format: T [timer number] T24

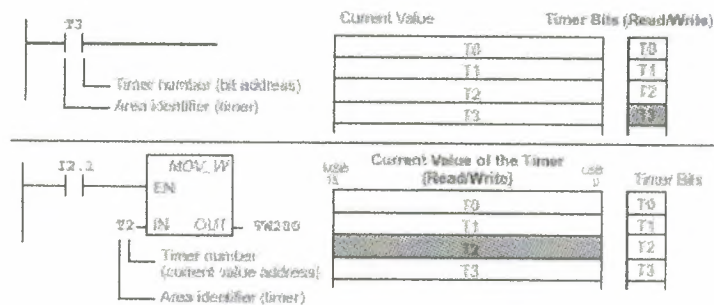


Figure 7.3 Normally Open Contact instruction accesses the timer bit

7.2.2 Addressing the Counter (C) Memory Area

In the S7-200 CPU, counters are devices that count each low-to-high transition event on the counter input(s). The CPU provides two types of counters: one type counts up only, and the other counts both up and down. There are two variables that are associated with a counter:

- **Current value:** this 16-bit signed integer stores the accumulated count.
- **Counter bit:** this bit turns on (is set to 1) when the current value of the counter is greater than or equal to the preset value. (The preset value is entered as part of the counter instruction.)

You access both of these variables by using the counter address ($C + \text{counter number}$).

Access to either the counter bit or the current value is dependent on the instruction used: Instructions with bit operands access the counter bit, while instructions with word operands access the current value. As shown in Figure 7.4, the Normally Open Contact instruction accesses the counter bit, while the Move Word (MOV_W) instruction accesses the current value of the counter.

Format: C [counter number]

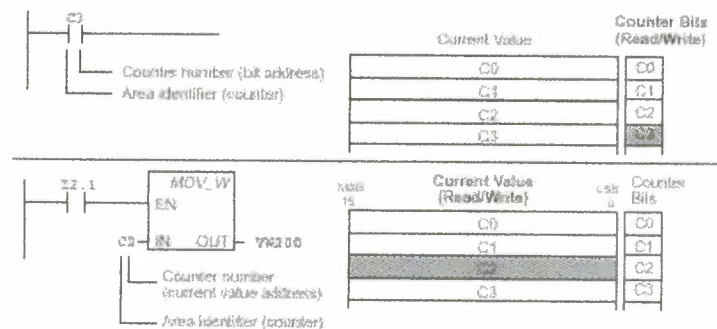


Figure 7.4 The Normally Open Contact instruction accesses the counter bit

7.2.3 On-Delay Timer, Retentive On-Delay Timer

The On-Delay Timer and Retentive On-Delay Timer instruction time up the maximum value when enabled. When the current value (Txxx) is \geq to the Preset Time (PT), the timer bit turns on. The On-Delay timer is reset when disable, while the Retentive On-Delay timer stops timing when disable. Both timers stop timing when disabled. Both timers stop timing when they reach the maximum value. (Figure 7.5)

| Operands: | Txxx: | <u>TON</u> | <u>TONR</u> |
|-----------|-------|---|-------------|
| | 1ms | T32, T96 | T0, T64 |
| | 10ms | T33 to T36 | T1 to T4 |
| | | T97 to T100 | T65 to T66 |
| | 100ms | T37 to T63 | T5 to T31 |
| | | T101 to T255 | T69 to T95 |
| | PT: | VW, T, C, IW, QW, MW, SMW, AC, AIW, Constant, VD, AC, SW | |

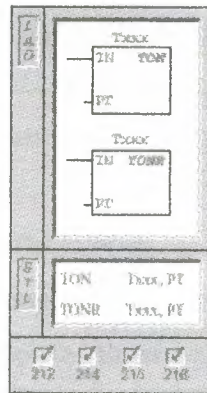


Figure 7.5

TON and TONR timers are available in three resolutions. The resolution is determined by the timer number and is shown in Table 7.1. Each count of the current value is a multiple of the time base. For example, a count of 50 on a 10-millisecond (ms) timer represents 500 ms.

Table 7.1 The resolution is determined by the timer number

| Timer | Resolution | Maximum Value | CPU 212 | CPU 214 | CPU 215/216 |
|-------|------------|--------------------|------------|--------------------------|--------------------------|
| TON | 1 ms | 32,767 seconds (s) | T32 | T32, T96 | T32, T96 |
| | 10 ms | 327.67 s | T33 to T36 | T33 to T36, T97 to T100 | T33 to T36, T97 to T100 |
| | 100 ms | 3276.7 s | T37 to T63 | T37 to T63, T101 to T127 | T37 to T63, T101 to T255 |
| TONR | 1 ms | 32,767 s | T0 | T0, T64 | T0, T64 |
| | 10 ms | 327.67 s | T1 to T4 | T1 to T4, T65 to T96 | T1 to T4, T65 to T96 |
| | 100 ms | 3276.7 s | T5 to T31 | T5 to T31, T97 to T95 | T5 to T31, T97 to T95 |

7.2.4 Understanding the S7-200 Timer Instructions

You can use timers to implement time-based counting functions. The S7-200 provides two different timer instructions: the On-Delay Timer (TON), and the Retentive On-Delay Timer (TONR). The two types of timers (TON and TONR) differ in the ways that they react to the state of the enabling input. Both TON and TONR timers time up while the enabling input is on: the timers do not time up while the enabling input is off, but when the

enabling input is off, a TON timer is reset automatically and a TONR timer is not reset and holds its last value. Therefore, the TON timer is best used when you are timing a single interval. The TONR timer is appropriate when you need to accumulate a number of timed intervals.

S7-200 timers have the following characteristics:

- Timers are controlled with a single enabling input, and have a current value that maintains the elapsed time since the timer was enabled. The timers also have a preset time value (PT) that is compared to the current value each time the current value is updated and when the timer instruction is executed.
- A timer bit is set or reset based upon the result of the comparison of current value to the preset time value.
- When the current value is greater than or equal to the preset time value, the timer bit (T-bit), is turned on.

When you reset a timer, its current value is set to zero and its T-bit is turned off. You can reset any timer by using the Reset instruction, but using a Reset instruction is the only method for resetting a TONR timer. Writing a zero to a timer's current value does not reset its timer bit. In the same way, writing a zero to the timer's T-bit does not reset its current value.

7.2.5 Updating Timers with 1-ms Resolution

The S7-200 CPU provides timers that are updated once per millisecond (1-ms timers) by the system routine that maintains the system time base. These timers provide precise control of an operation.

Since the current value of an active 1-ms timer is updated in a system routine, the update is automatic. Once a 1-ms timer has been enabled, the execution of the timer's controlling TON/TONR instruction is required only to control the enabled/disabled state of the timer.

Since the current value and T-bit of a 1-ms timer are updated by a system routine (independent from the programmable logic controller scan and the user program), the current value and T-bits of these timers can be updated anywhere in the scan and are updated more than once per scan if the scan time exceeds one millisecond. Therefore, these values are not guaranteed to remain constant throughout a given execution of the main user

program.

Resetting an enabled 1-ms timer turns the timer off, resets the timer's current value to zero, and clears the timer T-bit.

7.2.6 Updating Timers with 10-ms Resolution

The S7-200 CPU provides timers that count the number of 10-ms intervals that have elapsed since the active 10-ms timer was enabled. These timers are updated at the beginning of each scan by adding the accumulated number of 10-ms intervals (since the beginning of the previous scan) to the current value for the timer.

Since the current value of an active 10-ms timer is updated at the beginning of the scan, the update is automatic. Once a 10-ms timer is enabled, execution of the timer's controlling TON/TONR instruction is required only to control the enabled or disabled state of the timer.

Unlike the 1-ms timers, a 10-ms timer's current value is updated only once per scan and remains constant throughout a given execution of the main user program.

A reset of an enabled 10-ms timer turns it off, resets its current value to zero, and clears its T-bit.

7.2.7 Updating Timers with 100-ms Resolution

Most of the timers provided by the S7-200 use a 100-ms resolution. These timers count the number of 100-ms intervals that have elapsed since the 100-ms timer was last updated. These timers are updated by adding the accumulated number of 100-ms intervals (since the beginning of the previous scan) to the timer's current value when the timer instruction is executed.

The update of 100-ms timers is not automatic, since the current value of a 100-ms timer is updated only if the timer instruction is executed. Consequently, if a 100-ms timer is enabled but the timer instruction is not executed each scan, the current value for that timer is not updated and it loses time. Likewise, if the same 100-ms timer instruction is executed multiple times in a single scan, the number of 100-ms intervals is added to the timer's current value multiple times, and it gains time. Therefore, 100-ms timers should only be used where the timer instruction is executed exactly once per scan. A reset of a 100-ms

Timer sets its current value to zero and clears its T-bit.

7.2.8 Updating the Timer Current Value

The effect of the various ways in which current time values are updated depends upon how the timers are used. For example, consider the timer operation shown in Figure 7.6.

- In the case where the 1-ms timer is used, Q0.0 is turned on for one scan whenever the timer's current value is updated after the normally closed contact T32 is executed and before the normally open contact T32 is executed.
- In the case where the 10-ms timer is used, Q0.0 is never turned on, because the timer bit T33 is turned on from the top of the scan to the point where the timer box is executed. Once the timer box has been executed, the timer's current value and its T-bit is set to zero. When the normally open contact T33 is executed, T33 is off and Q0.0 is turned off.
- In the case where the 100-ms timer is used, Q0.0 is always turned on for one scan whenever the timer's current value reaches the preset value. By using the normally closed contact Q0.0 instead of the timer bit as the enabling input to the timer box, the output Q0.0 is guaranteed to be turned on for one scan each time the timer reaches the preset value (see Figure 7.6). Figure 7.7 and Figure 7.8 show examples of the Timer instructions for ladder logic and statement list.

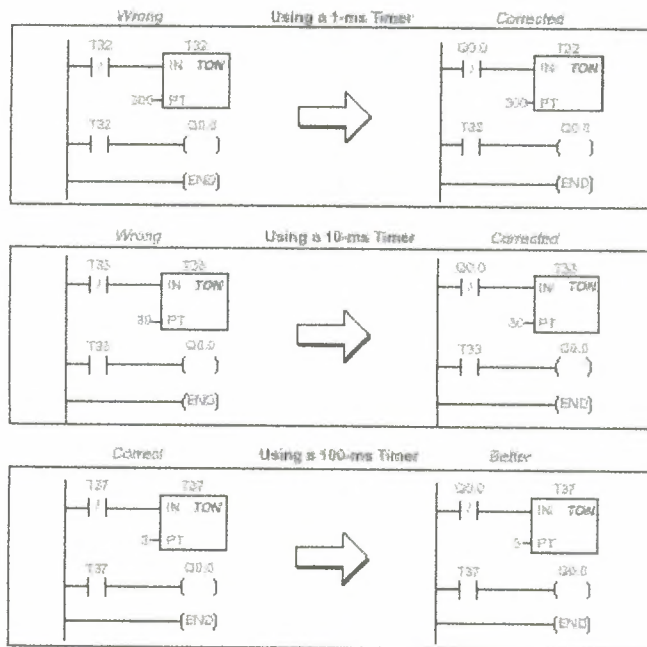


Figure 7.6 Example of Automatically Retargeted One Shot Timer

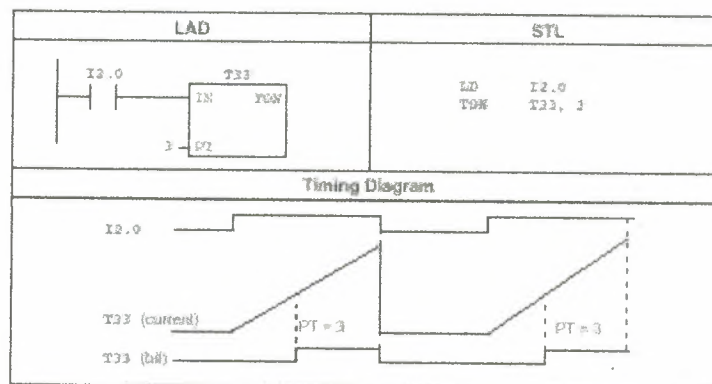


Figure 7.7 Example of On-Delay Timer Instruction for LAD and STL

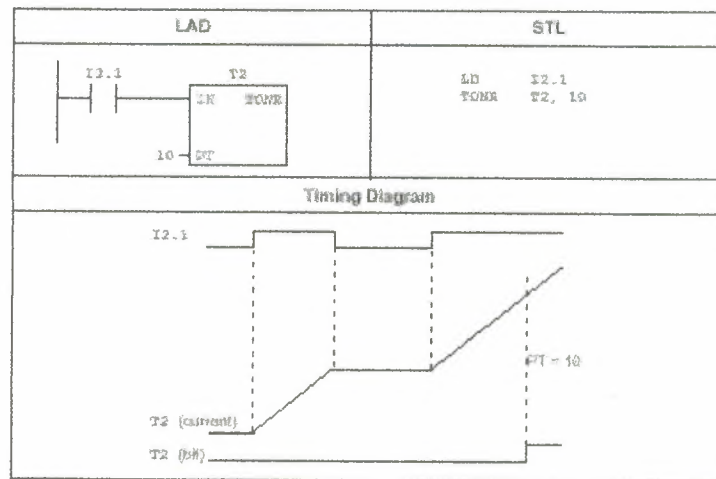


Figure 7.8 Example of Retentive On-Delay Timer Instruction for LAD and STL

7.2.9 Count Up Counter, Count Up/Down Counter

The Count Up instruction counts up to the maximum value on the rising edges of the Count Up (CU) input. When the current value (Cxxx) greater than or equal to the Preset Value (PV), the counter bit (Cxxx) turns on. The counter is reset when the Reset (R) input turns on. In STL, the Reset input is the top of the stack value, while the Count Up input is the value loaded in the second stack location. The Count Up/Down 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. When the current value (Cxxx) is greater than or equal to the Preset Value (PV), the counter bit (Cxxx) turns on. The counter is reset 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 (Figure 7.9).

Operands: Cxxx: 0 to 255

PV: VW, T, C, IW, QW, MW, SMW, AC,
AIW, Constant, *VD, *AC, SW

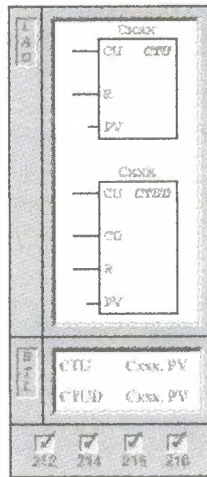


Figure 7.9

7.2.10 Understanding the High-Speed Counter Instructions

The Up Counter (CTU) counts up from the current value of that counter each time the count-up input makes the transition from off to on (Figure 7.10). The counter is reset when the reset input turns on, or when the Reset instruction is executed. The counter stops upon reaching the maximum value (32,767). The Up/Down Counter (CTUD) counts up each time the count-up input makes the transition from off to on, and counts down each time the countdown input makes the transition from off to on. The counter is reset when the reset input turns on, or when the Reset instruction is executed. Upon reaching maximum value (32,767), the next rising edge at the count-up input causes 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 countdown input causes the current count to wrap around to the maximum value (32,767). When you reset a counter using the Reset instruction, both the counter bit and the counter current value are reset. The Up and Up/Down counters have a current value that maintains the current count. They also have a preset value (PV) that is compared to the current value whenever the counter instruction is executed. When the current value is greater than or equal to the preset value, the counter bit (C-bit) turns on. Otherwise, the C-bit turns off.

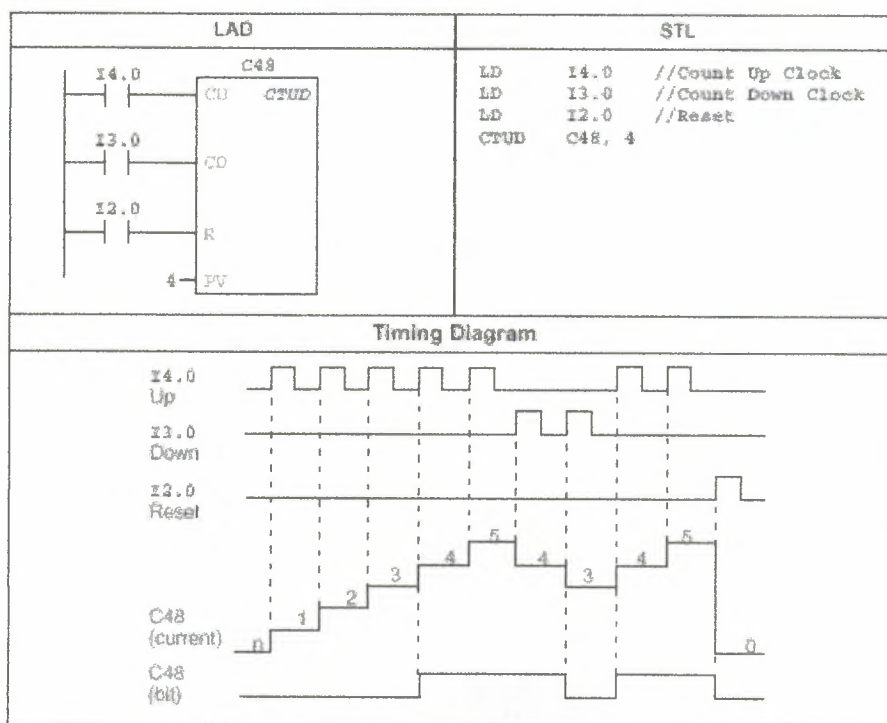


Figure 7.10 Example of Counter Instruction for LAD and STL

7.3 Addressing the Local and Expansion I/O

The local I/O provided by the CPU module provides a fixed set of I/O addresses. You can add I/O points to the CPU by connecting expansion I/O modules to the right side of the CPU, forming an I/O chain. The type of I/O and the position of the module in the chain, with respect to the preceding input or output module of the same type determine the addresses of the points of the module. For example, an output module does not affect the addresses of the points on an input module, and vice versa. Likewise, analog modules do not affect the addressing of digital modules, and vice versa.

Discrete or digital expansion modules always reserve process-image register space in increments of eight bits (one byte). If a module does not provide a physical point for each bit of each reserved byte, these unused bits cannot be assigned to subsequent modules in the I/O chain. For output modules, the unused bits in the reserved bytes can be used like internal memory bits (M bits). For input modules, the unused bits in reserved bytes are set to zero with each input update cycle, and therefore cannot be used as internal memory bits.

Analog expansion modules are always allocated in increments of two points. If a module does not provide physical I/O for each of these points, these I/O points are lost and are not available for assignment to subsequent modules in the I/O chain. Since there is no image memory provided for analog I/O, there is no way to use these unused analog I/O points. All analog I/O accesses are made immediately at the time of instruction execution.

7.3.1 Examples of Local and Expansion I/O

Figures 7-11, 7-12, and 7-13 provide examples that show how different hardware configurations affect the I/O numbering. Your program cannot use notice that some of the configurations contain gaps in the addressing that, while other I/O addresses can be used in the same manner as the internal memory (M) bits.

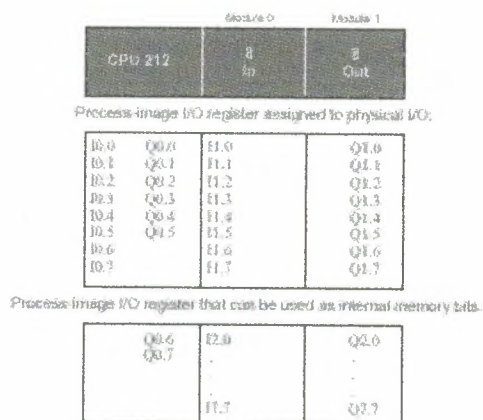


Figure 7-11 I/O Numbering Examples for a CPU 212



| CPU 214 or CPU 215 | | Module 0 | Module 1 | Module X | Module 3 | Module 4 |
|---|------|-----------------|----------|----------------|----------|----------------|
| | | 4 In / 4 Out | 8 In | 3 AI / 1 AO | 8 Out | 3 AI / 1 AO |
| Process-image I/O register assigned to physical I/O | | | | | | |
| I0.0 | Q0.0 | I2.0 | Q2.0 | I3.0 | | AIW0 AQW0 |
| I0.1 | Q0.1 | I2.1 | Q2.1 | I3.1 | | AIW2 |
| I0.2 | Q0.2 | I2.2 | Q2.2 | I3.2 | | AIW4 |
| I0.3 | Q0.3 | I2.3 | Q2.3 | I3.3 | | |
| I0.4 | Q0.4 | | | I3.4 | | |
| I0.5 | Q0.5 | | | I3.5 | | |
| I0.6 | Q0.6 | | | I3.6 | | |
| I0.7 | Q0.7 | | | I3.7 | | |
| I1.0 | Q1.0 | | | | Q3.0 | AIW8 AQW4 |
| I1.1 | Q1.1 | | | | Q3.1 | AIW10 |
| I1.2 | | | | | Q3.2 | AIW12 |
| I1.3 | | | | | Q3.3 | |
| I1.4 | | | | | Q3.4 | |
| I1.5 | | | | | Q3.5 | |
| I1.6 | | | | | Q3.6 | |
| I1.7 | | | | | Q3.7 | |
| Process-image I/O register that can be used as internal memory bits | | | | | | |
| Q1.2 | | Q2.4 | I3.0 | | Q4.0 | |
| Q1.3 | | Q2.5 | | | | |
| Q1.4 | | Q2.6 | | | | |
| Q1.5 | | Q2.7 | | | | |
| Q1.6 | | | I3.7 | | Q4.7 | |
| Q1.7 | | | | | | |
| Process-image I/O register memory that cannot be used | | | | | | |
| I1.6 | | I2.4 | | AIW6 AQW2 | | AIW14 AQW6 |
| I1.7 | | I2.5 | | | | |
| | | I2.6 | | | | |
| | | I2.7 | | | | |

Figure 7-12 I/O Numbering Examples for a CPU 214 or CPU 215

| CPU 216 | | Module 0 | Module 1 | Module 2 |
|---------|--|-----------------|-------------------|-------------------|
| | | 8 In / 8 Out | 16 In / 16 Out | 16 In / 16 Out |

Process-image I/O register assigned to physical I/O

| | | | | | |
|------|------|------|------|------|------|
| I0.0 | Q0.0 | I3.0 | Q2.0 | I6.0 | Q5.0 |
| I0.1 | Q0.1 | I3.1 | Q2.1 | I6.1 | Q5.1 |
| I0.2 | Q0.2 | I3.2 | Q2.2 | I6.2 | Q5.2 |
| I0.3 | Q0.3 | I3.3 | Q2.3 | I6.3 | Q5.3 |
| I0.4 | Q0.4 | I3.4 | Q2.4 | I6.4 | Q5.4 |
| I0.5 | Q0.5 | I3.5 | Q2.5 | I6.5 | Q5.5 |
| I0.6 | Q0.6 | I3.6 | Q2.6 | I6.6 | Q5.6 |
| I0.7 | Q0.7 | I3.7 | Q2.7 | I6.7 | Q5.7 |
| I1.0 | Q1.0 | | | I7.0 | Q6.0 |
| I1.1 | Q1.1 | | | I7.1 | Q6.1 |
| I1.2 | Q1.2 | | | I7.2 | Q6.2 |
| I1.3 | Q1.3 | | | I7.3 | Q6.3 |
| I1.4 | Q1.4 | | | I7.4 | Q6.4 |
| I1.5 | Q1.5 | | | I7.5 | Q6.5 |
| I1.6 | Q1.6 | | | I7.6 | Q6.6 |
| I1.7 | Q1.7 | | | I7.7 | Q6.7 |
| I2.0 | | | | | |
| I2.1 | | | | | |
| I2.2 | | | | | |
| I2.3 | | | | | |
| I2.4 | | | | | |
| I2.5 | | | | | |
| I2.6 | | | | | |
| I2.7 | | | | | |

Figure 7-13 I/O Numbering Examples for a CPU 216

7.4 Using the Selectable Input Filter to Provide Noise Rejection

Some S7-200 CPUs allow you to select an input filter that defines a delay time (selectable from 0.2 ms to 8.7 ms) for some or all of the local digital input points. As shown in Figure 7-14, this delay time is added to the standard response time for groups of four input points. This delay helps to filter noise on the input wiring that could cause inadvertent changes to the states of the inputs.

The input filter is part of the CPU configuration data that is downloaded and stored in the CPU memory.

Use the menu command CPU-Configure and click on the Input Filters tab to configure the delay times for the input filter.

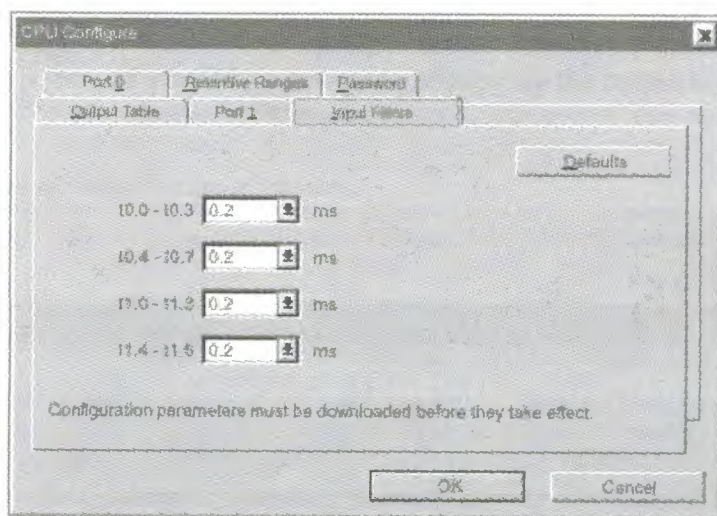


Figure 7-14 Configuring the Input Filters for Rejecting Noise

7.5 Using the Output Table to Configure the States of the Outputs

The S7-200 CPU provides the capability either to set the state of the digital output points to known values upon a transition to the STOP mode, or to leave the outputs in the state they were in prior to the transition to the STOP mode.

The output table is part of the CPU configuration data that is downloaded and stored in the CPU memory.

The configuration of output values applies only to the digital outputs. Analog output values are effectively frozen upon a transition to the STOP mode. This occurs because your

program is responsible for updating the analog outputs as required. The CPU does not update the analog inputs or outputs as a system function. The CPU maintains no internal memory image for these points.

Select the menu command CPU-Configure and click on the Output Table tab to access the output table configuration dialog. See Figure 7-15. You have two options for configuring the outputs:

- If you want to freeze the outputs in their last state, choose the Freeze Outputs box and click on “OK.”
- If you want to copy the table values to the outputs, then enter the output table values.

Click the checkbox for each output bit you want to set to On (1) after a run-to-stop transition, and click on “OK” to save your selections.

The default setting of the CPU is the mode of copying the output table values to the outputs.

The default values of the table are all zeroes.

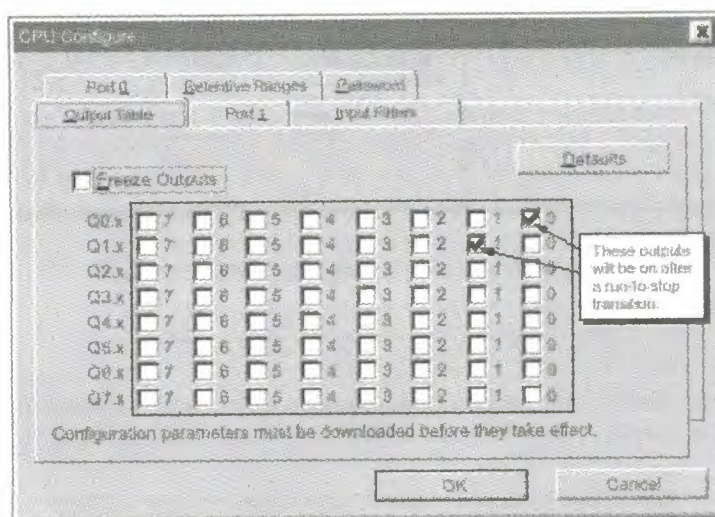


Figure 7-15 Configuring the State of the Outputs

7.6 Analog Adjustments

Your S7-200 CPU module provides one or two analog adjustments (potentiometers located under the access cover of the module). You can adjust these potentiometers to

increase or decrease values that are stored in bytes of Special Memory (SMB28 and SMB29). These read-only values can be used by the program 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.

SMB28 holds the digital value that represents the position of analog adjustment 0. SMB29 holds the digital value that represents the position of analog adjustment 1. The analog adjustment has a nominal range of 0 to 255 and a guaranteed range of 10 to 200.

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

Figure 7-16 shows an example program using the analog adjustment.

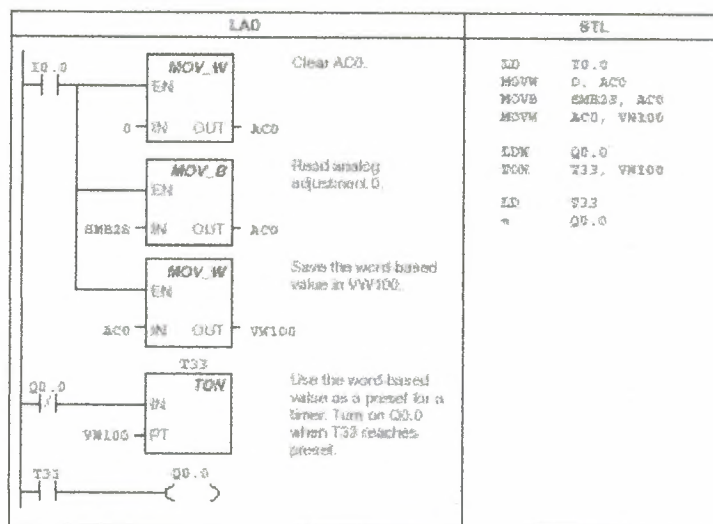


Figure 7-16 Example of Analog Adjustment

CHAPTER 8

GRADUATION PROJECT

8.1 Explanation of the Project

The counter that measures consumption is 3-phase and 165rotation/kmh. Every 300 seconds the measurement should be utilize. If it is above 10kwh, which means reaching to 11kwh's, then specific units will be out of work.

If 1kwh=165rotation/kwh then 1kwh=1815 rotation. For 300 seconds it is 151 rotations.

If counter makes more than 152 rotations in 300 seconds then the units will be out of work as stepped. It will be taken into circuit if they are below 150 rotations.

IO.1 energy control system (start)

IO.2 energy control system (stop)

IO.3 counter rotation sensor.

T32 It is the timer that defined measurement interval.

AIW 0

C0 The counter counts counter rotation.

C1 The counter that provides units to out as stepped and to real them.

Q1 The work out of 1st stepped load.

Q2 The work out of 2nd stepped load.

Mx.x The helper relays.

3.2 Program of the Project

//

PROGRAM TITLE COMMENTS

//

Press F1 for help and example program

//

NETWORK 1 //NETWORK TITLE (single line)

//

//NETWORK COMMENTS

//

LD I0.1

O M0.1

AN I0.2

= M0.1

NETWORK 2

LD M0.1

AN M0.2

TON T37, +3000

NETWORK 3

LD T37

= M0.2

NETWORK 4

LD M0.1

AN M0.2

A C0

LD M0.1

AN C0

A M0.2
LD M0.1
O I0.0
ALD
LD I0.2
CTUD C1,+6

NETWORK 5

LD C1
= M1.1

NETWORK 6

LDN C1
= M0.2

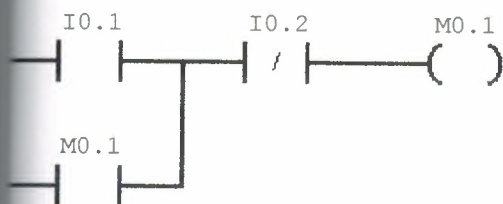
NETWORK 7

MEND

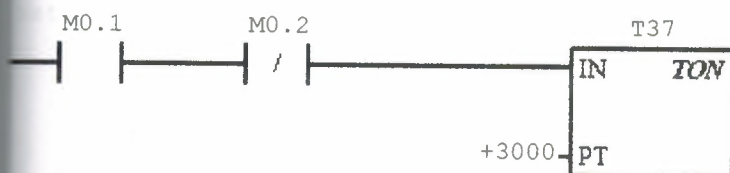
PROGRAM TITLE COMMENTS

Press F1 for help and example program

Network 1 NETWORK TITLE (single line)



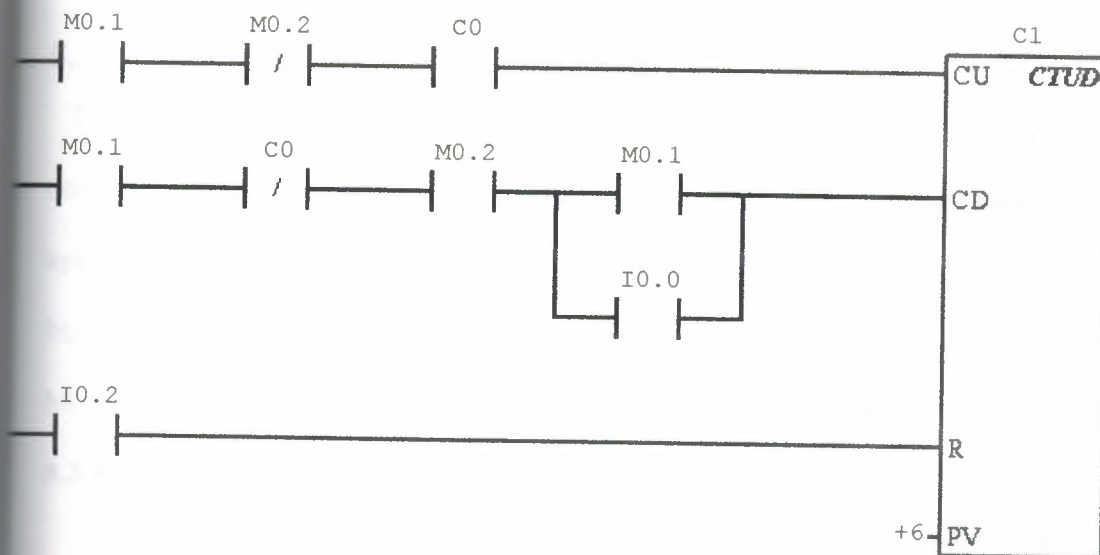
Network 2



Network 3



Network 4



Network 5



Network 6



Network 7

(END)

8.3 PLC Programmable Logic Controller

PLC's are known as industrial computers. They are designed to replace the conventional control mechanism. They are widely used in industrial automation. They have various application areas besides industry. They operate in the range of $0-55^{\circ}\text{C}$ and 0-35% of humidity. They have noise immunity. We have two types of PLC, compact PLC and Modular PLC.

8.3.1 Compact PLC's

All the parts are mounted in a special case in compact form (inputs, outputs, memory, processor power, supply etc.). Usually they have low capacity, but they are very cheap devices.

8.3.2 Modular PLC's

Mounting separate modules called RACKS together forms them. This three units are, input unit, process unit and output unit.

8.3.3 Input Unit

It converts the electrical signal (coming from the system that is a going to be controlled) in the logic levels. The digital or analog signals coming from the sensors like pressure, level, heat, optical is send to processed by the input unit.

The digital input signals are usually 24V dc or depending upon application can be 48V dc 110V dc etc.

Analog signals on the other hand can be 0-10V, -5V...0...+5V, -10V...0...+10V etc. The parasitic signals coming from the inputs are first filtered by an RC filter in this unit and then pass through optocouplers that produce galvanized isolation.

8.3.4 Output Units

Are suitable manufactured to successfully control the activators in the system?

Digital output signals control contactor relays NPN or PNP transistors or Triacs etc. PLC's output cannot supply large currents so by digital output relays and their contactor groups name contactor or winding are operates. In this way units like motors, hydraulic valves and heaters can be operated.

Parallel RC's suppresses the relay outputs of PLC's in order to prevent arcs. An optocoupler is present after output memory in order to prevent internal interference.

Analog outputs like the analog inputs are passed through DAC this time to convert the numerical values to standard output signals.

Special input output models, adjustable counter, heat sensing module and step motor output model.

8.3.5 Programming Technique

- 1) Statement list or instruction list programming
- 2) Ladder Programming

CHAPTER 9

PROCESSING UNIT

It's composed of the sub unit given below.

9.1 System Memory

Memory unit that the PLC's operating system is present. It has 3 main functions and it is composed of PROM or EPROM.

- a) To organize the relation between PLC and programming unit do the diagnostics test, give message when an error is present due to user and PLC.
- b) To convert the user program to a way that it can process by the help of a compiler in it.
- c) To function block and system modules formed by the software in it's library and to serve it to user program when it is necessary.

9.2 CPU

It is also given the name processing unit it processor all the input signals according to the user programs instructions and direct the related output signals to corresponding outputs. This process is controlled by a microprocessor some times instead of microprocessor a micro controller is used. The differences are that processor memory and I/O interfaces are all in one unit a micro controller as a memory ROM and RAM is used. Data for operating system and PLC that cannot be changed are kept in ROM and user program and I/O data are kept.

9.3 Program Memory

It is also defined as user memory it is the memory where the user program is kept. It's capacity is variable according to the instruction number.

9.4 Data Bus

It used to transfer data between the units.

9.5 Image Register

They are the register where the input and output signals are kept for one cycle. The input signals read at the beginning of the cycle are kept that at input image register, fixed unit the

beginner next cycle. The output signals obtained at the end of the cycle are kept at the image register until the end of next cycle.

9.6 PLC Operating System

In all PLC operating system similar operating system programs are used.

These programs are in ROM and they are loaded into the system while manufacturing. In general a PLC operating system does the following.

- Operates the user program
- Event and time dependent service program are operated by operating system.
- Organize the communication of PLC and control the operation of the system.

9.7 User Program Operating

A user program loaded to program memory of PLC starting from the first instruction until the last instruction is executed step by step.

If there is a jump or branching in the program the instruction until the jump address are not executed when the last instruction. This operation is like an infinite loop.

The time taken by the PLC to turn back to the same instruction is called scanning period. The scanning period of PLC is depending upon I/O number, programs length and operating frequency of the CPU.

9.7.1 When the PLC is in RUN mode

- 1) The value at input unit are transferred to display memory and kept there constant until next cycle.
- 2) According to the user programs type the instructions are executed step by step. While operating the system program provides calculated intermediate values of input signal at the display memories. They cannot change during the cycle.
- 3) The values calculated as a result of executing the user program are transferred to output display memory and are send to output unit. After transferring the data output unit the program returns to first step. The data at output display register and output unit does not change until next cycle.
- 4) In some PLC's the output data is directly sent to output unit (DSP Direct Processing System, Hitachi H-200)
- 5) In some PLC's you can reach real time input and output directly by some

instruction (Static S7)

9.8 Accessing Data Memory

Data memory for S7-200 consists of five areas. I (input), Q (output), M (internal memory bit), SM (special memory bit) and V (variable memory bit)

Memory areas can be accessed either as a bit, a byte, a word or a double word.

9.8.1 Bit Access

The access a bit specifies the address of the bit, which consists of an area identifier and the byte bit number. Zero is the first address for all data areas.

9.8.2 Byte word double word access

To access a byte, word specifies the address, which consist of an area identifier a letter signifying data size and the address number.

9.9 Advantage PLC

- Physical dimension: PLC is the minimum place-occupying device among all control devices.
- Cost you can find optimum cost PLC for any control system.
- Time saving while the following factory working you can program the PLC.
- Simplicity the manufacturing: it is very easy to mount the PLC on the panels than conversional system.
- Easy to find faults: PLC's need almost no maintenance they are give information when they have a fault.
- Easy in operation: You can change the program of PLC easily when it is necessary without changing the circuiting of the panel.

CONCLUSION

In this project, I learned how to write PLC Program with Simatic Program. The aim of my program is to control electric counters with PLC device and to work out the units that are connected to the counter when they reach the value that we determine.

PLC device control the red point on electric counter with sensor.

PLC device make the unit workout that are connected to the counter when the units reach the value that we determine. Counter (CTUD) is between the electric counter and connect units. It control the power of connects units.

Usually, this project is used industrial factory, because factory owner control the which units are take electric more than other.

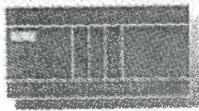
By using PLC device, we can control our electric usage in more economical way.

REFERENCES

- [1] "[http:// siemens.com](http://siemens.com) " Retrieved April 15,2001
- [2] "*S7-200 Programmable Controller System Manual*" Copyright Siemens AG 1998, April 17,2001
- [3] Özerdem, Özgür "PLC lecturer notes", April 26, 2001
- [4] "STEP7 –Micro/WIN 16", Siemens energy and automation manual 1998, June 15,2001
- [5] "<http://plc.com>" , June 5,2001



APENDIX

PRODUCT IMAGE



CPU 212

ID: SG01107 created 12/8/98

| | Format | File Size | Language(s) | Vers. |
|-----------|---|-----------|-------------|-------|
| Source: |  BildDB - Powerpoint (RGB or b/w; p) (.tif) | 142 KB | - | 1.0 |
| Variants: |  BildDB - Vector graphic (not specific) (.FH8) | 23 KB | - | 1.0 |

Properties

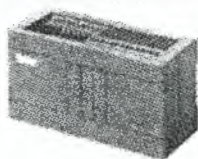
| | |
|----------------|---|
| Title German | CPU 212 |
| Title English | CPU 212 |
| Search term(s) | Grafiken; Graphiken; graphics; Symbole; symbols; CPU 212; S7-200; Micro-SPS; Automatisierungsgerät; programmable logic controller; PLC; Micro-PLC; SIMATIC Controller |
| Picture type | Product image |

Product related properties

| | |
|---------------------|---------|
| Product family | SIMATIC |
| Branch | |
| MLFB (e.g. 6ES7 4*) | 6ES7 2 |

Image Attributes

| | |
|--------|----------------|
| Format | Vector graphic |
|--------|----------------|



CPU 212

ID: ST70001 created 5/12/98

| | Format | File Size | Language(s) | Vers. |
|---------|---|-----------|-------------|-------|
| Source: |  BildDB - b/w (b) (.TIF) | 280 KB | - | 1.0 |

Variants:  BildDB - Powerpoint (RGB or b/w: 46 KB - 1.0
p) (.TIF)

Properties

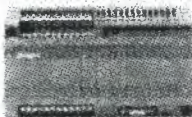
Title German CPU 212
Title English CPU 212
Search term(s) CPU 212; Zentralbaugruppe; central processing unit;
SIMATIC S7-200; SIMATIC Controller
Picture type Product image

Product related properties

Product family SIMATIC
Branch
MLFB (e.g. 6ES7 4*)

Image Attributes

Format Photo (masked)



CPU 212

ID: ST70003 created 5/12/98

| | Format | File Size | Language(s) | Vers. |
|------------------------------------|--|-----------|-------------|-------|
| Source: <input type="checkbox"/> | BildDB - b/w (b) (.TIF) | 286 KB | - | 1.0 |
| Variants: <input type="checkbox"/> | BildDB - Powerpoint (RGB or b/w: p) (.TIF) | 53 KB | - | 1.0 |

Properties

Title German CPU 212
Title English CPU 212
Search term(s) CPU 212; Zentralbaugruppe; central processing unit;
SIMATIC S7-200; SIMATIC Controller
Picture type Product image

Product related properties

Product family SIMATIC
Branch

MLFB (e.g. 6ES7 4*)

Image Attributes

Format Photo (masked)



CPU 212

ID: ST70300 created 3/30/98

| | Format | File Size | Language(s) | Vers. |
|-----------|---|-----------|-------------|-------|
| Source: | <input type="checkbox"/> BildDB - Colour (RGB; v) (.TIF) | 1,583 KB | - | 1.0 |
| Variants: | <input type="checkbox"/> BildDB - b/w (b) (.TIF) | 560 KB | - | 1.0 |
| | <input type="checkbox"/> BildDB - Powerpoint (RGB or b/w; p) (.tif) | 222 KB | - | 1.0 |

Properties

Title German CPU 212
Title English CPU 212
Search term(s) S7-200; CPU 212; SIMATIC S7-200; Zentralbaugruppe; Central processing unit; SIMATIC Controller
Picture type Product image

Product related properties

Product family SIMATIC
Branch
MLFB (e.g. 6ES7 4*)

Image Attributes

Format Photo (masked)

TECHNICAL DATA

| CPU 212 | |
|--|---|
| Program memory | 1 Kbyte /typ. 185 statements on integrated EEPROM (fail-safe) |
| Data memory | 512 words |
| Memory cartridge (optional) | - |
| Program backup | - |
| Data backup | Maintenance-free |
| | <ul style="list-style-type: none"> • 200 bytes (DB 1), stored on built-in EEPROM |
| | <ul style="list-style-type: none"> • data, retentive memory bits, etc. backed up by super capacitor |
| Backup time typ. | 50 hr (minimum 8 hr at 40 °C) |
| Charging time for super capacitor typ. | 20 min. (to 60% capacity) |
| Programming language | STL and LAD |
| Program organization | One organization block (subroutines contained in it are supported) |
| Program execution | <ul style="list-style-type: none"> • free cycle (OB 1) <p>interrupt-controlled</p> <p>time-controlled (85 to 255 ms)</p> |
| Subroutine levels | 8 |

| | |
|------------------------------------|--|
| User program protection | 3-level password protection |
| Operation set | |
| • Basic operations | Binary logic operations, result assignments, save, count, load, transfer, compare, shift, rotate, form complement, call subroutines |
| • User-friendly functions | Pulse width modulation, pulse train instructions, jump instructions, loop instructions, code conversions, math functions, (addition, subtraction, multiplication, division, square root) |
| Execution times for bit operations | 1.2 μ s |
| Cycle time monitoring | 300 ms (retriggerable) |
| Bit memories | 128 |
| • of these retentive | 0 to 127, programmable |
| Counter | 64 |
| • of these retentive | 0 to 63, programmable |
| Timers | 64 |
| • of these retentive | 32, selectable |
| • Range | 2 timers, 1 ms to 30 s 8 timers, 10 ms to 5 min. 54 timers, 100 ms to 54 min. |
| Integrated high-speed functions | |
| • Interrupt inputs | 1 (on positive and/or negative input signal edge, programmable interrupt response) |

| | |
|-------------------------------------|--|
| • Counter | 1 up or down counter; counting rate up to 2 kHz; 32 bits (incl. sign); interrupt capability (incl. calling of a subroutine with random contents) on reaching a setpoint |
| • Pulse outputs | -- |
| Interfaces | RS 485 communication interface; either: |
| | <ul style="list-style-type: none"> • PPI mode for programming and connecting programming devices or PCs (via PC/PPI cable), TD 200, OP (9.6 and 19.2 kbit/s) |
| | <ul style="list-style-type: none"> • User-programmable interface mode with interrupt capability for serial data exchange with devices from other vendors (0.3 to 19.2 kbit/s) (CPU 212, e.g. with ASCII protocol; PC/PPI cable can be used as a RS 232/RS 485 converter (from 0.6 kbit/s) |
| Backplane bus: | |
| | <ul style="list-style-type: none"> • Connection of expansion modules (EM)1). Only EMs from the S7-21x series can be used. |
| Connectable programming devices/PCs | PG 720P, PG 740, PG 760 and PCs (via PC/PPI cable) |
| Onboard I/Os | |
| • Digital inputs | 8; incl. 1 channel for use as a hardware interrupt or for high-speed functions |
| • Digital outputs | 6 |
| • Analog potentiometer | 1 analog potentiometer; resolution 1/200 |
| Connectable inputs/outputs | |

| | | | | | | |
|-------------------------------------|---|----------------------------|-----------------|-----------------|-----------------|-----------------|
| • Digital inputs/outputs | max. 40 input and 38 outputs (incl. the integr. input/output) | | | | | |
| • Analog inputs/outputs | 6 inputs and 4 outputs | | | | | |
| • AS-Interface inputs/outputs, max. | 248 | | | | | |
| Expansion, max. | 2 expansion modules from the S7-21x series ¹⁾ (digital and analog) | | | | | |
| Degree of protection | IP 20 in accordance with IEC 529 | | | | | |
| Ambient temperature | | | | | | |
| • With horizontal installation | 0 to 55 °C | | | | | |
| • When mounted vertically | 0 to 45 °C | | | | | |
| Relative humidity | 5 to 95 % (RH severity level 2 in accordance with IEC 1131-2) | | | | | |
| Atmospheric pressure | 860 to 1080 hPa | | | | | |
| Other ambient conditions | See "S7-200 Programmable Controller System Manual" | | | | | |
| Power supply: | 24 V DC | 120 to 230 V AC/ 24V AC | 120 to 230 V AC | 120 to 230 V AC | 120 to 230 V AC | 120 to 230 V AC |

| | | | | | | |
|----------|------------|-----------------------|-----------------------|-----------------------|-----------------------|-------------|
| Inputs: | 24 V DC | 24 V DC (sink), | 24 V DC (src.), | 24V AC | 120V AC | 120 V AC |
| Outputs: | 24 V DC | Relay | Relay | 120 to 230 V AC | 120 to 230 V AC | Relay |

| | | | | | | |
|--------------------------------------|--------------------|--------------------------------------|--------------------------------------|--------------------------------------|--------------------------------------|--------------------------------------|
| Rated value | 24 V DC | 120 to 230 V AC | 120 to 230 V AC | 120 to 230 V AC | 120 to 230 V AC | 120 to 230 V AC |
| • Permitted range | 20.4 bis 28.8 V | 85 to 264V AC (47 to 63 Hz) | 85 to 264V AC (47 to 63 Hz) | 85 to 264V AC (47 to 63 Hz) | 85 to 264V AC (47 to 63 Hz) | 85 to 264V AC (47 to 63 Hz) |
| Input current, typ. | 60 mA | 4 VA | 4 VA | 4 VA | 4 VA | 4 VA |
| Inrush surge, max. | 10 A | 20 A | 20 A | 20 A | 20 A | 20 A |
| Power consumption max. | 5 W | 6 W / 7 W | 6 W | 7 W | 7 W | 7 W |
| Output voltage for sensors | | | | | | |
| • Rated value | L+ (24 V DC) | 24 V DC | 24 V DC | 24 V DC | 24 V DC | 24 V DC |
| • Permitted range | L+ - max. 4 V | 20.4 bis 28.8 V | 20.4 bis 28.8 V | 20.4 bis 28.8 V | 20.4 bis 28.8 V | 20.4 bis 28.8 V |
| Output current for sensors (24 V DC) | | | | | | |
| • Rated value | 180 mA | 180 mA | 180 mA | 180 mA | 180 mA | 180 mA |

| | | | | | | |
|----------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|
| • Short-circuit protection | Electronic, non-latching | Electronic, non-latching | Electronic, non-latching | Electronic, non-latching | Electronic, non-latching | Electronic, non-latching |
|----------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|

| | | | | | | |
|---|-------------|-------------------------|--------------|-------------|-------------|-------------|
| Output current for expansion modules ¹⁾ (5 V DC) | 340 mA | 340 mA | 340 mA | 280 mA | 280 mA | 340 mA |
| Integrated inputs | 8 | 8 | 8 | 8 | 8 | 8 |
| Input voltage | | | | | | |
| • Rated value | 24 V DC | 24 V DC/AC | 24 V DC | 24 V AC | 120 V AC | 120 V AC |
| • For "1" signal | 15 to 30 V | 15 to 35 V / 15 to 30 V | -15 to -30 V | 15 to 30 V | 79 to 135 V | 15 to 30 V |
| • For "0" signal | 0 to 5 V | 0 to 5 V | 0 to 5 V | 0 to 5 V | 0 to 5 V | 0 to 5 V |
| Isolation | Optocoupler | Optocoupler | Optocoupler | Optocoupler | Optocoupler | Optocoupler |
| • In groups of | 4 | 4/8 | 4 | 8 | 8 | 8 |
| Input current | | | | | | |
| • For "1" signal | 7 mA | 7 mA | 7 mA | 7 mA | 7 mA | 7 mA |

| Input delay (at rated value of the input voltage) | | | | | | |
|---|--------------------------|--------------------------|--------|-------------------------|-------|-------|
| • For standard inputs max. | (I0.0 to I0.7) 0.3 ms | (I0.0 to I0.7) 0.3 ms | 0.3 ms | (I0.0 to I0.7) 15 ms | 15 ms | 15 ms |
| • For interrupt inputstyp./max. | (I0.0) - | (I0.0) - | | (I0.0) - | | |
| • For high-speed counter 0 typ./max. | (I0.0) 30/70 ms | (I0.0) 30/70 ms | | (I0.) 15 ms | | |
| • For high speed counters 1, 2, typ./max. | - | - | | - | | |
| Connection of 2-wire BERO | | | | | | |
| • Permissible quiescent current, max. | 1 mA / - | 1 mA / - | 1 mA | -- | -- | -- |
| Cable lengths | | | | | | |
| • Unshielded (not for high-speed signals)) | 300 m | 300 m | 300 m | 300 m | 300 m | 300 |

| | | | | | | |
|--|-------|-------|-------|-------|-------|-------|
| Shielded standard input | 500 m | 500 m | 500 m | 500 m | 500 m | 500 m |
| • Shielded, interrupt inputs, high-speed counter | 50 m | 50 m | 50 m | 50 m | 50 m | 50 m |

| On-board outputs | 6 (transistor) | 6 (relays) | 6 (relays) | 6 (triacs) | 6 (triacs) | 6 (relays) |
|--------------------------|-------------------|------------------------------|------------------------------|----------------|----------------|------------------------------|
| Rated load voltage L+/L1 | 24 V DC | 24 V DC/ 24 to 230 V AC | 24 V DC/ 24 to 230 V AC | 24 to 230 V AC | 24 to 230 V AC | 24 V DC/ 24 to 230 V AC |
| * Permitted range | 20.4 to 28.8 V DC | 5 to 30 V DC/ 20 to 250 V AC | 5 to 30 V DC/ 20 to 250 V AC | 24 to 264 V AC | 24 to 264 V AC | 5 to 30 V DC/ 20 to 250 V AC |
| Output voltage | | | | | | |
| * With signal "1", min. | L+ - 1.8 V | L+/L1 | E+/L1 | L1 - 1.5 V | L1 - 1.5 V | L1 - 1.5 |
| Isolation | Optocoupler | Relay | Relay | Optocoupler | Optocoupler | Relay |
| * In groups of | 6 | 3 | 3 | 3 | 3 | 3 |
| Maximum output current | | | | | | |
| * For "1" signal | | | | | | |
| - rated value at 40 °C | 0.75 A | 2 A | 2 A | 1.2 A | 1.2 A | 2 A |
| - rated value at 55 °C | 0.5 A | 2 A | 2 A | 1.0 A | 1.0 A | 2 A |
| - min. current | - | - | - | 10 mA | 10 mA | - |

| | | | | | | |
|--|--------|--------|--------|----------------------------|----------------------------|--------|
| • For "0" signal | 0,1 mA | 0 mA | 0,1 mA | 2.0 mA (at 240 V) | 2.0 mA (at 240 V) | 0 mA |
| Sum of all output currents per common | | | | | | |
| • At 40 °C, max. | 2.25 A | 12 A | 12 A | 3.5 A | 3.5 A | 12 A |
| • At 55 °C, max. (horizontal installation) | 1.75 A | 12 A | 12 A | 2.5 A | 2.5 A | 12 A |
| Sum of the currents from 2 adjacent outputs | | | | | | |
| • At 40 °C, max. | 1.0 A | 4.0 A | 4.0 A | 1.5 A | 1.5 A | 4.0 A |
| • At 55 °C, max. (horizontal installation) | 0.75 A | 4.0 A | 4.0 A | 1.25 A | 1.25 A | 4.0 A |
| Switching frequency of outputs | | | | | | |
| • For resistive load | 4 kHz | 5 Hz | 5 Hz | 2 x frequency of sup.volt. | 2 x frequency of sup.volt. | 5 Hz |
| • For inductive load | 0.5 Hz | 0.5 Hz | 0.5 Hz | 0.5 Hz | 0.5 Hz | 0.5 Hz |
| • For a lamp load | 1 Hz | 1 Hz | 1 Hz | 1 Hz | 1 Hz | 1 Hz |
| Switching capacity of outputs (1 output to 40 °C) | | | | | | |
| • For resistive load | 0.75 A | 2 A | 2 A | 1.2 A | 1.2 A | 2 A |